INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES MONTERREY

CAMPUS PUEBLA

PROYECTO FINAL GRÁFICAS COMPUTACIONALES

PARQUE DE DIVERSIONES CON AUTOMOVIL Y CONSTRUCCIÓN

SEMESTRE ENERO – MAYO 2015

PROFR. DANIEL PÉREZ ROJAS

KAREN STEPHANIE ABARCA GARCÍA A01323627

FERNANDO AREY DURÁN A00397411

13 DE MAYO DE 2015

ÍNDICE

INTRODUCCION

El proyecto final de la materia de Gráficas Computacionales consistió en realizar una escena 3D usando *WebGL* y *three.js*.

La escena consiste en un parque de diversiones con tres atracciones:

- **Carrusel** con 16 caballos que giran en sentido contrario a las manecillas del reloj subiendo y bajando.
- **Rueda de la fortuna** con 20 canastas que giran obedeciendo a la gravedad.
- **Tazas locas** con 16 tazas girando sobre su eje colocadas sobre una base que también está girando.

Se agregó, también, un auto con partes móviles controlado por el usuario y una construcción interesante, en este caso el castillo de *Mario Bros*.

IMPLEMENTACION

PARQUE DE DIVERSIONES

Carrusel

Para crear el carrusel se utilizaron las siguientes 15 geometrías (4 para formar el techo, 2 para la parte central, 7 para los caballos y 2 para la base):

- geometriaEsfera (Una esfera con un radio de 1)
- geometriaCilindroPunta (Un cilindro con 0.2 unidades de radio y 4 de alto)
- geometriaCono (Cilindro con 23 unidades de radio inferior, 0 de radio superior y 4 de alto)
- geometriaCilindroTecho (Cilindro con 23 unidades de radio y 1 de alto)
- geometriaCilindroCentro (Cilindro con 1.5 unidades de radio y 18 de alto)
- geometriaCilindroCaballos (Cilindro con 0.2 unidades de radio y 18 de alto que representaban los tubos donde se ubicaron los caballos)
- geometriaTorsoCaballo (Cubo con 3 unidades de ancho, 1.5 de largo y 1.5 de profundidad)
- geometriaCuelloCaballo (Cubo con 2 unidades de ancho, 1 de largo y 1.5 de profundidad que se giró 60° y se colocó de forma que simulará el cuello de un caballo)
- geometriaCabezaCaballo (Cubo con 1.5 unidades de ancho, 1 de largo y 1.5 de profundidad)
- geometriaOrejaCaballo (Cilindro con 0.2 unidades de radio inferior, 0 de radio superior y 0.9 de alto)
- geometriaPataCaballo (Cubo con 0.5 unidades de ancho, 2 de largo y 0.5 de profundidad)

- geometriaFinColaCaballo (Cubo con 1 unidad de ancho, 0.5 de largo y 0.5 de profundidad)
- geometriaCilindroEscalon1 (Cilindro con 23.6 unidades de radio y 0.5 de alto que representó el escalón superior del carrusel)
- geometriaCilindroEscalon2 (Cilindro con 23.8 unidades de radio y 0.2 de alto que representó el escalón inferior del carrusel)

Tamaño de Objetos

Los tamaños de estas geometrías fueron calculados en base a las necesidades de espacio y proporciones para lograr que todos los elementos se vieran agradables y evitar aglomeraciones de objetos o imágenes desproporcionadas.

Objetos

Estas geometrías se utilizaron para crear 16 objetos del carrusel. El techo fue creado con 4 objetos (una esfera y un cilindro para la punta y un cono con un cilindro más ancho para la forma cónica del techo). En el centro se utilizaron 16 cilindros angostos para los tubos de los caballos y un cilindro más ancho para el tubo central del carrusel. Finalmente, para la base se utilizaron 3 cilindros de distintos tamaños.

Se utilizaron 5 texturas diferentes:

- material: color blanco utilizado en los tubos de los caballos así como los cilindros y la esfera del techo.
- materialRojo: color rojo utilizado en el cilindro central del carrusel y en uno de los cilindros de la base
- materialDorado: color ocre que fue usado en 2 de los 3 cilindros de la base.

- materialCafe: utilizado para los caballos
- techoCarrusel: imagen que da la ilusión de un espiral girando.

Los objetos correspondientes a las patas de los caballos y los tubos (sobre los cuales se ubicaban) fueron clonados 4 y 16 veces respectivamente para evitar la declaración de objetos individuales.

Todos los elementos que integraron al caballo (torso, cuello, orejas, patas y colas se introdujeron en un único grupo).

Posiciones

El carrusel fue creado desde la punta hacia abajo por lo que el cilindro de la punta se ubicó en la posición 0 y, el resto de objetos se colocaron en base a este cilindro.

En la colocación de los tubos del centro se utilizaron dos ciclos *for*, el primero para ubicar los 8 tubos exteriores y el segundo para ubicar los 8 tubos que irían en el círculo más interno del carrusel.

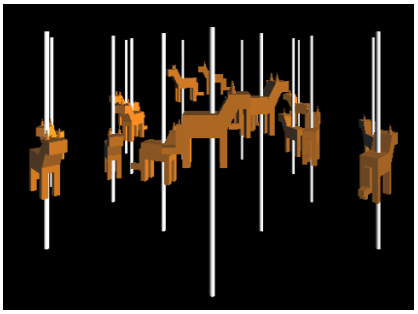
Durante estos ciclos se calculó la posición *x,z* de los tubos. Para ello se utilizaron las funciones de seno y coseno con un ángulo que se incrementaba 45°por cada iteración (iniciando en 0° y terminando en 315°). Se emplearon diferentes radios para lograr la diferencia en las distancias (20 unidades para los tubos externos y 8 para los internos).

```
//Tubos externos caballos
var angulo = 0;
for (var i = 0; i < 8; i++) {
      tubosCaballos[i].position.y = -13.5;
      tubosCaballos[i].position.x = Math.sin(angulo) * 20;
      tubosCaballos[i].position.z = Math.cos(angulo) * 20;

      angulo += (Math.PI * 2) / 8;
};

//Tubos internos caballos
var angulo = 0;
for (var i = 8; i < 16; i++) {
      tubosCaballos[i].position.y = -13.5;
      tubosCaballos[i].position.x = Math.sin(angulo + 0.392699082) * 12;
      tubosCaballos[i].position.z = Math.cos(angulo + 0.392699082) * 12;

      angulo += (Math.PI * 2) / 8;
};
```

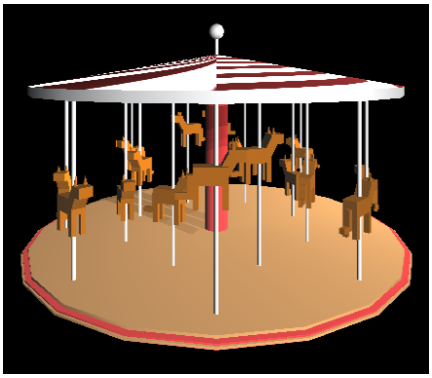
Los caballos se armaron comenzando con el torso y ubicando el resto de los objetos (cuello, cabeza, orejas, patas y cola) de tal forma que crearán un "único" objeto con la forma de un caballo.

Para ubicar los 16 caballos se utilizó el mismo principio que con los tubos (2 ciclos *for* con ley de senos y cosenos); se agregaron 22.5° grados al ángulo actual para hacer que los caballos tuvieran la rotación correcta. En este caso, también, se utilizó la función del seno para calcular la posición en *y* de los caballos; usando una altura máxima de -11 y restándole a esta el resultado de multiplicar el seno de su ángulo por las 6 unidades que se usaron para la diferencia de alturas. De esta forma cada caballo se ubicó en una altura distinta (en base a su posición).

```
//Clonando el grupo caballo y estableciendo su posicion
var angulo = 0;
var grupoCaballos = [];
for (var i = 0; i < 8; i++) {
      grupoCaballos[i] = grupoCaballo.clone();

      var x = Math.sin(angulo) * 20;
      var z = Math.cos(angulo) * 20;

      grupoCaballos[i].applyMatrix( new THREE.Matrix4().makeTranslation( x , - 11 - Math.abs(Math.sin(angulo) * 6), z));
      grupoCaballos[i].rotation.y = angulo;

      angulo += Math.PI * 2 / 8;

};

var angulo = 0;
for (var i = 8; i < 16; i++) {
      grupoCaballos[i] = grupoCaballo.clone();

      var x = Math.sin(angulo + 0.392699082) * 12;
      var z = Math.cos(angulo + 0.392699082) * 12;

      grupoCaballos[i].applyMatrix( new THREE.Matrix4().makeTranslation( x, - 11 - Math.abs(Math.sin(angulo) * 6), z));
      grupoCaballos[i].rotation.y = angulo + 0.392699082;

      angulo += (Math.PI * 2 / 8);

};
```



Finalmente, todos estos objetos se añadieron a un mismo grupo carrusel, que fue el que se agregó a la escena.



Funcionamiento

Para hacer girar el carrusel en dirección contraria a las manecillas del reloj en cada ciclo de la función *render* se

incrementó el ángulo de rotación en y de todo el carrusel en 0.01 unidades.

Para el movimiento de subir y bajar de los caballos se recalculó en cada ciclo su posición en *y* de la misma forma en se calculó su altura inicial (usando una altura máxima de -11 y restándole la multiplicación del seno de su ángulo actual por 6).

```
//ROTACION DEL CARRUSEL
grupoCarrusel.rotation.y += 0.01;

//Movimiento de los caballos
var angulo = [];
var rotacion = 0;
for(i = 0; i < 16; i++){

      var angulo = Math.sin(grupoCaballos[i].rotation.y + grupoCarrusel.rotation.y);
      grupoCaballos[i].position.y =  - 11 - Math.abs(angulo * 6);
}
```

Rueda de la fortuna

En la rueda de la fortuna se utilizaron 14 geometrías (3 para la base, 2 en la estructura de la rueda, 2 con el soporte y 7 para las canastas):

- geometriaBase1 (Un cubo de 70 unidades de ancho, 1 de alto y 54 de profundidad que representó el primer escalón de la base)
- geometriaBase2 (Un cubo de 56 unidades de ancho, 4 de alto y 40 de profundidad que se usó como el escalón superior en la rueda)
- geometriaRueda (Un toroide de 40 unidades de radio y 0.2 unidades como diámetro del tubo que formaron las 2 estructuras de la rueda)
- geometriaRuedaUnion (Cilindros de 0.1 unidades de radio y 80 de largo que formaron las divisiones de las ruedas)
- geometriaUnionRuedas (Cilindro con 0.1 unidades de radio y 5.6 de largo que se utilizaron para unir las dos ruedas y soportar las canastas)
- geometriaSoporte (Cilindro con 2 unidades de radio inferior, 0.5 de radio superior y 20 de alto que dieron forma a los 4 soportes que sostienen la rueda)

- geometriaUnionSoporte (La unión de los 2 lados de los soportes se realizó con un cilindro con 1 unidad de radio y 10 de largo)
- geometriaCanasta (Cilindro de 2.7 unidades de radio superior, 1.5 de radio inferior y 3 de alto; no se dibujaron las tapas de este cilindro para hacerlo abierto, esto se logra poniendo el parámetro *openEnded* como true)
- geometriaCanastaI (El cilindro interno de las canastas era 0.35 unidades más angosto que el externo pero con la misma altura)
- geometriaTechoCanasta (Cilindro con 1 unidad de radio superior, 2.7 de radio inferior y 0.5 de alto)
- geometriaPisoCanasta (Cilindro con 1.5 unidades de radio y 0.1 de alto, este cilindro fue utilizado para simular el piso de la canasta ya que, los cilindros estaban abiertos)
- geometriaUnionTecho (Cilindro con 0.1 unidades de radio y 2.55 de alto que fue usado para unir la canasta con el techo de esta)
- geometriaUnionCanastas (Un anillo de 2.7 unidades de radio interno y 2.35 de radio interno que se utilizó para unir los dos cilindros que conformaban la canasta, a manera de tapa)
- geometriaUnionCanastasRueda (Finalmente un cilindro de 0.1 unidades de radio y 0.95 unidades de alto fue el encargado de unir las canastas con su soporte en la rueda)

Tamaño de los objetos

Las geometrías y sus tamaños se calcularon en base al tamaño definido para el carrusel de forma que la rueda de la fortuna quedará más alta que este pero sin verse muy grande y totalmente desproporcionada.

El tamaño de las canastas se calculó en base a la distancia que había entre la rueda y su base dejando un máximo de 6 unidades para colocar la canasta inferior sin que topará con la base.

Objetos

Se crearon 16 objetos con las 14 geometrías: La base se formó con 2 cubos; el soporte con 5 cilindros; la estructura de la rueda con un toroide, un cilindro largo clonado 10 veces para los "ejes" y un cilindro más corto clonado 10 veces para la unión entre las dos ruedas; las canastas se crearon con 6 cilindros diferentes y un anillo.

Definimos y utilizamos 5 materiales para darle color a la rueda:

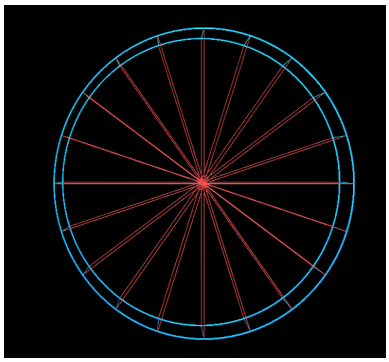
- materialBlanco: color blanco usado para los soportes y la base de la rueda de la fortuna.
- materialAzul: color azul claro que se aplicó en las dos estructuras de la rueda y las uniones entre ellas.
- materialRojo: color rojo que se usó para los ejes de las ruedas.
- materialDorado: color oro que utilizaron las canastas

Los ejes de la rueda, las canastas y como las uniones entre ruedas estaban formados por objetos individuales que fueron clonados 10 y 20 veces (ejes, canastas y uniones respectivamente) para evitar declarar 40 objetos individuales.

Posiciones

Lo primero que se hizo fue definir la posición del centro del grupo rueda para establecer la rotación en base a este punto. El resto de objetos fueron colocados partiendo de dicho centro, salvo las canastas que se dibujaron a partir del punto que las une con la rueda.

Después de esto se estableció una separación entre ruedas de 5.6 centímetros y se dibujaron los ejes y las uniones tomando en cuenta esta distancia.



Los ejes se ubicaron con la ayuda de 2 ciclos (uno para los frontales con 2.8 unidades en *z* y otro para los del fondo con -2.8 unidades). En este ciclo se incrementaba el ángulo de rotación en 36° grados dejando siempre el mismo centro.

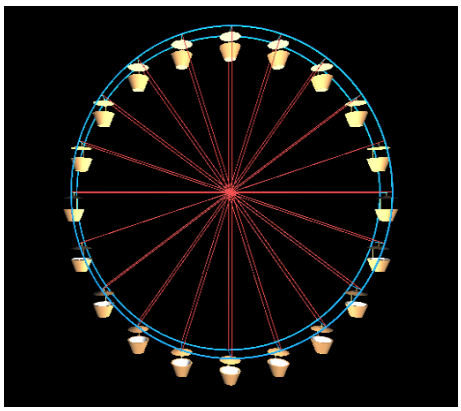
```
var i;
var angulo = 0;
for(var i = 0; i < 10; i++){
      uniones[i].position.z = 2.8;
      uniones[i].rotation.z = angulo;
      angulo += Math.PI/10;
}
angulo = 0;
for(var i = 10; i < 20; i++){
      uniones[i].position.z = -2.8;
      uniones[i].rotation.z = angulo;
      angulo += Math.PI/10;
}
```

Para las uniones entre ruedas también se usó un ciclo *for* en el cual se incrementaba el ángulo de rotación en 36° grados. A diferencia de los ejes, con las uniones se estableció su posición *x,y* con la ley de senos y cosenos; también se utilizó una rotación de 90° sobre el eje *x* para colocarlos en forma horizontal. Así se logró que las uniones quedaran colocadas en forma circular, con su centro en la el punto medio entre las ruedas.

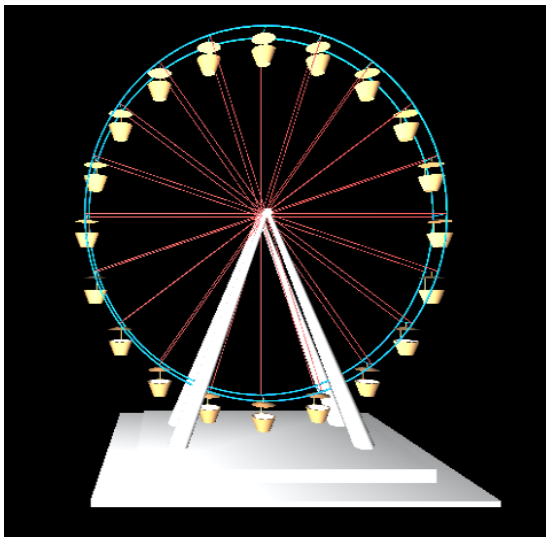
```
angulo = 0;
for (var i = 0; i < 20; i++) {
      unionRuedas[i].position.x = Math.cos(angulo) * 40;
      unionRuedas[i].position.y = Math.sin(angulo) * 40;
      unionRuedas[i].rotation.x = Math.PI / 2;

      angulo += Math.PI/10;
}
```

Las canastas, como ya se mencionó fueron dibujadas partiendo del punto donde se unen con la rueda. Se dibujó una sola canasta que formaba parte de un grupo que contenía el techo, las uniones y los dos cilindros base), este grupo se clonó 20 veces. La posición de estas canastas, al igual que las uniones, se definió dentro de un ciclo *for* con senos y cosenos multiplicados por el radio de la rueda.

```
//Clonando el grupo canasta y estableciendo sus posiciones
var angulo = 0;
var grupoCanastas = [];
for (var i = 0; i < 20; i++) {
      grupoCanastas[i] = grupoCanasta.clone();

      grupoCanastas[i].applyMatrix( new THREE.Matrix4().makeTranslation( Math.sin(angulo) * 40 , Math.cos(angulo) * 40, 0));

      angulo += Math.PI / 10;

      grupoRueda.add(grupoCanastas[i]);
};
```



Estos elementos se agregaron al grupo rueda y después se ubicaron los soportes, buscando que se unieran con las ruedas en el punto exacto para que pudieran rotar sin colisiones. Finalmente se colocó la base para que hiciera contacto con los soportes y que estos no quedaran en el aire. El grupo rueda, los soportes y la base se agregaron a un grupo rueda de la fortuna.



Funcionamiento

El giro de la rueda se implementó en el método del renderizado. Se incrementó el ángulo de rotación en z del grupo rueda con 0.003 unidades en cada ciclo.

Para que las canastas girarán respetando la gravedad y no quedaran de cabeza en determinado momento se hizo que estas rotaran, sobre el punto donde se une con la rueda, -0.003° grados (lo mismo que la rueda pero en sentido contrario).

```
//ROTACION DE LA RUEDA
grupoRueda.rotation.z += 0.003;

//Rotacion individual de cada canasta
for(i = 0; i < 20; i++)
      grupoCanastas[i].rotation.z -=0.003;
```

Tazas locas

En la creación de esta atracción se utilizaron 10 geometrías básicas (3 para la base, 2 para los platos, 4 para las tazas y una para el cilindro central).

- geometriaCilindroBaseTaza, geometriaCilindroE1 y geometriaCilindroE2 fueron los tres cilindros que formaron la base sobre la cual se asentaron las tazas. Estos cilindros tenían radios de 22, 22.3 y 22.5 unidades respectivamente y, 1, 0.5 y 0.2 unidades de alto.
- geometriaPlato (Un cilindro abierto con 2 unidades de radio superior, una unidad de radio inferior y 0.5 de alto)
- geometriaBasePlato (Un cilindro con 1 unidad de radio en sus dos extremos y 0.1 de alto que sirvió para "tapar" el cilindro abierto)
- geometriaTaza y geometriaTaza2 (Cilindros abiertos con diferentes radios que simularon las tazas, el radio inferior de estos cilindros fue de 1 y 0.9 unidades respectivamente, el radio superior fue de 2 y 1.9 unidades)
- geometriaUnionTaza (Anillo que sirvió para unir los dos cilindros que formaban las tazas con 1.9 como radio inter y 2 como radio externo)
- geometriaAza (Toroide con 0.6 de radio con un tubo de 0.2 de radio y una longitud de poco más de 180°)
- geometriaEsferaCentro (Una esfera de 3 unidades de radio que se colocó sobre el cilindro central)

Tamaño de los objetos

Las dimensiones utilizadas en la declaración de estas geometrías fueron calculadas usando como referencia el tamaño del carrusel y de la rueda de la fortuna de forma que quedaran más pequeños que la rueda pero similar al carrusel.

De igual forma se hizo que el tamaño de las tazas no fuera demasiado grande para que no se empalmaran o se vieran demasiado juntas, Tampoco podían ser demasiado grandes para que la tracción no quedará excesivamente grande.

Objetos

Se definieron 11 objetos con las geometrías que se especificaron anteriormente y uno del carrusel (3 para la base, 2 para el plato, 4 para las tazas y 2 para el centro).

Los 4 objetos que conformaban las tazas y los 2 elementos de los platos se agregaron a un mismo grupo taza. Este grupo se clonó 16 veces para crear el resto de las tazas.

En las tazas fueron utilizados 5 materiales diferentes (4 colores y una imagen):

- material: color gris claro utilizado en los platos y en el aza de la taza.
- materialBlanco: color blanco usado en el cilindro central.
- materialMorado: color morado usado en uno de los tres escalones y en la esfera del centro.
- materialDorado: color oro que utilizaron 2 de los 3 escalones.
- texturaTaza: Imagen con distintos tonos de morados que cubrió la taza.

Posiciones

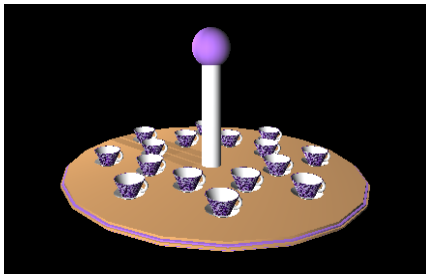
La atracción de las tazas locas fue ubicada partiendo de su base para terminar con hasta la esfera del centro.

El plato se colocó de forma que quedará exactamente sobre la base y la taza sobre el plato. La posición del aza se calculó para que quedará aproximadamente a la mitad de la taza.

El centro de todas las tazas fue calculado en dos ciclos *for* en los que se clonó la taza original y se calcularon las coordenadas *x,z.* Se usaron dos radios diferentes en cada ciclo y un ángulo que se incrementaba en 22.5° cada iteración.

```
//CLONANDO EL GRUPO TAZA Y ESTABLECIENDO SU CENTRO
var angulo = 0;
var grupoTazas = [];
for (var i = 0; i < 8 ; i++) {
      grupoTazas[i] = grupoTaza.clone();

      grupoTazas[i].applyMatrix( new THREE.Matrix4().makeTranslation(Math.sin(angulo) * 16, 0, Math.cos(angulo) * 16) );

      angulo += Math.PI / 4;
};

var angulo = 0;
for (var i = 8; i < 16 ; i++) {

      grupoTazas[i] = grupoTaza.clone();

      grupoTazas[i].applyMatrix(new THREE.Matrix4().makeTranslation(Math.sin(angulo + 0.392699082) * 10, 0, Math.cos(angulo + 0.392699082) * 10)
            );
      angulo += Math.PI / 4;
};
```

Finalmente las tazas se agregaron, su base y el cilindro central se agregaron a un único grupo que se agregó a la escena.



Funcionamiento

En esta atracción se tuvieron que hacer girar dos grupos diferentes, el grupo que contenía toda la atracción y los grupos que contenían solo las tazas.

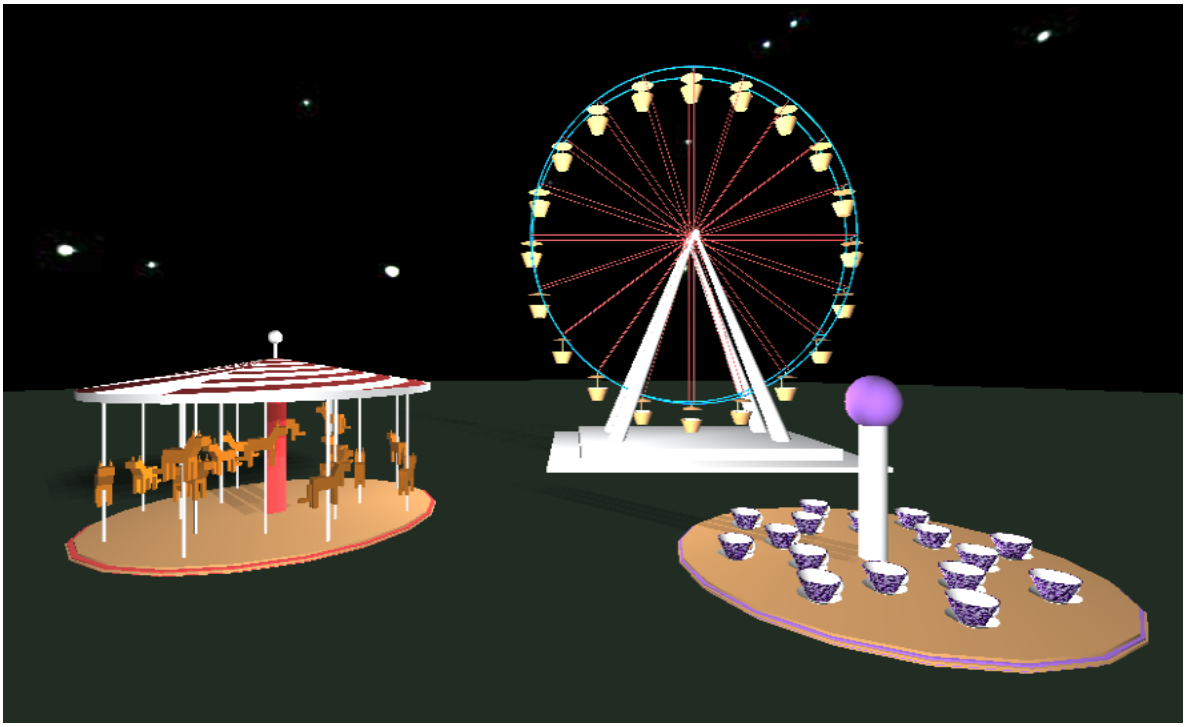
El grupo global rotó en cada iteración del método *render* 0.07° en contra de las manecillas de reloj y las tazas giraban, a su vez, 0.02 grado a favor de las manecillas del reloj.

```
//rotacion individual de cada taza
for(i = 0; i < 16; i++)
      grupoTazas[i].rotation.y +=0.07;

//rotacion de la base
grupoTazasLocas.rotation.y -= 0.02;
```

Resultado

La posición de las tres atracciones se fue ajustando hasta que las 3 estaban colocadas uniformemente generando una vista agradable de toda la escena.

Al final, se agregó un plano que simula un piso y una esfera que simula el cielo de la escena.



AUTOMÓVIL

Para la creación del automóvil se definieron 11 geometrías nuevas (3 para la cajuela, 2 para el techo, 3 para el cofre, 2 para las puertas y 1 para el piso):

- Cajuela:
    - estructuraBaseCajuela: estructura con 16 puntos definidos (8 frontales y 8 traseros) y 28 caras (6 para el frente, 6 para el fondo y 16 para las uniones) con una profundidad de 0.5 unidades.
    - estructuraCajuela: estructura con 16 puntos definidos (8 frontales y 8 traseros) y 28 caras (6 para el frente, 6 para el fondo y 16 para las uniones) con una profundidad de 9 unidades.
    - estructuraRellenoCajuela estructura con 14 puntos definidos (7 frontales y 7 traseros) y 22 caras (5 para el frente, 5 para el fondo y 12 para las uniones) con una profundidad de 0.5 unidades.

- Techo:
  - estructuraBaseTecho: estructura con 24 puntos definidos (12 frontales y 12 traseros) y 44 caras (10 para el frente, 10 para el fondo y 24 para las uniones) con una profundidad de 0.5 unidades.
  - estructuraTecho: estructura con 16 puntos definidos (8 frontales y 8 traseros) y 28 caras (6 para el frente, 6 para el fondo y 16 para las uniones) con una profundidad de 9 unidades.
- Cofre:
  - estructuraBaseCofre: estructura con 16 puntos definidos (8 frontales y 8 traseros) y 28 caras (6 para el frente, 6 para el fondo y 16 para las uniones) con una profundidad de 0.5 unidades.
  - estructuraCofre: estructura con 16 puntos definidos (8 frontales y 8 traseros) y 28 caras (6 para el frente, 6 para el fondo y 16 para las uniones) con una profundidad de 9 unidades.
  - estructuraRellenoCofre estructura con 14 puntos definidos (7 frontales y 7 traseros) y 22 caras (5 para el frente, 5 para el fondo y 12 para las uniones) con una profundidad de 0.5 unidades.
- Puertas:
  - estructuraPuertaTrasera: estructura con 18 puntos definidos (9 frontales y 9 traseros) y 30 caras (7 para el frente, 7 para el fondo y 16 para las uniones) con una profundidad de 0.5 unidades.
  - estructuraPuertaDelantera: estructura con 12 puntos definidos (6 frontales y 6 traseros) y 20 caras (4 para el frente, 4 para el fondo y 12 para las uniones) con una profundidad de 0.5 unidades.
- Piso:
  - estructuraPiso: estructura con 40 puntos definidos (20 frontales y 20 traseros) y 76 caras (18 para el frente,

18 para el fondo y 40 para las uniones) con una profundidad de 0.5 unidades.

Para la creación y definición de todas estas geometrías se utilizaron arreglos de puntos que se declararon al inicio. Con estos arreglos de puntos se ejecutaron ciclos que insertaron los puntos en las nuevas geometrías. Para insertar las caras se definieron patrones que permitieran generarlas de forma automática con distintos ciclos *for*.

Se utilizaron 10 geometrías ya definidas:

- estructuraSeparacion (Un cubo de 1 unidad de ancho, 13 de largo y 0.5 de profundidad que representa la división normal que existe entre dos puertas de un coche)
- estructuraRetrovisor (Un cubo de 6 unidades de ancho, 1 de largo y 9 de profundidad que representa el retrovisor de un auto)
- estructuraParabrisas (Un cubo de 6.5 unidades de ancho, 1 de largo y 9 de profundidad que representa el parabrisas de un auto)
- estructuraVentanaDelantera (Un cubo de 2.5 unidades de ancho, 5 de largo y 0.6 de profundidad que se utilizó como la ventana de la puerta trasera del auto)
- estructuraVentanaTrasera (Un cubo de 5 unidades de ancho, 5 de largo y 0.6 de profundidad que se utilizó como la ventana de la puerta delantera del auto)
- estructuraDefensa (Un cubo de 3 unidades de ancho, 1 de largo y 10 de profundidad que se utilizó como la defensa trasera y delantera)
- estructuraFaro (Media esfera de 0.75 de radio)
- estructuraLlanta (Cilindro de 2.5 de radio y 1 unidad de alto)

- estructuraUnion (Cubo de 0.5 unidades de ancho, 5.5 de alto y 10 de profundidad que une los dos lados del cofre y la cajuela a forma de "tapa")
- estructuraManija (Un cuarto de esfera con 0.5 de radio)

Tamaño de objetos

Por cuestiones de proporción se decidió que las medidas correctas para el coche eran 40 unidades de ancho, 15 de alto y 10 de profundidad. En base a estas medidas se hizo la distribución de cada uno de sus componentes.

Objetos

Se crearon 37 objetos con las geometrías anteriores (ya se las definidas desde 0 o las ya existentes) 8 de esto objetos fueron para la cajuela, 3 para el techo, 10 del cofre, 14 del centro (puertas, ventanas, manijas y separación), 1 del piso y 4 para las llantas.

Se definieron 6 grupos: Uno para todo el coche, uno para cada puerta y uno para las llantas. El grupo de las llantas está conformado por las 4 llantas, el de las puertas lo conforman la estructura de la puerta, su manija y su ventana; al coche lo integran los grupos anteriores, la cajuela, el cofre, los faros, el techo y el piso.

Posiciones

El coche y sus componentes fijos que trazaron a partir del centro geométrico de todos sus componentes sin embargo, los elementos móviles del automóvil se tuvieron que trazar a partir de aquel punto sobre el que rotarían: la cajuela y el cofre se dibujaron a

partir del punto de unión de estos con el resto del coche y las puertas se trazaron partiendo del extremo derecho.

Las 4 llantas tuvieron que rotarse 90° sobre el eje x para que quedaran con las caras paralelas a la vista. El parabrisas y el retrovisor se tuvieron que rotar 65° y 60° respectivamente para que encajaran con el techo. Los faros y las manijas también se rotaron para que la parte de la esfera dibujada quedara hacia afuera.

Funcionamiento

Para que el automóvil avanzara, retrocediera, girara a la izquierda y a la derecha se implementaron acciones con las teclas T, G, F y H respectivamente. Con estas teclas se manipulaban los valores de posición y rotación del grupo carro y de las llantas.

Cómo las llantas tenían un límite de rotación, se implementaron condiciones que se tenían que cumplir: si la llanta había alcanzado su nivel de rotación máximo/mínimo se mantenía ahí sin modificarse. Estos valores máximos/mínimos se obtuvieron en base a la observación de resultados con distintos valores)

```
case 84: /*T was pressed -> Auto avanza*/
      var dx = 5 * Math.sin(carroRotation - Math.PI / 2 );
      var dz = 5 * Math.cos(carroRotation - Math.PI / 2 );
      zPosCarro -= dz;
      xPosCarro -= dx;
      llantasRotation -= 0.5;
      llantasRotationZ = 0.0;
      break;

case 71: /*G was pressed -> Auto retrocede*/
      var dx = 5 * Math.sin(carroRotation + Math.PI / 2 );
      var dz = 5 * Math.cos(carroRotation + Math.PI / 2 );
      zPosCarro -= dz;
      xPosCarro -= dx;
      llantasRotation += 0.5;
      llantasRotationZ = 0.0;
      break;


case 72: /*H was pressed -> Auto rota a la derecha*/
      carroRotation -= (5 * Math.PI) / 180;
      if (llantasRotationZ + 0.003 < 0.4)
            llantasRotationZ += 0.003;
      break;

case 70: /*F was pressed -> Auto rota a la izquierda*/
      carroRotation += (5 * Math.PI) / 180;

      if (llantasRotationZ - 0.003 > -0.4)
            llantasRotationZ -= 0.003;
      break;
```

En el método *render* se tomaban esos valores y se aplicaban al carro y las llantas para proyectarlo en la escena.

```
/*MOVIMIENTO DEL CARRO*/
grupoCarro.position.x = xPosCarro;
grupoCarro.position.z = zPosCarro;

grupoCarro.position.y = -13;
grupoCarro.rotation.y = carroRotation;

//Llantas
llantaDelanteraFrente.rotation.y = llantasRotation;
llantaDelanteraFrente.rotation.z =  llantasRotationZ;
llantaTraseraFrente.rotation.y = llantasRotation;
llantaTraseraFrente.rotation.z = llantasRotationZ;
llantaDelanteraFondo.rotation.y = llantasRotation;
llantaDelanteraFondo.rotation.z = llantasRotationZ;
llantaTraseraFondo.rotation.y = llantasRotation;
llantaTraseraFondo.rotation.z = llantasRotationZ;
```

Para la funcionalidad del cofre y la cajuela se utilizaron las teclas R(cierra el cofre), Y(abre el cofre), V(abre la cajuela) y B(cierra la cajuela). Para estos movimientos también se implementaron condiciones a cumplir antes de realizar la modificación de valores.

```
case 66: /*B was pressed -> Cierra cajuela*/
      if (cajuelaRotation + 0.03 <= 0)
            cajuelaRotation += 0.03;
      break;


case 86: /*V was pressed -> Abre cajuela*/
      if (cajuelaRotation - 0.03 >= -1.1)
            cajuelaRotation -= 0.03;
      break;


case 82: /*R was pressed -> Cierra cofre*/
      if (cofreRotation - 0.03 >= 0)
            cofreRotation -= 0.03;
      break;


case 89: /*Y was pressed -> Abre cofre*/
      if (cofreRotation + 0.03 <= 1)
            cofreRotation += 0.03;
      break;
```

```
//Cajuela
cajuela.rotation.z = cajuelaRotation;

//Cofre
cofre.rotation.z = cofreRotation;
cofre.position.y = 1;
```

En el funcionamiento de las puertas se aplicó el mismo principio que en los dos movimientos anteriores sólo que aquí se manejaron

dos variables: una para las puertas del frente y otras para las del fondo ya que, mientras unas manejan ángulos positivos las otras manejan ángulos negativos.

```
case 74: /*J was pressed -> Cierra puertas*/
      if (puertasFrenteRotation - 0.03 >= 0) {
            puertasFrenteRotation -= 0.03;
            puertasFondoRotation += 0.03;
      }
      break;

case 85: /*U was pressed -> Abre puertas*/
      if (puertasFrenteRotation + 0.03 <= 0.75) {
            puertasFrenteRotation += 0.03;
            puertasFondoRotation -= 0.03;
      }
      break;
```

```
//Puertas
puertaTraseraFrente.rotation.y = puertasFrenteRotation;
puertaTraseraFrente.position.x = -2;

puertaDelanteraFrente.rotation.y = puertasFrenteRotation;
puertaDelanteraFrente.position.x = 6;

puertaTraseraFondo.rotation.y = puertasFondoRotation;
puertaTraseraFondo.position.x = -2;
puertaTraseraFondo.position.z = -9.5;

puertaDelanteraFondo.rotation.y = puertasFondoRotation;
puertaDelanteraFondo.position.x = 6;
puertaDelanteraFondo.position.z = -9.5;
```

Resultado

CONSTRUCCION

La construcción en la que se trabajo fue el castillo de pitch. Para su desarrollo se aplicaron diferentes geometiras utilizando la librería Three.js y la creación de geometrias nuevas.

Como parte de las geometrias de Three.js se utilizaron las geometrias Cylinder y Torus. Específicamente se utilizaron apra realizar la barda, las torres y pisos del castillo.

```
//Primer piso
var geomCilindro1Piso = new THREE.CylinderGeometry(72,72,48,20,20);
var geomTecho1Piso = new THREE.CylinderGeometry(56,72,16,20,20);

var geomCilindroTorre1 = new THREE.CylinderGeometry(16,16,80,20,20);
var geomConoTorre1 = new THREE.CylinderGeometry(3,12,30,28,20);

//Segundo piso
var geomTerraza = new THREE.CylinderGeometry(56,56,8,20,20);
var geomCilindro2Piso = new THREE.CylinderGeometry(40,40,24,20,20);
var geomTecho2Piso = new THREE.CylinderGeometry(24,40,16,20,20);

var geomCilindroTorre2 = new THREE.CylinderGeometry(20,20,90,20,20);


//Torre tercer piso
var geomCilindroTorre3 = new THREE.CylinderGeometry(24,24,24,20,20);
var geomConoTorre3 = new THREE.CylinderGeometry(3,16,30,20,20);

var geomBandera = new THREE.CylinderGeometry(1,1,10,20,20);
var geomAstaBandera = new THREE.CylinderGeometry(1,1,10,20,20);

//Barda castillo
var geometriaBarda = new THREE.TorusGeometry(140, 25, 3, 15, 5);
var geometriaTorre = new THREE.CylinderGeometry(20,30,80,15,20,false);
var geometriaTorreTecho = new THREE.CylinderGeometry(22,22,2,15,20,false);

//Base castillo
var geometriaBaseCastillo = new THREE.CylinderGeometry(110,110,1,8,20,false);
```

Para realizar geometrias específicas como las puertas, ventanas y los relieves de algunas torres se utilizaron geometrias creadas manualmente.

```
//geometria puerta
var geometriaPuerta = new THREE.Geometry();
//geometria ventana
var geometriaVentana = new THREE.Geometry();
//geometria relieve
var relieveGeom = new THREE.Geometry();
```

Tamaño de objetos

El castillo se realizó en base a las dimenciones en las que se trabajaron para los objetos como la rueda, el carro, y los demas juegos mecanicos, con el objetivo de ver el escenario de manera proporcional.

El castillo cuenta con un ancho total de 140 unidades de ancho por una altura de 120 unidades.

Objetos

Se crearon 30 objetos para crear el castillo de los cuales: 1 se utilizo para la puerta, 5 para las ventanas, 8 para conformar la barda, 1 para la base principal del castillo, 3 para el primer piso, 9 para las torres del castillo, 3 para el segundo piso.

Para poder manipular de mejor manera todos estos objetos se realizaron grupos. Con un total de 4 grupos se maneja la estructura completa.

Se cuenta con un grupo principal en el cual se cargan todos los elementos y demas grupos el cual es el grupo Castillo, los otros grupos son: torre1piso, grupoBarda, grupoCastillo.

```
/*GRUPOS*/
var castillo = new THREE.Object3D();
var torre1Piso = new THREE.Object3D();
var grupoBarda = new THREE.Object3D();
var grupoCastillo = new THREE.Object3D();
```

Posiciones

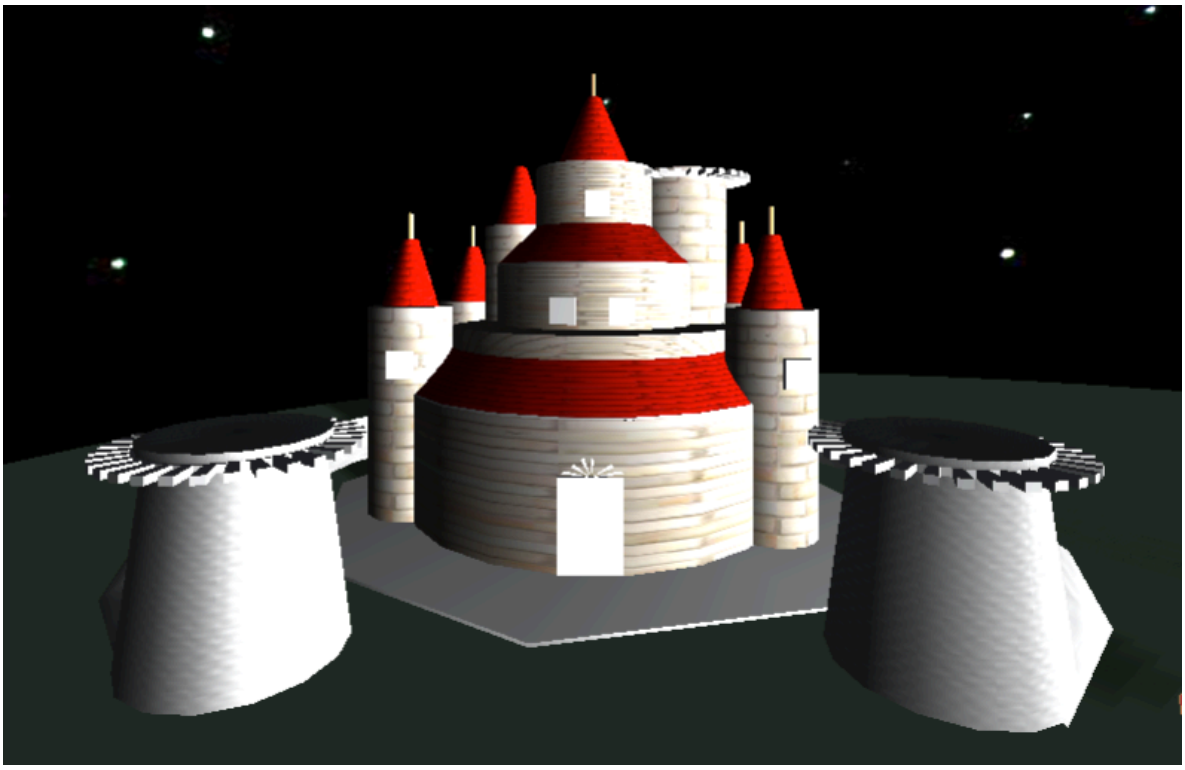
Las posiciones de todos los elementos y objetos creados que definen a la estructura fueron creados a partir del punto orgen y posteriormente a crear los grupos, fueron desplazados a las

diferentes posiciones en las que debian estar mostradas y rotados en caso de necesitarlo.

```
barda.rotation.x = Math.PI/2;
barda.rotation.z = Math.PI/2;
barda.position.y = -24;

torre1.position.y = -14;
torre1.position.z = 140;

relieve.rotation.x = Math.PI/2;
relieve.position.y = 27;
relieve.position.z = 122;
relieve.position.x = -70;

torre2.position.y = -14;
torre2.position.z = 35;
torre2.position.x =133;

relieve2.rotation.x = Math.PI/2;
relieve2.position.y = 27;
relieve2.position.z = 96;
relieve2.position.x = 98;

relieve3.rotation.x = Math.PI/2;
relieve3.position.y = 110;
relieve3.position.z = -24;
relieve3.position.x = 35;
```

Resultado

El resultado obtenido de la convinación de elementos geometricos proporcionados por three.js y las geometrias definidas independientemente fue el siguiente.



Resultado Final

Finalmente para la creación de sombras se declaró una luz ambiental y una luz direccional que se colocó de tal manera que abarcara toda la escena y generará una sombra diagonal sobre los objetos.

Todos los objetos de la escena se declararon como capaces de generar sombras y las bases y el piso de la ferio se declararon como objetos capaces de recibir sombras.

APENDICE: CÓDIGO

Automovil.js

```
/*DECLARACION DE GRUPOS*/
var grupoCarro = new THREE.Object3D();
var puertaDelanteraFrente = new THREE.Object3D();
var puertaDelanteraFondo = new THREE.Object3D();
var puertaTraseraFondo = new THREE.Object3D();
var puertaTraseraFrente = new THREE.Object3D();
var cofre = new THREE.Object3D();

/*MATERIALES*/
```

```
var vidrio= new THREE.MeshLambertMaterial ({ map: THREE.ImageUtils.loadTexture("img/vidrio.jpg")});

var aluminio = new THREE.MeshLambertMaterial ({ map: THREE.ImageUtils.loadTexture("img/aluminio.jpg")});

var foco = new THREE.MeshLambertMaterial ({ map: THREE.ImageUtils.loadTexture("img/faro.jpg")});

var plastico= new THREE.MeshLambertMaterial ({ map: THREE.ImageUtils.loadTexture("img/plasticoTextura.jpg")});

var caucho = new THREE.MeshLambertMaterial ({ map: THREE.ImageUtils.loadTexture("img/llanta.jpg")});

var rojo = new THREE.MeshLambertMaterial({color: 0xFF0000, side: THREE.DoubleSide});

var negro = new THREE.MeshLambertMaterial({color: 0x575757, side: THREE.DoubleSide});


/*VARIABLES DE ANCHO*/

z = 0.5;

z1 = 10;

z2 = 9;


/*CREANDO LA ESTRUCTURA DE LA BASE DE LA CAJUELA*/

var estructuraBaseCajuela = new THREE.Geometry();




//PUNTOS


//Arreglo de puntos

var x = [ -19 , -19 , -18 , -11 ,  -20 , -20 , -19, -11];

var y = [-6.5 ,  -1 ,   0 ,   0 , -6.5 ,   0 ,  1 ,   1];

var numPuntos = 8;


//Puntos Frentes (0 - 7)

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

    estructuraBaseCajuela.vertices.push (new THREE.Vector3 (x[i], y[i], 0));

};


//Puntos Fondos (8 - 15)

for (var i = 0; i < numPuntos; i++) {
```

```
    estructuraBaseCajuela.vertices.push (new THREE.Vector3 (x[i], y[i], -z));
};

//CARAS
for (var i = 0; i < 3; i++) {
    //Caras delanteras
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i, i + 1, i + 5));
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i, i + 5, i + 4));

    //Caras traseras
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i + 8, i + 13, i + 9));
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i + 8, i + 12, i + 13));

    //Uniones externas
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i + 4, i + 5, i + 13));
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i + 4, i + 13, i + 12));

    //Uniones internas
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i , i + 9, i + 1));
    estructuraBaseCajuela.faces.push (new THREE.Face3 (i , i + 8, i + 9));
};

//Uniones finales
//Superior
estructuraBaseCajuela.faces.push (new THREE.Face3 (7, 3, 11));
estructuraBaseCajuela.faces.push (new THREE.Face3 (7, 11, 15));
//Inferior
estructuraBaseCajuela.faces.push (new THREE.Face3 (4, 0, 8));
estructuraBaseCajuela.faces.push (new THREE.Face3 (4, 8, 12));


//COMPUTE
estructuraBaseCajuela.computeFaceNormals();
```

```
/*CREANDO LA CAJUELA*/
var estructuraCajuela = new THREE.Geometry();


//PUNTOS
//Arreglo de puntos
var x = [ -7.5 ,  -7.5 , -6.5 , 0.5 ,  -8.5 , -8.5 , -7.5 , 0.5];
var y = [ -7.5 ,    -2 ,   -1 ,  -1 ,  -7.5 ,   -1 ,    0 , 0];
var numPuntos = 8;
//Puntos Frentes (0 - 7)
for (var i = 0; i < numPuntos; i++) {
	estructuraCajuela.vertices.push (new THREE.Vector3 (x[i], y[i], -z));
};
//Puntos Fondos (8 - 15)
for (var i = 0; i < numPuntos; i++) {
	estructuraCajuela.vertices.push (new THREE.Vector3 (x[i], y[i], -z2 - z));
};

//CARAS
for (var i = 0; i < 3; i++) {
	//Caras delanteras
	estructuraCajuela.faces.push (new THREE.Face3 (i, i + 1, i + 5));
	estructuraCajuela.faces.push (new THREE.Face3 (i, i + 5, i + 4));

	//Caras traseras
	estructuraCajuela.faces.push (new THREE.Face3 (i + 8, i + 13, i + 9));
	estructuraCajuela.faces.push (new THREE.Face3 (i + 8, i + 12, i + 13));

	//Uniones externas
	estructuraCajuela.faces.push (new THREE.Face3 (i + 4, i + 5, i + 13));
	estructuraCajuela.faces.push (new THREE.Face3 (i + 4, i + 13, i + 12));

	//Uniones internas
	estructuraCajuela.faces.push (new THREE.Face3 (i , i + 9, i + 1));
	estructuraCajuela.faces.push (new THREE.Face3 (i , i + 8, i + 9));
};
```

```
//Uniones finales
//Superior
estructuraCajuela.faces.push (new THREE.Face3 (7, 3, 11));
estructuraCajuela.faces.push (new THREE.Face3 (7, 11, 15));
//Inferior
estructuraCajuela.faces.push (new THREE.Face3 (4, 0, 8));
estructuraCajuela.faces.push (new THREE.Face3 (4, 8, 12));


//COMPUTE
estructuraCajuela.computeFaceNormals();


/*CREANDO EL RELLENO DE LA CAJUELA*/
var estructuraRellenoCajuela = new THREE.Geometry();

//PUNTOS
//Arreglo de puntos
var x = [ -19 , -19 , -18 , -11 ,  -11 ,  -12 ,  -12];
var y = [-6.5 ,  -1 ,   0 ,   0 , -4.5 , -4.5 , -6.5];
var numPuntos = 7;

//Puntos Frentes (0 - 6)
for (var i = 0; i < numPuntos; i++) {
      estructurRellenoCajuela.vertices.push (new THREE.Vector3 (x[i], y[i], 0));
};

//Puntos Fondos (7 - 13)
for (var i = 0; i < numPuntos; i++) {
      estructuraRellenoCajuela.vertices.push  (new  THREE.Vector3  (x[i],  y[i],  -
z));
};


//CARAS
```

```
for (var i = 4; i >= 1; i--) {
	//Caras delanteras
	estructuraRellenoCajuela.faces.push (new THREE.Face3 (5, i, i - 1));
	//Caras traseras
	estructuraRellenoCajuela.faces.push (new THREE.Face3 (12, i + 7, i + 6));
};
estructuraRellenoCajuela.faces.push (new THREE.Face3 (5, 0, 6));
estructuraRellenoCajuela.faces.push (new THREE.Face3 (12, 13, 7));
for (var i = 0; i < 6; i++) {
	//Uniones
	estructuraRellenoCajuela.faces.push (new THREE.Face3 (i, i + 1, i + 8));
	estructuraRellenoCajuela.faces.push (new THREE.Face3 (i, i + 8, i + 7));
};

//COMPUTE
estructuraRellenoCajuela.computeFaceNormals();




/*CREANDO LA ESTRUCTURA BASE DEL TECHO*/
var estructuraBaseTecho = new THREE.Geometry();


//PUNTOS

//Arreglo de puntos
var x = [ -11 ,  -8 ,   -7 ,   1 ,  2 , 6 , -12 ,  -9 , -7.5 , 1.5 ,   3 , 7];
var y = [   0 ,   6 ,  6.5 , 6.5 ,  6 , 0 ,   0 , 6.5 ,  7.5 , 7.5 , 6.5 , 0];
var numPuntos = 12;

//Puntos Frentes (0 - 11)
for (var i = 0; i < numPuntos; i++) {
```

```
	estructuraBaseTecho.vertices.push (new THREE.Vector3 (x[i], y[i], 0));
};

//Puntos Fondos (12 - 23)
for (var i = 0; i < numPuntos; i++) {
	estructuraBaseTecho.vertices.push (new THREE.Vector3 (x[i], y[i], -z));
};

//CARAS
for (var i = 0; i < 5; i++) {
	//Caras delanteras
	estructuraBaseTecho.faces.push (new THREE.Face3 (i, i + 1, i + 7));
	estructuraBaseTecho.faces.push (new THREE.Face3 (i, i + 7, i + 6));

	//Caras traseras
	estructuraBaseTecho.faces.push (new THREE.Face3 (i + 12, i + 18, i + 19));
	estructuraBaseTecho.faces.push (new THREE.Face3 (i + 12, i + 19, i + 13));

	//Uniones externas
	estructuraBaseTecho.faces.push (new THREE.Face3 (i + 6, i + 7, i + 19));
	estructuraBaseTecho.faces.push (new THREE.Face3 (i + 6, i + 19, i + 18));

	//Uniones internas
	estructuraBaseTecho.faces.push (new THREE.Face3 (i , i + 12, i + 13));
	estructuraBaseTecho.faces.push (new THREE.Face3 (i , i + 13, i + 1));
};

//Uniones finales
//Derecha
estructuraBaseTecho.faces.push (new THREE.Face3 (6, 0, 12));
estructuraBaseTecho.faces.push (new THREE.Face3 (6, 12, 18));
//Izquierda
estructuraBaseTecho.faces.push (new THREE.Face3 (5, 11, 23));
estructuraBaseTecho.faces.push (new THREE.Face3 (5, 23, 17));
```

```
//COMPUTE
estructuraBaseTecho.computeFaceNormals();



/*CREANDO EL TECHO*/
var estructuraTecho = new THREE.Geometry();

//PUNTOS
//Arreglo de puntos
var x = [-8 ,  -7 ,   1 ,  2 ,   -9 , -7.5 , 1.5 ,   3 ];
var y = [ 6 , 6.5 , 6.5 ,  6 ,  6.5 ,  7.5 , 7.5 , 6.5 ];
var numPuntos = 8;

//Puntos Frentes (0 - 7)
for (var i = 0; i < numPuntos; i++) {
      estructuraTecho.vertices.push (new THREE.Vector3 (x[i], y[i], 0));
};

//Puntos Fondos (8 - 15)
for (var i = 0; i < numPuntos; i++) {
      estructuraTecho.vertices.push (new THREE.Vector3 (x[i], y[i], -z2 - z));
};
//CARAS
for (var i = 0; i < 3; i++) {
      //Caras delanteras
      estructuraTecho.faces.push (new THREE.Face3 (i, i + 1, i + 5));
      estructuraTecho.faces.push (new THREE.Face3 (i, i + 5, i + 4));



      //Caras traseras
      estructuraTecho.faces.push (new THREE.Face3 (i + 8, i + 12, i + 13));
      estructuraTecho.faces.push (new THREE.Face3 (i + 8, i + 13, i + 9));

      //Uniones externas
      estructuraTecho.faces.push (new THREE.Face3 (i + 4, i + 5, i + 13));
```

```
	estructuraTecho.faces.push (new THREE.Face3 (i + 4, i + 13, i + 12));

	//Uniones internas
	estructuraTecho.faces.push (new THREE.Face3 (i , i + 8, i + 9));
	estructuraTecho.faces.push (new THREE.Face3 (i , i + 9, i + 1));
};

//Uniones finales
//Derecha
estructuraTecho.faces.push (new THREE.Face3 (4, 0, 8));
estructuraTecho.faces.push (new THREE.Face3 (4, 8, 12));
//Izquierda
estructuraTecho.faces.push (new THREE.Face3 (3, 7, 15));
estructuraTecho.faces.push (new THREE.Face3 (3, 15, 11));


//COMPUTE
estructuraTecho.computeFaceNormals();




/*CREANDO LA ESTRUCTURA DE LA BASE DEL COFRE*/
var estructuraBaseCofre = new THREE.Geometry();


//PUNTOS
//Arreglo de puntos
var x = [ 6 , 18 , 19 ,   19 ,   6 , 19 , 20,   20];
var y = [ 0 ,  0 , -1 , -6.5 ,   1 ,  1 ,  0, -6.5];
var numPuntos = 8;
//Puntos Frentes (0 - 7)
for (var i = 0; i < numPuntos; i++) {
	estructuraBaseCofre.vertices.push (new THREE.Vector3 (x[i], y[i], 0));
};
```

```
//Puntos Fondos (8 - 15)
for (var i = 0; i < numPuntos; i++) {
	estructuraBaseCofre.vertices.push (new THREE.Vector3 (x[i], y[i], -z));
};


//CARAS
for (var i = 0; i < 3; i++) {
	//Caras delanteras
	estructuraBaseCofre.faces.push (new THREE.Face3 (i, i + 1, i + 5));
	estructuraBaseCofre.faces.push (new THREE.Face3 (i, i + 5, i + 4));


	//Caras traseras
	estructuraBaseCofre.faces.push (new THREE.Face3 (i + 8, i + 13, i + 9));
	estructuraBaseCofre.faces.push (new THREE.Face3 (i + 8, i + 12, i + 13));

	//Uniones externas
	estructuraBaseCofre.faces.push (new THREE.Face3 (i + 4, i + 5, i + 13));
	estructuraBaseCofre.faces.push (new THREE.Face3 (i + 4, i + 13, i + 12));

	//Uniones internas
	estructuraBaseCofre.faces.push (new THREE.Face3 (i , i + 9, i + 1));
	estructuraBaseCofre.faces.push (new THREE.Face3 (i , i + 8, i + 9));
};

//Uniones finales
//Superior
estructuraBaseCofre.faces.push (new THREE.Face3 (7, 3, 11));
estructuraBaseCofre.faces.push (new THREE.Face3 (7, 11, 15));
//Inferior
estructuraBaseCofre.faces.push (new THREE.Face3 (4, 0, 8));
estructuraBaseCofre.faces.push (new THREE.Face3 (4, 8, 12));
```

```
//COMPUTE
estructuraBaseCofre.computeFaceNormals();




/*CREANDO EL COFRE*/
var estructuraCofre = new THREE.Geometry();

//PUNTOS
//Arreglo de puntos
var x = [ 0 , 11 , 12 ,   12 , -0.5 , 12 , 13 ,   13];
var y = [-1 , -1 , -2 , -7.5 ,    0 ,  0 , -1 , -7.5];
var numPuntos = 8;

//Puntos Frentes (0 - 7)
for (var i = 0; i < numPuntos; i++) {
      estructuraCofre.vertices.push (new THREE.Vector3 (x[i], y[i], -z));
};

//Puntos Fondos (8 - 15)
for (var i = 0; i < numPuntos; i++) {
      estructuraCofre.vertices.push (new THREE.Vector3 (x[i], y[i], -z1));
};


//CARAS
for (var i = 0; i < 3; i++) {
      //Caras delanteras
      estructuraCofre.faces.push (new THREE.Face3 (i, i + 1, i + 5));
      estructuraCofre.faces.push (new THREE.Face3 (i, i + 5, i + 4));


      //Caras traseras
      estructuraCofre.faces.push (new THREE.Face3 (i + 8, i + 13, i + 9));
      estructuraCofre.faces.push (new THREE.Face3 (i + 8, i + 12, i + 13));
```

```
        //Uniones externas
        estructuraCofre.faces.push (new THREE.Face3 (i + 4, i + 5, i + 13));
        estructuraCofre.faces.push (new THREE.Face3 (i + 4, i + 13, i + 12));

        //Uniones internas
        estructuraCofre.faces.push (new THREE.Face3 (i , i + 9, i + 1));
        estructuraCofre.faces.push (new THREE.Face3 (i , i + 8, i + 9));
};

//Uniones finales
//Superior
estructuraCofre.faces.push (new THREE.Face3 (7, 3, 11));
estructuraCofre.faces.push (new THREE.Face3 (7, 11, 15));
//Inferior
estructuraCofre.faces.push (new THREE.Face3 (4, 0, 8));
estructuraCofre.faces.push (new THREE.Face3 (4, 8, 12));



//COMPUTE
estructuraCofre.computeFaceNormals();







/*CREANDO EL RELLENO DEL COFRE*/
var estructuraRellenoCofre = new THREE.Geometry();



//PUNTOS

//Arreglo de puntos
var x = [ 6 , 18 , 19 ,   19 ,   13 ,   13 ,    6];
var y = [ 0 ,  0 , -1 , -6.5 , -6.5 , -4.5 , -4.5];
var numPuntos = 7;
```

```
//Puntos Frentes (0 - 6)
for (var i = 0; i < numPuntos; i++) {
	estructuraRellenoCofre.vertices.push (new THREE.Vector3 (x[i], y[i], 0));
};

//Puntos Fondos (7 - 13)
for (var i = 0; i < numPuntos; i++) {
	estructuraRellenoCofre.vertices.push (new THREE.Vector3 (x[i], y[i], -z));
};


//CARAS
for (var i = 0; i < 4; i++) {
	//Caras delanteras
	estructuraRellenoCofre.faces.push (new THREE.Face3 (5, i + 1, i));

	//Caras traseras
	estructuraRellenoCofre.faces.push (new THREE.Face3 (12, i + 8, i + 7));
};
estructuraRellenoCofre.faces.push (new THREE.Face3 (5, 0, 6));
estructuraRellenoCofre.faces.push (new THREE.Face3 (12, 7, 13));
for (var i = 0; i < 6; i++) {
	//Uniones
	estructuraRellenoCofre.faces.push (new THREE.Face3 (i, i + 1, i + 8));
	estructuraRellenoCofre.faces.push (new THREE.Face3 (i, i + 8, i + 7));
};
estructuraRellenoCofre.faces.push (new THREE.Face3 (13, 0, 7));
estructuraRellenoCofre.faces.push (new THREE.Face3 (13, 6, 0));


//COMPUTE
estructuraRellenoCofre.computeFaceNormals();
```

```
/*CREANDO LA PUERTA TRASERA*/
var estructuraPuertaTrasera = new THREE.Geometry();


//PUNTOS

//Arreglo de puntos
var x = [ -9 , -6 , -5 ,   0 ,    0 , -3 , -3 , -9];
var y = [    0 ,    6 ,  6.5 , 6.5 , -6.5 , -6.5 , -4.5 , -4.5];
var numPuntos = 8;

//Puntos Frentes (0 - 8)
for (var i = 0; i < numPuntos; i++) {
      estructuraPuertaTrasera.vertices.push (new THREE.Vector3 (x[i], y[i], 0));
};
estructuraPuertaTrasera.vertices.push (new THREE.Vector3 (-1.5, -4.5, 0));//8

//Puntos Fondos (9 - 17)
for (var i = 0; i < numPuntos; i++) {
      estructuraPuertaTrasera.vertices.push (new THREE.Vector3 (x[i], y[i], -z));
};
estructuraPuertaTrasera.vertices.push (new THREE.Vector3 (-1.5, -4.5, -z));//17


//CARAS
for (var i = 0; i < 3; i++) {
      //Caras delanteras
      estructuraPuertaTrasera.faces.push (new THREE.Face3 (7, i + 1, i));

      //Caras traseras
      estructuraPuertaTrasera.faces.push (new THREE.Face3 (16, i + 10, i + 9));
```

```
};
//Restantes delanteros
estructuraPuertaTrasera.faces.push (new THREE.Face3 (7, 8, 3));
estructuraPuertaTrasera.faces.push (new THREE.Face3 (4, 8, 6));
estructuraPuertaTrasera.faces.push (new THREE.Face3 (4, 6, 5));
estructuraPuertaTrasera.faces.push (new THREE.Face3 (5, 4, 3));

//Restantes traseras
estructuraPuertaTrasera.faces.push (new THREE.Face3 (16, 17, 12));
estructuraPuertaTrasera.faces.push (new THREE.Face3 (13, 12, 15));
estructuraPuertaTrasera.faces.push (new THREE.Face3 (13, 15, 14));
estructuraPuertaTrasera.faces.push (new THREE.Face3 (14, 13, 12));

for (var i = 0; i < 7; i++) {
	//Uniones
	estructuraPuertaTrasera.faces.push (new THREE.Face3 (i, i + 1, i + 10));
	estructuraPuertaTrasera.faces.push (new THREE.Face3 (i, i + 10, i + 9));
};
estructuraPuertaTrasera.faces.push (new THREE.Face3 (7, 0, 9));
estructuraPuertaTrasera.faces.push (new THREE.Face3 (7, 9, 16));

//COMPUTE
estructuraPuertaTrasera.computeFaceNormals();




/*CREANDO LA PUERTA DELANTERA*/
var estructuraPuertaDelantera = new THREE.Geometry();
```

```
//PUNTOS

//Arreglo de puntos
var x = [  -7 ,  -5 , -4 , 0 ,    0 , -7];
var y = [ 6.5 , 6.5,   6 , 0 , -6.5 , -6.5];
var numPuntos = 6;

//Puntos Frentes (0 - 5)
for (var i = 0; i < numPuntos; i++) {
      estructuraPuertaDelantera.vertices.push  (new  THREE.Vector3  (x[i],  y[i],
0));
};

//Puntos Fondos (6 - 11)
for (var i = 0; i < numPuntos; i++) {
      estructuraPuertaDelantera.vertices.push (new THREE.Vector3 (x[i], y[i], -
z));
};



//CARAS
for (var i = 0; i < 4; i++) {
      //Caras delanteras
      estructuraPuertaDelantera.faces.push (new THREE.Face3 (5, i + 1, i));

      //Caras traseras
      estructuraPuertaDelantera.faces.push (new THREE.Face3 (11, i + 7, i + 6));
};

for (var i = 0; i < 5; i++) {
      //Uniones
      estructuraPuertaDelantera.faces.push (new THREE.Face3 (i, i + 1, i + 7));
      estructuraPuertaDelantera.faces.push (new THREE.Face3 (i, i + 7, i + 6));
};
estructuraPuertaDelantera.faces.push (new THREE.Face3 (5, 0, 6));
estructuraPuertaDelantera.faces.push (new THREE.Face3 (5, 6, 11));
```

```
//COMPUTE
estructuraPuertaDelantera.computeFaceNormals();









/*CREANDO EL PISO*/
var estructuraPiso = new THREE.Geometry();


//PUNTOS

//Arreglo de puntos
var x = [ -19 ,  -12 ,  -12 ,   -5 ,   -5 ,    6 ,    6 ,   13 ,   13 ,   19,
               -18 ,  -11 ,  -11 ,   -6 ,   -6 ,    7 ,    7 ,   12 ,   12 ,
18];
var y = [-6.5 , -6.5 , -4.5 , -4.5 , -6.5 , -6.5 , -4.5 , -4.5 , -6.5 , -6.5,
              -7.5 , -7.5 , -5.5 , -5.5 , -7.5 , -7.5 , -5.5 , -5.5 , -7.5 , -
7.5];
var numPuntos = 20;

//Puntos Frentes (0 - 19)
for (var i = 0; i < numPuntos; i++) {
      estructuraPiso.vertices.push (new THREE.Vector3 (x[i], y[i], 0));
};

//Puntos Fondos (20 - 39)
for (var i = 0; i < numPuntos; i++) {
      estructuraPiso.vertices.push (new THREE.Vector3 (x[i], y[i], -z1));
};


//CARAS
```

```
for (var i = 0; i < 9; i++) {
	//Caras delanteras
	estructuraPiso.faces.push (new THREE.Face3 (i + 10, i + 11, i + 1));
	estructuraPiso.faces.push (new THREE.Face3 (i + 10, i + 1, i));

	//Caras traseras
	estructuraPiso.faces.push (new THREE.Face3 (i + 30, i + 20, i + 21));
	estructuraPiso.faces.push (new THREE.Face3 (i + 30, i + 21, i + 31));

	//Uniones externas
	estructuraPiso.faces.push (new THREE.Face3 (i, i + 1, i + 21));
	estructuraPiso.faces.push (new THREE.Face3 (i, i + 21, i + 20));

	//Uniones internas
	estructuraPiso.faces.push (new THREE.Face3 (i + 30, i + 31, i + 11));
	estructuraPiso.faces.push (new THREE.Face3 (i + 30, i + 11, i + 10));
};

//Uniones finales
//Izquierda
estructuraPiso.faces.push (new THREE.Face3 (30, 10, 0));
estructuraPiso.faces.push (new THREE.Face3 (30,  0, 20));
//Derecha
estructuraPiso.faces.push (new THREE.Face3 (19, 39, 29));
estructuraPiso.faces.push (new THREE.Face3 (19, 29,  9));


//COMPUTE
estructuraPiso.computeFaceNormals();
```

```
/*OBJETOS DE LA ESCENA*/


//Geometrias
var estructuraSeparacion = new THREE.CubeGeometry(1, 13, z);
var estructuraRetrovisor = new THREE.CubeGeometry(6, 1, z2);
var estructuraParabrisas = new THREE.CubeGeometry(6.5, 1, z2);
var estructuraVentanaDelantera = new THREE.CubeGeometry(2.5, 5, 0.6);
var estructuraVentanaTrasera = new THREE.CubeGeometry(5, 5, 0.6);
var estructuraDefensa = new THREE.CubeGeometry(3, 1, z1);
var estructuraFaro = new THREE.SphereGeometry(0.75,50,50, 0 * Math.PI / 180, 180
* Math.PI / 180 );
var estructuraLlanta = new THREE.CylinderGeometry(2.5,2.5,1,20,20);
var estructuraUnion = new THREE.CubeGeometry(0.5, 5.5, 10);
var estructuraManija = new THREE.SphereGeometry(0.5,50,50, 0 * Math.PI / 180, 90
* Math.PI / 180, 0 , Math.PI * 2);



//Cajuela
var soporteFrenteCajuela = new THREE.Mesh( estructuraBaseCajuela, rojo);
var soporteFondoCajuela = new THREE.Mesh( estructuraBaseCajuela, rojo);
var rellenoCajuelaFrente = new THREE.Mesh( estructuraRellenoCajuela, rojo);
var rellenoCajuelaFondo = new THREE.Mesh( estructuraRellenoCajuela, rojo);
var unionCajuela = new THREE.Mesh( estructuraUnion, rojo);
var cajuela = new THREE.Mesh( estructuraCajuela, rojo);
var retrovisor = new THREE.Mesh( estructuraRetrovisor, vidrio);
var defensaTrasera = new THREE.Mesh( estructuraDefensa, aluminio);



//Techo
var baseTechoFrente = new THREE.Mesh( estructuraBaseTecho, rojo);
var baseTechoFondo = new THREE.Mesh( estructuraBaseTecho, rojo);
var techo = new THREE.Mesh( estructuraTecho, rojo);

//Cofre
var baseCofreFrente = new THREE.Mesh( estructuraBaseCofre, rojo);
var baseCofreFondo = new THREE.Mesh( estructuraBaseCofre, rojo);
```

```
var rellenoCofreFrente = new THREE.Mesh( estructuraRellenoCofre, rojo);
var rellenoCofreFondo = new THREE.Mesh( estructuraRellenoCofre, rojo);
var unionCofre = new THREE.Mesh( estructuraUnion, rojo);
var cofreGeom = new THREE.Mesh( estructuraCofre, rojo);
var parabrisas = new THREE.Mesh( estructuraParabrisas, vidrio);
var defensaDelantera = new THREE.Mesh( estructuraDefensa, aluminio);
var faroFrente = new THREE.Mesh( estructuraFaro, foco);
var faroFondo = new THREE.Mesh( estructuraFaro, foco);

//Centro
var puertaTraseraFr = new THREE.Mesh( estructuraPuertaTrasera, rojo);
var puertaTraseraFo = new THREE.Mesh( estructuraPuertaTrasera, rojo);
var puertaDelanteraFr = new THREE.Mesh( estructuraPuertaDelantera, rojo);
var puertaDelanteraFo = new THREE.Mesh( estructuraPuertaDelantera, rojo);

var manijaDelanteraFrente = new THREE.Mesh( estructuraManija, negro);
var manijaDelanteraFondo = new THREE.Mesh( estructuraManija, negro);
var manijaTraseraFrente = new THREE.Mesh( estructuraManija, negro);
var manijaTraseraFondo = new THREE.Mesh( estructuraManija, negro);


var separacionPuertasFrente = new THREE.Mesh( estructuraSeparacion, plastico);
var separacionPuertasFondo = new THREE.Mesh( estructuraSeparacion, plastico);

var ventanaDFrente = new THREE.Mesh( estructuraVentanaDelantera, vidrio);
var ventanaTFrente = new THREE.Mesh( estructuraVentanaTrasera, vidrio);
var ventanaDFondo = new THREE.Mesh( estructuraVentanaDelantera, vidrio);
var ventanaTFondo = new THREE.Mesh( estructuraVentanaTrasera, vidrio);

//Piso
var piso = new THREE.Mesh( estructuraPiso, negro);

//Llantas
var llantaDelanteraFrente = new THREE.Mesh( estructuraLlanta, caucho);
var llantaDelanteraFondo = new THREE.Mesh( estructuraLlanta, caucho);
var llantaTraseraFrente = new THREE.Mesh( estructuraLlanta, caucho);
```

```
var llantaTraseraFondo = new THREE.Mesh( estructuraLlanta, caucho);


/*POSICIONES*/

//Llantas
llantaTraseraFondo.rotation.x = Math.PI / 2;
llantaTraseraFondo.position.set(-8.5,-8, -9.5);

llantaTraseraFrente.rotation.x = Math.PI / 2;
llantaTraseraFrente.position.set(-8.5,-8, -0.5);

llantaDelanteraFondo.position.set(9.5,-8, -9.5);
llantaDelanteraFondo.rotation.x = Math.PI / 2;

llantaDelanteraFrente.position.set(9.5,-8, -0.5);
llantaDelanteraFrente.rotation.x = Math.PI / 2;

//Cajuela
soporteFondoCajuela.position.set(0, 0, -z2 -z);
rellenoCajuelaFondo.position.set(0, 0, -z2 -z);
unionCajuela.position.set(-11.5, -1.75, -5);
cajuela.position.set(-11.5, 1, 0);
defensaTrasera.position.set(-19.5, -7, -5);

//Techo
baseTechoFondo.position.set(0, 0, -z2 -z);
retrovisor.position.set(-10, 3.75, -5);
retrovisor.rotation.z = 65 * Math.PI / 180;
parabrisas.position.set(4.5, 3.5, -5);
parabrisas.rotation.z = -60 * Math.PI / 180;

//Cofre
baseCofreFondo.position.set(0, 0, -z2 -z);
cofre.position.set(7, 1, 0);
rellenoCofreFondo.position.set(0, 0, -z2 -z);
```

```
unionCofre.position.set(6.5, -1.75, -5);
defensaDelantera.position.set(19.5, -7, -5);
faroFrente.position.set(13, -2, -1.5);
faroFrente.rotation.y = Math.PI / 2;
faroFondo.position.set(13, -2, -9);
faroFondo.rotation.y = Math.PI / 2;

//Puertas
puertaTraseraFrente.position.set(0,0,0);
puertaTraseraFondo.position.set(-1.5,0,-z2 -z);
puertaDelanteraFrente.position.set(6, 0,0);
puertaDelanteraFondo.position.set(6, 0,-z2 -z);

manijaTraseraFrente.position.set(-1.75, -1, 0);
manijaTraseraFrente.rotation.z = -Math.PI / 2;

manijaTraseraFondo.position.set(-1.75, -1, -0.5);
manijaTraseraFondo.rotation.z = Math.PI / 2;

manijaDelanteraFrente.position.set(-5.75, -1, 0);
manijaDelanteraFrente.rotation.z = -Math.PI / 2;

manijaDelanteraFondo.position.set(-5.75, -1, -0.5);
manijaDelanteraFondo.rotation.z = Math.PI / 2;

separacionPuertasFrente.position.set(-1.5, 0, -0.25);
separacionPuertasFondo.position.set(-1.5, 0, -z2  - (z * 2) + 0.25);

ventanaTFondo.position.set(-3.5, 3, -0.25);
ventanaDFondo.position.set(-5.25, 3, -0.25);
ventanaTFrente.position.set(-3.5, 3, -0.25);
ventanaDFrente.position.set(-5.25, 3, -0.25);

/*DECLARACION Y CREACION DE GRUPOS*/
var grupoCarro = new THREE.Object3D();
var puertaDelanteraFrente = new THREE.Object3D();
```

```
var puertaDelanteraFondo = new THREE.Object3D();
var puertaTraseraFondo = new THREE.Object3D();
var puertaTraseraFrente = new THREE.Object3D();
var grupoLlantas = new THREE.Object3D();


//Puertas
puertaTraseraFrente.add(ventanaTFrente);
puertaTraseraFrente.add(puertaTraseraFr);
puertaTraseraFrente.add(manijaTraseraFrente);
puertaDelanteraFrente.add(ventanaDFrente);
puertaDelanteraFrente.add(puertaDelanteraFr);
puertaDelanteraFrente.add(manijaDelanteraFrente);
puertaTraseraFondo.add(ventanaTFondo);
puertaTraseraFondo.add(puertaTraseraFo);
puertaTraseraFondo.add(manijaTraseraFondo);
puertaDelanteraFondo.add(ventanaDFondo);
puertaDelanteraFondo.add(puertaDelanteraFo);
puertaDelanteraFondo.add(manijaDelanteraFondo);

//Llantas
grupoLlantas.add(llantaTraseraFondo);
grupoLlantas.add(llantaTraseraFrente);
grupoLlantas.add(llantaDelanteraFondo);
grupoLlantas.add(llantaDelanteraFrente);

//Cajuela
//grupoCarro.add(manijaTraseraFrente);
grupoCarro.add(soporteFrenteCajuela);
grupoCarro.add(soporteFondoCajuela);
grupoCarro.add(rellenoCajuelaFondo);
grupoCarro.add(rellenoCajuelaFrente);
grupoCarro.add(unionCajuela);
grupoCarro.add(cajuela);
grupoCarro.add(defensaTrasera);
```

```
//Techo
grupoCarro.add(baseTechoFrente);
grupoCarro.add(baseTechoFondo);
grupoCarro.add(techo);
grupoCarro.add(retrovisor);

//Cofre
cofre.add(cofreGeom);
cofre.add(faroFrente);
cofre.add(faroFondo);

grupoCarro.add(cofre);
grupoCarro.add(baseCofreFrente);
grupoCarro.add(baseCofreFondo);
grupoCarro.add(rellenoCofreFondo);
grupoCarro.add(rellenoCofreFrente);
grupoCarro.add(unionCofre);
grupoCarro.add(parabrisas);
grupoCarro.add(defensaDelantera);


//Centro
grupoCarro.add(puertaTraseraFondo);
grupoCarro.add(puertaTraseraFrente);
grupoCarro.add(puertaDelanteraFondo);
grupoCarro.add(puertaDelanteraFrente);

grupoCarro.add(separacionPuertasFrente);
grupoCarro.add(separacionPuertasFondo);

//Piso
grupoCarro.add(piso);
grupoCarro.add(grupoLlantas);
```

Castillo.js

```
/*GRUPOS*/

var castillo = new THREE.Object3D();

var torre1Piso = new THREE.Object3D();

var grupoBarda = new THREE.Object3D();

var grupoCastillo = new THREE.Object3D();




/*MATERIALES*/

var teja = new THREE.MeshLambertMaterial ({ map:
THREE.ImageUtils.loadTexture("img/teja.jpg")});

var tabique = new THREE.MeshLambertMaterial ({ map:
THREE.ImageUtils.loadTexture("img/tabique.jpg")});

var dorado = new THREE.MeshLambertMaterial({color: 0xE49E56, side:
THREE.DoubleSide});

var texturaBarda = new THREE.MeshLambertMaterial({map:
THREE.ImageUtils.loadTexture("img/barda1.jpeg"), side: THREE.DoubleSide});

var texturaPuerta = new THREE.MeshLambertMaterial({map:
THREE.ImageUtils.loadTexture("img/puerta.jpg"), side: THREE.DoubleSide});

var material = new THREE.MeshLambertMaterial({color: 0xEEEEEE, side:
THREE.DoubleSide});

var vidrio= new THREE.MeshLambertMaterial ({ map:
THREE.ImageUtils.loadTexture("img/vidrio.jpg")});




/*GEOMETRIAS*/
```

```
//Primer piso
var geomCilindro1Piso = new THREE.CylinderGeometry(72,72,48,20,20);
var geomTecho1Piso = new THREE.CylinderGeometry(56,72,16,20,20);

var geomCilindroTorre1 = new THREE.CylinderGeometry(16,16,80,20,20);
var geomConoTorre1 = new THREE.CylinderGeometry(3,12,30,28,20);

//Segundo piso
var geomTerraza = new THREE.CylinderGeometry(56,56,8,20,20);
var geomCilindro2Piso = new THREE.CylinderGeometry(40,40,24,20,20);
var geomTecho2Piso = new THREE.CylinderGeometry(24,40,16,20,20);

var geomCilindroTorre2 = new THREE.CylinderGeometry(20,20,90,20,20);



//Torre tercer piso
var geomCilindroTorre3 = new THREE.CylinderGeometry(24,24,24,20,20);
var geomConoTorre3 = new THREE.CylinderGeometry(3,16,30,20,20);

var geomBandera = new THREE.CylinderGeometry(1,1,10,20,20);
var geomAstaBandera = new THREE.CylinderGeometry(1,1,10,20,20);

//Barda castillo
var geometriaBarda = new THREE.TorusGeometry(140, 25, 3, 15, 5);
var geometriaTorre = new THREE.CylinderGeometry(20,30,80,15,20,false);
var geometriaTorreTecho = new THREE.CylinderGeometry(22,22,2,15,20,false);

//Base castillo
```

```
var geometriaBaseCastillo = new THREE.CylinderGeometry(110,110,1,8,20,false);


//geometria puerta

var geometriaPuerta = new THREE.Geometry();

//geometria ventana

var geometriaVentana = new THREE.Geometry();

//geometria relieve

var relieveGeom = new THREE.Geometry();


var v1 = new THREE.Vector3(0,0,0);

var v2 = new THREE.Vector3(20,0,0);

var v3 = new THREE.Vector3(0,30,0);

var v4 = new THREE.Vector3(20,30,0);


var v18 = new THREE.Vector3(0,0,-3);

var v19 = new THREE.Vector3(20,0,-3);

var v20 = new THREE.Vector3(0,30,-3);

var v21 = new THREE.Vector3(20,30,-3);


var v5 = new THREE.Vector3(10,30,0);

var v6 = new THREE.Vector3(1,31,0);

var v7 = new THREE.Vector3(2,32,0);

var v8 = new THREE.Vector3(4,33,0);

var v9 = new THREE.Vector3(5,34,0);

var v10 = new THREE.Vector3(7,35,0);

var v11 = new THREE.Vector3(8,36,0);

var v12 = new THREE.Vector3(10,36,0);

var v13 = new THREE.Vector3(11,35,0);
```

```
var v14 = new THREE.Vector3(13,35,0);

var v15 = new THREE.Vector3(14,34,0);

var v14 = new THREE.Vector3(16,34,0);

var v15 = new THREE.Vector3(17,33,0);

var v16 = new THREE.Vector3(19,32,0);

var v17 = new THREE.Vector3(20,31,0);


geometriaPuerta.vertices.push(v1);

geometriaPuerta.vertices.push(v2);

geometriaPuerta.vertices.push(v3);

geometriaPuerta.vertices.push(v4);

geometriaPuerta.vertices.push(v5);

geometriaPuerta.vertices.push(v6);

geometriaPuerta.vertices.push(v7);

geometriaPuerta.vertices.push(v8);

geometriaPuerta.vertices.push(v9);

geometriaPuerta.vertices.push(v10);

geometriaPuerta.vertices.push(v11);

geometriaPuerta.vertices.push(v12);

geometriaPuerta.vertices.push(v13);

geometriaPuerta.vertices.push(v14);

geometriaPuerta.vertices.push(v15);

geometriaPuerta.vertices.push(v16);

geometriaPuerta.vertices.push(v17);

geometriaPuerta.vertices.push(v18);

geometriaPuerta.vertices.push(v19);

geometriaPuerta.vertices.push(v20);

geometriaPuerta.vertices.push(v21);
```

```
geometriaPuerta.faces.push(new THREE.Face3(0,1,2));
geometriaPuerta.faces.push(new THREE.Face3(1,2,3));
geometriaPuerta.faces.push(new THREE.Face3(4,5,6));
geometriaPuerta.faces.push(new THREE.Face3(4,7,8));
geometriaPuerta.faces.push(new THREE.Face3(0,17,19));
geometriaPuerta.faces.push(new THREE.Face3(19,2,0));
geometriaPuerta.faces.push(new THREE.Face3(2,3,19));
geometriaPuerta.faces.push(new THREE.Face3(3,20,19));
geometriaPuerta.faces.push(new THREE.Face3(3,1,20));
geometriaPuerta.faces.push(new THREE.Face3(1,18,20));

geometriaPuerta.faces.push(new THREE.Face3(4,9,10));
geometriaPuerta.faces.push(new THREE.Face3(4,11,12));
geometriaPuerta.faces.push(new THREE.Face3(4,13,14));
geometriaPuerta.faces.push(new THREE.Face3(4,15,16));

//COMPUTE
geometriaPuerta.computeFaceNormals();

var puerta = new THREE.Mesh(geometriaPuerta, material);
puerta.position.set(0, -23, 73);//72
puerta.rotation.y = Math.PI/18;



//GEOMETRIA VENTANA

var vv1 = new THREE.Vector3(0,0,0);     //0
```

```
var vv2 = new THREE.Vector3(10,0,0);//1

var vv3 = new THREE.Vector3(0,10,0);//2

var vv4 = new THREE.Vector3(10,10,0);//3

var vv5 = new THREE.Vector3(0,0,-3);//4

var vv6 = new THREE.Vector3(10,0,-3);//5

var vv7 = new THREE.Vector3(0,10,-3);//6

var vv8 = new THREE.Vector3(10,10,-3);//7


geometriaVentana.vertices.push(vv1);

geometriaVentana.vertices.push(vv2);

geometriaVentana.vertices.push(vv3);

geometriaVentana.vertices.push(vv4);

geometriaVentana.vertices.push(vv5);

geometriaVentana.vertices.push(vv6);

geometriaVentana.vertices.push(vv7);

geometriaVentana.vertices.push(vv8);


geometriaVentana.faces.push(new THREE.Face3(0,1,2));

geometriaVentana.faces.push(new THREE.Face3(1,2,3));


geometriaVentana.faces.push(new THREE.Face3(0,1,4));

geometriaVentana.faces.push(new THREE.Face3(1,5,4));

geometriaVentana.faces.push(new THREE.Face3(1,7,3));

geometriaVentana.faces.push(new THREE.Face3(1,5,7));

geometriaVentana.faces.push(new THREE.Face3(0,4,6));

geometriaVentana.faces.push(new THREE.Face3(0,6,2));

geometriaVentana.faces.push(new THREE.Face3(2,3,6));

geometriaVentana.faces.push(new THREE.Face3(3,7,6));
```

```
//COMPUTE
geometriaVentana.computeFaceNormals();

var ventana = new THREE.Mesh(geometriaVentana, material);
ventana.position.set(-70, 30, 42);//41
ventana.rotation.y = Math.PI/18;

var ventana2 = new THREE.Mesh(geometriaVentana, material);
ventana2.position.set(70, 30, 42);
ventana2.rotation.y = Math.PI/9;

var ventana3 = new THREE.Mesh(geometriaVentana, material);
ventana3.position.set(-11, 50, 33);
ventana3.rotation.y=Math.PI/-18;

var ventana4 = new THREE.Mesh(geometriaVentana, material);
ventana4.position.set(11, 50, 33);
ventana4.rotation.y=Math.PI/9;

var ventana5 = new THREE.Mesh(geometriaVentana, material);
ventana5.position.set(0, 90, 19);
ventana5.rotation.y=Math.PI/18;

//geometria de las torres cubos


//radio de las circunferencias del relieveGeom
```

```
var radioCircunferenciaInterna = 10;

var radioCircunferenciaMedia = 11;

var radioCircunferenciaExterna = 10;

//numero de circulos por relieveGeom

var numCirculos = 3.0;

//numero de vertices por circulo del relieveGeom

var numVertices = 62.0;

var angulo = 360;


 //obtenemos los vertices para cada circunferencia del relieveGeom cara frontal

            for(i = 0; i<numCirculos; i++){

                switch(i){

                    case 0:     //circunferencia interna

                        radioTemp =
radioCircunferenciaInterna+(radioCircunferenciaInterna*i);

                        for(j = 0; j < numVertices; j++){   //para cada vertice
saco x,y,z

                            x =
radioTemp*Math.cos((j*(angulo/numVertices))*(Math.PI/180.0));

                            y =
radioTemp*Math.sin((j*(angulo/numVertices))*Math.PI/180.0);

                            z = 2;

                            //creamos el vector

                            var vector = new THREE.Vector3(x,y,z);

                            //agregamos los vecotres a la geometria

                            relieveGeom.vertices.push(vector);


                        }

                        break;
```

```
                    case 1:     //circunferencia media
                        radioTemp =
radioCircunferenciaMedia+(radioCircunferenciaMedia*i);
                        for(j = 0; j < numVertices; j++){   //para cada vertice
saco x,y,z
                            x =
radioTemp*Math.cos((j*(angulo/numVertices))*(Math.PI/180.0));
                            y =
radioTemp*Math.sin((j*(angulo/numVertices))*Math.PI/180.0);
                            z = 2;
                            //creamos el vector
                            var vector = new THREE.Vector3(x,y,z);
                            //agregamos los vecotres a la geometria
                            relieveGeom.vertices.push(vector);

                        }
                        break;
                    case 2:     //circunferencia externa
                        radioTemp =
radioCircunferenciaExterna+(radioCircunferenciaExterna*i);
                        for(j = 0; j < numVertices; j++){   //para cada vertice
saco x,y,z
                            x =
radioTemp*Math.cos((j*(angulo/numVertices))*(Math.PI/180.0));
                            y =
radioTemp*Math.sin((j*(angulo/numVertices))*(Math.PI/180.0));
                            z = 2;
                            //creamos el vector
                            var vector = new THREE.Vector3(x,y,z);
                            //agregamos los vecotres a la geometria
                            relieveGeom.vertices.push(vector);
```

```
                        }
                        break;
                }

            }

            //obtengo los vertices para cada circunferencia del relieveGeom cara
tracera
            for(i = 0; i<numCirculos; i++){
                switch(i){
                    case 0:     //circunferencia interna
                        radioTemp =
radioCircunferenciaInterna+(radioCircunferenciaInterna*i);
                        for(j = 0; j < numVertices; j++){   //para cada vertice
saco su x,y,z
                            x =
radioTemp*Math.cos((j*(angulo/numVertices))*(Math.PI/180.0));
                            y =
radioTemp*Math.sin((j*(angulo/numVertices))*Math.PI/180.0);
                            z = 0;
                            //creamos el vector
                            var vector = new THREE.Vector3(x,y,z);
                            //agregamos los vecotres a la geometria
                            relieveGeom.vertices.push(vector);

                        }
                        break;
                    case 1:     //circunferencia media
```

```
                        radioTemp =
radioCircunferenciaMedia+(radioCircunferenciaMedia*i);
                        for(j = 0; j < numVertices; j++){   //para cada vertice
saco su x,y,z
                            x =
radioTemp*Math.cos((j*(angulo/numVertices))*(Math.PI/180.0));
                            y =
radioTemp*Math.sin((j*(angulo/numVertices))*Math.PI/180.0);
                            z = 0;
                            //creamos el vector
                            var vector = new THREE.Vector3(x,y,z);
                            //agregamos los vecotres a la geometria
                            relieveGeom.vertices.push(vector);

                        }
                        break;
                    case 2:     //circunferencia externa
                        radioTemp =
radioCircunferenciaExterna+(radioCircunferenciaExterna*i);
                        for(j = 0; j < numVertices; j++){   //para cada vertice
saco su x,y,z
                            x =
radioTemp*Math.cos((j*(angulo/numVertices))*(Math.PI/180.0));
                            y =
radioTemp*Math.sin((j*(angulo/numVertices))*Math.PI/180.0);
                            z = 0;
                            //creamos el vector
                            var vector = new THREE.Vector3(x,y,z);
                            //agregamos los vecotres a la geometria
                            relieveGeom.vertices.push(vector);
```

```
                    }
                    break;
                }

            }
            //caras del frente
            for(i=0; i < numCirculos-1; i++)
            {
                for(j=0; j < numVertices; j++) // obtenemos todos los vertices
                {
                    if(i==numCirculos-2)    //caras de los dientes
                        j++;
                    if(j==numVertices-1)    //si es la ultima cara pinta tomando el primer vertice y el ultimo para hacer las dos caras
                    {
                        relieveGeom.faces.push( new THREE.Face3( j+(numVertices*i), j+(numVertices*i)+numVertices, j+(numVertices*i)+1 ));
                        relieveGeom.faces.push( new THREE.Face3( j+(numVertices*i), j+(numVertices*i)+1, numVertices*i ));

                    }
                    else
                    {
                        relieveGeom.faces.push( new THREE.Face3( j+(numVertices*i), j+(numVertices*i)+numVertices, j+(numVertices*i)+numVertices+1 ));
                        relieveGeom.faces.push( new THREE.Face3( j+(numVertices*i), j+(numVertices*i)+numVertices+1, j+(numVertices*i)+1 ));

                    }
```

```
            }
        }
        //caras atras
        for(i=0; i < numCirculos-1; i++)
        {
            for(j=0; j < numVertices; j++)
            {
                if(i==numCirculos-2) //cara de los dientes
                    j++;

                if(j==numVertices-1)    //si es la ultima cara toma los
vertices primeros y ultimos para hacer la cara
                {
                    relieveGeom.faces.push( new THREE.Face3(
j+(numVertices*i)+1 +(numCirculos*numVertices) , j+(numVertices*i)+numVertices
+(numCirculos*numVertices), j+(numVertices*i)+(numCirculos*numVertices) ));
                    relieveGeom.faces.push( new THREE.Face3(
j+(numVertices*i)+1 +(numCirculos*numVertices),
j+(numVertices*i)+(numCirculos*numVertices) , (numVertices*i)
+(numCirculos*numVertices)));

                }
                else
                {
                    relieveGeom.faces.push( new THREE.Face3(
j+(numVertices*i)+numVertices+1 +(numCirculos*numVertices),
j+(numVertices*i)+numVertices+(numCirculos*numVertices),
j+(numVertices*i)+(numCirculos*numVertices)   ));
```

```
                        relieveGeom.faces.push( new THREE.Face3(
j+(numVertices*i)+numVertices+1 +(numCirculos*numVertices), j+(numVertices*i)
+(numCirculos*numVertices), j+(numVertices*i)+1 +(numCirculos*numVertices) ));
                    }


                }
            }

            //crea las caras frontales del interior del relieveGeom
            for(j=0; j < numVertices; j++)
            {
                if(j==numVertices-1)    //si es el ultimo vertice
                    {
                        relieveGeom.faces.push( new THREE.Face3(
(numCirculos*numVertices) , j+(numCirculos*numVertices), j ));
                        relieveGeom.faces.push( new THREE.Face3(
(numCirculos*numVertices) , j , 0));


                    }
                    else
                    {
                        relieveGeom.faces.push( new THREE.Face3( j+1
+(numCirculos*numVertices), j+(numCirculos*numVertices), j ));


                        relieveGeom.faces.push( new THREE.Face3(  j+1
+(numCirculos*numVertices), j , j +1  ));
                    }
            }


            //crea las caras frontales de afuera
```

```
            for(j=0; j < numVertices; j++)
            {
                if( (j%2)!=0 )  //es el diente
                {   if(j==numVertices-1)    //si es el ultimo vertice
                    {
                        //cara superior
                        relieveGeom.faces.push( new THREE.Face3(
j+numVertices*(numCirculos-1)+(numCirculos*numVertices) ,
                        numVertices*(numCirculos-1)+(numCirculos*numVertices),
numVertices*(numCirculos-1)  ));

                        relieveGeom.faces.push( new THREE.Face3(
j+numVertices*(numCirculos-1)+(numCirculos*numVertices) ,
                        numVertices*(numCirculos-1), numVertices*numCirculos-1
));

                        //lateral de los dientes
                        relieveGeom.faces.push( new
THREE.Face3(j+numVertices+(numCirculos*numVertices) ,
                        j+numVertices*(numCirculos-1)+(numCirculos*numVertices),
j+numVertices*(numCirculos-1) ));

                        relieveGeom.faces.push( new THREE.Face3(
j+numVertices+(numCirculos*numVertices),
                        j+numVertices*(numCirculos-1), j+numVertices ));

                    }
                    else
                    {
                        //superior
```

```
                        relieveGeom.faces.push( new THREE.Face3(
j+numVertices*(numCirculos-1)+(numCirculos*numVertices) ,
                        j+numVertices*(numCirculos-
1)+(numCirculos*numVertices)+1, j+numVertices*(numCirculos-1)+1  ));


                        relieveGeom.faces.push( new THREE.Face3(
j+numVertices*(numCirculos-1)+(numCirculos*numVertices) ,
                        j+numVertices*(numCirculos-1)+1,
j+numVertices*(numCirculos-1)  ));




                        //lateral de los dientes
                        relieveGeom.faces.push( new
THREE.Face3(j+numVertices+(numCirculos*numVertices) ,
                        j+numVertices*(numCirculos-1)+(numCirculos*numVertices),
j+numVertices*(numCirculos-1) ));


                        relieveGeom.faces.push( new THREE.Face3(
j+numVertices+(numCirculos*numVertices),
                        j+numVertices*(numCirculos-1), j+numVertices ));
                    }
                }
                else    //es la parte donde no esta el diente
                {
                    //frontales
                    relieveGeom.faces.push( new
THREE.Face3(j+numVertices+(numCirculos*numVertices) ,
                        j+numVertices+(numCirculos*numVertices)+1, j+numVertices
+1  ));
```

```
                    relieveGeom.faces.push( new THREE.Face3(
j+numVertices+(numCirculos*numVertices),
                        j+numVertices +1 , j+numVertices  ));

                    //laterales
                    relieveGeom.faces.push( new
THREE.Face3(j+numVertices*(numCirculos-1)+(numCirculos*numVertices) ,
                    j+numVertices+(numCirculos*numVertices), j+numVertices ));

                    relieveGeom.faces.push( new THREE.Face3(
j+numVertices*(numCirculos-1)+(numCirculos*numVertices),
                    j+numVertices , j+numVertices*(numCirculos-1) ));
                }
            }

            relieveGeom.computeFaceNormals();

/*OBJETOS DE LA ESCENA*/
//var tam = new THREE.Mesh(geomCilindroTam, tabique);
//Barda
var barda = new THREE.Mesh(geometriaBarda,texturaBarda);
```

```
var torre1 = new THREE.Mesh(geometriaTorre,texturaBarda);

var torre2 = new THREE.Mesh(geometriaTorre,texturaBarda);

var techotorre1 = new THREE.Mesh(geometriaTorreTecho,texturaBarda);

var techotorre2 = new THREE.Mesh(geometriaTorreTecho,texturaBarda);

var relieve = new THREE.Mesh(relieveGeom,material);

var relieve2 = new THREE.Mesh(relieveGeom,material);

var relieve3 = new THREE.Mesh(relieveGeom,material);


//base castillo

var baseCastillo = new THREE.Mesh(geometriaBaseCastillo,material);



//Primer piso

var cilindro1Piso = new THREE.Mesh(geomCilindro1Piso, tabique);

var techo1Piso = new THREE.Mesh(geomTecho1Piso, teja);


var astaBandera = new THREE.Mesh(geomAstaBandera, dorado);


//Torres

var baseTorre1 = new THREE.Mesh(geomCilindroTorre1, tabique);

var conoTorre1 = new THREE.Mesh(geomConoTorre1, teja);



//Segundo piso

var terraza = new THREE.Mesh(geomTerraza, tabique);

var cilindro2Piso = new THREE.Mesh(geomCilindro2Piso, tabique);

var techo2Piso = new THREE.Mesh(geomTecho2Piso, teja);
```

```
var baseTorre2 = new THREE.Mesh(geomCilindroTorre1, tabique);

var conoTorre2 = new THREE.Mesh(geomConoTorre1, teja);

var astaBandera2 = new THREE.Mesh(geomAstaBandera, dorado);


var baseTorre2Derecha = new THREE.Mesh(geomCilindroTorre2, tabique);



//Torre tercer piso

var baseTorre3 = new THREE.Mesh(geomCilindroTorre3, tabique);

var conoTorre3 = new THREE.Mesh(geomConoTorre3, teja);

var astaBandera3 = new THREE.Mesh(geomAstaBandera, dorado);




//Primer piso

techo1Piso.position.set(0,32,0);

baseTorre1.position.set(0,16,0);

conoTorre1.position.set(0,67,0);

astaBandera.position.set(0,87,0);


torre1Piso.add(baseTorre1);

torre1Piso.add(conoTorre1);

torre1Piso.add(astaBandera);

var torres = [];

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

       torres[i] = torre1Piso.clone();

};
```

```
//Segundo Piso
terraza.position.set(0, 44, 0);
cilindro2Piso.position.set(0, 60, -6.4);
techo2Piso.position.set(0,80, -6.4);

baseTorre2.position.set(-36, 50, -24);
conoTorre2.position.set(-36, 102, -24);
astaBandera.position.set(-36, 122, -24);

baseTorre2Derecha.position.set(35, 67, -24);

//Torre tercer piso
baseTorre3.position.set(0, 100, -6.4);
conoTorre3.position.set(0, 127, -6.4);
astaBandera3.position.set(0, 147, -6.4);









/*POSICIONES*/
var radio = 72;

var angulo = 20 * Math.PI / 180;//30°
```

```
torres[0].position.set(Math.cos(angulo) * radio, 0, Math.sin(angulo) * radio);

var angulo = 160 * Math.PI / 180;//30°
torres[1].position.set(Math.cos(angulo) * radio, 0, Math.sin(angulo) * radio);

var angulo = 210 * Math.PI / 180;//30°
torres[2].position.set(Math.cos(angulo) * radio, 0, Math.sin(angulo) * radio);

var angulo = 330 * Math.PI / 180;//30°
torres[3].position.set(Math.cos(angulo) * radio, 0, Math.sin(angulo) * radio);

barda.rotation.x = Math.PI/2;
barda.rotation.z = Math.PI/2;
barda.position.y = -24;

torre1.position.y = -14;
torre1.position.z = 140;

relieve.rotation.x = Math.PI/2;
relieve.position.y = 27;
relieve.position.z = 122;
relieve.position.x = -70;

torre2.position.y = -14;
torre2.position.z = 35;
torre2.position.x =133;

relieve2.rotation.x = Math.PI/2;
```

```
relieve2.position.y = 27;

relieve2.position.z = 96;

relieve2.position.x = 98;


relieve3.rotation.x = Math.PI/2;

relieve3.position.y = 110;

relieve3.position.z = -24;

relieve3.position.x = 35;


techotorre1.position.z = 140;

techotorre1.position.y = 28;


techotorre2.position.y = 28;

techotorre2.position.z = 35;

techotorre2.position.x = 133;


baseCastillo.position.y = -23;


/*var radio = 72;

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


	torres[i].position.set(Math.cos(angulo) * 92, 0, Math.sin(angulo) * 92);


	angulo += 90 * Math.PI / 180;//90°

};*/



/*Añadiendo objetos a grupos*/
```

```
//Barda
grupoBarda.add(barda);
grupoBarda.add(torre1);
grupoBarda.add(torre2);
grupoBarda.add(techotorre1);
grupoBarda.add(techotorre2);
grupoBarda.rotation.y = Math.PI/-6;
//relieve de las torres
castillo.add(relieve);
castillo.add(relieve2);
castillo.add(relieve3);

/*ANADIENDO OBJETOS A GRUPO CASTILLO*/

//Castillo
castillo.add(baseCastillo);
castillo.add(grupoBarda);
//gerometrias creadas
castillo.add(puerta);
castillo.add(ventana);
castillo.add(ventana2);
castillo.add(ventana3);
castillo.add(ventana4);
castillo.add(ventana5);



//Primer piso
```

```
castillo.add(cilindro1Piso);

castillo.add(techo1Piso);


//Segundo piso

castillo.add(terraza);

castillo.add(cilindro2Piso);

castillo.add(techo2Piso);

castillo.add(baseTorre2);

castillo.add(conoTorre2);

castillo.add(astaBandera2);

castillo.add(baseTorre2Derecha);


//Torre tercer piso

castillo.add(baseTorre3);

castillo.add(conoTorre3);

castillo.add(astaBandera3);


//Torres

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

       castillo.add(torres[i]);
```

index.js

```
<html>


    <head>

        <title> .:: Proyecto Final ::.</title>


             <style>
```

```
      canvas { width: 100%; height: 100% }
  </style>

</head>



<body>

  <script type='text/javascript' src='three.min.js'></script>
  <script src="Automovil.js"></script>
  <script src="Castillo.js"></script>

       <script>
              window.addEventListener('keydown',doKeyDown,true);

              /*POSICIONES INICIALES DE LA CAMARA*/
            var zPos = 150;
            var xPos = -70.0;
            var ypos = 0.0;
            var camRotation = 0;

            /*POSICIONES INICIALES DEL AUTOMOVIL*/
            var zPosCarro = 0;
            var xPosCarro = -150;
            var carroRotation = 0;
            var llantasRotation = 0;
            var llantasRotationZ = 0;
            var cajuelaRotation = 0;
```

```
var cofreRotation = 0;
var puertasTraserasPos = -1.5;
var puertasFrenteRotation = 0;
var puertasFondoRotation = 0;



/*MANEJO DE TECLAS PARA LA NAVEGACION*/
function doKeyDown(evt){

      switch (evt.keyCode) {

            /*NAVEGACION*/
            case 38:  /* Up arrow was pressed */
                  var dx = 5 * Math.sin(camRotation);
                  var dz = 5 * Math.cos(camRotation);
                  zPos -= dz;
                  xPos -= dx;
                  break;

            case 40:  /* Down arrow was pressed */
                  var dx = 5 * Math.sin(camRotation);
                  var dz = 5 * Math.cos(camRotation);
                  zPos += dz;
                  xPos += dx;
                  break;

            case 37:  /* Left arrow was pressed */
```

```
                camRotation += (5 * Math.PI) / 180;
                break;

            case 39:  /* Right arrow was pressed */
                camRotation -= (5 * Math.PI) / 180;
                break;

            case 87:  /* W was pressed */
                camera.position.y += 3;
                break;

            case 83:  /* S was pressed */
                camera.position.y -= 3;
                break;

            case 65:  /* A was pressed */
                var dx = 5 * Math.sin(camRotation + Math.PI /
2 );
                var dz = 5 * Math.cos(camRotation + Math.PI /
2 );
                zPos -= dz;
                xPos -= dx;
                break;

            case 68:  /* D was pressed */
                var dx = 5 * Math.sin(camRotation - Math.PI /
2);
                var dz = 5 * Math.cos(camRotation- Math.PI /
2);
```

```
                zPos -= dz;
                xPos -= dx;
                break;

            case 84: /*T was pressed -> Auto avanza*/
                var dx = 5 * Math.sin(carroRotation - Math.PI
/ 2 );
                var dz = 5 * Math.cos(carroRotation - Math.PI
/ 2 );
                zPosCarro -= dz;
                xPosCarro -= dx;
                llantasRotation -= 0.5;
                llantasRotationZ = 0.0;
                break;

            case 71: /*G was pressed -> Auto retrocede*/
                var dx = 5 * Math.sin(carroRotation + Math.PI
/ 2 );
                var dz = 5 * Math.cos(carroRotation + Math.PI
/ 2 );
                zPosCarro -= dz;
                xPosCarro -= dx;
                llantasRotation += 0.5;
                llantasRotationZ = 0.0;
                break;


            case 72: /*H was pressed -> Auto rota a la
derecha*/
                carroRotation -= (5 * Math.PI) / 180;
```

```
                        if (llantasRotationZ + 0.003 < 0.4)
                              llantasRotationZ += 0.003;
                        break;

                  case 70: /*F was pressed -> Auto rota a la
izquierda*/
                        carroRotation += (5 * Math.PI) / 180;

                        if (llantasRotationZ - 0.003 > -0.4)
                              llantasRotationZ -= 0.003;
                        break;

                  case 66: /*B was pressed -> Cierra cajuela*/
                        if (cajuelaRotation + 0.03 <= 0)
                              cajuelaRotation += 0.03;
                        break;



                  case 86: /*V was pressed -> Abre cajuela*/
                        if (cajuelaRotation - 0.03 >= -1.1)
                              cajuelaRotation -= 0.03;
                        break;



                  case 82: /*R was pressed -> Cierra cofre*/
                        if (cofreRotation - 0.03 >= 0)
```

```
            cofreRotation -= 0.03;
      break;



case 89: /*Y was pressed -> Abre cofre*/
      if (cofreRotation + 0.03 <= 1)
            cofreRotation += 0.03;
      break;




case 74: /*J was pressed -> Cierra puertas*/
      if (puertasFrenteRotation - 0.03 >= 0) {
            puertasFrenteRotation -= 0.03;
            puertasFondoRotation += 0.03;
      }
      console.log(puertasFrenteRotation);
      break;

case 85: /*U was pressed -> Abre puertas*/
      if (puertasFrenteRotation + 0.03 <= 0.75) {
            puertasFrenteRotation += 0.03;
            puertasFondoRotation -= 0.03;
      }
      console.log(puertasFrenteRotation);

      break;
```

```
            }//Fin switch


        }//Fin funcion



        /*TRES COMPONENTES BASICOS DE LA ESCENA*/

        var scene = new THREE.Scene();              //Escena

        var camera = new THREE.PerspectiveCamera(75,
window.innerWidth/window.innerHeight, 0.1, 1000);  //Camara

        var renderer = new THREE.WebGLRenderer();   //Render







        /*DECLARACION Y CREACION DE GRUPOS*/

        var grupoCarrusel = new THREE.Object3D();

        var grupoRuedaFortuna = new THREE.Object3D();

        var grupoRueda = new THREE.Object3D();

        var grupoTazasLocas = new THREE.Object3D();


        //Grupos que se clonaran para contener caballos, tazas, platos
y canastas individuales

        var grupoCaballo = new THREE.Object3D();

        var grupoTaza = new THREE.Object3D();

        var grupoPlato = new THREE.Object3D();

        var grupoCanasta = new THREE.Object3D();
```

```
                /*MATERIALES*/

                var techoCarrusel = new THREE.MeshLambertMaterial({map:
THREE.ImageUtils.loadTexture("img/techoCarrusel.png")});

                var texturaTaza = new THREE.MeshLambertMaterial({map:
THREE.ImageUtils.loadTexture("img/texturaTaza.jpeg")});

                var texturaPasto = new THREE.MeshLambertMaterial({map:
THREE.ImageUtils.loadTexture("img/texturaPasto.jpg"), side: THREE.DoubleSide});

                 var texturaCielo = new THREE.MeshLambertMaterial({map:
THREE.ImageUtils.loadTexture("img/texturaCielo.jpg"), side: THREE.DoubleSide});


                var material = new THREE.MeshLambertMaterial({color: 0xEEEEEE,
side: THREE.DoubleSide});

                var materialRojo = new THREE.MeshLambertMaterial({color:
0xFF3333, side: THREE.DoubleSide});

                var materialDorado = new THREE.MeshLambertMaterial({color:
0xE49E56, side: THREE.DoubleSide});

                var materialCafe = new THREE.MeshLambertMaterial({color:
0x994C00, side: THREE.DoubleSide});

                var materialBlanco = new THREE.MeshLambertMaterial({color:
0xFFFFFF, side: THREE.DoubleSide});

                var materialAzul = new THREE.MeshLambertMaterial({color:
0x0080FF, side: THREE.DoubleSide});

                var materialMorado = new THREE.MeshLambertMaterial({color:
0xB266FF, side: THREE.DoubleSide});

                var materialVerde = new THREE.MeshLambertMaterial({color:
0x00FF00, side: THREE.DoubleSide});
```

```
                /****CARRUSEL****/


                /*GEOMETRIAS*/

                //Geometrias del techo

                var geometriaEsfera = new THREE.SphereGeometry(1,20,20);

                var geometriaCilindroPunta = new
THREE.CylinderGeometry(0.2,0.2,4,20,20,false);

                var GeometriaCono = new
THREE.CylinderGeometry(0,23,4,20,20,false);

                var geometriaCilindroTecho = new
THREE.CylinderGeometry(23,23,1,20,20,false);


                //Geometrias del centro

                var geometriaCilindroCentro = new
THREE.CylinderGeometry(1.5,1.5,18,20,20,false);

                var geometriaCilindroCaballos = new
THREE.CylinderGeometry(0.2,0.2,18,20,20,false);


                //Geometria de los caballos
```

```
                var geometriaTorsoCaballo = new THREE.CubeGeometry(3, 1.5,
1.5);

                var geometriaCuelloCaballo = new THREE.CubeGeometry(2, 1, 1.5);

                var geometriaCabezaCaballo = new THREE.CubeGeometry(1.5, 1,
1.5);

                var geometriaOrejaCaballo = new THREE.CylinderGeometry(0, 0.2,
0.9,20,20,false);

                var geometriaPataCaballo = new THREE.CubeGeometry(0.5, 2, 0.5);

                var geometriaBaseColaCaballo = new THREE.CubeGeometry(0.5, 2,
0.5);

                var geometriaFinColaCaballo = new THREE.CubeGeometry(1, 0.5,
0.5);



                //Geometrias base

                var geometriaCilindroEscalon1 = new
THREE.CylinderGeometry(23.6,23.6,.5,20,20);

                var geometriaCilindroEscalon2 = new
THREE.CylinderGeometry(23.8,23.8,.2,20,20);




                /*OBJETOS DE LA ESCENA*/

                //Techo

                var esfera = new THREE.Mesh(geometriaEsfera,material);

                var punta = new THREE.Mesh(geometriaCilindroPunta,material);

                var cono = new THREE.Mesh(GeometriaCono,techoCarrusel);

                var cilindroTecho = new
THREE.Mesh(geometriaCilindroTecho,material);



                //Centro
```

```
                var cilindroCentro = new
THREE.Mesh(geometriaCilindroCentro,materialRojo);


                var tubosCaballos = [];

                for (var i = 0; i < 16; i++) {

                      tubosCaballos[i] = new
THREE.Mesh(geometriaCilindroCaballos,material);

                }




                //Caballo

                var torsoCaballo = new
THREE.Mesh(geometriaTorsoCaballo,materialCafe);

                var cuelloCaballo = new
THREE.Mesh(geometriaCuelloCaballo,materialCafe);

                var cabezaCaballo = new
THREE.Mesh(geometriaCabezaCaballo,materialCafe);

                var orejaFCaballo = new
THREE.Mesh(geometriaOrejaCaballo,materialCafe);

                var orejaACaballo = new
THREE.Mesh(geometriaOrejaCaballo,materialCafe);

                var baseColaCaballo = new
THREE.Mesh(geometriaBaseColaCaballo,materialCafe);

                var finColaCaballo = new
THREE.Mesh(geometriaFinColaCaballo,materialCafe);

                var patasCaballo = [];

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

                      patasCaballo[i] = new
THREE.Mesh(geometriaPataCaballo,materialCafe);


                //Piso
```

```
                var base = new
THREE.Mesh(geometriaCilindroTecho,materialDorado);

                var escalon1 = new
THREE.Mesh(geometriaCilindroEscalon1,materialRojo);

                var escalon2 = new
THREE.Mesh(geometriaCilindroEscalon2,materialDorado);









                /*POSICIONES DE OBJETOS*/

                //Techo

                esfera.position.y = 3;

                punta.position.y = 0;

                cono.position.y = -2;

                cilindroTecho.position.y = -4.5;


                //Piso

                base.position.y = -23;

                escalon1.position.y = -23;

                escalon2.position.y = -23;


                //Tubo Centro

                cilindroCentro.position.y = -13.5;


                //Tubos externos caballos

                var angulo = 0;

                for (var i = 0; i < 8; i++) {

                      tubosCaballos[i].position.y = -13.5;
```

```
                        tubosCaballos[i].position.x = Math.sin(angulo) * 20;
                        tubosCaballos[i].position.z = Math.cos(angulo) * 20;

                        angulo += (Math.PI * 2) / 8;
                  };

                  //Tubos internos caballos
                  var angulo = 0;
                  for (var i = 8; i < 16; i++) {
                        tubosCaballos[i].position.y = -13.5;
                        tubosCaballos[i].position.x = Math.sin(angulo +
0.392699082) * 12;
                        tubosCaballos[i].position.z = Math.cos(angulo +
0.392699082) * 12;

                        angulo += (Math.PI * 2) / 8;
                  };

                  //Caballo
                  torsoCaballo.position.x = -0.5;

                  cuelloCaballo.position.x = torsoCaballo.position.x + 1.5;
                  cuelloCaballo.position.y = 0.8;
                  cuelloCaballo.position.z = torsoCaballo.position.z;
                  cuelloCaballo.rotation.z = 1.04719755; //60 grados

                  cabezaCaballo.position.x = cuelloCaballo.position.x + 0.8;
                  cabezaCaballo.position.y = cuelloCaballo.position.y + 0.8;
```

```
orejaACaballo.position.x = cabezaCaballo.position.x - 0.4;
orejaACaballo.position.y = cabezaCaballo.position.y + 0.95;
orejaACaballo.position.z = cabezaCaballo.position.z - 0.5;

orejaFCaballo.position.x = cabezaCaballo.position.x - 0.4;
orejaFCaballo.position.y = cabezaCaballo.position.y + 0.95;
orejaFCaballo.position.z = cabezaCaballo.position.z + 0.5;

baseColaCaballo.position.x = torsoCaballo.position.x - 1.75;
baseColaCaballo.position.y = torsoCaballo.position.y - 1;
baseColaCaballo.position.z = torsoCaballo.position.z;

finColaCaballo.position.x = baseColaCaballo.position.x - 0.75;
finColaCaballo.position.y = baseColaCaballo.position.y - 0.75;
finColaCaballo.position.z = torsoCaballo.position.z;

for (var i = 0; i < 4; i++)
      patasCaballo[i].position.y = torsoCaballo.position.y -
1.75;

//Patas delanteras
patasCaballo[0].position.z = torsoCaballo.position.z + 0.5;
patasCaballo[1].position.z = torsoCaballo.position.z - 0.5;
patasCaballo[0].position.x = torsoCaballo.position.x + 1.25;
patasCaballo[1].position.x = torsoCaballo.position.x + 1.25;

//Patas traseras
patasCaballo[2].position.z = torsoCaballo.position.z + 0.5;
```

```
patasCaballo[3].position.z = torsoCaballo.position.z - 0.5;

patasCaballo[2].position.x = torsoCaballo.position.x - 1.25;

patasCaballo[3].position.x = torsoCaballo.position.x - 1.25;


//Se agregan los objetos torso, cuello, cabeza, patas y cola al
grupo caballo

grupoCaballo.add(torsoCaballo);

grupoCaballo.add(cuelloCaballo);

grupoCaballo.add(cabezaCaballo);

grupoCaballo.add(orejaACaballo);

grupoCaballo.add(orejaFCaballo);

grupoCaballo.add(baseColaCaballo);

grupoCaballo.add(finColaCaballo);

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

      grupoCaballo.add(patasCaballo[i]);



//Clonando el grupo caballo y estableciendo su posicion

var angulo = 0;

var grupoCaballos = [];

for (var i = 0; i < 8; i++) {

      grupoCaballos[i] = grupoCaballo.clone();


      var x = Math.sin(angulo) * 20;

      var z = Math.cos(angulo) * 20;


      grupoCaballos[i].applyMatrix( new
THREE.Matrix4().makeTranslation( x , - 11 - Math.abs(Math.sin(angulo) * 6), z));

      grupoCaballos[i].rotation.y = angulo;
```

```
		angulo += Math.PI * 2 / 8;


	};


	var angulo = 0;
	for (var i = 8; i < 16; i++) {
		grupoCaballos[i] = grupoCaballo.clone();


		var x = Math.sin(angulo + 0.392699082) * 12;
		var z = Math.cos(angulo + 0.392699082) * 12;


		grupoCaballos[i].applyMatrix( new
THREE.Matrix4().makeTranslation( x, - 11 - Math.abs(Math.sin(angulo) * 6), z));
		grupoCaballos[i].rotation.y = angulo + 0.392699082;


		angulo += (Math.PI * 2 / 8);


	};



	//Se agregan los objetos al grupo Carrusel
	grupoCarrusel.add(esfera);
	grupoCarrusel.add(punta);
	grupoCarrusel.add(cono);
	grupoCarrusel.add(cilindroTecho);
	grupoCarrusel.add(cilindroCentro);
	grupoCarrusel.add(base);
```

```
grupoCarrusel.add(escalon1);

grupoCarrusel.add(escalon2);

for (var i = 0; i < 16; i++) {

      grupoCarrusel.add(tubosCaballos[i]);


      grupoCarrusel.add(grupoCaballos[i]);

}


//Posicion del grupo Carrusel

grupoCarrusel.position.x = -40;

grupoCarrusel.position.z = 20;


//Anadiendo el carrusel a la escena

scene.add(grupoCarrusel);
```

```
/****RUEDA DE LA FORTUNA****/
```

```
                //GEOMETRIAS

                //Base

                var geometriaBase1 = new THREE.CubeGeometry(54, 70, 1);

                var geometriaBase2 = new THREE.CubeGeometry(40, 56, 4);


                //Rueda

                var geometriaRueda = new THREE.TorusGeometry(40, 0.2, 16,100);

                var geometriaRuedaUnion = new THREE.CylinderGeometry(0.1, 0.1,
80, 20, 20,false);

                var geometriaUnionRuedas = new THREE.CylinderGeometry(0.1, 0.1,
5.6, 20, 20,false);


                //Soporte

                var geometriaSoporte = new THREE.CylinderGeometry(0.5, 2, 52,
20, 20, false);

                var geometriaunionSoporte = new THREE.CylinderGeometry(1, 1,
10, 0, 20,false);


                //Canasta

                var geometriaCanasta = new THREE.CylinderGeometry(2.7, 1.5, 3,
20, 20, true,2,2);

                var geometriaCanastaI = new THREE.CylinderGeometry(2.35, 1.15,
3, 20, 20, true, 0, Math.PI * 1.5);

                var geometriaTechoCanasta = new THREE.CylinderGeometry(1, 2.7,
0.5, 20, 20,false);

                var geometriaPisoCanasta = new THREE.CylinderGeometry(1.5, 1.5,
0.1, 20, 20,false);

                var geometriaUnionTecho = new THREE.CylinderGeometry(0.1, 0.1,
2.55, 20, 20,false);
```

```
                var geometriaUnionCanastas = new
THREE.RingGeometry(2.35,2.7,50);

                var geometriaUnionCanastasRueda = new
THREE.CylinderGeometry(0.1, 0.1, 0.95, 20, 20,false);




                //OBJETOS DE LA ESCENA

                //Base

                var base1 = new THREE.Mesh(geometriaBase1,materialBlanco);

                var base2 = new THREE.Mesh(geometriaBase2,materialBlanco);


                //Rueda

                var rueda1 = new THREE.Mesh(geometriaRueda,materialAzul);

                var rueda2 = new THREE.Mesh(geometriaRueda,materialAzul);


                //Union de el centro

                var uniones = [];

                for(var i = 0; i < 20; i++)

                      uniones[i] = new
THREE.Mesh(geometriaRuedaUnion,materialRojo);


                //Union entre ruedas para soportar canastas

                var unionRuedas = [];

                for(var i = 0; i < 20; i++)

                      unionRuedas[i] = new
THREE.Mesh(geometriaUnionRuedas,material);


                //Soportes
```

```
                    var soporte1 = new THREE.Mesh(geometriaSoporte,materialBlanco);
                    var soporte2 = new THREE.Mesh(geometriaSoporte,materialBlanco);
                    var soporte3 = new THREE.Mesh(geometriaSoporte,materialBlanco);
                    var soporte4 = new THREE.Mesh(geometriaSoporte,materialBlanco);

                    var unionSoporte = new
THREE.Mesh(geometriaunionSoporte,materialBlanco);

                    //Canastas
                    canasta = new THREE.Mesh(geometriaCanasta, materialDorado);
                    canastaI = new THREE.Mesh(geometriaCanastaI, materialBlanco);
                    techo = new THREE.Mesh(geometriaTechoCanasta, materialDorado);
                    piso = new THREE.Mesh(geometriaPisoCanasta, materialDorado);
                    unionTecho = new THREE.Mesh(geometriaUnionTecho,
materialDorado);
                    unionCanasta = new THREE.Mesh(geometriaUnionCanastas,
materialDorado);
                    unionCanastaRueda = new THREE.Mesh(geometriaUnionCanastasRueda,
materialDorado);




                    //POSICIONES DE LOS OBJETOS

                    //Centro de la rueda
                    grupoRueda.applyMatrix( new THREE.Matrix4().makeTranslation( 0,
28, 0) );
```

```
//Base
base1.position.y = -23;
base1.rotation.x =Math.PI/2;
base1.rotation.z = Math.PI/2;

base2.position.y= -22;
base2.rotation.x = Math.PI/2;
base2.rotation.z = Math.PI/2;

//Rueda
rueda1.position.z = 2.8;
rueda2.position.z = -2.8;

soporte1.position.y = 3.75;
soporte1.position.z = 6;
soporte1.rotation.x = -0.08;
soporte1.position.x = 10;
soporte1.rotation.z = -5.9;

soporte2.position.y = 3.75;
soporte2.position.z = -6;
soporte2.rotation.x = 0.08;
soporte2.position.x = 10;
soporte2.rotation.z = -5.9;

soporte3.position.y = 3.75;
soporte3.position.z = 6;
soporte3.rotation.x = -0.08;
```

```
soporte3.position.x = -10;
soporte3.rotation.z = 5.9;

soporte4.position.y = 3.75;
soporte4.position.z = -6;
soporte4.rotation.x = 0.08;
soporte4.position.x = -10;
soporte4.rotation.z = 5.9;

unionSoporte.rotation.x = Math.PI/2;
unionSoporte.position.y= 27.9;

var i;
var angulo = 0;
for(var i = 0; i < 10; i++){
      uniones[i].position.z = 2.8;
      uniones[i].rotation.z = angulo;
      angulo += Math.PI/10;
}
angulo = 0;
for(var i = 10; i < 20; i++){
      uniones[i].position.z = -2.8;
      uniones[i].rotation.z = angulo;
      angulo += Math.PI/10;
}

angulo = 0;
for (var i = 0; i < 20; i++) {
```

```
      unionRuedas[i].position.x = Math.cos(angulo) * 40;
      unionRuedas[i].position.y = Math.sin(angulo) * 40;
      unionRuedas[i].rotation.x = Math.PI / 2;

      angulo += Math.PI/10;
}


//Posicion de las canastas
canasta.position.y = - 5.5;
canastaI.position.y = - 5.5;

unionCanasta.position.y = - 4;
unionCanasta.rotation.x = Math.PI / 2;

techo.position.y = - 1.2;
piso.position.y = - 7;

unionTecho.position.y = - 2.725;
unionTecho.position.z = - 2.55;

unionCanastaRueda.position.y = -0.475;



//AGREGANDO OBJETOS AL GRUPO CANASTA
grupoCanasta.add(canasta);
grupoCanasta.add(canastaI);
```

```
grupoCanasta.add(unionCanasta);

grupoCanasta.add(techo);

grupoCanasta.add(piso);

grupoCanasta.add(unionTecho);

grupoCanasta.add(unionCanastaRueda);



//Clonando el grupo canasta y estableciendo sus posiciones

var angulo = 0;

var grupoCanastas = [];

for (var i = 0; i < 20; i++) {

      grupoCanastas[i] = grupoCanasta.clone();


      grupoCanastas[i].applyMatrix( new
THREE.Matrix4().makeTranslation( Math.sin(angulo) * 40 , Math.cos(angulo) * 40,
0));


      angulo += Math.PI / 10;


      grupoRueda.add(grupoCanastas[i]);

};



//AGREGANDO OBJETOS AL GRUPO RUEDA

grupoRueda.add(rueda1);

grupoRueda.add(rueda2);

for(var i = 0; i<32; i++){

      grupoRueda.add(uniones[i]);

}
```

```
for(var i = 0; i < 20; i++)
      grupoRueda.add(unionRuedas[i]);



//AGREGANDO OBJETOS AL GRUPO RUEDA FORTUNA
grupoRuedaFortuna.add(grupoRueda);
grupoRuedaFortuna.add(base1);
grupoRuedaFortuna.add(base2);
grupoRuedaFortuna.add(soporte1);
grupoRuedaFortuna.add(soporte2);
grupoRuedaFortuna.add(soporte3);
grupoRuedaFortuna.add(soporte4);
grupoRuedaFortuna.add(unionSoporte);



grupoRuedaFortuna.position.x = 35;
grupoRuedaFortuna.position.z = -35;
grupoRuedaFortuna.position.y = 1;



scene.add(grupoRuedaFortuna);
```

```
                /*TAZAS LOCAS*/


                //GEOMETRIAS
                var geometriaCilindroBaseTaza = new
THREE.CylinderGeometry(22,22,1,20,20,false);
                var geometriaCilindroE1 = new
THREE.CylinderGeometry(22.3,22.3,.5,20,20);
                var geometriaCilindroE2 = new
THREE.CylinderGeometry(22.5,22.5,.2,20,20);


                var geometriaPlato = new THREE.CylinderGeometry(2,1,0.5,20,20,
true);
                var geometriaBasePlato = new
THREE.CylinderGeometry(1,1,0.1,20,20, false);


                var geometriaTaza = new THREE.CylinderGeometry(2,1,2,20,20,
true);
                var geometriaTaza2 = new
THREE.CylinderGeometry(1.9,0.9,2,20,20, true);
                var geometriaUnionTaza = new THREE.RingGeometry(1.9,2,32);
                var geometriaAza = new THREE.TorusGeometry( 0.6, 0.2, 16, 100,
Math.PI + 0.3);
```

```
                var geometriaEsferaCentro = new THREE.SphereGeometry(3,20,20);




                //OBJETOS DE LA ESCENA

                var baseTazas = new
THREE.Mesh(geometriaCilindroBaseTaza,materialDorado);

                var escalon1Tazas = new
THREE.Mesh(geometriaCilindroE1,materialMorado);

                var escalon2Tazas = new
THREE.Mesh(geometriaCilindroE2,materialDorado);


                var plato = new THREE.Mesh(geometriaPlato,material);

                var basePlato = new THREE.Mesh(geometriaBasePlato,material);


                var taza = new THREE.Mesh(geometriaTaza,texturaTaza);

                var tazaI = new THREE.Mesh(geometriaTaza2,materialBlanco);

                var unionTaza = new THREE.Mesh(geometriaUnionTaza,texturaTaza);

                var aza = new THREE.Mesh( geometriaAza, material);


                var centro = new THREE.Mesh(geometriaCilindroCentro,
materialBlanco)

                var esferaCentro = new THREE.Mesh(geometriaEsferaCentro,
materialMorado)




                //POSICIONES DE LOS OBJETOS
```

```
centro.position.y = 9;

esferaCentro.position.y = 18;


plato.position.set(0, 0.8, 0);

basePlato.position.set(0, 0.5, 0);


taza.position.set(0, 1.7, 0);

tazaI.position.set(0, 1.7, 0);


aza.position.set(1.61, 1.8, 0);

aza.rotation.z = -Math.PI / 4 - 1.4;


unionTaza.position.set(0,2.7,0);

unionTaza.rotation.x = -Math.PI / 2;



//ANADIENDO OBJETOS AL GRUPO TAZA

grupoTaza.add(plato);

grupoTaza.add(basePlato);

grupoTaza.add(taza);

grupoTaza.add(tazaI);

grupoTaza.add(unionTaza);

grupoTaza.add(aza);



//CLONANDO EL GRUPO TAZA Y ESTABLECIENDO SU CENTRO

var angulo = 0;

var grupoTazas = [];
```

```
                for (var i = 0; i < 8 ; i++) {
                      grupoTazas[i] = grupoTaza.clone();

                      grupoTazas[i].applyMatrix( new
THREE.Matrix4().makeTranslation(Math.sin(angulo) * 16, 0, Math.cos(angulo) * 16)
);

                      angulo += Math.PI / 4;
                };

                var angulo = 0;
                for (var i = 8; i < 16 ; i++) {

                      grupoTazas[i] = grupoTaza.clone();

                      grupoTazas[i].applyMatrix(new
THREE.Matrix4().makeTranslation(Math.sin(angulo + 0.392699082) * 10, 0,
Math.cos(angulo + 0.392699082) * 10) );
                      angulo += Math.PI / 4;
                };




                //Anadiendo objetos al grupo general de las tazas locas
            grupoTazasLocas.add(baseTazas);
                grupoTazasLocas.add(escalon1Tazas);
                grupoTazasLocas.add(escalon2Tazas);
                grupoTazasLocas.add(centro);
```

```
                grupoTazasLocas.add(esferaCentro);



                for(i = 0; i < 16; i++)
                      grupoTazasLocas.add(grupoTazas[i]);



                //Posicion del grupo
                grupoTazasLocas.position.x = 40;
                grupoTazasLocas.position.z = 40;
                grupoTazasLocas.position.y = -23;



                //Agregando el grupo a la escena
                scene.add(grupoTazasLocas);









                /*CARRO*/
                scene.add(grupoCarro);
                puertaTraseraFrente.applyMatrix( new
THREE.Matrix4().makeTranslation( -10.5, 0,0));
```

```
                    /*CASTILLO*/
                    scene.add(castillo);
                    castillo.position.set(-200,0,-100);









                    /*CREANDO EL PASTO*/
                    var geometriaPisoFeria = new THREE.PlaneGeometry(2000,2000);
                    var pisoFeria = new
THREE.Mesh(geometriaPisoFeria,materialVerde);
                    pisoFeria.rotation.x = Math.PI/-2;
                    pisoFeria.position.y= -23.6;
                    scene.add(pisoFeria);



                    /*CREANDO EL CIELO*/
                    var geometriaCielo = new THREE.SphereGeometry(500,200,200);
                    var cieloFeria = new THREE.Mesh(geometriaCielo,texturaCielo);


                    scene.add(cieloFeria);
```

```
/*CREANDO LUCES*/

var pointLight = new THREE.PointLight( 0xFFFFFF );

var pointLight2 = new THREE.PointLight( 0xFFFFFF );

var pointLight3 = new THREE.PointLight( 0xFFFFFF );

var pointLight4 = new THREE.PointLight( 0xFFFFFF );

var pointLight5 = new THREE.PointLight( 0xFFFFFF );


pointLight.position.x = 0;

pointLight.position.y = 10;

pointLight.position.z = 00;


scene.add(pointLight);



pointLight2.position.x = 10;

pointLight2.position.y = -50;

pointLight2.position.z = -130;


scene.add(pointLight2);
```

```
pointLight3.position.x = 0;

pointLight3.position.y = 0;

pointLight3.position.z = 130;

scene.add(pointLight3);


scene.add( new THREE.AmbientLight( 0x212223));


var light = new THREE.DirectionalLight(0xffffff, 0.5);

//light.position.z = 0;

//light.position.y = 100;

//light.position.x = 100;


var lightRueda = new THREE.DirectionalLight(0xffffff, 0);

lightRueda.position.z = 200;

lightRueda.position.y = 100;

lightRueda.position.x = 200;


//scene.add(light);

scene.add(lightRueda);



renderer.shadowMapEnabled = true;

renderer.shadowMapSoft = false;


renderer.shadowCameraNear = 250;

renderer.shadowCameraFar = camera.far;
```

```
renderer.shadowCameraFov = 500;


renderer.shadowMapBias = 0.0039;

renderer.shadowMapDarkness = 0.5;

renderer.shadowMapWidth = 1024;

renderer.shadowMapHeight = 1024;


//light.castShadow = true;

lightRueda.castShadow = true;




//Carro

piso.castShadow = true;

llantaDelanteraFrente.castShadow = true;

llantaDelanteraFondo.castShadow = true;

llantaTraseraFrente.castShadow = true;

llantaTraseraFondo.castShadow = true;

grupoCarro.castShadow = true;



//Rueda

soporte1.castShadow = true;

soporte2.castShadow = true;

soporte3.castShadow = true;

soporte4.castShadow = true;

rueda1.castShadow = true;

rueda2.castShadow = true;
```

```
canasta.castShadow = true;
grupoRueda.castShadow = true;

base2.receiveShadow = true;
base1.receiveShadow = true;



//Carrusel
cilindroTecho.castShadow = true;
cilindroCentro.castShadow = true;
esfera.castShadow = true;
punta.castShadow = true;
cono.castShadow = true;
tubosCaballos.castShadow = true;
torsoCaballo.castShadow = true;
cuelloCaballo.castShadow = true;
for(i = 0; i < 20; i++)
      grupoCanastas[i].castShadow = true;
for(i = 0; i < 16; i++){
      grupoCaballos[i].castShadow = true;
      grupoTazas[i].castShadow = true;
      grupoPlato.castShadow = true;
}
base.receiveShadow = true;
escalon2.receiveShadow = true;
escalon1.receiveShadow = true;
```

```
//Tazas
centro.castShadow = true;
esferaCentro.castShadow = true;

baseTazas.receiveShadow = true;
escalon1Tazas.receiveShadow = true;
escalon2Tazas.receiveShadow = true;


//Castillo
castillo.castShadow = true;
cilindro1Piso.castShadow = true;
techo1Piso.castShadow = true;
baseTorre1.castShadow = true;
conoTorre1.castShadow = true;

cilindro2Piso.castShadow = true;
techo2Piso.castShadow = true;


pisoFeria.receiveShadow = true;
```

```
                /*RENDER*/
                renderer.setClearColor(0x000000, 1);
                //tamaño del render
                renderer.setSize(window.innerWidth, window.innerHeight);
                //envio el render al html
                document.body.appendChild(renderer.domElement);
                var render = function () { //creo la variable render y le
asigno una funcion
                 requestAnimationFrame(render);  // pido el render
                      camera.position.z = zPos;
                      camera.position.x = xPos;
                      camera.rotation.y = camRotation;

                      //ROTACION DEL CARRUSEL
                      grupoCarrusel.rotation.y += 0.01;

                      //Movimiento de los caballos
                      var angulo = [];
                      var rotacion = 0;
                      for(i = 0; i < 8; i++){
                            var angulo = Math.sin(grupoCaballos[i].rotation.y +
grupoCarrusel.rotation.y);
                            grupoCaballos[i].position.y =  - 11 -
Math.abs(angulo * 6);
                      }

                      for(i = 8; i < 16; i++){
```

```
                              var angulo = Math.sin(grupoCaballos[i].rotation.y +
grupoCarrusel.rotation.y);
                              grupoCaballos[i].position.y =  - 11 -
Math.abs(angulo * 6);
                        }




                        //ROTACION DE LA RUEDA
                        grupoRueda.rotation.z += 0.003;


                        //Rotacion individual de cada canasta
                        for(i = 0; i < 20; i++)
                              grupoCanastas[i].rotation.z -=0.003;




                        //ROTACION DE LAS TAZAS LOCAS


                        //rotacion individual de cada taza
                        for(i = 0; i < 16; i++)
                              grupoTazas[i].rotation.y +=0.07;


                        //rotacion de la base
                        grupoTazasLocas.rotation.y -= 0.02;
```

```
/*MOVIMIENTO DEL CARRO*/
grupoCarro.position.x = xPosCarro;
grupoCarro.position.z = zPosCarro;
llantaDelanteraFrente.rotation.y = llantasRotation;
llantaDelanteraFrente.rotation.z =  llantasRotationZ;
llantaTraseraFrente.rotation.y = llantasRotation;
llantaTraseraFrente.rotation.z = llantasRotationZ;
llantaDelanteraFondo.rotation.y = llantasRotation;
llantaDelanteraFondo.rotation.z = llantasRotationZ;
llantaTraseraFondo.rotation.y = llantasRotation;
llantaTraseraFondo.rotation.z = llantasRotationZ;



//Carro
grupoCarro.position.y = -13;
grupoCarro.position.z = 95;
grupoCarro.rotation.y = carroRotation;

//Cajuela
cajuela.rotation.z = cajuelaRotation;

//Cofre
cofre.rotation.z = cofreRotation;
cofre.position.y = 1;

//Puertas
puertaTraseraFrente.rotation.y = puertasFrenteRotation;
puertaTraseraFrente.position.x = -2;
```

```
                        puertaDelanteraFrente.rotation.y = puertasFrenteRotation;
                        puertaDelanteraFrente.position.x = 6;

                        puertaTraseraFondo.rotation.y = puertasFondoRotation;
                        puertaTraseraFondo.position.x = -2;
                        puertaTraseraFondo.position.z = -9.5;

                        puertaDelanteraFondo.rotation.y = puertasFondoRotation;
                        puertaDelanteraFondo.position.x = 6;
                        puertaDelanteraFondo.position.z = -9.5;

                        renderer.render(scene, camera);     //actualiza la escena
y camara
                    };
                    render();
                </script>
        </body>
</html>
```