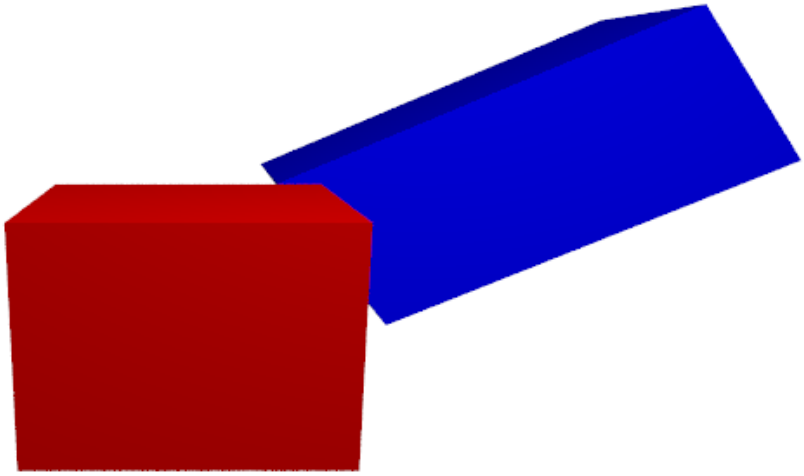
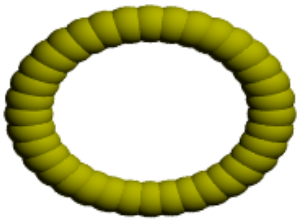
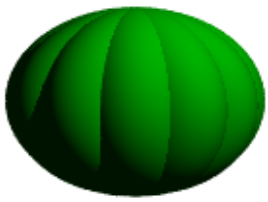
Computer Graphics - Babylon.js Sheet

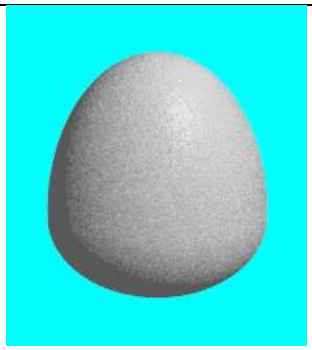
1-Write a Babylon.js program to implement red cube and then apply necessary transformations to get the blue cube.



https://playground.babylonjs.com/#3EXFJZ

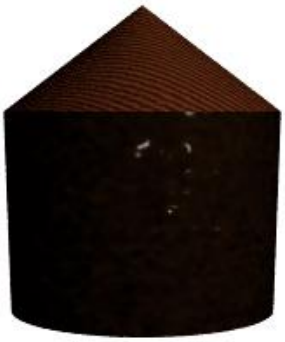
2-Write a Babylon.js program to implement the following shapes **as closely as you can**:

 (a) https://playground.babylonjs.com/#1AFWZ1#13	 (b) https://playground.babylonjs.com/#NZE6UA#2
 (c) https://playground.babylonjs.com/#FXL70W#1	 (d) https://playground.babylonjs.com/#NZE6UA#3
 (e) https://playground.babylonjs.com/#WVK46M	 (f) https://playground.babylonjs.com/#JCV4PM

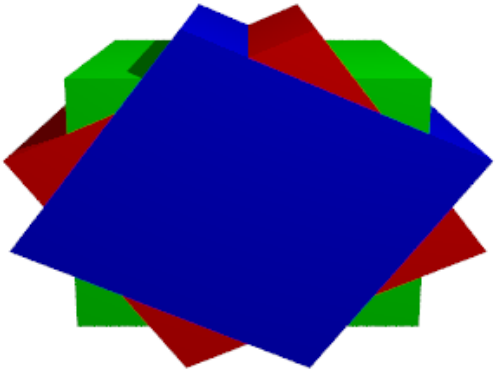


(g)
https://playground.babylonjs.com/#LU6GGT

(h)
https://playground.babylonjs.com/#VYZEEQ#162



(i)
https://playground.babylonjs.com/#0RTVPI#1



(j)
https://playground.babylonjs.com/#310BSG#5

https://playground.babylonjs.com/#YHKFIC#1
(k)

3- Use Babylon.js do draw the following shapes each in a separate scene with appropriate light and camera position:

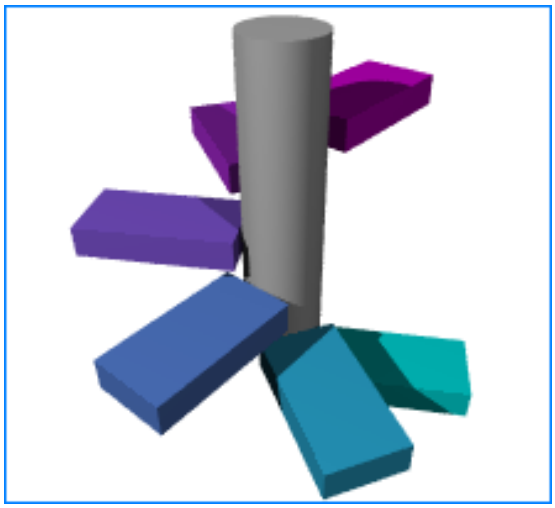
a) A Set of 10 spheres of equal radius of 5 units and color blue having centers on the line segment from point A = (-200, 0, 200) to point B = (200, 0, -200)

b) A set of 20 spheres of equal radius of 5 units and color red having centers on a circle C in the x-y plane and center at the origin with radius 200 units. The first of the 20 spheres should be at point (200, 0, 0) and the last one at the same point while the others cover the circle C uniformly.
c) A cube with side length of 100 units, colored red, with center at the origin, and sides parallel to the main x- y- z- axis. In addition, draw a set of 8 right cones with color blue each with base radius of 10 units, height of 20 units and base center at each corner of the cube. The main axis of each cone should be such that it passes through the origin and one corner of the cube while pointing outwards relative to the cube.
d) Repeat problem c but this time while the cones are pointing inwards relative to the cube. Use joint to make the shapes into a single compound shape.
e) Repeat problem d while using difference to cut the cones from the cube.

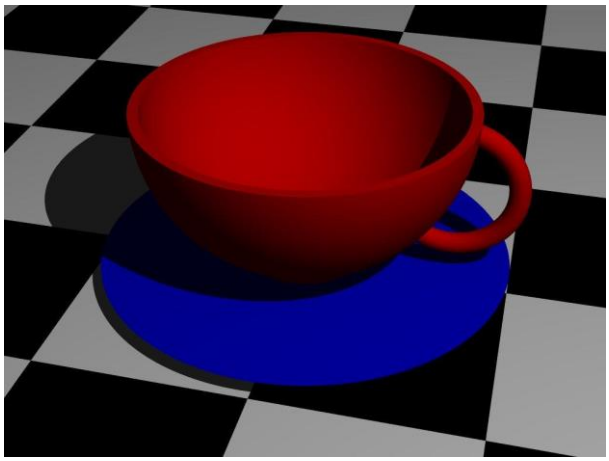
4- Given the following Babylon.js scene write the code to output the associated clock animation. The final animation should have a rotation of the clock hands to illustrate it works like a fast clock. You can use whatever colors you want.

https://playground.babylonjs.com/#1AFWZ1#16

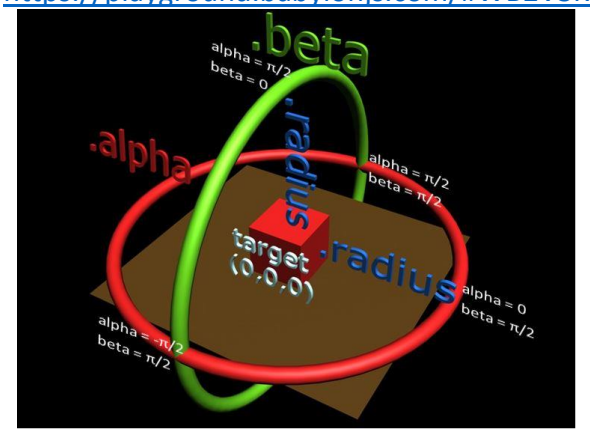
5- Write and execute Babylon.js scene code to output each of the shown images as close as you can. You can use whatever colors you want.



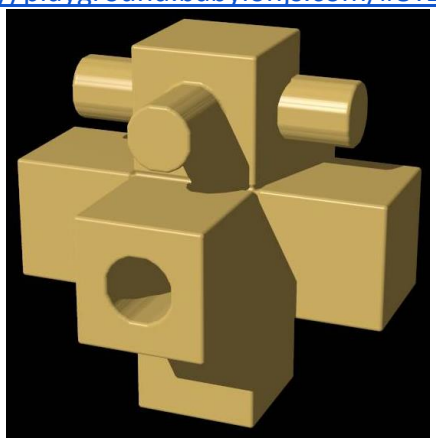
(a)
https://playground.babylonjs.com/#WB2V3K



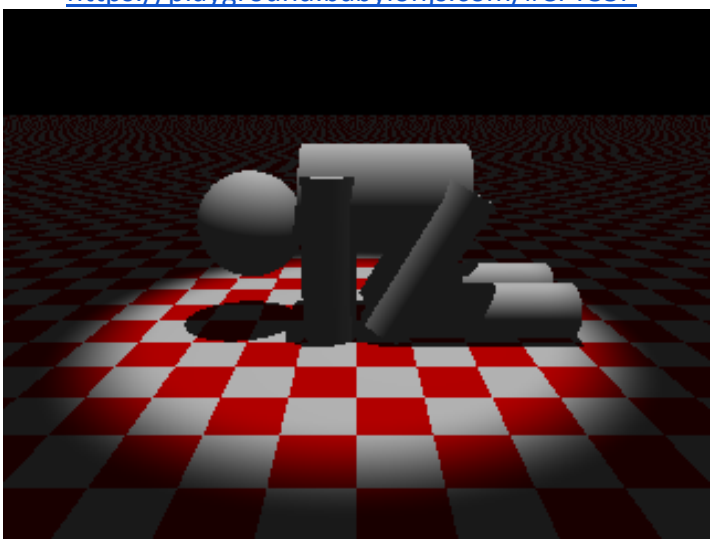
(b)
https://playground.babylonjs.com/#8YZIZX#4



(c)
https://playground.babylonjs.com/#CFT35P



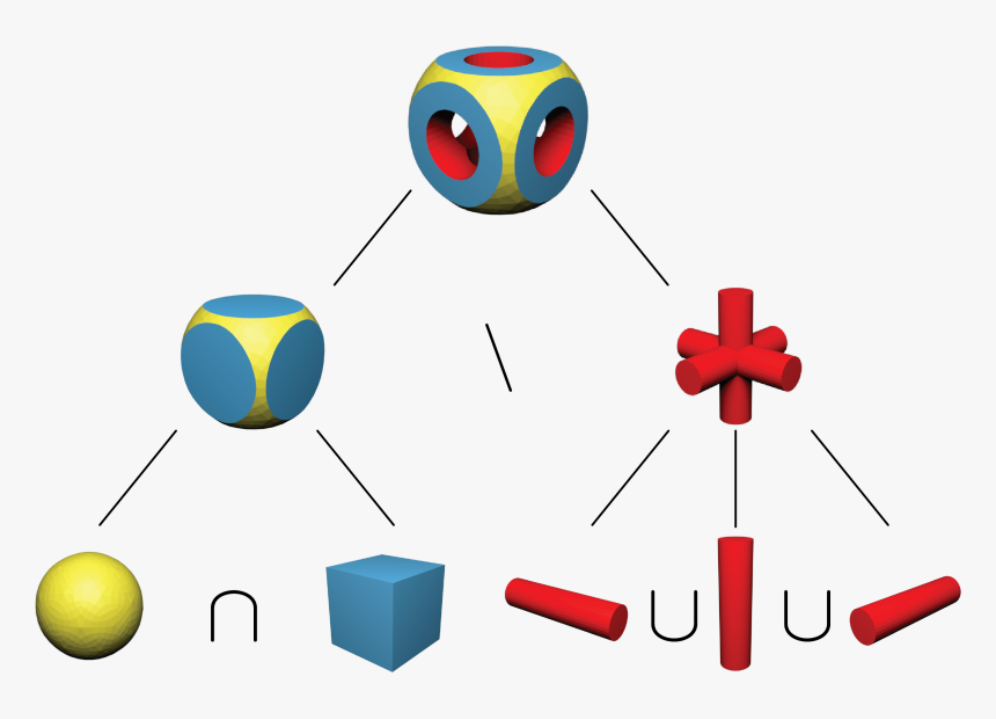
(d)
https://playground.babylonjs.com/#YTCCTT#3



(e)
https://playground.babylonjs.com/#YT01AZ#2

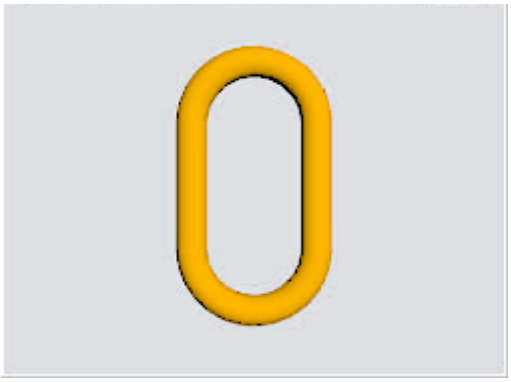
(f)
https://playground.babylonjs.com/#GUEYF2#10

6- Use Babylon.js to implement the following scene as best as you can. Use any colors you want.



https://playground.babylonjs.com/#Y8027Q#12

7- Use Babylon.js to implement the following two scenes as best as you can. Use any colors you want.



(a)

(b)

https://playground.babylonjs.com/#Y8027Q#7

```
var createScene = function () {
    // This creates a basic Babylon Scene object (non-mesh)
    var scene = new BABYLON.Scene(engine);

    // This creates and positions a free camera (non-mesh)
    // var camera = new BABYLON.FreeCamera("camera1", new BABYLON.Vector3(0, 0, 0), scene);

    // // This targets the camera to scene origin
    // camera.setTarget(BABYLON.Vector3.Zero());

    // // This attaches the camera to the canvas
    // camera.attachControl(canvas, true);


    // Creates, angles, distances and targets the camera
    var camera = new BABYLON.ArcRotateCamera("Camera", 0, 0, 10, new BABYLON.Vector3(0, 0, 0)
, scene);

    // This positions the camera
    camera.setPosition(new BABYLON.Vector3(0, 0, -5));

    // This attaches the camera to the canvas
    camera.attachControl(canvas, true);

    // This creates a light, aiming 0,1,0 - to the sky (non-mesh)
    var light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0, 1, 0), scene);

    // Default intensity is 1. Let's dim the light a small amount
    light.intensity = 0.7;

////////////////////////////////////
    // // Our built-in 'sphere' shape.
    // // var sphere = BABYLON.MeshBuilder.CreateSphere("sphere", {diameter: 2, segments: 32}
, scene);
    // // Move the sphere upward 1/2 its height
    // //sphere.position.y = 1;

    // var sphere = BABYLON.MeshBuilder.CreateSphere("sphere", {diameterX: 0.5, diameterY:1.5
, diameterZ:1}, scene);
    // sphere.position.y = 1;
    // sphere.scaling = new BABYLON.Vector3(5, 0.8, 2);

    // var cylinder = BABYLON.MeshBuilder.CreateCylinder("cylinder", {height: 2, diameter: 1}
, scene);
    // cylinder.position.y = 1;
    // cylinder.position.z = 0;
    // cylinder.rotation.x = Math.PI/2;
    // cylinder.scaling = new BABYLON.Vector3(1, 1, 1);
    // // Our built-in 'ground' shape.
```

```
    // let cntr1 = sphere.getTotalVertices();
    // let ver1 = [];
    // for (let i = 0; i < cntr1; i++)
    // {
    //     ver1.push(1,1,0,1);
    // }
    // sphere.setVerticesData(BABYLON.VertexBuffer.ColorKind, ver1);

    // let cntr2 = cylinder.getTotalVertices();
    // let ver2 = []
    // for (let i = 0; i < cntr2;i++)
    // {
    //     ver2.push(1,0,0,1);
    // }
    // cylinder.setVerticesData(BABYLON.VertexBuffer.ColorKind, ver2);


    // let cyl = BABYLON.CSG.FromMesh(cylinder);
    // let sph = BABYLON.CSG.FromMesh(sphere);

    // let sub = sph.subtract(cyl);

    // let newmesh = sub.toMesh("newmesh", null, scene);


    // cylinder.visibility = false;
    // sphere.visibility = false;

//////////////////////////////////////////

    // var tor = BABYLON.MeshBuilder.CreateTorus("tours1", {diameter:9, tessellation:64, thic
kness:0.3}, scene);
    // tor.rotation.x = Math.PI/2;
    // tor.position = new BABYLON.Vector3(3.5,0,0);

    // var cylinder = BABYLON.MeshBuilder.CreateCylinder("cylinder", {height: 1, diameter: 0.
3}, scene);
    // cylinder.rotation.z = Math.PI/2;


    // let cly = [];
    // for (let i =0; i < 12; i++)
    // {
    //     cly[i] = cylinder.createInstance("ins"+i);
    //     cly[i].rotateAround(new BABYLON.Vector3(3.5,0,0), new BABYLON.Vector3(0,0,1), i *
2 * Math.PI / 12);
    // }

    // var sphere = BABYLON.MeshBuilder.CreateSphere("sphere", {diameter: 0.5}, scene);
    // sphere.position.x = 3.5;
```

```
    // var cylinder1 = BABYLON.MeshBuilder.CreateCylinder("cylinder1", {height: 4, diameter:
0.3}, scene);
    // cylinder1.rotation.z = Math.PI/4;
    // cylinder1.position = new BABYLON.Vector3(4.816, -1.316,0);

    // var cylinder2 = BABYLON.MeshBuilder.CreateCylinder("cylinder2", {height:2, diameter: 0
.3}, scene);
    // cylinder2.rotation.z = Math.PI/1.6;
    // cylinder2.position = new BABYLON.Vector3(4.493, 0.477,0);


//////////////////////////////////////

    // var mt = new BABYLON.StandardMaterial("myMaterial", scene);
    // mt.diffuseColor = new BABYLON.Color3(1, 1, 0);

    // var mt1 = new BABYLON.StandardMaterial("myMaterial", scene);
    // mt1.diffuseColor = new BABYLON.Color3(1, 0, 0);

    // var cylinder = BABYLON.MeshBuilder.CreateCylinder("cylinder", {height: 1, diameter: 0.
3}, scene);

    // var box = BABYLON.MeshBuilder.CreateBox("box", {size:1, height:0.1, width:0.2, depth:0
.6}, scene);
    // box.position = new BABYLON.Vector3(0.10, -0.44, -0.19);
    // box.rotation = new BABYLON.Vector3(0, Math.PI/0.17, 0);

    // var box1 = BABYLON.MeshBuilder.CreateBox("box1", {size:1, height:0.1, width:0.2, depth
:0.6}, scene);
    // box1.position = new BABYLON.Vector3(-0.010, -0.24, -0.19);
    // box1.rotation = new BABYLON.Vector3(0, Math.PI/0.25, 0);
    // box1.material = mt1;

    // var box2 = BABYLON.MeshBuilder.CreateBox("box2", {size:1, height:0.1, width:0.2, depth
:0.6}, scene);
    // box2.position = new BABYLON.Vector3(-0.10, -0.02, -0.17);
    // box2.rotation = new BABYLON.Vector3(0, Math.PI/0.24, 0);
    // box2.material = mt;

    // var box3 = BABYLON.MeshBuilder.CreateBox("box2", {size:1, height:0.1, width:0.2, depth
:0.6}, scene);
    // box3.position = new BABYLON.Vector3(-0.19, 0.19, -0.03);
    // box3.rotation = new BABYLON.Vector3(0, Math.PI/0.29, 0);
    // box3.material = mt1;

//////////////////////////////

    // var sphere = BABYLON.MeshBuilder.CreateSphere("sphere", {diameter:1}, scene);
    // sphere.position.y = 0;
```

```
// var box = BABYLON.MeshBuilder.CreateBox("box", {size: 1}, scene);
// box.position.y = 0.5;


// let cntr1 = sphere.getTotalVertices();
// let ver1 = [];
// for (let i = 0; i < cntr1; i++)
// {
//     ver1.push(1,1,0,1);
// }
// sphere.setVerticesData(BABYLON.VertexBuffer.ColorKind, ver1);

// let cntr2 = box.getTotalVertices();
// let ver2 = [];
// for (let i = 0; i < cntr2; i++)
// {
//     ver2.push(1,1,0,1);
// }
// box.setVerticesData(BABYLON.VertexBuffer.ColorKind, ver2);

// let boxCSG = BABYLON.CSG.FromMesh(box);
// let sphereCSG = BABYLON.CSG.FromMesh(sphere);

// let newshape = sphereCSG.subtract(boxCSG);

// let lastshape = newshape.toMesh("newshapess", null, scene);

// box.visibility = false;
// sphere.visibility = false;

// var sphere2 = BABYLON.MeshBuilder.CreateSphere("sphere", {diameter: 0.9}, scene);

// let cntr3 = sphere2.getTotalVertices();
// let ver3 = [];
// for (let i = 0; i < cntr3; i ++)
// {
//     ver3.push(1,1,0,1);
// }
// sphere2.setVerticesData(BABYLON.VertexBuffer.ColorKind, ver3);

// let sp1 = BABYLON.CSG.FromMesh(lastshape);
// let sp2 = BABYLON.CSG.FromMesh(sphere2);

// let cub = sp1.subtract(sp2);

// let newcub = cub.toMesh("newcub", null, scene);

// sphere2.visibility = false;
// lastshape.visibility = false;
```

```
    // var hand = BABYLON.MeshBuilder.CreateTorus("handle",
    // {
    //     diameter: 0.3,
    //     thickness:0.06,
    //     tessellation: 20
    // }
    // , scene);
    // hand.rotation.x = Math.PI/2;
    // hand.position = new BABYLON.Vector3(0.60, -0.22, 0);

    // var mt = new BABYLON.StandardMaterial("myMaterial", scene);
    // mt.diffuseColor = new BABYLON.Color3(1, 1, 0);
    // hand.material = mt;

    // var cylinder = BABYLON.MeshBuilder.CreateCylinder("cylinder", {height: 0.04, diameter:
 3}, scene);
    // cylinder.position.y = -0.5;

    // var mt1 = new BABYLON.StandardMaterial("myMaterial", scene);
    // mt1.diffuseColor = new BABYLON.Color3(1, 0, 0);

    // cylinder.material = mt1;

////////////////////////////////////////////

    // var box = BABYLON.MeshBuilder.CreateBox("box", {size: 0.5, width:2, height:0.5, depth:
0.5}, scene);
    // var box1 = BABYLON.MeshBuilder.CreateBox("box", {size: 0.5, width:0.5, height:2, depth
:0.5}, scene);
    // var box2 = BABYLON.MeshBuilder.CreateBox("box", {size: 0.5, width:0.5, height:0.5, dep
th:2}, scene);

    // var cylinder = BABYLON.MeshBuilder.CreateCylinder("cylinder", {height: 1.5, diameter:
0.3}, scene);
    // cylinder.rotation.x = Math.PI/2;
    // cylinder.position.y = 0.6;

    // var cylinder2 = BABYLON.MeshBuilder.CreateCylinder("cylinder2", {height: 1.5, diameter
: 0.3}, scene);
    // cylinder2.rotation.z = Math.PI/2;
    // cylinder2.position.y = 0.6;

    // var cylinder3 = BABYLON.MeshBuilder.CreateCylinder("cylinder3", {height: 4, diameter:
0.3}, scene);
    // cylinder3.rotation.x = Math.PI/2;
    // cylinder3.position.y = 0;

    // let cntr1 = cylinder3.getTotalVertices();
    // let ver1 = []
```

```
    // for (let i = 0; i < cntr1; i++)
    // {
    //     ver1.push(1,1,0,1);
    // }
    // cylinder3.setVerticesData(BABYLON.VertexBuffer.ColorKind, ver1);

    // let cntr2 = box2.getTotalVertices();
    // let ver2 = [];
    // for(let i = 0; i < cntr2; i++)
    // {
    //     ver2.push(1,1,0,1);
    // }
    // box2.setVerticesData(BABYLON.VertexBuffer.ColorKind, ver2);

    // let boxCSG = BABYLON.CSG.FromMesh(box2);
    // let cylCSG = BABYLON.CSG.FromMesh(cylinder3);

    // let newcyl = boxCSG.subtract(cylCSG);
    // let lascyl = newcyl.toMesh("lastcyl", null, scene);

    // box2.visibility = false;
    // cylinder3.visibility = false;

    // var mt1 = new BABYLON.StandardMaterial("myMaterial", scene);
    // mt1.diffuseColor = new BABYLON.Color3(1, 0, 0);

    // lascyl.material = mt1;
    // cylinder2.material = mt1;
    // cylinder.material = mt1;
    // box.material = mt1;
    // box1.material = mt1;

////////////////////////////////

    var cylinder = BABYLON.MeshBuilder.CreateCylinder("cylinder", {height: 2, diameter: 1, },
 scene);


    //var ground = BABYLON.MeshBuilder.CreateGround("ground", {width: 6, height: 6}, scene);



    return scene;
};
```