Explicações dos Exemplos

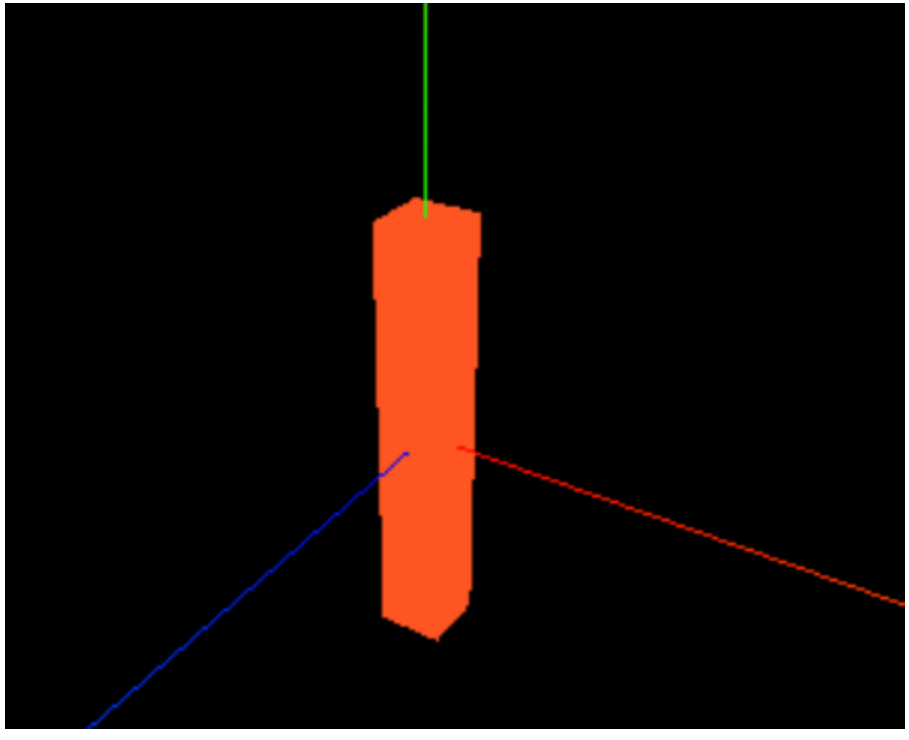
Grouping

Prof. Forster

Vamos ver a explicação dos exemplos da cena das cadeiras.

São 4 exemplos construídos sucessivamente.

O primeiro exemplo define uma perna da cadeira e permite alterar a visualização com o mouse.



Os eixos cartesianos são X (vermelho), Y (verde) e Z (azul).

Pergunta: Como podemos melhorar esse serrilhado aparente?

Parte HTML do código:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8>
<title>My first three.js app</title>
<style>
   body { margin: 0; }
   canvas { width: 100%; height: 100% } </style>
</head>
<body>
<script src="js/three.js"></script>
<script src="js/OrbitControls.js"></script>
<script>
...
</script>
</body>
</html>
```

Observe que inclui o módulo OrbitControls para controle da câmera com o mouse.

```
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75,
   window.innerWidth / window.innerHeight, 0.1, 1000 );
var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
camera.position.z = 5;

var controls = new THREE.OrbitControls( camera );

var axesHelper = new THREE.AxesHelper( 5 );
scene.add( axesHelper );
```

Aqui temos a construção dos objetos básicos do THREE.JS, uma cena, uma câmera e um renderizador. A canvas é criada no código.
É criado o objeto do OrbitControls e a câmera é passada para que esse módulo possa modificá-la.
Um objeto AxesHelper é adicionado para visualizar os eixos cartesianos.

Aqui criamos o objeto da perna da cadeira: uma caixa de dimensões 0.3, 2.0, 0.3, centrada na origem O material é simplesmente a cor 0xFF5522.

```
var leg_geom = new THREE.BoxGeometry(0.3,2.0,0.3)
var leg_mat = new THREE.MeshBasicMaterial({color:0xFF5522})
var leg = new THREE.Mesh(leg_geom, leg_mat)

scene.add(leg)
```

```
        var animate = function () {
                requestAnimationFrame( animate );
                controls.update()

                renderer.render( scene, camera );
        };

        animate()
```

Não há muito o que fazer na função de animação. Ela é invocadauma vez pelo script e as demais vezes pelo navegador, sendo a chamada agendada pelo `requestAnimationFrame`. A cada chamada da `animate`, é feito um novo agendamento.
Aqui simplesmente atualizamos a câmera pelo OrbitControls e mandamos desenhar a cena.

No segundo exemplo, montamos um grupo com as 4 pernas da cadeira.

Pergunta: Como podemos fazer para que cada face tenha um sombreamento adequado?

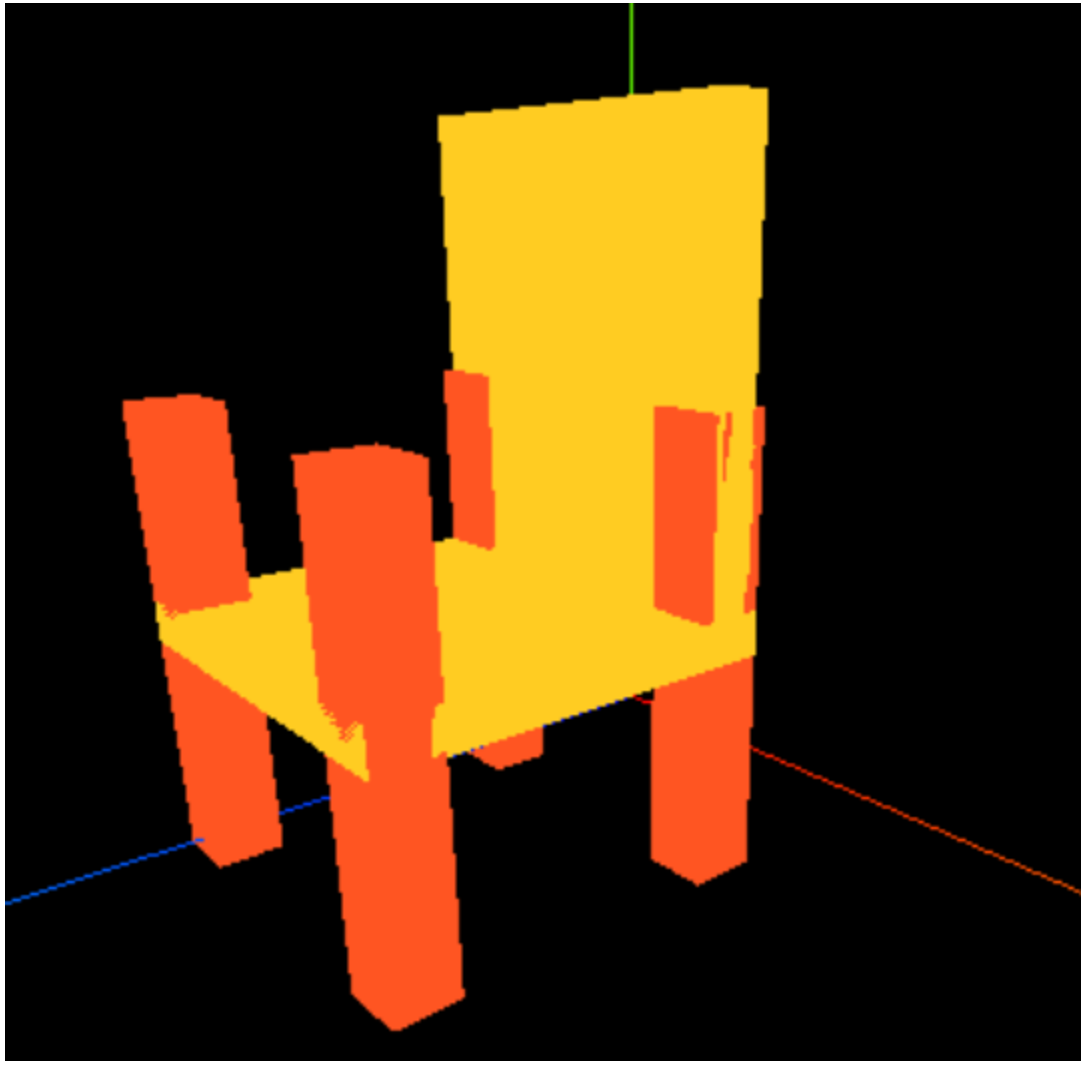
As coordenadas de inserção das pernas são descritas na lista `positions`. A variável `g` contem um grupo e para cada par que corresponde a uma posição, é criada uma cópia da perna com `leg.clone()` e reposicionada e adicionada ao grupo.

```
positions=[ [0,0], [1.5, 0], [1.5, 1.5], [0, 1.5] ]

var g= new THREE.Group()
for (var i=0; i<positions.length; i++)
{
  x=positions[i][0]
  z=positions[i][1]
  m = leg.clone()
  m.position.x=x
  m.position.z=z
  g.add(m)
}

scene.add(g)
```

No terceiro exemplo, já podemos ver como o grupo funciona com as transformações.



Observe o defeito nos planos em que há intersecção dos dois objetos: **Z-fighting**

Definimos a assento e o encosto da cadeira. São caixas com material de cor 0xFFCC22

```
var seat_geom = new THREE.BoxGeometry(1.8,0.2,1.8)
var seat_mat= new THREE.MeshBasicMaterial({color:0xFFCC22})
var seat=new THREE.Mesh(seat_geom,seat_mat)

seat.position.set(0.75,1.0,0.75)

var backrest_geom = new THREE.BoxGeometry(1.8,2.0,0.2)
var backrest_mat= new THREE.MeshBasicMaterial({color:0xFFCC22})
var backrest=new THREE.Mesh(backrest_geom,backrest_mat)

backrest.position.set(0.75,2.0,0.0)
```

Formamos um novo grupo `g2` com o grupo `g` das pernas, mais o assento e o encosto da cadeira.

```
var g2 = new THREE.Group()
g2.add(g)
g2.add(seat)
g2.add(backrest)

scene.add(g2)
```

Assim há uma hierarquia onde `g2` é a cadeira, `g` são as pernas da cadeira. Se aplicamos uma transformação em `g2`, as pernas da cadeira vão junto. Se aplicamos uma transformação apenas em `g`, as quatro pernas são transformadas, mas o encosto e o assento permanecem.

Com a animação do exemplo 3 fica clara esta dinâmica.
Criamos um movimento de `g` no eixo y (verde e vertical) e de `g2` no eixo z (azul e horizontal).
Podemos observar que `g` se move diagonalmente com a soma desses componentes de movimento.

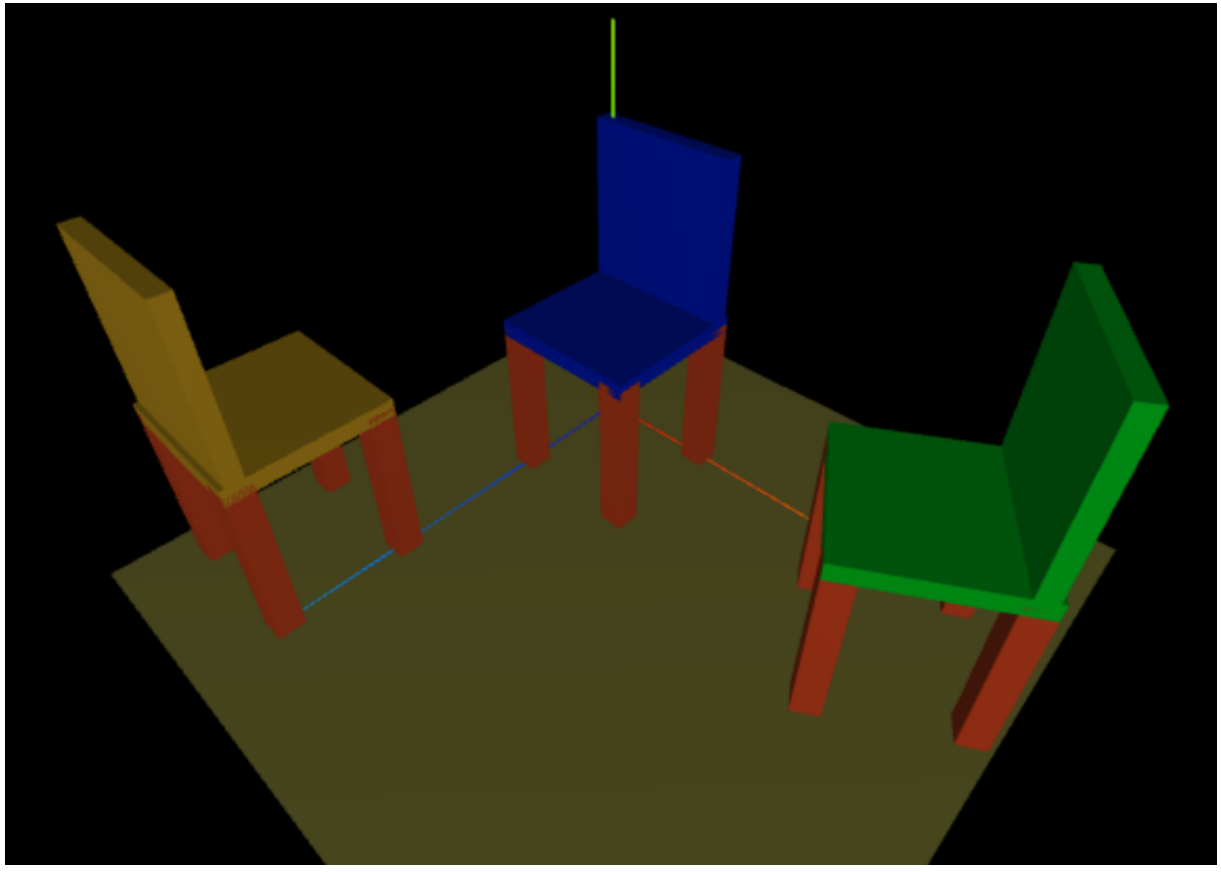
```
var t=0
var animate = function () {
        requestAnimationFrame( animate );
        controls.update()
        t=(t+1)%60

        g.position.y=Math.sin(t*Math.PI/30.0)

        g2.position.z=Math.sin(t*Math.PI/30.0)

        renderer.render( scene, camera );
};
animate()
```

No quarto exemplo, definimos uma função para criar as cadeiras. Assim podemos construir proceduralmente uma saleta com 3 cadeiras, posicionando-as individualmente e colocando uma luz variável com o tempo.



Bom, aqui já corrigimos o serrilhado:

```
var renderer = new THREE.WebGLRenderer({antialias:true});
```

E também colocamos sombreamento nos objetos através do material Lambert e da definição de fontes de luz.

```
var leg_mat= new THREE.MeshLambertMaterial({color:0xFF5522})
```

```
var light = new THREE.AmbientLight( 0x404040 );
scene.add( light );

var light2 = new THREE.PointLight( 0xffffff, 1, 100 );
light2.position.set( 50, 20, 50 );
scene.add( light2 );
```

Nossa animação só altera a intensidade da luz:

```
light2.color.g=Math.abs(Math.sin(t*Math.PI/30.0))
```

O mais interessante aqui é definir a cena. Uma função cria cadeiras dada a cor como parâmetro `col` (com valor default). E essas cadeiras são objetos que podem ser posicionados na cena.

```
function makeChair(col=0xFFCC22)
{
  var leg_geom= new THREE.BoxGeometry(0.3,2.0,0.3)
  var leg_mat= new THREE.MeshLambertMaterial({color:0xFF5522})
  var leg = new THREE.Mesh(leg_geom, leg_mat)

  var g= new THREE.Group()
  positions=[ [0,0], [1.5, 0], [1.5, 1.5], [0, 1.5] ]
  for (var i=0; i<positions.length; i++)
        {
          x=positions[i][0]
          z=positions[i][1]
          m = leg.clone()
          m.position.x=x
          m.position.z=z
          m.position.y=0
          g.add(m)
        }
```

Nesta parte do código, criamos as quatro pernas.

```
var seat_geom = new THREE.BoxGeometry(1.8,0.2,1.8)
var seat_mat= new THREE.MeshLambertMaterial({color:col})
var seat=new THREE.Mesh(seat_geom,seat_mat)

seat.position.set(0.75,1.0,0.75)

var backrest_geom = new THREE.BoxGeometry(1.8,2.0,0.2)
var backrest_mat= new THREE.MeshLambertMaterial({color:col})
var backrest=new THREE.Mesh(backrest_geom,backrest_mat)

backrest.position.set(0.75,2.0,0.0)

var g2 = new THREE.Group()
g2.add(g)
g2.add(seat)
g2.add(backrest)

return g2
}
```

Nesta segunda parte, criamos o encosto e o assento. Veja que colocamos `col` como cor dos materiais. Por fim, retornamos o grupo, o objeto que contém os componentes da cadeira.

A função de criar cadeira é utilizada para criar as três cadeiras coloridas e em seguida estas são posicionadas, colocadas num novo grupo `g_chairs` e colocadas na cena.

```
chairs = [ makeChair(),
  makeChair(0x0022FF),
  makeChair(0x00FF22)]

chairs[0].position.z=5
chairs[0].rotation.y=Math.PI

chairs[2].position.x=5
chairs[2].rotation.y=-Math.PI/3.0

g_chairs=new THREE.Group()
for( var i=0; i<3; i++) g_chairs.add(chairs[i]);
scene.add(g_chairs);

g_chairs.position.y=1
```

Veja que o conjunto das cadeiras pode ser transformado como um todo através do grupo `g_chairs`.