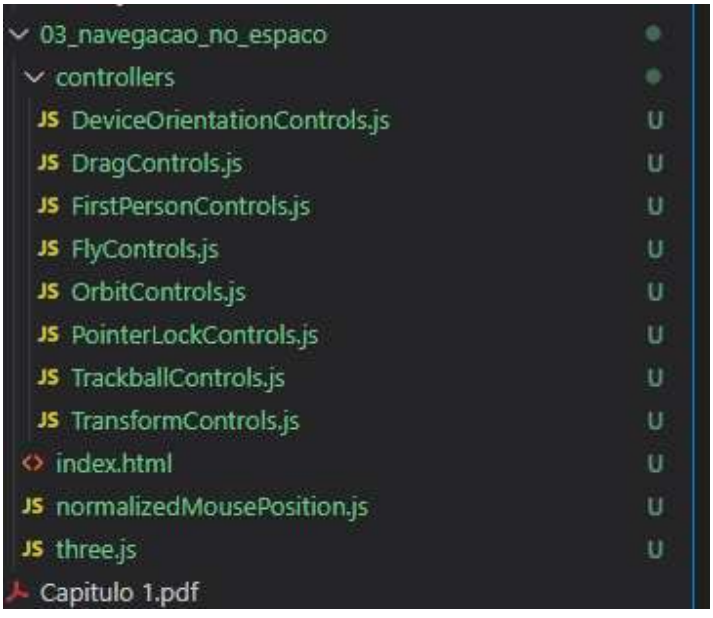
4)Navegando no espaço

A utilidade do 3d é para ter a capacidade de se movimentar em um universo tridimensional. Neste capitulo aprenderemos como controlar a câmera para rotações, translações e zoom navegando ao longo da cena.

4.1)Preparando o ambiente

- Criaremos uma nova pasta, 03_navegacao_no_espaco, colocaremos nela a index.html da lição anterior, o js da three.js e o normalizedMousePosition.js da lição anterior e iniciaremos o servidor. Ou seja, continuaremos a partir da lição anterior.
- Um dos grandes benefícios da Three.js é ela já lidar com muito da complexidade da computação gráfica. Uma dessas complexidades são os controladores de câmera. Criá-los do zero envolve um conhecimento profundo de geometria espacial como quaternions e é fácil cometer erros. O ideal é usar controladores de câmera prontos sempre que possível. A Threejs os provê na forma dos Controls. Vamos instalá-los em nosso projeto.
- Eles se encontram em https://github.com/mrdoob/three.js/tree/9ef27d1af7809fa4d9943f8d4c4644e365ab6d2d/examples/js/controls . Criei uma pasta chamada controllers e baixei todos os controladores nessa url. Esse mecanismo de adição dos controllers está obsoleto, porém, e será desativado logo. O ideal é usar o sistema de módulos como será visto em uma lição mais à frente mas para esse primeiro momento isto serve..



- Adiciono o OrbitControl. O OrbitControl é um controlador para rotação ao redor de um ponto no espaço:

```
<script src="three.js"></script>
<script src="normalizedMousePosition.js"></script>
<script src="controllers/OrbitControls.js"></script>
```

4.2)Construção da cena

- Faremos uma cena um pouco mais complexa desta vez, com vários cubos e um plano.
- Para tornar o código mais legível o processo de criação dos cubos e de planos está contido em funções específicas, a createCube e a createPlane.
- CreateCube:

```
            function createCube(x,y,z, color, name){
                const geometry = new THREE.BoxGeometry();
                const material = new THREE.MeshBasicMaterial({color:color});
                const cube = new THREE.Mesh(geometry, material);
                cube.position.x = x;
                cube.position.y = y;
                cube.position.z = z;
                cube.name = name
                return cube;
            }
```

- CreatePlane:

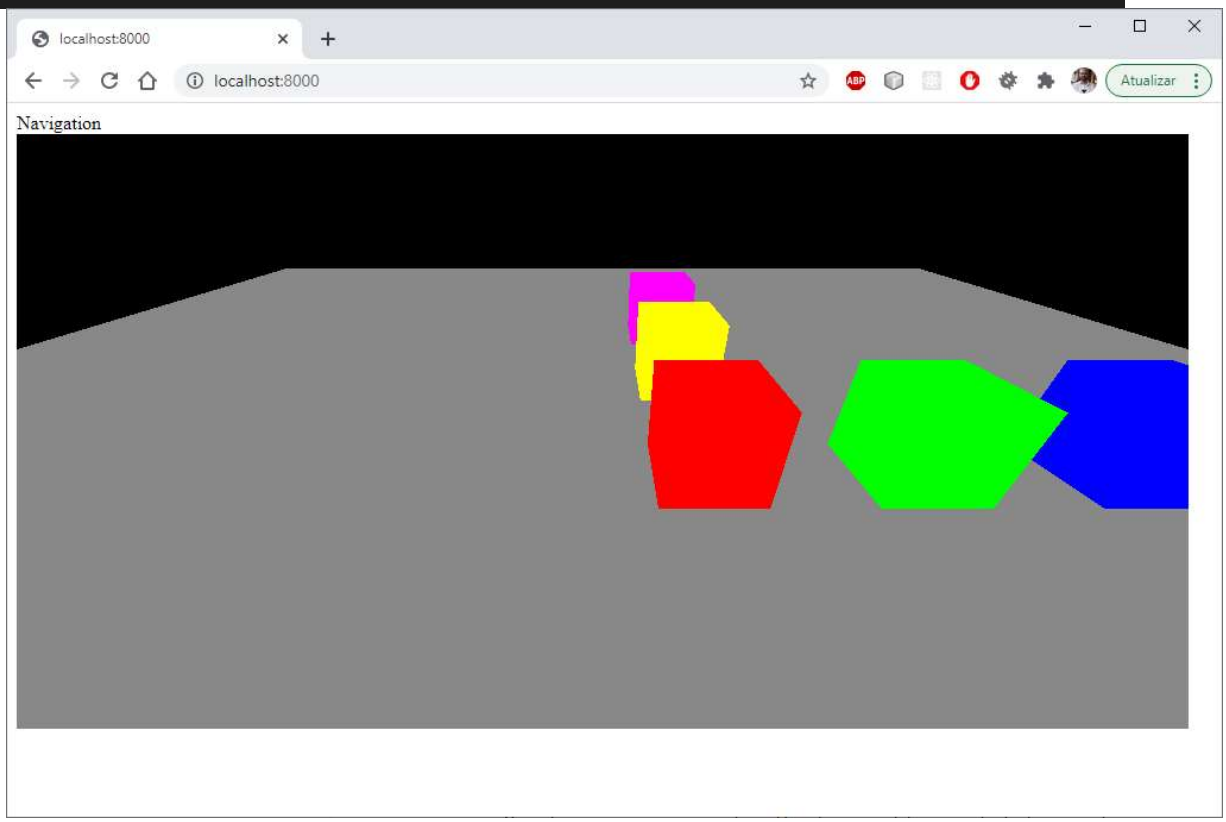
```
            function createPlane(x,y,z, width, height, color, name){
                const geometry = new THREE.PlaneGeometry(width, height)
                const material = new THREE.MeshBasicMaterial({color:color, side:
THREE.DoubleSide});
                const plane = new THREE.Mesh(geometry, material);
                plane.position.x = x;
                plane.position.y = y;
                plane.position.z = z;
                plane.width = width;
                plane.height = height
                plane.name = name
                plane.rotation.x = 3.14 / 2.0;
                return plane;
            }
```

- Na CreatePlane nós vamos rotacionar o plano, pois ele é criado por padrão na vertical. As rotações de uma Mesh estão no atributo rotation e são rotações eulerianas, portanto

ao redor dos eixos x, y, z. O atributo x é a rotação em graus ao redor do eixo x, o atributo y é a quantidade de graus ao redor do eixo y e o atributo z é a quantidade de graus ao redor do eixo z. Os graus são sempre em radianos. Em createPlane o plano foi rotacionado 90º ao redor do eixo x, indo portanto para a horizontal.

- Definidas as funções que criam objetos, populemos a cena com objetos e modifiquemos a posição inicial da câmera. Na maioria das situações reais a cena será carregada de arquivos ao invés de montada manualmente mas a carga de arquivos é assunto para as próximas seções. Além disso o sistema de shading (colorimento) dos objetos está bem simples: cores sólidas, sem sombra e efeitos de iluminação. Isso também não é o assunto da seção atual:

```
scene.add(createCube(1,0,0, 0xff0000, "vermelho"));
scene.add(createCube(3,0,0, 0x00ff00, "verde"));
scene.add(createCube(5,0,0, 0x0000ff, "azul"));
scene.add(createCube(1,0,-2, 0xffff00, "amarelo"));
scene.add(createCube(1,0,-4, 0xff00ff, "roxo"));
scene.add(createPlane(0,-0.5,0, 20, 20, 0x888888, "plano"));
camera.position.set(0,2,3);
```



4.3)Orbit Control

- Agora usamos o OrbitControl (https://threejs.org/docs/index.html#examples/en/controls/OrbitControls) . Um orbit

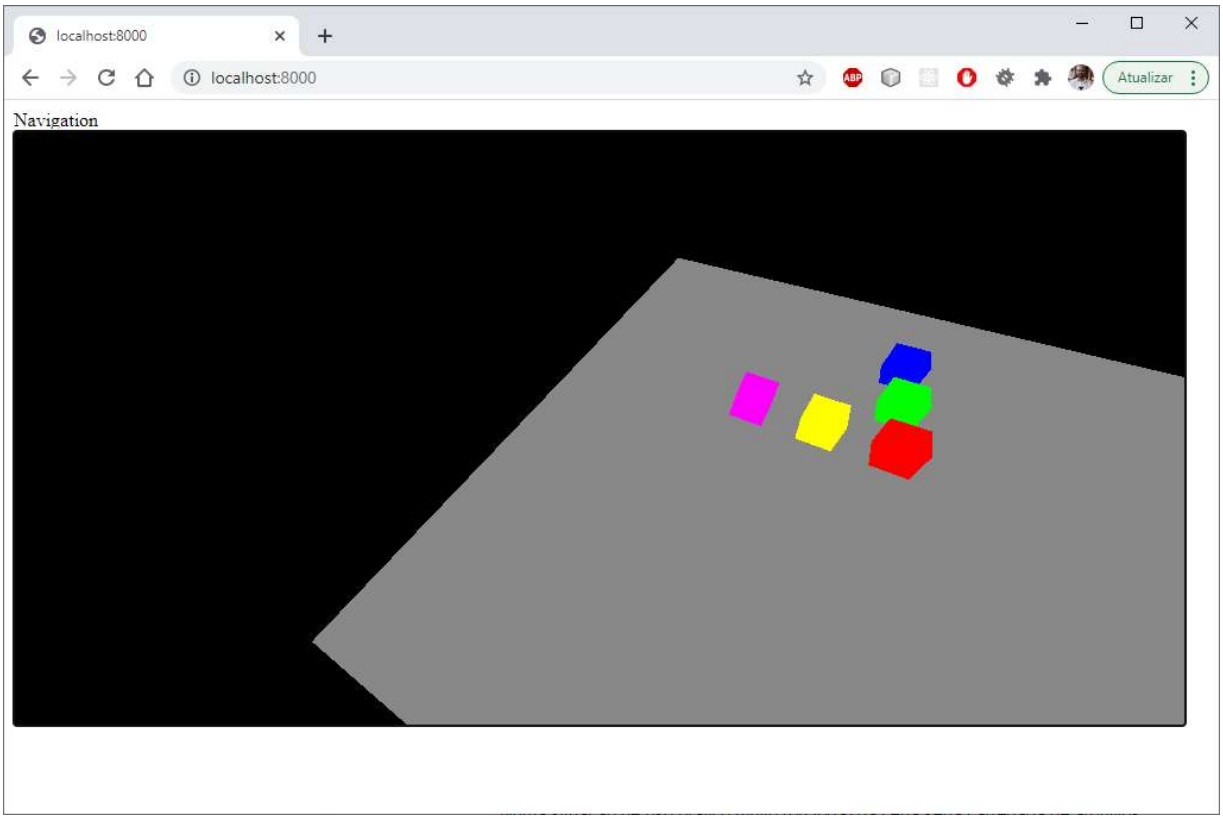
control é criado vinculado a uma câmera e ao elemento do DOM do renderer porque ele toma o controle tanto da câmera quanto dos event listeners do elemento. O Control configurará seus próprios event listeners necessários para tratar as ações do usuário como click, drag, mousewheel e controlará a matemática da câmera. Isso não impede que nós adicionemos os nossos próprios event listeners e mudemos da nossa própria maneira as propriedades da câmera mas o fato do Control tomar o controle destas duas coisas deve ser levado em consideração.

```
 //NOVO - O orbit controlller
const control = new THREE.OrbitControls(camera, renderer.domElement);
//Depois de qualquer alteração manual da camera
control.update();
```

- Toda vez que mudarmos manualmente as propriedades da câmera, como sua posição no espaço, devemos atualizar o control. Como nós posicionamos a câmera manualmente na montagem da cena devemos atualizar o control.
- Por fim no animate devemos atualizar o control para que ele atualize a cena de acordo com as ações do usuário:

```
function animate(){
    requestAnimationFrame(animate);
    control.update();//Aqui o controller atualiza a cãmera.
    renderer.render(scene, camera);
}
```

- O resultado é podermos rotacionar, dar zoom e poder mover a câmera com o mouse:



O código final:

```
    <head>
    </head>
    <body>
        <div>Navigation</div>
        <div id="target_div" style="height:500px"></div>
        <script src="three.js"></script>
        <script src="normalizedMousePosition.js"></script>
        <script src="controllers/OrbitControls.js"></script>
        <script>
            //NOVO: Encapsulei a criação de cubos nesta função.
            function createCube(x,y,z, color, name){
                const geometry = new THREE.BoxGeometry();
                const material = new THREE.MeshBasicMaterial({color:color});
                const cube = new THREE.Mesh(geometry, material);
                cube.position.x = x;
                cube.position.y = y;
                cube.position.z = z;
```

```
                cube.name = name
                return cube;
            }
            //NOVO:Criação de um plano
            function createPlane(x,y,z, width, height, color, name){
                const geometry = new THREE.PlaneGeometry(width, height)
                const material = new THREE.MeshBasicMaterial({color:color, side:
THREE.DoubleSide});
                const plane = new THREE.Mesh(geometry, material);
                plane.position.x = x;
                plane.position.y = y;
                plane.position.z = z;
                plane.width = width;
                plane.height = height
                plane.name = name
                plane.rotation.x = 3.14 / 2.0;
                return plane;
            }
            const targetDiv = document.getElementById("target_div");
            const scene = new THREE.Scene();
            const camera = new THREE.PerspectiveCamera(75,
                targetDiv.clientWidth / targetDiv.clientHeight, 0.1, 1000);
            const renderer = new THREE.WebGLRenderer();
            renderer.setSize(targetDiv.clientWidth, targetDiv.clientHeight);
            targetDiv.appendChild(renderer.domElement);
            //Novo: Objetos sendo adicionados à cena.
            scene.add(createCube(1,0,0, 0xff0000, "vermelho"));
            scene.add(createCube(3,0,0, 0x00ff00, "verde"));
            scene.add(createCube(5,0,0, 0x0000ff, "azul"));
            scene.add(createCube(1,0,-2, 0xffff00, "amarelo"));
            scene.add(createCube(1,0,-4, 0xff00ff, "roxo"));
            scene.add(createPlane(0,-0.5,0, 20, 20, 0x888888, "plano"))
            camera.position.set(0,2,3);
            //NOVO - O orbit controlller
            const control = new THREE.OrbitControls(camera, renderer.domElement);
            //Depois de qualquer alteração manual da camera
            control.update();
            const raycaster = new THREE.Raycaster();
            targetDiv.addEventListener('click',(event)=>{
```

```
                const pickPosition = mousePositionAsNormalizedCoordinates(event,
                    renderer)
                raycaster.setFromCamera(pickPosition, camera);
                const interceptedObjects =
raycaster.intersectObjects(scene.children,
                    true);
                if(interceptedObjects.length > 0 ){
                    control.target = interceptedObjects[0].object.position;
                    control.update();
                }
            });
            function animate(){
                requestAnimationFrame(animate);
                control.update();//Aqui o controller atualiza a cãmera.
                renderer.render(scene, camera);
            }
            animate();
        </script>
    </body>
```

- Existem outros controls mas para o começo o OrbitControl é suficiente. Na próxima seção será visto como carregar arquivos FBX.
- Alguns experimentos com o que foi aprendido até agora: testar os outros controls, construir cenas mais complexas.