- A partir desta seção eu passo a usar o Web Server for Chrome como servidor web. O sistema de módulos no navegador exige um servidor que forneça os arquivos do módulo com o MIME type apropriado. Além disso não dá para abrir a index.html no navegador pelo explorador de arquivos porque nesse caso o erro será de CORS. O servidor de testes que o https://chrome.google.com/webstore/detail/web-server-for-chrome/ofhbbkphhbklhfoeikjpcbhemlocgigb/related?hl=en provê serve para os testes da threejs. Na produção seu servidor (ex. Apache ou Nginx) servirá sem problemas.

Feito todas as cópias de arquivo e levantado o servidor testo se os módulos estão sendo carregados corretamente:

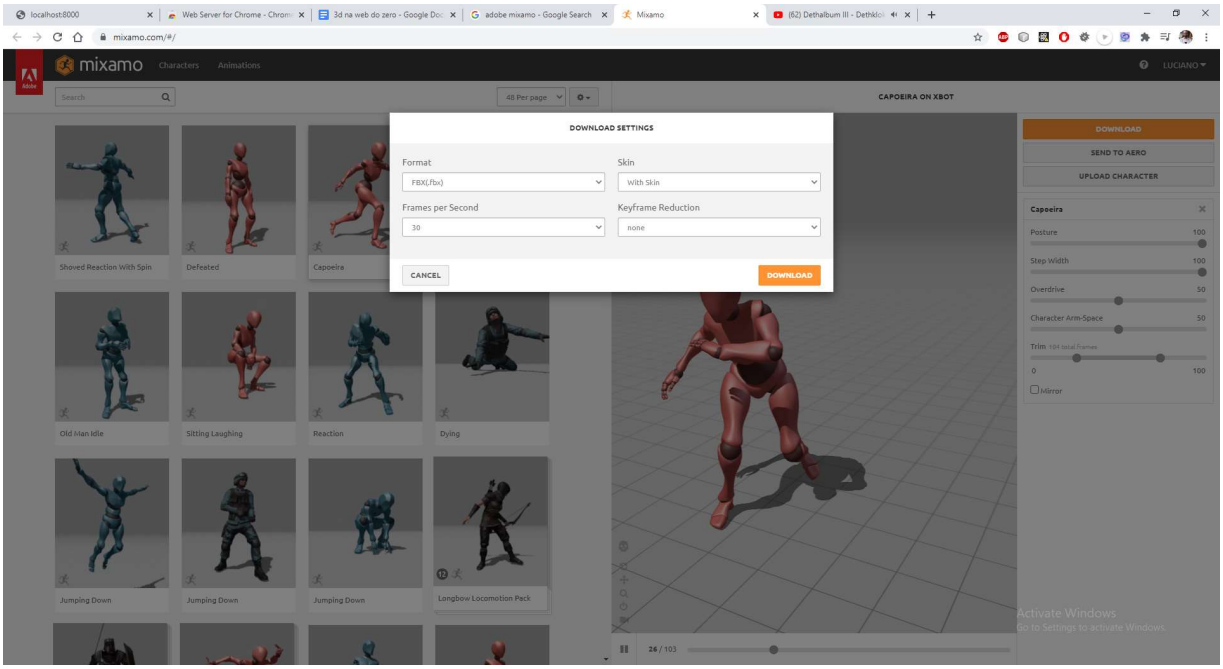
```
<html>
    <head title="Carregando FBX"></head>
    <body>
        <div id="c" style="height:500px"></div>
    </body>
    <script type="module">
        //NOVO: Importando a three como módulo.
        import * as THREE from "./build/three.module.js";
        import {OrbitControls} from "./jsm/controls/OrbitControls.js"
        import { FBXLoader } from './jsm/loaders/FBXLoader.js';
        function createRendererCameraScene(canvas){
            const renderer = new THREE.WebGLRenderer();
            const camera = new THREE.PerspectiveCamera(75,
canvas.clientWidth/canvas.clientHeight, 0.1, 1000);
            const scene = new THREE.Scene();
            renderer.setSize(canvas.clientWidth, canvas.clientHeight);
            canvas.appendChild(renderer.domElement);
            return {renderer, camera, scene};
        }
        function main() {
            const canvas = document.querySelector('#c');
            const {renderer, camera, scene} = createRendererCameraScene(canvas);
            function animate(){
                requestAnimationFrame(animate);
                renderer.render(scene, camera);
            }
            animate();
        }
        main();
```

```
    </script>
</html>
```

- Se tudo estiver correto será visto uma tela preta e se conferirmos no debugger do chrome todos os objetos estarão inicializados corretamente, podendo portanto prosseguir para a próxima etapa.

5.2)Carga do FBX

- A primeira parte é baixar o arquivo do Mixamo. Tanto faz qual, desde que seja baixado como FBX e tenha a opção “with skin” selecionada. Escolhi o de capoeira e o pus na pasta de Assets.



- Crio o FBXLoader e uso a função de load dele. A carga é assincrona, com callback de progresso, de sucesso e de erro. O FBXLoader é instanciado da segunte forma:

```
 //NOVO: Carga do objeto
const loader = new FBXLoader();
```

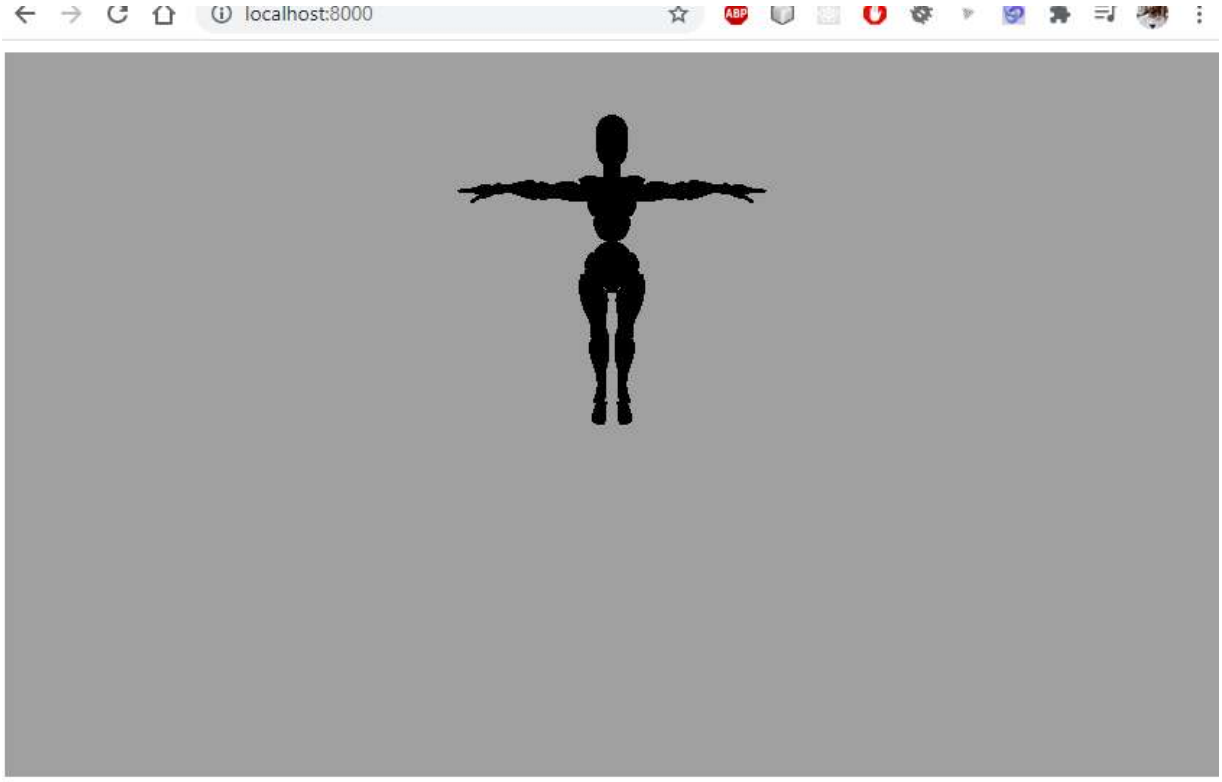
- A assinatura da função de load é xxxxxx. Seu primeiro parâmetro é a url do arquivo a ser carregado, o segundo é o callback que é invocado quando o arquivo é carregado com sucesso, o terceiro é o callback de progresso e o último é o callback de erro.

```
        load: function ( url, onLoad, onProgress, onError ) {
```

- onLoad recebe como parâmetro o objeto carregado, on progress recebe um objeto de progresso que tem o total do que está sendo carregado e o quanto já foi carregado, de onde dá pra se calcular a porcentagem do progresso para avançar uma barra de progresso por exemplo. OnError recebe o erro.
- Um exemplo de uso da função load:

```
loader.load('./assets/Capoeira.fbx',
  (object)=>{//Após a carga, o objeto carregado estará aqui.
    scene.add(object);
  },
  (progress)=>{ //Lida com o progresso da carga
    const totalToLoad = progress.total;
    const loaded = progress.loaded;
    const loadPercentage = loaded / totalToLoad;
    console.log("TODO: Mostrar numa barra de progressos "+loadPercentage);
  },
  (error)=>{console.log(error)});
```

- Se rodarmos da forma que está o objeto será carregado mas não será visível. Isso acontece porque esse objeto em particular tem um material que exige iluminação.



- Se o fundo estiver preto nem o contorno do objeto preto pela falta de luzes será visível. É possivel modificar a cor de fundo da cena passando um THREE.Color para o campo background do objeto scene. É util mudar a cor de fundo para depurar erros pois muitas vezes, devido a problemas nos materiais que dão cor ao objeto, como falta de luzes ou outros valores, eles virão pretos e um dos procedimentos de debug visual é mudar a cor do fundo para poder enxergá-los:

```
//NOVO: Determina a cor de fundo da cena
scene.background = new THREE.Color(0xa0a0a0);
```

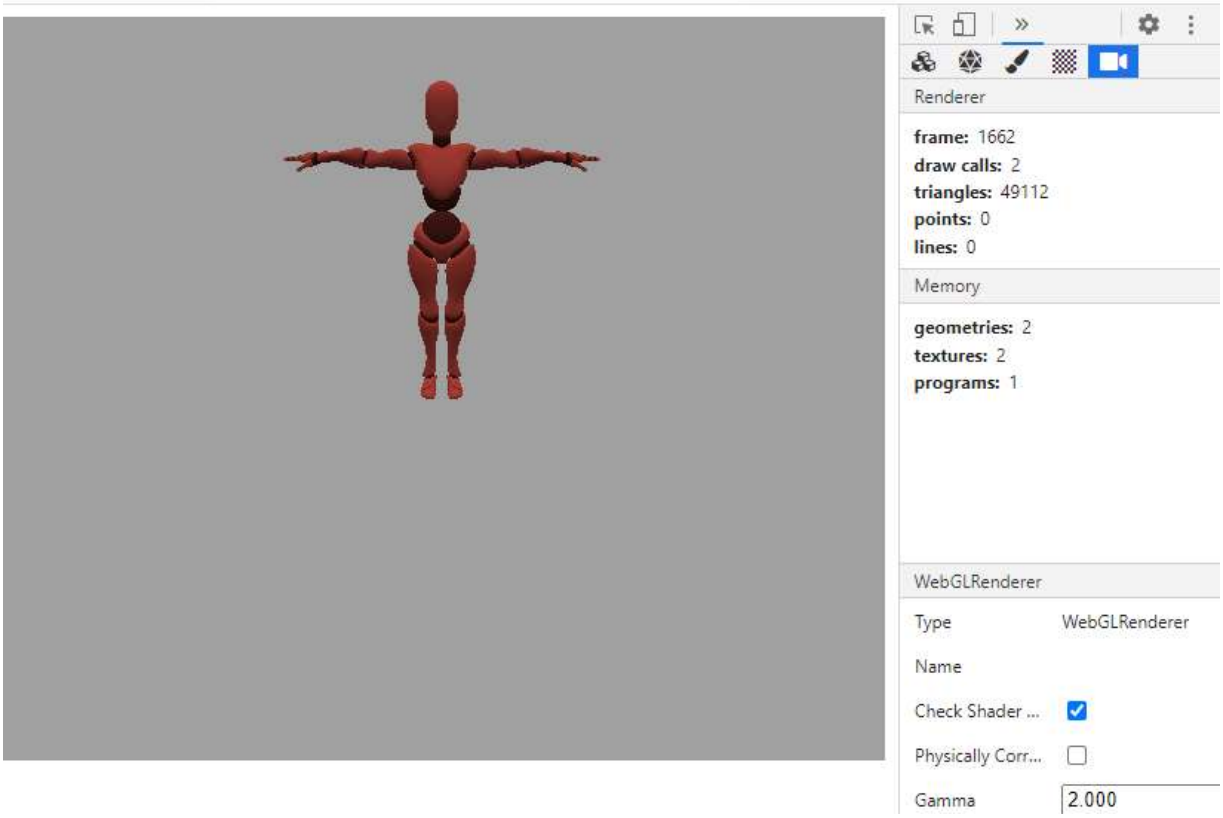
- O objeto também estará na chamada T-Pose. Isso é específico do fato de estarmos usando um boneco animado pois pois os bonecos costumam ser construídos nesta pose padrão onde eles são modelados e então costurados aos sistemas controladores dos vértices. Quando o boneco for animado na próxima etapa ele sairá da T-Pose.

5.3)Necessidade de iluminação

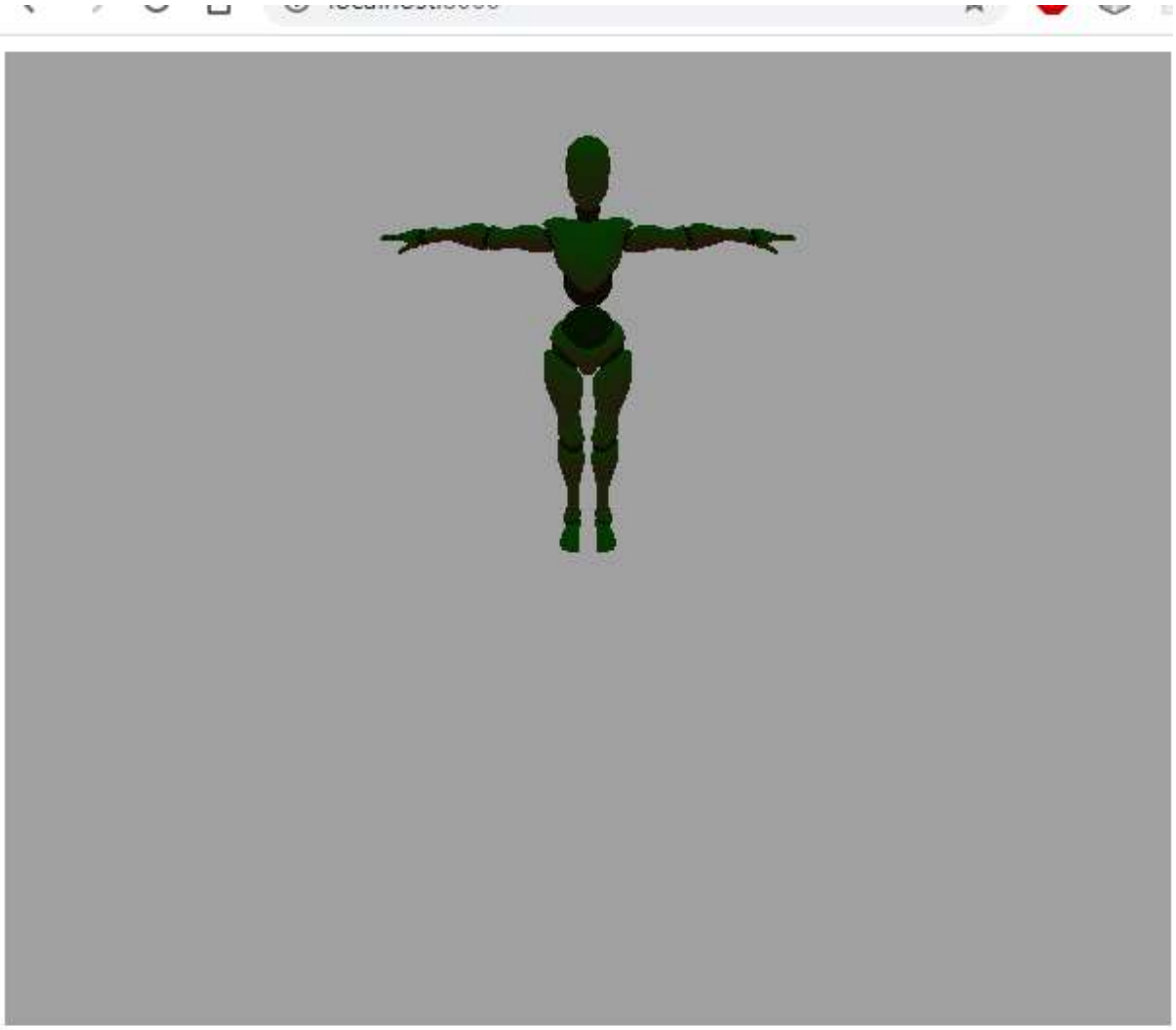
- A equação fundamental da renderização é a modelagem matemática do processo de renderização. Ela é o somatório dos efeitos de cada fonte de luz, do espaço entre a fonte de luz e a superficie sendo vista, dos efeitos de reflexão, refração, absorção e emissão das superfícees, dos efeitos do espaço entre a superfície e o observador e por fim dos efeitos na lente do observador.
- Na Threejs muitos destes efeitos estão nos materiais. Outra parte está nas propriedades das luzes e outros nos sistemas de névoa e sombra. As implementações vivem nos shaders, que transformam dados como vértices, texturas, posição de luz em cor na tela. Para esta cena bastará por a luz na cena para que o objeto se torna visível.
- Os materiais que usamos nas seções anteriores eram muito simples e dispensavam a necessidade de iluminação, por isso ela não foi usada até agora.
- O boneco está todo preto porque seu material, definido no arquivo, exige iluminação. Sem uma fonte de luz a cor de cada fragmento vai a zero.
- Usaremos uma das luzes mais simples, a HemisphereLight, definida por posição no espaço, cor na origem e cor final depois de decair. Ela não gera sombras. Para adicionarmos a HemisphereLight usamos:

```
const hemiLight = new THREE.HemisphereLight( 0xffffff, 0x444444 );
hemiLight.position.set( 0, 200, 0 );
scene.add( hemiLight );
```

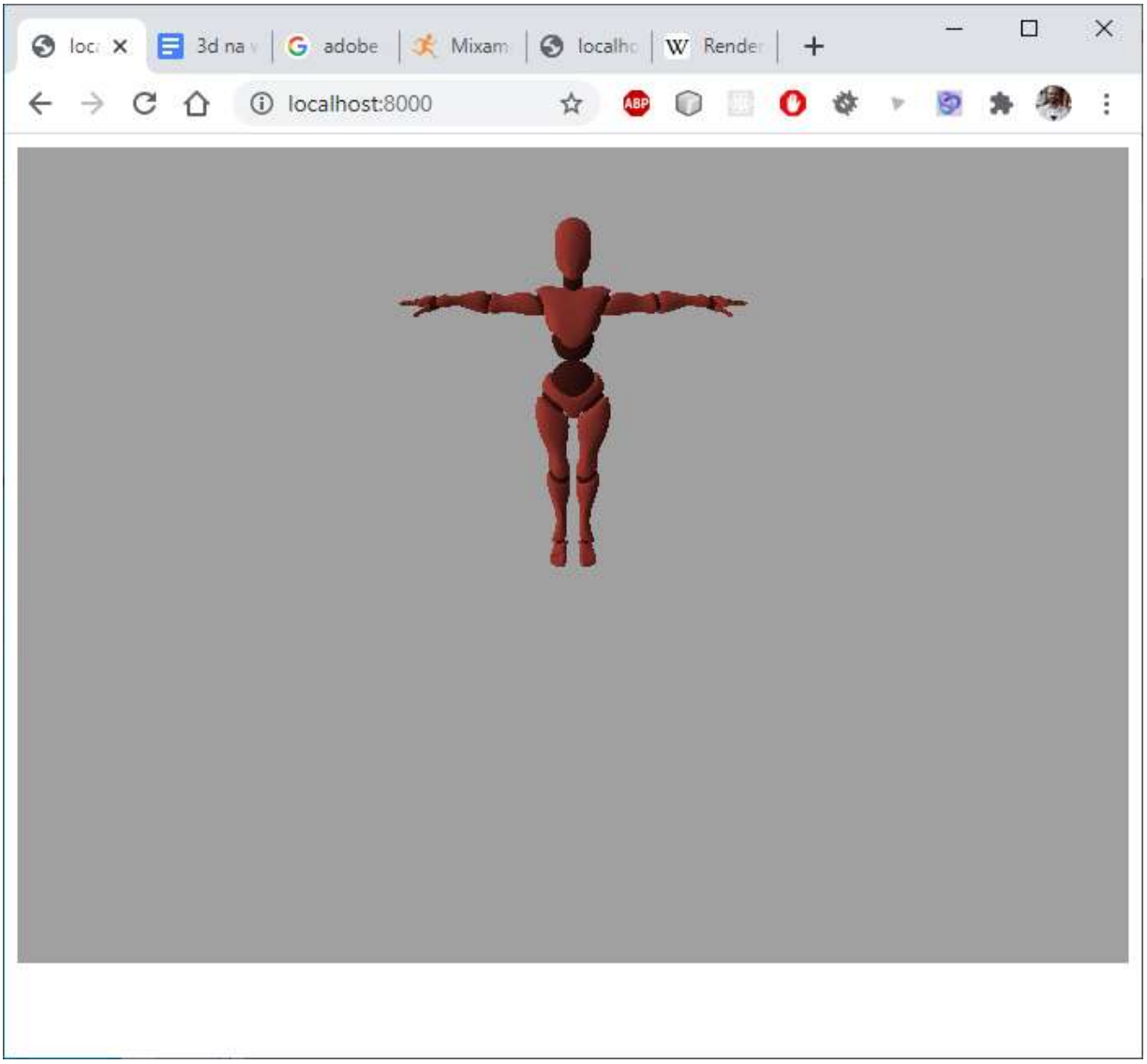
- O resultado é:



- Se mudarmos a cor da luz para verde, a cor das coisas na cena muda junt



- Se mudarmos a posição da luz o brilho muda. Neste screenshot a luz está em -500, 200,0:



- Com isso nós carregamos um arquivo FBX. Na próxima seção será mostrado como usar as animações do arquivo.