**程序人生 | 与足球共舞的火柴人（致敬格拉利什，赋予足球更深的意义）**

**一、前言**

花有重开日，人无再少年。**格拉利什** 这类人的存在，**赋予了足球更深的意义！**

**程序人生，用技术记录世界杯**，接下来我们使用 **Three.js 技术**，来实现一个 **与足球共舞的火柴人（致敬格拉利什）**

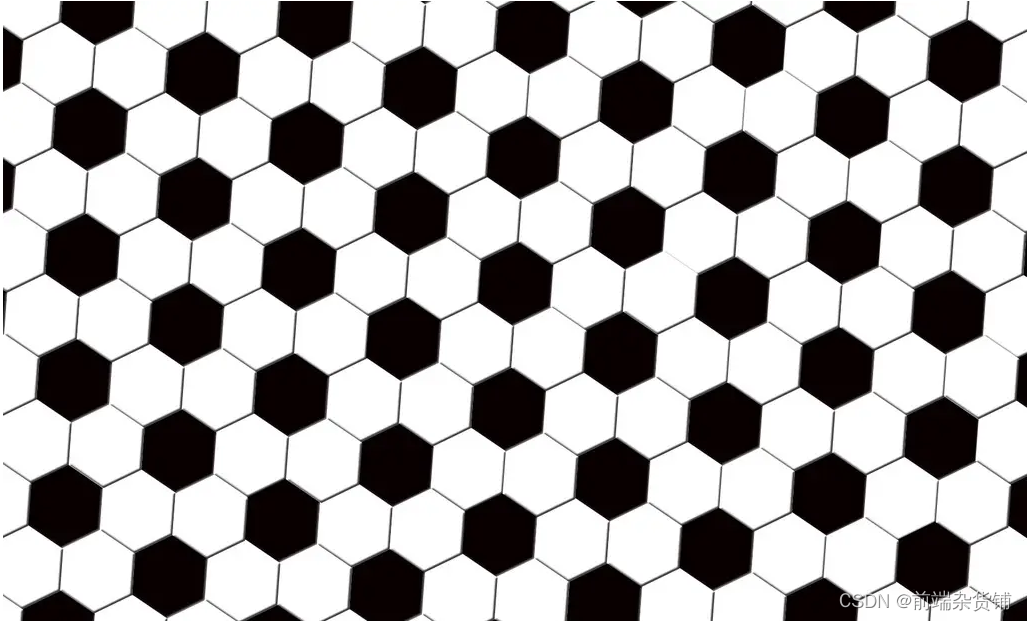
备注：其实我自己在电脑上运行效果是非常顺滑流畅的，可能是录屏软件的问题，会导致观看效果不佳（看着卡顿，其实很丝滑）

与足球共舞的火柴人

**二、使用 Three.js 渲染足球**

简介：Three.js 是 JavaScript 编写的 WebGL 第三方库。提供了非常多的 3D 显示功能。

渲染足球前我们得先有一张足球的材质贴图（football.png），用于把图贴到我们创建的球体上。如下所示：



创建一个HTML文件，World\_Cup.html

**备注：以下仅简单的解释了部分代码的含义，详细内容请自行学习**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>2022世界杯</title>

<style>

body {

margin: 0px;

}

</style>

</head>

<body>

<script type="text/javascript" src="https://cdn.bootcss.com/three.js/90/three.min.js"></script>

<script src="https://threejs.org/examples/js/controls/OrbitControls.js"></script>

<script type="text/javascript">

// 初始化相机

let camera = new THREE.PerspectiveCamera(50, window.innerWidth / window.innerHeight, 1, 10000)

camera.position.z = 500;

// 初始化场景

let scene = new THREE.Scene()

// 初始化渲染器

let renderer = new THREE.WebGLRenderer()

renderer.setSize(window.innerWidth, window.innerHeight)

document.body.appendChild(renderer.domElement)

// 定义一个球体，这里的参数可以根据需要调节

let geometry = new THREE.SphereGeometry(50, 32, 32)

let texture = new THREE.TextureLoader().load('./imgs/football.png')

let material = new THREE.MeshBasicMaterial({map: texture})

// 将材质和几何体进行绑定

let cube = new THREE.Mesh(geometry, material)

// 将绑定后的几何体放入场景中

scene.add(cube)

// 进行渲染

function animate() {

requestAnimationFrame(animate)

renderer.render(scene, camera)

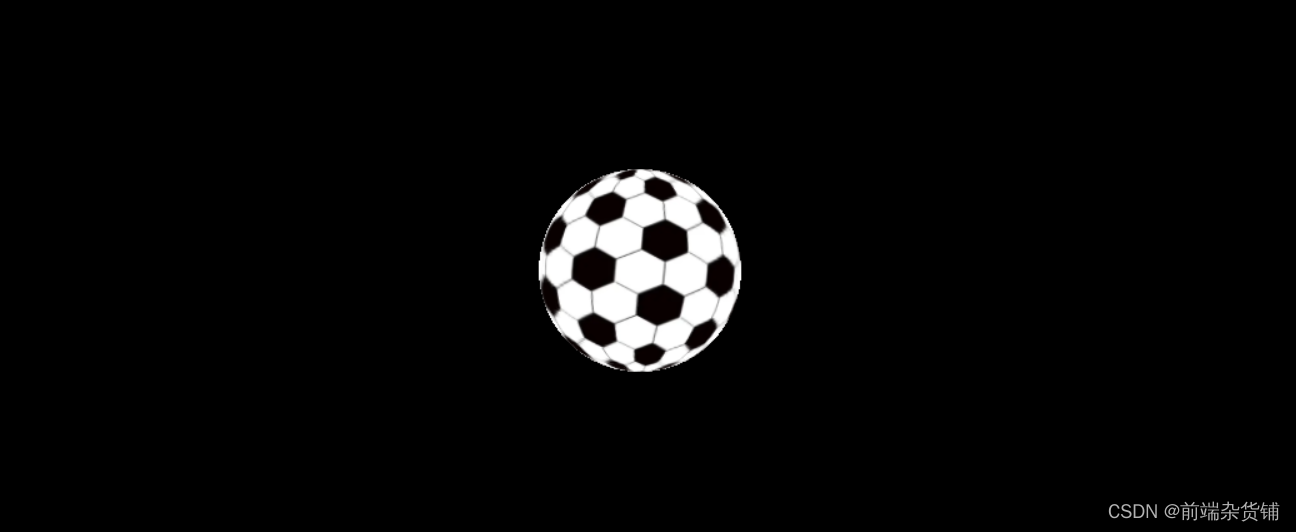
}

animate()

</script>

</body>

</html>



**三、使用 Three.js 渲染跳舞的火柴人**

创建一个HTML文件，Matchman.html。该部分代码比较复杂，不仅需要创建场景生成火柴人，还需要给他添加一些列的动作，并完美的渲染出来（不白屏，不卡顿等）。所以需要引入很多内置的 js 文件，在此就不列举了，想要代码的可以找我要。

<!DOCTYPE html>

<html lang="en">

<head>

<title>three.js webgl - loaders - BVHLoader</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">

<link type="text/css" rel="stylesheet" href="main.css">

<style>

body {

background-color: #eee;

color: #444;

}

a {

color: #08f;

}

h2 {

color: orange;

}

</style>

</head>

<body>

<div id="info">

<h2>与足球共舞的火柴人【前端杂货铺】</h2>

</div>

<script async src="https://unpkg.com/es-module-shims@1.3.6/dist/es-module-shims.js"></script>

<script type="importmap">

{

"imports": {

"three": "../build/three.module.js",

"three/addons/": "./jsm/"

}

}

</script>

<script type="module">

import \* as THREE from 'three';

import { OrbitControls } from 'three/addons/controls/OrbitControls.js';

import { BVHLoader } from 'three/addons/loaders/BVHLoader.js';

const clock = new THREE.Clock();

let camera, controls, scene, renderer;

let mixer, skeletonHelper;

init();

animate();

const loader = new BVHLoader();

loader.load( 'models/bvh/pirouette.bvh', function ( result ) {

skeletonHelper = new THREE.SkeletonHelper( result.skeleton.bones[ 0 ] );

skeletonHelper.skeleton = result.skeleton; // allow animation mixer to bind to THREE.SkeletonHelper directly

const boneContainer = new THREE.Group();

boneContainer.add( result.skeleton.bones[ 0 ] );

scene.add( skeletonHelper );

scene.add( boneContainer );

// play animation

mixer = new THREE.AnimationMixer( skeletonHelper );

mixer.clipAction( result.clip ).setEffectiveWeight( 1.0 ).play();

} );

function init() {

camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 1, 1000 );

camera.position.set( 0, 200, 300 );

scene = new THREE.Scene();

scene.background = new THREE.Color( 0xeeeeee );

scene.add( new THREE.GridHelper( 400, 10 ) );

// renderer

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

renderer.setPixelRatio( window.devicePixelRatio );

renderer.setSize( window.innerWidth, window.innerHeight );

document.body.appendChild( renderer.domElement );

controls = new OrbitControls( camera, renderer.domElement );

controls.minDistance = 300;

controls.maxDistance = 700;

window.addEventListener( 'resize', onWindowResize );

}

function onWindowResize() {

camera.aspect = window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();

renderer.setSize( window.innerWidth, window.innerHeight );

}

function animate() {

requestAnimationFrame( animate );

const delta = clock.getDelta();

if ( mixer ) mixer.update( delta );

renderer.render( scene, camera );

}

</script>

</body>

</html>

跳舞的火柴人

**四、总结（充能）**

把这两部分代码 **结合一下** 就可以完成 **与足球共舞的火柴人** 了。

怎么样，是不是觉得 Three.js 还是很有意思的。

**扩展：其实 Three.js 的用途还是很多的**

* 智慧城市
* 房屋 3D 视图
* 开发工业软件（CAD，CAE等）

总的来说 Three.js 就是来渲染 3D 效果的。目前 Three.js 是个很强大好用的 3D 渲染库，接下来我也会进行这方面的学习，到时候可以 **和大家一起探索 Three.js 的世界。**