**Getting Started - Chapter 8 - Follow That Character**

**@@启程-第八章-跟随角色@@**

**Follow That Character**

**@@跟随角色@@**

Without using parenting we can also track the characters movements with a *FollowCamera*.

@@不把网格设为相机的父元素我们也能跟随角色，只需使用“跟随相机”即可。@@

We give a *FollowCamera* a start position and a target to follow and a goal position from which to view the target.

@@我们可以给“跟随相机”设置一个起始位置，一个跟随目标以及一个观察目标的目的位置。@@

We create the *FollowCamera* with a name, start position and the optional scene parameter.

@@我们使用名称、起始位置、以及可省略的场景参数建立“跟随相机”对象。@@

const camera = new BABYLON.FollowCamera("FollowCam", new BABYLON.Vector3(-6, 0, 0), scene);

We then set the goal: height above the the center of the target;

@@接着我们设置目标：在目标对象的中心的上面；@@

camera.heightOffset = 8;

radial distance from target plus height offset;

@@与目标的径向距离，实际距离还要加上高度偏移（也就是说跟随相机位于角色头顶9单位长度的位置）@@

camera.radius = 1;

rotation, in radians, center of target in x y plane;

@@姿态，以弧度为单位，以目标为中心，在xy平面内旋转@@

camera.rotationOffset = 0;

acceleration in moving from current to goal position;

@@从当前位置到目标位置的移动加速度@@

camera.cameraAcceleration = 0.005

speed at which acceleration is halted

@@最大速度@@

camera.maxCameraSpeed = 10

Of course we also attach the camera to the canvas

@@当然，我们还要把相机绑定在画布上@@

camera.attachControl(canvas, true);

Finally we set the target

@@最后我们设置目标网格@@

camera.lockedTarget = targetMesh;

Follow The Character

跟随角色

https://playground.babylonjs.com/#KBS9I5#98

Having created a 3D world, albeit just a small village, it would be good to see it in all its 3D glory. All you need is a 3D viewer whether a simple one such as the Google Carboard or a high tech one and the appropriate camera from Babylon.js.

@@建立了一个3D世界之后，即使只是一个小村庄，你最好尝试使用各种方式浏览这个世界。你所需要的是一个虚拟现实观察设备，它可能是一个简单的设备，比如谷歌纸板，也可能是一个高科技设备，以及Babylon.js中的一种合适的相机@@

const createScene = function () {

    const scene = new BABYLON.Scene(engine);

    // This creates and initially positions a follow camera

    const camera = new BABYLON.FollowCamera("FollowCam", new BABYLON.Vector3(-6, 0, 0), scene);

    //The goal distance of camera from target

    camera.radius = 1;

    // The goal height of camera above local oriin (centre) of target

    camera.heightOffset = 8;

    // The goal rotation of camera around local origin (centre) of target in x y plane

    camera.rotationOffset = 0;

    //Acceleration of camera in moving from current to goal position

    camera.cameraAcceleration = 0.005

    //The speed at which acceleration is halted

    camera.maxCameraSpeed = 10

    //camera.target is set after the target's creation

    // This attaches the camera to the canvas

    camera.attachControl(canvas, true);

    const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(1, 1, 0));

    const walk = function (turn, dist) {

        this.turn = turn;

        this.dist = dist;

    }

    const track = [];

    track.push(new walk(86, 7));

    track.push(new walk(-85, 14.8));

    track.push(new walk(-93, 16.5));

    track.push(new walk(48, 25.5));

    track.push(new walk(-112, 30.5));

    track.push(new walk(-72, 33.2));

    track.push(new walk(42, 37.5));

    track.push(new walk(-98, 45.2));

    track.push(new walk(0, 47))

    // Dude

    BABYLON.SceneLoader.ImportMeshAsync("him", "/scenes/Dude/", "Dude.babylon", scene).then((result) => {

        var dude = result.meshes[0];

        dude.scaling = new BABYLON.Vector3(0.008, 0.008, 0.008);

        dude.position = new BABYLON.Vector3(-6, 0, 0);

        dude.rotate(BABYLON.Axis.Y, BABYLON.Tools.ToRadians(-95), BABYLON.Space.LOCAL);

        const startRotation = dude.rotationQuaternion.clone();

        camera.lockedTarget = dude;

        scene.beginAnimation(result.skeletons[0], 0, 100, true, 1.0);

        let distance = 0;

        let step = 0.015;

        let p = 0;

        scene.onBeforeRenderObservable.add(() => {

            dude.movePOV(0, 0, step);

            distance += step;

            if (distance > track[p].dist) {

                dude.rotate(BABYLON.Axis.Y, BABYLON.Tools.ToRadians(track[p].turn), BABYLON.Space.LOCAL);

                p +=1;

                p %= track.length;

                if (p === 0) {

                    distance = 0;

                    dude.position = new BABYLON.Vector3(-6, 0, 0);

                    dude.rotationQuaternion = startRotation.clone();

                }

            }

        })

    });

    const spriteManagerTrees = new BABYLON.SpriteManager("treesManager", "textures/palm.png", 2000, {width: 512, height: 1024}, scene);

    //We create trees at random positions

    for (let i = 0; i < 500; i++) {

        const tree = new BABYLON.Sprite("tree", spriteManagerTrees);

        tree.position.x = Math.random() \* (-30);

        tree.position.z = Math.random() \* 20 + 8;

        tree.position.y = 0.5;

    }

    for (let i = 0; i < 500; i++) {

        const tree = new BABYLON.Sprite("tree", spriteManagerTrees);

        tree.position.x = Math.random() \* (25) + 7;

        tree.position.z = Math.random() \* -35  + 8;

        tree.position.y = 0.5;

    }

    //Skybox

    const skybox = BABYLON.MeshBuilder.CreateBox("skyBox", {size:100}, scene);

      const skyboxMaterial = new BABYLON.StandardMaterial("skyBox", scene);

      skyboxMaterial.backFaceCulling = false;

      skyboxMaterial.reflectionTexture = new BABYLON.CubeTexture("textures/skybox", scene);

      skyboxMaterial.reflectionTexture.coordinatesMode = BABYLON.Texture.SKYBOX\_MODE;

      skyboxMaterial.diffuseColor = new BABYLON.Color3(0, 0, 0);

      skyboxMaterial.specularColor = new BABYLON.Color3(0, 0, 0);

      skybox.material = skyboxMaterial;

    BABYLON.SceneLoader.ImportMeshAsync("", "https://assets.babylonjs.com/meshes/", "valleyvillage.glb");

    return scene;

};