**Getting Started - Chapter 5 - Distant Hills**

**@@启程-第五章-远山@@**

**Distant Hills**

**@@远山@@**

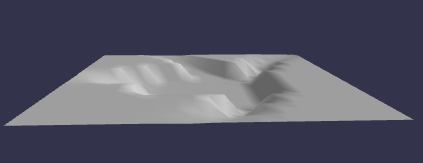
We want to set the village in a valley. It would be possible to create hills from meshes, however there is another way to add vertical height to a ground mesh. This is achieved using a height map which uses shades of gray to determine the height of the ground. White areas are the highest parts and black the lowest. This simple height map

@@我们希望把村庄放置在一个山谷里。当然，通过网格建立丘陵是可行的，但是还有一种为地面网格添加垂直高度的方法。这种方法是用“高度图”实现的，它用高度图中像素的灰度决定地面相应顶点的高度。这时白色的区域将是高度最高的区域，黑色的区域是高度最低的区域。这是一张简单的高度图：@@



has a large black area in the middle to host the village, the white areas create the hills and the gray ones ways out of the valley for roads.

@@图的中心有一片大的黑色区域用来放置村庄，白色的区域产生山峦，灰色的区域则是离开村庄的道路。@@



In this image the camera has been pulled further out and the vertical height exaggerated.

@@在这张图里相机被拉远，而垂直高度被加大了。@@

const largeGround = BABYLON.MeshBuilder.CreateGroundFromHeightMap("largeGround", "url to height map",

{width:150, height:150, subdivisions: 20, minHeight:0, maxHeight: 10});

The subdivisions property of *options* splits the ground into 20 x 20 = 400 sections. The more subdivisions the finer gradation for height calculation. The two properties minHeight and maxHeight determine the vertical heights for the black and white areas respectively, gray areas scaled accordingly.

@@“设置”参数的“细分度”属性是20，这表示地面将被分为20\*20=400个区域。细分度越高则在高度计算上有更佳的分层（译者注：但地面网格的顶点结构决定了，再怎么提高细分度也无法十分精细）。“最低高度”和“最高高度”两个属性分别决定黑色区域和白色区域的垂直高度，灰色区域则按比例进行插值。@@

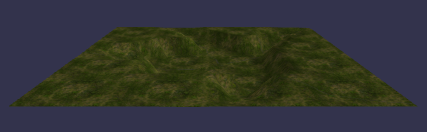
Basic Height Map

基础高度图

https://playground.babylonjs.com/#KBS9I5#39

We can add textures in the usual way.

我们可以用常规方式添加纹理。



Height Map With Texture

具有纹理的高度图

[https://playground.babylonjs.com/#KBS9I5#40](https://playground.babylonjs.com/#40)

（译者注：仔细观察可以发现地形起伏处的纹理被拉伸变形，这是Babylon.js地面网格的纹理坐标算法决定的）

Finally we give the village ground a similar, but clearer texture, to that of the large ground.

@@最后我们给村庄的地面一个小一点，但是更加清晰的纹理，与外面更大的地面网格相比@@

//Create Village ground 建立村庄内的地面网格

const groundMat = new BABYLON.StandardMaterial("groundMat");

groundMat.diffuseTexture = new BABYLON.Texture("url to ground texture");

groundMat.diffuseTexture.hasAlpha = true;

const ground = BABYLON.MeshBuilder.CreateGround("ground", {width:24, height:24});

ground.material = groundMat;

//large ground 更大的地面（村庄周围的山）

const largeGroundMat = new BABYLON.StandardMaterial("largeGroundMat");

largeGroundMat.diffuseTexture = new BABYLON.Texture("url to large ground texture");

const largeGround = BABYLON.MeshBuilder.CreateGroundFromHeightMap("largeGround", "url to heightmap",

{width:150, height:150, subdivisions: 20, minHeight:0, maxHeight: 4});

largeGround.material = largeGroundMat;

largeGround.position.y = -0.01;

This line 这一行

largeGround.position.y = -0.01;

ensures the two grounds do not fight and cause flickering.

@@确保这两个地面网格不会争抢位置并引起闪烁。@@

Layering Textures

多层地面纹理

[https://playground.babylonjs.com/#KBS9I5#85](https://playground.babylonjs.com/#85)

We add back the dwellings and create a file to import

@@我们把房子添加回来，然后把这个场景保存为一个模型文件。@@

Adding The Dwellings

添加房子

https://playground.babylonjs.com/#KBS9I5#86

Then we can add back the car but this time passing through the village.

@@接下来我们可以把小车加回来，但是这一次小车是穿过村庄的。@@

Adding The Car

添加小车

https://playground.babylonjs.com/#KBS9I5#87

Let's now further improve the environment by adding a sky.

@@接下来让我们通过添加天空来进一步改进环境。@@

const createScene = function () {

    const scene = new BABYLON.Scene(engine);

    const camera = new BABYLON.ArcRotateCamera("camera", -Math.PI / 2, Math.PI / 2.5, 15, new BABYLON.Vector3(0, 0, 0));

    camera.attachControl(canvas, true);

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

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

    BABYLON.SceneLoader.ImportMeshAsync("", "https://assets.babylonjs.com/meshes/", "car.glb").then(() => {

        const car = scene.getMeshByName("car");

        car.rotation = new BABYLON.Vector3(Math.PI / 2, 0, -Math.PI / 2);

        car.position.y = 0.16;

        car.position.x = -3;

        car.position.z = 8;

        const animCar = new BABYLON.Animation("carAnimation", "position.z", 30, BABYLON.Animation.ANIMATIONTYPE\_FLOAT, BABYLON.Animation.ANIMATIONLOOPMODE\_CYCLE);

      const carKeys = [];

      carKeys.push({

        frame: 0,

        value: 10

      });

      carKeys.push({

        frame: 200,

        value: -15

      });

      animCar.setKeys(carKeys);

      car.animations = [];

      car.animations.push(animCar);

      scene.beginAnimation(car, 0, 200, true);

      //wheel animation

      const wheelRB = scene.getMeshByName("wheelRB");

      const wheelRF = scene.getMeshByName("wheelRF");

      const wheelLB = scene.getMeshByName("wheelLB");

      const wheelLF = scene.getMeshByName("wheelLF");

      scene.beginAnimation(wheelRB, 0, 30, true);

      scene.beginAnimation(wheelRF, 0, 30, true);

      scene.beginAnimation(wheelLB, 0, 30, true);

      scene.beginAnimation(wheelLF, 0, 30, true);

    });

    return scene;

};