**Getting Started - Chapter 3 - Mesh Parents**

**@@启程-第三章-父元素@@**

**Parents and Children**

**@@父元素与子元素@@**

We are going to add a very simple car that moves through the village.

@@我们将添加一个穿过村庄的非常简单的小车。@@

No matter how simple a car needs wheels and we have to combine the body of the car and the wheels.

@@不论是多简单的车都需要轮子，并且我们必须把车体和轮子合并起来。@@



Using merge meshes to combine them would result in the wheels not being able to rotate. Instead we set the body of the car as a parent of each wheel.

@@使用“合并网格”方法合并它们将导致轮子无法旋转。取而代之，我们将车体设置为每个轮子的父元素@@

Before building our simple car let's take a way to set a parent and what this means.

@@在建造我们简单的小车之前，让我们先了解如何设置父元素以及父元素意味着什么。@@

messhChild.parent = meshParent

Any use of position, scaling and rotation on the parent will also be applied to the child. Setting the position of the child is done in the parent space, setting the rotation and scale of the child takes place in the child's local space.

@@对父元素位置、缩放、旋转属性的任何修改，都将被应用在子元素上。对子元素的位置设置则在父元素的局部坐标系中生效，对旋转和缩放的设置则将发生在子元素的局部坐标系中@@

You can vary the values in the following playground to see the effect on the parent and child.

@@你可以在下面的训练场中修改这些值，观察它们对父元素和子元素的影响。@@

Understanding Parenting

理解父元素

https://playground.babylonjs.com/#GMEI6U

Now we are ready to build the car and afterwards to animate it.

@@现在我们已经准备好建立小车，然后为它添加动画。@@

You can read more about [parenting](https://doc.babylonjs.com/divingDeeper/mesh/transforms/parent_pivot).

@@你可以在这里阅读更多关于父元素的内容@@

https://doc.babylonjs.com/divingDeeper/mesh/transforms/parent\_pivot

**Further reading 延伸阅读**

Parents and Pivots 父元素与基准点

Learn all about parent relationships and pivots in Babylon.js.

@@学习Babylon.js所有关于父元素关系和基准点的内容。@@

https://doc.babylonjs.com/divingDeeper/mesh/transforms/parent\_pivot

const createScene =  () => {

    const scene = new BABYLON.Scene(engine);

    const camera = new BABYLON.ArcRotateCamera("camera", -Math.PI / 2.2, Math.PI / 2.5, 15, new BABYLON.Vector3(0, 0, 0));

    camera.attachControl(canvas, true);

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

    const faceColors = [];

    faceColors[0] = BABYLON.Color3.Blue();

    faceColors[1] = BABYLON.Color3.Teal()

    faceColors[2] = BABYLON.Color3.Red();

    faceColors[3] = BABYLON.Color3.Purple();

    faceColors[4] = BABYLON.Color3.Green();

    faceColors[5] = BABYLON.Color3.Yellow();

    const boxParent = BABYLON.MeshBuilder.CreateBox("Box", {faceColors:faceColors});

    const boxChild = BABYLON.MeshBuilder.CreateBox("Box", {size: 0.5, faceColors:faceColors});

    boxChild.setParent(boxParent);

    boxChild.position.x = 0;

    boxChild.position.y = 2;

    boxChild.position.z = 0;

    boxChild.rotation.x = Math.PI / 4;

    boxChild.rotation.y = Math.PI / 4;

    boxChild.rotation.z = Math.PI / 4;

    boxParent.position.x = 2;

    boxParent.position.y = 0;

    boxParent.position.z = 0;

    boxParent.rotation.x = 0;

    boxParent.rotation.y = 0;

    boxParent.rotation.z = -Math.PI / 4;

    const boxChildAxes = localAxes(1, scene);

    boxChildAxes.parent = boxChild;

    showAxis(6, scene);

    return scene;

}

/\*\*\*\*\*\*\*\*\*\*\*Create and Draw Axes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const showAxis = (size, scene) => {

    const makeTextPlane = (text, color, size) => {

        const dynamicTexture = new BABYLON.DynamicTexture("DynamicTexture", 50, scene, true);

        dynamicTexture.hasAlpha = true;

        dynamicTexture.drawText(text, 5, 40, "bold 36px Arial", color , "transparent", true);

        const plane = new BABYLON.Mesh.CreatePlane("TextPlane", size, scene, true);

        plane.material = new BABYLON.StandardMaterial("TextPlaneMaterial", scene);

        plane.material.backFaceCulling = false;

        plane.material.specularColor = new BABYLON.Color3(0, 0, 0);

        plane.material.diffuseTexture = dynamicTexture;

        return plane;

    };

    const axisX = BABYLON.Mesh.CreateLines("axisX", [

        new BABYLON.Vector3.Zero(), new BABYLON.Vector3(size, 0, 0), new BABYLON.Vector3(size \* 0.95, 0.05 \* size, 0),

        new BABYLON.Vector3(size, 0, 0), new BABYLON.Vector3(size \* 0.95, -0.05 \* size, 0)

    ]);

    axisX.color = new BABYLON.Color3(1, 0, 0);

    const xChar = makeTextPlane("X", "red", size / 10);

    xChar.position = new BABYLON.Vector3(0.9 \* size, -0.05 \* size, 0);

    const axisY = BABYLON.Mesh.CreateLines("axisY", [

        new BABYLON.Vector3.Zero(), new BABYLON.Vector3(0, size, 0), new BABYLON.Vector3( -0.05 \* size, size \* 0.95, 0),

        new BABYLON.Vector3(0, size, 0), new BABYLON.Vector3( 0.05 \* size, size \* 0.95, 0)

    ]);

    axisY.color = new BABYLON.Color3(0, 1, 0);

    const yChar = makeTextPlane("Y", "green", size / 10);

    yChar.position = new BABYLON.Vector3(0, 0.9 \* size, -0.05 \* size);

    const axisZ = BABYLON.Mesh.CreateLines("axisZ", [

        new BABYLON.Vector3.Zero(), new BABYLON.Vector3(0, 0, size), new BABYLON.Vector3( 0 , -0.05 \* size, size \* 0.95),

        new BABYLON.Vector3(0, 0, size), new BABYLON.Vector3( 0, 0.05 \* size, size \* 0.95)

    ]);

    axisZ.color = new BABYLON.Color3(0, 0, 1);

    const zChar = makeTextPlane("Z", "blue", size / 10);

    zChar.position = new BABYLON.Vector3(0, 0.05 \* size, 0.9 \* size);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Local Axes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

localAxes = (size, scene) => {

    const local\_axisX = BABYLON.Mesh.CreateLines("local\_axisX", [

        new BABYLON.Vector3.Zero(), new BABYLON.Vector3(size, 0, 0), new BABYLON.Vector3(size \* 0.95, 0.05 \* size, 0),

        new BABYLON.Vector3(size, 0, 0), new BABYLON.Vector3(size \* 0.95, -0.05 \* size, 0)

    ], scene);

    local\_axisX.color = new BABYLON.Color3(1, 0, 0);

    local\_axisY = BABYLON.Mesh.CreateLines("local\_axisY", [

        new BABYLON.Vector3.Zero(), new BABYLON.Vector3(0, size, 0), new BABYLON.Vector3(-0.05 \* size, size \* 0.95, 0),

        new BABYLON.Vector3(0, size, 0), new BABYLON.Vector3(0.05 \* size, size \* 0.95, 0)

    ], scene);

    local\_axisY.color = new BABYLON.Color3(0, 1, 0);

    const local\_axisZ = BABYLON.Mesh.CreateLines("local\_axisZ", [

        new BABYLON.Vector3.Zero(), new BABYLON.Vector3(0, 0, size), new BABYLON.Vector3(0, -0.05 \* size, size \* 0.95),

        new BABYLON.Vector3(0, 0, size), new BABYLON.Vector3(0, 0.05 \* size, size \* 0.95)

    ], scene);

    local\_axisZ.color = new BABYLON.Color3(0, 0, 1);

    const local\_origin = new BABYLON.TransformNode("local\_origin");

    local\_axisX.parent = local\_origin;

    local\_axisY.parent = local\_origin;

    local\_axisZ.parent = local\_origin;

    return local\_origin;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End Local Axes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/