**Unity3D人物行走脚本示例**

Posted on 2013年03月15日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 569 次

[**Unity3D**](http://www.unitymanual.com)人物行走脚本示例。

新建一个C＃的脚本文件，然后将下面的代码灌入其中，文件名为FPSWalker\_edit02.cs。然后将官方提供的First Person Controller中的所有脚本删除，再链入这个脚本即可。经过测试，在IE和火狐中都可以正常使用。

脚本如下：

using UnityEngine;  
using System.Collections;  
public class FPSWalker\_edit02 : MonoBehaviour  
{  
///  
/// 类似于VRML的控制方式  
/// ↑前进 ↓后退 →右转 ←左转  
/// Ctrl + →右平移 Ctrl + ←左平移  
/// 按住鼠标左键，可以通过鼠标上下左右转动视角  
///  
public float speed = 6.0f;  
public float jumpSpeed = 8.0f;  
public float gravity = 20.0f;  
public bool MouseChange = true;  
private Vector3 moveDirection = Vector3.zero;  
private bool grounded = false;  
void FixedUpdate()  
{  
if (grounded)  
{  
if (!(Input.GetKey(KeyCode.LeftControl)||Input.GetKey(KeyCode.RightControl))) //如果没有按下左Ctrl键  
{  
///只能前后平移  
moveDirection = new Vector3(0, 0, Input.GetAxis("Vertical"));  
moveDirection = transform.TransformDirection(moveDirection);  
moveDirection \*= speed;  
}  
else //  
{  
///可前后左右平移  
moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));  
moveDirection = transform.TransformDirection(moveDirection);  
moveDirection \*= speed;  
}  
if (Input.GetButton("Jump")) //跳跃  
{  
moveDirection.y = jumpSpeed;  
}  
}  
//重力  
moveDirection.y -= gravity \* Time.deltaTime;  
//移动controller  
CharacterController controller = GetComponent("CharacterController") as CharacterController;  
CollisionFlags flags = controller.Move(moveDirection \* Time.deltaTime);  
grounded = (flags & CollisionFlags.CollidedBelow) != 0; //当controller处在空中间，grounded为false，即跳动和行走都无效  
///鼠标控制视角  
///  
// if (Input.GetMouseButton(0) && (Input.GetKey(KeyCode.LeftControl) || Input.GetKey(KeyCode.RightControl)) && MouseChange) //如果按下鼠标左键并且鼠标MouseChange为真  
if (Input.GetMouseButton(0) && MouseChange) //如果按下鼠标左键并且鼠标MouseChange为真  
{  
///鼠标旋转视角部分  
///  
if (axes == RotationAxes.MouseXAndY)  
{  
// Read the mouse input axis  
//这里，rotationX和rotationY用来保存对象现有的角度，同时还将鼠标的移动中计算出增减的角度并合进来  
rotationX += Input.GetAxis("Mouse X") \* sensitivityX;  
rotationY += Input.GetAxis("Mouse Y") \* sensitivityY;  
rotationX = ClampAngle(rotationX, minimumX, maximumX);  
rotationY = ClampAngle(rotationY, minimumY, maximumY);  
Quaternion xQuaternion = Quaternion.AngleAxis(rotationX, Vector3.up); //通过左右值和Vector3.up（作为以Y为旋转轴的向量值）求出左右旋转度的四元数值  
Quaternion yQuaternion = Quaternion.AngleAxis(rotationY, Vector3.left); //通过上下值和Vector3.left（作为以X为旋转轴的向量值）求出上下旋转度的四元数值  
//originalRotation = transform.localRotation;  
transform.localRotation = originalRotation \* xQuaternion \* yQuaternion; //将上面求出来的左右和上下两个四元数值添加入角度中  
}  
else if (axes == RotationAxes.MouseX)  
{  
rotationX += Input.GetAxis("Mouse X") \* sensitivityX;  
rotationX = ClampAngle(rotationX, minimumX, maximumX);  
Quaternion xQuaternion = Quaternion.AngleAxis(rotationX, Vector3.up);  
transform.localRotation = originalRotation \* xQuaternion;  
}  
else  
{  
rotationY += Input.GetAxis("Mouse Y") \* sensitivityY;  
rotationY = ClampAngle(rotationY, minimumY, maximumY);  
Quaternion yQuaternion = Quaternion.AngleAxis(rotationY, Vector3.left);  
transform.localRotation = originalRotation \* yQuaternion;  
}  
}  
else  
{  
///左右旋转  
///并且没有按下左或右ctrl键时  
if (!(Input.GetKey(KeyCode.LeftControl) || Input.GetKey(KeyCode.RightControl)))  
{  
Vector3 angle\_temp = transform.eulerAngles;  
angle\_temp.y += Input.GetAxis("Horizontal") \* sensitivityX \* 0.3f;  
rotationX = ClampAngle(angle\_temp.y, minimumX, maximumX);  
//rotationY = ClampAngle(rotationY, minimumY, maximumY);  
transform.eulerAngles = angle\_temp;  
}  
//键盘上下键控制俯仰角  
//else  
//{  
// Vector3 angle\_temp = transform.eulerAngles;  
// angle\_temp.x += Input.GetAxis("Vertical") \* -1 \* sensitivityY \* 0.3f;  
// rotationY = ClampAngle(angle\_temp.x , minimumY , maximumY);  
// transform.eulerAngles = angle\_temp;  
//}  
}  
}  
public enum RotationAxes { MouseXAndY = 0, MouseX = 1, MouseY = 2 }  
public RotationAxes axes = RotationAxes.MouseXAndY;  
public float sensitivityX = 15F;  
public float sensitivityY = 15F;  
public float minimumX = -360F;  
public float maximumX = 360F;  
public float minimumY = -60F;  
public float maximumY = 60F;  
public float rotationX = 0F;  
public float rotationY = 0F;  
Quaternion originalRotation;  
void Start()  
{  
// Make the rigid body not change rotation  
//使刚体不会改变角度  
if (rigidbody)  
rigidbody.freezeRotation = true;  
originalRotation = transform.localRotation;  
}  
public static float ClampAngle(float angle, float min, float max)  
{  
if (angle < -360F)  
angle += 360F;  
if (angle > 360F)  
angle -= 360F;  
return Mathf.Clamp(angle, min, max);  
}  
}