**行为控制脚本**

Posted on 2013年05月24日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 164 次

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **using** UnityEngine; |
| 02 | **using** System.Collections; |
| 03 | **public** **class** FPSWalker\_edit02 : MonoBehaviour |
| 04 | { |
| 05 | */// 类似于VRML的控制方式* |
| 06 | */// ↑前进 ↓后退 →右转 ←左转* |
| 07 | */// Ctrl + →右平移 Ctrl + ←左平移* |
| 08 | */// 按住鼠标左键，可以通过鼠标上下左右转动视角* |
| 09 | **public** **float** speed = 6.0f; |
| 10 | **public** **float** jumpSpeed = 8.0f; |
| 11 | **public** **float** gravity = 20.0f; |
| 12 | **public** **bool** MouseChange = **true**; |
| 13 | **private** Vector3 moveDirection = Vector3.zero; |
| 14 | **private** **bool** grounded = **false**;Unity3D教程手册 |
| 15 | **void** FixedUpdate() |
| 16 | { |
| 17 | **if** (grounded) |
| 18 | { |
| 19 | **if** (!(Input.GetKey(KeyCode.LeftControl)||Input.GetKey(KeyCode.RightControl))) *//如果没有按下左Ctrl键* |
| 20 | { |
| 21 | *//只能前后平移* |
| 22 | moveDirection = new Vector3(0, 0, Input.GetAxis("Vertical")); |
| 23 | moveDirection = transform.TransformDirection(moveDirection); |
| 24 | moveDirection \*= speed; |
| 25 | } |
| 26 | **else** *//* |
| 27 | { |
| 28 | *//可前后左右平移* |
| 29 | moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical")); |
| 30 | moveDirection = transform.TransformDirection(moveDirection); |
| 31 | moveDirection \*= speed; |
| 32 | } |
| 33 | **if** (Input.GetButton("Jump")) *//跳跃* |
| 34 | { |
| 35 | moveDirection.y = jumpSpeed; |
| 36 | } |
| 37 | } |
| 38 | *//重力* |
| 39 | moveDirection.y -= gravity \* Time.deltaTime; |
| 40 | *//移动controller* |
| 41 | CharacterController controller = GetComponent("CharacterController") **as** CharacterController; |
| 42 | CollisionFlags flags = controller.Move(moveDirection \* Time.deltaTime); |
| 43 | grounded = (flags & CollisionFlags.CollidedBelow) != 0; *//当controller处在空中间，grounded为false，即跳动和行走都无效* |
| 44 | *//鼠标控制视角* |
| 45 | *// 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* |
| 46 | **if** (Input.GetMouseButton(0) && MouseChange) *//如果按下鼠标左键并且鼠标MouseChange为真* |
| 47 | { |
| 48 | *///鼠标旋转视角部分* |
| 49 | *///* |
| 50 | **if** (axes == RotationAxes.MouseXAndY)<code lang="csharp">Unity3D教程手册 |

{  
// 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; //BUG处  
originalRotation = new Quaternion(0f, 0f, 0f, 1f); //修正代码  
}  
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);  
}  
}