**双摇杆设计思路**

Posted on 2013年05月27日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 4 次

脚本JoyStick.cs挂在摇杆上并设置相关参数。

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

|  |  |
| --- | --- |
| 01 | **using** UnityEngine; |
| 02 | **using** System.Collections; |
| 03 |  |
| 04 | **public** **enum** JoyStickType |
| 05 | { |
| 06 | leftJoyStick, |
| 07 | rightJoyStick |
| 08 | } |
| 09 |  |
| 10 | **public** **class** JoyStick : MonoBehaviour { |
| 11 | **public** JoyStickType joyStickType;*//摇杆类型，左摇杆还是右摇杆* |
| 12 | **public** Vector2 centerPos;*//摇杆的中心点位置，世界坐标* |
| 13 | **public** Transform centerBall;*//摇杆中心球* |
| 14 | **public** **float** joyStickRadius;*//摇杆的半径，世界空间中的半径* |
| 15 | **private** **int** lastFingerID = -1;*//最后一次触摸的手指id* |
| 16 | **private** **bool** centerBallMoving = **false**;*//摇杆中心球移动开关* |
| 17 | *// Use this for initialization* |
| 18 | **void** Start () { |
| 19 | } |
| 20 | *// Update is called once per frame* |
| 21 | **void** Update () { |
| 22 | JoyStickController(); |
| 23 | } |
| 24 | **void** JoyStickController() |
| 25 | { |
| 26 | **int** count = Input.touchCount;*//获取触摸数量* |
| 27 | **for** (**int** i = 0; i < count; i++)*//逐个分析触摸* |
| 28 | { |
| 29 | Touch touch = Input.GetTouch(i);Unity3D教程手册 |
| 30 | *//将当前的触摸位置屏幕转世界坐标* |
| 31 | Vector2 currentTouchPos = new Vector2(Camera.main.ScreenToWorldPoint(touch.position).x, Camera.main.ScreenToWorldPoint(touch.position).y); |
| 32 | Vector2 temp = currentTouchPos - centerPos;*//得到方向向量temp（触摸转换后的位置和摇杆）* |
| 33 | **if** (touch.phase == TouchPhase.Began) |
| 34 | { |
| 35 | **if** (Mathf.Round(temp.magnitude) <= joyStickRadius)*//如果方向向量temp的长度没有超出摇杆的半径* |
| 36 | { |
| 37 | lastFingerID = touch.fingerId;*//记录该触摸的id* |
| 38 | centerBallMoving = **true**;*//摇杆中心球移动开关打开* |
| 39 | } |
| 40 | } |
| 41 | *//若中心球移动开关打开，摇杆中心球就会跟随手指移动。但需要加以限制，当手指触摸没有超出摇杆的圆形区域时，中心球完全跟随手指触摸；* |
| 42 | *//当手指触摸超出圆形区域时，中心球处于触摸位置和摇杆中心点所形成的方向上并且不能超出半径* |
| 43 | **if** (touch.fingerId == lastFingerID && centerBallMoving) |
| 44 | { |
| 45 | **if** (Mathf.Round(temp.magnitude) <= joyStickRadius) *//如果手指触摸没有超出摇杆的圆形区域，即摇杆半径，摇杆中心球的位置一直跟随手指* |
| 46 | { |
| 47 | centerBall.transform.position = Camera.main.ScreenToWorldPoint(new Vector3(touch.position.x, touch.position.y, Camera.main.WorldToScreenPoint(centerBall.transform.position).z)); |
| 48 | } |
| 49 | **else** |
| 50 | { |
| 51 | temp.Normalize(); |
| 52 | *//由于摇杆中心球是在一个平面上移动的，本人做法是中心球位置的z值固定为18，改变x,y值，大家可以视情况做一些修改* |
| 53 | centerBall.transform.position = new Vector3((temp \* joyStickRadius + centerPos).x, (temp \* joyStickRadius + centerPos).y, 18); |
| 54 | } |
| 55 |  |
| 56 | **if** (temp.x >= 0)*//0-180* |
| 57 | { |
| 58 | *//一下为示例代码：控制旋转方向，主要利用Vector2.Angle(temp, new Vector2(0, 5))得到角度并利用* |
| 59 | *//initialization\_script.current\_player\_tank\_script.BodyRotation(Vector2.Angle(temp, new Vector2(0, 5)));* |
| 60 | } |
| 61 | **if** (temp.x< 0)*//180-360* |
| 62 | { |
| 63 | *//一下为示例代码：控制旋转方向，主要利用Vector2.Angle(temp, new Vector2(0, 5))得到角度并利用* |
| 64 | *//initialization\_script.current\_player\_tank\_script.BodyRotation(-1 \* Vector2.Angle(temp1, new Vector2(0, 5)));* |
| 65 | } |
| 66 | *//控制移动的函数或者控制开火的函数，假设左摇杆控制移动，右摇杆控制开火* |
| 67 | **switch**(joyStickType) |
| 68 | { |
| 69 | **case** JoyStickType.leftJoyStick: |
| 70 | *//Move();* |
| 71 | **break**; |
| 72 | **case** JoyStickType.rightJoyStick: |
| 73 | *//Fire()* |
| 74 | **break**; |
| 75 | } |
| 76 | *//当释放触摸的时候中心球位置重置* |
| 77 | **if** (touch.phase == TouchPhase.Ended || touch.phase == TouchPhase.Canceled) |
| 78 | { |
| 79 | centerBall.transform.position = new Vector3(centerPos.x, centerPos.y, 18); |
| 80 | *// 示例代码，视情况自己添加 initialization\_script.current\_player\_tank\_script.CoverRotation(initialization\_script.current\_player\_tank.transform.eulerAngles.y);* |
| 81 | centerBallMoving = **false**; |
| 82 | lastFingerID = -1; |
| 83 | } |
| 84 | } |
| 85 | } |
| 86 | } |
| 87 | } |