**角色控制**

Posted on 2013年04月27日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 169 次

从官方的范例里面抽出来的程式，还没有改完成，但可以参考，程式复杂已失去重力，所以脚色不会掉落，直接拖曳到物件上就可以使用，但必须先设定角色碰撞。

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

|  |  |
| --- | --- |
| 01 | @script RequireComponent(Rigidbody) |
| 02 |  |
| 03 | **public** **var** movementDirection : Vector3 = Vector3(1,0,0); |
| 04 | **public** **var** movementTarget : Vector3; |
| 05 | **public** **var** facingDirection : Vector3; |
| 06 | **public** **var** walkingSpeed: **float** = 5.0; |
| 07 | **public** **var** walkingSnappyness: **float** = 50; *//走路的節拍* |
| 08 | **public** **var** turningSmoothing: **float** = 0.3; *//平滑轉動* |
| 09 | **private** **var** character: Transform; |
| 10 | **public** **var** cursorPlaneHeight: **float** = 0; |
| 11 | **private** **var** mainCamera: Camera; |
| 12 | **private** **var** mainCameraTransform: Transform; |
| 13 | **private** **var** playerMovementPlane: Plane; |
| 14 | **private** **var** screenMovementSpace: Quaternion; |
| 15 | **private** **var** screenMovementForward: Vector3; |
| 16 | **private** **var** screenMovementRight: Vector3; |
| 17 |  |
| 18 | function Awake() { |
| 19 | movementDirection = Vector2.zero; |
| 20 | facingDirection = Vector2.zero; |
| 21 | mainCamera = Camera.main; |
| 22 | mainCameraTransform = mainCamera.transform; |
| 23 | **if** (!character) character = transform; |
| 24 | playerMovementPlane = new Plane(character.up, character.position + character.up \* cursorPlaneHeight); |
| 25 | } |
| 26 |  |
| 27 | function Start() { |
| 28 | screenMovementSpace = Quaternion.Euler(0, mainCameraTransform.eulerAngles.y, 0); |
| 29 | screenMovementForward = screenMovementSpace \* Vector3.forward; |
| 30 | screenMovementRight = screenMovementSpace \* Vector3.right; |
| 31 | } |
| 32 |  |
| 33 | function Update() { |
| 34 | movementDirection = Input.GetAxis("Horizontal") \* screenMovementRight + Input.GetAxis("Vertical") \* screenMovementForward; |
| 35 | **if** (movementDirection.sqrMagnitude > 1) movementDirection.Normalize(); |
| 36 | playerMovementPlane.normal = character.up; |
| 37 | playerMovementPlane.distance = -character.position.y + cursorPlaneHeight; |
| 38 |  |
| 39 | **var** cameraAdjustmentVector: Vector3 = Vector3.zero; |
| 40 | cameraAdjustmentVector = facingDirection; |
| 41 |  |
| 42 | **var** cursorWorldPosition: Vector3 = ScreenPointToWorldPointOnPlane(Input.mousePosition, playerMovementPlane, mainCamera); |
| 43 | facingDirection = (cursorWorldPosition - character.position); |
| 44 | facingDirection.y = 0; |
| 45 | } |
| 46 |  |
| 47 | **public** **static** |
| 48 | function PlaneRayIntersection(plane: Plane, |
| 49 | ray: Ray): Vector3 { |
| 50 | **var** dist: **float**; |
| 51 | plane.Raycast(ray, dist); |
| 52 | **return** ray.GetPoint(dist); |
| 53 | } |
| 54 |  |
| 55 | **public** **static** |
| 56 | function ScreenPointToWorldPointOnPlane(screenPoint: Vector3, |
| 57 | plane: Plane, |
| 58 | camera: Camera): Vector3 { |
| 59 | **var** ray: Ray = camera.ScreenPointToRay(screenPoint); |
| 60 | **return** PlaneRayIntersection(plane, ray); |
| 61 | } |
| 62 |  |
| 63 | function FixedUpdate() { |
| 64 |  |
| 65 | *// Handle the movement of the character* |
| 66 | **var** targetVelocity: Vector3 = movementDirection \* walkingSpeed; |
| 67 | **var** deltaVelocity: Vector3 = targetVelocity - rigidbody.velocity; |
| 68 | **if** (rigidbody.useGravity) deltaVelocity.y = 0; |
| 69 | rigidbody.AddForce(deltaVelocity \* walkingSnappyness, ForceMode.Acceleration); |
| 70 |  |
| 71 | *// 設定玩家面對的方向，或如果是零，則(面對)運動方向* |
| 72 | **var** faceDir: Vector3 = facingDirection; |
| 73 | **if** (faceDir == Vector3.zero) faceDir = movementDirection; |
| 74 | *// 使人物轉動朝著目標的旋轉* |
| 75 | **if** (faceDir == Vector3.zero) { |
| 76 | rigidbody.angularVelocity = Vector3.zero; |
| 77 | } **else** { |
| 78 | **var** rotationAngle: **float** = AngleAroundAxis(transform.forward, faceDir, Vector3.up); |
| 79 | rigidbody.angularVelocity = (Vector3.up \* rotationAngle \* turningSmoothing); |
| 80 | } |
| 81 | } |
| 82 |  |
| 83 | *// The angle between dirA and dirB around axis* |
| 84 | **static** function AngleAroundAxis(dirA: Vector3, dirB: Vector3, axis: Vector3) { |
| 85 | *// Project A and B onto the plane orthogonal target axis* |
| 86 | dirA = dirA - Vector3.Project(dirA, axis); |
| 87 | dirB = dirB - Vector3.Project(dirB, axis); |
| 88 | *// Find (positive) angle between A and B* |
| 89 | **var** angle: **float** = Vector3.Angle(dirA, dirB); |
| 90 | *// Return angle multiplied with 1 or -1* |
| 91 | **return** angle \* (Vector3.Dot(axis, Vector3.Cross(dirA, dirB)) < 0 ? -1 : 1); |
| 92 | } |