**Unity3D脚本：WOW视角控制**

Posted on 2013年02月04日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 284 次

// This camera is similar to the one used in Jak & Dexter

var target : Transform;

var distance = 4.0;

var height = 1.0;

var smoothLag = 0.2;

var maxSpeed = 10.0;

var snapLag = 0.3;

var clampHeadPositionScreenSpace = 0.75;

var lineOfSightMask : LayerMask = 0;

var zoomSpeed = 50;

private var isSnapping = false;

private var headOffset = Vector3.zero;

private var centerOffset = Vector3.zero;

private var controller : ThirdPersonController;

private var velocity = Vector3.zero;

private var targetHeight = 100000.0;

private var theta;

private var theta2;

private var isColliding = false;

var targetOffset = Vector3.zero;

var closerRadius : float = 0.2;

var closerSnapLag : float = 0.2;

var xSpeed = 200.0;

var ySpeed = 80.0;

var yMinLimit = -20;

var yMaxLimit = 80;

private var currentDistance = 10.0;

private var x = 0.0;

private var y = 0.0;

private var distanceVelocity = 0.0;

function Start () {

var angles = target.transform.eulerAngles;

x = angles.y;

y = angles.x;

currentDistance = distance;

// Make the rigid body not change rotation

if (rigidbody)

rigidbody.freezeRotation = true;

}

function Awake ()

{

theta = 0;

theta2 = 0;

var characterController : CharacterController = target.collider;

if (characterController)

{

centerOffset = characterController.bounds.center -

target.position;

headOffset = centerOffset;

headOffset.y = characterController.bounds.max.y -

target.position.y;

}

if (target)

{

controller = target.GetComponent(ThirdPersonController);

}

if (!controller)

Debug.Log("Please assign a target to the camera that has a Third Person Controller script component.");

}

function LateUpdate () {

var targetCenter = target.position + centerOffset;

var targetHead = target.position + headOffset;

// When jumping don't move camera upwards but only down!

if (controller.IsJumping ())

{

// We'd be moving the camera upwards, do that only if it's really high

var newTargetHeight = targetCenter.y + height;

if (newTargetHeight < targetHeight || newTargetHeight - targetHeight > 5)

targetHeight = targetCenter.y + height;

}

// When walking always update the target height

else

{

targetHeight = targetCenter.y + height;

}

// We start snapping when user pressed Fire2!

if (Input.GetButton("Fire2") && !isSnapping)

{

//velocity = Vector3.zero;

isSnapping = true;

}

if (isSnapping)

{

//ApplySnapping (targetCenter);

OrbitPlayerFromBehind (targetCenter);

}

else

{

//ApplyPositionDamping (Vector3(targetCenter.x, targetHeight, targetCenter.z));

}

SetUpRotation(targetCenter, targetHead);

if (Input.GetMouseButton(1))

{

OrbitPlayerFromBehind (targetCenter);

}

if (Input.GetMouseButton(0))

{

OrbitPlayer (targetCenter);

isSnapping = false;

}

else

{

//ApplyPositionDamping (Vector3(targetCenter.x, targetHeight, targetCenter.z));

isSnapping = true;

}

distance += -(Input.GetAxis("Mouse ScrollWheel") \* Time.deltaTime) \* zoomSpeed \* Mathf.Abs(distance);

if (distance < 5)

{

distance = 5;

}

if (distance > 50)

{

distance = 50;

}

}

function ApplySnapping (targetCenter : Vector3)

{

}

function OrbitPlayer (targetPlayer : Vector3)

{

y = transform.position.y;

x += Input.GetAxis("Mouse X") \* xSpeed \* 0.02;

y -= Input.GetAxis("Mouse Y") \* ySpeed \* 0.02;

var rotation = Quaternion.Euler(0, x, 0);

var targetPos = target.position;

var direction = rotation \* -Vector3.forward;

var targetDistance = AdjustLineOfSight(targetPos, direction);

currentDistance = distance;

transform.rotation = rotation;

transform.position = targetPos + direction \* currentDistance;

if (y - target.position.y > 50)

{

y = target.position.y + 50;

}

transform.position.y = y;

camera.transform.LookAt(targetPlayer);

}

function OrbitPlayerFromBehind (targetPlayer : Vector3)

{

var newPosition = Vector3();

var fwd = new Vector3();

var rightVector = new Vector3();

var upVector = new Vector3();

var movingVector = new Vector3();

var collisionVector = new Vector3();

var zeroVector = new Vector3();

if (Input.GetMouseButton(1))

{

var mouseY = Input.GetAxis("Mouse Y");

}

theta = jangle(transform.position - targetPlayer) \* Time.deltaTime;

theta2 = theta2 + mouseY \* Time.deltaTime;

if (1.40 <= theta2)

theta2 = 1.40;

fwd = target.TransformDirection(Vector3.forward);

fwd = fwd.normalized;

newPosition = target.position - (distance \* fwd);

newPosition.y = target.position.y + distance \* Mathf.Sin(theta2);

y = target.transform.eulerAngles.z;

x = target.transform.eulerAngles.y;

rightVector = Vector3(fwd.z, 0, -fwd.x);

rightVector = rightVector.normalized;

upVector = Vector3.Cross(fwd, rightVector);

upVector = upVector.normalized;

movingVector = Vector3.Slerp(transform.position, newPosition, Time.deltaTime \* 50);

collisionVector = AdjustLineOfSight(transform.position, targetPlayer);

if (collisionVector != Vector3.zero)

{

Debug.Log("I am trying to get new position");

movingVector = collisionVector - (fwd \* -1);

distance = movingVector.z - collisionVector.z;

}

transform.position = movingVector;

camera.transform.LookAt(targetPlayer, upVector);

}

function AdjustLineOfSight (newPosition : Vector3, target : Vector3)

{

var hit : RaycastHit;

if (Physics.Linecast (target, newPosition, hit, lineOfSightMask.value))

{

Debug.Log("I hit someting at: "+hit.point);

velocity = Vector3.zero;

isColliding = true;

return hit.point;

}

//return newPosition;

}

function ApplyPositionDamping (targetCenter : Vector3)

{

}

function SetUpRotation (centerPos : Vector3, headPos : Vector3)

{

}

function AngleDistance (a : float, b : float)

{

a = Mathf.Repeat(a, 360);

b = Mathf.Repeat(b, 360);

return Mathf.Abs(b - a);

}

function jangle(pTemp : Vector3)

{

var tTheta:Number;

pTemp.Normalize();

// quadrant I & II

tTheta = Mathf.Acos(pTemp.x);

// quadtrant III & IV

if (pTemp.z < 0)

tTheta = 2\*Mathf.PI - tTheta;

return tTheta;

}

function bindAngle(tDegrees:Number)

{

if (tDegrees < 0)

tDegrees += Mathf.PI\*2;

else if (Mathf.PI\*2 <= tDegrees)

tDegrees += - Mathf.PI\*2;

return tDegrees;

}

static function ClampAngle (angle : float, min : float, max : float) {

if (angle < -360)

angle += 360;

if (angle > 360)

angle -= 360;

return Mathf.Clamp (angle, min, max);

}

@script AddComponentMenu ("Third Person Camera/Spring Follow Camera")