**Unity3D教程：实现谷歌地球旋转效果**

Posted on 2013年04月09日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 264 次

**Unity3D**实现谷歌地球旋转效果。

#pragma strict

var numberAverages : int = 3;

private var originalRotation : Quaternion;

private var offsetRotation : Quaternion;

// Make sure there are always SphereCollider and Rigidbody

@script RequireComponent (SphereCollider)

@script RequireComponent (Rigidbody)

function Awake () {

numberAverages = Mathf.Clamp (numberAverages, 1, numberAverages);

}

function OnMouseDown () {

var hit : RaycastHit;

var dir : Vector3;

// Stop spinning

rigidbody.angularVelocity = Vector3.zero;

// Record initial variables

if (Physics.Raycast (Camera.main.ScreenPointToRay(Input.mousePosition), hit)) {

originalRotation = transform.rotation;

dir = hit.point - transform.position;

offsetRotation = Quaternion.Inverse (Quaternion.LookRotation (dir));

Spin (dir);

}

}

function Spin (dir : Vector3) {

var hit : RaycastHit;

var privDirList : Array = new Array ();

var currentDir : Vector3;

// Initialize previous direction list

for (var i : int = 0; i < numberAverages; i++) {

privDirList.Add (currentDir);

}

currentDir = dir;

// Make the object rotate with the cursor while we are grabbing it

while (Input.GetButton ("Fire1") && Physics.Raycast (camera.main.ScreenPointToRay(Input.mousePosition), hit)) {

// Remove first element of the array

privDirList.RemoveAt (0);

// Add current direction to the end

privDirList.Add (currentDir);

currentDir = hit.point - transform.position;

transform.rotation = Quaternion.LookRotation (currentDir) \* offsetRotation \* originalRotation;

yield;

}

// User let go of the mouse so make the object spin on its own

var avgPrevDir : Vector3 = Vector3.zero;

var newPDirList : Vector3[] = privDirList.ToBuiltin(Vector3);

for (dir in newPDirList) {

avgPrevDir += dir;

}

// Keep spinning with the direction when you end touching

avgPrevDir /= numberAverages;

MoveAcceleration (currentDir, avgPrevDir);

}

function MoveAcceleration (r2 : Vector3, r1 : Vector3) {

var linearVelocity : Vector3;

var angVelocity : Vector3;

// Calculate the angular velocity: Omega = r x v / r^2

linearVelocity = (r2 - r1) / Time.deltaTime;

rigidbody.angularVelocity = Vector3.Cross (r2, linearVelocity) / r2.sqrMagnitude;

}