**Unity3D脚本：C＃计时类脚本**

Posted on 2013年02月16日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 204 次

using UnityEngine;  
using System.Collections;

/// <summary>  
/// C# timer 改编自Jeff 'PsychicParrot' Murray 的js的timer  
/// </summary>  
public class Timer : MonoBehaviour {

private float timeElapsed = 0.0f;  
private float currentTime = 0.0f;  
private float lastTime = 0.0f;  
private float cdTime = 0.0f;  
private float timeScaleFactor = 1.0f;  
private bool isTimerRunning;  
private bool doneCallback;  
private float parseTime;  
private GameObject callback;  
private float aHour;  
private float aMinute;  
private float aSecond;  
private float aMillis;  
private string seconds;  
private string minutes;  
private string hour;  
private string mills;  
private string timeString;

void Update ()  
{  
timeElapsed=Time.time-lastTime;

if(isTimerRunning){  
currentTime+=timeElapsed\*timeScaleFactor;  
}

lastTime=Time.time;  
}

public void StartTimer()  
{  
isTimerRunning=true;  
doneCallback=false;  
lastTime=Time.time;  
}

public void StopTimer()  
{  
isTimerRunning=false;  
}

public void ResetTimer()  
{  
doneCallback=false;  
timeElapsed=0.0f;  
lastTime=0.0f;  
currentTime=0.0f;  
lastTime=Time.time;  
}

public void SetCountdownTime(float aTime)  
{  
cdTime=aTime;  
}

public void SetCallback(GameObject aRef)  
{  
callback=aRef;  
doneCallback=false;  
}

public float GetTime()  
{  
return currentTime;  
}

public string GetFormattedTime(int returnType)  
{  
if(cdTime>0){  
// if a countdown time has been set, we parse the reverse time value  
parseTime=cdTime-currentTime;  
if(parseTime<0){  
if(!doneCallback){  
// SOUND THE ALARM!!!!! WE HIT 0!  
callback.SendMessage("alarmDone");  
// set our doneCallback flag so as not to repeat call alarmDone()  
doneCallback=true;  
}  
parseTime=0;  
}  
} else {  
// if no countdown time has been set, we just parse the regular time  
parseTime=currentTime;  
}

// grab hours  
aHour = parseTime/3600;  
aHour=aHour%24;

// grab minutes  
aMinute=parseTime/60;  
aMinute=aMinute%60;

// grab seconds  
aSecond=parseTime%60;

// grab milliseconds  
aMillis=(parseTime\*100)%100;

// format string into mm:ss:mm

seconds=Mathf.Round(aSecond).ToString();  
if(seconds.Length<2)  
seconds="0"+seconds;

minutes=Mathf.Round(aMinute).ToString();  
if(minutes.Length<2)  
minutes="0"+minutes;

hour=Mathf.Round(aHour).ToString();  
if(hour.Length<2)  
hour="0"+hour;

mills=Mathf.Round(aMillis).ToString();  
if(mills.Length<2)  
mills="0"+mills;

switch(returnType){  
case 1:  
timeString=minutes+":"+seconds+":"+mills;  
break;  
case 2:  
timeString=minutes+":"+seconds;  
break;  
default:  
timeString=hour+":"+minutes+":"+seconds+":"+mills;  
break;  
}

return timeString;  
}

public float GetHours()  
{  
if(cdTime>0){  
// if a countdown time has been set, we parse the reverse time value  
parseTime=cdTime-currentTime;  
if(parseTime<0){  
parseTime=0;  
}  
} else {  
// if no countdown time has been set, we just parse the regular time  
parseTime=currentTime;  
}

// grab hours  
aHour = parseTime/3600;

return aHour;  
}

public float GetMinutes()  
{  
if(cdTime>0){  
// if a countdown time has been set, we parse the reverse time value  
parseTime=cdTime-currentTime;  
if(parseTime<0){  
parseTime=0;  
}  
} else {  
// if no countdown time has been set, we just parse the regular time  
parseTime=currentTime;  
}

// grab minutes  
aMinute=parseTime/60;  
aMinute=aMinute%60;

return aMinute;  
}

public float GetSeconds()  
{  
if(cdTime>0){  
// if a countdown time has been set, we parse the reverse time value  
parseTime=cdTime-currentTime;  
if(parseTime<0){  
parseTime=0;  
}  
} else {  
// if no countdown time has been set, we just parse the regular time  
parseTime=currentTime;  
}

// grab seconds  
aSecond=parseTime%60;

return aSecond;  
}

public int getClockTimeHour()  
{  
int aTime=System.DateTime.Now.Hour;  
return aTime;  
}

public int getClockTimeMinute()  
{  
int aTime=System.DateTime.Now.Minute;  
return aTime;  
}

public int getClockTimeSeconds()  
{  
int aTime=System.DateTime.Now.Second;  
return aTime;  
}

public string GetClockFormattedTime(int returnType)  
{  
int aHour =System.DateTime.Now.Hour;  
int aMinute =System.DateTime.Now.Minute;  
int aSecond =System.DateTime.Now.Second;  
int aMillis =System.DateTime.Now.Millisecond;

// format string into mm:ss:mm  
seconds=Mathf.Round(aSecond).ToString();  
if(seconds.Length<2)  
seconds="0"+seconds;

minutes=Mathf.Round(aMinute).ToString();  
if(minutes.Length<2)  
minutes="0"+minutes;

hour=Mathf.Round(aHour).ToString();  
if(hour.Length<2)  
hour="0"+hour;

mills=Mathf.Round(aMillis).ToString();  
if(mills.Length<2)  
mills="0"+mills;

if(mills.Length>2)  
mills=mills.Substring(0,2);

switch(returnType){  
case 1:  
timeString=minutes+":"+seconds+":"+mills;  
break;  
case 2:  
timeString=minutes+":"+seconds;  
break;  
default:  
timeString=hour+":"+minutes+":"+seconds+":"+mills;  
break;  
}

return timeString;

}  
}

**Unity3D脚本：边缘高光脚本**

Posted on 2013年02月16日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 240 次

[**Shader**](http://www.unitymanual.com)着色器，边缘高光，只需一个shader文件，直接可用。

Shader "Example/Rim" {  
Properties {  
\_MainTex ("Texture", 2D) = "white" {}  
\_BumpMap ("Bumpmap", 2D) = "bump" {}  
\_RimColor ("Rim Color", Color) = (0.26,0.19,0.16,0.0)  
\_RimPower ("Rim Power", Range(0.5,8.0)) = 3.0  
}  
SubShader {  
Tags { "RenderType" = "Opaque" }  
CGPROGRAM  
#pragma surface surf Lambert  
struct Input {  
float2 uv\_MainTex;  
float2 uv\_BumpMap;  
float3 viewDir;  
};  
sampler2D \_MainTex;  
sampler2D \_BumpMap;  
float4 \_RimColor;  
float \_RimPower;  
void surf (Input IN, inout SurfaceOutput o) {  
o.Albedo = tex2D (\_MainTex, IN.uv\_MainTex).rgb;  
o.Normal = UnpackNormal (tex2D (\_BumpMap, IN.uv\_BumpMap));  
half rim = 1.0 - saturate(dot (normalize(IN.viewDir), o.Normal));  
o.Emission = \_RimColor.rgb \* pow (rim, \_RimPower);  
}  
ENDCG  
}  
Fallback "Diffuse"

}