fx

本页由 胡四娃 翻译自官方的教程

## 特效教程(FX)

## 综述

参看前面的教程基础,对象创建,时钟,帧层次结构[]动画[]视口与摄像机,和声音与音乐[]这篇教程介绍了什么是特效以及如何创建它们

特效是将曲线及其组合而成的一组数据(正弦线、三角型边、矩形或者线性),应用在不同类型的参数中。如:缩放、旋转、位置、速度、颜色等。

特效在配置文件中设置,仅仅只需要一行代码就可以在对象上使用这些特效。

可以有最多8条任意类型的曲线组合在一起形成一个特效。

在同一时间,可以有最多4个特效应用于同一个对象上面。

特效可以使用绝对值或者相对值,这取决于配置文件中Absolute标签。

控制曲线的周期、相位、和振幅都是允许的。

对于位置和速度特效来说,输出值可以使用对象的方向 和/或 缩放值,以相对方式应用于对象目前的状态。

这也就允许我们创造极其拉风的视觉特效。

除非特效已经缓存在内存中,否则特效参数全部在配置文件中进行调整,并且使用退格键来即时重载[] (cf. 通过 KeepInCache 属性来实现内存的缓存).

比如说: 你不能调整正在运行的循环特效,因为他已经在默认的配置文件中定义好了。在这个测试程序运行的时候,所有其它的特效能够被更新。

通常说来, 随机值的使用可以给特效带来更多的变化。

比如, 晃动方式的缩放[the wobble scale), 带颜色的闪光(the flash color) 和 攻击式的移动(the "attack" move) 等特效就使用了少量的随机值.

就像显示事件一样,我们也可以注册特效的开始播放和停止的事件。 因为循环时间是永远不会停下来的,所以对应的停止事件(orxFX\_EVENT\_STOP)永远不会发生. 我们也会简单的介绍一下如何一些个性数据(仅仅包含一个布尔值的结构)添加到orxOBJECT中。<sup>1)</sup>

在事件的回调函数中,我们通过它,在特效开始的时候为对象加锁,在结束的时候解锁。

我们使用锁是为了让soldier(士兵)在同一时刻只有一个特效在发挥作用。

把这些东西写在这里,仅仅具有教育意义。2)

## 详细内容

通常,我们先载入配置文件,创建一个时钟,然后注册更新函数,最后,创建我们的士兵和盒子对象。请从之前的教程中获取更多信息。.

然后, 我们注册输入和特效事件

```
orxEvent_AddHandler(orxEVENT_TYPE_FX, EventHandler);
orxEvent_AddHandler(orxEVENT_TYPE_INPUT, EventHandler);
```

大家可以看到,在这两个事件中,我们使用了同一个回调函数\(\text{EventHandler\(\text{\Pi}\).

现在我们迅速的扫一眼自己的"对象"数据结构。

```
typedef struct MyObject
{
  orxBOOL bLock;
} MyObject;
```

接下来,看看如何用 orxObject SetUserData()将它绑定到soldier上

```
MyObject *pstMyObject;

pstMyObject = orxMemory_Allocate(sizeof(MyObject), orxMEMORY_TYPE_MAIN);
pstMyObject->bLock = orxFALSE;
orxObject_SetUserData(pstSoldier, pstMyObject);
```

现在看看如何在Update函数中使用特效

```
orxSTRING zSelectedFX;
if(orxInput_IsActive("SelectWobble"))
{
   zSelectedFX = "WobbleFX";
}
else if(orxInput_IsActive("SelectCircle"))
{
   zSelectedFX = "CircleFX";
}

[...]

// Soldier not locked?
if(!((MyObject *)orxObject_GetUserData(pstSoldier))->bLock)
{
   if(orxInput_IsActive("ApplyFX") && orxInput_HasNewStatus("ApplyFX"))
   {
     orxObject_AddFX(pstSoldier, zSelectedFX);
   }
}
```

可以看到,我们通过orxObject\_GetUserData()这个函数得到了我们想要的数据,向solder里添加特效的方法跟添加声音的方法如出一辙,用的都是这个函数orxObject\_AddFX()[]

接下来,看看EventHandler这个函数

首先是输入方面,这里只展示了每次输入时哪个按键被使用了。

```
if(_pstEvent->eType == orxEVENT_TYPE_INPUT)
{
   if(_pstEvent->eID == orxINPUT_EVENT_ON)
   {
```

fx

```
orxINPUT_EVENT_PAYLOAD *pstPayload;

pstPayload = (orxINPUT_EVENT_PAYLOAD *)_pstEvent->pstPayload;

if(pstPayload->aeType[1] != orxINPUT_TYPE_NONE)
{
    orxLOG("[%s] triggered by '%s' + '%s'.", pstPayload->zInputName,
orxInput_GetBindingName(pstPayload->aeType[0], pstPayload->aeID[0]),
orxInput_GetBindingName(pstPayload->aeType[1], pstPayload->aeID[1]));
}
else
{
    orxLOG("[%s] triggered by '%s'.", pstPayload->zInputName,
orxInput_GetBindingName(pstPayload->aeType[0], pstPayload->aeID[0]));
}
}
```

正如你所见,我们通过按下的是一个单键还是一个组合键来判断展示不同的信息。

我们仅使用了两个首次输入点,因为我们知道,我们的配置文件中没有超过两个的组合键。尽管orx支持最多四个组合键来做为一个单键。

orxInput\_GetBindingName() 函数给了我们一个输入的文字显示。

注意: 这些名称在配置文件中也绑定到了对应的按键上面。

现在来看下如何处理这个事件

```
if(_pstEvent->eType == orxEVENT_TYPE_FX)
  orxFX_EVENT_PAYLOAD *pstPayload;
  orxOBJECT
                      *pst0bject;
  pstPayload = _pstEvent->pstPayload;
  pst0bject = orx0BJECT( pstEvent->hRecipient);
  switch( pstEvent->eID)
    case orxFX_EVENT START:
      orxLOG("FX <%s>@<%s> has started!", pstPayload->zFXName,
orxObject GetName(pstObject));
      if(pst0bject == pstSoldier)
        // Locks it
        ((MyObject *)orxObject GetUserData(pstObject))->bLock = orxTRUE;
      break:
    case orxSOUND EVENT STOP:
      orxLOG("FX <%s>@<%s> has stoped!", pstPayload->zFXName,
```

```
orxObject_GetName(pstObject));

if(pstObject == pstSoldier)
{
    // Unlocks it
    ((MyObject *)orxObject_GetUserData(pstObject))->bLock = orxFALSE;
}
break;
}
```

在soldier上的动画开始的时候,我们用自己的数据结构来锁定它,相应的,停止的时候解锁。

看完了代码部分, 我们再去看看配置文件。

首先看个简单的特效: 盒子上旋转的特效。

```
[RotateLoopFX]
SlotList = Rotate
Loop = true
[Rotate]
      = rotation
Type
StartTime = 0.0
         = 2.0
EndTime
Curve
         = sine
Pow
         = 2.0
StartValue = 0
EndValue = 360
[Box]
FXList = RotateLoopFX
```

看到了吧,特效是在它创建之初直接应用在盒对象上面的,而不是在代码中。

RotateLoopFX包含仅包含一个时间段[Rotate]并且一直循环[attribute Loop[]

然后定义Rotates时间段。时间的单位都是秒,角度的单位都是度。

定义这个旋转动画的时候,我们使用了一个正弦曲线,让他每两秒旋转360度。

下面看下我们的摇摆特效。

fx

```
Amplification = 0.0

StartValue = (1.0, 1.0, 1.0)

EndValue = (2.0, 2.0, 1.0) \sim (6.0, 6.0, 1.0)
```

我们修改了scale属性,并赋予它一个StartValue(开始值)和EndValue(结束值)。 他们都是用向量来表示的,如果不想使用任何各向异性的值(译者注:专业名词 anisotropic(各向异性) 去知道确切意思)的话,也可是使用float类型来表示。 虽然看起来我们正在使用一个 a isotropic(各向同性) 3)的值,这个EndValue也不过是一个随机值。

也就是说,它的X和Y部分可能是完全统统的随机值!

除此之外,我们使用了一个简单的周期为0.2 秒的正弦曲线,它将会播放1秒钟。

看到了吧,我们将Amplification[]增幅) 的值设为0,这就是说,随着时间的推进,曲线的振幅会逐渐变低。 注意: 默认的Amplification是1,表示不随时间变化,保持稳定,当值大于1时,振幅就会加大; 当值小于1时,振幅就会减少。

看看圆是如何运动的。

```
[CircleFX]
SlotList
               = CircleX#CircleY
KeepInCache
                = true
[CircleX]
Type
                = position
StartTime
                = 0.0
                = 1.0
EndTime
Curve
                = sine
               = (0.0, 0.0, 0.0)
StartValue
                = (-50.0, 0.0, 0.0)
EndValue
UseOrientation = true
                = true
UseScale
[CircleY@CircleX]
            = 0.25
Phase
StartValue = (0.0, -25.0, 0.0)
         = (0.0, 25.0, 0.0)
EndValue
```

Here we need to use 2 slots that affects the position so as to be able to have a circle motion. The first slot, CircleX, will apply a sine curve on the X component of our object's position. The second slot, CircleY, will apply the same curve (with a different amplitude) on its Y component.

我们使用两个时间段来控制它的位置,这样才能做出一个圆形的运动。第一个时间段是CircleX□他将会应用在对象的X轴向的振幅。第二个时间段CircleY,会产生一个同样幅度的作用效果在Y轴上。

如果我们不更改CircleY的相位,是不会发生圆形的运动。

现在假设一个正弦曲线,在初始值(StartValue)是相位0,准备增加

在相位0。25的时候,到达中间点,将会继续增加

在相位0.5的时候,到达最高值(EndValue),准备下降

在相位0.75的时候,回到中间点,继续下降

在相位1.0的时候,就跟相位0(StartValue)是一样的了

注意: 这段描述正弦曲线的工作过程也同样适用于三角形,但是却不适用于线形。

我们将略过大多数其他的特效,因为那里没有什么我们不知道的新知识了。

但是我们还是要迅速的看一眼翻转的特效,他将会向我们展示如何翻转一个对象。就像Paper Mario Wii<sup>4)</sup>的风格.

看到了吧,我们很简单的使用负值完成了这个效果! 同时也注意到,我们给Period(周期)设了一个明确的值。 我们选了一个两倍于定义的正弦曲线的Period,这样我们就只使用了正弦曲线的上升的那一半。同时,我们也将Amplification改回了1。(在""Wobble""中被设为0)

## 资源

源代码: 07 FX.c

配置文件: 07 FX.ini

1)

九天注:这里作者有点穿越了,需要看下面的例子才能懂,作者定义了一个仅包含一个Bool值的结构MyObject[]这里的括号,作者打在了orxOBJECT后面,我费解了N久,其实应该修饰个性数据,所以个人调整了一下。

九天注:本来一个对象可以同时有4个特效发生,这里作者仅仅是告诉你怎么使用"个性数据"才这样做的,所以说仅仅具有教育意义。

Z值不影响2D元素

4)

九天注[Wii上的 ① 纸片马里奥是个很出名的游戏,作者的意思就是这里的flip描述的就是那个游戏里面的风格和效果

From

https://orx-project.org/wiki/ - Orx Learning

Permanent link:

https://orx-project.org/wiki/cn/orx/tutorials/fx?rev=1446085730

Last update: 2025/09/30 17:26 (2 months ago)

