

本页由 killman.liu 翻译自原英文教程[Frame](#)

帧教程(Frame tutorial)

综述

请查看前面的[基础教程](#)获得更多有关[基本对象创建](#)和[时钟处理](#)的信息。

所有对象的坐标，缩放和旋转都保存在`orxFRAME` 结构中。

这些帧(frames)被汇编到成一个层次结构，意味着修改父帧的属性将会影响它的所有子帧。

在本篇教程中，我们有四个链接到同一个副对象的对象¹⁾，还有第五个没有父对象的对象。

前面两个子对象使用父对象的配置属性`ChildList`隐式创建，另外两个子对象在代码中创建并连接到副对象（为教学目的这么做）。

不可见的父对象间跟踪鼠标指针，左-`shift` 和 左-`ctrl` 键将分别放大和缩小父对象，而鼠标左击和右击将使对象旋转。

所有这些变化将影响它的四个子对象。

这为我们提供了一种简单的方法去创建复杂的或组合的对象，并可十分方便地改变它们（坐标，缩放，旋转，速度 ……）。

详情

在[前面的教程](#)中，我们是从加载配置文件和创建视口(viewport)开始的。

```
orxConfig_Load("../03_Frame.ini");  
  
orxViewport_CreateFromConfig("Viewport");
```

然后，我们创建父对象。

```
pstParentObject = orxObject_CreateFromConfig("ParentObject");
```

由于在配置文件里，我们给`ParentObject`做了这样的定义：

```
[ParentObject]  
ChildList = Object3 # Object4
```

于是，当我们创建父对象时，那两个子对象²⁾也被自动地创建并连接(link)到父对象了。

我们可以这样创建全部四个对象，但是为了学习，我们将在代码中手动创建和连接(link)剩下的两个对象。

```
orxOBJECT *pstObject;  
orxObject_CreateFromConfig("Object0");  
pstObject = orxObject_CreateFromConfig("Object1");  
orxObject_SetParent(pstObject, pstParentObject);  
pstObject = orxObject_CreateFromConfig("Object2");
```

```
orxObject_SetParent(pstObject, pstParentObject);
```

这里, `Object0` 是我们的静态对象: 唯一的不会被link(连接)到`ParentObject`的对象。

请注意当我们在代码中手动地创建和连接对象时, 删除这些对象是我们的职责。 相反, `Object3` 和 `Object4` 将在`ParentObject`删除时被自动删除。

然后, 我们创建一个100Hz的时钟并将我们的`Update`函数注册给它。这个函数将是我们管理缩放/旋转`ParentObject`对象的输入, 确保`ParentObject`跟随鼠标指针移动的地方。

```
pstClock = orxClock_Create(orx2F(0.01f), orxCLOCK_TYPE_USER);

orxClock_Register(pstClock, Update, orxNULL, orxMODULE_ID_MAIN,
orxCLOCK_PRIORITY_NORMAL);
```

Let's now have a look to our `Update` function.

First, we make sure we can find the position in our world space that corresponds to our mouse cursor in the screen space.

We then copy our `ParentObject` Z coordinate (ie. we keep the same depth as before) over it and we finally set it back on our `ParentObject`.

现在, 让我们来看看我们的`Update`函数。

首先, 我们确保可以找到位于对应于鼠标指针的屏幕空间(screen space)的世界空间(world space)中的位置。

然后复制`ParentObject`的Z坐标属性复制到这个位置信息, 最后将这个位置信息设置到`ParentObject`对象上。

```
if(orxRender_GetWorldPosition(orxMouse_GetPosition(&vPosition), &vPosition))
{
    orxVECTOR vParentPosition;
    orxObject_GetWorldPosition(pstParentObject, &vParentPosition);
    vPosition.fZ = vParentPosition.fZ;
    orxObject_SetPosition(pstParentObject, &vPosition);
}
```

唯一剩下要做的就是根据输入去应用缩放和旋转了。

在我们的例子中, 我们在`03_Frame.ini`中定义如下的输入: `RotateLeft`, `RotateRight`, `ScaleUp` and `ScaleDown`

让我们看看怎么处理它们。首先, 处理旋转:

```
if(orxInput_IsActive("RotateLeft"))
{
    orxObject_SetRotation(pstParentObject,
orxObject_GetRotation(pstParentObject) + orx2F(-4.0f) * _pstClockInfo->fDT);
}
if(orxInput_IsActive("RotateRight"))
{
    orxObject_SetRotation(pstParentObject,
orxObject_GetRotation(pstParentObject) + orx2F(4.0f) * _pstClockInfo->fDT);
}
```

现在，处理缩放：

```
if(OrxInput_IsActive("ScaleUp"))
{
    OrxObject_SetScale(pstParentObject, OrxVector_Mulf(&vScale,
OrxObject_GetScale(pstParentObject, &vScale), Orx2F(1.02f)));
}
if(OrxInput_IsActive("ScaleDown"))
{
    OrxObject_SetScale(pstParentObject, OrxVector_Mulf(&vScale,
OrxObject_GetScale(pstParentObject, &vScale), Orx2F(0.98f)));
}
```

全部完成了! 😊 我们的ParentObject将会被更新，所有的子对象也会随同它一起被更新。

注意：

- 我们可以使用配置值来替代旋转和缩放值的常量。这样，我们可以修改它们而不需要重新编译，甚至可以按下Backspace³⁾键实时更新。
- 因为我们使用时钟的DT来实现旋转，它们将受益于时间上的一致⁴⁾。不幸的是，缩放的情况不是这样（这通常是我们所不想要的）。

资源

源代码: [03_Frame.c](#)

配置文件: [03_Frame.ini](#)

1)

一个不可显示的空对象

2)

Object3 and Object4

3)

Orx启动器重新加载配置信息的默认按键

4)

它们不会依赖于帧频，并且可以进行加速/减速

From:

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

Permanent link:

<https://wiki.orx-project.org/cn/orx/tutorials/frame?rev=1278642891>

Last update: **2017/05/30 00:50 (8 years ago)**

