vJass 系列教程 6

面向对象编程(一) 结构基本语法

Aeris ▸ NJU ▸ 2008-7-9

vJass 系列教程 6

面向对象编程(一) 结构基本语法

内容提要

好了，从这一章开始，我们终于接触到 vJass 最强大，最令人兴奋的语法特征——面向对象编程了。这是 vJass 对现有 Jass 作的最重要的扩展。在面向对象深入人心的今天，名不见经传的 Jass 也可以使用面向对象的语法特征了，怎么称赞都不过分。

vJass 的面向对象扩展是一个很大的内容，本教程仍然本着由易到难，由浅入深的原则，先向各位读者介绍 vJass 面向对象的基本单元——结构的语法。如果读者有面向对象的基础，将会很容易理解本章的内容。

简介

什么是对象？我们用过对象么？实际上，魔兽里面就有很多对象，我们在地图的编写过程中已经用到很多了，只不过没有注意而已。举个例子，单位（unit）就是对象，对象有属性，而单位有诸如当前生命值、最大生命值、当前魔法值等和每个单位实例“捆绑”的属性；对象有操作方法，而单位有一系列诸如设置生命值、发布命令等等的函数。许多具有共性的对象的集合，我们称之为**“类”（Class）**，每个对象都是类的一个**实例（Instance）**。例如单位都有相同的属性，相同的操作函数，我们可以说所有单位属于单位类，而每个单位是单位类的一个实例。由于“对象”在文中既可能指实例，又可能指类，为避免混淆，下文将使用类（结构）和实例的术语，不再使用“对象”这个术语。另外，由于“方法”和“函数”指的是同一个东西，下文将不区分这两个概念，例如，将混用“成员函数”和“成员方法”。

使用面向对象的编程方法，可以让复杂的问题容易分割成简单问题，从而更好处理，同时也符合人们的一般思维习惯。在 vJass 里，使用面向对象的编程方法，可以使得以前许多复杂的，甚至根本不可能实现的问题，比如多维数组的定义和使用，函数传入数组作参数或者把数组作为返回值等变得可行而且简单。

结构（一）

在 vJass 里，等价于“类”的概念是**“结构”（Structure）**。之所以称为“结构”而不是“类”，是因为 vJass 并不能实现彻底、完全的面向对象特性，一些特性，诸如多态，还受到一定的限制，因此作者把 vJass 的“类”叫“结构”。

由于结构的语法相对较复杂，这里结合示例来说明。

结构定义

先看一个简单示例：

```
struct Point
endstruct
```

这里定义了一个叫 Point 的结构，里面什么也没有。可以看到，结构的定义是放在一对 **struct** 和 **endstruct** 之间的，结构名写在 **struct** 关键字之后。

成员变量

给上面的例子加一点东西：

```
struct Point
    integer x = 0
    integer y = 0
endstruct
```

这里给 Point 结构定义了两个成员变量：x 和 y，表示坐标，默认初始化值为。这样，每个 Point 的实例变量都会有整数 x 和 y 两个成员了。成员变量的定义语法是“**类型 变量名**”这种形式，成员变量可以有默认初始化值，和全局变量的定义类似。

成员方法

给上面的例子再加一点东西：

```
struct Point
    integer x
    integer y

    method move takes integer nx, integer ny returns nothing
        set this.x = nx
        set this.y = ny
    endmethod
endstruct
```

我们定义了一个叫 move 的方法。可以看到，定义方法的语法和定义函数的语法基本相同，只是把 function 关键字换成了 method 而已。而成员方法和普通的函数也基本相同，只是成员方法可以使用 this 引用（Reference）访问实例的其他成员变量或者成员方法，使用“**this.成员名**”的形式。this 引用代表结构的“当前”实例，例如你使用“**call** p.move(0, 0)”来调用 move 方法，那么 move 方法里的 this 就是指 p。

使用成员方法还要注意：根据实际情况，某些方法（具有多态性的虚函数）编译后是使用触发来模拟的，这些方法的执行使用了 TriggerExecute/TriggerEvaluate 函数。因此，在这类方法里调用 GetTriggeringTrigger 函数可能不会产生你想要的结果。另外在这类方法里调用等待、同步类函数非常容易出问题。当然，这类方法不多，因此一般情况下可以放心用，但是如果要使用多态的话还是小心点。

结构的创建、使用和销毁

定义了一个结构就要动态创建和销毁它的实例，这要求动态分配和回收内存。虽然 Jass 无法动态分配和回收内存，vJass 仍然可以通过数组模拟出这种效果。

我们用一个简单的例子来说明结构实例怎样创建、使用和删除。

```
function PointTest takes nothing returns nothing
    local Point p

    set p = Point.create()
    set p.x = 5
    set p.x = 8
```

```
    call BJDebugMsg(I2S(p.x) + " : " + I2S(p.y))

    call p.destroy()
endfunction
```

在这个测试函数里，我们首先定义了一个 Point 类型的局部变量 p，然后把 p 初始化为一个新创建的 Point 实例，接着设置这个点的 x 和 y 坐标并且打印出来，最后销毁这个对象。

可以看到，创建结构实例是通过调用**结构**的 create 方法（静态方法，下一篇教程会说明），这个方法会返回一个结构的实例，而销毁实例是调用**实例**的 destroy 方法。访问实例的成员变量或者调用成员方法是通过“**实例名.成员名**”进行的。

封装

封装是面向对象的基本特性之一，vJass 也提供了封装的机制。这里同样使用例子说明。

```
struct encap
    real a = 0.0
    private real b = 0.0
    public real c = 4.5

    method randomize takes nothing returns nothing
        // 结构自己的成员可以访问所有其他成员
        set this.a = GetRandomReal(0, 45.0)
        set this.b = GetRandomReal(0, 45.0)
        set this.c = GetRandomReal(0, 45.0)
    endmethod
endstruct

function test takes nothing returns nothing
    local encap e = encap.create()

    call BJDebugMsg(R2S(e.a)) // 合法
    call BJDebugMsg(R2S(e.c)) // 合法
    //call BJDebugMsg(R2S(e.b)) // 非法，b为私有
endfunction
```

结构的成员可以加上 private 和 public 访问修饰符，**默认为 public**，因此不加任何修饰符就是 public。public 成员可以被任何函数自由访问，而 private 函数只能被结构自己的成员访问。

总结

本节教程到此为止，最后总结一下结构的基本语法：

```
struct 结构名
    // 定义成员变量
    [private][public] 类型 成员变量名
    ...

    // 定义成员方法（函数）
    [private][public] method 方法名 takes 参数列表 returns 返回值
        // 方法体
```

```
    endmethod
    ...
endstruct
```

下一章教程将介绍结构的高级语法。