
实践篇：

第七讲 连宗



本次课我们要学习的内容如下：

- 1) 研究如何让游戏容纳多个不同类型角色？如何消去程序中的重复代码？从而自然引出面向对象的第二大机制：继承！
- 2) 游戏角色相互作用机制。



7.1 打砖

游戏策划7.1 打砖

游戏开始时，有三个演员出现在舞台上：一个圆球从左上角飞出，向某个方向运动；一个是蓝色的矩形挡板，位于在舞台的下方，进行水平方向的移动；一个是黑色的砖头，位于在舞台的上方，静止不动。圆球在碰到舞台边缘和板子时弹回；蓝色挡板在碰到舞台的左右边缘时弹回。



分 析

与前面的“桃园三结义”游戏相比，此游戏的区别在于，两个地方：**1)** 三个游戏角色的行为有些不一样了：首先，“更新”行为不一样；其次，“渲染”方式不一样。所以，原来只需创建一个类**Ball**，现在需要创建三个新类型；**2)** 三个游戏演员有了相互作用！这是一个本质的变化！



7.2 游戏角色的碰撞检验机制

所谓游戏角色的相互作用是指，在某些事件发生时，游戏角色的行为将受到另一个（些）角色影响。

游戏中的相互作用的形式很多，有些相互作用可能很复杂，比如，角色之间可以互相交谈（语言或文字）、碰撞（物理限制）、拾取（拿武器、摘一朵花等）、伤害（战斗）、治疗、交换物品（把花献给女朋友）等等。正是存在多种多样的相互作用，才使游戏变化多端、引人入胜。

游戏角色之间的“碰撞检验”是一种特殊的相互作用。



7.2 游戏角色的碰撞检验机制

在程序设计中，采用什么机制才能判断和处理游戏角色的碰撞呢？

其实在上一个程序中，我们已经初步接触到了碰撞检验机制：是判断球是否撞上了窗口。我们采取的策略是：

让窗口对象（主类对象）与球对象合作，在窗口对象的每一次调用`paint()`方法时，向球对象发送消息：`ball.collide(paneW,paneH);`
这行代码的语义是：

“嘿，球！我的大小是这样的：宽为`paneW`，高为`paneH`；你自己去判断是否撞上我啦！”



7.2 游戏角色的碰撞检验机制

在这个解决方案中，最重要机制有两点：**1）碰撞检验的责任交给了球对象自己；2）窗口要告知球对象，自己（球要判断碰撞的对象）在那里（窗口的宽和高）。这是通过参数把此信息传给球。**

现在，球对象除了要判断是否撞上窗口外，还要判断是否撞上挡板和砖头。

同样道理，还是把碰撞检验的责任交给球自己。

同样道理，还是要告知球，窗口的边、挡板和砖头**在哪里！**



7.2 游戏角色的碰撞检验机制

一个重要的问题是：谁知道这些信息（窗口的边、挡板和砖头的位置）？显然，球不知道，它只保存着自己的那些信息，并不知道它的世界中都有那些东西（按照面向对象软件的设计原则，它也应该如此）。

显然，主类对象知道所有的这一切！如何才能让球对象知道主对象所知道的一切呢？

最简单的方法就是：

主对象把自己作为参数传入球对象的**collide**方法中：

b.collide(this);

在Ball的collide()方法中，Ball根据作为参数传入的主对象，获得获知：窗口大小、挡板的位置、砖头的位置。然后根据这些信息，一一判断是否撞上它们。



7.2 游戏角色的碰撞检验机制

在这个解决方案中，最重要机制有两点：**1）碰撞检验的责任交给了球对象自己；2）窗口要告知球对象，自己（球要判断碰撞的对象）在那里（窗口的宽和高）。这是通过参数把此信息传给球。**

现在，球对象除了要判断是否撞上窗口外，还要判断是否撞上挡板和砖头。

同样道理，还是把碰撞检验的责任交给球自己。

同样道理，还是要告知球，窗口的边、挡板和砖头**在哪里！**

源程序参见“ThreeBrothers2.java”，请仔细阅读程序的注释文字！



§ 7.3 面向对象的继承机制

我们知道：行为完全一样的对象可以属于一个类；行为完全不同的对象肯定属于不同的类。问题是：假设有若干个对象，它们的行为既有相同点又有不同点，怎么处理呢？

在上面的软件实现中，因为这个三个角色的行为不太一样，我们创建了三个不同的类。但是这三个角色并不是完全不同，还有相当多的行为是一样的！



§ 7.3 面向对象的继承机制

若干对象的行为存在相似性，又使它们分属不同类的后果是：
三个类定义中存在重复的代码！

程序中的重复代码是邪恶的！

面向对象范型对此问题的解决方案是：继承机制！

把对象的共同属性和行为提取出来，放在一个“父类”中，然后再创建若干“子类”，继承于父类（得到大家的共同属性和行为），然后在子类中，分别定义它们的“个性”（不一样的属性和行为）！

参见源代码“Sprite.java”，请仔细阅读和体会其中的注释文字！



作业：实现上述**游戏策划7.1**
提交到教学平台