第 5 章

“小猫顶球”游戏

前面章节编写的游戏主要使用控制台和玩家进行交互，无论从游戏界面还是游戏操作来讲，都不是很友好。从本章开始，将在 Python 中使用 Pygame 来编写带图形界面的游戏。

本章介绍的小猫顶球的游戏虽然较为简单，但读者将从这个游戏中掌握 Pygame 模块的基础用法，从而为后续章节复杂游戏的设计打下良好基础。

5.1 “小猫顶球”游戏运行示例

2min

运行本书附带的 catBall 游戏工程后，会出现如图 5-1 所示的界面。

在图 5-1 所示的界面中，玩家可通过键盘上的左和右方向键控制小猫左右移动，以此来顶从空中落下的球，小猫顶到球后会发出“咚”的声音，同时分数增加，如果小猫没有顶到球，则让球落地，游戏结束，弹出如图 5-2 所示的游戏结束界面。

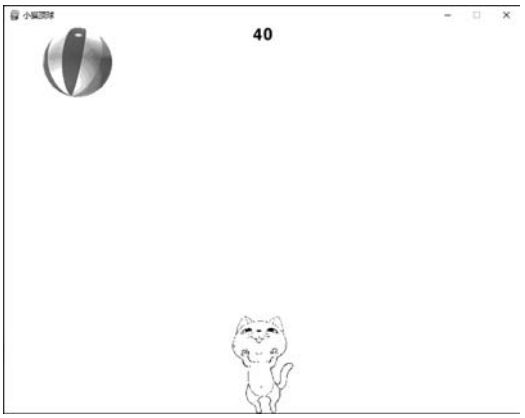

图 5-1 “小猫顶球”游戏开始界面

图 5-2 “小猫顶球”游戏结束

在游戏结束界面显示玩家已经获得的分数，同时提示玩家按 Space 键（键盘上的空格）开始新游戏，如果单击右上角的“×”按钮，则游戏结束。根据上述游戏过程，可画出如图 5-3 所示的小猫顶球的游戏流程图。

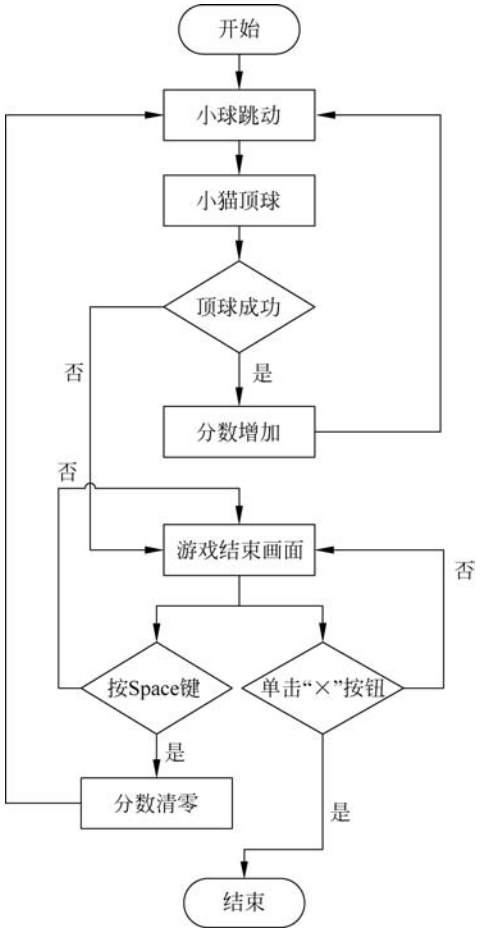

图 5-3 “小猫顶球”游戏流程图

5.2 Pygame 模块简介

“小猫顶球”游戏采用了 Pygame 模块进行编程。Pygame 是一个免费并且开源的编程语言库，其可用于 2D 游戏制作，包含对图像、声音、视频、事件、碰撞等支持，到现在 Pygame 已经有了 20 余年的发展历史。因 Pygame 建立在 SDL(Simple DirectMedia Layer)的基础上，SDL 是一套跨平台的多媒体开发库，底层用 C 语言实现，故 Pygame 的性能非常优越。

Pygame 模块在游戏开发上的易用性和跨平台的特性，使开发者不用被底层语言、游戏性能和所运行的操作系统平台所束缚，从而可以在游戏功能和逻辑上下更多功夫。

Pygame 模块的开发和支持者众多，开发上碰到的大部分问题可以在官网上找到答案。当读者碰到模块使用上的问题时，可以登录 http://www.pygame.org 求得帮助，同时在官网上提供了各个游戏种类的示例代码，读者也可根据这些开源的游戏示例代码学习更多游戏编程的知识。

说了这么多 Pygame 的优点，读者可能已经迫不及待地想掌握其用法。接下来一起用 Pygame 来完成本章的“小猫顶球”游戏。

5.3 “小猫顶球”游戏环境搭建

1. 创建“小猫顶球”游戏工程文件

在前面几章的游戏编程中，都是使用工程默认生成的 main. py 文件进行编码。虽然使用这个文件省事，但是不能从文件名中看出其要完成的功能，最好的办法就是文件根据功能的不同而有不同的名字。接下来一起看一下如何将新建的 catBall 里的 main. py 文件名修改为 catBall. py。

运行 PyCharm 后单击 File 菜单里的 New Project 按钮创建 catBall 工程，如图 5-4 所示。

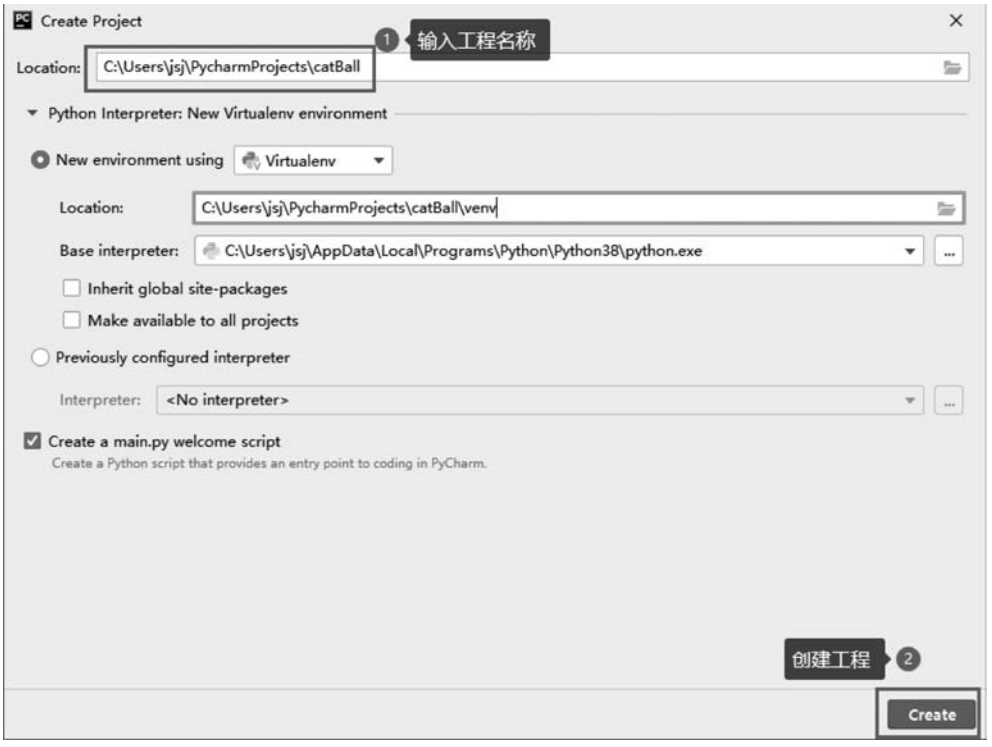

图 5-4 创建 catBall 工程

创建 catBall 工程后，在 main. py 文件上右击并单击 Refactor 按钮，在弹出的菜单中单击 Rename 按钮，如图 5-5 所示。

在弹出的对话框里将 main. py 修改为 catBall. py，单击 Refactor 按钮，如图 5-6 所示。

“小猫顶球”游戏需要小猫和球的素材，本书的附带资源已经提供了此素材。打开本章的附带资源后，复制 cat. png、ball. gif 和 dong. wav 文件，在 catBall 工程上右击，在弹出的菜单上单击 Paste 按钮，如图 5-7 所示。

至此，“小猫顶球”游戏的工程便建立完毕，读者应把 catBall. py 文件里的内容清空，从而为后续写入游戏代码做好准备。

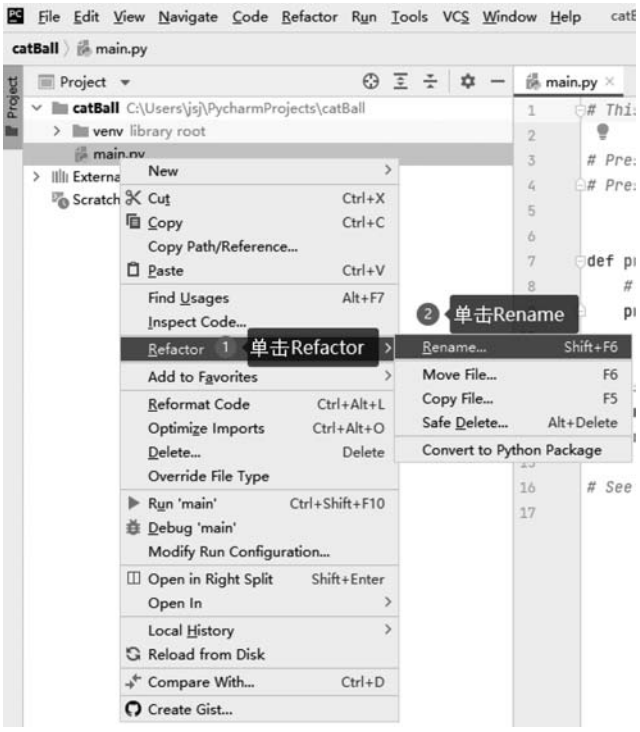

图 5-5　更改 main. py 文件

图 5-6　修改文件名称

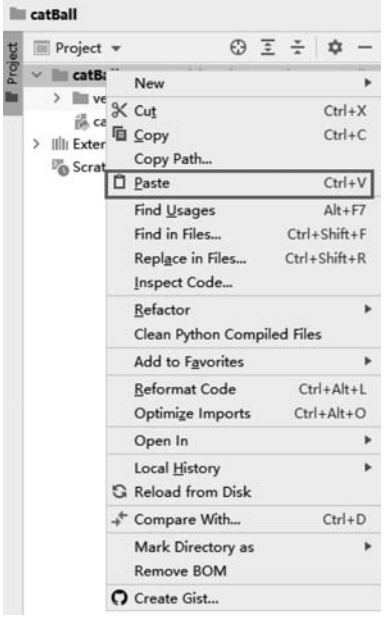

图 5-7　将资源文件粘贴到 catBall

2. 导入 Pygame 模块

Pygame 模块为外置模块，在使用前必须进行导入。在 PyCharm 里导入 Pygame 模块，既可以使用 Terminal 方式导入，也可以在图形界面下进行导入。

1）Terminal 导入

单击 PyCharm 下方的 Terminal 按钮，在弹出的窗口里输入命令，如图 5-8 所示，命令如下：

```
pip3 install pygame
```

图 5-8 命令行导入 Pygame

输入命令并按 Enter 键执行后，pip3 包管理器会自动在 Python 仓库里查找最新版本的 Pygame 安装包，并将其安装在当前的工程环境中。

2）图形界面导入

图形界面下安装 Pygame 要稍微复杂。运行 PyCharm 后，单击 File 菜单下的 Settings 按钮，在弹出的窗口里打开 Project：catBall 下拉列表后，单击 Python Interpreter 选项，在右侧的窗口里单击“＋”按钮，如图 5-9 所示。

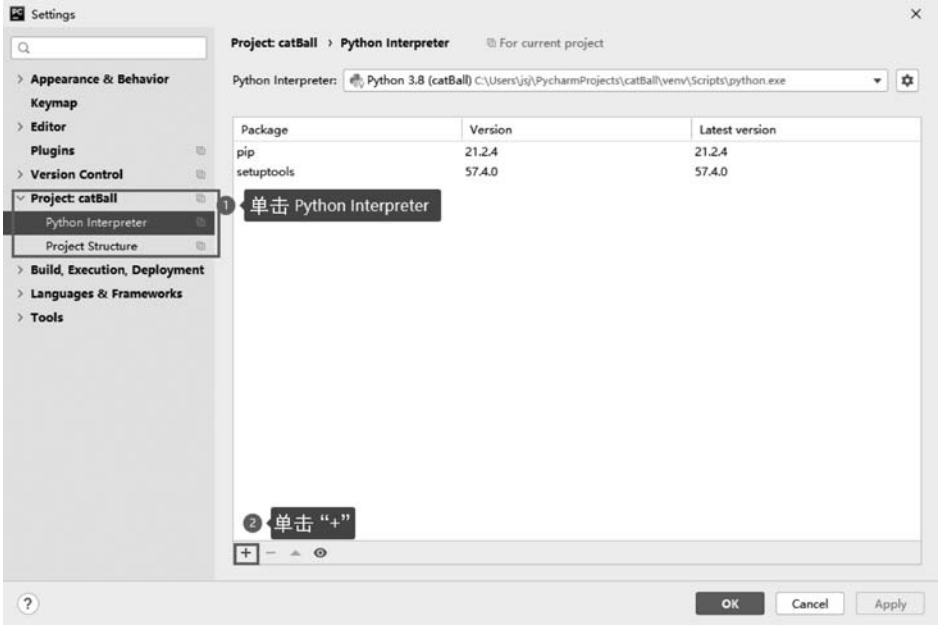

图 5-9 打开 Pygame 安装窗口

单击“+”后，在弹出的窗口里输入 pygame，单击 Install Package 按钮，如图 5-10 所示。

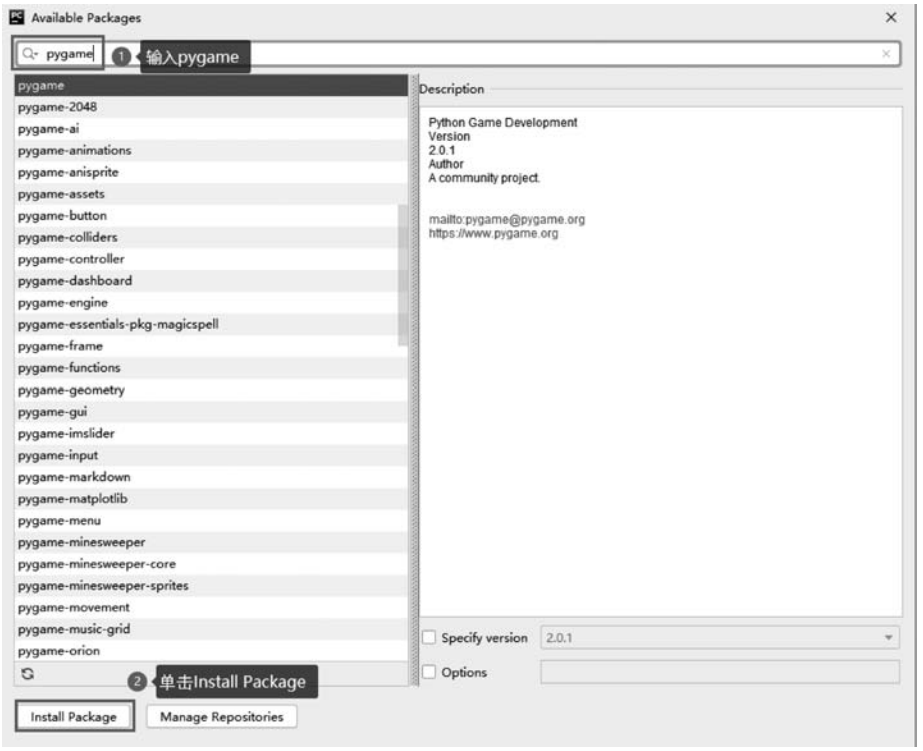

图 5-10　查找 pygame 并安装

5.4　图形界面初始化

Pygame 图形界面游戏编程非常类似于画家在一块儿画布上进行画图创作，作为一个“画家”首先需要做的就是掌握画布的创建方法。接下来一起看一下如何使用 Pygame 创建一个白色的画布，从而为后续“创作”打下良好基础。

5.4.1　无交互的图形界面创建

使用 PyCharm 打开 5.3 节创建好的 catBall 工程，双击 catBall.py 文件，输入代码如下：

```
#第5章/catBall/catBall.py
import pygame
pygame.init()
SIZE = WIDTH, HEIGHT = (800, 600)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption('小猫顶球')
WHITE = (255, 255, 255)
screen.fill(WHITE)
while True:
    pygame.display.update()
```

运行上述的代码，神奇的事情发生了，短短 9 行代码竟然运行出如图 5-11 所示的完整图形界面，而且这个图形界面无论在 Windows 上还是在 Linux 上都具有相同的显示效果！这就是 Pygame 强大之处，一次编码，多平台使用。

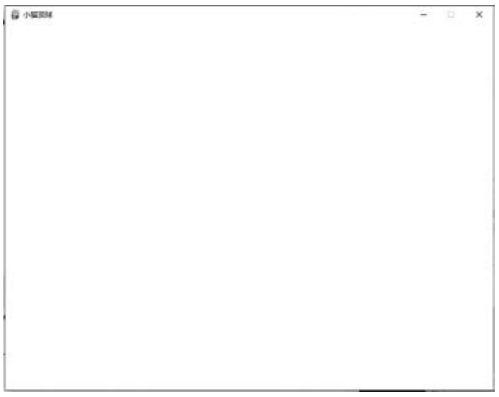

图 5-11 无交互的图形界面

上述代码虽然简单，但是 Pygame 游戏编程基本上使用这种编程方法。接下来一起看一下代码都代表什么意义。

在代码的第 2 行使用了 pygame.init()函数，此函数用于对 Pygame 游戏编程进行初始化。调用此函数时，Pygame 在初始化游戏编程时可能会用到的声卡、显卡、游戏手柄等硬件的调用接口，如果硬件有问题，会以元组的形式返回相关的错误代码。

screen＝pygame.display.set_mode(SIZE)语句创建出了一个大小为 800×600 像素的画布，screen 代表创建后返回的画布对象，以后需要在画布上画其他图形时，可以通过得到的 screen 画布对象进行创作。

pygame.display.set_caption('小猫顶球')语句用于将画布标题设置为小猫顶球。pygame.display 还有很多和窗口设置相关的属性，在后续章节中将进一步说明。

screen.fill(WHITE)语句将当前画布设置为 WHITE 颜色填充，其中 WHITE 变量由 RGB 三原色搭配而得，Pygame 将 RGB 三原色中每种原色的数值范围都设置为 0～255，通过 3 个不同的数值将搭配出不同颜色。例如白色为(255,255,255)，黑色为(0,0,0)。

程序的最后两行为一个 while 无限循环，在无限循环里不停地调用 pygame.display.update()语句。while 无限循环是 Pygame 保持窗口一直可以在屏幕上显示的关键，如果不使用无限循环，则创建的画布将一闪而过。pygame.display.update()语句用来更新画布上所有的图像，本节的例程代码不涉及画布上的图像创作，后续章节读者将看到此语句的具体使用。

5.4.2 画布相关属性

在 5.4.1 节创建了一个大小为 800×600 像素的空白白色画布，并且将画布的标题设置为“小猫顶球”。为了在画布上创作出更多佳作，下面看一下和画布相关的一些属性。

1. 画布坐标系

画布大小为 800×600 像素，什么位置是其开始计算像素的(0,0)点？是和数学中常用

的笛卡儿坐标系一样，即画布中心是(0,0)点吗？或者其他坐标体系？

Pygame 使用了一种新的坐标体系，在这种坐标体系下，坐标原点(0,0)是画布的左上角，如图 5-12 所示。

从图 5-12 可知，如果画布大小为 800×600 像素，则从画布左上角的坐标原点水平向右，x 坐标递增，最大值为 800，从画布坐标原点垂直向下，y 坐标递增，最大值为 600，画布右下角的坐标值为(800,600)。

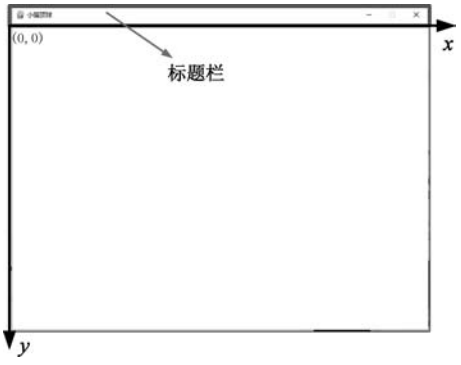

图 5-12　Pygame 坐标体系

2. 画布属性设置

在 5.4.1 节使用 pygame.display.set_mode(SIZE)语句将画布大小设置为 SIZE 大小，即 800×600 像素，画布还有一些其他的属性可进行设置，其属性设置如下。

```
pygame.display.set_mode(resolution=(0,0),flags=0,depth=0)
```

(1) resolution：一个二元组参数，表示宽和高。如果不传递这个参数或使用默认值(0,0)，则将设置成和当前计算机屏幕一样的分辨率。

(2) flags：指定要显示的类型，共有如表 5-1 所示的几个属性。当要使用多个属性时可以使用“|”进行连接。例如 pygame.FULLSCREEN | pygame.HWSURFACE 表示同时使用了两个属性。

表 5-1　flags 常见参数

参　数	参数描述
pygame.FULLSCREEN	画布全屏显示
pygame.DOUBLEBUF	双缓冲模式，在 HWSURFACE 或 OPENGL 下使用
pygame.HWSURFACE	硬件加速，只有全屏下才可以使用
pygame.OPENGL	使用 OPENGL 显示接口
pygame.RESIZABLE	创建的画布可以缩放
pygame.NOFRAME	画布没有边框，没有右上角最小化和关闭等按钮

(3) depth：表示要使用的颜色深度。通常情况下不要传递 depth 参数，Pygame 会自动根据当前操作系统选择最好和最快的颜色深度。

5.5　认识小猫等 Surface 对象

空白的画布创建好后，便可以在画布上进行艺术创作。本章介绍的是小猫顶球游戏，该如何加载小猫和球，这就必须认识一下 Pygame 里的 Surface 对象。

Pygame 认为创建好的画布是 Surface 对象，画布上所描绘的任何图形都是 Surface 对象，从外部加载的图形资源也是 Surface 对象。可以说，Surface 对象是 Pygame 编程的灵魂，掌握好 Surface 对象对于学习 Pygame 游戏编程会事半功倍。

1. 认识 Surface 对象

Surface 对象是 Pygame 模块的一个子集，用于表示任何一张图像，其具有固定的分辨率和像素格式，只需指定尺寸，就能通过 pygame.Surface()方法创建一个新的图像 Surface。创建好的 Surface 对象可以做很多事情，例如在其上绘制图形、写入文字、填充颜色等。Surface 对象支持不同对象间的叠加显示，本章介绍的小猫顶球游戏就是利用 Surface 对象这个特性来完成的。

Surface 对象极其重要，以致 Pygame 为其设计了多达 50 余个属性和方法，覆盖了 Surface 对象操作的方方面面。在后续章节中，读者将接触到 Surface 对象的各种用法。

Surface 对象创建后是一个矩形区域，其默认填充颜色为黑色，如果没有指定其他参数，则将创建出最适合当前显示分辨率的 Surface 对象。

2. Surface 对象创建方法

在 5.4.1 节使用 screen＝pygame.display.set_mode(SIZE)语句得到了一个 800×600 像素的 Surface 对象 screen，screen 是一个特殊的 Surface 对象，它是游戏编程的主画布，是后续 Surface 对象显示的前提。通常来讲，使用 Pygame 游戏编程都要使用这一语句来创建主画布，在后续章节的游戏设计中，读者会经常看到此代码的出现。

大多数情况下，游戏编程中涉及的各种图形资源都不是由 CPU 或 GPU 即时绘制出来的，往往通过加载美工已经绘制好的图片资源来完成图形资源的显示，Pygame 游戏编程也不例外。Pygame 通过 pygame.image.load()方法来加载图片资源，加载后的图片资源将变为 Surface 对象。加载 cat.png 图片并赋值给 catSurface 对象，代码如下：

```
catSurface = pygame.image.load("cat.png")
```

上述代码对 cat.png 图片进行了加载，需要说明的是，Pygame 支持大多数图片格式，但是图片格式如果过于小众，则存在加载不出来的情况。Pygame 支持的图片格式有 JPG、PNG、GIF、PCX、TGA、TIF、LBM、PBM、PGM、PPM 和 XPM。

读者在编程中可能碰到这样的问题，图片资源的分辨率太大了，在 Pygame 的主画布里如果按照 1∶1 加载，则图片会过大，不符合编程中需要的图片大小。最容易解决这个问题的办法是让美工按照需求调整图片的分辨率，但图片资源往往要在各个游戏场景里复用，其分辨率也不尽相同，让美工按照各个场景需求调整图片分辨率似乎不大现实。幸运的是，Pygame 提供了 transform 方法来满足游戏场景里缩放图片资源的 Surface 对象，从而解决图片分辨率的问题。将小猫 Surface 对象缩小成 94×153 像素的 Surface 对象，并赋值给 catSurface，代码如下：

```
catSurface = pygame.image.load("cat.png")
catSurface = pygame.transform.scale(catSurface,(94,153))
```

3. Surface 对象属性获取

Pygame 游戏编程的实质就是对多个 Surface 对象的灵活运用。对于各个 Surface 对象，游戏编程中经常需要获取其位置、高度等信息，从而进行控制，为此，Pygame 提供了众多的 Surface 对象属性获取方法来满足游戏开发者的需求。Surface 对象常见属性如表 5-2 所示。

表 5-2 Surface 对象常见属性

方 法	方法描述
Surface.get_width()	获取 Surface 宽度,单位为像素
Surface.get_height()	获取 Surface 高度,单位为像素
Surface.get_size()	获取 Surface 的(width,height)尺寸
Surface.get_rect()	获取 Surface 的 Rect 矩形对象,Rect 矩形对象的值为(0,0,width,height)

加载 cat.png 图片,并使用 Surface 的属性及方法进行输出,代码如下:

```
#第 5 章/catBall/catShow.py
import pygame
pygame.init()
SIZE = WIDTH, HEIGHT = (800, 600)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption('小猫顶球')
WHITE = (255, 255, 255)
catSurface = pygame.image.load("cat.png")
print(catSurface.get_width())
print(catSurface.get_height())
print(catSurface.get_size())
print(catSurface.get_rect())
catSurface = pygame.transform.scale(catSurface,(100,200))
print(catSurface.get_width())
print(catSurface.get_height())
print(catSurface.get_size())
print(catSurface.get_rect())
screen.fill(WHITE)
while True:
    pygame.display.update()
```

上述代码的运行结果如图 5-13 所示。

```
pygame 2.0.1 (SDL 2.0.14, Python 3.8.7)
Hello from the pygame community. https://www.pygame.org/contribute.html
765
1165
(765, 1165)
<rect(0, 0, 765, 1165)>
100
200
(100, 200)
<rect(0, 0, 100, 200)>

Process finished with exit code -1
```

图 5-13 Surface 对象属性获取

从图 5-13 可知,直接用 pygame.image.load()方法加载图片后,得到的是图片的原始宽度、高度、尺寸、Rect 等属性。使用 pygame.transform.scale()方法对 Surface 对象进行调整后,获取的就是调整后的各个属性。

5.6 显示小猫等 Surface 对象

4min

在 5.5 节使用 pygame.image.load()方法加载了小猫等图片资源，并使用 pygame.transform.scale()方法对加载的图片资源根据需求进行了缩放。加载好的图片资源该如何显示到画布上，并且其在画布上的位置怎么确定？

在 5.5 节的最后使用 Surface.get_rect()方法得到了 Surface 对象的 Rect 对象，Pygame 使用 Rect 对象的位置来定位 Surface 的坐标位置。换句话说，如果改变 Rect 对象的位置，则代表 Surface 对象的图片位置也将随之跟着改变。Rect 对象是图像显示的关键，下面看一下 Rect 对象的相关知识。

5.6.1 创建 Rect 对象

Rect 对象是一个四元组，其由 4 个数字来表示一个矩形区域，例如(100,100,200,300)表示的 Rect 对象如图 5-14 所示。

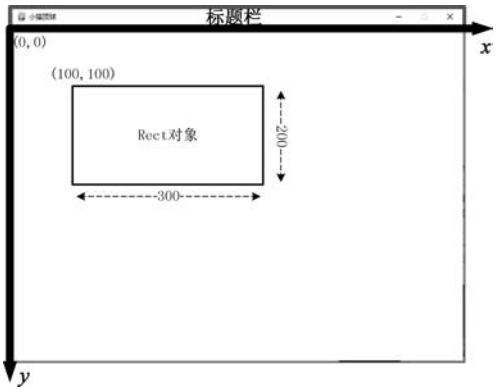

图 5-14 (100,100,200,300)Rect 对象图示

从图 5-14 可知，在 Rect 对象的四元组中，前两个数字表示 Rect 对象的左上角坐标为(100,100)，第 3 个数字表示 Rect 对象的宽度为 300 像素，第 4 个数字表示 Rect 对象的高度为 200 像素。Pygame 还支持用户自己创建 Rect 对象，其语法如下：

```
Rect(left,top,width,height)
```

(1) left：从左上角坐标原点开始向右计算的 x 坐标。

(2) top：从左上角坐标原点开始向下计算的 y 坐标。

(3) width：Rect 对象的宽度。

(4) height：Rect 对象的高度。

5.5 节加载图片资源为 Surface 对象后，通过 Surface 对象的 get_rect()方法得到图片的 Rect 对象，此 Rect 对象将继承图片的宽度和高度等信息，其四元组信息为(0,0,图片宽度,图片高度)，如果改变 Rect 对象的位置信息，则图片的位置将随之改变。

5.6.2 Rect 对象位置属性

从 5.6.1 节可知，改变 Rect 对象的位置后，得到 Rect 对象的 Surface 对象的位置也随之改变，那么 Rect 对象都有哪些位置属性？其位置属性如图 5-15 所示。

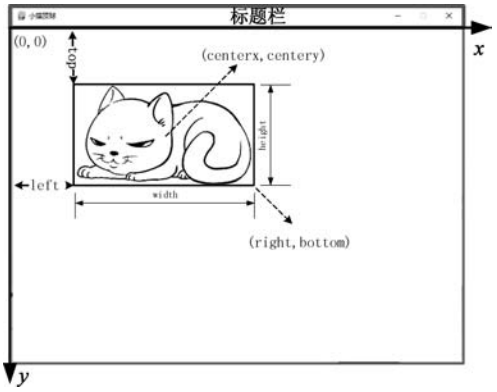

图 5-15 Rect 对象位置属性

图 5-15 为加载图片资源产生的 Surface 对象在更改位置后得到的 Rect 对象。虽然 Rect 对象有很多位置属性，但掌握了如图 5-15 所示的 Rect 对象的位置属性已经足够读者进行游戏编程，其每个位置的含义如下。

(1) top：Rect 对象的左上角 y 坐标，也可以使用 y 代替。

(2) left：Rect 对象的左上角 x 坐标，也可以使用 x 代替。

(3) width：Rect 对象的宽度，也可以使用 w 代替。

(4) height：Rect 对象的高度，也可以使用 h 代替。

(5) right：Rect 对象右下角 x 坐标。

(6) bottom：Rect 对象右下角 y 坐标。

(7) centerx：Rect 对象中心点的 x 坐标。

(8) centery：Rect 对象中心点的 y 坐标。

Rect 对象的属性在代码中都可以改变。用 PyCharm 打开 catBall 工程后，新建 catSleep.py 文件，加载 catSleep.jpg 图片，并将其 Rect 对象的 top 移动到 200 像素，并且将 left 移动到 300 像素，代码如下：

```
#第 5 章/catBall/catSleep.py
import pygame
pygame.init()
SIZE = WIDTH, HEIGHT = (800, 600)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption('小猫顶球')
WHITE = (255, 255, 255)
catSleepSurface = pygame.image.load("catSleep.jpg")
catSleepRect = catSleepSurface.get_rect()
catSleepRect.top = 200
catSleepRect.left = 300
```

```
screen.fill(WHITE)
while True:
    screen.blit(catSleepSurface,catSleepRect)
    pygame.display.update()
```

上述代码的运行结果如图 5-16 所示。

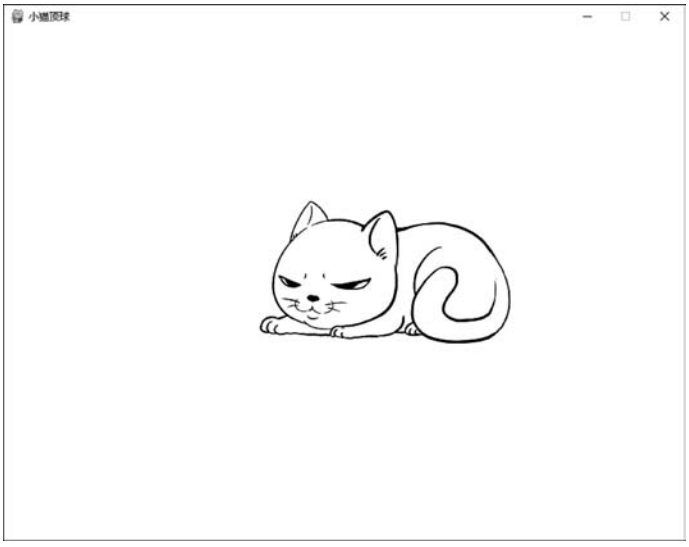

图 5-16 更改位置后的 Rect 对象

读者在输入代码并运行时，可能已经发现，当修改了 Rect 对象属性值后，其他的属性也会随之联动。如果代码中修改了 Rect 对象的 top 和 left，则此时 centerx 和 centery 的坐标会变成什么？读者可以编写代码加以验证。

5.6.3 Rect 对象进行移动

小猫顶球游戏中要控制小猫来顶球，如何让小猫和球运动起来？Rect 对象提供了移动的方法，又从前述章节可知，小猫和球都是 Surface 对象，都有 Rect 对象属性，因此可以通过 Rect 对象的移动来使 Surface 对象移动，Rect 对象的移动方法如表 5-3 所示。

表 5-3 Rect 对象移动方法

方 法	方法描述
move(x,y)	向 x 和 y 坐标移动 Rect 对象。如果 x 为正值，则向右移动，如果 x 为负值，则向左移动；如果 y 为正值，则向下移动，如果 y 为负值，则向上移动；x 和 y 值必须为整数；方法会返回一个新的 Rect 对象，原对象不变
move_ip(x,y)	向 x 和 y 坐标移动 Rect 对象。如果 x 为正值，则向右移动，如果 x 为负值，则向左移动；如果 y 为正值，则向下移动，如果 y 为负值，则向上移动；x 和 y 值必须为整数

万事开头难，先编写代码尝试着让一个 Surface 对象移动起来吧。

接下来将通过 Rect 对象的 move()方法实现一个在屏幕上跳动的小球。在 catBall 工程里新建 movingBall.py 文件，输入的代码如下：

```
#第5章/catBall/movingBall.py
import pygame,sys
pygame.init()
SIZE = WIDTH, HEIGHT = (800, 600)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption('跳动的小球')
WHITE = (255, 255, 255)
x,y = 3,3
ballSurface = pygame.image.load("ball.gif")
ballRect = ballSurface.get_rect()
screen.fill(WHITE)
tick = pygame.time.Clock()  #创建时钟对象(可以控制游戏的循环频率)
while True:
    tick.tick(60)  #每秒循环60次
    ballRect = ballRect.move(x, y)
    screen.fill(WHITE)
    if ballRect.left > screen.get_width() - ballRect.width or ballRect.left < 0:
        x = -x
    if ballRect.top < 0 or ballRect.top > screen.get_height() - ballRect.height:
        y = -y
    screen.blit(ballSurface,ballRect)
    pygame.display.update()
```

运行上边的代码后,会出现如图5-17所示的场景:小球在屏幕里跳动,碰到边界后自动改变跳动方向。

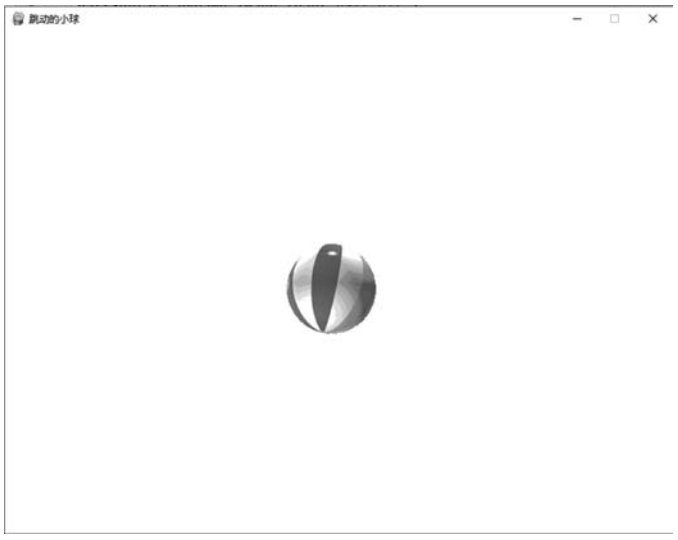

图5-17 跳动的小球

代码中将 x 和 y 的值设为3,每次小球的Surface对象调用move(x,y)方法时,小球将向 x 坐标和 y 坐标移动3像素。

pygame.time.Clock()方法可以帮助程序确定要以最大多少帧的速率运行,这样在游戏的每一次while循环后会设置一个暂停,以防程序运行过快。因为计算机的配置不同,所

以编程的时候需要使用这种方法来让计算机以一个固定的速度运行。tick.tick(60)方法设置计算机程序每秒运行60次，这个值设置得越高，程序运行得越快，反之则越慢。

if ballRect.left > screen.get_width()-ballRect.width or ballRect.left < 0 语句对小球的Rect对象的left值进行判断，如果小球的Rect对象的left值大于画布右边界或小于左边界，则改变小球的 x 轴移动方向。

if ballRect.top < 0 or ballRect.top > screen.get_height()-ballRect.height 语句对小球的Rect对象的top值进行判断，如果小球的Rect对象的top值大于画布下边界或小于上边界，则改变小球的 y 轴移动方向。

screen.blit(ballSurface,ballRect)语句为Surface上的绘制语句。screen为游戏的主画布，本行代码将小球的Surface对象以小球的Rect位置为基准绘制到screen主画布上。

pygame.display.update()语句为更新屏幕语句，每次while循环都必须运行此语句，否则屏幕上将不会显示任何Surface对象。

读者可能好奇，小球为什么能移动？它的移动机制到底是什么？其实小球的移动非常类似于读者看到的动画片。大家知道，动画片通过一秒播放30张或以上的连续的图片来让眼睛误认为是动画，从而形成了移动动画效果。Pygame不断地把Surface对象以非常小的坐标改变绘制到屏幕上，当绘制的画面频率大于30画面/s时，人的眼睛就认为Surface对象进行了移动。需要注意的是，每次绘制前需要把画布用底色填充，从而覆盖上次的绘制结果。

每次绘制前，如果不用底色填充画布会发生什么情况？读者可以尝试将while循环里的screen.fill(WHITE)语句注释掉，其运行结果如图5-18所示。

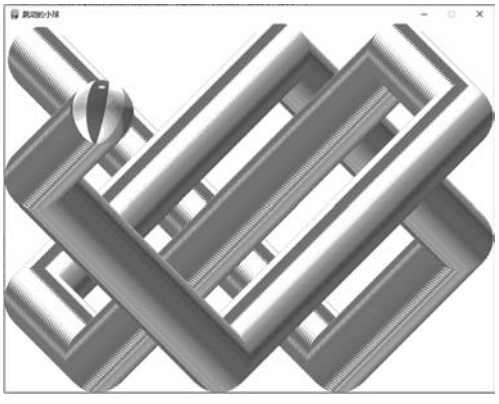

图5-18 每次循环时不填充画布的结果

5.7 键盘和鼠标事件响应

3min

读者可能已经发现，本章前面章节的游戏代码运行后无法关闭，必须通过停止程序任务的方式结束游戏，这个问题该如何解决？Pygame提供了事件处理模块来帮助用户对游戏程序进行控制，Pygame提供的事件处理模块为pygame.event、pygame.key、pygame.mouse。

1. pygame.event

操作系统采取消息驱动机制来对键盘输入、鼠标移动等进行响应，每当按下键盘和移动鼠标时都会产生响应的 Event 事件，操作系统会自动将产生的 Event 事件存储到消息队列中。

Pygame 使用 pygame.event.get()方法从消息队列中获取操作系统的 event 事件，得到 event 事件后，可根据 event 事件的类型而对键盘、鼠标等进行响应。pygame.event.get()方法得到的 event 事件共有两类属性，分别是 type 和 dict。需要注意的是，消息队列严重依赖 pygame.display 模块，假如 pygame.display 模块没有正确初始化，消息队列则可能工作不正常。消息队列最多只能容纳 128 个 event 事件，当消息队列容量到达 128 像素后，新的事件将不会得到存储。

常用的 event 事件的 type 和 dict 属性如表 5-4 所示，在游戏编程中可以通过 type 和 dict 值来处理 event 事件。

表 5-4 常用的 event 事件的 type 和 dict 属性

type 属性	dict 属性
QUIT	None
ACTIVEEVENT	gain、state
KEYDOWN	key、mod、unicode、scancode
KEYUP	key、mod
MOUSEMOTTON	pos、rel、buttons
MOUSEBUTTONUP	pos、button
MOUSEBUTTONDOWN	pos、button
VIDEORESIZE	size、w、h
VIDEOEXPOSE	None
USEREVENT	code

表 5-4 对常用的 event 事件的 type 和 dict 属性进行了描述，有了这个表格，就可以响应玩家的程序关闭事件了。

以下代码将保证用户单击主画布右上方的“×”按钮时，程序会终止运行，代码如下：

```
#第 5 章/catBall/catBall.py
import pygame
import sys
pygame.init()
SIZE = WIDTH, HEIGHT = (800, 600)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption('小猫顶球')
WHITE = (255, 255, 255)
screen.fill(WHITE)
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    pygame.display.update()
```

在上述代码中引入了 Python 的 sys 模块，sys 模块有很多重要的方法，读者用到的时候可以去查阅相关文献，此处使用 sys.exit()方法来退出程序。在程序中使用 for 循环得到每条消息队列的事件也是以后游戏编程中常用的一种方法，读者需掌握。

2. pygame.key 模块

pygame.key 模块是 Pygame 关于键盘操作的响应模块，其常见的 key 事件和含义如表 5-5 所示。

表 5-5 常见的 key 事件和含义

事 件	含 义
get_focused()	如果有键盘输入，则事件返回值为 True
get_pressed()	得到键盘按键的所有状态
get_modes()	检测是否有组合键被按下
set_mods()	将某些组合键设置为被按下状态
set_repeat()	控制重复响应持续按下按键的时间
get_repeat()	得到重复响应按键的参数

小猫顶球游戏要使用左右方向键控制小猫的左右移动，有了 pygame.key 模块知识就可以完成控制小猫左右移动的编码，代码如下：

```
import pygame
import sys
pygame.init()
SIZE = WIDTH, HEIGHT = (800, 600)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption('小猫顶球')
#加载小猫图片
catSurface = pygame.image.load("cat.png")
#将小猫图片缩小为 94 像素 × 153 像素
catSurface = pygame.transform.scale(catSurface,(94,153))
catRect = catSurface.get_rect()
#将小猫的 Rect 对象放置到画布下方的中心
catRect.left = WIDTH//2 - catRect.width//2
catRect.top = HEIGHT - catRect.height
WHITE = (255, 255, 255)
screen.fill(WHITE)
tick = pygame.time.Clock()
while True:
    tick.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    screen.fill(WHITE)
    keyPressed = pygame.key.get_pressed()
    #小猫向左移动，当移到最左边界时，不能继续移动
    if keyPressed[pygame.K_LEFT]:
        catRect.left -= 2
```

```
    if catRect.left <= 0:
        catRect.left = 0
# 小猫向右移动,当移到最右边界时,不能继续移动
    if keyPressed[pygame.K_RIGHT]:
        catRect.left += 2
        if catRect.left >= WIDTH - catRect.width:
            catRect.left = WIDTH - catRect.width
    screen.blit(catSurface,catRect)
    pygame.display.update()
```

运行上述代码,结果如图 5-19 所示,当玩家按键盘的左右方向键时,小猫也将左右移动。

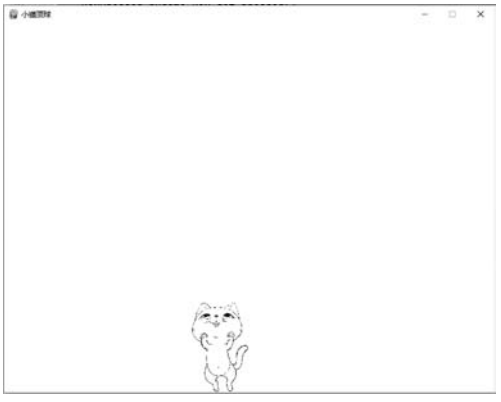

图 5-19　可左右移动的小猫

图像加载生成的 Surface 对象默认的 Rect 位置在画布的左上角。但根据游戏设计,Surface 对象的初始位置都应该在一个特定的地方,例如小猫顶球游戏就要求小猫的初始位置在画布下方的中心。游戏编码时可以通过 Surface 对象的 Rect 位置来初始化 Surface 的位置,例如代码中使用 catRect.left＝WIDTH//2-catRect.width//2 和 catRect. top＝HEIGHT-catRect. height 来使位置下方居中。

代码中使用 keyPressed[pygame. K_LEFT]和 keyPressed[pygame. K_RIGHT]来判断按键是否是左和右方向键,pygame. K_LEFT 和 pygame. K_RIGHT 是 Pygame 里提供的左和右方向键的键定义,Pygame 给键盘的每个键都提供了键定义,游戏编码时可以通过键定义来判断所按的键。常见的键定义如表 5-6 所示。

表 5-6　常见的键定义

键　定　义	含　　义	键　定　义	含　　义
K_BACKSPACE	空格	K_A	a 键
K_TAB	Tab 键	K_F1	F1 键
K_RETURN	回车键	K_NUMLOCK	Num Lock 键
K_0	数字 0 键	K_LCTRL	左 Ctrl 键
K_LEFT	方向左键	K_RCTRL	右 Ctrl 键
K_RIGHT	方向右键	K_POWER	Power 键

当小猫左右移动时，需要对画布的边界情况进行判断，防止小猫不停地移动，以至移到画布外部。

5.8 小猫和球类碰撞检测

小猫顶球游戏必须对小猫和球的碰撞加以检测，一种可行的方法是当小猫的 Rect 对象和球的 Rect 对象有重叠的时候，认为小猫顶到了球。使用这种方法需要时时对两个 Rect 对象的矩形区域进行位置判断，如果加入更多的球，则数学计算将较为烦琐。幸运的是，Pygame 提供了更好的机制进行碰撞检测，这就是接下来要学习的 Sprite。

5.8.1 类与类的继承

Sprite 又称为“精灵”，指游戏中经常出现的可视化对象，小猫顶球游戏中的小猫和球都可称为 Sprite 对象，Pygame 对 Sprite 类的碰撞检测提供了调用方法，那么什么是类？

类是对具有共同特性对象的一种抽象。例如，小猫顶球游戏目前只顶一个球，后期如果想顶多个球，则在采用传统方法时，每个球都要进行单独定义，代码将变得冗余不堪。如果找出各个要顶的球的共有特征，并将其抽象为球这个类，则以后想生成更多球时，可直接生成球这个类的对象，代码将简洁且易维护。

类有很多抽象术语，了解这些抽象术语可以帮助读者更好地掌握类的用法。

(1) 类属性：类中所定义的属性，有共有的类属性和私有的类属性两种。

(2) 构造函数：生成类对象时，负责初始化类的各个参数。

(3) 方法：类的方法就是类中定义的函数，可以完成相应的功能，如改变类的状态，进行某个计算等。

(4) 继承：一个派生类继承一个或多个基类，派生类可以继承父类的属性和方法。

(5) 实例化：创建一个类的实例对象。

(6) 封装：将类变成一个黑匣子，外部无法看清内部的工作细节。

1. 类的定义和实例化对象

类定义的语法格式如下：

```
class 类名:
    类体
```

上述语法格式中 class 是关键字，定义类必须以 class 开头，类名为类的名字，类名也必须符合 Python 变量命名规则，一般来讲类名的首字母用大写。当然，读者如果喜欢用中文命名类名也可以，Python 完美支持中文类名。

有了类名后，可以通过“对象名＝类名([参数列表])”的语法来实例化类的对象。

以下代码将创建 Ball 类，并实例化两个 Ball 对象，代码如下：

```
#第 5 章/catBall/classBall.py
class Ball:
    def __init__(self,pos,color):
        self.pos = pos
```

```
        self.color = color
    def print(self):
        print(self.color + "球的位置在" + str(self.pos))

redBall = Ball((30,30),"red")
blueBall = Ball((100,100),"blue")
redBall.print()
blueBall.print()
```

运行上述代码，结果如图 5-20 所示。

```
red球的位置在(30, 30)
blue球的位置在(100, 100)

Process finished with exit code 0
```

图 5-20　Ball 类与其对象

在上述代码中，首先定义了一个 Ball 类，类中有一个构造函数和一个类方法，其中构造函数负责 Ball 类中 pos 和 color 两个变量的初始化，类方法负责显示类的属性。

类的构造函数必须以__init__作为开始，括号里的 self 不可省略，其表明此函数属于类。代码中类的 pos 属性和 color 属性需要在类构造时初始化，需要注意的是，所有类中的变量必须以 self 作为前缀。

类定义完成后，代码通过 redBall = Ball((30, 30)," red") 和 blueBall = Ball((100, 100),"blue")生成两个 Ball 对象，每个 Ball 对象都有自己的 pos 和 color 属性，这一点从两个类对象的 print()方法调用也可以看出。

读者需要牢记，类方法中的参数里必须有 self 关键字，这也是和前边所学的函数的最大区别。

并不是所有类都必须有构造函数和方法，没有构造函数的类，生成对象时也不必带参数。以下代码就是一个不含有构造函数的类：

```
class Ball:
    def print(self):
        print("我是一个小球")
redBall = Ball()
redBall.print()
```

2. 类的继承

类的重要的特点就是继承。通过类的继承，开发者可以在已有类的基础上加上自己需要的部分，从而使类的使用更加灵活。在继承关系中，已有的类称为基类，新设计的类称为派生类。派生类可以继承父类的公有属性和方法，派生类可以同时继承多个基类。

设计 Animal 基类和 Cat 子类，并实例化 Cat 对象，代码如下：

```
#第 5 章/catBall/animal.py
class Animal:
    def __init__(self):
        self.head = True #具有大脑
    def speak(self):
        print("动物可以叫")
```

```
class Cat(Animal):
    def speak(self):
        print("猫咪喵喵叫")
    def jump(self):
        print("猫咪跳得高")

blackCat = Cat()
print(blackCat.head)
blackCat.speak()
blackCat.jump()
```

运行上述代码，结果如图 5-21 所示。

```
True
猫咪喵喵叫
猫咪跳得高

Process finished with exit code 0
```

图 5-21 类继承运行结果

从运行结果可知，Cat 类继承了 Animal 基类后，基类里的 head 变量被 Cat 类继承下来。基类里的 speak()方法，如果 Cat 类有同名方法，则 Cat 类的同名方法将覆盖基类里的方法。Cat 类除了继承基类的方法，也可以有自己的方法。

5.8.2 小猫和球类

有了 5.8.1 节类的知识，可以创建小猫和球两个类并实现对象，代码如下：

```
#第 5 章/catBall/catBall.py
#创建 Cat 类
class Cat(pygame.sprite.Sprite):
def __init__(self, image):
#图片加载
        self.image = pygame.image.load(image)
              #缩小图片
        self.image = pygame.transform.scale(self.image, (94, 153))
#得到 Rect 对象
        self.rect = self.image.get_rect()
        self.rect.left = WIDTH //2 - self.rect.width //2
        self.rect.top = HEIGHT - self.rect.height

    #向左移动
    def moveLeft(self):
        self.rect.left -= 2
        if self.rect.left <= 0:
            self.rect.left = 0

#向右移动
    def moveRight(self):
        self.rect.left += 2
        if self.rect.left >= WIDTH - self.rect.width:
            self.rect.left = WIDTH - self.rect.width

#显示 Cat
```

```
    def display(self):
        screen.blit(self.image, self.rect)


#创建 Ball 类
class Ball(pygame.sprite.Sprite):
    def __init__(self, image):
        self.image = pygame.image.load(image)
        self.rect = self.image.get_rect()

    def move(self, moveStep):
        self.rect = self.rect.move(moveStep)

    def display(self):
        screen.blit(self.image, self.rect)


#实例化 Cat 和 Ball 的对象
boy = Cat('cat.png')
ball = Ball('ball.gif')
```

从代码可知，Cat 类和 Ball 类都继承自 Sprite，类都有 image 和 rect 变量，通过这两个个变量来存储图片的 Surface 和 Surface 产生的 Rect 对象，Cat 类通过 moveLeft()和 moveRight()方法左右移动，Ball 类通过 move()方法进行移动，两个类都通过 display()方法在画布上显示。

5.8.3 使用碰撞函数进行碰撞检测

Cat 类和 Ball 类都继承自 Sprite 类，Pygame 对 Spirte 类有专门的碰撞检测函数，其函数为 collide_rect(sprite1,sprite2)，当 sprite1 和 sprite2 的 Rect 对象有重合时，函数的返回值为 True，当无重合时，返回值为 False。

根据上述知识，可以使用下边的代码对小猫和球进行碰撞检测，代码如下：

```
b = pygame.sprite.collide_rect(boy, ball)
```

当 b 为 True 时，小猫顶到了球；当 b 为 False 时，小猫没有顶到球。

5.9 信息显示和音效播放

小猫顶球游戏的一个重要的环节就是显示游戏分数，当一局游戏结束后，需提示玩家按照操作进行下一局的游戏，这就需要掌握 Pygame 显示文字的方法了。

从前边的小猫和球的显示可知，Pygame 采用的都是 Surface 对象和 Surface 对象对应的 Rect 对象相结合的方法来显示图片，显示文字也没有什么更好的办法，也只能采取这样的方法。

5.9.1 字体显示

1. 生成字体对象

Pygame 提供了字体类 pygame.font.Font，其构造函数有两个参数，第 1 个参数是字体文件名称，第 2 个参数为字体大小。每一台计算机所拥有的字体也不尽相同，那么该如何保证任一台计算机都可以正确地显示？使用 Pygame 提供的 get_default_font()方法可以得到操作系统的默认字体。以下代码得到了 ff 字体对象，其中 font_size 由主程序传递，代码如下：

```
ff = pygame.font.Font(pygame.font.get_default_font(), font_size)
```

2. 字体对象生成 Surface

字体对象的 render()方法可以生成 Surface 对象，其方法参数为

render(text,antialias,color,background = None)。

(1) text：要显示的文字信息，仅支持一行文本。

(2) antialias：是否打开抗锯齿功能，如果打开抗锯齿功能，则文字将更平滑。

(3) color：文字颜色，支持 RGB 三元组。

(4) background：背景颜色，如果为 None，则文字背景将是透明的。

以下代码将通过 render()方法生成 Surface 对象，其中 text 和 font_color 由主程序传递，代码如下：

```
textSurface = ff.render(text, True, font_color)
```

3. Surface 生成 Rect 并居中显示

有了 Surface 对象，生成 Rect 对象并居中显示就简单了，代码如下：

```
textRect = textSurface.get_rect()
textRect.center = (xPos, yPos)
```

5.9.2 字体显示函数

字体显示在小猫顶球游戏中要频繁地用到，可以将其封装成字体显示函数，其具体的代码如下：

```
def draw_text(text, xPos, yPos, font_color, font_size):
    """ 绘制文本，xPos 和 yPos 为坐标，font_color 为字体颜色，font_size 为字体大小"""
ff = pygame.font.Font(pygame.font.get_default_font(), font_size)
    textSurface = ff.render(text, True, font_color)
    textRect = textSurface.get_rect()
    textRect.center = (xPos, yPos)
    screen.blit(textSurface, textRect)
```

5.9.3 音效播放

当小猫顶到球时，播放出相应音效会大大提高游戏的趣味性，Pygame 通过 mixer 模块来支持音效播放，其调用非常简单，代码如下：

```
sound = pygame.mixer.Sound("dong.wav")
sound.play()
```

sound 为 Pygame 产生的 Sound 类的对象，当需要播放 dong.wav 这个音效时，调用 sound 对象的 play()方法即可。

5.10 “小猫顶球”游戏主程序完善

至此，“小猫顶球”游戏各个主要功能都已经得到实现。接下来完成主程序，从而把各个功能串联起来，完整代码如下：

```
#第5章/catBall/catBall.py
import pygame, sys
pygame.init()
SIZE = WIDTH, HEIGHT = (800, 600)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption('小猫顶球')
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255,0,0)
score = 0
scoreFont = pygame.font.Font(pygame.font.get_default_font(), 20)
scoreDis = scoreFont.render(str(score),1,BLACK)
scorePos = [WIDTH//2,0]
gameOver = False


def draw_text(text, xPos, yPos, font_color, font_size):
    """ 绘制文本，xPos 和 yPos 为坐标，font_color 为字体颜色，font_size 为字体大小"""
    ff = pygame.font.Font(pygame.font.get_default_font(), font_size)
    textSurface = ff.render(text, True, font_color)
    textRect = textSurface.get_rect()
    textRect.center = (xPos, yPos)
    screen.blit(textSurface, textRect)


class Cat():
    def __init__(self, image):
        self.image = pygame.image.load(image)
    #缩放图片
        self.image = pygame.transform.scale(self.image, (94, 153))
        self.rect = self.image.get_rect()
```

```
        self.rect.left = WIDTH //2 - self.rect.width //2
        self.rect.top = HEIGHT - self.rect.height

    def moveLeft(self):
        self.rect.left -= 2
        if self.rect.left <= 0:
            self.rect.left = 0

    def moveRight(self):
        self.rect.left += 2
        if self.rect.left >= WIDTH - self.rect.width:
            self.rect.left = WIDTH - self.rect.width

    def display(self):
        screen.blit(self.image, self.rect)


class Ball():
    def __init__(self, image):
        self.image = pygame.image.load(image)
        self.rect = self.image.get_rect()

    def move(self, moveStep):
        self.rect = self.rect.move(moveStep)

    def display(self):
        screen.blit(self.image, self.rect)


boy = Cat('cat.png')
ball = Ball('ball.gif')
sound = pygame.mixer.Sound("dong.wav")
moveStep = [2, 2]
angle = 0
tick = pygame.time.Clock()      #创建时钟对象(可以控制游戏循环的频率)

while True:
    tick.tick(120)              #每秒循环 120 次
for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    screen.fill(WHITE)
    keyPressed = pygame.key.get_pressed()
    if keyPressed[pygame.K_SPACE]:
        gameOver = False
        score = 0
        ball.rect.left,ball.rect.top = 0,0
        boy.rect.left = WIDTH //2 - boy.rect.width //2
        boy.rect.top = HEIGHT - boy.rect.height
```

```
    if not gameOver:
        if keyPressed[pygame.K_LEFT]:
            boy.moveLeft()
        if keyPressed[pygame.K_RIGHT]:
            boy.moveRight()
        ball.move(moveStep)
        if ball.rect.left <= 0 or ball.rect.right >= WIDTH:
            moveStep[0] = -moveStep[0]
        if ball.rect.top <= 0:
            moveStep[1] = -moveStep[1]
        if ball.rect.bottom >= HEIGHT:
            gameOver = True
        b = pygame.sprite.collide_rect(boy, ball)
        if b:
            sound.play()
            moveStep[1] = -moveStep[1]
            score += 10
        boy.display()
        ball.display()
        draw_text(str(score), WIDTH //2, 15, BLACK, 25)
    else:
        draw_text("Your Score is:" + str(score),WIDTH//2,HEIGHT//2 - 150,BLACK,50)
        draw_text("GAME OVER",WIDTH//2,HEIGHT//2,BLACK,100)
        draw_text("Press Space Begin New Game",WIDTH//2,HEIGHT//2 + 150,RED,30)


    pygame.display.update()
```

主程序中通过判断 gameOver 变量是否为真来判断游戏是否结束，当游戏结束的时候，监测用户的按键是否为空格键，如果是空格键，则游戏分数清零，小猫重新回到初始位置。

笔者将小猫顶到球的每次得分设置为 10，读者可以通过更改 score ＋＝ 10 语句设置每次顶球的分数。

5.11 小结

本章主要介绍了小猫顶球游戏的具体实现，同时对本章涉及的 Pygame 里的图形界面初始化、Surface 对象显示、键盘和鼠标事件响应、碰撞检测、声音显示、音效播放等知识点进行了简要介绍。

学习本章后，读者应能掌握 Pygame 图形界面编程思想、Surface 对象的相关知识以及 Pygame 的 Event 事件处理方法等。

读者可以用本章介绍的知识完成接红包等常见游戏。

本章知识可为后续章节的学习打下良好基础。