

控制，另外还有其他一些模块，例如 pygame.display 是显示模块、pygame.keyboard 是键盘模块、pygame.mouse 是鼠标模块，如表 17-1 所示。

表 17-1 Pygame 软件包中的模块

模 块 名	功 能
pygame.cdrom	访问光驱
pygame.cursors	加载光标
pygame.display	访问显示设备
pygame.draw	绘制形状、线和点
pygame.event	管理事件
pygame.font	使用字体
pygame.image	加载和存储图片
pygame.joystick	使用游戏手柄或者类似的东西
pygame.key	读取键盘按键
pygame.mixer	声音
pygame.mouse	鼠标
pygame.movie	播放视频
pygame.music	播放音频
pygame.overlay	访问高级视频叠加
pygame	Python 模块，专为电子游戏设计
pygame.rect	管理矩形区域
pygame.sndarray	操作声音数据
pygame.sprite	操作移动图像
pygame.surface	管理图像和屏幕
pygame.surfarray	管理点阵图像数据
pygame.time	管理时间和帧信息
pygame.transform	缩放和移动图像

建立 Pygame 项目和建立其他 Python 项目的方法一样，在 IDLE 或文本编辑器中新建一个空文档，需要告诉 Python 该程序用到了 Pygame 模块。

为了实现此目的，这里使用一个 import 指令，该指令告诉 Python 载入外部模块。例如输入下面两行在新项目中引入必要的模块：

```
import pygame, sys, time, random  
from pygame.locals import *
```

第 1 行引入 Pygame 的主要模块、sys 模块、time 模块和 random 模块。

第 2 行告诉 Python 载入 pygame.locals 的所有指令使它们成为原生指令，这样在使用这些指令时就不需要用全名调用。

由于硬件和游戏的兼容性或者请求的驱动没有安装的问题，有些模块可能在某些平台上不存在，可以用 None 测试一下。例如测试字体是否载入：

```
if pygame.font is None:
```



```
print("The font module is not available!")
pygame.quit() #如果没有则退出 Pygame 的应用环境
```

下面对常用模块进行简要说明。

① pygame.surface

该模块中有一个 surface() 函数，surface() 函数的一般格式如下：

```
pygame.surface((width, height), flags=0, depth=0, masks=None)
```

它返回一个新的 surface 对象。这里的 surface 对象是一个有确定尺寸的空图像，可以用它进行图像的绘制与移动。

② pygame.locals

在 pygame.locals 模块中定义了 Pygame 环境中用到的各种常量，而且包括事件类型、按键和视频模式等的名字，在导入所有内容（from pygame.locals import *）时用起来很安全。

如果用户知道需要的内容，也可以导入具体的内容（例如 from pygame.locals import FULLSCREEN）。

③ pygame.display

pygame.display 模块包括处理 Pygame 显示方式的函数，其中包括普通窗口和全屏模式。游戏程序通常需要下面的函数：

1) flip()/update()

(1) flip(): 更新显示。一般来说，在修改当前屏幕的时候要经过两步，首先需要对 get_surface() 函数返回的 surface 对象进行修改，然后调用 pygame.display.flip() 更新显示以反映所做的修改。

(2) update(): 在只想更新屏幕一部分的时候使用 update() 函数，而不是 flip() 函数。

2) set_mode()

该函数建立游戏窗口，返回 surface 对象。它有 3 个参数，第 1 个参数是元组，用于指定窗口的尺寸；第 2 个参数是标志位，具体含义如表 17-2 所示，例如 FULLSCREEN 表示全屏，默认值为不对窗口进行设置，读者可根据需要选用；第 3 个参数为色深，用于指定窗口的色彩位数。

表 17-2 set_mode 的窗口标志位的参数取值

窗口标志位	功 能
FULLSCREEN	创建一个全屏窗口
DOUBLEBUF	创建一个“双缓冲”窗口，建议在 HWSURFACE 或者 OPENGL 时使用
HWSURFACE	创建一个硬件加速的窗口，必须和 FULLSCREEN 同时使用
OPENGL	创建一个 OPENGL 渲染的窗口
RESIZABLE	创建一个可以改变大小的窗口
NOFRAME	创建一个没有边框的窗口

3) set_caption()

该函数设定游戏程序的标题。当游戏以窗口模式（对应于全屏）运行时尤其有用，因



为该标题会作为窗口的标题。

4) get_surface()

该函数返回一个可用来画图的 surface 对象。

④ pygame.font

pygame.font 模块用于表现不同字体，可以用于文本。

⑤ pygame.sprite

pygame.sprite 模块有两个非常重要的类——sprite 精灵类和 group 精灵组。

sprite 精灵类是所有可视游戏的基类。为了实现自己的游戏对象，需要子类化 sprite，覆盖它的构造函数，以设定 image 和 rect 属性（决定 sprite 的外观和放置的位置），再覆盖 update()方法。在 sprite 需要更新的时候可以调用 update()方法。

group 精灵组的实例用作 sprite 精灵对象的容器。在一些简单的游戏中，只要创建名为 sprites、allsprite 或是其他类似的组，然后将所有 sprite 精灵对象添加到上面即可。当 group 精灵组对象的 update()方法被调用时会自动调用所有 sprite 精灵对象的 update()方法。group 精灵组对象的 clear()方法用于清理它包含的所有 sprite 对象（使用回调函数实现清理），group 精灵组对象的 draw()方法用于绘制所有的 sprite 对象。

⑥ pygame.mouse

该模块用来管理鼠标。

- pygame.mouse.set_visible(False/true): 隐藏/显示鼠标光标。
- pygame.mouse.get_pos(): 获取鼠标位置。

⑦ pygame.event

pygame.event 模块会追踪鼠标单击、鼠标移动、按键按下和释放等事件。其中，pygame.event.get()可以获取最近事件列表。

⑧ pygame.image

这个模块用于处理保存在 GIF、PNG 或者 JPEG 内的图形，用户可以用 load()函数来读取图像文件。

17.2 Pygame 的使用

本节主要讲解用 Pygame 开发游戏的逻辑、鼠标事件的处理、键盘事件的处理、字体的使用和声音的播放等基础知识，最后以一个“移动的坦克”例子来体现这些基础知识的应用。

17.2.1 Pygame 开发游戏的主要流程

Pygame 开发游戏的基础是创建游戏窗口，核心是处理事件、更新游戏状态和在屏幕上绘图。游戏状态可理解为程序中所有变量值的列表。在有些游戏中，游戏状态包括存放人物健康和位置的变量、物体或图形位置的变化，这些值可以在屏幕上显示。

物体或图形位置的变化只有通过在屏幕上绘图才能看出来。



可以简单地抽象出 Pygame 开发游戏的主要流程，如图 17-1 所示。

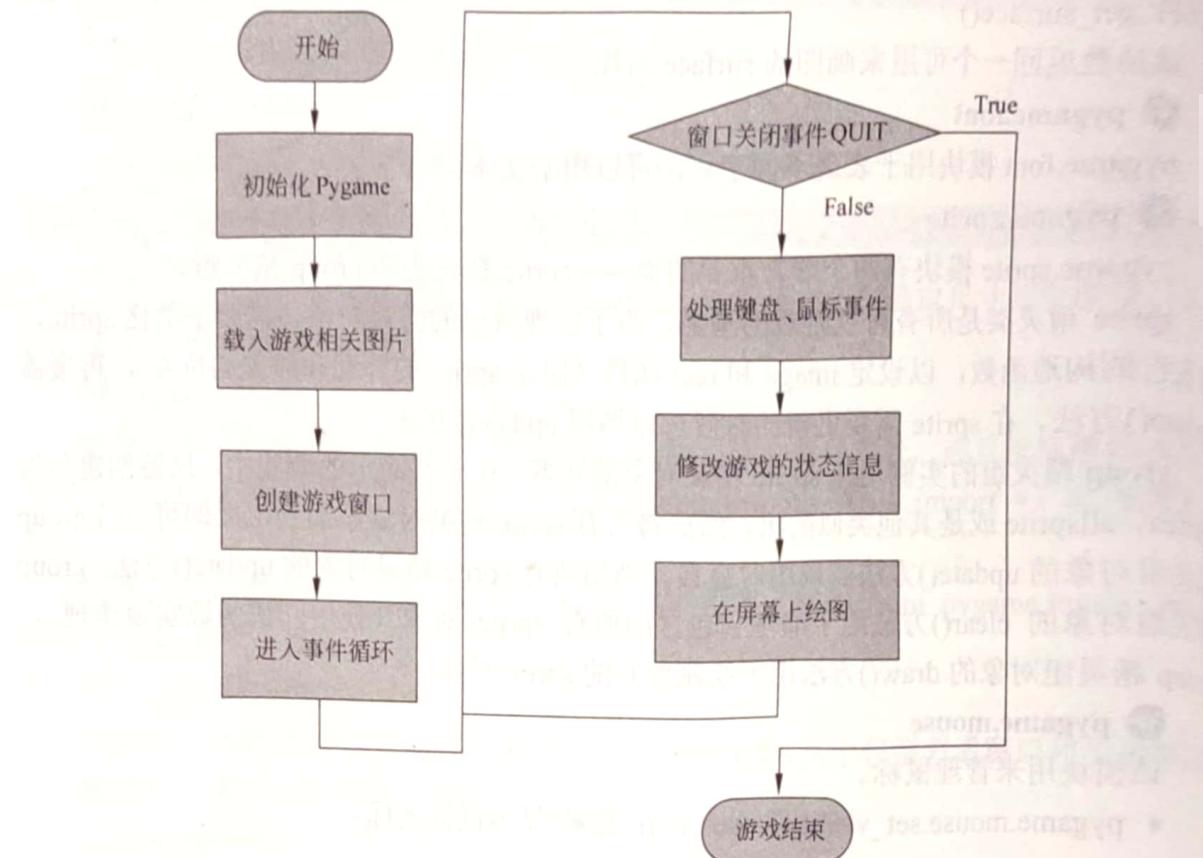


图 17-1 Pygame 开发游戏的主要流程

下面举一个具体例子来说明。

例 17-1 使用 Pygame 开发一个显示“Hello World!”标题的游戏窗口。

```
import pygame # 导入 pygame 模块
from pygame.locals import *
import sys

def hello_world():
    pygame.init() # 任何 Pygame 程序均需要执行此语句进行模块的初始化
    # 设置窗口的模式, (680,480) 表示窗口像素
    # 此函数返回一个 surface 对象, 本程序不使用它, 故没保存到对象变量中
    pygame.display.set_mode((680,480))
    pygame.display.set_caption('Hello World!') # 设置窗口标题

    # 无限循环, 直到接收到窗口关闭事件
    while True:
        # 处理事件
        for event in pygame.event.get():
            if event.type==QUIT: # 接收到窗口关闭事件
                pygame.quit() # 退出
                sys.exit()
        # 将 surface 对象绘制在屏幕上
        pygame.display.update()
```



```
if __name__=="__main__":
    hello_world()
```

程序运行后仅见到黑色的游戏窗口，标题是“Hello World！”，如图 17-2 所示。

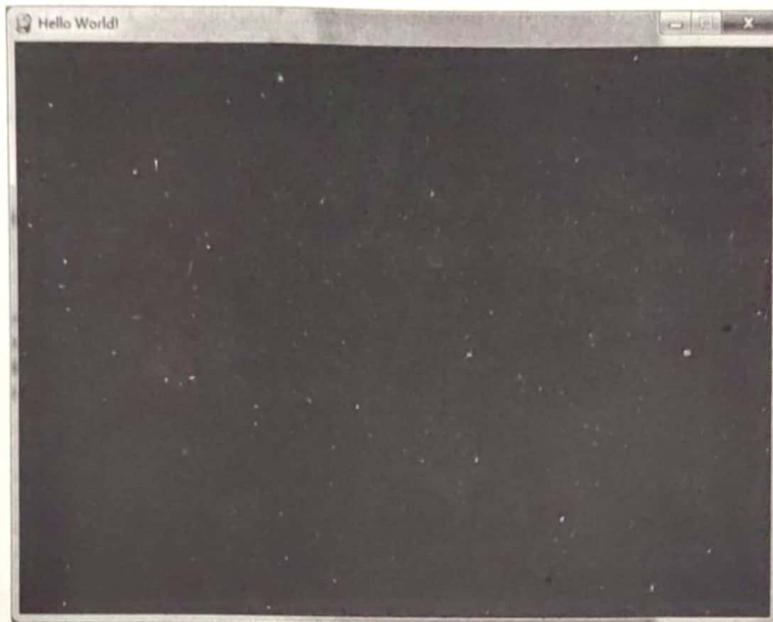


图 17-2 用 Pygame 开发的游戏窗口

在导入 Pygame 模块后，任何 Pygame 游戏程序均需要执行 `pygame.init()` 语句进行模块的初始化，它必须在进入游戏的无限循环之前被调用。这个函数会自动初始化其他所有模块（例如 `pygame.font` 和 `pygame.image`），通过它载入驱动和硬件请求，这样游戏程序才可以使用计算机上的所有设备，比较费时间。如果只使用少量模块，应该分别初始化这些模块以节省时间，例如 `pygame.sound.init()` 仅仅初始化声音模块。

该代码中有个无限循环，每个 Pygame 程序都需要它，在无限循环中可以做以下操作。

- (1) 处理事件：例如鼠标、键盘、关闭窗口等事件。
- (2) 更新游戏状态：例如坦克的位置变化、数量变化等。
- (3) 在屏幕上绘图：例如绘制新的敌方坦克等。

不断重复上面的 3 个步骤，从而完成游戏逻辑。

在本例代码中仅仅处理关闭窗口事件，也就是玩家关闭窗口时 `pygame.quit()` 退出游戏。

17.2.2 Pygame 的图像/图形绘制

① Pygame 的图像绘制

Pygame 支持多种存储图像的方式（也就是图片格式），例如 JPEG、PNG 等，具体支持 JPEG（一般扩展名为.jpg 或者.jpeg，数码相机、网上的图片基本上都是这种格式，这是一种有损压缩方式，尽管对图片的质量有些损坏，但对于减小文件尺寸非常棒，其优点很多，只是不支持透明）、PNG（支持透明，无损压缩）、GIF（网上使用的很多，支持透明和动画，但只能有 256 种颜色，在软件和游戏中的使用很少）以及 BMP、PCX、TGA、TIF 等格式。



Pygame 使用 surface 对象来加载绘制的图像。对于 Pygame，加载图片使用 `pygame.image.load()`，给它一个文件名然后就返回一个 surface 对象。尽管读入的图像格式有多种操作。事实上，游戏屏幕只是一个 surface，`pygame.display.set_mode()` 返回了一个 surface 对象。

对于任何一个 surface 对象，可以用 `get_width()`、`get_height()` 和 `get_size()` 函数来获取它的尺寸，`get_rect()` 用来获取它的区域形状。

例 17-2 使用 Pygame 开发一个显示坦克自由移动的游戏窗口。

```

import pygame
from pygame.locals import *
import sys

def play_tank():
    pygame.init()
    window_size=(width, height)=(600, 400)          # 窗口大小
    speed=[1, 1]           # 坦克运行偏移量，即[水平, 垂直]，值越大，移动越快
    color_black=(255, 255, 255)          # 窗口背景色 RGB 值(白色)
    screen=pygame.display.set_mode(window_size)      # 设置窗口模式
    pygame.display.set_caption('自由移动的坦克')      # 设置窗口标题
    tank_image=pygame.image.load('tankU.bmp')         # 加载坦克图片，返回一个 surface 对象

    tank_rect=tank_image.get_rect()                  # 获取坦克图片的区域形状
    while True:                                     # 无限循环
        for event in pygame.event.get():
            if event.type==pygame.QUIT:               # 退出事件处理
                pygame.quit()
                sys.exit()

        # 使坦克移动，速度由 speed 变量控制
        tank_rect=tank_rect.move(speed)
        # 当坦克运动出窗口时重新设置偏移量
        if(tank_rect.left < 0) or (tank_rect.right > width):   # 水平方向
            speed[0]=-speed[0]                                # 水平方向反向
        if(tank_rect.top < 0) or (tank_rect.bottom > height): # 垂直方向
            speed[1]=-speed[1]                                # 垂直方向反向
        screen.fill(color_black)                            # 填充窗口背景
        screen.blit(tank_image, tank_rect) # 在窗口指定区域 tank_rect 上绘制坦克
        pygame.display.update()                           # 更新窗口显示内容

    if __name__=='__main__':
        play_tank()

```

程序运行后，可以看到白色背景的游戏窗口，标题是“自由移动的坦克”，如图 17-3 所示。



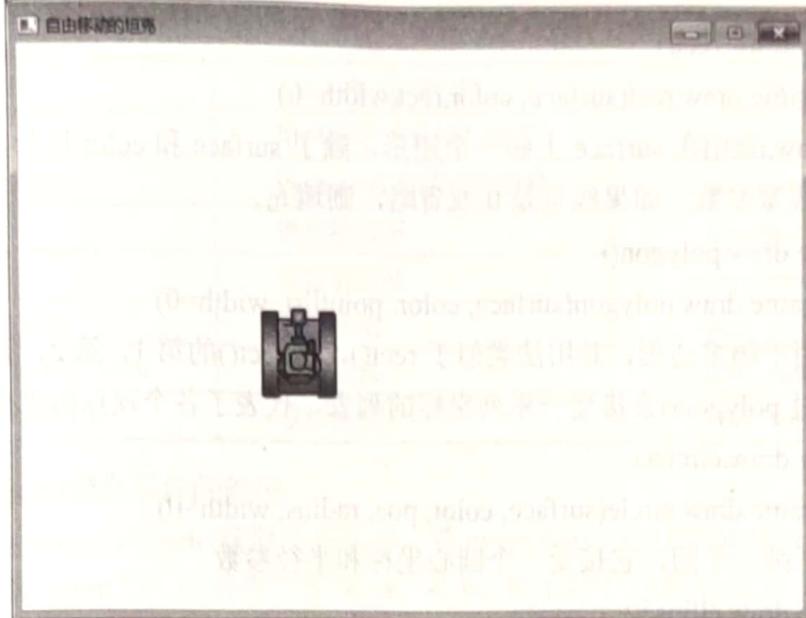


图 17-3 自由移动的坦克游戏窗口

在该游戏中通过修改坦克图像（surface 对象）区域的 left 属性（可以认为是 x 坐标）、surface 对象的 top 属性（可以认为是 y 坐标）改变坦克位置，从而显示出坦克自由移动的效果。在窗口（窗口也是 surface 对象）使用的 blit() 函数上绘制坦克图像，最后注意需要更新窗口显示内容。

设置 fpsClock 变量的值即可控制游戏速度，语法如下：

```
fpsClock=pygame.time.Clock()
```

在无限循环中写入 fpsClock.tick(50)，可以按指定帧频 50 更新游戏画面（即每秒钟刷新 50 次屏幕）。

② Pygame 的图形绘制

在屏幕上绘制各种图形时使用 pygame.draw 模块中的一些函数，事实上 Pygame 可以不加载任何图片，而使用图形来制作一个游戏。

pygame.draw 中函数的第一个参数总是一个 surface，然后是颜色，接着是一系列的坐标等。对于计算机中的坐标，(0,0) 代表左上角，水平向右为 X 轴的正方向，垂直向下为 Y 轴的正方向。该函数的返回值是一个 rect 对象，包含了绘制的区域，这样就可以很方便地更新那个部分了。pygame.draw 中的函数如表 17-3 所示。

表 17-3 pygame.draw 中的函数

函 数	作 用	函 数	作 用
rect()	绘制矩形	line()	绘制线
polygon()	绘制多边形（3 个及 3 个以上的边）	lines()	绘制一系列的线
circle()	绘制圆	aaline()	绘制一根平滑的线
ellipse()	绘制椭圆	aalines()	绘制一系列平滑的线
arc()	绘制圆弧		

下面详细说明 pygame.draw 中各个函数的使用。



1) pygame.draw.rect()

格式: pygame.draw.rect(surface, color, rect, width=0)

pygame.draw.rect()在 surface 上画一个矩形, 除了 surface 和 color 以外, rect 接受矩形的坐标和线宽参数, 如果线宽是 0 或省略, 则填充。

2) pygame.draw.polygon()

格式: pygame.draw.polygon(surface, color, pointlist, width=0)

polygon()用于画多边形, 其用法类似于 rect(), 与 rect()的第 1、第 2、第 4 个参数都是相同的, 只不过 polygon()会接受一系列坐标的列表, 代表了各个顶点的坐标。

3) pygame.draw.circle()

格式: pygame.draw.circle(surface, color, pos, radius, width=0)

circle()用于画一个圆, 它接受一个圆心坐标和半径参数。

4) pygame.draw.ellipse()

格式: pygame.draw.ellipse(surface, color, rect, width=0)

用户可以把 ellipse 想象成一个被压扁的圆, 事实上, 它可以被一个矩形装起来。pygame.draw.ellipse()的第 3 个参数就是这个椭圆的外接矩形。

5) pygame.draw.arc()

格式: pygame.draw.arc(surface, color, rect, start_angle, stop_angle, width=1)

arc 是椭圆的一部分, 所以它的参数比椭圆多一点。但是它是不封闭的, 因此没有 fill()方法。start_angle 和 stop_angle 为开始和结束的角度。

6) pygame.draw.line()

格式: pygame.draw.line(surface, color, start_pos, end_pos, width=1)

line()用于画一条线段, start_pos、end_pos 是线段起点和终点坐标。

7) pygame.draw.lines()

格式: pygame.draw.lines(surface, color, closed, pointlist, width=1)

closed 是一个布尔变量, 指明是否需要多画一条线来使这些线条闭合 (这样就和 polygon 一样了), pointlist 是一个顶点坐标的数组。

17.2.3 Pygame 的键盘和鼠标事件的处理

所谓事件, 就是程序上发生的事。例如用户按键盘上的某个键或是单击、移动鼠标。对于这些事件, 游戏程序需要做出反应。在例 17-2 中, 程序会一直运行下去, 直到用户关闭窗口而产生了一个 QUIT 事件, Pygame 会接收用户的各种操作 (例如按键盘上的键、移动鼠标等) 产生事件。事件随时可能发生, 而且量可能会很大, Pygame 的做法是把一系列事件存放到一个队列里逐个处理。

在例 17-2 中使用了 pygame.event.get()来处理所有事件, 如果使用 pygame.event.wait(), Pygame 会等到发生一个事件时才继续下去, 一般在游戏中不太实用, 因为游戏往往是需要动态运作的。Pygame 中的常用事件如表 17-4 所示。



表 17-4 Pygame 中的常用事件

事 件	产 生 途 径	参 数
QUIT	用户按下“关闭”按钮	none
ACTIVEEVENT	Pygame 被激活或者隐藏	gain、state
KEYDOWN	键盘被按下	unicode、key、mod
KEYUP	键盘被放开	key、mod
MOUSEMOTION	鼠标移动	pos、rel、buttons
MOUSEBUTTONDOWN	鼠标被按下	pos、button
MOUSEBUTTONUP	鼠标被放开	pos、button

① Pygame 的键盘事件的处理

通常用 `pygame.event.get()` 获取所有事件，若 `event.type == KEYDOWN`，这时是键盘事件，再判断按键 `event.key` 的种类（即 `K_a`、`K_b`、`K_LEFT` 这种形式）。用户也可以使用 `pygame.key.get_pressed()` 获取所有被按下的键值，它会返回一个元组。这个元组的索引就是键值，对应的就是键是否被按下。

```
pressed_keys=pygame.key.get_pressed()
if pressed_keys[K_SPACE]:
    #空格键被按下
    fire()#发射子弹
```

在 `key` 模块下有很多函数。

- `key.get_focused()`: 返回当前的 Pygame 窗口是否被激活。
- `key.get_pressed()`: 获得所有按下的键值。
- `key.get_mods()`: 按下的组合键（Alt、Ctrl、Shift）。
- `key.set_mods()`: 模拟按下组合键的效果（`KMOD_ALT`、`KMOD_CTRL`、`KMOD_SHIFT`）。

例 17-3 使用 Pygame 开发一个由用户控制坦克移动的游戏。在例 17-2 的基础上增加通过方向键控制坦克运动的功能，并为游戏增加背景图片。程序的运行效果如图 17-4 所示。

```
import os
import sys
import pygame
from pygame.locals import *
def control_tank(event):          #控制坦克运动函数
    speed=[x,y]=[0, 0]             #相对坐标
    speed_offset=1                 #速度
    #当方向键被按下时进行位置计算
    if event.type==pygame.KEYDOWN:
        if event.key==pygame.K_LEFT:
            speed[0]-=speed_offset
        if event.key==pygame.K_RIGHT:
            speed[0]=speed_offset
```



```

        if event.key==pygame.K_UP:
            speed[1]-=speed_offset
        if event.key==pygame.K_DOWN:
            speed[1]=speed_offset
        #当方向键被释放时相对偏移为 0, 即不移动
        if event.type in (pygame.KEYUP, pygame.K_LEFT, pygame.K_RIGHT, pygame.K_DOWN):
            speed=[0, 0]
        return speed

def play_tank():
    pygame.init()
    window_size=Rect(0, 0, 600, 400)
    speed=[1, 1]           #坦克运行偏移量, 即[水平, 垂直], 值越大, 移动越快
    color_black=(255, 255, 255)   #窗口背景色 RGB 值(白色)
    screen=pygame.display.set_mode(window_size.size)   #设置窗口模式
    pygame.display.set_caption('用户方向键控制坦克移动') #设置窗口标题
    tank_image=pygame.image.load('tankU.bmp')          #加载坦克图片
    #加载窗口背景图片
    back_image=pygame.image.load('back_image.jpg')
    tank_rect=tank_image.get_rect()           #获取坦克图片的区域形状
    while True:
        #退出事件处理
        for event in pygame.event.get():      #pygame.event.get() 获取事件序列
            if event.type==pygame.QUIT:
                pygame.quit()
                sys.exit()
        #使坦克移动, 速度由 speed 变量控制
        cur_speed=control_tank(event)
        #rect 的 clamp() 方法使移动范围限制在窗口内
        tank_rect=tank_rect.move(cur_speed).clamp(window_size)
        screen.blit(back_image,(0, 0))          #设置窗口背景图片
        screen.blit(tank_image,tank_rect)        #在窗口上绘制坦克
        pygame.display.update()                 #更新窗口显示内容
    if __name__=='__main__':
        play_tank()

```

当用户按下方向键, 计算出相对位置 cur_speed 后, 使用 tank_rect.move(cur_speed) 函数向指定方向移动坦克。当释放方向键时坦克停止移动。

② Pygame 的鼠标事件的处理

pygame.mouse 的函数如下。

- pygame.mouse.get_pressed(): 返回按键的按下情况, 返回的是一元组, 分别为左键、中键、右键, 如果被按下则为 True。



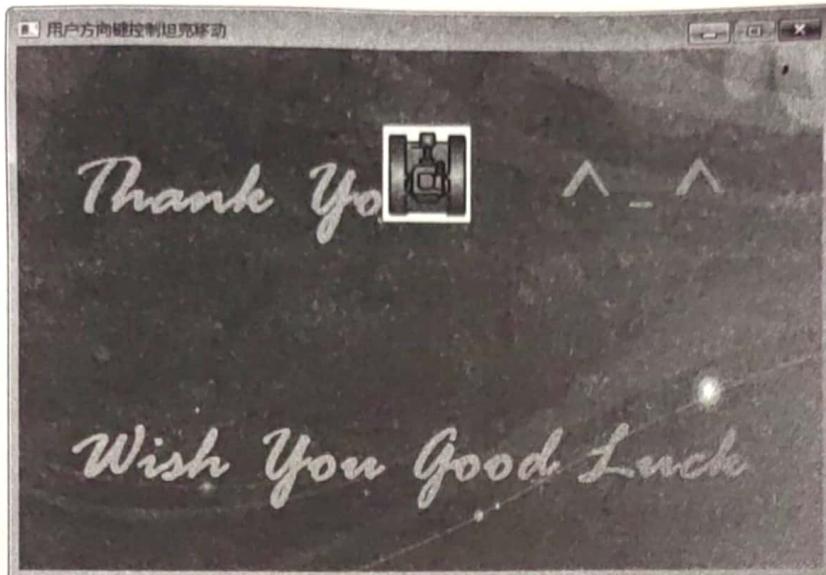


图 17-4 用方向键控制坦克运动的游戏窗口

- `pygame.mouse.get_rel()`: 返回相对偏移量, 即(x 方向偏移量, y 方向偏移量)的一元组。
 - `pygame.mouse.get_pos()`: 返回当前鼠标位置(x, y)。
- 例如“`x, y = pygame.mouse.get_pos()`”用于获取鼠标位置。
- `pygame.mouse.set_pos()`: 设置鼠标位置。
 - `pygame.mouse.set_visible()`: 设置鼠标光标是否可见。
 - `pygame.mouse.get_focused()`: 如果鼠标在 Pygame 窗口内有效, 返回 True。
 - `pygame.mouse.set_cursor()`: 设置鼠标的默认光标样式。
 - `pygame.mouse.get_cursor()`: 返回鼠标的光标样式。

例 17-4 演示鼠标事件处理的程序, 运行效果如图 17-5 所示。

```
import pygame
from pygame.locals import *
from sys import exit
from random import *
from math import pi
pygame.init()
screen=pygame.display.set_mode((640, 480), 0, 32)
points=[]
while True:
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
            exit()
        if event.type==KEYDOWN:
            #按任意键可以清屏, 并把点回复到原始状态
            points=[]
            screen.fill((255,255,255)) #用白色填充窗口背景
        if event.type==MOUSEBUTTONDOWN:
            #鼠标按下
```



```

screen.fill((255,255,255))
#画随机矩形
rc=(255,0,0) #红色
rp=(randint(0,639), randint(0,479))
rs=(639-randint(rp[0],639),479-randint(rp[1],479))
pygame.draw.rect(screen, rc, Rect(rp,rs))
#画随机圆形
rc=(0,255,0) #绿色
rp=(randint(0,639), randint(0,479))
rr=randint(1,200)
pygame.draw.circle(screen, rc, rp, rr)
#获得当前鼠标单击位置
x,y=pygame.mouse.get_pos()
points.append((x,y))
#根据单击位置画弧线
angle=(x/639.)*pi*2.
pygame.draw.arc(screen, (0,0,0), (0,0,639,479), 0, angle, 3)
#根据单击位置画椭圆
pygame.draw.ellipse(screen, (0,255,0), (0,0,x,y))
#从左上和右下画两根线连接到单击位置
pygame.draw.line(screen, (0,0,255), (0,0), (x,y))
pygame.draw.line(screen, (255,0,0), (640,480), (x,y))
#画单击轨迹图
if len(points)>1:
    pygame.draw.lines(screen, (155,155,0), False, points, 2)
#和轨迹图基本一样，只不过是闭合的，因为会覆盖，所以这里注释了
#if len(points)>=3:
#    pygame.draw.polygon(screen, (0,155,155), points, 2)
#把每个点画明显一点
for p in points:
    pygame.draw.circle(screen, (155,155,155), p, 3)
pygame.display.update()

```

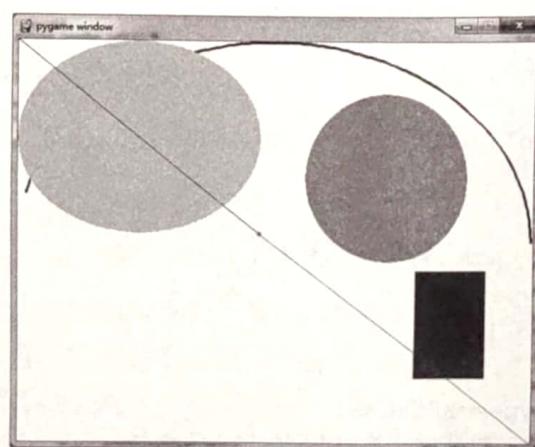


图 17-5 演示鼠标事件处理的程序的运行效果

运行这个程序，在窗口上单击鼠标就会有图形出来，按任意键可以重新开始。



17.2.4 Pygame 的字体使用

Pygame 可以直接调用系统字体，也可以调用 TTF 字体。为了使用字体，首先应该创建一个 Font 对象，对于系统自带的字体应该这样调用：

```
font1=pygame.font.SysFont('arial', 16)
```

第 1 个参数是字体名，第 2 个参数是字号。在正常情况下系统里都会有 arial 字体，如果没有会使用默认字体，默认字体和用户使用的系统有关。

用户可以使用 pygame.font.get_fonts() 来获取当前系统所有的可用字体：

```
>>> pygame.font.get_fonts()
['gisha', 'fzshuti', 'simsunnsimsun', 'estrangeloedessa',
'symboltigerexpert', 'juiceitc', 'onyx', 'tiger', 'webdings',
'franklingothicmediumcond', 'edwardianscriptitc']
```

另外还有一种调用方法是使用自己的 TTF 字体：

```
my_font=pygame.font.Font("my_font.ttf", 16)
```

这个方法的好处是可以把字体文件和游戏一起打包分发，避免玩家计算机上没有这个字体而无法显示的问题。一旦有了 Font 对象，就可以用 render() 方法来设置文字内容，然后通过 blit() 方法写到屏幕上：

```
text=font1.render("坦克大战", True, (0, 0, 0), (255, 255, 255))
```

render() 方法的第一个参数是写入的文字内容；第 2 个参数是布尔值，说明是否开启抗锯齿；第 3 个参数是字体本身的颜色；第 4 个参数是背景的颜色。如果不想要背景色，也就是让背景透明，可以不加第 4 个参数。

例如自己定义一个文字处理函数 show_text()，其中参数 surface_handle 为 surface 句柄，pos 为文字显示位置，color 为文字颜色，font_bold 为是否加粗，font_size 为字体大小，font_italic 为是否斜体。

```
def show_text(surface_handle, pos, text, color, font_bold=False, font_
size=13, font_italic=False):
    #cur_font=pygame.font.SysFont("宋体", font_size)      #获取系统字体
    cur_font=pygame.font.Font('simfang.ttf', 30)        #获取字体，并设置文字大小
    cur_font.set_bold(font_bold)                         #设置是否加粗
    cur_font.set_italic(font_italic)                     #设置是否斜体
    text_fmt=cur_font.render(text, 1, color)            #设置文字内容
    surface_handle.blit(text_fmt, pos)                  #绘制文字
```

在更新窗口内容 pygame.display.update() 之前加入：

```
text_pos=u "坦克大战"
show_text(screen, (20, 220), text_pos, (255, 0, 0), True)
text_pos=u"坦克位置:(%d,%d)" % (tank_rect.left, tank_rect.top)
```



```
show_text(screen, (20, 420), text_pos, (0, 255, 255), True)
```

此时会在屏幕上的(20, 220)处显示红色的“坦克大战”文字，并且在(20, 420)处显示现在坦克所处位置的坐标，移动坦克，位置坐标文字同时会改变。

17.2.5 Pygame 的声音播放

① Sound 对象

在初始化声音设备后就可以读取一个音乐文件到一个 Sound 对象中。`pygame.mixer.Sound()`接受一个文件名，也可以是一个文件对象，不过这个文件必须是 WAV 或者 OGG 文件。

```
hello_sound=pygame.mixer.Sound("hello.ogg")      #建立 Sound 对象
hello_sound.play()                            #声音播放一次
```

一旦这个 Sound 对象出来了，就可以使用 `play()` 来播放它。`play(loop, maxtime)` 可以接受两个参数，`loop` 是重复的次数（取 1 是两次，注意是重复的次数而不是播放的次数），`-1` 意味着无限循环；`maxtime` 是指多少毫秒后结束。

若不使用任何参数调用，意味着把这个声音播放一次。一旦 `play()` 方法调用成功，就会返回一个 Channel 对象，否则返回一个 `None`。

② music 对象

在 Pygame 中还提供了 `pygame.mixer.music` 类来控制背景音乐的播放。`pygame.mixer.music` 用来播放 MP3 和 OGG 音乐文件，不过 MP3 并不是所有的系统都支持（Linux 默认就不支持 MP3 播放）。用户可以用 `pygame.mixer.music.load()` 加载一个文件，然后使用 `pygame.mixer.music.play()` 播放，不放的时候就用 `stop()` 方法停止，当然也有类似录影机上的 `pause()` 和 `unpause()` 方法。

```
#加载背景音乐
pygame.mixer.music.load("hello.mp3")
pygame.mixer.music.set_volume(music_volume/100.0)
#循环播放，从音乐的第 30 秒开始
pygame.mixer.music.play(-1, 30.0)
```

在游戏退出事件中加入停止音乐播放的代码：

```
#停止音乐播放
pygame.mixer.music.stop()
```

`music` 对象提供了丰富的函数方法，下面分别介绍。

1) `pygame.mixer.music.load()`

功能：加载音乐文件。

格式：`pygame.mixer.music.load(filename)`

2) `pygame.mixer.music.play()`

功能：播放音乐。



格式：pygame.mixer.music.play(loops=0,start=0.0)

其中，loops 表示循环次数，如果设置为-1，表示不停地循环播放，如果 loops 为 5，则播放 $5+1=6$ 次；start 参数表示从音乐文件的哪一秒开始播放，设置为 0 表示从开始完整播放。

3) pygame.mixer.music.rewind()

功能：重新播放。

格式：pygame.mixer.music.rewind()

4) pygame.mixer.music.stop()

功能：停止播放。

格式：pygame.mixer.music.stop()

5) pygame.mixer.music.pause()

功能：暂停播放。

格式：pygame.mixer.music.pause()

用户可通过 pygame.mixer.music.unpause() 恢复播放。

6) pygame.mixer.music.set_volume()

功能：设置音量。

格式：pygame.mixer.music.set_volume(value)

其中，value 的取值为 0.0~1.0。

7) pygame.mixer.music.get_pos()

功能：获取当前播放了多长时间。

格式：pygame.mixer.music.get_pos(): return time

17.2.6 Pygame 的精灵使用

pygame.sprite.Sprite 是 Pygame 中用来实现精灵的一个类，在使用时并不需要对它实例化，只需要继承它，然后按需写出自己的类，因此非常简单、实用。

① 精灵

精灵可以被认为是一个个小图片（帧）序列（例如人物行走），它可以在屏幕上移动，并且可以与其他图形对象交互。精灵图像可以是使用 Pygame 绘制形状函数绘制的形状，也可以是图像文件。图 17-6 所示为由 16 帧图片组成的人物行走序列。

② Sprite 类的成员

pygame.sprite.Sprite 用来实现精灵类，Sprite 的数据成员和函数方法主要如下。

1) self.image

其负责显示什么图形，例如 self.image=pygame.Surface([x,y]) 说明该精灵是一个 $x \times y$ 大小的矩形，self.image=pygame.image.load(filename) 说明该精灵显示 filename 这个图片文件。

self.image.fill([color]) 负责对 self.image 着色，例如：

```
self.image=pygame.Surface([x,y])
self.image.fill([255,0,0]) #对 x×y 大小的矩形填充红色
```





图 17-6 精灵图片序列

2) self.rect

其负责在哪里显示。一般来说，先用 `self.rect=self.image.get_rect()` 获取 `image` 矩形大小，然后给 `self.rect` 设定显示的位置，一般用 `self.rect.topleft` 确定左上角的显示位置，当然也可以用 `topright`、`bottomright`、`bottomleft` 分别确定其他几个角的位置。

另外，`self.rect.top`、`self.rect.bottom`、`self.rect.left`、`self.rect.right` 分别表示上、下、左、右。

3) self.update()

其负责使精灵行为生效。

4) Sprite.add()

添加精灵到 `groups` 中。

5) Sprite.remove()

从精灵组 `groups` 中删除。

6) Sprite.kill()

从精灵组 `groups` 中删除全部精灵。

7) Sprite.alive()

判断某个精灵是否属于精灵组 `groups`。

③ 建立精灵

所有精灵在建立时都是从 `pygame.sprite.Sprite` 中继承的，建立精灵要设计自己的精灵类。

例 17-5 建立 Tank 精灵。

```
import pygame,sys
pygame.init()
class Tank(pygame.sprite.Sprite):
    def __init__(self,filename,initial_position):
        pygame.sprite.Sprite.__init__(self)
        self.image=pygame.image.load(filename)
        self.rect=self.image.get_rect()          #获取 self.image 的大小
        self.rect.topleft=initial_position       #确定左上角的显示位置
```



```
self.rect.bottomright=initial_position #右下角的显示位置是[150,100]

screen=pygame.display.set_mode([640,480])
screen.fill([255,255,255])
fi='tankU.jpg'
b=Tank(fi,[150,100])
while True:
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            sys.exit()
    screen.blit(b.image,b.rect)
    pygame.display.update()
```

例 17-6 使用图 17-6 所示的精灵图片序列建立动画效果的人物行走精灵。

在游戏动画中，人物行走是基本动画，在精灵中不断切换人物行走图片，从而达到动画的效果。

```
import pygame
from pygame.locals import *
class MySprite(pygame.sprite.Sprite):
    def __init__(self, target):
        pygame.sprite.Sprite.__init__(self)
        self.target_surface=target
        self.image=None
        self.master_image=None
        self.rect=None
        self.topleft=0,0
        self.frame=0
        self.old_frame=-1
        self.frame_width=1
        self.frame_height=1
        self.first_frame=0      #第 1 帧序号
        self.last_frame=0       #最后一帧序号
        self.columns=1          #列数
        self.last_time=0
```

在加载一个精灵图片序列的时候需要告知程序一帧的大小（传入帧的宽度和高度以及文件名和列数）。

```
def load(self, filename, width, height, columns):
    self.master_image=pygame.image.load(filename).convert_alpha()
    self.frame_width=width
    self.frame_height=height
    self.rect=0,0,width,height
    self.columns=columns
    rect=self.master_image.get_rect()
```



```
self.last_frame=(rect.width//width)*(rect.height//height)-1
```

一个循环动画通常就是这样工作的：从第 1 帧开始不断地加载直到最后一帧，然后再返回第 1 帧，并不断重复这个操作。

但是如果只是这样做，程序会一股脑地将动画播放完，这里想让它根据时间间隔一张一张地播放，因此加入定时的代码，将帧速率 ticks 传递给 Sprite 的 update() 函数，这样就可以轻松地让动画按照帧速率来播放。

```
def update(self, current_time, rate=60):
    if current_time>self.last_time + rate: #如果时间超过上次时间+60毫秒
        self.frame+=1 #帧号加1，意味着显示下一帧图像
    if self.frame>self.last_frame: #帧号超过最后一帧
        self.frame=self.first_frame #回到第1帧
        self.last_time=current_time
    if self.frame!=self.old_frame:
        #首先需要计算单个帧左上角的x、y位置值
        frame_x=(self.frame % self.columns) * self.frame_width
        frame_y=(self.frame // self.columns) * self.frame_height
        #然后将计算好的x,y值传递给位置属性
        rect=(frame_x, frame_y, self.frame_width, self.frame_height)
        #要显示区域
        self.image=self.master_image.subsurface(rect)
        #截取要显示区域图像
        self.old_frame=self.frame
pygame.init()
screen=pygame.display.set_mode((800,600),0,32)
pygame.display.set_caption("精灵类测试")
font=pygame.font.Font(None, 18)
#启动一个定时器，然后调用 tick(num) 函数就可以让游戏以 num 帧来运行
framerate=pygame.time.Clock()
cat=MySprite(screen)
cat.load("sprite2.png", 92, 95, 4) #精灵图片，每帧 92×95 大小，共 4 列
group=pygame.sprite.Group()
group.add(cat)
while True:
    framerate.tick(10) #指定帧速率
    ticks=pygame.time.get_ticks() #获取运行时间
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            pygame.quit()
            exit()
        key=pygame.key.get_pressed()
        if key[pygame.K_ESCAPE]: #Esc 键
            exit()
```



```
screen.fill((0,0,100))
#cat.draw(screen)
cat.update(ticks)
screen.blit(cat.image,cat.rect)
#group.update(ticks)
#group.draw(screen)
pygame.display.update()
```

运行后可见一个人物行走动画，用户也可以使用精灵组的 update() 和 draw() 函数实现精灵动画。

```
group.update(ticks)
    #将帧速率 ticks 传递给 sprite 的 update() 函数，让动画按照帧速率来播放
group.draw(screen)
```

④ 建立精灵组

当程序中有大量实体的时候，操作这些实体将会是一件相当麻烦的事，那么有没有什么容器可以将这些精灵放在一起统一管理呢？答案就是使用精灵组。

Pygame 使用精灵组来管理精灵的绘制和更新，精灵组是一个简单的容器。

使用 pygame.sprite.Group() 函数可以创建一个精灵组：

```
group=pygame.sprite.Group()
group.add(sprite_one)
```

精灵组也有 update() 和 draw() 函数：

```
group.update()
group.draw()
```

Pygame 还提供了精灵与精灵之间的冲突检测、精灵与组之间的碰撞检测，这些碰撞检测技术在 17.4 节的飞机大战游戏中要使用。

⑤ 精灵与精灵之间的碰撞检测

1) 两个精灵之间的矩形检测

在只有两个精灵的时候可以使用 pygame.sprite.collide_rect() 函数进行一对一的冲突检测。这个函数需要传递两个精灵，并且每个精灵都需要继承自 pygame.sprite.Sprite。

这里举个例子：

```
sprite_1=MySprite("sprite_1.png",200,200,1)
    #MySprite 是例 17-6 创建的精灵类
sprite_2=MySprite("sprite_2.png",50,50,1)
result=pygame.sprite.collide_rect(sprite_1,sprite_2)
if result:
    print("精灵碰撞上了")
```

2) 两个精灵之间的圆检测

矩形冲突检测并不适用于所有形状的精灵，因此在 Pygame 中还提供了圆形冲突检测。pygame.sprite.collide_circle() 函数是基于每个精灵的半径值来进行检测的，用户可以自己指



定精灵半径，或者让函数计算精灵半径。

```
result=pygame.sprite.collide_circle(sprite_1,sprite_2)
if result:
    print("精灵碰撞上了")
```

3) 两个精灵之间的像素遮罩检测

如果矩形检测和圆形检测都不能满足需求，Pygame 还为用户提供了一个更加精确的检测——`pygame.sprite.collide_mask()`。

这个函数接受两个精灵作为参数，返回值是一个 bool 变量。

```
if pygame.sprite.collide_mask(sprite_1,sprite_2):
    print("精灵碰撞上了")
```

4) 精灵和组之间的矩形冲突检测

在调用 `pygame.sprite.spritecollide(sprite,sprite_group,bool)` 函数的时候，一个组中的所有精灵都会逐个地对另外一个单个精灵进行冲突检测，发生冲突的精灵会作为一个列表返回。

这个函数的第一个参数是单个精灵，第 2 个参数是精灵组，第 3 个参数是一个 bool 值，最后这个参数起了很大的作用，当为 True 的时候会删除组中所有冲突的精灵，当为 False 的时候不会删除冲突的精灵。

```
list_collide=pygame.sprite.spritecollide(sprite,sprite_group,False);
```

另外，这个函数还有一个变体——`pygame.sprite.spritecollideany()`，这个函数在判断精灵组和单个精灵冲突的时候会返回一个 bool 值。

5) 精灵组之间的矩形冲突检测

利用 `pygame.sprite.groupcollide()` 函数可以检测两个组之间的冲突，它返回一个字典(键值对)。

以上学习了几种常用的冲突检测函数，下面在游戏实例中实际运用这些函数。

17.3 基于 Pygame 设计贪吃蛇游戏

贪吃蛇游戏通过玩家控制蛇移动，不断吃到食物（红色草莓）增长，直到蛇身碰到边界游戏结束。其运行效果如图 17-7 所示。

```
import pygame, sys, time, random
from pygame.locals import *
```

输入下面的两行来启用 Pygame，这样 Pygame 在该程序中就可以用了：

```
pygame.init()
fpsClock=pygame.time.Clock()
```

第 1 行告诉 Pygame 初始化，第 2 行创建一个名为 `fpsClock` 的变量，该变量用来控制游戏的速度。然后用下面的两行代码新建一个 Pygame 显示层（游戏元素画布）。



图 17-7 基于 Pygame 设计的贪吃蛇

```
playSurface=pygame.display.set_mode((640,480))
pygame.display.set_caption('Raspberry Snake')
```

接下来定义一些颜色，虽然这一步并不是必不可少的，但它们在程序中用到的颜色：

```
redColour=pygame.Color(255, 0, 0)
blackColour=pygame.Color(0, 0, 0)
whiteColour=pygame.Color(255, 255, 255)
greyColour=pygame.Color(150, 150, 150)
```

下面的几行代码初始化了程序中用到的一些变量。开始时这些变量为空，Python 将无法正常运行。

```
snakePosition=[100,100]
snakeSegments=[[100,100],[80,100],...]
raspberryPosition=[300,300]
raspberrySpawned=1
direction='right'
changeDirection=direction
```

此时可以看到变量 `snakePosition`、`snakeSegments` 和 `raspberryPosition` 分隔的列表。

用下面几行代码来定义 `gameOver()` 函数。

```
def gameOver():
    gameOverFont=pygame.font.Font('freesansbold.ttf', 40)
    gameOverSurf=gameOverFont.render('Game Over', True, whiteColour)
```



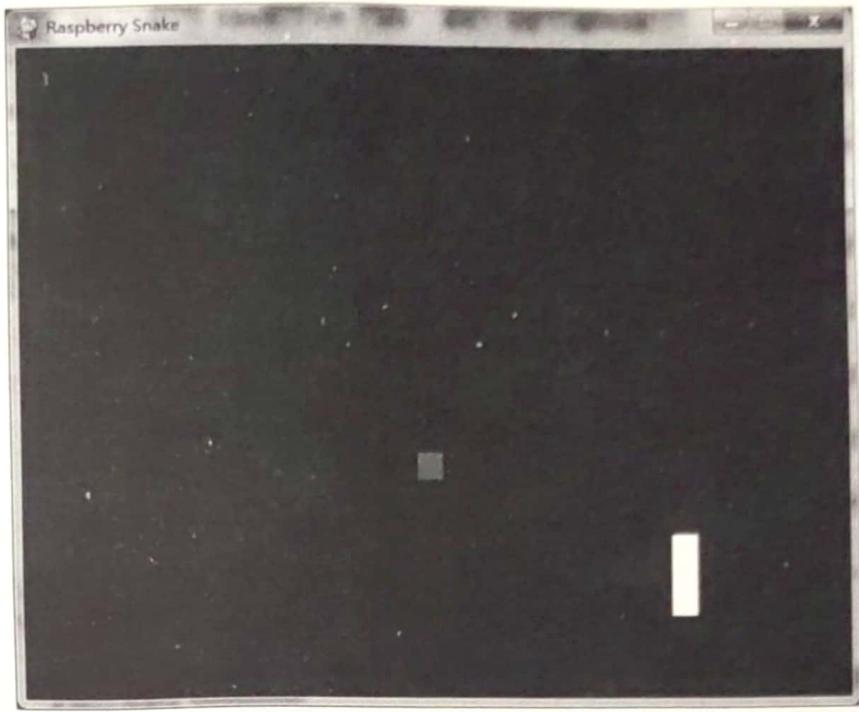


图 17-7 基于 Pygame 设计的贪吃蛇游戏的运行效果

```
playSurface=pygame.display.set_mode((640, 480))  
pygame.display.set_caption('Raspberry Snake')
```

接下来定义一些颜色，虽然这一步并不是必需的，但它会减少代码量。下面的代码定义了程序中用到的颜色：

```
redColour=pygame.Color(255, 0, 0)  
blackColour=pygame.Color(0, 0, 0)  
whiteColour=pygame.Color(255, 255, 255)  
greyColour=pygame.Color(150, 150, 150)
```

下面的几行代码初始化了程序中用到的一些变量，这是很重要的一步，因为如果游戏开始时这些变量为空，Python 将无法正常运行。

```
snakePosition=[100,100] #蛇头位置  
snakeSegments=[[100,100],[80,100],[60,100]] #蛇身序列  
raspberryPosition=[300,300] #草莓位置  
raspberrySpawned=1 #是否吃到草莓，1 为没有吃到，0 为吃到  
direction='right' #运动方向，初始向右  
changeDirection=direction
```

此时可以看到变量 snakePosition、snakeSegments 和 raspberryPosition 被设置为用逗号分隔的列表。

用下面几行代码来定义 gameOver() 函数：

```
def gameOver():  
    gameOverFont=pygame.font.Font('freesansbold.ttf', 72)  
    gameOverSurf=gameOverFont.render('Game Over', True, greyColour)
```



```

gameOverRect=gameOverSurf.get_rect()
gameOverRect.midtop=(320, 10)
playSurface.blit(gameOverSurf, gameOverRect)
pygame.display.flip()
time.sleep(5)
pygame.quit()
sys.exit()

```

gameOver()函数用了一些 Pygame 命令来完成一个简单的任务：用大号字体将 Game Over 打印在屏幕上，停留 5 秒钟，然后退出 Pygame 和 Python 程序。在游戏开始之前就定义了结束函数，这看起来有点奇怪，但是所有的函数都应该在被调用前定义。Python 是不会自己执行 gameOver() 函数的，直到用户调用该函数。

至此程序的开头部分已经完成，接下来进入主要部分。该程序运行在一个无限循环（一个永不退出的 while 循环）中，直到蛇撞到了墙或者自己才会结束游戏。首先用下面的代码开始主循环：

```
while True:
```

没有其他的比较条件，Python 会检测 True 是否为真。如果 True 一直为真，循环会一直进行，直到用户调用 gameOver() 函数告诉 Python 退出该循环。

```

for event in pygame.event.get():
    if event.type==QUIT:
        pygame.quit()
        sys.exit()
    elif event.type==KEYDOWN:
        elif event.type==KEYDOWN:
            if event.key==K_RIGHT or event.key==ord('d'):
                changeDirection='right'
            if event.key==K_LEFT or event.key==ord('a'):
                changeDirection='left'
            if event.key==K_UP or event.key==ord('w'):
                changeDirection='up'
            if event.key==K_DOWN or event.key==ord('s'):
                changeDirection='down'
            if event.key==K_ESCAPE:
                pygame.event.post(pygame.event.Event(QUIT))

```

for 循环用来检测按键等 Pygame 事件。

第 1 个检测 if event.type == QUIT 告诉 Python 如果 Pygame 发出了 QUIT 信息（当用户按下 Esc 键时），执行下面缩进的代码。之后的两行类似 gameOver() 函数，通知 Pygame 和 Python 程序结束并退出。

第 2 个检测以 elif 开头的行用来检测 Pygame 是否发出 KEYDOWN 事件，该事件在用户按下键盘时产生。

KEYDOWN 事件修改变量 changeDirection 的值，该变量用于控制蛇的运动方向。在



本例中提供了两种控制蛇的方法，即用鼠标或者键盘上的 W、D、A、S 键让蛇向上、右、下、左移动。程序开始时，蛇会按照 changeDirection 预设的值向右移动，直到用户按下键盘改变其方向。

在程序开始的初始化部分有一个叫 direction 的变量，这个变量协同 changeDirection 检测用户发出的命令是否有效。蛇不应该立即向后运动（如果发生该情况，蛇会死亡，同时游戏结束）。为了防止这样的情况发生，将用户发出的请求（保存在 changeDirection 里）和目前的方向（保存在 direction 里）进行比较，如果方向相反，忽略该命令，蛇会继续按原方向运动。这里用下面几行代码进行比较：

```
if changeDirection=='right' and not direction=='left':  
    direction=changeDirection  
if changeDirection=='left' and not direction=='right':  
    direction=changeDirection  
if changeDirection=='up' and not direction=='down':  
    direction=changeDirection  
if changeDirection=='down' and not direction=='up':  
    direction=changeDirection
```

这样就保证了用户输入的合法性，蛇（在屏幕上显示为一系列块）就能够按照用户的输入移动。每次转弯时，蛇都会向该方向移动一小节。每个小节为 20 像素，用户可以告诉 Pygame 在任何方向移动一小节。

```
if direction=='right':  
    snakePosition[0]+=20  
if direction=='left':  
    snakePosition[0]-=20  
if direction=='up':  
    snakePosition[1]-=20  
if direction=='down':  
    snakePosition[1]+=20
```

snakePosition 为蛇头的新位置，程序开始处的另一个列表变量 snakeSegments 却不是这样。该列表存储蛇身体的位置（头部后边），随着蛇吃掉草莓导致长度增加，列表会增加长度同时提高游戏难度。随着游戏的进行，避免蛇头撞到身体的难度变大。如果蛇头撞到身体，蛇会死亡，同时游戏结束。此时用下面的代码使蛇的身体增长：

```
snakeSegments.insert(0, list(snakePosition))
```

这里用 insert() 方法向 snakeSegments 列表（存有蛇当前的位置）中添加新项目。每当 Python 运行到这一行，它就会将蛇的身体增加一节，同时将这一节放在蛇的头部，在玩家看来蛇在增长。当然，用户只希望蛇吃到草莓时才增长，否则蛇会一直变长。输入下面的几行代码：

```
if snakePosition[0]==raspberryPosition[0]  
and snakePosition[1]==raspberryPosition[1]:  
    raspberrySpawned=0
```



```

else:
    snakeSegments.pop()

```

第1条if语句检查蛇头部的x和y坐标是否等于草莓（玩家的目标点）的坐标。如果等于，该草莓就会被蛇吃掉，同时raspberrySpawned变量置为0。else语句告诉Python如果草莓没有被吃掉，将snakeSegments列表中最早的一项pop出来。

pop语句简单、易用，它返回列表中末尾的项目并从列表中删除，使列表缩短一项。在snakeSegments列表里，它使Python删掉距离头部最远的一部分。在玩家看来，蛇整体在移动而不会增长。实际上，它在一端增加小节，在另一端删除小节。由于有else语句，pop语句只有在没吃到草莓时执行。如果吃到了草莓，列表中的最后一项不会被删掉，所以蛇会增加一小节。

现在，蛇就可以通过吃草莓让自己变长了。但是如果游戏中只有一个草莓会有些无聊，所以若蛇吃了一个草莓，用下面的代码增加一个新的草莓到游戏界面中：

```

if raspberrySpawned==0:
    x=random.randrange(1, 32)
    y=random.randrange(1, 24)
    raspberryPosition=[int(x*20), int(y*20)]
    raspberrySpawned=1

```

这部分代码通过判断变量raspberrySpawned是否为0来判断草莓是否被吃掉了，如果被吃掉，使用程序开始引入的random模块获取一个随机的位置。然后将这个位置和蛇的每个小节的长度（20像素宽，20像素高）相乘来确定它在游戏界面中的位置。随机地放置草莓是很重要的，防止用户预先知道下一个草莓出现的位置。最后将raspberrySpawned变量置1，以保证每个时刻界面上只有一个草莓。

现在有了让蛇移动和生长的代码，包括草莓被吃和新建的操作（在游戏中称为草莓重生），但是还没有在界面上画东西，输入下面的代码：

```

playSurface.fill(blackColour)
for position in snakeSegments:      #画蛇（一系列方块）
    pygame.draw.rect(playSurface, whiteColour, Rect(position[0],
        position[1], 20, 20))
pygame.draw.rect(playSurface, redColour, Rect(raspberryPosition[0],
    raspberryPosition[1], 20, 20))      #草莓
pygame.display.flip()

```

这些代码让Pygame填充背景色为黑色，蛇的头部和身体为白色，草莓为红色。最后一行的pygame.display.flip()让Pygame更新界面（如果没有这条语句，用户将看不到任何东西。每次在界面上画完对象时，记得使用pygame.display.flip()让用户看到更新）。

现在还没有涉及蛇死亡的代码。如果游戏中的角色永远死不了，玩家很快会感到无聊，所以用下面的代码设置一些让蛇死亡的场景：

```

if snakePosition[0]>620 or snakePosition[0]<0:
    gameOver()
if snakePosition[1]>460 or snakePosition[1]<0:

```



gameOver()

第1个if语句检查蛇是否已经走出了界面的上、下边界，第2个if语句检查蛇是否已经走出了左、右边界。这两种情况都是蛇的末日，触发前面定义的gameOver()函数，打印游戏结束信息并退出游戏。如果蛇头撞到了自己身体的任何部分，也会让蛇死亡，所以输入下面几行代码：

```
for snakeBody in snakeSegments[1:]:
    if snakePosition[0]==snakeBody[0] and
        snakePosition[1]==snakeBody[1]:
        gameOver()
```

这里的for语句遍历蛇的每一小节的位置（从列表的第2项开始到最后一项），同时和当前蛇头的位置比较。这里用snakeSegments[1:]来保证从列表的第2项开始遍历。列表的第1项为头部位置，如果从第1项开始比较，那么游戏一开始蛇就死亡了。

最后，只需要设置fpsClock变量的值即可控制游戏速度。

fpsClock.tick(20)

使用IDLE的Run Module选项或者在终端中输入python snake.py来运行程序。

贪吃蛇snake.py的完整源代码如下：

```
import pygame, sys, time, random
from pygame.locals import *

pygame.init()
fpsClock=pygame.time.Clock()
playSurface=pygame.display.set_mode((640, 480))
pygame.display.set_caption('Raspberry Snake')

# 定义一些颜色
redColour=pygame.Color(255, 0, 0)
blackColour=pygame.Color(0, 0, 0)
whiteColour=pygame.Color(255, 255, 255)
greyColour=pygame.Color(150, 150, 150)

# 初始化程序中用到的一些变量
snakePosition=[100,100]
snakeSegments=[[100,100], [80,100], [60,100]]
raspberryPosition=[300,300]      # 草莓位置
raspberrySpawned=1                # 是否吃到草莓，1为没有吃到，0为吃到
direction='right'                  # 运动方向
changeDirection=direction

def gameOver():
    gameOverFont=pygame.font.Font('simfang.ttf', 72)
    gameOverSurf=gameOverFont.render('Game Over', True, greyColour)
    gameOverRect=gameOverSurf.get_rect()
    gameOverRect.midtop=(320, 10)
    playSurface.blit(gameOverSurf, gameOverRect)
    pygame.display.flip()
```



```
time.sleep(5)
pygame.quit()
sys.exit()

while True:
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
            sys.exit()
        elif event.type==KEYDOWN:
            if event.key==K_RIGHT or event.key==ord('d'):
                changeDirection='right'
            if event.key==K_LEFT or event.key==ord('a'):
                changeDirection='left'
            if event.key==K_UP or event.key==ord('w'):
                changeDirection='up'
            if event.key==K_DOWN or event.key==ord('s'):
                changeDirection='down'
            if event.key==K_ESCAPE:
                pygame.event.post(pygame.event.Event(QUIT))
        if changeDirection=='right' and not direction=='left':
            direction=changeDirection
        if changeDirection=='left' and not direction=='right':
            direction=changeDirection
        if changeDirection=='up' and not direction=='down':
            direction=changeDirection
        if changeDirection=='down' and not direction=='up':
            direction=changeDirection
        if direction=='right':
            snakePosition[0]+=20
        if direction=='left':
            snakePosition[0]-=20
        if direction=='up':
            snakePosition[1]-=20
        if direction=='down':
            snakePosition[1]+=20
    #将蛇的身体增加一节，同时将这一节放在蛇的头部
    snakeSegments.insert(0,list(snakePosition))
    #检查蛇头部的x和y坐标是否等于草莓（玩家的目标点）的坐标
    if snakePosition[0]==raspberryPosition[0] and snakePosition[1]==raspberryPosition[1]:
        raspberrySpawned=0
    else:
```



扫描全能王 创建

```

        snakeSegments.pop()
    #在游戏界面中增加一个新的草莓
    if raspberrySpawned==0:
        x=random.randrange(1,32)
        y=random.randrange(1,24)
        raspberryPosition = [int(x*20),int(y*20)]
    raspberrySpawned=1
    playSurface.fill(blackColour)
    for position in snakeSegments:      #画蛇 (一系列方块)
        pygame.draw.rect(playSurface,whiteColour,Rect(position[0],
            position[1], 20, 20))

    #画草莓
    pygame.draw.rect(playSurface,redColour,Rect(raspberryPosition[0],
        raspberryPosition[1], 20, 20))
    pygame.display.flip()
    if snakePosition[0]>620 or snakePosition[0] < 0:
        gameOver()
    if snakePosition[1]>460 or snakePosition[1] < 0:
        gameOver()
    for snakeBody in snakeSegments[1:]:
        if snakePosition[0]==snakeBody[0] and snakePosition[1]==
            snakeBody[1]:
            gameOver()
    fpsClock.tick(10)

```

17.4 基于 Pygame 设计飞机大战游戏

相信玩过《雷电》的朋友都熟悉打飞机，这里将游戏做了简化。飞机的速度固定，子弹的速度固定，基本操作是通过键盘移动玩家飞机，敌机随机从屏幕上方出现并匀速落到下方，子弹从玩家飞机发出，碰到目标飞机会击毁，如果目标飞机碰到玩家飞机，则游戏结束并显示分数。飞机大战游戏的运行效果如图 17-8 所示。

17.4.1 游戏角色

本游戏中所需的角色包括玩家飞机、敌机及子弹。用户可以通过键盘移动玩家飞机在屏幕上的位置来打不同位置的敌机，因此设计玩家类 Player、敌机类 Enemy 和子弹类 Bullet 对应 3 种游戏角色。

对于玩家类 Player，需要的操作有射击和移动两种，移动又分为上、下、左、右 4 种情况。



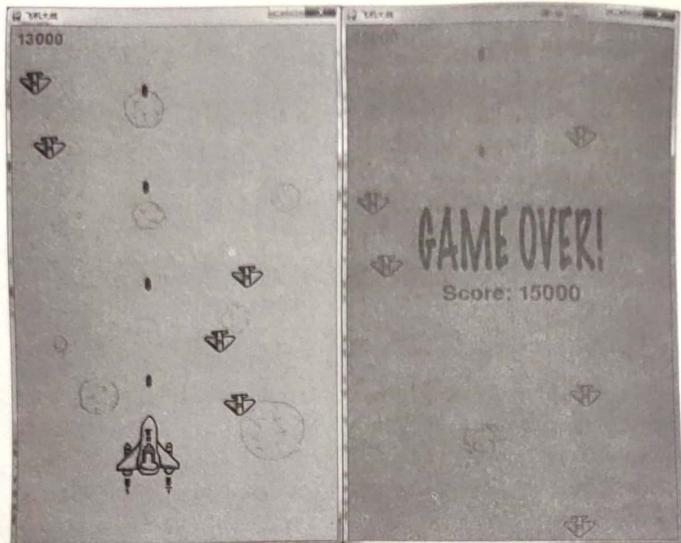


图 17-8 飞机大战游戏的运行效果

对于敌机类 Enemy，则比较简单，只需要移动即可，从屏幕上方出现并移动到屏幕下方。对于子弹类 Bullet，与飞机相同，仅需要以一定的速度移动即可。

玩家、子弹和敌机都可以写成一个类，继承 Pygame 的 Sprite 类，实现一些动画效果以及检测碰撞。

```

import pygame
from sys import exit
from pygame.locals import *
# from gameRole import *
import random

SCREEN_WIDTH=480
SCREEN_HEIGHT=800
TYPE_SMALL=1
TYPE_MIDDLE=2
TYPE_BIG=3

#子弹类
class Bullet(pygame.sprite.Sprite):          #继承 Sprite 精灵类
    def __init__(self, bullet_img, init_pos):
        pygame.sprite.Sprite.__init__(self)
        self.image=bullet_img
        self.rect=self.image.get_rect()
        self.rect.midbottom=init_pos

```

| 334

```

        self.speed=10
    def move(self):
        self.rect.top-=self.speed
#玩家类
class Player(pygame.sprite.Sprite):
    def __init__(self, plane_img, player_rect):
        pygame.sprite.Sprite.__init__(self)
        self.image=[]
        for i in range(len(player_rect)):
            self.image.append(plane_img.subsurface(player_rect[i]).convert_alpha())
        self.rect=player_rect[0]
        self.rect.topleft=init_pos
        self.speed=8
        self.bullets=pygame.sprite.Group()
        self.img_index=0
        self.is_hit=False
    def shoot(self, bullet_img):
        bullet=Bullet(bullet_img, self.rect.midtop)
        self.bullets.add(bullet)
    def moveUp(self):
        if self.rect.top <= 0:
            self.rect.top=0
        else:
            self.rect.top-=self.speed
    def moveDown(self):
        if self.rect.top>=SCREEN_HEIGHT:
            self.rect.top=SCREEN_HEIGHT
        else:
            self.rect.top+=self.speed
    def moveLeft(self):
        if self.rect.left<=0:
            self.rect.left=0
        else:
            self.rect.left-=self.speed
    def moveRight(self):
        if self.rect.left>=SCREEN_WIDTH:
            self.rect.left=SCREEN_WIDTH
        else:
            self.rect.left+=self.speed
#敌机类
class Enemy(pygame.sprite.Sprite):

```



扫描全能王 创建

```

        self.speed=10

    def move(self):
        self.rect.top-=self.speed

#玩家类

class Player(pygame.sprite.Sprite):          #继承 Sprite 精灵类
    def __init__(self, plane_img, player_rect, init_pos):
        pygame.sprite.Sprite.__init__(self)
        self.image=[]                         #用来存储玩家对象精灵图片的列表
        for i in range(len(player_rect)):
            self.image.append(plane_img.subsurface(player_rect[i]))
            .convert_alpha()
        self.rect=player_rect[0]               #初始化图片所在的矩形
        self.rect.topleft=init_pos           #初始化矩形的左上角坐标
        self.speed=8                         #初始化玩家速度，这里是一个确定的值
        self.bullets=pygame.sprite.Group()   #玩家飞机所发射的子弹的集合
        self.img_index=0                     #玩家精灵图片索引
        self.is_hit=False                   #玩家是否被击中

    def shoot(self, bullet_img):
        bullet=Bullet(bullet_img, self.rect.midtop)
        self.bullets.add(bullet)

    def moveUp(self):
        if self.rect.top <= 0:
            self.rect.top=0
        else:
            self.rect.top-=self.speed

    def moveDown(self):
        if self.rect.top>=SCREEN_HEIGHT-self.rect.height:
            self.rect.top=SCREEN_HEIGHT-self.rect.height
        else:
            self.rect.top+=self.speed

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

    def moveRight(self):
        if self.rect.left>=SCREEN_WIDTH-self.rect.width:
            self.rect.left=SCREEN_WIDTH-self.rect.width
        else:
            self.rect.left+=self.speed

#敌机类

class Enemy(pygame.sprite.Sprite):          #继承 Sprite 精灵类

```



```

def __init__(self, enemy_img, enemy_down_imgs, init_pos):
    pygame.sprite.Sprite.__init__(self)
    self.image=enemy_img
    self.rect=self.image.get_rect()
    self.rect.topleft=init_pos
    self.down_imgs=enemy_down_imgs
    self.speed=2
    self.down_index=0
def move(self):
    self.rect.top+=self.speed

```

以上设计了游戏中的3个角色。

17.4.2 游戏界面显示

在游戏画面中使用了一些飞机、子弹图像，这里使用 shoot.png 文件（如图 17-9 所示）存储所有飞机、子弹、爆炸等图像，在程序中需要分割出来显示。当然，可以用图像处理软件分解成一个个独立文件，这样处理后开发程序简单些。

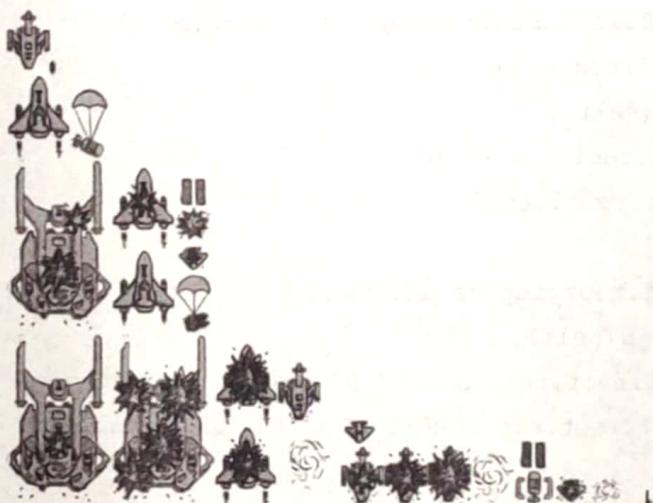


图 17-9 飞机大战游戏的图像文件 shoot.png

所有的飞机都在 shoot.png 图片中。在游戏中显示的元素（包括飞机、子弹等）在 Pygame 中都是一个 surface，这时可以利用 Pygame 提供的 subsurface()方法，首先载入一张大图，然后调用 subsurface()方法选取其中的一小部分生成一个新的 surface。

```

#载入飞机图片
plane_img=pygame.image.load('resources/image/shoot.png')
#选择飞机在大图中的位置，并生成 subsurface，然后初始化飞机开始的位置
player_rect=pygame.Rect(0, 99, 102, 126)
player1=plane_img.subsurface(player_rect)          #获取飞机图片
player_pos=[200, 600]
screen.blit(player1, player_pos)                  #绘制飞机

```



初始化游戏并根据设置好的大小生成游戏窗口；载入游戏音乐、背景图片 background.png、游戏结束画面 gameover.png 以及飞机和子弹图像 shoot.png；设置相关参数。最后定义存储敌人的飞机精灵组 enemies1 和用来渲染击毁精灵动画的爆炸飞机精灵组 enemies_down。

```
# 初始化游戏
pygame.init()
screen=pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption('飞机大战')

# 载入游戏音乐
bullet_sound=pygame.mixer.Sound('resources/sound/bullet.wav')
enemy1_down_sound=pygame.mixer.Sound('resources/sound/enemy1_down.wav')
game_over_sound=pygame.mixer.Sound('resources/sound/game_over.wav')
bullet_sound.set_volume(0.3)
enemy1_down_sound.set_volume(0.3)
game_over_sound.set_volume(0.3)
pygame.mixer.music.load('resources/sound/game_music.wav')
pygame.mixer.music.play(-1, 0.0)
pygame.mixer.music.set_volume(0.25)

background=pygame.image.load('resources/image/background.png')
.convert()                                     # 载入背景图
game_over=pygame.image.load('resources/image/gameover.png')      # 载入游戏结束图 gameover.png
filename='resources/image/shoot.png'
plane_img=pygame.image.load(filename)          # 载入飞机和子弹图 shoot.png

# 设置玩家相关参数
player_rect=[]
player_rect.append(pygame.Rect(0, 99, 102, 126))      # 玩家精灵图片区域
player_rect.append(pygame.Rect(165, 360, 102, 126))
player_rect.append(pygame.Rect(165, 234, 102, 126))    # 玩家爆炸精灵图片区域
player_rect.append(pygame.Rect(330, 624, 102, 126))
player_rect.append(pygame.Rect(330, 498, 102, 126))
player_rect.append(pygame.Rect(432, 624, 102, 126))
player_pos=[200, 600]
player=Player(plane_img, player_rect, player_pos)

# 定义子弹对象使用的 surface 相关参数
bullet_rect=pygame.Rect(1004, 987, 9, 21)
bullet_img=plane_img.subsurface(bullet_rect)

# 定义敌机对象使用的 surface 相关参数
enemy1_rect=pygame.Rect(534, 612, 57, 43)
```



扫描全能王 创建

```

enemy1_down_imgs=[]
enemy1_down_imgs.append(plane_img.subsurface(pygame.Rect(267, 347, 57, 43)))
enemy1_down_imgs.append(plane_img.subsurface(pygame.Rect(873, 697, 57, 43)))
enemy1_down_imgs.append(plane_img.subsurface(pygame.Rect(267, 296, 57, 43)))
enemy1_down_imgs.append(plane_img.subsurface(pygame.Rect(930, 697, 57, 43)))
enemies1=pygame.sprite.Group()          #存储敌人的飞机
enemies_down=pygame.sprite.Group()       #存储被击毁的飞机，用来渲染击毁精灵动画
shoot_frequency=0
enemy_frequency=0
player_down_index=16
score=0
clock=pygame.time.Clock()
running=True

```

17.4.3 游戏的逻辑实现

下面进入游戏主循环，在主循环中进行了以下操作：

① 处理键盘输入的事件，增加游戏操作交互

处理键盘输入的事件指上、下、左、右按键操作，增加游戏操作交互指玩家飞机的上、下、左、右移动。

```

key_pressed=pygame.key.get_pressed()
#若玩家被击中，则无效
if not player.is_hit:
    if key_pressed[K_w] or key_pressed[K_UP]: #处理键盘事件（移动飞机的位置）
        player.moveUp()
    if key_pressed[K_s] or key_pressed[K_DOWN]:#处理键盘事件（移动飞机的位置）
        player.moveDown()
    if key_pressed[K_a] or key_pressed[K_LEFT]:#处理键盘事件（移动飞机的位置）
        player.moveLeft()
    if key_pressed[K_d] or key_pressed[K_RIGHT]: #处理键盘事件（移动飞机的位置）
        player.moveRight()

```

② 处理子弹

这里控制发射子弹的频率并发射子弹，移动已发射过的子弹，若超出窗口范围则删除。

```

#控制发射子弹的频率并发射子弹
if not player.is_hit: # (1) 判断玩家飞机没有被击中
    if shoot_frequency%15==0:

```



```

        bullet_sound.play()
        player.shoot(bullet_img)
        shoot_frequency+=1
        if shoot_frequency>=15:
            shoot_frequency=0
    #移动已发射过的子弹，若超出窗口范围则删除
    for bullet in player.bullets:
        bullet.move()                      #(2)以固定速度移动子弹
        if bullet.rect.bottom<0:           #(3)子弹移动出屏幕后删除子弹
            player.bullets.remove(bullet)   #删除子弹

```

③ 敌机处理

敌机需要在界面上方随机产生，并以一定的速度向下移动，详细步骤如下：

- (1) 生成敌机，需要控制生成频率。
- (2) 移动敌机。
- (3) 敌机与玩家飞机碰撞效果的处理。
- (4) 移出屏幕后删除敌机。
- (5) 敌机被子弹击中效果的处理。

④ 得分显示

在游戏界面的固定位置显示消灭了多少目标敌机。

```

score_font=pygame.font.Font(None, 36)
score_text=score_font.render(str(score), True, (128, 128, 128))text_rect
=score_text.get_rect()
text_rect.topleft=[10, 10]
screen.blit(score_text, text_rect)

```

游戏主循环的完整代码如下：

```

while running:
    clock.tick(60)                      #控制游戏最大帧率为 60
    #控制发射子弹的频率并发射子弹
    if not player.is_hit:
        if shoot_frequency % 15==0:
            bullet_sound.play()
            player.shoot(bullet_img)
            shoot_frequency+=1
            if shoot_frequency>=15:
                shoot_frequency=0
    #移动子弹，若超出窗口范围则删除
    for bullet in player.bullets:
        bullet.move()
        if bullet.rect.bottom<0:
            player.bullets.remove(bullet)

    #生成敌机

```



```

-- enemy_frequency % 50==0:                      # (1)生成敌机，需要控制生成频率
    enemy1_pos=[random.randint(0, SCREEN_WIDTH - enemy1_rect.width),
    enemy1=Enemy(enemy1_img, enemy1_down_imgs, enemy1_pos)
    enemies1.add(enemy1)
enemy_frequency+=1
if enemy_frequency>=100:
    enemy_frequency=0

#移动敌机，若超出窗口范围则删除
for enemy in enemies1:
    enemy.move()                                # (2)移动敌机
#判断玩家是否被击中
    if pygame.sprite.collide_circle(enemy, player):
        # (3)敌机与玩家飞机碰撞效果的处理
        enemies_down.add(enemy)
        enemies1.remove(enemy)
        player.is_hit=True
        game_over_sound.play()
        break
    if enemy.rect.top > SCREEN_HEIGHT: # (4)移出屏幕后删除飞机
        enemies1.remove(enemy)

# (5)敌机被子弹击中效果的处理
#将被击中的敌机对象添加到击毁敌机 Group 中，用来渲染击毁动画
enemies1_down = pygame.sprite.groupcollide(enemies1, player.bullets, 1, 1)
for enemy_down in enemies1_down:
    enemies_down.add(enemy_down)

#绘制背景
screen.fill(0)
screen.blit(background, (0, 0))

#绘制玩家飞机
if not player.is_hit:
    screen.blit(player.image[player.img_index], player.rect)
    #更换图片索引使飞机有动画效果
    player.img_index=shoot_frequency           // 8
else:
    player.img_index=player_down_index         // 8
    screen.blit(player.image[player.img_index], player.rect)
    player_down_index+=1
    if player_down_index>47:
        running=False

#绘制击毁动画

```



扫描全能王 创建

```

if enemy_down.down_index > 7:
    enemy1_down_sound.play()
if enemy_down.down_index > 7:
    enemies_down.remove(enemy_down)
    score += 1000
    continue
screen.blit(enemy_down.down_imgs[enemy_down.down_index // 2],
            enemy_down.rect)
enemy_down.down_index += 1

#绘制子弹和敌机
player.bullets.draw(screen)
enemies1.draw(screen)

#绘制得分
score_font = pygame.font.Font(None, 36)
score_text = score_font.render(str(score), True, (128, 128, 128))
text_rect = score_text.get_rect()
text_rect.topleft = [10, 10]
screen.blit(score_text, text_rect)

#更新屏幕
pygame.display.update()

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        exit()

#监听键盘事件
key_pressed = pygame.key.get_pressed()

#若玩家被击中，则无效
if not player.is_hit:
    if key_pressed[K_w] or key_pressed[K_UP]:
        player.moveUp()
    if key_pressed[K_s] or key_pressed[K_DOWN]:
        player.moveDown()
    if key_pressed[K_a] or key_pressed[K_LEFT]:
        player.moveLeft()
    if key_pressed[K_d] or key_pressed[K_RIGHT]:
        player.moveRight()

font = pygame.font.Font(None, 48)
text = font.render('Score: ' + str(score), True, (255, 0, 0))
text_rect = text.get_rect()
text_rect.centerx = screen.get_rect().centerx
text_rect.centery = screen.get_rect().centery + 24
screen.blit(game_over, (0, 0))
screen.blit(text, text_rect)

while 1:
    for event in pygame.event.get():

```



```
if event.type==pygame.QUIT:  
    pygame.quit()  
    exit()  
pygame.display.update()
```

目前基本实现了玩家移动并发射子弹、随机生成敌机、击中敌机并爆炸、玩家飞机击毁、背景音乐及音效、游戏结束并显示分数这几项功能，已经是一个简单、可玩的游戏。整个游戏的实现不到 300 行代码，可以看出 Python 代码是多么简洁和高效。

