第 5 章 绘图及图像处理

5.1 绘 制 图 形


视频录像

5.1.1 用画布组件绘图

画布是图形用户界面 tkinter 的组件，是一个矩形区域，用于绘制图形或作为容器放置其他组件。

1. 创建画布对象

创建画布对象的基本语法形式如下：

```
w = Canvas(master, option=value, … )
```

其中：

- master：代表父窗口。
- options：为属性参数，其意义如表 5.1 所示。

表 5.1 画布的常用参数

Option 参数	说明
bg	背景颜色
height	画布的高
width	画布的宽

2. 图形的绘制方法

Canvas 对象包含了大量的绘图方法，表 5.2 列出了常用的绘图方法。

表 5.2 Canvas 对象常用的绘图方法

方法	说明
create_line(x1, y1, x2, y2)	绘制一条从(x1,y1)到(x2,y2)的直线
create_rectangle(x1, y1, x2, y2)	绘制一个左上角为(x1,y1)，右下角为(x2,y2)的矩形
create_polygon(x1, y1, x2, y2，x3, y3, x4, y4，x5, y5, x6, y6)	绘制一个顶点为(x1,y1)，(x2,y2),…,的多边形
create_oval(x1, y1, x2, y2, fill='color')	绘制一个左上角为(x1,y1)，右下角为(x2,y2)的外接矩形包围的圆，fill 为填充颜色
create_arc(x1, y1, x2, y2, start=s0, extent=s)	绘制在左上角为(x1,y1)，右下角为(x2,y2)的外接矩形所包围的一段圆弧，圆弧角度为 s，从 s0 开始
create_image(w, h, anchor=NE, image=filename)	在 w 宽 h 高的矩形区域内，显示文件名为 filename 的图像
move(obj, x, y)	移动组件 obj。x 为水平方向变化量，y 为垂直方向变化量

【例 5-1】 绘制几何图形示例。

程序代码如下：

```
'''
   窗体中的画布示例：
    绘制小球和扇形
'''
import tkinter
import tkinter.messagebox
win = tkinter.Tk()
win.title('画布示例')          # 定义窗体标题
win.geometry('400×200')       # 定义窗体的大小400×200像素
can = tkinter.Canvas(win, height=200, width=400)          # 定义画布
id = can.create_line(15,15,190,15)                        # 画一条直线
io1 = can.create_oval(50, 50, 100, 100, fill='blue')      # 画一蓝色圆
io2 = can.create_oval(59, 59, 68, 68, fill='white')       # 画一白色小圆
coord = 15, 120, 210, 220
arc = can.create_arc(coord, extent=150, fill="green")     # 画一个扇形
can.pack()
win.mainloop()
```

程序运行结果如图 5.1 所示。

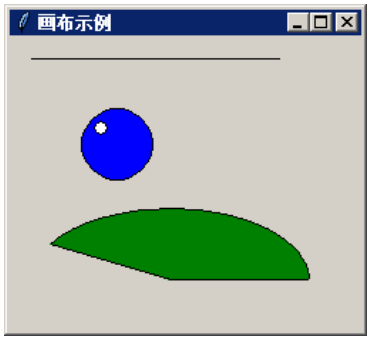


图 5.1　绘制几何图形

【例 5-2】 绘制笑脸。

程序代码如下：

```
'''
   窗体中的画布示例：
    绘制笑脸
'''
import tkinter
import tkinter.messagebox
win = tkinter.Tk()
win.title('画布示例')
```

```
win.geometry('250×250')

can = tkinter.Canvas(win, height=250, width=250)        # 定义画布
io1 = can.create_oval(35,30,210,210, fill='yellow')     # 画一黄色圆
io2 = can.create_oval(70,70,180,180, fill='black')
io3 = can.create_oval(65,70,185,170, outline='yellow', fill='yellow')
io4 = can.create_oval(80,100,110,130, fill='black')
io5 = can.create_oval(150,100,180,130, fill='black')

can.pack()
win.mainloop()
```

程序运行结果如图 5.2 所示。

图 5.2　绘制笑脸

【例 5-3】 显示图像示例。

程序代码如下：

```
import tkinter.messagebox
from tkinter import *

win = tkinter.Tk()
win.title('绘图示例')          # 定义窗体标题
win.geometry('200×200')       # 定义窗体的大小200×200像素
can = tkinter.Canvas(win, height=200, width=200)   # 定义画布
filename = PhotoImage(file = "test.gif")
image = can.create_image(150, 10, anchor=NE, image=filename)
can.pack()
win.mainloop()
```

程序运行结果如图 5.3 所示。

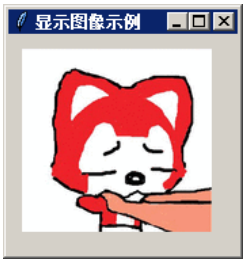


图 5.3　显示图像

5.1.2　用 turtle 模块绘图

turtle 模块是 Python 中的一个简单绘图工具，用它绘图非常方便。使用 turtle 绘制图形时，它会显示出一个箭头（又称为“海龟”），该箭头在一个横轴为 x、纵轴为 y 的坐标系中，从原点(0, 0)位置开始，按照所绘图形的轨迹绘制图形。

下面介绍 turtle 模块的一些基础知识。

1. turtle 模块的画布 Canvas

画布 Canvas 是 turtle 用于绘图区域，可以设置它的大小和初始位置。

（1）设置画布大小

```
turtle.screensize(canvwidth=None,canvheight=None,bg=None)
```

其中，参数 canvwidth 为画布的宽(单位像素)；canvheight 为高；bg 为背景颜色。

例如：

```
turtle.screensize(800, 600, "green")
```

当 screensize()函数无参数时，则返回一个默认为宽 400，高 300 像素的画布

即

```
turtle.screensize() # 返回默认大小(400,300)
```

（2）设置画布初始位置

```
turtle.setup(width=0.5,height=0.75,startx=None,starty=None)
```

其中参数：

width,height：当宽和高为整数时，表示像素；为小数时，表示占据屏幕的比例。

(startx,starty)：表示矩形窗口左上角顶点的坐标位置，如果为空，则位于屏幕中心。

例如：

```
turtle.setup(width=800,height=800,startx=100,starty=100)
turtle.setup(width=0.6,height=0.6)    # 画布位于屏幕中心
```

2. turtle 模块的基本指令

操纵 turtle 模块的“海龟”绘图有许多命令，这些命令分为两种：一种为画笔控制命

令；另一种为运动命令。

（1）画笔控制命令

turtle 模块的画笔控制命令如表 5.3 所示。

表 5.3　画笔控制命令

画笔控制命令	说明
turtle.down()	画笔落下，移动时绘制图形
turtle.up()	画笔抬起，移动时不绘制图形
turtle.pensize(width)	设置画笔的宽度，即绘制图形线条的宽度
turtle.color(colorstring)	设置画笔的颜色，即绘制图形的颜色
turtle.fillcolor(colorstring)	设置绘制图形的填充颜色
turtle.fill(true)	绘制填充图形
turtle.fill(false)	绘制线条图形
turtle. circle(radius, extent)	绘制一个圆形，其中 radius 为半径；extent 为角度。例如，若 extent 为 180，则画一个半圆；如画一个圆形，则不必写第二个参数

（2）运行命令

turtle 模块的运行命令如表 5.4 所示。

表 5.4　运行命令

运动命令	说明
turtle.forward(d)	向前移动距离，d 代表距离
turtle.backward(d)	向后移动距离，d 代表距离
turtle.right(degree)	向右转动多少角度
turtle.left(degree)	向左转动多少角度
turtle.goto(x,y)	将画笔移动到坐标为(x,y)的位置
turtle.stamp()	绘制当前图形
turtle.speed(speed)	画笔绘制的速度，取值范围为[0,10]的整数，值越大速度越快
turtle.clear()	清空 turtle 画的笔迹
turtle.reset()	清空窗口，重置 turtle 的状态为起始状态
turtle.undo()	撤销上一个 turtle 动作
turtle.isvisible()	设置当前 turtle 是否可见
turtle.stamp()	复制当前图形
turtle.write('str')	写字符串'str'
turtle.write(str[, font=("font-name", font_size,"font_type")])	写文本，str 为文本内容，font 是字体的参数，里面分别为字体名称、大小和类型；font 为可选项，font 的参数也是可选项

【例 5-4】 绘制一个边长为 60 的三角形图形。

程序代码如下：

```
import turtle
import time
a=60

for n in range(1, 4):
```

```
        turtle.forward(a)
        turtle.left(120)
        turtle.speed(1)
time.sleep(5)
```

程序运行结果如图 5.4 所示。

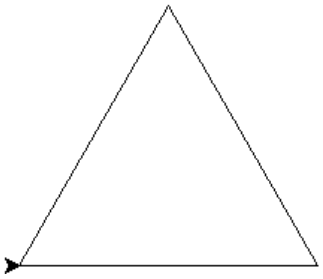


图 5.4　绘制三角形图形

5.2　数字图像处理基础

视频录像

5.2.1　Python 图像处理类库 PIL

图像处理类库（Python Imaging Library，PIL）提供了通用的图像处理功能，以及大量实用的基本图像操作，如图像缩放、裁剪、旋转、颜色转换等。由于 PIL 仅支持 Python 2.7 以前版本，Python 3.x 的兼容版本称为 Pillow。

1. 安装 Pillow 模块

在命令行窗口中使用 pip 安装 Pillow 模块，其命令为：

```
pip  install  pillow
```

安装过程如图 5.5 所示。

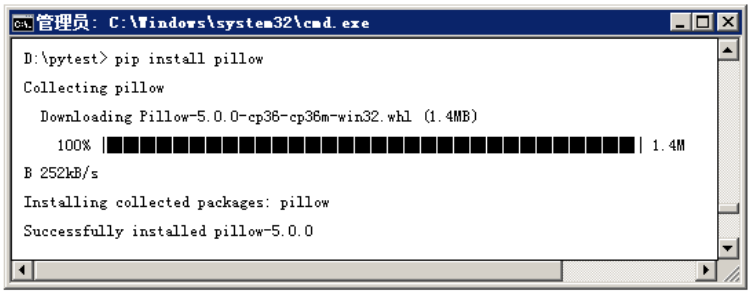


图 5.5　安装 Pillow 模块

2. Pillow 模块的方法

Pillow 模块提供了大量用于图像处理的方法，通过创建的图像对象可以调用这些图像

处理方法。Pillow 模块图像处理的常用方法如表 5.5 所示。

表 5.5　Pillow 模块图像处理的常用方法

方法	说明
Image.open("图像文件名")	打开图像文件，返回图像对象
show()	显示图像
save("文件名")	保存图像文件
resize(宽高元组)	图像缩放
thumbnail()	创建图像的缩略图
rotate()	旋转图像
transpose(Image.FLIP_LEFT_RIGHT)	图像水平翻转
transpose(Image.FLIP_TOP_BOTTOM)	图像垂直翻转
crop(矩形区域元组)	裁剪图像
paste(裁剪图像对象，矩形区域)	粘贴图像
ImageGrab.grab(矩形区域元组)	屏幕截图，若区域为空，则表示全屏幕截图
filter(ImageFilter.EDGE_ENHANCE)	图像增强
filter(ImageFilter.BLUR)	图像模糊
filter(ImageFilter.FIND_EDGES)	图像边缘提取
point(lambda　i:i*r)	图像点运算。r>1，图像变亮；r<1,图像变暗
format	查看图像格式的属性值
size	查看图像大小的属性值，格式为（宽度，高度）
getpixel(坐标元组)	读取像素的属性值，参数为(x,y)坐标元组，返回值为红、绿、蓝三色分量的值
putpixel((元组 1), (元组 2))	元组 2 的值改变目标像素元组 1 的颜色值
split()	将彩色图像分离为红、绿、蓝三个分量通道。 例如：r, g, b = im.split()
Image.merge(im.mode, (r,g,b))	将红、绿、蓝三个分量通道合并成一个彩色图像
enhance(n)	对比度增强为原来的 n 倍(n 为实数)。例如： img = ImageEnhance.Contrast(img) img = im.enhance(1.5) #对比度增强为原图的 1.5 倍

5.2.2　图像处理技术

利用 PIL 中的函数，可以从大多数图像格式的文件中读取数据，然后写入最常见的图像格式文件中。PIL 中最重要的模块为 Image。要读取一幅图像，可以使用下列语句：

```
from PIL import Image
img = Image.open("img1.gif")
```

上述代码的返回值 pil_im 是一个 PIL 图像对象。可以对这个 PIL 图像对象进行各种处理。下面介绍几个典型的图像处理的应用示例。

1. 图像的打开、旋转和显示

【例 5-5】 打开和显示图像示例。

程序代码如下：

```
import tkinter
```

```
from PIL import Image, ImageTk

win = tkinter.Tk()
win.title('图像显示')
win.geometry('300×300')                  # 定义窗体的大小300×300像素

can = tkinter.Canvas(win,                # 创建画布组件
    bg='white',                          # 指定画布组件的背景色
    width=300,                           # 指定画布组件的宽度
    height=300)                          # 指定画布组件的高度

image = Image.open("dukou.jpg")          # 打开图像文件
img = ImageTk.PhotoImage(image)          # 获取图像像素
can.create_image(160,120,image=img) # 将图像添加到画布组件中
can.pack()                               # 将画布组件添加到主窗口

win.mainloop()
```

程序运行结果如图 5.6 所示。

图 5.6　打开和显示图像

2. 建立图像的缩略图

使用 PIL 可以方便地创建图像的缩略图。PIL 图像对象的 thumbnail(size)方法将图像转换成由元组参数设定大小的缩略图。

【例 5-6】 建立图像缩略图示例。

程序代码如下：

```
import tkinter
from tkinter import Label
from PIL import Image, ImageTk
import glob, os
```

```
win = tkinter.Tk()
win.title('建立图像缩略图')
win.geometry('200×200')                          # 定义窗体的大小400×200像素

def imgshow():
    size = 64, 64                                # 设置缩略图尺寸的元组参数
    for infile in glob.glob("dukou.jpg"):
        file, ext = os.path.splitext(infile)
        im = Image.open(infile)
        im.thumbnail(size)
        im.save(file + "(1).jpg", "JPEG")        # 保存缩略图为dukou(1).jpg
    photo = ImageTk.PhotoImage(file='dukou(1).jpg')
    label = Label(win, image=photo).pack()
    label.image = photo

tkinter.Button(win, text='建立图像缩略图 ',command=imgshow).pack()
win.mainloop()
```

运行程序，单击按钮后，将当前文件夹中名为 dukou.jpg 的图像文件生成 64×64 像素的缩略图，如图 5.7 所示。

图 5.7　生成图像缩略图

3. 增强图像处理

使用 PIL 模块可以方便地对图像进行各种处理。例如，应用 filter()方法的 ImageFilter.EDGE_ENHANCE 属性可以将图像的对比度增强。

【例 5-7】 增加图像的对比度示例。

程序代码如下：

```
import tkinter
from tkinter import Label
from PIL import Image, ImageTk, ImageEnhance, ImageFilter

win = tkinter.Tk()
win.title('增强图像')
win.geometry('400×200')                  # 定义窗体的大小400×200像素
```

```
photo = Image.open('dukou.jpg')
img1 = ImageTk.PhotoImage(photo)              # 获取图片像素
label_1 = Label(win, image=img1)              # 显示原图

def imgshow():
    img = photo.filter(ImageFilter.EDGE_ENHANCE)
    img2 = ImageTk.PhotoImage(img)            # 获取图片像素
    label_2 = Label(win, image=img2).grid(row=1, column=1)  # 显示增强后的图
    label_2.image = img2

button = tkinter.Button(win, text='增强图像处理 ',command=imgshow)

button.grid(row=0, column=0, columnspan=2)
label_1.grid(row=1, column=0)

win.mainloop()
```

程序运行结果如图 5.8 所示。

图 5.8　图像增强

5.3　案 例 精 选


视频录像

【例 5-8】 动画效果的签名。

程序代码如下：

```
import turtle

turtle.color('red','green')
turtle.pensize(5)
turtle.goto(0,0)
turtle.speed(10)
for i in range(15):
    turtle.forward(100)
```

```
    turtle.right(150)
    turtle.up()

turtle.goto(100,-120)
turtle.color('black')
turtle.write("Python爱好者",font="隶书 -36 bold")
turtle.up()

turtle.goto(135,-140)
turtle.color('black')
turtle.write("2018 年 1 月 1 日",font="隶书 -18" )
turtle.up()
turtle.goto(240,-160)
turtle.color('black')
turtle.write("." )
turtle.done()
```

程序运行结果如图 5.9 所示。

图 5.9　绘制有动画效果的签名

【例 5-9】 绘制一个指针式时钟。

程序代码如下：

```
import turtle
from datetime import *
# 抬起画笔，向前运动一段距离放下
def Skip(step):
  turtle.penup()
  turtle.forward(step)
  turtle.pendown()
def mkHand(name, length):
  # 注册Turtle形状，建立表针Turtle
  turtle.reset()
  Skip(-length * 0.1)
  # 开始记录多边形的顶点。当前的乌龟位置是多边形的第一个顶点
  turtle.begin_poly()
  turtle.forward(length * 1.1)
```

```
    # 停止记录多边形的顶点。当前的乌龟位置是多边形的最后一个顶点。将最后一个顶点与第一个
    # 顶点相连
    turtle.end_poly()
    # 返回最后记录的多边形
    handForm = turtle.get_poly()
    turtle.register_shape(name, handForm)
def Init():
    global secHand, minHand, hurHand, printer
    # 重置Turtle指向北
    turtle.mode("logo")
    # 建立三个表针Turtle并初始化
    mkHand("secHand", 135)
    mkHand("minHand", 125)
    mkHand("hurHand", 90)
    secHand = turtle.Turtle()
    secHand.shape("secHand")
    minHand = turtle.Turtle()
    minHand.shape("minHand")
    hurHand = turtle.Turtle()
    hurHand.shape("hurHand")
    for hand in secHand, minHand, hurHand:
        hand.shapesize(1, 1, 3)
        hand.speed(0)
    # 建立输出文字Turtle
    printer = turtle.Turtle()
    # 隐藏画笔的turtle形状
    printer.hideturtle()
    printer.penup()
def SetupClock(radius):
    # 建立表的外框
    turtle.reset()
    turtle.pensize(7)
    for i in range(60):
        Skip(radius)
        if i % 5 == 0:
            turtle.forward(20)
            Skip(-radius - 20)

            Skip(radius + 20)
            if i == 0:
                turtle.write(int(12), align="center", font=("Courier", 14, "bold"))
            elif i == 30:
                Skip(25)
                turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
```

```
        Skip(-25)
      elif (i == 25 or i == 35):
        Skip(20)
        turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
        Skip(-20)
      else:
        turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
        Skip(-radius - 20)
    else:
      turtle.dot(5)
      Skip(-radius)
    turtle.right(6)
def Week(t):
  week = ["星期一", "星期二", "星期三", \
      "星期四", "星期五", "星期六", "星期日"]
  return week[t.weekday()]
def Date(t):
  y = t.year
  m = t.month
  d = t.day
  return "%s %d%d" % (y, m, d)
def Tick():
  # 绘制表针的动态显示
  t = datetime.today()
  second = t.second + t.microsecond * 0.000001
  minute = t.minute + second/60.0
  hour = t.hour + minute/60.0
  secHand.setheading(6 * second)
  minHand.setheading(6 * minute)
  hurHand.setheading(30 * hour)
  turtle.tracer(False)
  printer.forward(65)
  printer.write(Week(t), align="center",
        font=("Courier", 14, "bold"))
  printer.back(130)
  printer.write(Date(t), align="center",
        font=("Courier", 14, "bold"))
  printer.home()
  turtle.tracer(True)
  # 100ms后继续调用tick
  turtle.ontimer(Tick, 100)
def main():
  # 打开/关闭龟动画，并为更新图纸设置延迟
```

```
    turtle.tracer(False)
    Init()
    SetupClock(160)
    turtle.tracer(True)
    Tick()
    turtle.mainloop()

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

程序运行结果如图 5.10 所示。

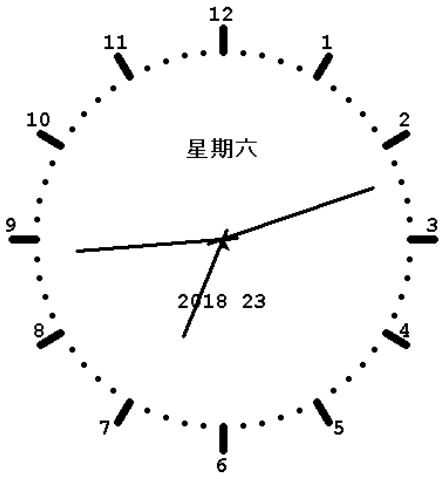


图 5.10　指针式时钟

画布 Canvas 类可以用于设计动画，使用 move(tags，dx，dy)方法实现移动图片或文字等组件。

canvas.update()为刷新界面，重新显示画布。

【例 5-10】 用方向键移动小矩形块。

程序代码如下：

```
import time
from tkinter import *

x = 50
y = 50

# （1）定义窗口
win = Tk()
win.title("移动小矩形块")
# （2）定义画布
```

```
canvas = Canvas(win, width = 400, height = 400)
canvas.pack() # 显示画布

# （3）定义矩形块
rect = canvas.create_rectangle(x, y, x+30, y+30, fill='red')
print(rect)

# （4）定义移动小矩形的函数
def moveRect(event):
    if event.keysym == 'Up':
        canvas.move(rect, 0, -3)
    elif event.keysym == 'Down':
        canvas.move(rect, 0, +3)
    elif event.keysym == 'Left':
        canvas.move(rect, -3, 0)
    elif event.keysym == 'Right':
        canvas.move(rect, 3, 0)
    win.update()          # 界面刷新
    time.sleep(0.05)      # 休眠

# （5）绑定方向键
canvas.bind_all('<KeyPress-Up>', moveRect)
canvas.bind_all('<KeyPress-Down>', moveRect)
canvas.bind_all('<KeyPress-Left>', moveRect)
canvas.bind_all('<KeyPress-Right>', moveRect)

win.mainloop()
```

Keysym == 键值（方向键）
move(组件，x 坐标增量，y 坐标增量)

绑定键盘事件

程序运行结果如图 5.11 所示。

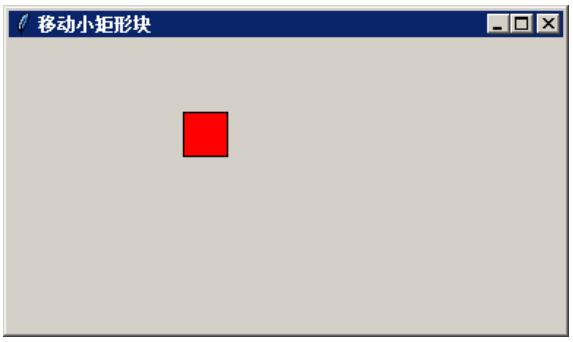


图 5.11　用方向键移动小矩形块

【例 5-11】 设计一个小球遇到窗体边缘或挡板则弹回来的动画程序。
程序代码如下：

```
from tkinter import *
```

```
import random
import time

class Ball:                          # 小球的类
    def __init__(self,canvas,paddle,color):
        self.canvas=canvas       # 传递画布值
        self.paddle=paddle       # 把挡板传递进来
        self.id=canvas.create_oval(10,10,35,35,fill=color)# 画椭圆并保存其ID
        self.canvas.move(self.id,245,100)
        start=[-3,-2,-1,1,2,3]
        random.shuffle(start)    # 随机化列表
        self.x=start[0]
        self.y=-3
        self.canvas_heigh=self.canvas.winfo_height()    # 获取窗口高度并保存
        self.canvas_width=self.canvas.winfo_width()
    def draw(self):
        self.canvas.move(self.id,self.x,self.y)
        # 返回相应ID代表的图形的当前坐标(左上角和右上角坐标)
        pos=self.canvas.coords(self.id)
        # 使得小球不会超出窗口
        pad=self.canvas.coords(self.paddle.id)          # 获取挡板的坐标
        if pos[1]<=0 :
            self.y=3
        if pos[3]>=self.canvas_heigh or(pos[3]>=pad[1] and \
pos[2]>=pad[0] and pos[2]<=pad[2]):
            self.y=-3
        if pos[0]<=0:
            self.x=3
        if pos[2]>=self.canvas_width:
            self.x=-3

class Paddle:  # 挡板的类
    def _ _init_ _(self,canvas,color):
        self.canvas=canvas
        self.color=color
        self.id=canvas.create_rectangle(0,0,100,10,fill=color)
        self.canvas.move(self.id,200,300)
        self.canvas_width=self.canvas.winfo_width()
        self.l=0
        self.r=0

    def draw(self):
        pos=self.canvas.coords(self.id)
        if pos[0]<=0:
            self.l=0
```

```
            if pos[2]>=self.canvas_width:
                self.r=0

        def turn_left(self,event):
            self.canvas.move(self.id,self.l,0)
            self.l=-20

        def turn_right(self,event):
            self.canvas.move(self.id,self.r,0)
            self.r=20
    tk=Tk()
    tk.title('Game')
    tk.resizable(0,0)                       # 使得窗口大小不可调整
    tk.wm_attributes('-topmost',1)          # 包含画布的窗口放在其他窗口的前面
    canvas=Canvas(tk,width=500,height=400,bd=0,highlightthickness=0)
                                            # 后面两个参数去掉边框
    canvas.pack()
    tk.update()
    paddle=Paddle(canvas,'blue')
    ball=Ball(canvas,paddle,'red')

    canvas.bind_all('<KeyPress-Left>',paddle.turn_left)     # 绑定方向键
    canvas.bind_all('<KeyPress-Right>',paddle.turn_right)

    while 1:
        ball.draw()
        paddle.draw()
        tk.update_idletasks()                               # 快速重画屏幕
        tk.update()
        time.sleep(0.01)
```

运行程序，红色小球一直处于运行状态，遇到墙壁（窗体边缘）或挡板则按相反路径弹回来，如图 5.12 所示。

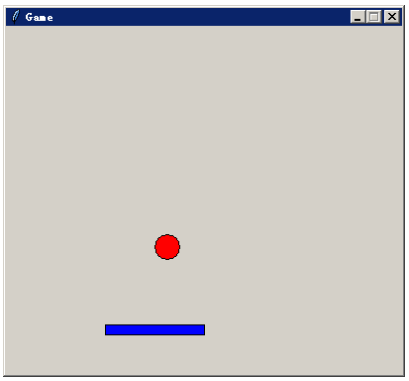


图 5.12　小球碰撞游戏

【例 5-12】 应用图像处理技术，编写一个简易图像处理器。

为了实现更多的图像处理功能，这里安装一个 Python 系统专业绘图库模块 matplotlib。通过 matplotlib，开发者可以仅需要几行代码，便可以方便地进行图像处理，也可以生成绘图、直方图、功率谱、条形图、散点图等。

可以用 pip 安装 matplotlib 模块，其安装命令如下：

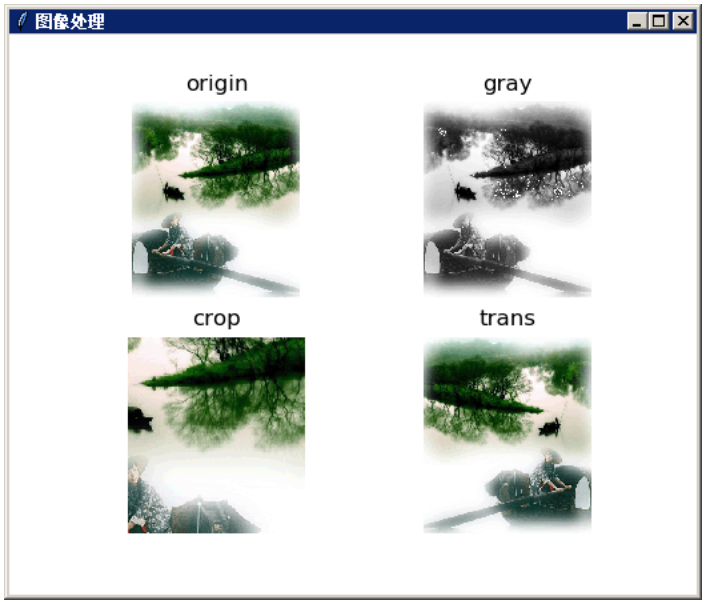
```
pip install matplotlib
```

程序代码如下：

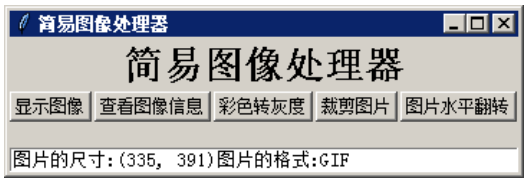
```
import tkinter
from tkinter import *
from PIL import Image
import matplotlib.pyplot as plt

# 定义窗体
win = Tk()
win.title("简易图像处理器")

# 定义标题
lab = Label(win, text='简易图像处理器', font=('Times','20','bold'))
lab.grid(row=0, column=0, columnspan=5)

# 定义空白标签
labss = Label(win,text="",width = 50,height = 1)
labss.grid(row=3, column=0, columnspan=5)

# 定义显示图像信息的文本框
s = StringVar()
txt = Entry(win,width=50,font=('宋体','10'),textvariable=s)
txt.grid(row=4, column=0, columnspan=5)

# 定义图像对象
img=Image.open('dukou.gif')
plt.figure("图像处理")

# 显示原像函数
def com_show():
    plt.subplot(2,2,1), plt.title('origin')   # 区域分成1行2列, 第1
    plt.imshow(img)
    plt.axis('off')
    plt.show()

# 查看图像信息函数
def com_info():
```

```
    plt.imshow(img)
    s.set('图片的尺寸:'+str(img.size)+'图片的格式:'+str(img.format))

# 转换灰度函数
def com_gray():
    plt.subplot(2,2,2), plt.title('gray')       # 区域分成1行2列, 第2
    gray=img.convert('L')                       # 转换成灰度
    plt.imshow(gray,cmap='gray')
    plt.axis('off')
    plt.show()

# 裁剪图片函数
def com_roi():
    box=(80,100,260,300)
    roi=img.crop(box)
    plt.subplot(2,2,3), plt.title('crop')       # 区域分成1行2列, 第3
    plt.imshow(roi),plt.axis('off')
    plt.show()

# 图片左右翻转函数
def com_trans():
    plt.subplot(2,2,4), plt.title('trans')      # 区域分成1行2列, 第4
    dst=img.transpose(Image.FLIP_LEFT_RIGHT)    # 左右翻转
    #img.rotate(45)    # 顺时针旋转45°
    plt.imshow(dst)
    plt.axis('off')
    plt.show()

# 定义按钮
btn_show = Button(win, text='显示图像', command=com_show)
btn_show.grid(row=2, column=0)

btn_show = Button(win, text='查看图像信息', command=com_info)
btn_show.grid(row=2, column=1)

btn_show = Button(win, text='彩色转灰度', command=com_gray)
btn_show.grid(row=2, column=2)

btn_show = Button(win, text='裁剪图片', command=com_roi)
btn_show.grid(row=2, column=3)

btn_show = Button(win, text='图片水平翻转', command=com_trans)
btn_show.grid(row=2, column=4)

win.mainloop()
```

程序运行结果如图 5.13 所示。



（a）单击按钮，显示图像处理结果



（b）程序运行窗体

图 5.13　简易图像处理器

【例 5-13】 应用 matplotlib 模块绘制指数曲线。

（1）首先用 arange()函数生成一个等差数列的数组。

arange()函数的一般格式为：

```
arange(初值, 终值, 等差值)
```

例如，函数 arange(0，10，2)所创建的等差数列为[2, 4, 6, 8]。

（2）应用 matplotlib 模块 plt 类的 plot()方法绘制指数曲线图。

程序代码如下：

```
import numpy as np
import matplotlib.pyplot as plt

t = np.arange(0, 5, 0.2)  # 生成等差数列, 其中等差值为0.2
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')  # 绘制指数曲线
plt.show()
```

程序运行结果如图 5.14 所示。

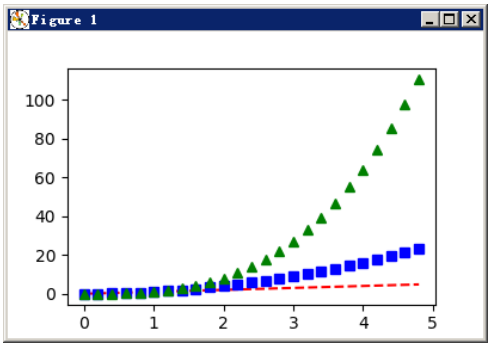


图 5.14　绘制曲线图形

习　题　5

1．绘制一个带阴影的小矩形块。

2．设计一个图片浏览器，单击“上一张”按钮，则显示前一张图片，单击“下一张”按钮，则显示后一张图片。