16 图形界面

Python支持多种图形界面的第三方库，包括：

- Tk
- wxWidgets
- Qt
- GTK

等等。

但是Python自带的库是支持Tk的Tkinter，使用Tkinter，无需安装任何包，就可以直接使用。本章简单介绍如何使用Tkinter进行GUI编程。

Tkinter

我们来梳理一下概念：

我们编写的Python代码会调用内置的Tkinter，Tkinter封装了访问Tk的接口；

Tk是一个图形库，支持多个操作系统，使用Tcl语言开发；

Tk会调用操作系统提供的本地GUI接口，完成最终的GUI。

所以，我们的代码只需要调用Tkinter提供的接口就可以了。

第一个GUI程序

使用Tkinter十分简单，我们来编写一个GUI版本的“Hello, world!”。

第一步是导入Tkinter包的所有内容：

```
from tkinter import *
```

第二步是从`Frame`派生一个`Application`类，这是所有Widget的父容器：

```
class Application(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.pack()
        self.createWidgets()

    def createWidgets(self):
        self.helloLabel = Label(self, text='Hello,
world!')
        self.helloLabel.pack()
        self.quitButton = Button(self, text='Quit',
command=self.quit)
        self.quitButton.pack()
```

在GUI中，每个Button、Label、输入框等，都是一个Widget。Frame则是可以容纳其他Widget的Widget，所有的Widget组合起来就是一棵树。

`pack()`方法把Widget加入到父容器中，并实现布局。`pack()`是最简单的布局，`grid()`可以实现更复杂的布局。

在`createWidgets()`方法中，我们创建一个`Label`和一个`Button`，当Button被点击时，触发`self.quit()`使程序退出。

第三步，实例化`Application`，并启动消息循环：

```
app = Application()
# 设置窗口标题:
app.master.title('Hello World')
# 主消息循环:
app.mainloop()
```

GUI程序的主线程负责监听来自操作系统的消息，并依次处理每一条消息。因此，如果消息处理非常耗时，就需要在新线程中处理。

运行这个GUI程序，可以看到下面的窗口：

点击“Quit”按钮或者窗口的“x”结束程序。

输入文本

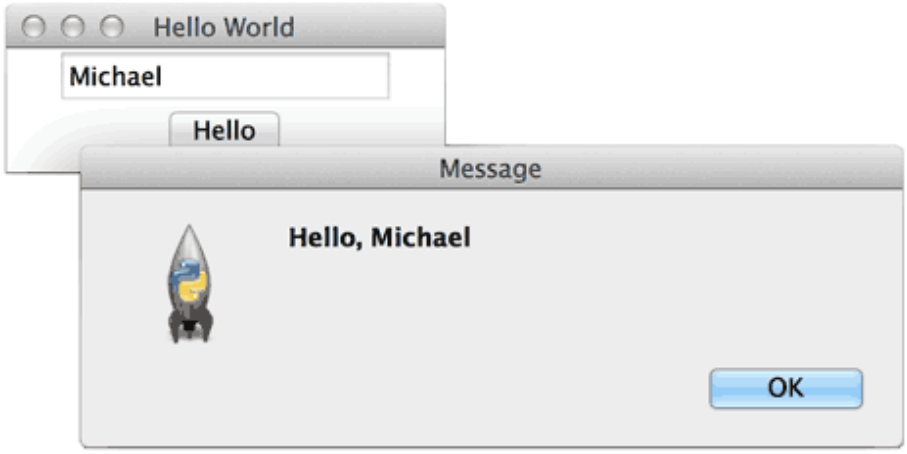
我们再对这个GUI程序改进一下，加入一个文本框，让用户可以输入文本，然后点按钮后，弹出消息对话框。

```
from tkinter import *
import tkinter.messagebox as messagebox
```

```
class Application(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.pack()
        self.createWidgets()

    def createWidgets(self):
        self.nameInput = Entry(self)
        self.nameInput.pack()
        self.alertButton = Button(self, text='Hello',
command=self.hello)
        self.alertButton.pack()

    def hello(self):
        name = self.nameInput.get() or 'world'
        messagebox.showinfo('Message', 'Hello, %s' % name)

app = Application()
# 设置窗口标题:
app.master.title('Hello World')
# 主消息循环:
app.mainloop()
```

当用户点击按钮时，触发`hello()`，通过`self.nameInput.get()`获得用户输入的文本后，使用`tkMessageBox.showinfo()`可以弹出消息对话框。

程序运行结果如下：



小结

- Python内置的Tkinter可以满足基本的GUI程序的要求，如果是非常复杂的GUI程序，建议用操作系统原生支持的语言和库来编写。

海龟绘图

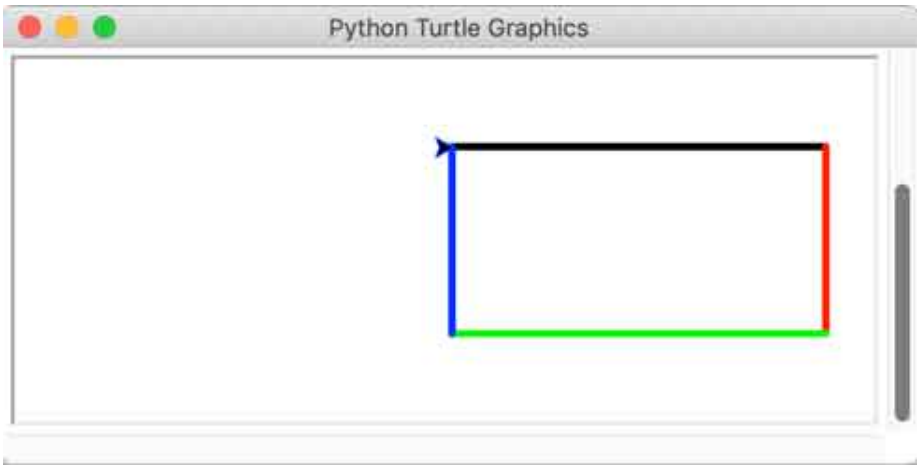
在1966年，Seymour Papert和Wally Feurzig发明了一种专门给儿童学习编程的语言——LOGO语言，它的特色就是通过编程指挥一个小海龟（turtle）在屏幕上绘图。

海龟绘图（Turtle Graphics）后来被移植到各种高级语言中，Python内置了turtle库，基本上100%复制了原始的Turtle Graphics的所有功能。

我们来看一个指挥小海龟绘制一个长方形的简单代码：

```
# 导入turtle包的所有内容:
from turtle import *

# 设置笔刷宽度:
width(4)

# 前进:
forward(200)
# 右转90度:
right(90)

# 笔刷颜色:
pencolor('red')
forward(100)
right(90)

pencolor('green')
forward(200)
right(90)

pencolor('blue')
forward(100)
right(90)

# 调用done()使得窗口等待被关闭，否则将立刻关闭窗口:
done()
```

在命令行运行上述代码，会自动弹出一个绘图窗口，然后绘制出一个长方形：



从程序代码可以看出，海龟绘图就是指挥海龟前进、转向，海龟移动的轨迹就是绘制的线条。要绘制一个长方形，只需要让海龟前进、右转90度，反复4次。

调用`width()`函数可以设置笔刷宽度，调用`pencolor()`函数可以设置颜色。更多操作请参考turtle库的说明。

绘图完成后，记得调用`done()`函数，让窗口进入消息循环，等待被关闭。否则，由于Python进程会立刻结束，将导致窗口被立刻关闭。

`turtle`包本身只是一个绘图库，但是配合Python代码，就可以绘制各种复杂的图形。例如，通过循环绘制5个五角星：

```
from turtle import *

def drawStar(x, y):
    pu()
    goto(x, y)
    pd()
    # set heading: 0
    seth(0)
    for i in range(5):
        fd(40)
        rt(144)

for x in range(0, 250, 50):
    drawStar(x, 0)

done()
```

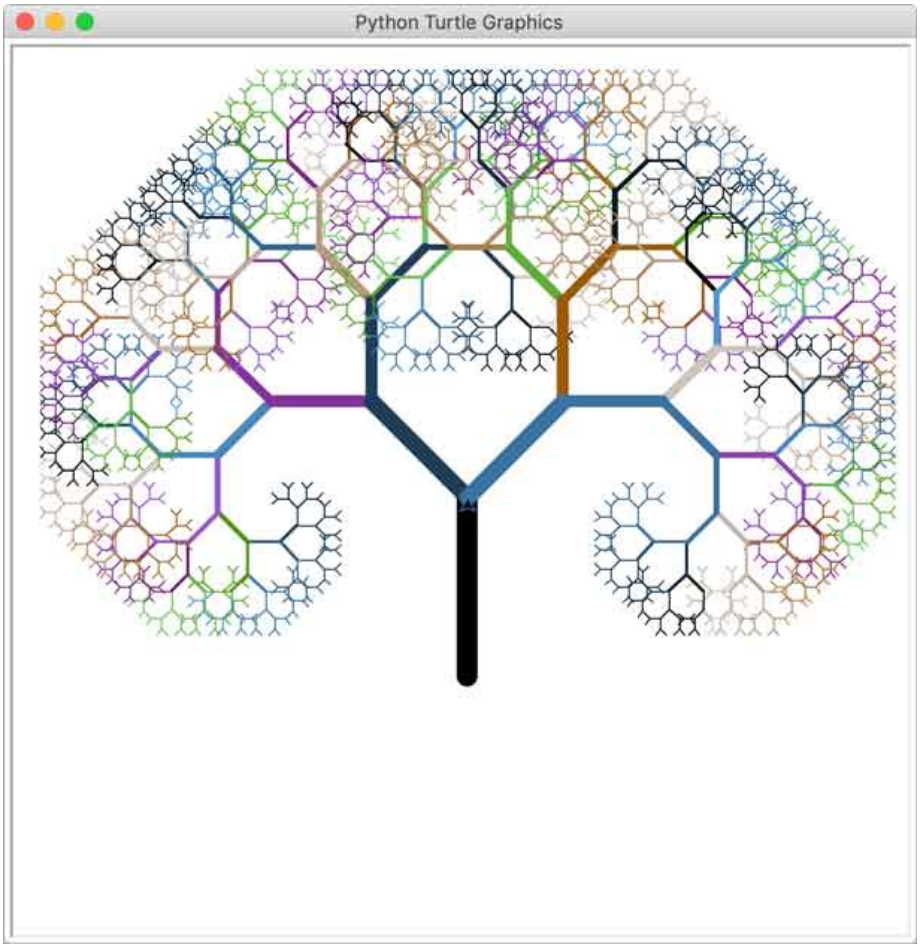
程序执行效果如下：

使用递归，可以绘制出非常复杂的图形。例如，下面的代码可以绘制一棵分型树：

```
from turtle import *

# 设置色彩模式是RGB:
colormode(255)

lt(90)

lv = 14
l = 120
s = 45

width(lv)

# 初始化RGB颜色:
```

```
r = 0
g = 0
b = 0
pencolor(r, g, b)

penup()
bk(l)
pendown()
fd(l)

def draw_tree(l, level):
    global r, g, b
    # save the current pen width
    w = width()

    # narrow the pen width
    width(w * 3.0 / 4.0)
    # set color:
    r = r + 1
    g = g + 2
    b = b + 3
    pencolor(r % 200, g % 200, b % 200)

    l = 3.0 / 4.0 * l

    lt(s)
    fd(l)

    if level < lv:
        draw_tree(l, level + 1)
    bk(l)
    rt(2 * s)
    fd(l)

    if level < lv:
        draw_tree(l, level + 1)
    bk(l)
    lt(s)

    # restore the previous pen width
    width(w)

speed("fastest")

draw_tree(l, 4)

done()
```

执行上述程序需要花费一定的时间，最后的效果如下：


Python Turtle Graphics