# 第十九章 图形用户界面编程



## 19.1 简介

Python 的默认 GUI 工具集是 Tk，它也是我们将使用的最基本的 GUI 工具集，我们可以通过 Python 接口 Tkinter 来使用 Tk(Tkinter 正是“Tk 接口”之意).

#### Tcl,Tk,Tkinter

如果导入模块Tkinter出错,这时不得不重编译 Python 解释器来访问 Tkinter。这通常会涉及编辑 Modules/Setup 文件和启用所有正确选项来编译您的 Python 解释器，以确保 Tkinter 能被选择安装在系统中。

在C/S模式中,窗口系统就是软件服务器的一个例子，它们运行在一个有显示设备的机器上， 比如带有一个某种类型的显示器。
当然还有**客户端**(那些需要窗口环境来运行的程序，也就是我们所说的 GUI 程序)，这些程序无法脱离窗口系统单独运行。

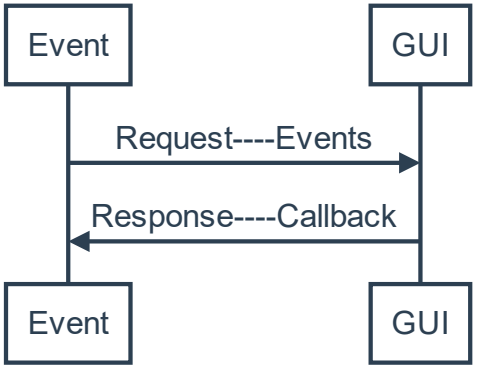
# Tkinter与Python编程

创建一个GUI程序的步骤:

1. 导入 Tkinter 模块– `from Tkinter import *`
2. 创建一个顶层窗口对象，来容纳您的整个 GUI 程序。– `top = Tkinter.Tk()`
3. 在您的顶层窗口对象上（或者说在“其中”）创建所有的 GUI 模块。–用 `packer/Grid` 布局组件
4. 把这些 GUI 模块与底层程序代码相连接。–功能实现
5. 进入事件主循环–即服务器式的无限循环 `Tkinter.mainloop()`

组件(Button,Label,Menu,Text等)通常会有一些行为,如Button被按下,Text文本框被写入等,这种形式的用户行为称为**事件Event**。而GUI程序对事件所采取的响应动作被称为**回调Callback**
GUI程序正式由按钮被按下释放,文本写入按下Enter键,鼠标的移动等这些一些列的整套事件体系所驱动的。这个过程称作**事件驱动处理**。

> 鼠标移动事件的详细过程
> 一个事件及其回调的例子是鼠标移动。我们假设鼠标指针停在您 GUI 程序的某处。如果鼠标被移到了程序的别处，一定是有什么东西引起了屏幕上指针的移动，从而表现这种位置的转移。系统必须处理这些鼠标移动事件才能展现（并实现）鼠标在窗口上的移动。一旦您释放了鼠标，就不再会有事件需要处理，相应地，屏幕上的一切又复归平静。




GUI程序的事件驱动特性恰好体现出它的客户端/服务器架构。

Tk 有两个坐标管理器用来协助把组件放在正确的位置上
* packer
* Grid

一旦packer决定好所有组件的尺寸和对齐方式，它将为您在屏幕上放置它们。
这时，GUI程序就会进入一个“服务器式”的无限循环。
这个无限循环包括
* 等待GUI事件
* 处理事件
* 等待下一个事件,即回到第一步

### 19.2.4 Tk组件

Tk 目前有 15 种组件。

| 组件 | 描述 |
| --- | --- |
| Canvas | 画布,提供绘图功能(直线/椭圆/多边形/矩形)，可以包含图形或位图 |
| Label | 标签。用来显示文字或图片 |
| Message | 消息框。类似于标签，但可以显示多行文本 |
| Button | 鼠标掠过、 按下、释放以及键盘操作/事件 |
| Entry | 文本框，单行文字域，用于键盘的输入 |
| Text | 文本域，多行文字域 |
| Radiobutton | radio button |
| Checkbutton | checkbox |

| | |
|---|---|
| Listbox | 列表框 |
| Menu | 点下菜单按钮后弹出的一个选项列表，用户可以从中选择 |
| Menubutton | 菜单按钮。用来包含菜单的组件 |
| Scale | 进度条 |
| Scrollbar | 滚动条,对其支持的组件(文本域/画布/列表框/文本框)提供滚动功能 |
| Frame | 包含其他组件的纯容器 |
| Toplevel | 类似Frame，但提供一个独立的窗口容器。 |

## 19.3 Tkinter举例

```
from Tkinter import * # 1.导入模块

def tk1():
    top = Tk(className="Label") # 2.创建用于容纳其他组件的顶层窗口对象top
    label = Label(top, text = "Hello World")
    label.pack() # 3.在顶层窗口对象top上创建label
    mainloop()# 5. 进入主循环

def tk2():
    top = Tk(className="Button")
    button = Button(top,text="Hello World", command=top.quit)
    button.pack()
    mainloop()

def tk3():
    top = Tk(className="Mix")
    label = Label(top, text="Hello World")
    label.pack()
    btn = Button(top, text="QUIT",command=top.quit,bg="red",fg="white")
    btn.pack(fill=X,expand=1) # 源码中有pack()的原型。btn的布局:填充水平方向的剩余空间,注意expand
参数,随着父窗口的变化而变化
    mainloop()

# 添加进度条Scale
def resize(ev=None):
    label.config(font='Courier -%d bold' % scale.get()) # 访问全局的label和scale
    print 'resize\t',scale.get()
```

```
top = Tk(className="Scale")
top.geometry('250x150') # 一定是小写的x，不是大写的X
'''Lable的具体参数配置
https://infohost.nmt.edu/tcc/help/pubs/tkinter/web/ttk-Label.html
其中font参数是字体的详细配置,其中的-12表示的是盖字体单位是像素pixels,如果是正的表示字体的单位是points,
points和pixels是字体单位的两种不同形式,另外还有其他形式,这个http://www.lai18.com/content/527389
7.html网址可以简单了解下
'''
label = Label(top,text='Hello World',font='Courier -12 bold') #Courier New不支持,而Courier
支持
label.pack(fill=Y,expand=1)

scale = Scale(top,from_=10,to=40,orient=HORIZONTAL,command=resize)
scale.set(24)
scale.pack(fill=X,expand=1)

def tk4():
    btnQuit = Button(top,text="QUIT",command=top.quit,activeforeground='white',activeback
ground='red')
    btnQuit.pack()
    mainloop()

from functools import partial as pto
from Tkinter import Tk,Button,X
from tkMessageBox import showinfo,showwarning,showerror

def tk5():
```

```
    WARN = 'warn'
    CRIT = 'crit'
    REGU = 'regu'

    SIGNS = {'do not enter':CRIT,
             'railroad crossing':WARN,
             '55\nspeed limit':REGU,
             'wrong way':CRIT,
             'merging traffic':WARN,
             'one way':REGU}
    critCB = lambda :showerror('Error','Error Button Pressed')
    warnCB = lambda :showwarning('Warning','Warning Button Pressed')
    infoCB = lambda :showinfo('Info','Info Button Pressed')

    top = Tk()
    top.title('Road Signs')
    Button(top,text='QUIT',command=top.quit,bg='red',fg='white').pack()
    MyButton = pto(Button,top) # PFA的功能
    CritButton = pto(MyButton,command=critCB,bg='white',fg='red')
    WarnButton = pto(MyButton,command=warnCB,bg='goldenrod1')
    ReguButton = pto(MyButton,command=infoCB,bg='white')

    for eachSign in SIGNS:
        signType = SIGNS[eachSign]
        #cmd = '%s Button(text=%r%s).pack(fill=X,expand=1)' % (signType.title(),eachSig
n,'.upper()' if signType == CRIT else '.title()') %sButton之间不能有空格
        cmd = '%sButton(text=%r%s).pack(fill=X, expand=1)' % (signType.title(), eachSig
n,'.upper()' if signType == CRIT else '.title()')
```

```
        eval(cmd)

    mainloop()


if __name__ == "__main__":
    #tk1()
    #tk2()
    #tk3()
    #tk4()
    tk5()
```

# 19.4 其他GUI简介

4种常用的图形工具集,Tix是包含在标准库中,其他3个位第三方的

* Tix(Tk interface eXtensions)
* Pmw(Python MegaWidgets的Tkinter扩展)
* wxPython(wxWidgets的Python绑定)
* PyGTK(GTK+的Python绑定)

```
from Tkinter import Label,Button,END
from Tix import Tk,Control,ComboBox

top = Tk()
top.tk.eval('package require Tix')

lb = Label(top, text='Animals (in pairs; min: pair, max: dozen)')
lb.pack()

ct = Control(top,label='Number:',integer=True,max=12,min=2,value=2,step=2)
ct.label.config(font='Helvetica -14 bold')
ct.pack()

cb = ComboBox(top, label='Type:',editable=True)
for animal in ('dog','cat','hamster','python'):
    cb.insert(END,animal)
cb.pack()

qb = Button(top,text='QUIT',command=top.quit,fg='white')
qb.pack()

top.mainloop()
```

## 19.5 相关模块和其他GUI

其他的 GUI 开发系统,以下列出适当的模块及其对应的窗口系统

Python 可用的 GUI 系统

| GUI模块或系统 | 描述 |
| --- | --- |
| `Tkinter` | TK INTERface: Python 的默认 GUI 工具集 |
| `Pwm` | Python MegaWidgets(Tkinter 扩展),http://pmw.sf.net |
| `Tix` | Tk Interface eXtension,Tk扩展http://tix.sf.net |
| `TkZinc` | Extended Tk canvas type,Tk扩展,http://www.tkzinc.org |
| `EasyGUI` | 非常简单的非事件驱动 GUI,Tkinter扩展,http://ferg.org/easygui |
| `TIDE+(IDE Studio)` | Tix集成开发环境,http://starship.python.net/crew/mike |

具体参见课本Page 779