Python Short Course
Lecture 6: Tk Graphics

Richard P. Muller

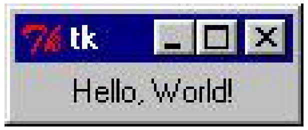
Materials and Process Simulation Center

June 22, 2000

Tk Overview

- Set of widgets designed by John K. Ousterhout, 1987
- Based on Apple Hypercard idea of putting together graphics program
- Tk == Tool Kit
- Mean to be driven by Tcl (Toolkit Control Language)
  - Many people find Tcl limited
  - Can also drive Tk with Perl, Python
- Tkinter is the Python Tk Interface
  - Very easy to use

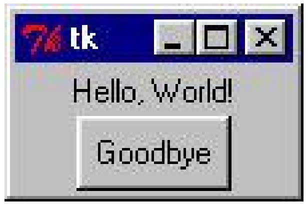
Hello, World



```
from Tkinter import *
w=Label(text="Hello, World!")
w.pack()
w.mainloop()
```

- Label() defines a label to be displayed
  - text= specifies a parameter to be passed in
- pack() resizes the window to the proper size
- mainloop() enters the event loop, and the program idles until a button is pushed, a menu is pulled, etc. It has to idle until the program is killed, since we didn't define any events.

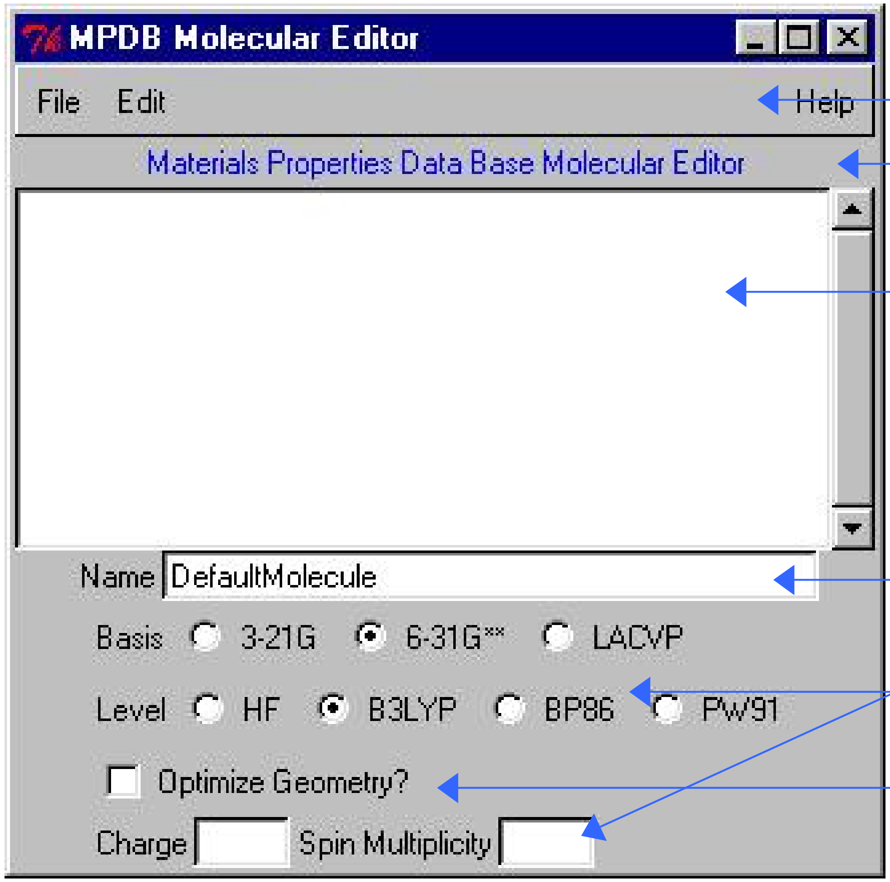
Events (Hello, Goodbye)



```
from Tkinter import *
w=Label(text="Hello, World).pack()
b=Button(text="Goodbye",command='exit').pack()
mainloop()
```

- Button label defined by text parameter
- Button defines a callback function, something to run when it is pushed.
- Now mainloop() has an event to catch, so when we push the button, mainloop() executes the exit command.

Creating a Molecular Editor



Materials Properties Data Base Molecular Editor

Name DefaultMolecule

Basis ○ 3-21G ● 6-31G** ○ LACVP

Level ○ HF ● B3LYP ○ BP86 ○ PW91

☐ Optimize Geometry?

Charge ____ Spin Multiplicity ____

Molecular Editor Overview

- We're going to whiz through this fairly quickly
  - Example is online for those who want more
  - Just a survey of some different widgets
  - How you can build a professional looking interface

Widgets Creation Routine

```
def makeWidgets(self):
       frame = Frame(self)
       self.makeMenuBar(frame)
       self.makeLogo(frame)
       self.makeMolEdit(frame)
       self.makeNameEntry(frame)
       self.makeSelectQM(frame)
       frame.pack()
       self.pack()
       return
```

Frames & Containers

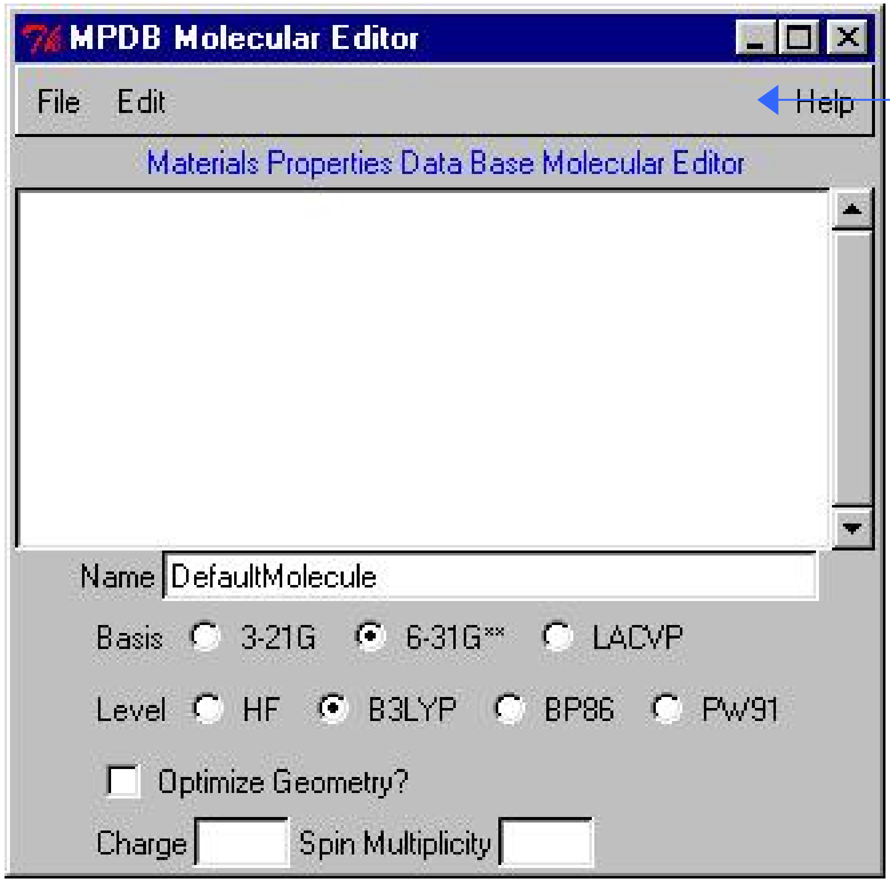
```
frame = Frame(self)
```

- Frame() is a general command to create a container for other widgets
- It doesn't do much other than hold other things.
- Takes as an argument the parent (here self)
- Returns the frame object (here frame)
- We can then pass the frame object to other widgets as their parent

```
self.makeMenuBar(frame)
```

- Frame is also useful for doing sophisticated layouts
  - Tk doesn't give much control over precise layout
  - Often have to pack frames within frames within frames

Menubars and Menus



Materials Properties Data Base Molecular Editor

Name DefaultMolecule

Basis ( ) 3-21G [✓ 6-31G**] ( ) LACVP

Level ( ) HF [✓ B3LYP] ( ) BP86 ( ) PW91

☐ Optimize Geometry?

Charge Spin Multiplicity

Menu bar

Menubars

- A menubar is just a frame that holds menus:

```
menubar = Frame(frame,relief=RAISED,borderwidth=1)
menubar.pack(side=TOP)
```

  - We've specified a raised relief, and a slight border
  - We've also specified where to pack the widget (TOP)
  - We will then pass menubar to all of the subsequent menus we'll define (File, Edit, Help, etc.) as the parent function.

Menus

- A menu in Tk is a combination of a Menubutton (the title of the menu) and the Menu (what drops down when the Menubutton is pressed

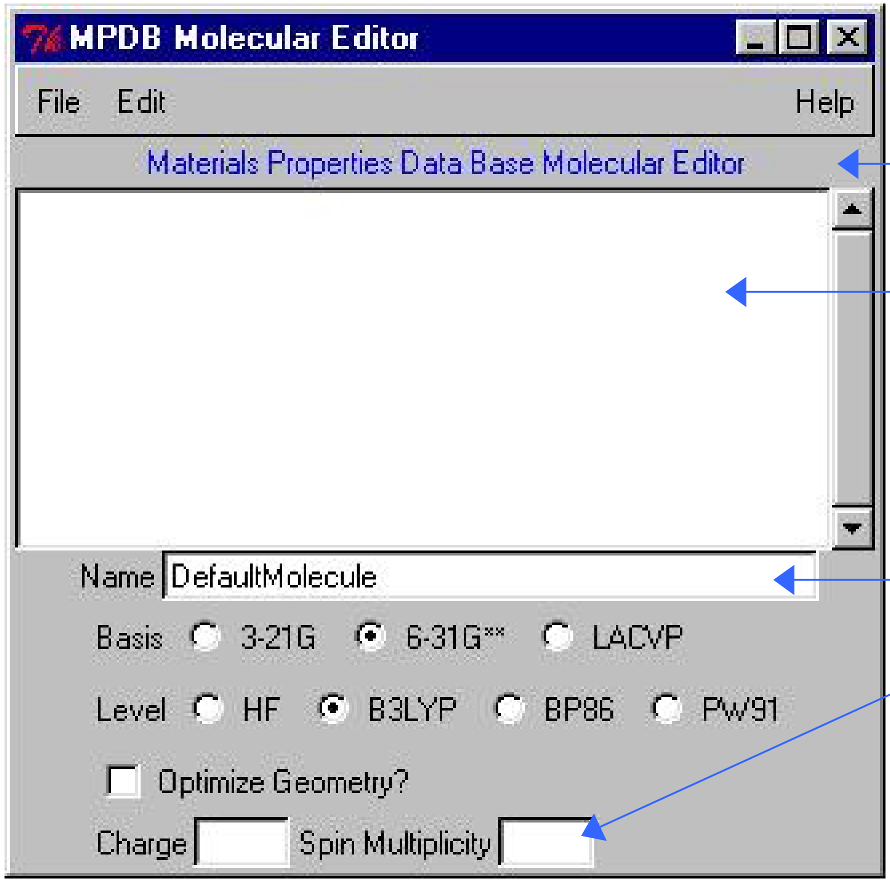
```
    mb_file = Menubutton(menubar, text='File')
    mb_file.pack(side=LEFT)
    mb_file.menu = Menu(mb_file)
```

- Once we've specified the menubutton and the menu, we can add different commands to the menu

```
    mb_file.menu.add_command(
         label='New...',
         command = self.new_mol)
```

  - Here we've defined a new type of callback, one that points to one of our functions (self.new_mol) rather than a predefined function

Text Widgets



Materials Properties Data Base Molecular Editor

Name: DefaultMolecule

Basis: ( ) 3-21G [✓ 6-31G**] ( ) LACVP

Level: ( ) HF [✓ B3LYP] ( ) BP86 ( ) PW91

☐ Optimize Geometry?

Charge: Spin Multiplicity:

Label

Text area (for geometry input)

Text entry

Text Areas

- Text areas contain room for multiple lines of text
  - Define a new frame and put a text area in it

```
textfr = Frame(frame)
self.text = Text(textfr,height=10,width=50)
```

  - Put a scrollbar in this frame

```
scroll = Scrollbar(textfr,command =
         self.text.yview)
self.text.configure(yscrollcommand=scroll.set)
```

  - Pack everything

```
self.text.pack(side=LEFT)
scroll.pack(side=RIGHT,fill=Y)
textfr.pack(side=TOP)
```

Text Entries

- Text entries contain single lines of text
  - Create a new frame for the entry, and put a label on it

```
entry_frame = Frame(frame)
Label(entry_frame,text = 'Name')
  .pack(side=LEFT)
```

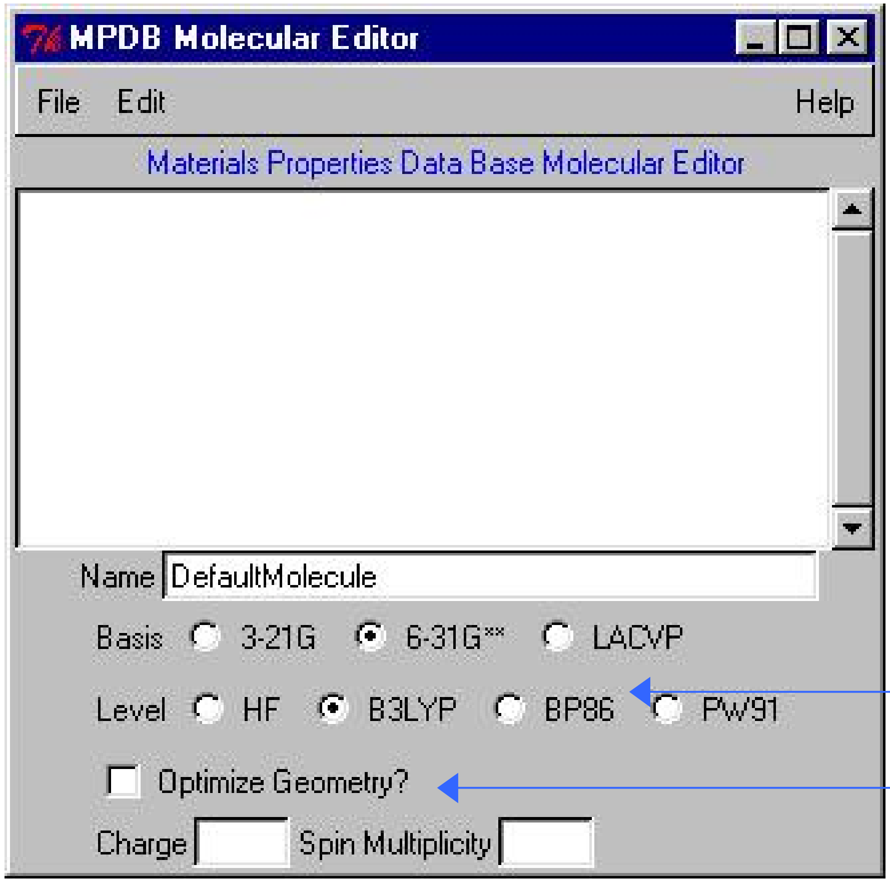
  - Define the entry, connect it to a variable, and put the current value of the variable in the entry

```
entry = Entry(f1,width=40,
  textvariable = self.mol_name)
entry.insert(0,self.mol_name)
```

  - Pack everything

```
entry.pack(side=LEFT)
entry_frame.pack(side=TOP,fill=Y)
```

Radiobuttons and Checkboxes



MPDB Molecular Editor

File Edit Help

Materials Properties Data Base Molecular Editor

Name DefaultMolecule

Basis ○ 3-21G [✓ ● 6-31G**] ○ LACVP

Level ○ HF [✓ ● B3LYP] ○ BP86 ○ PW91

☐ Optimize Geometry?

Charge Spin Multiplicity

Radio buttons

Checkbox

Radiobuttons

- Radiobuttons signify a choice between exclusive options
  - Create a frame and label

```
rbfr = Frame(f)
Label(rbfr,text='Basis').pack(side=LEFT)
```

  - Add the buttons. Note that the variable connected to all buttons is self.basis

```
r321 = Radiobutton(rbfr,text='3-21G',
      value = '3-21G',variable=self.basis)
r321.pack(side=LEFT)
r631 = Radiobutton(rbfr,text='6-31G**',
      value = '6-31G**', variable=self.basis)
r631.pack(side=LEFT)
```

  - Set the default and pack

```
r631.select()
rbfr.pack(side=TOP,fill=X)
```

Checkboxes

- Check boxes represent boolean choices (T or F)

```
cbfr = Frame(f)
```

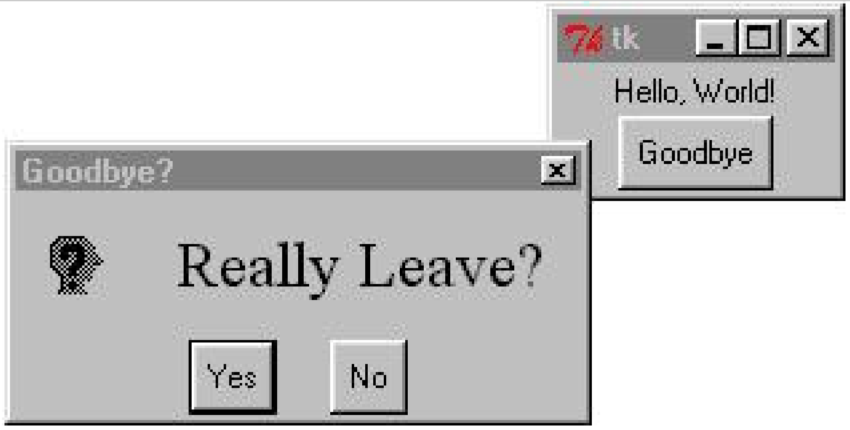
  - Add the buttons. Note that the variables are different.

```
cbgeo = Checkbutton(cbfr,
      text='Optimize Geometry?',
      state=NORMAL,
      variable=self.geo_opt).pack(side=LEFT)
cbsolv = Checkbutton(cbfr,
      text='Solvate?',
      state=NORMAL,
      variable=self.solvated).pack(side=LEFT)
cbfr.pack(side=TOP)
```

Notes

- This interface doesn't do anything; to make it work
  - Add Run command to File menu?
  - Put Submit button at the bottom?
  - Tie these commands to function calls
- Synergy between objects and widgets
  - Variables are passed automatically within class; you can refer to them as self.whatever and not have to worry about passing variables
  - Callback functions are similarly easy to handle; this is a particularly good deal because often programmers jump through hoops to define callbacks on the fly (lambda functions). IMHO this is a source of confusion and should be avoided.

Dialog boxes



- Convenient way to get feedback from a user
  - Confirm quit
  - Inputs data directly into program
  - Here 0 is returned for Yes, and 1 is returned for No

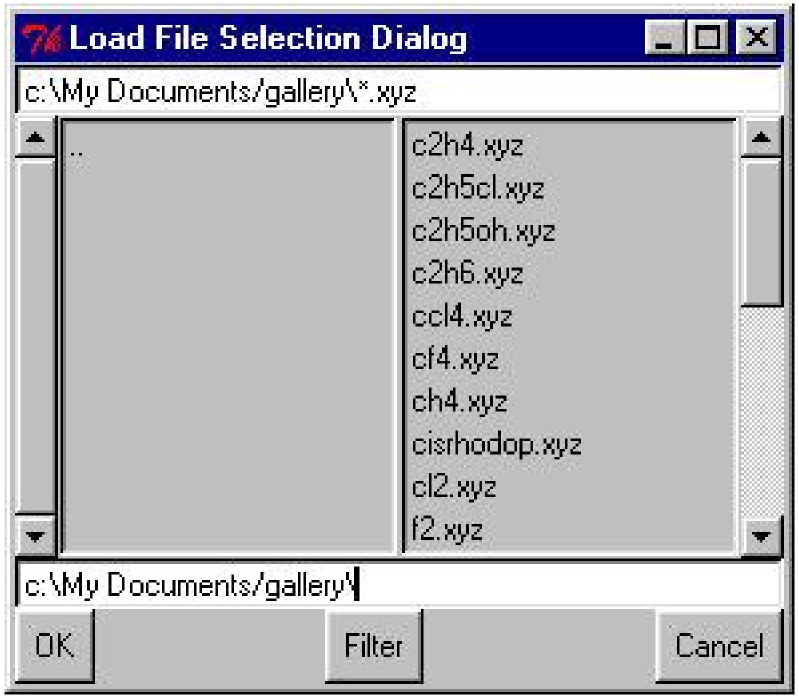
Simple Dialog Box Example

```
import sys
from Tkinter import *
from Dialog import *

def confirm_quit():
  d = Dialog(None, title="Goodbye?",
       text="Really Leave?", default=0,
       bitmap=DIALOG_ICON, strings=("Yes","No"))
  if d.num ==0:sys.exit()
  return

l = Label(text="Hello, World!").pack()
b = Button(text="Goodbye",
  command=confirm_quit).pack()
mainloop()
```

File Browser Dialog



File Dialog Example Code

```
from Tkinter import *
from FileDialog import *

root = Tk()
```

- Set up the dialog box

```
filename=LoadFileDialog(root)
```

- Run it. Optionally you can give it a default directory and file filter, as shown here:

```
filename.go("~/gallery","*.xyz")
print filename
```

Python Mega Widgets

- Very extensive set of sophisticated widgets
  - counters, panes, dialogs, fields already having scrollbars, groups of widgets, etc.
- Built from basic Tk widgets
  - People are adding new ones all the time
- On MSC machines at /source/python/Pmw
  - Not currently installed
  - I'll be glad to install if anyone wants them
- Available on the web at http://www.dscpl.com.au/pmw

wxPython

- Python bindings for wxWindows widget set
- Very professionally done
- wxWindows is available on all platforms
- Many notables in the Python community (Eric Raymond) are calling for wxPython to become the standard
- Not currently installed at MSC
  - I'll be glad to do so if there is desire
  - Still much more acceptance and much more use for Tkinter
- Available on the web at http://wxpython.org

References

- Web Pages
  - Tkinter: http://www.python.org/topics/tkinter/doc.html
  - Python megawidgets: http://www.dscpl.com.au/pmw
  - wxPython: http://wxpython.org
- Books
  - Programming Python, Mark Lutz, ORA
  - Python and Tkinter Programming, John E. Grayson, Manning Press
  - Tcl and the Tk Toolkit, John K. Ousterhout, Addison-Wesley Professional Computing Series