CS2021
Week #6
GUI Programming using Tkinter

Tkinter structure

Requires integration with Tk – a GUI library

Python program makes widgets and registers functions to handle widget events

Program consist of theses registrations, and is event driven, so does nothing but wait for events to occur

Programs use event handlers

called callback handlers – since Tk calls back to python code when events occur.

Example 7-12. PP4E\Gui \Intro\gui3.py

```
import sys
from tkinter import *

def quit():                         # a
custom callback handler
    print('Hello, I must be going...')
    sys.exit()

widget = Button(None,
```

```
text='Hello event world',
                    command=quit)
widget.pack()
widget.mainloop()
```

Lambda Callback Handler

Example 7-13 - gui3b.py

```
import sys
from tkinter import *      # lambda generates a function
widget = Button(None, # but contains just an expression
    text='Hello event world',
    command=(lambda: print('Hello
```

```
    lambda world') or     sys.exit()) )
widget.pack()
widget.mainloop()
```

Callback Scope Issues

Python's enclosing scope reference model means that the value of X in the local scope enclosing the lambda function is automatically retained, for use later when the button press occurs.

Lambdas simply defer the call to the actual handler and provide extra handler arguments. Variables from the enclosing

scope used by the lambda are automatically retained, even after the enclosing function exits. Enclosing scope references are resolved at function call time, not at function creation time.

```
def handler(A, B):
...use A and B...
def makegui():
    X = 42
    Button(text='ni', command=(lambda:
    handler(X, 'spam')))       # remembers X
makegui()
mainloop()
```

What does this code

do?

```
def weird():
   spam = 42
   handler = (lambda: spam * 2)
   spam = 50
   print(handler())
   spam = 60
   print(handler()

 weird()
```

Binding Handlers

Modified gui3e.py

```
import sys
from tkinter import *
def hello(event):
    print('Press twice to exit')
# on single-left click
def quit(event):
# on double-left click
```

```
    print('Hello, I must be going...’)
# event gives widget
    top.destroy()

top = Tk()
widget = Button(top, text='Hello event
world')
widget.pack()
widget.bind('<Button-1>', hello)          #
bind left mouse clicks
widget.bind('<Double-1>', quit)          #
bind double-left clicks
widget.mainloop()
```

Multiple Widgets
gui4.py (p 395)

```
from tkinter import *

def greeting():
    print('Hello stdout world!...')

win = Frame()
```

```
win.pack(side=TOP, expand=YES,
fill=BOTH)
Button(win, text='Hello',
command=greeting).pack(side=L
EFT)
Label(win,  text='Hello container
world').pack(side=TOP)
Button(win, text='Quit',
command=win.destroy).pack(sid
e=RIGHT)

win.mainloop()
```

Widgets with Classes

gui5.py

```
from tkinter import *
```

```
class HelloButton(Button):
    def __init__(self, parent=None,
**config):
          Button.__init__(self, parent,
**config)
          self.pack()

self.config(command=self.callback)

    def callback(self):
         print('Goodbye world...')
         self.quit()

if __name__ == '__main__':
    HelloButton(text='Hello subclass
world').mainloop()
```

Themed Widgets - gui5b.py

```
from tkinter import *

class
```

```
ThemedButton(Button):
# config my style too
    def __init__(self,
parent=None, **configs):
# used for each instance

Button.__init__(self,
parent, **configs)   #
later for options
        self.pack()

self.config(fg='red',
bg='black',
font=('courier', 12),
relief=RAISED, bd=5)

def onSpam():
```

```
print('Spam')
B1 =
ThemedButton(text='spam'
, command=onSpam)

# normal button widget
objects
B2 =
ThemedButton(text='eggs'
)          # but same
appearance by
inheritance
B2.pack(expand=YES,
fill=BOTH)
mainloop()
```

User themed – gui5b-

themed.py

```
from tkinter import *
from user_preferences import bcolor, bfont, bsize     # get user settings

class ThemedButton(Button):
   def __init__(self, parent=None, **configs):
      Button.__init__(self, parent, **configs)
      self.pack()
      self.config(bg=bcolor, font=(bfont, bsize))

def onSpam(): print('Spam')
def onEggs(): print('Eggs')

ThemedButton(text='spam', command=onSpam)  # normal button widget
objects
ThemedButton(text='eggs', command=onEggs)  # all inherit user
preferences

class MyButton(ThemedButton):             # subclasses inherit prefs too
   def __init__(self, parent=None, **configs):
      ThemedButton.__init__(self, parent,  **configs)
      self.config(text='subclass')

MyButton(command=onSpam)
mainloop()
```

Reusable components using Classes

gui6.py

```
from tkinter import *
```

```
class Hello(Frame):                         # an extended Frame
    def __init__(self, parent=None):
        Frame.__init__(self, parent)            # do superclass init
        self.pack()
        self.data = 42
        self.make_widgets()                  # attach widgets to self

    def make_widgets(self):
        widget = Button(self, text='Hello frame world!',
command=self.message)
        widget.pack(side=LEFT)

    def message(self):
        self.data += 1
        print('Hello frame world %s!' % self.data)

Hello().mainloop()
```

Attaching to an Enclosing Widget

gui6b.py

```
from sys import exit
from tkinter import *
# get Tk widget classes
from gui6 import Hello
```

```
# get the subframe class

parent = Frame(None)
# make a container widget
parent.pack()
Hello(parent).pack(side=RIGHT)
                            # attach Hello
instead of running it

Button(parent, text='Attach',
command=exit).pack(side=LEFT)
parent.mainloop()
```

Hello Container – gui6c.py

```
from tkinter import *
```

```
# get Tk widget classes
from gui6 import Hello
# get the subframe class

class HelloContainer(Frame):
    def __init__(self,
parent=None):
        Frame.__init__(self, parent)
        self.pack()
        self.makeWidgets()

    def makeWidgets(self):
        Hello(self).pack(side=RIGHT)
# attach a Hello to me
        Button(self, text='Attach',
command=self.quit).pack(side=L
EFT)
```

```
if __name__ == '__main__':
HelloContainer().mainloop()
```

Standalone Container – gui7.py

```
from tkinter import *

class HelloPackage:                     # not a widget subbclass
   def __init__(self, parent=None):
      self.top = Frame(parent)            # embed a Frame
      self.top.pack()
      self.data = 0
      self.make_widgets()               # attach widgets to self.top

   def make_widgets(self):
      Button(self.top, text='Bye',
command=self.top.quit).pack(side=LEFT)
      Button(self.top, text='Hye',
command=self.message).pack(side=RIGHT)

   def message(self):
      self.data += 1
      print('Hello number', self.data)

if __name__ == '__main__': HelloPackage().top.mainloop()
```

Variable Classes

Some widgets (like text entry widgets, radio buttons and so on) can be connected directly to application variables by using special options: variable, textvariable, onvalue, offvalue, and value

x = StringVar() # Holds a string; default value ""

x = IntVar() # Holds an integer; default value 0

x = DoubleVar() # Holds a float; default value 0.0

x = BooleanVar() # Holds a boolean, returns 0 for False

and 1 for True

Simple RadioButtons

```
from tkinter import *
root = Tk()
v = IntVar()
Label(root,
      text="""Choose the best programming
language:""",
      justify = LEFT,
      padx = 20).pack()
Radiobutton(root, text="Python", padx = 20,
variable=v,value=1).pack(anchor=W)
Radiobutton(root, text="Perl", padx = 20,
variable=v, value=2).pack(anchor=W)
Radiobutton(root, text="C++", padx = 20,
variable=v, value=3).pack(anchor=W)
Radiobutton(root, text="Java", padx = 20,
variable=v, value=4).pack(anchor=W)

mainloop()
```

Improvements

```
from tkinter import *
root = Tk()
v = IntVar()
v.set(1)  # initializing the choice, i.e. Python
```

```
languages = [("Python",1), ("Perl",2),("Java",3),
("C++",4),("C",5)]

def ShowChoice():
    print v.get()

Label(root,
      text="""Choose your favourite programming
language:""",
      justify = LEFT,padx = 20).pack()

for (txt, val) in languages:
    Radiobutton(root,
                text=txt,
                padx = 20,
                variable=v,
                command=ShowChoice,
                value=val).pack(anchor=W)
mainloop()
```

Entrys and Grids

```
from tkinter import *

master = Tk()
Label(master, text="First
Name").grid(row=0)
Label(master, text="Last
```

Name").grid(row=1)

e1 = Entry(master)
e2 = Entry(master)

e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

mainloop( )

Read the content of an Entry

```
from tkinter import *

def show_entry_fields():
   print("First Name: %s\nLast Name: %s" % (e1.get(),
e2.get()))

master = Tk()
Label(master, text="First Name").grid(row=0)
```

```
Label(master, text="Last Name").grid(row=1)

e1 = Entry(master)
e2 = Entry(master)

e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

Button(master, text='Quit',
command=master.quit).grid(row=3, column=0, sticky=W,
pady=4)
Button(master, text='Show',
command=show_entry_fields).grid(row=3, column=1,
sticky=W, pady=4)

mainloop( )
```