# ENGR 102 PROGRAMMING PRACTICE

## WEEK 4

# Graphical User Interface (GUI) Programming

# Organizing Widgets – Layouts

- **Place geometry manager**: (**we will skip it**)
  - lets you explicitly place a widget in a given position.
  - to use this geometry manager, use the **place** method.
- **Pack geometry manager**:
  - treats widgets as rectangular blocks placed in a frame.
  - to use this geometry manager, use the **pack** method
- **Grid geometry manager**:
  - allows you to create table-like layouts
  - organizing the widgets in a 2-dimensional grid
  - to use this geometry manager, use the **grid** method.

# Place Geometry Manager

- simplest geometry manager
- lets you explicitly place a widget in a given position.
- to use this geometry manager, use the **place** method.
- ***avoid*** it in practice as much as possible

```
var = StringVar()
b = Button(self, width=12, height=12, text = 'hello!')
b.place(x=10, y=2, anchor=NW)
label = Label(self, textvariable=var)
var.set("Hey!? How are you doing?")
label.place(relx=0.5, rely=0.5, anchor=NW)
```

# Pack Geometry Manager

- Treats widgets as rectangular blocks
- Useful in three different settings:
  - Place a number of widgets **on top of each other along a vertical line**
  - Place a number of widgets **side by side along a horizontal line**
  - Put widget(s) inside a container (window, frame, etc.), and have it fill the **entire** container

# Pack Geometry Manager

**1.** Place a number of widgets **on top of each other**.

(This is the default behavior)

```
from tkinter import *

root = Tk()

red = Label(root, text="Red", bg="red", fg="white")
red.pack()
green = Label(root, text="Green", bg="green", fg="black")
green.pack()
blue = Label(root, text="Blue", bg="blue", fg="white")
blue.pack()

mainloop()
```

# Pack Geometry Manager

**2.** Place a number of widgets **side by side**.

```
from tkinter import *

root = Tk()

red = Label(root, text="Red", bg="red", fg="white")
red.pack(side=LEFT)
green = Label(root, text="Green", bg="green", fg="black")
green.pack(side=LEFT)
blue = Label(root, text="Blue", bg="blue", fg="white")
blue.pack(side=LEFT)

mainloop()
```

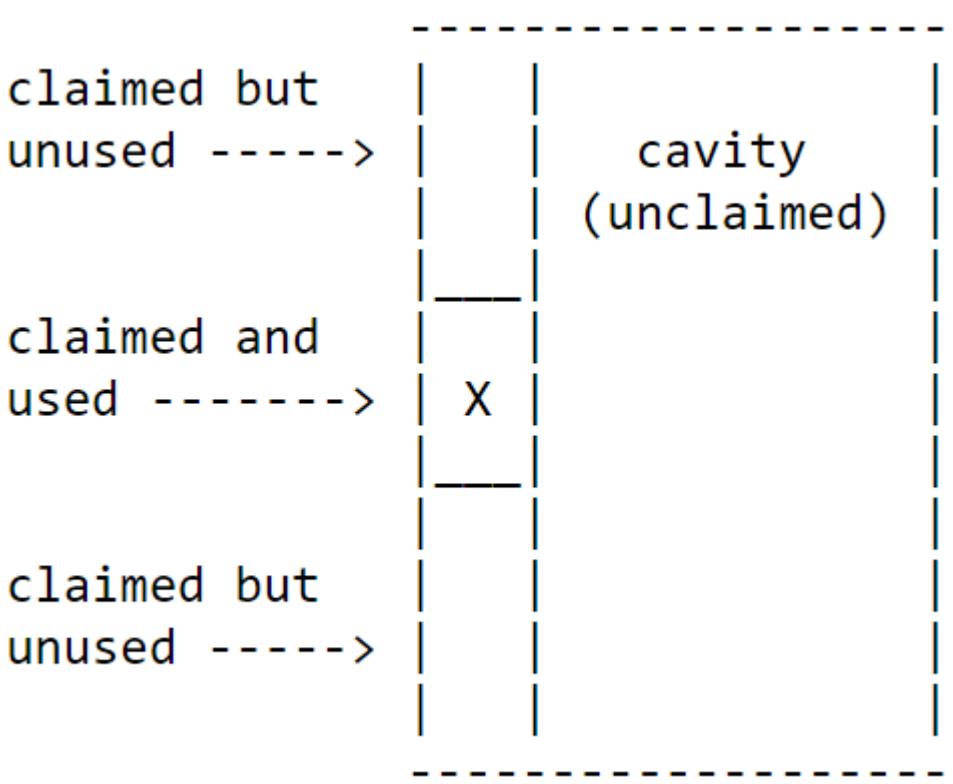
# Pack Geometry Manager

**3.** Put widget(s) inside a container (window, frame, etc.), and have it fill the **entire** container.

```
from tkinter import *

root = Tk()

red = Label(root, text="Red", bg="red", fg="white")
red.pack(side=LEFT, fill=BOTH, expand=True)
green = Label(root, text="Green", bg="green", fg="black")
green.pack(side=LEFT, fill=BOTH, expand=True)
blue = Label(root, text="Blue", bg="blue", fg="white")
blue.pack(side=LEFT, fill=BOTH, expand=True)

mainloop()
```

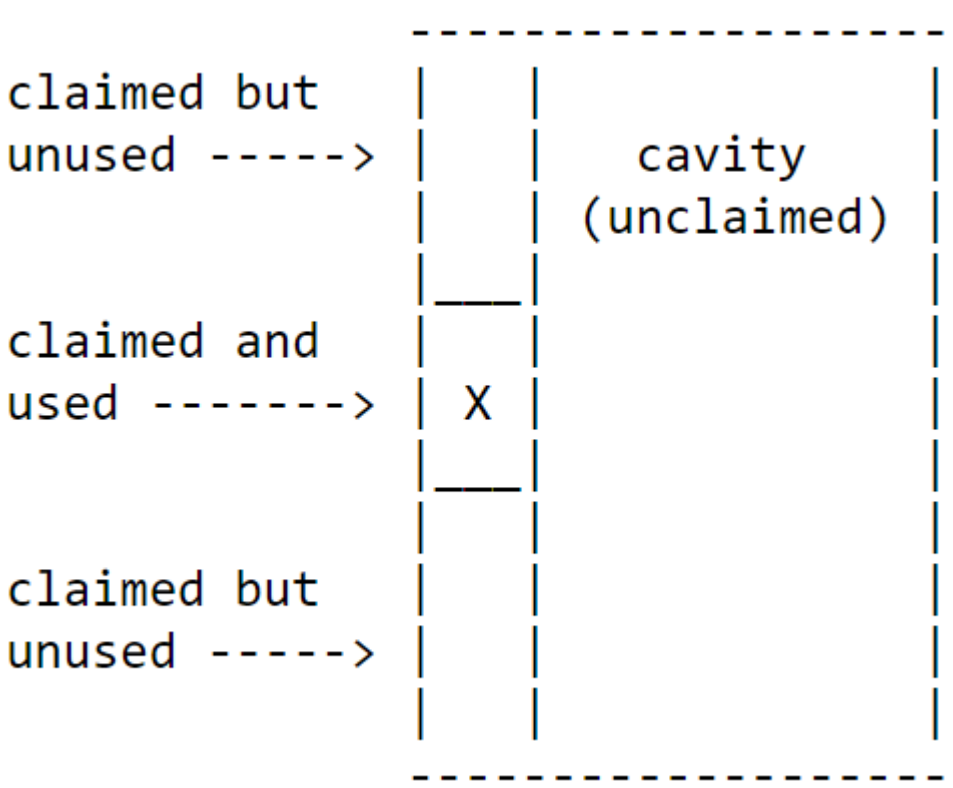
# Pack Geometry Manager
## expand & fill

X: a widget



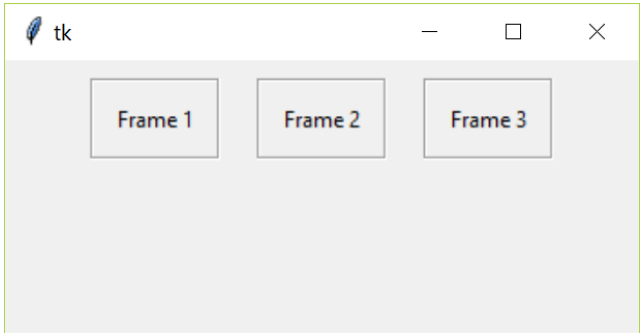

- unclaimed space (that is, the cavity)
- claimed but unused space
- claimed and used space

# Pack Geometry Manager
## expand & fill

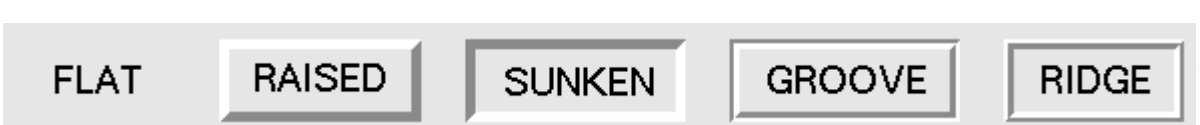



- **fill** = X | Y | BOTH
  - fill the claimed space in specified direction/axis
- **expand** = True
  - claim the available space (i.e., cavity)

# Frame Widget



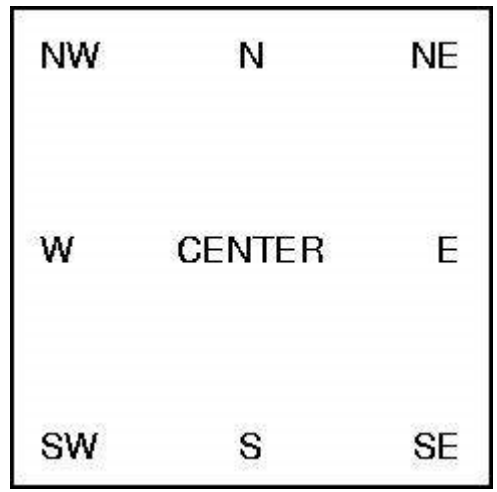

```
# frame 1
frame1 = Frame(self, borderwidth=2, relief=GROOVE)
frame1.pack(side=LEFT, padx=10, pady=10)
# frame 2
frame2 = Frame(self, borderwidth=2, relief=GROOVE)
frame2.pack(side=LEFT, padx=10, pady=10)
# frame 3
frame3 = Frame(self, borderwidth=2, relief=GROOVE)
frame3.pack(side=LEFT, padx=10, pady=10)
# Adjust the labels
label1 = Label(frame1, text="Frame 1")
label1.pack(side=LEFT, padx=10, pady=10)
label2 = Label(frame2, text="Frame 2")
label2.pack(padx=10, pady=10)
label3 = Label(frame3, text="Frame 3")
label3.pack(padx=10, pady=10)
```




Relief styles

# Pack Geometry Manager

## Anchor Parameter



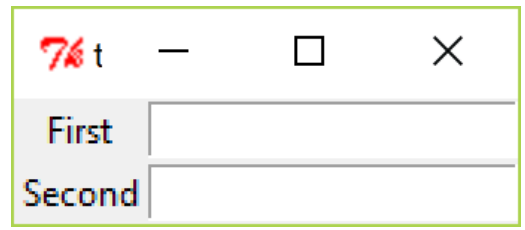

Play with sandbox app (LiveConfig.py) to see how different parameters affect widget positioning with pack manager.

# Grid Geometry Manager

- Puts the widgets in a 2-dimensional table.
- The parent container is split into a number of rows and columns
  - each “cell” in the resulting table can hold a widget.
- First, create the widgets, and
- Then, use the **grid** method to tell the manager in which row and column to place them.
- No need to specify the size of the grid beforehand; the manager automatically figures that out.

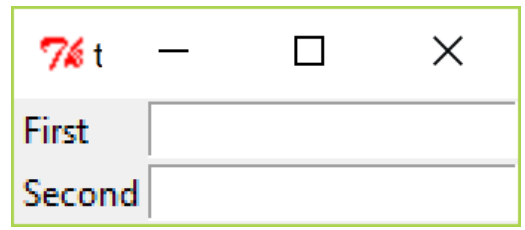
# Grid Geometry Manager

```
from tkinter import *

root = Tk()

l1 = Label(root, text="First")
l2 = Label(root, text="Second")

l1.grid(row=0, column=0)
l2.grid(row=1, column=0)

e1 = Entry(root)
e2 = Entry(root)

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

root.mainloop()
```

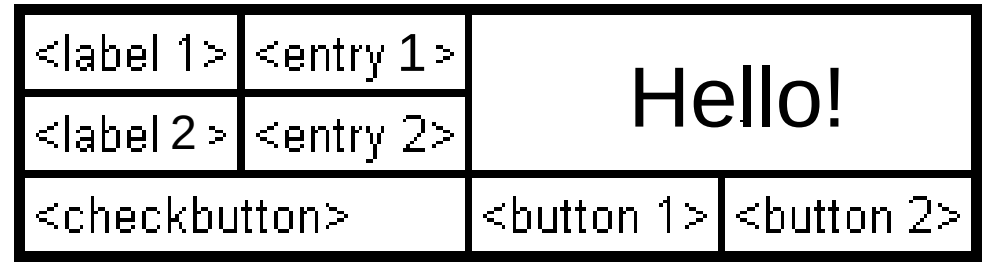


# Grid Geometry Manager

```
from tkinter import *

root = Tk()

l1 = Label(root, text="First")
l2 = Label(root, text="Second")

l1.grid(row=0, column=0, sticky = W)
l2.grid(row=1, column=0)

e1 = Entry(root)
e2 = Entry(root)

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

root.mainloop()
```



# Grid Geometry Manager




```
root = Tk()
label1 = Label(root, text="First")
label2 = Label(root, text="Second")
entry1 = Entry(root)
entry2 = Entry(root)
checkbutton = Checkbutton(root, text="Show title")
label3 = Label(root, text='Hello!')
ok_button = Button(root, text="OK")
cancel_button = Button(root, text="Cancel")

label1.grid(sticky=E) # by default column 0 of the next empty row
label2.grid(sticky=E) # can you guess row and column number of this guy?
entry1.grid(row=0, column=1)
entry2.grid(row=1, column=1)
checkbutton.grid(columnspan=2, sticky=W)
label3.grid(row=0, column=2, columnspan=2, rowspan=2,
            sticky=W+E+N+S, padx=5, pady=5)

ok_button.grid(row=2, column=2)
cancel_button.grid(row=2, column=3)
```

# Grid Geometry Manager

```
from tkinter import *

root = Tk()
colours = ['red','green','orange','white','yellow','blue']
r = 0


for c in colours:
    color = Label(root, text=c)
    color.grid(row=r,column=0, sticky=E)
    entry = Entry(root, bg=c)
    entry.grid(row=r,column=1, sticky=EW)

    r = r + 1
root.mainloop()
```

# Grid Geometry Manager
## Expand with Resize

```
from tkinter import *

root = Tk()
colours = ['red','green','orange','white','yellow','blue']
r = 0
Grid.columnconfigure(root, 0, weight=1)
Grid.columnconfigure(root, 1, weight=1)
for c in colours:
    color = Label(root, text=c)
    color.grid(row=r,column=0, sticky=E)
    entry = Entry(root, bg=c)
    entry.grid(row=r,column=1, sticky=EW)
    Grid.rowconfigure(root, r, weight=1)
    r = r + 1
root.mainloop()
```