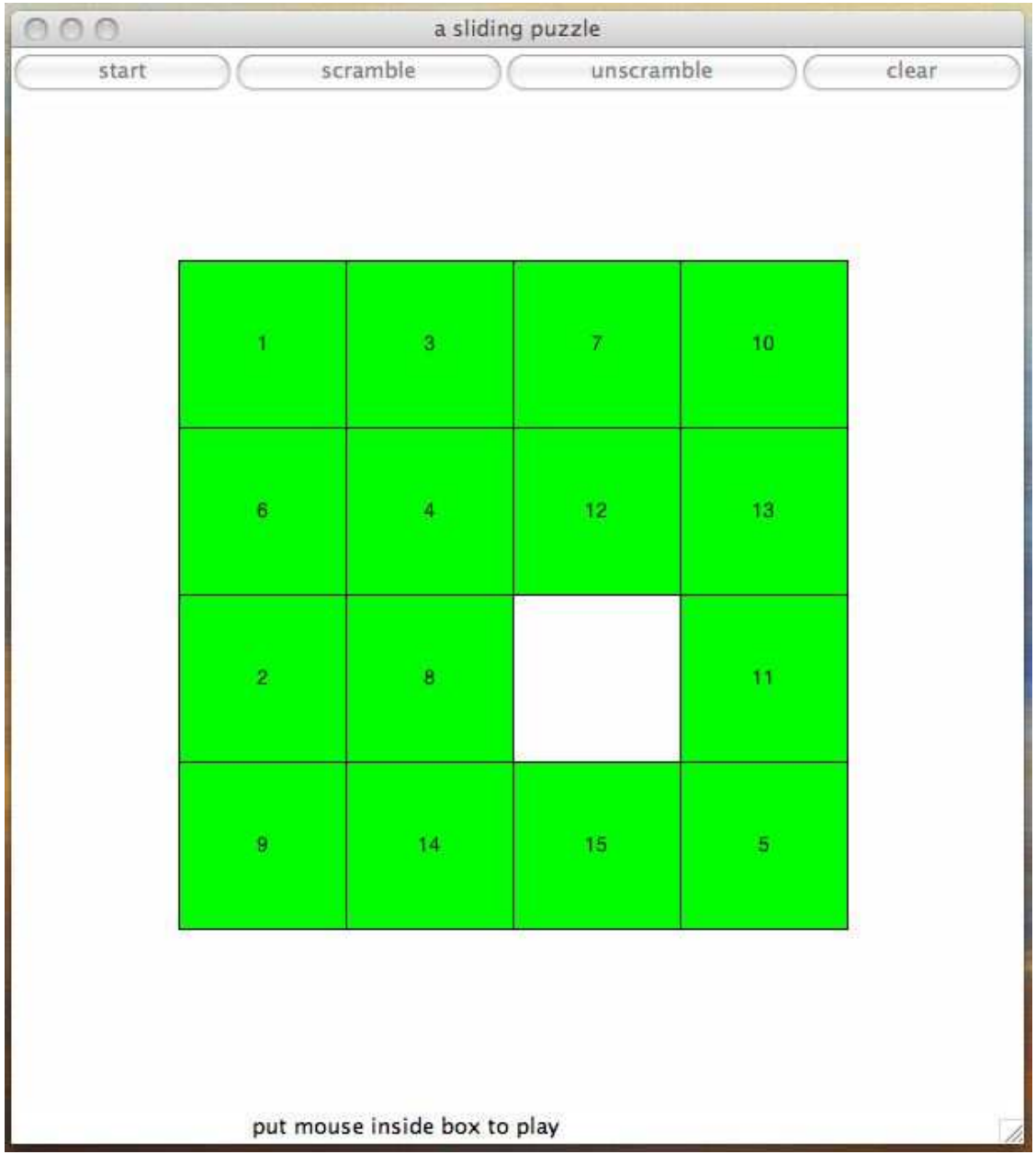
Outline

MCS 260 Lecture 30
Introduction to Computer Science
Jan Verschelde, 28 March 2016

graphical user interfaces using tkinter

A Sliding Puzzle



Elements of the GUI

The GUI for the sliding puzzle contains:

1. buttons to start, scramble, unscramble, and clear;
2. a canvas widget to display the sliding puzzle;
3. a list of lists stores the data of the puzzle;
4. the scramble/unscramble buttons are animations;
5. mouse events allow the user to solve the puzzle.

Functionality:

1. the start button of the GUI shows an ordered board;
2. the scramble button perturbs the current board;
3. pushing unscramble shows how to solve the puzzle;
4. the user can move rectangles by clicking on a rectangle that can slide to the free position.

Object oriented programming is used to implement GUIs.

graphical user interfaces using tkinter

User Interfaces (UI)

GUI, WUI, and HUI

Interfaces to our programs so far were command line.

Consequences:

1. strict linear control of execution order imposed on the user by program;
2. user must remember a lot, e.g.: file names.

Modern interfaces are mostly ***event driven*** and give the user control over the actions.

Command line interfaces are call non-GUI.
Specific categories of UI are
WUI: *Web* UI and HUI: *Handheld* UI.

Goal: increase *usability* of software.

Characteristics of GUIs

The most significant features of a GUI are

- **Window:** area of display device to view and interact with an object. Information is *viewed*.
- **Icon:** pictorial representation of an object.
- **Menu:** collection of choices, typically to perform actions on an object.
- **Pointer:** is similar to a typing cursor, controlled by a pointing device, typically a mouse.
- **Client Area:** area within a window where the user can enter data, strings or numbers.
- **Directness:** direct manipulation of objects via the pointer, such as moving and dragging windows.

The development of GUIs should be *user centered*.
Goals: simple, aesthetic, productive, and customizable.

GUI components

GUI components are *widgets* = window gadgets.

Some GUI components are

- **Button:** to trigger an event
- **Label:** to display text or icons
- **Entry:** to accept single line user input
- **Text:** to accept multiple line user input
- **Menu:** to display list of items
- **Listbox:** to display list of text options
- **Canvas:** to draw geometric figures

graphical user interfaces using tkinter

tkinter, Tk, and Tcl

GUIs for Python programmers

The `tkinter` (= `Tk` interface) library provides an object-oriented interface to the `Tk` GUI toolkit, the graphical interface development tool for `Tcl`, `Tk` = Tool Kit, `Tcl` = Tool Command Language.

Benefit: platform independent GUI development.

graphical user interfaces using tkinter

Hello World!

our first GUI

Five steps:

1. `from tkinter import Tk, Label`
2. create a new object of the class `Tk`
3. a label defines the text message
4. apply the geometry manager to the label
5. enter the main event loop

Until we close the window, the program stays in the loop.

the code for Hello World! in guihello.py

```
"""
Hello world with a Graphical User Interface.
The code below displays "Hello World!" in a
new window, using the tkinter GUI library.
"""
from tkinter import Tk, Label
TOP = Tk()          # TOP is the new window
# Label is a widget to design the interface
LBL = Label(TOP, text="Hello World!", \
            width=20, height=5)
# to arrange the widget in a window we call
LBL.pack()          # the geometry manager
TOP.mainloop()      # enter main event loop
```

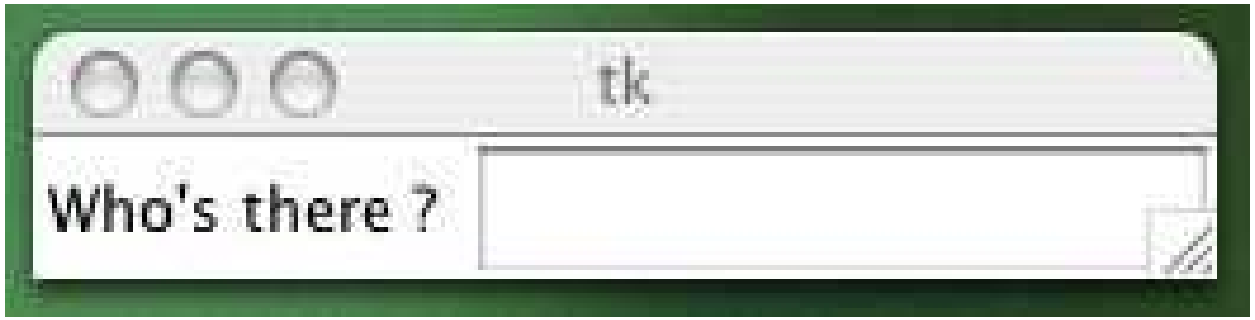
Dialogue with User – prompting for a name

We prompt the user to enter a name:

After the user typed in a name:

Pushing the `enter` button pops up the window:

the widget Entry to accept user input




```
from tkinter import Tk, Entry, Label

TOP = Tk()
Label(TOP, text="Who's there ? ").grid(row=0)
ENT = Entry(TOP)
ENT.grid(row=0, column=1)
TOP.mainloop()
```

Observe:

1. use of `row` and `column` with `grid`
2. the widget `Entry` to accept user strings

the widget Button to enter user input


```
from tkinter import Tk, Entry, Label, Button
from tkinter import messagebox

def hello():
    "opens a window to say hello"
    data = 'hello ' + ENT.get()
    messagebox.showinfo("enter", data)

BTT = Button(TOP, text="enter", command=hello)
BTT.grid(row=1, column=1)
TOP.mainloop()
```

1. `Button` will call `hello` when pressed
2. `hello` calls `showinfo` of `messagebox`

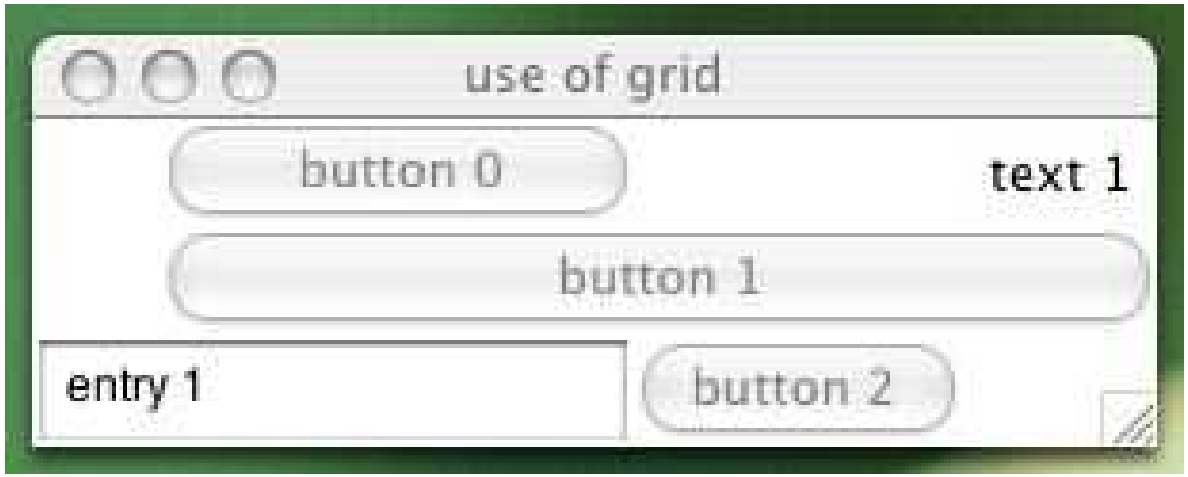
the complete script guihello2.py

```
from tkinter import Tk, Entry, Label, Button
from tkinter import messagebox

TOP = Tk()
Label(TOP, text="Who's there ? ").grid(row=0)
ENT = Entry(TOP)
ENT.grid(row=0, column=1)

def hello():
    "opens a window to say hello"
    data = 'hello ' + ENT.get()
    messagebox.showinfo("enter", data)

BTT = Button(TOP, text="enter", command=hello)
BTT.grid(row=1, column=1)
TOP.mainloop()
```

graphical user interfaces using tkinter

the Geometry Manager grid

To layout the geometry like this



Observe:

1. the title of the window
2. buttons spanning multiple columns
3. an entry field containing text

the geometry manager in the script `usegrid.py`

```
from tkinter import Tk, Label, Button, Entry
from tkinter import W, E, N, S, INSERT

TOP = Tk()
TOP.title("use of grid")
TX1 = Label(TOP, text=" text 1 ")
TX1.grid(row=0, column=4)
BT0 = Button(TOP, text=" button 0 ")
BT0.grid(row=0, column=1, sticky=W+E+N+S)
BT1 = Button(TOP, text=" button 1 ")
BT1.grid(row=1, column=1, columnspan=4, sticky=W+E+N+S)
EN1 = Entry(TOP)
EN1.insert(INSERT, "entry 1 ")
EN1.grid(row=2, column=0, columnspan=2)
BT2 = Button(TOP, text="button 2 ")
BT2.grid(row=2, column=3)
TOP.mainloop()
```

use of Radiobuttons

The GUI should do the following:

1. the entry field starts at 0
2. when user clicks on +1: add one
3. when user clicks on -1: subtract one

the script `guiradio.py` uses Radiobuttons

An Entry widget displays the value tuned by the Radiobuttons.

```
from tkinter import Tk, Entry, Radiobutton
from tkinter import INSERT, END

TOP = Tk()
TOP.title("use Radiobutton")
TEXT = Entry(TOP)
TEXT.insert(INSERT, "0")  # initialization
TEXT.grid(row=0, columnspan=2)

def plus():
    "Callback function, does +1"
    data = TEXT.get()           # data in Entry
    data = str(int(data) + 1) # add one to it
    TEXT.delete(0, END)         # clear Entry
    TEXT.insert(INSERT, data) # insert result

ADD = Radiobutton(TOP, text="+1", command=plus)
```

the script `guiradio.py` continued ...

The functions `plus()` and `minus()` are *callback functions*, called when the user touches the Radiobutton.

```
def minus():
    "Callback function does -1"
    data = TEXT.get()           # data in Entry
    data = str(int(data) - 1) # subtract one
    TEXT.delete(0, END)         # clear Entry
    TEXT.insert(INSERT, data) # insert result

SUB = Radiobutton(TOP, text="-1", command=minus)
ADD.grid(row=1, column=0)
SUB.grid(row=1, column=1)

TOP.mainloop()
```

Using CheckButtons

Functionality:

1. the user can check one or two boxes
2. click on the enter button
3. to see a message displayed in the Entry box

using Checkbutton in the `guicheck.py`

```
from tkinter import Tk, IntVar, Checkbutton, Entry, Button
from tkinter import W, E, N, S, INSERT, END

TOP = Tk()
TOP.title("use Checkbutton")
H = IntVar()  # determined by HOT button
C = IntVar()  # determined by COLD button

HOT = Checkbutton(TOP, text="hot", \
    variable = H, onvalue = 1, offvalue = 0)
HOT.grid(row=0, column=0)

COLD = Checkbutton(TOP, text="cold", \
    variable = C, onvalue = 1, offvalue = 0)
COLD.grid(row=0, column=1)
```

`H` and `C` are variables, toggled on or off by the user selecting the corresponding box.

code in `guicheck.py` continued...

```
ENT = Entry(TOP)
ENT.grid(row=2, columnspan=2)

def act():
    "callback function for enter button"
    ENT.delete(0, END)
    if H.get() == 1 and C.get() == 0:
        ENT.insert(INSERT, "it is hot")
    if H.get() == 0 and C.get() == 1:
        ENT.insert(INSERT, "it is cold")
    if H.get() == 1 and C.get() == 1:
        ENT.insert(INSERT, "it is hot and cold")

BTT = Button(TOP, text = "enter", command=act)
BTT.grid(row=1, columnspan=2, sticky=W+E+N+S)
TOP.mainloop()
```

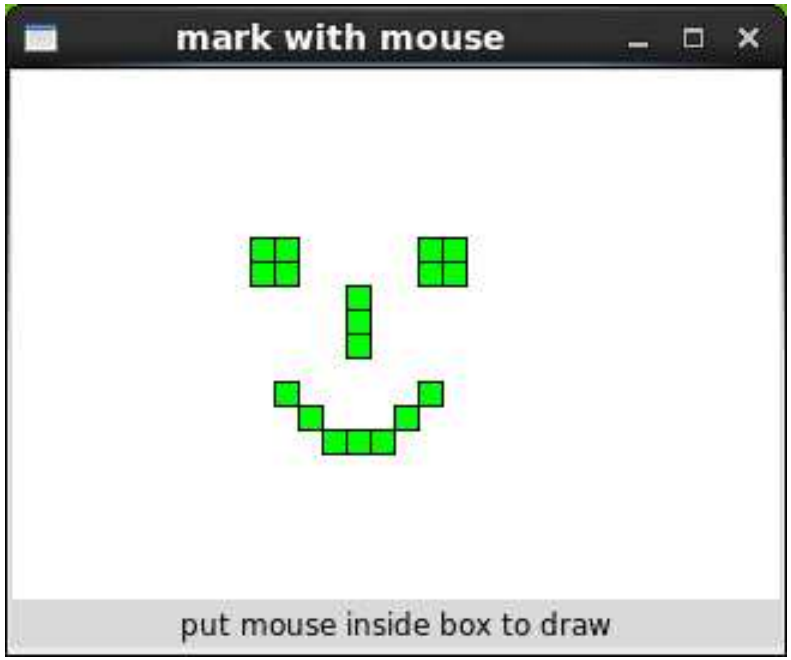
Observe the use of `H.get()` and `C.get()`.

graphical user interfaces using tkinter

handling mouse events

Often the amount of data we generate is too huge
for an orderly display in a classical terminal window.

Much more data can be stored in an image on canvas
and via the mouse we may interact with the data.



filling squares on a grid

```
from tkinter import Tk, Canvas, StringVar, Label

class FillSquares(object):
    """
    Filling squares on canvas with mouse clicks.
    """
    def __init__(self, wdw, r, c):
        """
        the mouse is bound to the canvas
        a label displays mouse position
        """
        wdw.title("mark with mouse")
        self.mag = 10    # magnification factor
        self.rows = r    # number of rows on canvas
        self.cols = c    # number of columns on canvas
```

__init__ continued

```
def __init__(self, wdw, r, c):
    ....
    self.cnv = Canvas(wdw, \
        width=self.mag*self.cols+2*self.mag, \
        height=self.mag*self.rows+2*self.mag, \
        bg='white')
    self.cnv.grid(row=1, column=0, columnspan=3)
    # to display mouse position :
    self.mouse_position = StringVar()
    self.mouse_position.set("put mouse inside box to draw")
    self.position_label = Label(wdw, \
        textvariable = self.mouse_position)
    self.position_label.grid(row=2, column=0, columnspan=3)
    # bind mouse events
    self.bind_mouse_events()
    self.filled = []
    for _ in range(r):
        self.filled.append([False for _ in range(c)])
```

binding the mouse events

To bind the mouse events to canvas:

```
def bind_mouse_events(self):
    """
    binds mouse events to the canvas
    """
    self.cnv.bind("<Button-1>", self.button_pressed)
    self.cnv.bind("<ButtonRelease-1>", self.button_released)
    self.cnv.bind("<Enter>", self.entered_window)
    self.cnv.bind("<Leave>", self.exited_window)
    self.cnv.bind("<B1-Motion>", self.mouse_dragged)
```

The methods `button_pressed`, `button_released`, `entered_window`, `exited_window`, and `mouse_dragged` are activated by the mouse.

button_pressed and button_released

```
def button_pressed(self, event):
    """
    Display coordinates of button press.
    """
    self.mouse_position.set("currently at [ " + \
        str(event.x) + ", " + str(event.y) + " ]" + \
        " release to fill, or drag")

def button_released(self, event):
    """
    display coordinates of button release
    """
    self.mouse_position.set("drawn at [ " + \
        str(event.x) + ", " + str(event.y) + " ]" + \
        " redo to clear")
    self.draw_rectangle(event.x, event.y)
```

the other methods

```
def entered_window(self, event):
    """
    Display message that mouse entered window.
    """
    self.mouse_position.set("press mouse to give coordinates")

def exited_window(self, event):
    """
    Display message that mouse exited window.
    """
    self.mouse_position.set("put mouse inside box to draw")

def mouse_dragged(self, event):
    """
    Display coordinates of moving mouse.
    """
    self.mouse_position.set("dragging at [ " + \
        str(event.x) + ", " + str(event.y) + " ]" + \
        " release to draw")
```

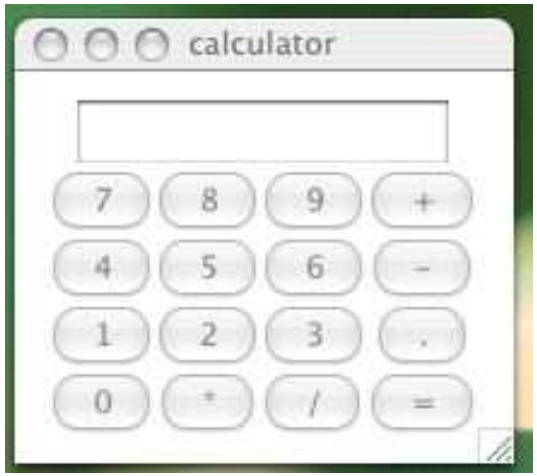
Summary + Assignments

For a manual on tkinter:

`http://infohost.nmt.edu/tcc/help/pubs/tkinter.pdf`

Assignments:

1. Write Python code to display:



   You should not provide any functionality.

2. Add functionality to the calculator shown above.
3. Design a GUI to convert temperatures between Fahrenheit and Celsius. Draw the layout and decide what widgets you will use.
4. Give Python code for the previous exercise.