PLTW COMPUTER SCIENCE

Activity 1.5.2

API for Tkinter Canvas

Introduction

Graphical user interfaces (GUIs) are made from widgets: buttons, sliders, and so on. The widgets are standard components of GUIs. Classes of widgets are created in code in a GUI toolkit. In this lesson you will create a GUI using the *Python*® programming language. We will use the Tkinter GUI toolkit, one of several GUI toolkits that are common across C++, *Python*, and other languages. One of the common widgets is a canvas. You use a canvas if you need to draw shapes.

Drawing on a GUI canvas is a little different than drawing shapes on images in the way you did in the last lesson. Canvas drawings are made out of objects. What's the difference between drawing with objects and drawing with pixels? The two circles at the right had identical shapes when they were smaller. The top circle is stored in this document as a circle. If enlarged, it will still be a smooth circle. The circle on the bottom is stored in this document as an image–an array of pixel values. At the size you are probably viewing, it is already pixelated!

Is the image data for a TV show stored pixel-by-pixel or object-by-object? What about the data for a video game? How can you tell?

Materials

- Computer with Enthought Canopy distribution of *Python*
- Graph paper with fine grid, for example 10 squares per inch

Resources

1.5.2.PY StudentSourceFiles.zip

Reference Card for Tkinter.docx

Procedure

ır partner to practice professional skills. Set team norms for pair programming.

anopy. Open an editor window. Set the working directory to your folder. The IPython onal.

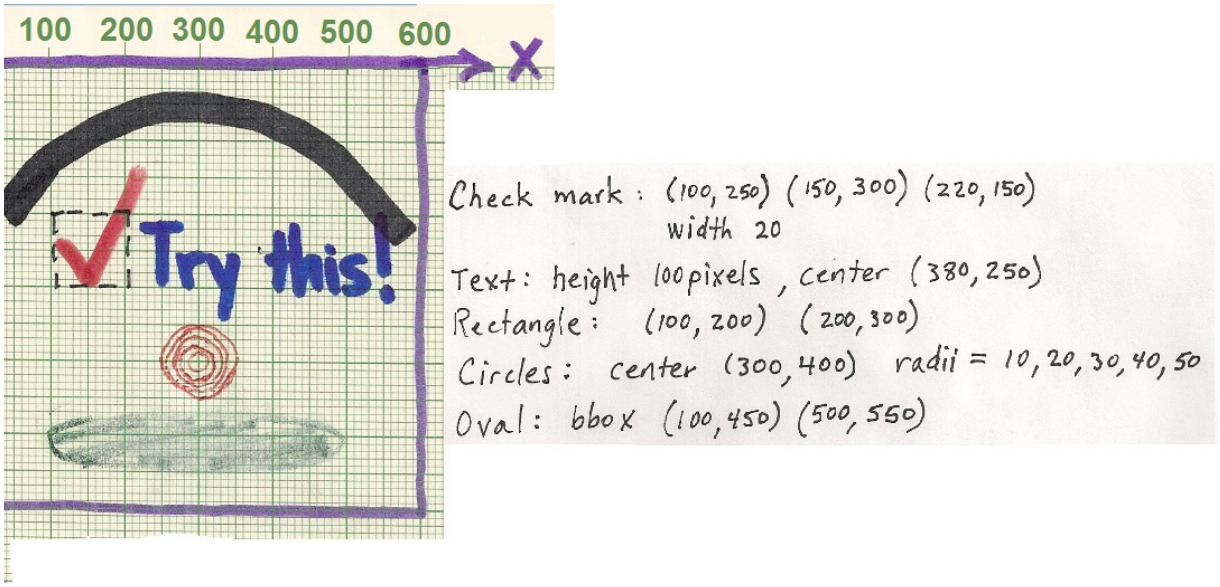
ıning *Python* programs with Tkinter, there is an additional step that is necessary. only one main GUI can be running, Canopy needs to be taken out of its interactive

ıe Canopy Welcome window, select Edit > Preferences... from the menu at the top. ıe Preferences dialog box that appears, select the ***Python*** tab. In the Python tab's dow, from the dropdown selection for **Pylab backend**, select **Inline (SVG)**. If you e already launched a code editor window, You will be alerted that the *Python* kernel st be restarted for the change to take effect; if that alert appears, select **Restart nel**.

er, if you want to return to the interactive mode for programs not using Tkinter, repeat above step but select **Interactive (Qt4)** for the **Pylab backend**.

to your downloadable resources for this material. Interactive ıt may not be available in the PDF edition of this course.

the canvas design shown on the graph paper below on the left. On the right, the worked out the design (except the arc) using a coordinate system. We will use these to implement the design using the methods of the `Canvas` widget.



ıg your pen or pencil on the graph above, plot the three points whose coordinates

given for the check mark.

designer's work indicates they think they have drawn the checkmark 20 pixels thick.
w a line across the width of the check mark to indicate what is meant by 20 pixels.

:e a dot at the point (380, 250) indicated as the center of the text. Is the actual center
ıe text further to the left or further to the right?

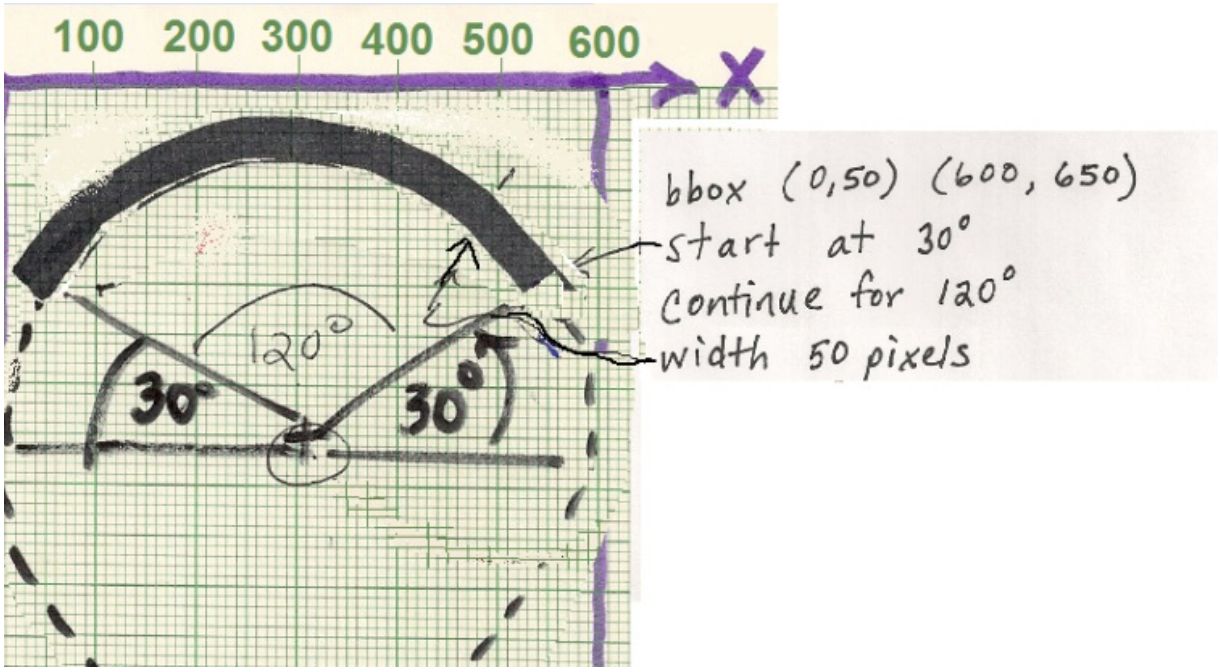
list of radii shown above for the concentric circles' radii (10, 20, 30, 40, 50) can be
luced with the `range (start, stop, step)` function. An example of `range()`
ıown here.

```
[]: range(10, 16, 2)
[]: [10, 12, 14]
```

at arguments for `range()` will produce the list of radii? Check your answer in the
hon session.

oval is specified by the coordinates of two points: the upper left and lower right of a
nding box. The bounding box circumscribes the circle and is specified with the (x,
oordinates of opposite vertices. The designer noted (100, 450) (500, 550) for the
's bounding box in the notes above. Sketch this bounding box in the figure above.
ch of these coordinates from the designer's notes is off by a little bit?

gner made a separate diagram, shown below, to work out the coordinates and angles
.



k the two points specified for the bounding box in the figure. Are these points
ectly identified?

“start” of the arc is given as the angle in standard position: counterclockwise from positive x direction. The “continue” is given as the central angle: the angle of the tor of the circle. Check your understanding of these terms by stating the starting le and the central angle size of this arc shown in red. Then compare your answer another team of two.

file `TkinterDraw.py`.

built with widgets like buttons and sliders. The program `TkinterDraw.py` is low. It creates a Canvas widget and then demonstrates some methods of the `.Canvas` API that add objects to the canvas.

```
inter import * # Don't import like this except for Tkinter
Tk() # Create main window

and place a Canvas widget for events and drawing
= Canvas(root, height=600, width=600, relief=RAISED, bg='white')
grid()# Puts the canvas in the main Tk window

four objects on the canvas
x = canvas.create_rectangle(100, 200, 200, 300, dash=[1,4])
 canvas.create_line(100, 250, 150, 300, 220, 150, fill='red', width=20)
 = canvas.create_text(380, 250, text='Try this!', font=('Arial', -100))
= canvas.create_oval(100,450,500,550, fill='#888888', outline='#888888')

an array of objects on the canvas
=[]
n range(10, 60, 10):
cles.append(canvas.create_oval(300-r, 400-r, 300+r, 400+r, outline='red'))

one more object on the canvas
= canvas.create_arc(0, 50, 600, 650, start=30, extent=120,
                    width=50, style=ARC)

 event loop. This displays the GUI and starts listening for events.
rogram ends when you close the window.
inloop()
```

the code. The output window should look like the one below and must be closed re the IPython prompt will come back.

your note taking convenience and for your future reference, the following list shows
ıe Canvas methods listed alphabetically:

eful `Canvas` methods

```
ıvas(window, width, height [, options] )
```
Constructs a new canvas object

```
_canvas.create_image(x1, y1, image, [, options] )
```
Creates an image by pasting it into the canvas

```
_canvas.create_arc(x1, y1, x2, y2 [, options] )
```

```
_canvas.create_line(x1,y1, x2,y2 [, …, xn,yn] [,
ıions])
```

```
_canvas.create_oval(x1, y1, x2, y2 [, options] )
```

```
_canvas.create_rectangle(x1, y1, x2, y2 [, options] )
```

```
_canvas.create_text(x1, y1, [, options] )
```

```
ıd()
```
Adds the canvas to the window's geometry manager

official documentation at http://effbot.org/tkinterbook/canvas.htm identifies
ons in keyword=value pairs. Using that documentation, identify what color the
ckmark in line 13 would be if the `fill='red'` keyword-value pair had not been
d.

third party documentation at
://infohost.nmt.edu/tcc/help/pubs/tkinter/web/fonts.html explains the 2-tuple
d in the `create_text()` method: `font=('Arial', -100)`. What does the
ative sign in -100 cause the method to do? What does the documentation say the
hod will do if the negative sign is omitted?

methods listed in part b above all return the value of the `item_id` attribute for the
ct being placed on the canvas. You have to retain these for later use. Notice that
h of these `item_id`s was stored in a variable. How many objects were created to put
he canvas?

the variables storing those `item_id` numbers:

- Tkinter doesn't give us direct access to the methods or attributes for those item objects enumerated in part e. But the `canvas` object knows about them. The `Canvas` class has methods that let us retrieve or set the attributes of the `canvas` items.

Methods to retrieve or set `Canvas` attributes

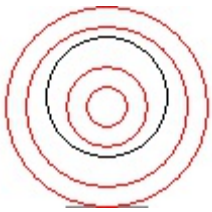
`my_canvas.coords(`*item*`)`
Gets the coordinates of the my_canvas item

`my_canvas.coords(`*item*`,` x_1`,` y_1`,` x_2`,` y_2 `)`
Sets the coordinates of the my_canvas item

`my_canvas.itemcget(`*item*`,` *keyword*`)`
Get one option's value for a canvas item

`my_canvas.itemconfig(`*item*`,` *keyword=value*`)`
Sets one option's value for a canvas item

Insert the lines of code as shown below. The code must occur before `mainloop()`. The code will alter the concentric circles so that the middle circle is black and moved up 5 pixels, as shown below.



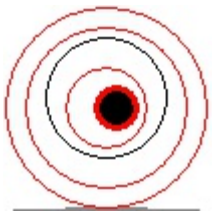
```
# Demonstrate changing a property of canvas' item.
```

```
canvas.itemconfig(circles[2], outline='black') # Change color
a, b, c, d = canvas.coords(circles[2]) # Get coordinates
canvas.coords(circles[2], a, b-5, c, d-5) # Change coordinates

# Enter event loop. This displays the GUI and starts listening
```

Why do you think the code had to be inserted before `mainloop()` to have an effect?

ert lines of code so that the inner circle is filled black, with a wider outline, and moved xels to the right:



nage to your canvas.

ert the following lines of code before the `mainloop()` and run it. Your line numbers ht be different.

```
et a filename in the same directory as this program
ort os.path
ectory = os.path.dirname(os.path.abspath(__file__))
ename = os.path.join(directory, 'canopyIcon.jpg')

pen the image file and convert to an ImageTk object
ort PIL.Image, PIL.ImageTk
 = PIL.Image.open(filename) # create a PIL.Image from the jpg i
mg = PIL.ImageTk.PhotoImage(img) # convert the PIL.Image to a I

dd the ImageTk object to the canvas.
n = canvas.create_image(tkimg, 150, 250)

nter event loop. This displays the GUI and starts listening for
```

code above will produce an error. Looking at the traceback, which line of code sed the error?

the official documentation for Tkinter's `Canvas` class (see Step 7c) to determine to fix the arguments in your function call. *Hint: the arguments were in the wrong er, and one argument needs a keyword=value syntax.*

official documentation for Tkinter includes more than the `Canvas` class. Refer to the

ial documentation at http://effbot.org/tkinterbook/tkinter-index.htm. At the bottom
art II: Class Reference, follow the link for the `PhotoImage` class. According to the
umentation, what types of image files can be created by the `PhotoImage` class
out help from the `PIL` library?

third-party documentation in Step 7d above is also part of a larger package of
umentation for the entire `Tkinter` library. Refer to
://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html and see Section 8.6 on
hods of the `Canvas` class. Read the explanation of the `tag_lower()` method.
dict what will result if you add lines 48-49 shown below. Then try it and finish the
ıment in line 48.

```
n = canvas.create_image(150, 250, image=tkimg)

o what?
vas.tag_lower(icon, check)

nter event loop. This displays the GUI and starts listening for
```

Conclusion

1. One objective of this assignment was to increase your comfort level using online documentation. Describe something you like about using the “real” documentation, and describe something that is challenging about using real documentation.

2. How is a GUI canvas different than an image? Specifically, how do you think the data that stores information about the canvas is different than data that would store a similar-looking rectangular image with an identical appearance but as an image? Consider Steps 7f, 7g, and 8e in thinking about your answer.