CSC 110: Week 9, Lecture 2

Dr. Bodine

Last time…

- Basics of graphics
- Graphics programming and OOP
- Custom Python graphics library
    - Objects: GraphWin, Point, Line, Circle, …
    - Methods: setFill, draw, …
- Basic drawing

This time…

- Continue our discussion of graphics and OOP
- Coordinates
- Adding text to our graphics
- Interactive graphics

Basic Graphics with custom library

- Start with a graphics window -> place to draw
  - GraphWin
- Create objects (instantiate the class!)
  - Example: Point, Circle, Line, …
- Set attributes for those objects
  - Example: color, x position, y position, …
- Draw the objects
  - Object method takes location to draw as a parameter

Coordinate transformations

- The GraphWin class has a coordinate transformation that allows you to redefine the x and y divisions as you see fit

```
mywindow.setCoords(xlowerleft, ylowerleft, xupperright, yupperright)
```

- For example, may want to divide the canvas up into four parts
  - `mywindow.setCoords(0,0,2,2)` will make the lower left be the origin (0,0) and the upper right be the point (2,2)
  - Center becomes (1,1)

Using coordinate transformations

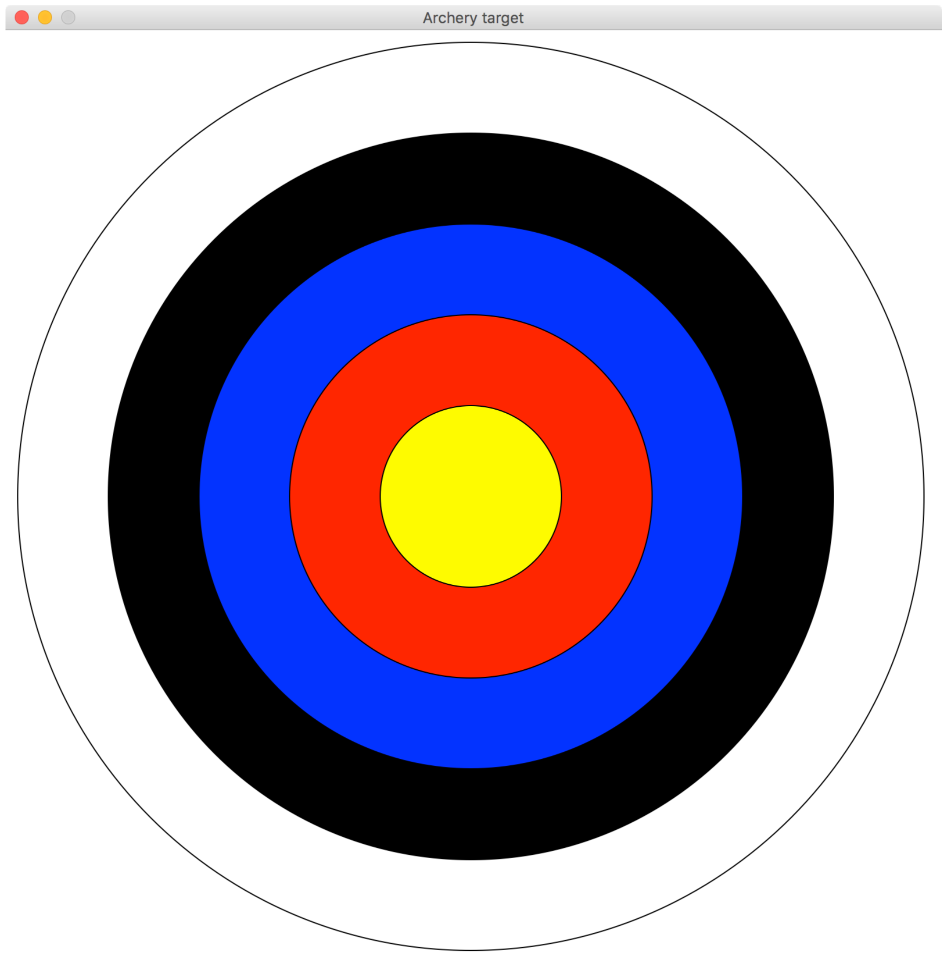
- Suppose we wanted to draw the archery target on a canvas that someone provided to us. We could do a coordinate transformation to and determine the appropriate proportions rather than size…

- In cases like a bar graph, coordinate transformations make it easier to do the accounting…

But what about when we aren’t graphing multiple objects…

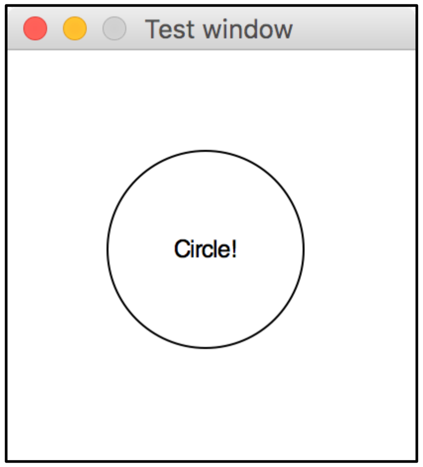
- Does it matter how we keep track of our coordinates?
- We can assign variables to keep track of our length and width so we can use it to assign the correct radius and center
- Or we can accept a canvas as a parameter and use methods to find the length and width so that we can draw on it appropriately….

```
>>> winwidth=800
>>> winlength=800
>>> archwin = graphics.GraphWin("Archery target",winwidth,winlength)
>>>
>>> center = graphics.Point(winwidth/2,winlength/2)

>>> radius = (winwidth-20)/10
>>>
>>> yellowcircle = graphics.Circle(center,radius)
>>> yellowcircle.setFill("yellow")
>>>
>>> redcircle = graphics.Circle(center,2*radius)
>>> redcircle.setFill("red")
>>>
>>> bluecircle = graphics.Circle(center,3*radius)
>>> bluecircle.setFill("blue")
>>>
>>> blackcircle = graphics.Circle(center,4*radius)
>>> blackcircle.setFill("black")
>>>
>>> whitecircle = graphics.Circle(center,5*radius)
>>>
>>> whitecircle.draw(archwin)
>>> blackcircle.draw(archwin)
>>> bluecircle.draw(archwin)
>>> redcircle.draw(archwin)
>>> yellowcircle.draw(archwin)
>>>
```



Printing text

```
>>> testwin = graphics.GraphWin("Test window",200,200)
>>> testwin.setCoords(0,0,2,2)
>>> graphics.Circle(graphics.Point(1,1),0.5).draw(testwin)
>>> graphics.Text(graphics.Point(1,1),"Circle!").draw(testwin)
```



What is the difference between:

```
>>>graphics.Text(graphics.Point(1,1),"Circle!").draw(testwin)
```

And the two line code:

```
>>> mytext = graphics.Text(graphics.Point(1,1),"Circle!")
>>> mytext.draw(testwin)
```

Discuss…what is the difference here:

```
mytext = Text(Point(x,y), <what you want to say>)
mytext.draw(<window>)
```

------------------- VS --

```
Text(Point(x,y),<what you want to say>).draw(<window>)
```

Interactive graphics

- We can get user input with our graphics window!
- "*Event-driven programming*"
    - An *event* is generated by user action such as clicking, moving the mouse, etc.
    - The *event* stores the input so it can be used elsewhere in the program

Interactive graphics basics

- Widgets
  - Interface elements drawn on the screen
  - Example: a box with two buttons
- Events
  - Objecst that encapsulate information about what happened
  - Example: button-event that captures which button the user clicked
- Results…
  - Rest of the program can access that information to do a larger task
  - Example: decide which algorithm to use based on which button the user clicked

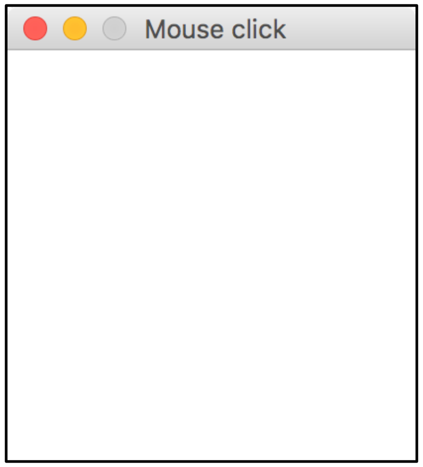
Mouse clicks

- getMouse : GraphWin method

- Pauses program
- Waits for user to click in window
- Returns the location of the click to the program as a Point

Mouse clicks example

```
from graphics import *

def main():
    win = GraphWin("Mouse click")
    win.setCoords(-1,-1,1,1)

    p = win.getMouse()

    print("You clicked at",p.getX(),p.getY())


main()
```



What kind of coordinates are returned?

Mouse clicks: prompting user

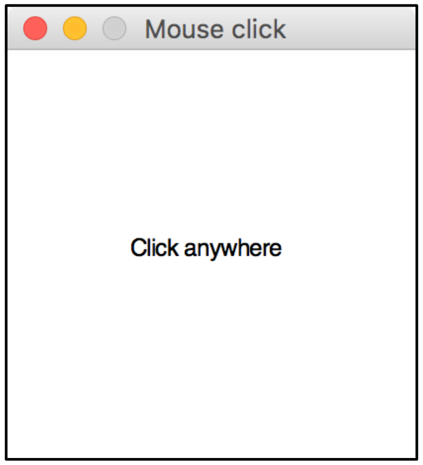
- Our little program didn’t really prompt the user to click…
    - Book example calls GraphWin “Click me!”

- Can we have the graphics prompt the user?
    - Use text?

- What happens to the text after the user has complied?

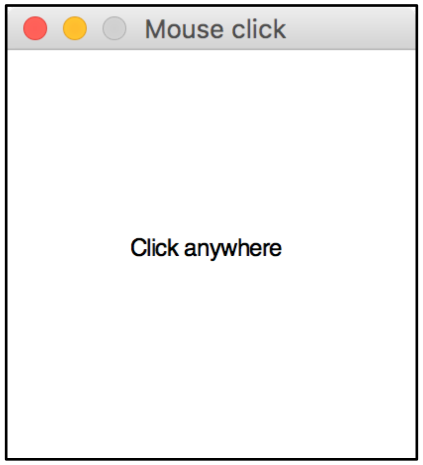
Adding a prompt

```
from graphics import *

def main():
    win = GraphWin("Mouse click")
    win.setCoords(-1,-1,1,1)

    message = Text(Point(0,0),"Click anywhere")
    message.draw(win)

    p = win.getMouse()

    print("You clicked at",p.getX(),p.getY())


main()
```



Adding a prompt/message

```
from graphics import *

def main():
    win = GraphWin("Mouse click")
    win.setCoords(-1,-1,1,1)

    message = Text(Point(0,0),"Click anywhere")
    message.draw(win)

    p = win.getMouse()

    print("You clicked at",p.getX(),p.getY())


main()
```

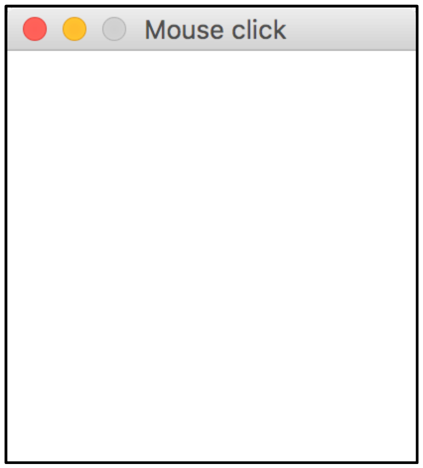


Does the graphics window change when the user clicks?

Updating prompt/message

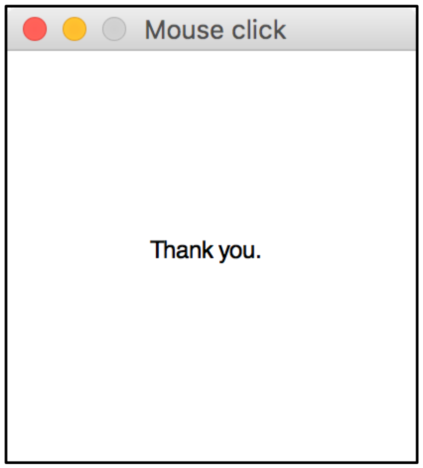
- We can update the message or prompt after the fact!

- First, we could undraw the text:
  - message.undraw()



Updating prompt/message

- We can update the message or prompt after the fact!
- First, we could undraw the text:
    - message.undraw()
- Or we could change the text:
    - message.setText(“Thank you.”)



Discuss

- What is the significance of using the name message for our text variable?
- Are there other valid choices of text variable names?
- Can you think of a limitation to the utility of this name?

We didn’t do anything close our windows…

- Typically want to wait for the user to signal the end.

- Example:
  - message.setText(“Click anywhere to exit”)
  - win.getMouse
  - win.close()

- Is win.close() really necessary?

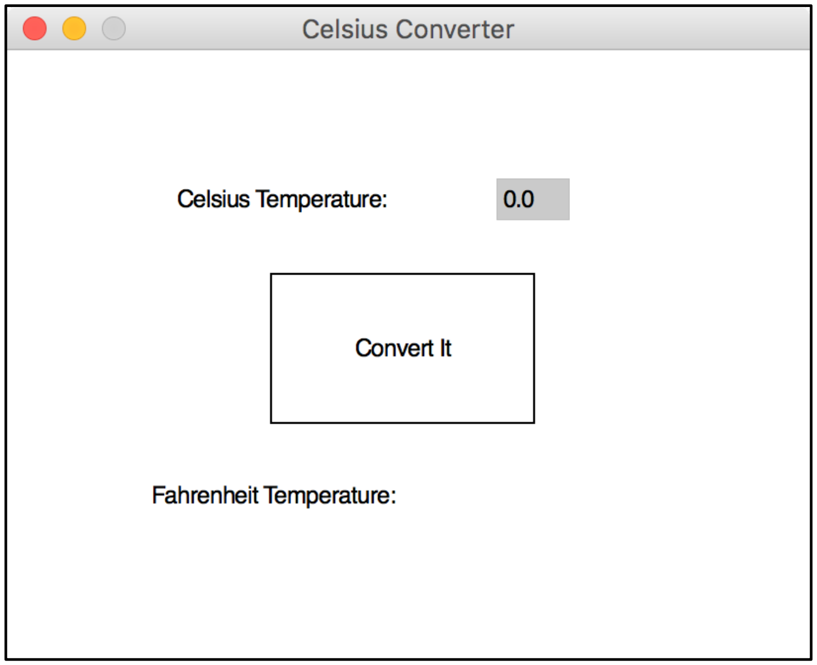
Interactive graphics with textual input

- Entry object
- Draws a box on the screen where the user can input text
- setText
- getText

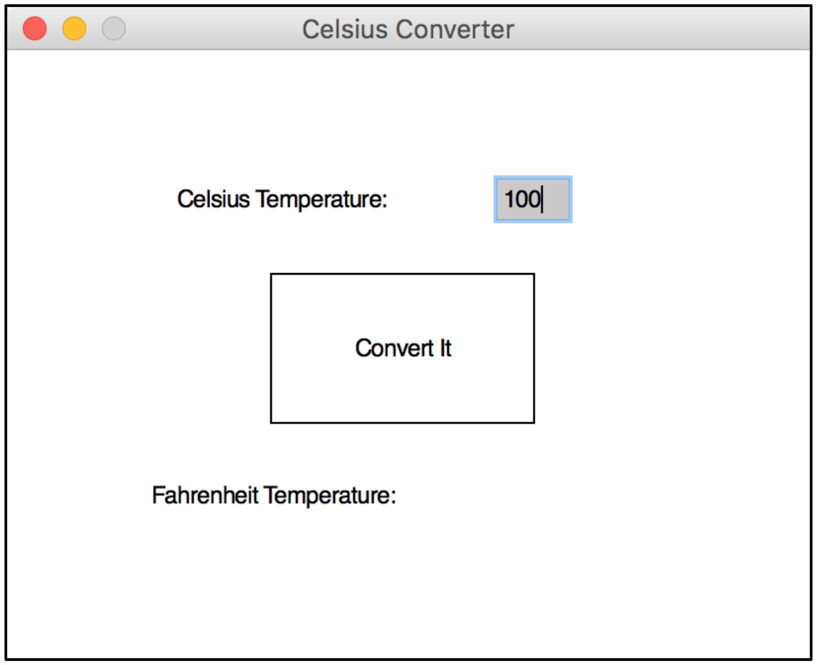
Example of textual input

- Book uses an example of the converter program from before…
- Makes a window where the user can enter the Celsius temperature and get the Fahrenheit temperature back
- Let’s look at this…
  - Uploaded a file: converter.py into this week’s Canvas module.

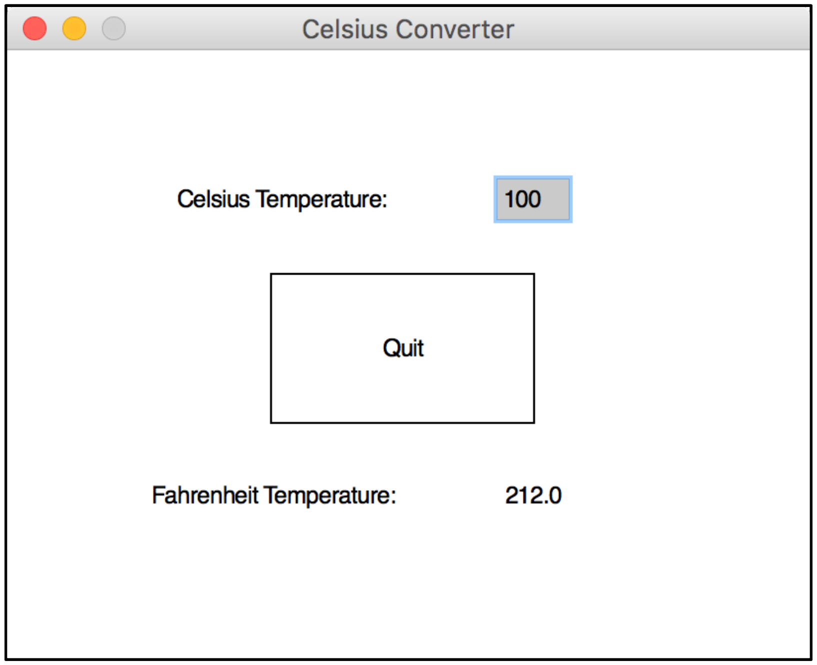
```python
def main():
    win = GraphWin("Celsius Converter",400,300)
    win.setCoords(0.0,0.0,3.0,4.0)

    #Draw interface
    Text(Point(1,3),"    Celsius Temperature:").draw(win)
    Text(Point(1,1)," Fahrenheit Temperature:").draw(win)
    input = Entry(Point(2,3), 5)
    input.setText("0.0")
    input.draw(win)
    output = Text(Point(2,1),"")
    output.draw(win)
    button = Text(Point(1.5,2.0),"Convert It")
    button.draw(win)
    Rectangle(Point(1,1.5), Point(2,2.5)).draw(win)

    #Wait for mouse click
    win.getMouse()

    #convert input
    celsius = eval(input.getText())
    fahrenheit = 9.0/5.0*celsius+32

    #display output and change button
    output.setText(fahrenheit)
    button.setText("Quit")

    #wait for click and then quit
    win.getMouse()
    win.close()

main()
```



```
def main():
    win = GraphWin("Celsius Converter",400,300)
    win.setCoords(0.0,0.0,3.0,4.0)

    #Draw interface
    Text(Point(1,3),"    Celsius Temperature:").draw(win)
    Text(Point(1,1)," Fahrenheit Temperature:").draw(win)
    input = Entry(Point(2,3), 5)
    input.setText("0.0")
    input.draw(win)
    output = Text(Point(2,1),"")
    output.draw(win)
    button = Text(Point(1.5,2.0),"Convert It")
    button.draw(win)
    Rectangle(Point(1,1.5), Point(2,2.5)).draw(win)

    #Wait for mouse click
    win.getMouse()

    #convert input
    celsius = eval(input.getText())
    fahrenheit = 9.0/5.0*celsius+32

    #display output and change button
    output.setText(fahrenheit)
    button.setText("Quit")

    #wait for click and then quit
    win.getMouse()
    win.close()

main()
```



```
def main():
    win = GraphWin("Celsius Converter",400,300)
    win.setCoords(0.0,0.0,3.0,4.0)

    #Draw interface
    Text(Point(1,3),"    Celsius Temperature:").draw(win)
    Text(Point(1,1)," Fahrenheit Temperature:").draw(win)
    input = Entry(Point(2,3), 5)
    input.setText("0.0")
    input.draw(win)
    output = Text(Point(2,1),"")
    output.draw(win)
    button = Text(Point(1.5,2.0),"Convert It")
    button.draw(win)
    Rectangle(Point(1,1.5), Point(2,2.5)).draw(win)

    #Wait for mouse click
    win.getMouse()

    #convert input
    celsius = eval(input.getText())
    fahrenheit = 9.0/5.0*celsius+32

    #display output and change button
    output.setText(fahrenheit)
    button.setText("Quit")

    #wait for click and then quit
    win.getMouse()
    win.close()

main()
```



Celsius Converter

Celsius Temperature: 100

Quit

Fahrenheit Temperature: 212.0

Questions about converter program

- Is it really a button? What happens if we click outside the button?
    - Text and rectangle are drawn separately…does order matter here?
- What happens if the person accidentally clicks before entering text?
- What happens when the person clicks at the end?
    - And if they don’t click?

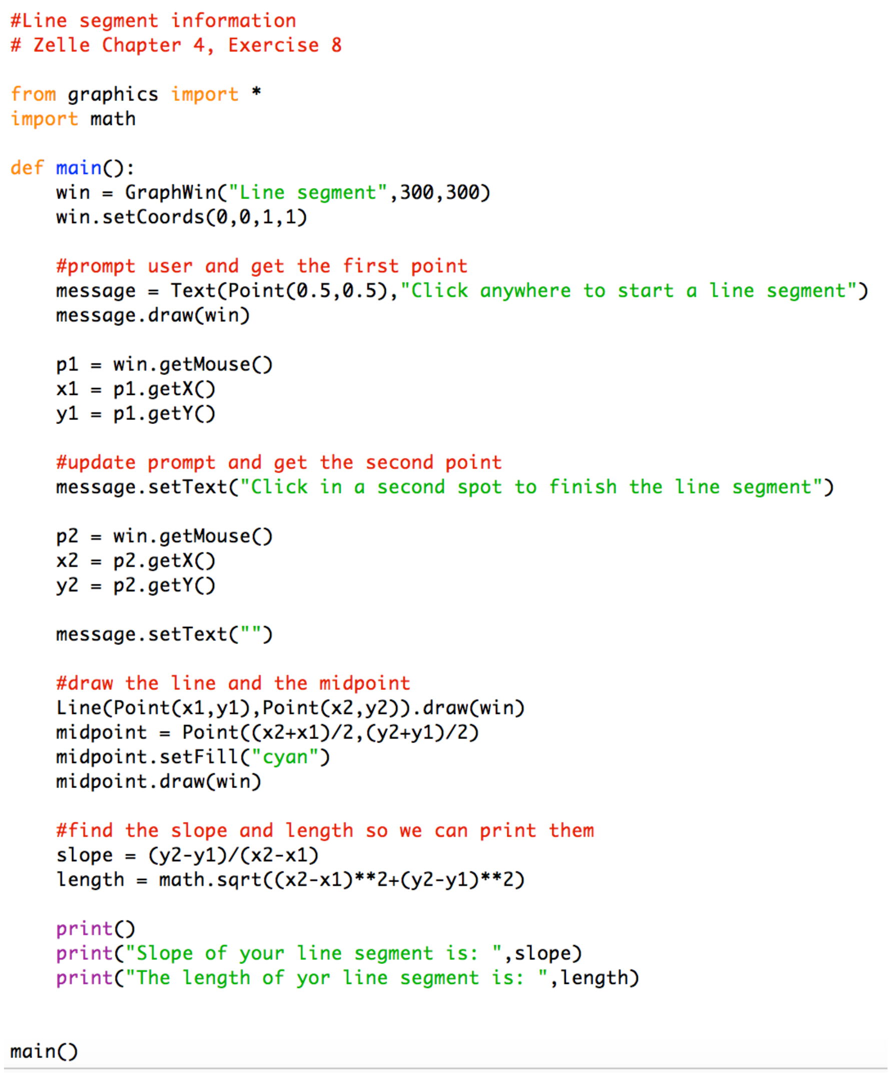
Let’s do an example!

- Zelle Chapter 4, Exercise 8: Line Segment Information.
- This program allows the user to draw a line segment and then displays some graphical and textual information about the line segment

- Input: Two mouse clicks for the end points of the line segment

- Output: Draw the midpoint of the segment in cyan.
  Draw the line.
  Print the length and slope of the line (segment)

- Formulas: $dx = x_2 – x_1$
  $dy = y_2 – y_1$
  slope = dy/dx
  length = sqrt(dx^2+dy^2)



```
#Line segment information
# Zelle Chapter 4, Exercise 8

from graphics import *
import math

def main():
    win = GraphWin("Line segment",300,300)
    win.setCoords(0,0,1,1)

    #prompt user and get the first point
    message = Text(Point(0.5,0.5),"Click anywhere to start a line segment")
    message.draw(win)

    p1 = win.getMouse()
    x1 = p1.getX()
    y1 = p1.getY()

    #update prompt and get the second point
    message.setText("Click in a second spot to finish the line segment")

    p2 = win.getMouse()
    x2 = p2.getX()
    y2 = p2.getY()

    message.setText("")

    #draw the line and the midpoint
    Line(Point(x1,y1),Point(x2,y2)).draw(win)
    midpoint = Point((x2+x1)/2,(y2+y1)/2)
    midpoint.setFill("cyan")
    midpoint.draw(win)

    #find the slope and length so we can print them
    slope = (y2-y1)/(x2-x1)
    length = math.sqrt((x2-x1)**2+(y2-y1)**2)

    print()
    print("Slope of your line segment is: ",slope)
    print("The length of yor line segment is: ",length)


main()
```

- This used a coordinate transformation…why?
- What would happen if we didn’t?
- Would the slope make sense?
- Would the length make sense?
- How do we know what coordinates to use?

Next week…

- Chapter 9: Simulation and Design
- Chapter 12: OOP Design