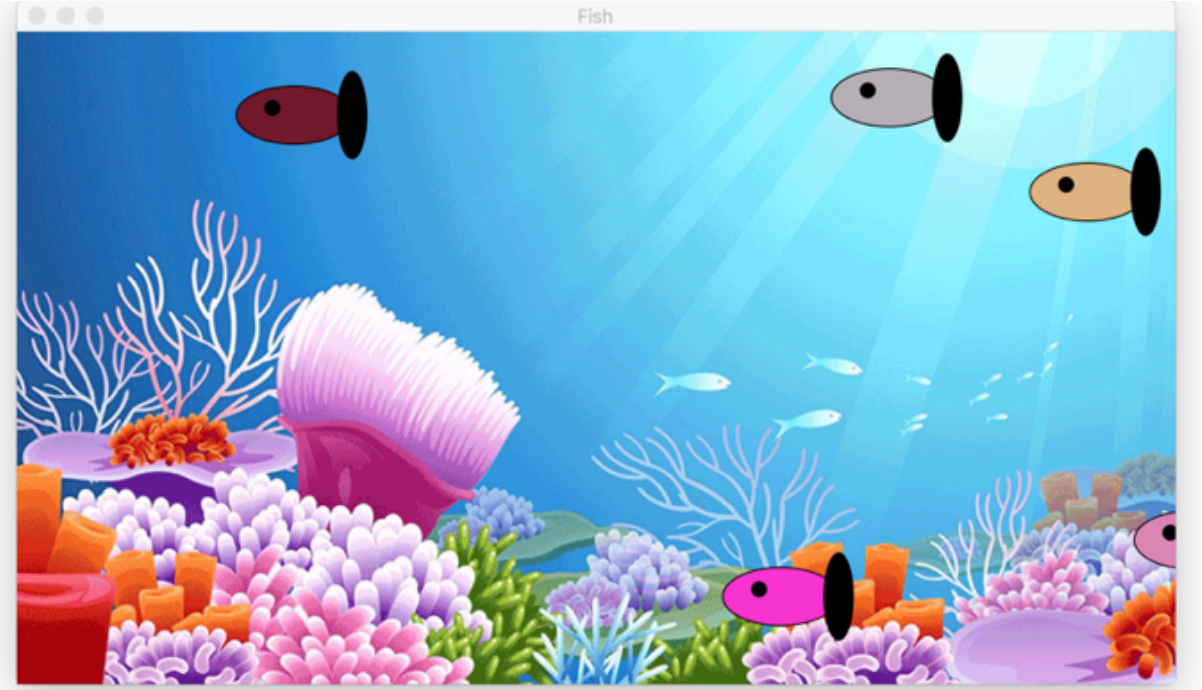
Topic: Graphics and Animation

Goals: By the end of this topic, we will discuss…

- *Graphics: quick demo, the python modules, drawing pictures with graphics*
- *Animation: understanding motion, .move(), keeping objects on the screen*

Acknowledgements: These class notes build on the content of my previous courses as well as the work of R. Jordan Crouser, and Jeffrey S. Castrucci.

Fish Demo



Discussion:
How do you think they built that?

What components did they need?

Components of a Game:

1. Draw elements
2. Make it move
3. Get input from the user and react

Note: If these are the basic components of every game, it's probably the case that someone else has had to build them before…

Drawing Elements

Elements:

- Points
- Lines
- Areas

The graphics module*

Two kinds of objects:
- stuff you draw (Graphics objects)
- stuff you draw on (GraphWin objects)

Basic formula for drawing graphics:
- open a graphic window (a GraphWin)
- construct some Point, Line, Circle, Oval, Rectangle, Polygon, and Text objects
- draw them to the window
- close the window when you're done
- terminate the program

*written by John Zelle to go along with his book
"Python Programming: An Introduction to Computer Science"
(Franklin, Beedle & Associates)
Available from: http://mcsp.wartburg.edu/zelle/python/

```
from graphics import *

def main():
    # 1. Build a window
    win = GraphWin("CSC111 - Graphics Demo", 600, 400)

    # 2. Define and draw a circle
    c = Circle(Point(50,50), 10)
    c.draw(win)

    # 3. Wait for user click, then close window
    win.getMouse()
    win.close()

if __name__ == "__main__":
    main()
```

Notes:
- import the module with * (reminder: this way we don't have to type "graphics" every time)
- build a GraphWin object (pass in the width and height as variables)
- construct a circle object (centered at (50,50) with a radius of 10)
- c.draw(win) actually draws the circle in the window
- win.getMouse() waits for the user to click the mouse
- win.close() closes the GraphWin

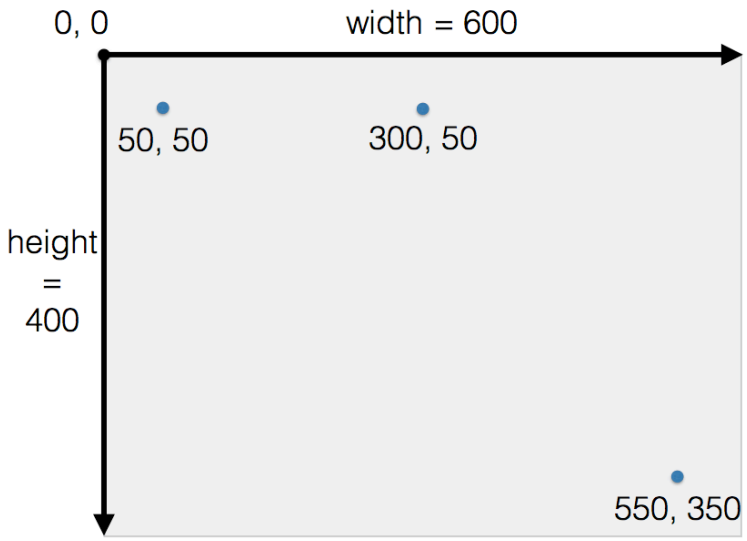
Discussion: How would we change this code to create a circle in the center of the window?

Points

- Used to anchor other objects (circles or rectangles)
- Defined by x and y coordinates

```
# create a point at
# location (50,50)
p1 = Point(50,50)

# create a point at
# location (300,50)
p2 = Point(300,50)
```



Circles

- Defined by a center and a radius
- The center is a Point

```
# create a circle centered at (50,50) with radius 70
c = Circle(Point(50,50), 70)
c.draw(win)
```

Rectangles

- Defined by a top-left, and a bottom-right point

```
# create a rectangle with top-left corner at (5,5) and
# bottom-right at (50,50)
r3 = Rectangle(Point(5,5), Point(50,50))
r3.draw(win)
```

Filling an object with color

```
# create a rectangle with top-left corner at (5,5) and
# bottom-right at (50,50)
r3 = Rectangle(Point(5,5), Point(50,50))
r3.setFill("red")
r3.draw(win)
```

Note: Color can be specified by RGB values:

```
my_color = color_rgb(200,100,150)
r3.setFill(my_color)
```

Oval

- Defined by a top-left, and a bottom-right point

```
# create a rectangle with top-left corner at (5,5) and
# bottom-right at (50,50)
r4 = Oval(Point(5,5), Point(50,50))
r4.draw(win)
```

Activity: Drawing a Fish

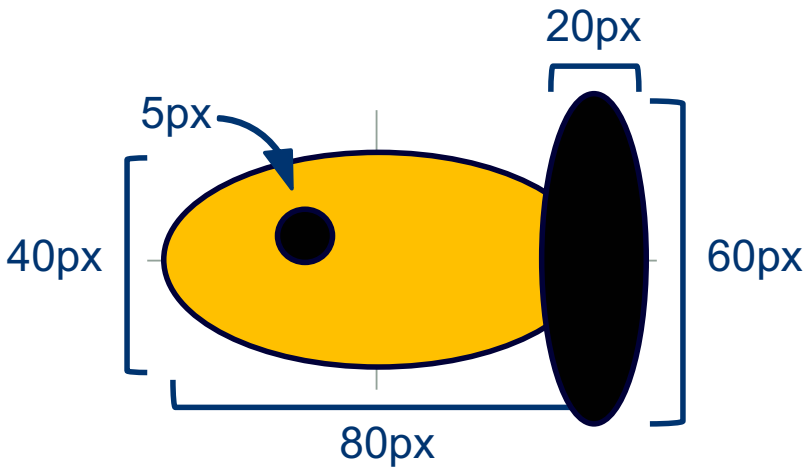


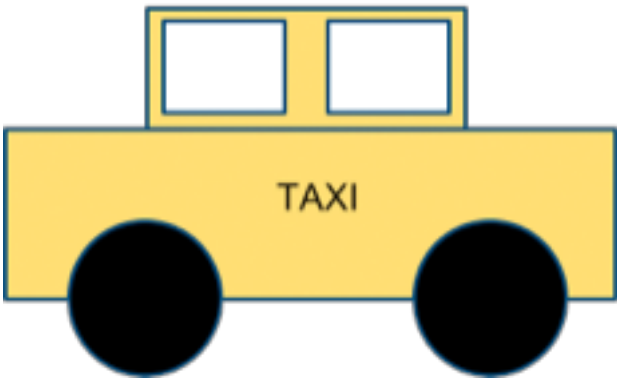
Figure out how to draw a fish.
How can we get all of these parts to work together?

Check out the graphics documentation:
https://mcsp.wartburg.edu//zelle/python/graphics/graphics/index.html

Recall: Graphics

Elements:

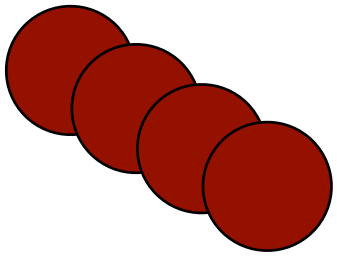
- Points
- Lines
- Areas



Animation

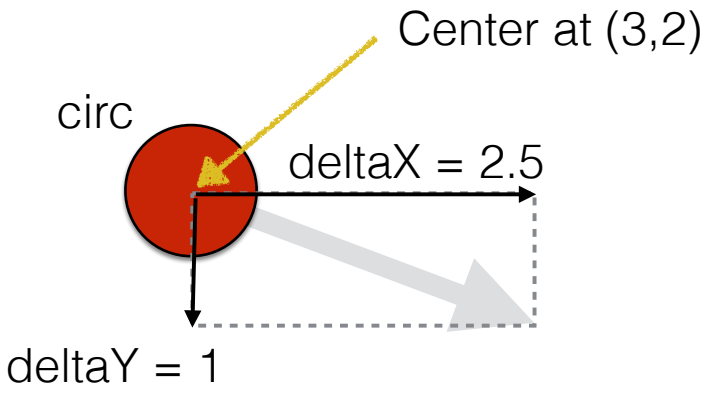
How do we make elements move? Graphics: http://mcsp.wartburg.edu/zelle/python/

Before video...
Flipbook LINE: https://youtu.be/9wN2RcAPhqA?t=337
Flipbook Dancing: https://youtu.be/7n2YF7mfP5s?t=6

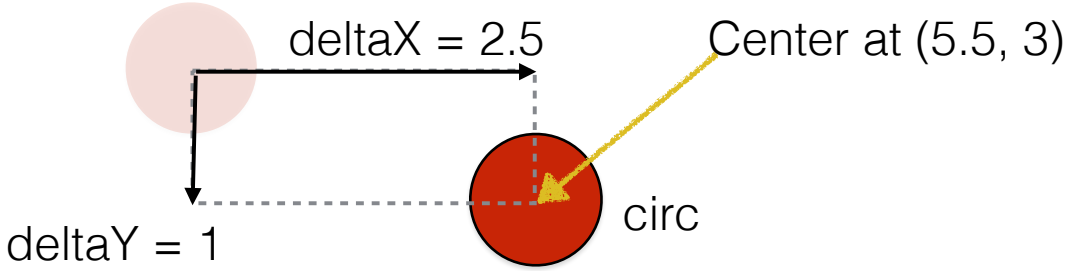


Discussion: What do I need to be able do to make that happen?

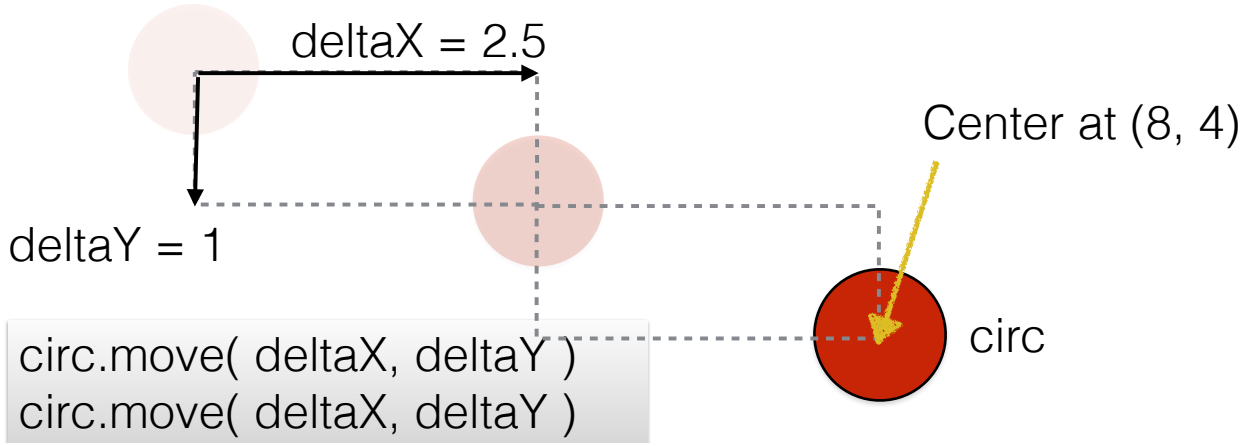
Understanding Motion




```
circ.move(deltaX, deltaY)
```




```
circ.move(deltaX, deltaY)
circ.move(deltaX, deltaY)
```



Basic organization of animation main()

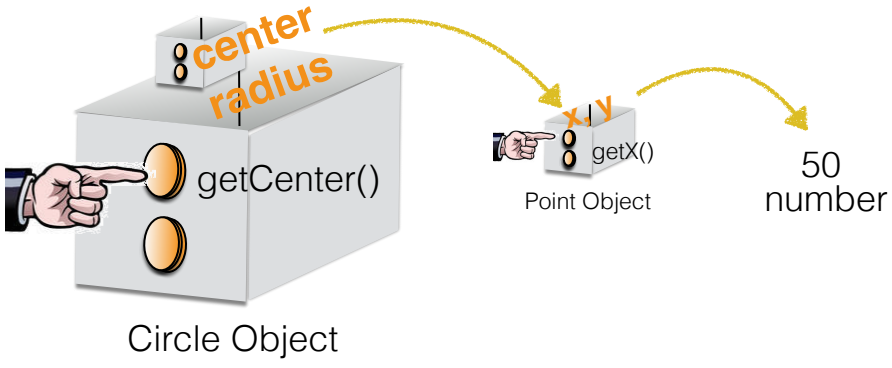
```
def main():
     # 1. open the graphics window
     # 2. define/initialize graphic objects
     # 3. start animation loop, stop on
     #    specific user interaction
     while win.checkMouse() == None:
          # 4. move/update each object
     # Loop is over.
     # 5. close the graphic window
```

Our first animation program...

```
from graphics import *
def main():
  win = GraphWin("My Circle", 600, 400)
  c = Circle(Point(50,50), 10)
  c.draw(win)
  for i in range(150):
    c.move(3,2)      # The move function takes two values, dx and dy
  win.getMouse()     # Pause to view result
  win.close()        # Close window when done
main()
```

Recall: Every graphical element is an object.

```
x = circ.getCenter().getX()
```



Discussion

How do we keep an object from moving off the screen?

Demo: Bouncing Ball

Introduction: Interaction Basics

- mouse
- keyboard

Recall: User Input

The `.input()` function

- Python has a built-in `.input()` function that allows us to ask the user to type in information
- The `.input()` function takes in a value, which will be printed to the console as a prompt:

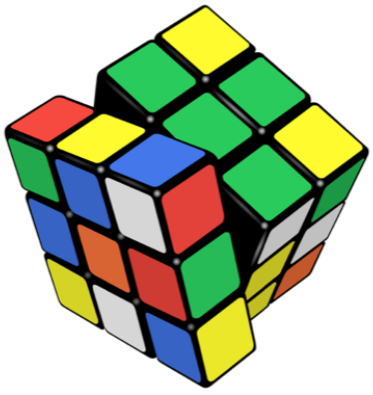
```
input("Enter some text: ")
```

Interaction (Definition)

- Ways for the user to affect change in what's happening in the program
- Low level: between human and interface
  - the set of operations available
  - happens between the human and the physical computer
- High level: between human and problem space
  - a cognitive act enabled by the interface
  - happens between the human and the digital objects

Discussion: Rubik's Cube

- What low-level interactions can you have?
- What high-level interactions can you have?

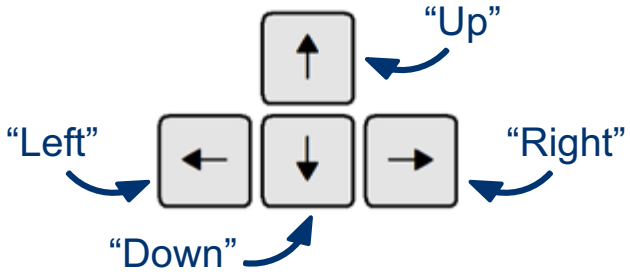


Interaction with graphics objects

- The `GraphWin` object has methods to detect interactions
- Mouse:
  - `.getMouse()`: stop the program and wait for user to click
  - `.checkMouse()`: continuously check if the user has clicked
  - both return a `Point` object
- Keyboard:
  - `.getKey()`: stop the program and wait for user to type
  - `.checkKey()`: continuously check if the user has typed
  - both return a *string*

Notes about keyboard interaction

The strings returned by the .getKey() / .checkKey() methods are called keycodes
Some keys don't have an obvious letter attached to them, but their keycodes are still pretty intuitive, e.g.



See also: "space", "Escape", "minus", "underscore", "equal", "plus", "BackSpace", "Return", etc.

Our first interactive graphics program: Animate Ball

```
    # Animate ball
    while (win.checkMouse() == None):
        keyPressed = win.getKey()
        print(keyPressed)
        if keyPressed == "Left":
            ball.move(-1,0)
        elif keyPressed == "Right":
            ball.move(1,0)
        elif keyPressed == "Up":
            ball.move(0,-1)
        elif keyPressed == "Down":
            ball.move(0,1)
```