CSC 111:
Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson

Admin

- **Lab 11** this week on recursion and sorting (last lab, short)
- **Final project** is due May 3 (Wednesday)
- TA hours this week: help with final project (Wed/Thurs)
- TA hours next week: only final review help (Sun-Wed)
- TA review session: next Wed (May 3)
- Self-scheduled **final exam** (similar style to the midterm)
- **Office hours tomorrow: 10am-12pm** in Ford 015

Outline: 4/26

- Recap dice roller
- Go over Homework 9
- Intro to sort and search (last topic)
- Next week: review

Recap Dice Roller

set_value method in DieView class

```
def set_value(self, value):
    """Change which pips are the foreground color to simulate
    rolling a different number (value should be one of: 1,2,3,4,5,6)."""

    # first set ALL the pips back to the background color
    for pip in self.pip_lst:
        pip.setFill(self.background)

    # based on the value, define a custom list of pips to be "on"
    if value == 1:
        pips_on = [3]
    elif value == 2:
        pips_on = [0,6]
    elif value == 3:
        pips_on = [0,3,6]
    elif value == 4:
        pips_on = [0,2,4,6]
    elif value == 5:
        pips_on = [0,2,3,4,6]
    elif value == 6:
        pips_on = [0,1,2,4,5,6]

    # for each of the "on" pips, change its color to the foreground
    for i in pips_on:
        self.pip_lst[i].setFill(self.foreground)
```

DieView constructor

```
def __init__(self, x, y, size):
    """DieView constructor: make a square of the given size and 7 pips."""

    # define a few instance variables that we'll use throughout
    self.background = "yellow"
    self.foreground = "purple"
    self.pip_size = 6

    # make a square for the die outline
    p1 = Point(x-size/2,y-size/2)
    p2 = Point(x+size/2,y+size/2)
    self.square = Rectangle(p1, p2)
    self.square.setFill(self.background)

    # draw all pips
    offset = size/4
    pip0 = self.make_pip(x-offset, y-offset)
    pip1 = self.make_pip(x-offset, y)
    pip2 = self.make_pip(x-offset, y+offset)
    pip3 = self.make_pip(x, y)
    pip4 = self.make_pip(x+offset, y-offset)
    pip5 = self.make_pip(x+offset, y)
    pip6 = self.make_pip(x+offset, y+offset)

    # create an instance variable for the list of pips
    self.pip_lst = [pip0, pip1, pip2, pip3, pip4, pip5, pip6]

    # choose a random value to start (call helper method!)
    self.set_value(random.randint(1,6))
```

Recap Homework 9

Fish constructor (Homework 9)

```
def __init__(self, x, y, color, speed, count):
    """Fish constructor: set up tail, body, and eye."""

    # set up the tail as an Oval
    p1 = Point(x-5,y-30)
    p2 = Point(x+5,y+30)
    tail = Oval(p1,p2)
    tail.setFill(color)

    # set up the body as an Oval
    p1 = Point(x-30,y-18)
    p2 = Point(x+30,y+18)
    body = Oval(p1,p2)
    body.setFill(color)

    # set up the eye as a Circle
    eye = Circle(Point(x,y-5), 3)
    eye.setFill("white")

    # based on the speed, move the eye and tail
    if speed < 0:
        eye.move(-15,0)
        tail.move(15,0)
    else:
        eye.move(15,0)
        tail.move(-15,0)

    # four instance variables
    self.x = x # keep track of the x-position
    self.shape_lst = [tail,body,eye] # order important!
    self.speed = speed # each fish has their own speed
    self.count = count # each fish has their own bob count
```

Fish move method (Homework 9)

```
def move(self, width):
    """Move each shape of the fish, including wrapping and bobbing."""
    dx = self.speed # default: move the fish at their speed

    # wrapping left case
    if self.x < 0:
        dx = width

    # wrapping right case
    elif self.x > width:
        dx = -width

    # bobbing: move up or down based on the count
    dy = 1
    if self.count % 20 < 10:
        dy = -1
    self.count += 1

    # use a for loop to move each shape the desired amount
    for shape in self.shape_lst:
        shape.move(dx,dy)
    self.x += dx # update x position (very important!)
```

Final project recommendation

- To make your animation more smooth, turn off autoflush and update within the animation loop as shown below:

```
win = GraphWin("Swimming Fish", width, height, autoflush=False)
```

```
while keep_swimming:
    update() # call update to flush the window
```

Intro to sort and search

Motivation

- If I Google “compueter science”, I should still get results for “computer science”, which is what I meant
- The idea is to first sort a list of elements (which could be keywords that link to more data)
- Then after we’ve sorted, we can see which elements are closest to my query or desired search item