

CS65: Introduction to Computer Science

Graphics library
Writing user-defined functions



Md Alimoor Reza
Assistant Professor of Computer Science

Recap

- Graphics library
 - Installation in Thonny

- Quiz 1
 - Quick discussion!
 - Expect similar questions for future quizzes/midterm/final
 - Don't panic, your lowest quiz score will be dropped
 - out of 6 quizzes

Topics

- Creating a graphical window
- Drawing shapes inside the window
 - Circle
 - Rectangle
 - Line, Text, and combinations of these shapes
- Changing coordinate system
- Mouse interaction inside graphics window

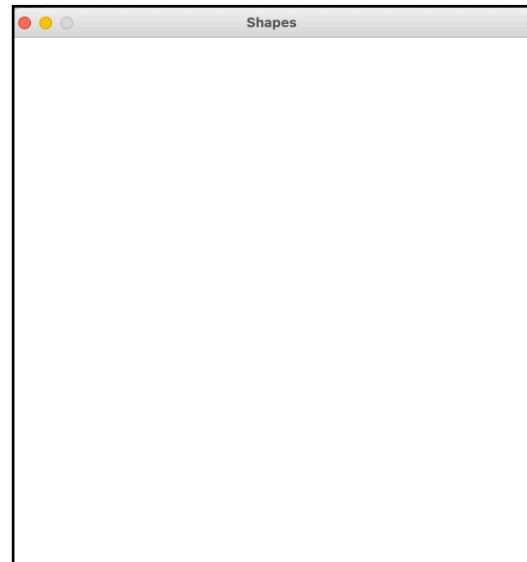
Graphics library

- A simple library (containing other python codes) that makes it easy to experiment with graphics components
- Graphics library: <https://mcsp.wartburg.edu/zelle/python/graphics/graphics/index.html>
- Graphics library provides different graphical objects
 - **Point**, Line, **Circle**
 - Oval, Rectangle, Polygon
 - Text, Image
- You can manipulate properties of these shapes/objects
 - change color and sizes

A simple program using graphics

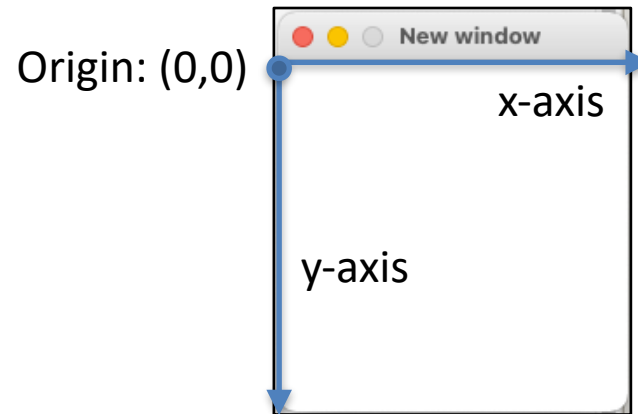
- *GraphWin(...)*: creates the **canvas** or **panel** where everything will be drawn

```
6 from graphics import *
7
8 win = GraphWin("Shapes ", 500, 500)
```



Changing window size

- *GraphWin(...)*: creates the **canvas** or **panel** where everything will be drawn

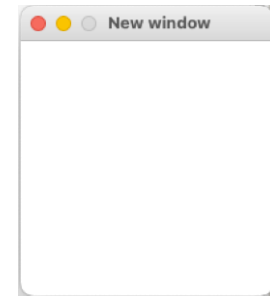
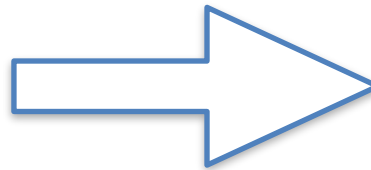


- Coordinate system
 - x: top-left \rightarrow top-right
 - y: top-left \rightarrow bottom-left
- You can set the dimensions of the window by mentioning the width and height (in pixel units)
 - x-axis \rightarrow width
 - y-axis \rightarrow height

Changing window size

- Changing the shape of the window of size (500, 500), just need to change the values inside *GraphWin()*

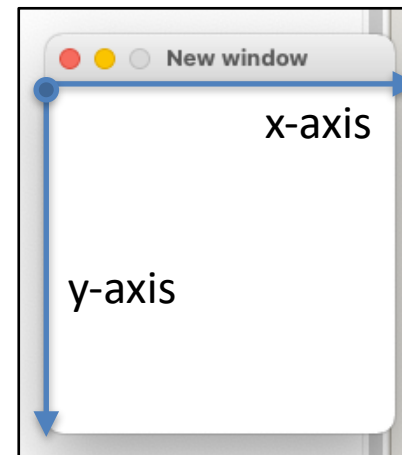
```
6 from graphics import *  
7  
8 win = GraphWin("Shapes ", 500, 500)
```



Changing window size

- Changing the shape of the window of size (500, 500), just need to change the values inside *GraphWin()*
- Write a function for a simple window
 - keep writing your code inside such functions
 - make a habit

Origin: (0,0)



Drawing rectangular window

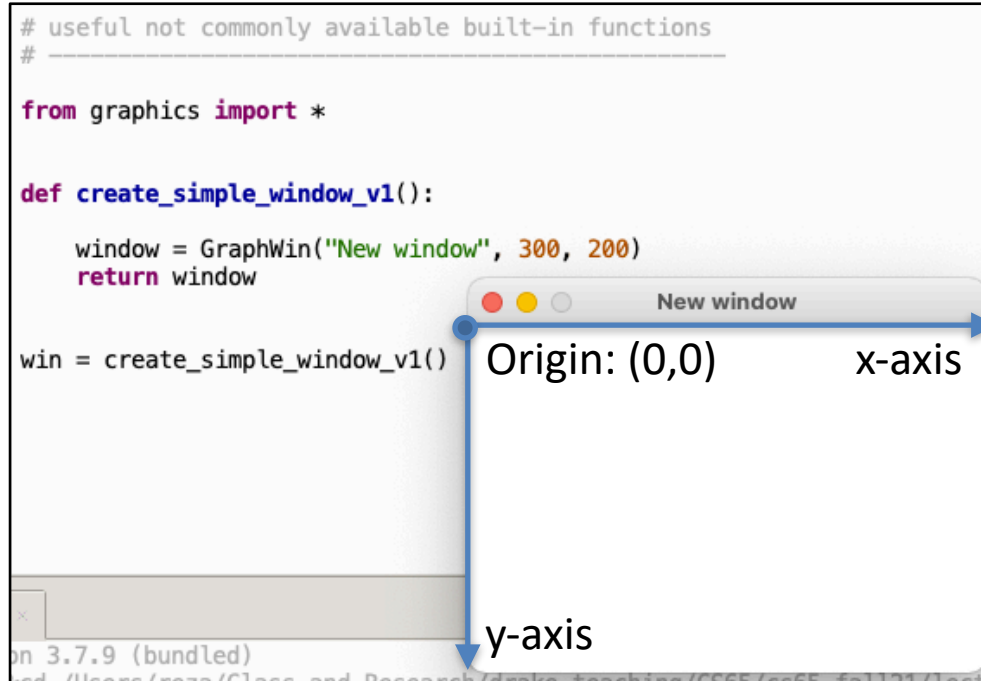
- You can set the dimensions of the window by mentioning the width and height (in pixel units)
 - x-axis — — —> width
 - y-axis — — —> height

```
# useful not commonly available built-in functions
# -----

from graphics import *

def create_simple_window_v1():
    window = GraphWin("New window", 300, 200)
    return window

win = create_simple_window_v1()
```



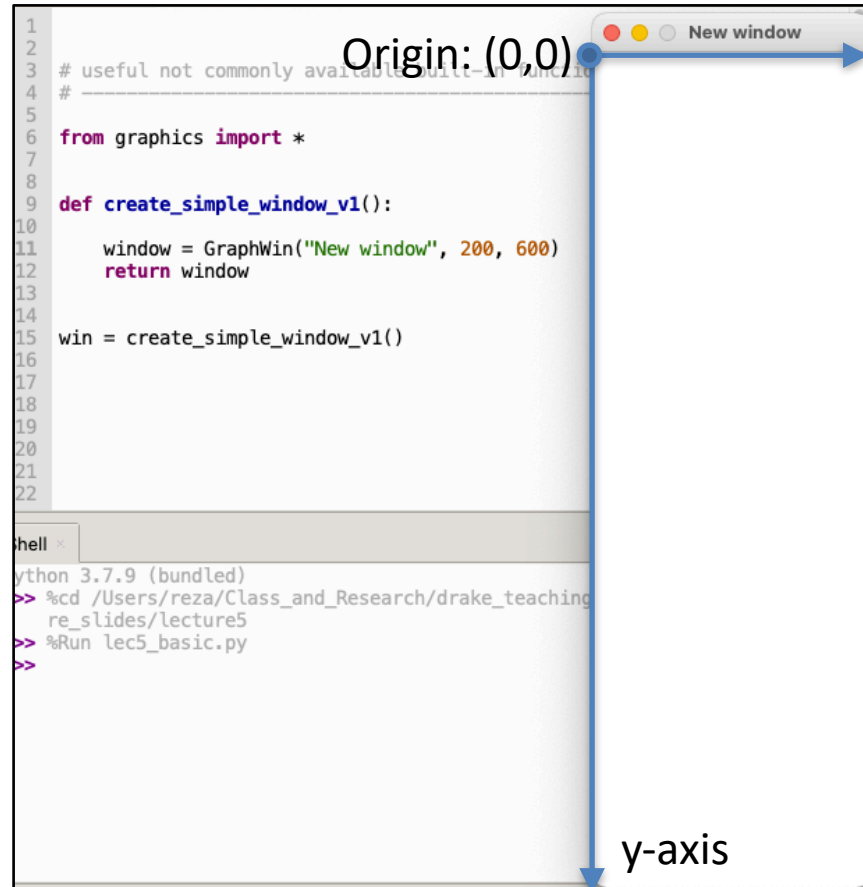
Drawing rectangular window

- You can set the dimensions of the window by mentioning the width and height (in pixel units)

- x-axis — — —> width
- y-axis — — —> height

- Coordinate system

- x: top-left —> top-right
- y: top-left —> bottom-left



```
1
2
3 # useful not commonly available built-in module
4 #
5
6 from graphics import *
7
8
9 def create_simple_window_v1():
10
11     window = GraphWin("New window", 200, 600)
12     return window
13
14
15 win = create_simple_window_v1()
16
17
18
19
20
21
22
```

hell x

```
Python 3.7.9 (bundled)
>> %cd /Users/reza/Class_and_Research/drake_teaching
re_slides/lecture5
>> %Run lec5_basic.py
>>
```

Coding demo

Topics

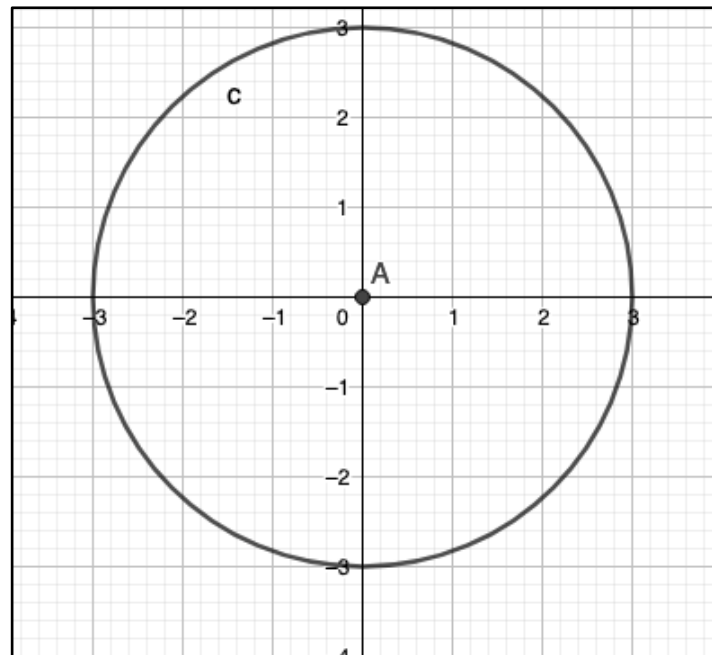
- Creating a graphical window
- Drawing shapes inside the window
 - Circle
 - Rectangle
 - Line, Text, and combinations of these shapes
- Changing coordinate system
- Mouse interaction inside graphics window

Graphical objects from graphics library

- Graphics library provides different shapes (graphical objects):
 - **Point**, Line, **Circle**
 - Oval, **Rectangle**, Polygon
 - Text, Image
- You can manipulate properties of these shapes/objects
 - change color and sizes
- You can also move them around inside the window

Drawing inside the window

- You can draw inside the window
- Drawing a circle inside
 - how many variables do we need for a circle?



Drawing inside the window

- Step 1: Construct a circle
 - Step 1.1: construct a point \rightarrow the center of the circle
 - Step 1.2: fix the radius
 - Step 1.3: put them together
- Step 2: **Draw** the newly constructed circle inside the window

```
from graphics import *

def create_simple_window_v1():

    window = GraphWin("New window", 400, 400)

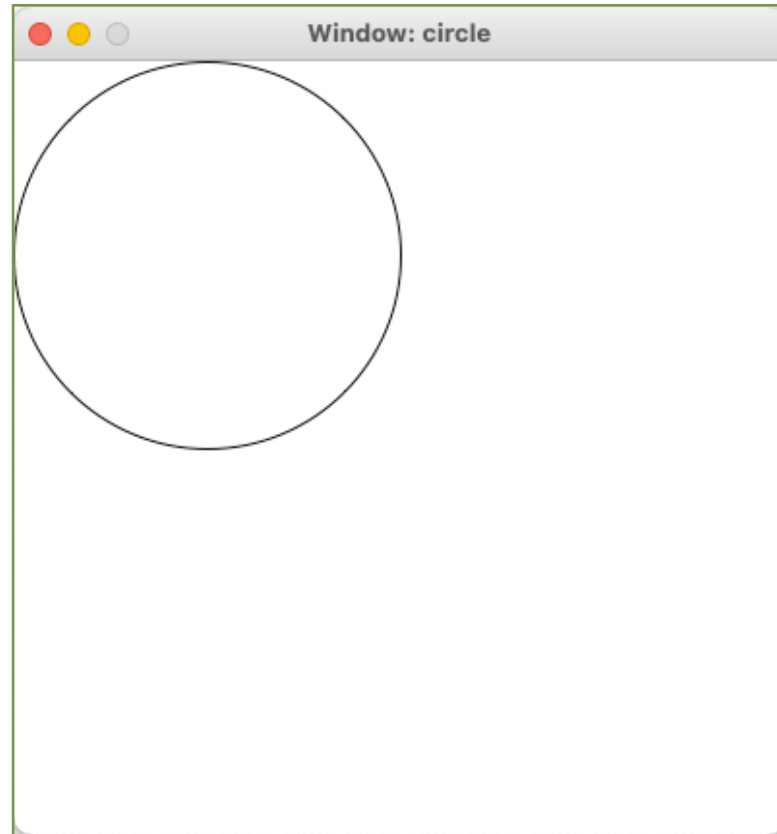
    point = Point(100, 100)      # step 1.1
    radius = 100                 # step 1.2
    circle = Circle(point, radius) # step 1.3

    circle.draw(window)         # step 2

    return window

w1 = create_simple_window_v1()
```

Coding demo



Exercise 1

- Write a function that draws a circle based on
 - user specified **center** (2D point)
 - user specified **radius**
- Optional: the size of the window can also be specified by the user
- What changes do you need to make?

```
from graphics import *

def create_simple_window_v1():

    window = GraphWin("New window", 400, 400)

    point = Point(100, 100)      # step 1.1
    radius = 100                 # step 1.2
    circle = Circle(point, radius) # step 1.3

    circle.draw(window)         # step 2

    return window

w1 = create_simple_window_v1()
```

Drawing rectangle

- Step 1: Construct a rectangle
 - Step 1.1: construct a point → one corner
 - Step 1.2: construct a point → opposite corner
 - Step 1.3: put them together
- Step 2: **Draw** the newly constructed rectangle inside the window

```
from graphics import *

def create_simple_window_w_rect():
    window = GraphWin("Window: Rectangle", 400, 400)

    point1 = Point(50,50)
    point2 = Point(250, 350)

    rect = Rectangle(point1, point2)
    rect.setFill("blue")
    rect.draw(window)

    return window

w1 = create_simple_window_w_rect()
```

<

In 3.7.9 (bundled)
cd /Users/reza/Class_and_Research/drake_teaching/CS65
Run lec6_rect.py
Run lec6_rect.py

Exercise 2

- Write a function that draws a **Rectangle** based on
 - user specified left most corner (2D point)
 - user specified right most corner (2D point)
- What changes do you need to make?

<https://mcsp.wartburg.edu/zelle/python/graphics/graphics/node8.html>

Coding demo

```
from graphics import *

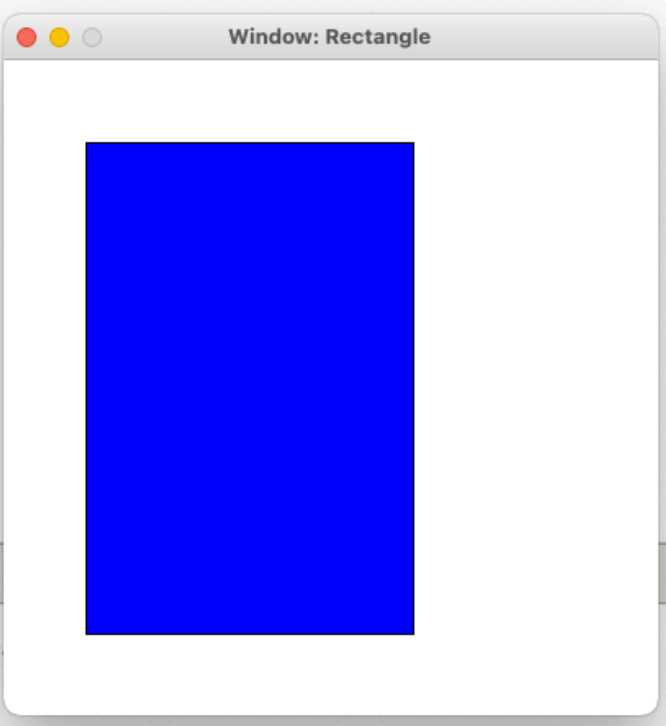
def create_simple_window_w_rect():
    window = GraphWin("Window: Rectangle", 400, 400)

    point1 = Point(50,50)
    point2 = Point(250, 350)

    rect = Rectangle(point1, point2)
    rect.setFill("blue")
    rect.draw(window)

    return window

w1 = create_simple_window_w_rect()
```



in 3.7.9 (bundled)
scd /Users/reza/Class_and_Research/drake_teaching/CS65
Run lec6_rect.py
Run lec6_rect.py

Draw multiple shapes

- Drawing circle, rectangle, and a line connecting them
- Demo

```
from graphics import *
def create_random_shapes():

    win = GraphWin("Multiple shapes: ", 400, 400)

    # create circle
    cir_center = Point(100, 100)
    cir_radius = 100
    my_cir = Circle(cir_center, cir_radius)

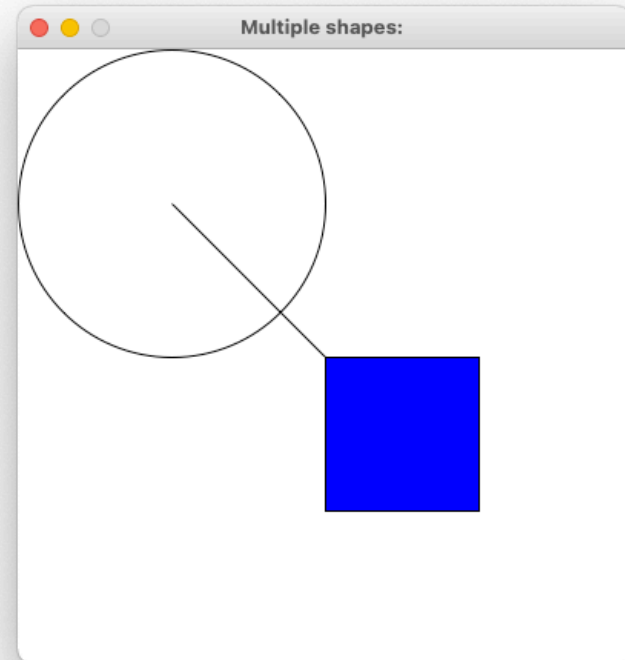
    # create rectangle
    rect_point_a = Point(200, 200)
    rect_point_b = Point(300, 300)
    my_rect = Rectangle(rect_point_a, rect_point_b)
    my_rect.setFill("blue")

    # line connecting circle and rectangle
    my_line = Line(cir_center, rect_point_a)

    # draw all the components
    my_cir.draw(win)
    my_rect.draw(win)
    my_line.draw(win)

    return win

new_window = create_random_shapes()
```



Draw multiple shapes

- Drawing **Text** inside the window
- Demo

```
from graphics import *
def create_random_shapes():
    win = GraphWin("Multiple shapes: ", 400, 400)

    # create circle
    cir_center = Point(100, 100)
    cir_radius = 100
    my_cir = Circle(cir_center, cir_radius)

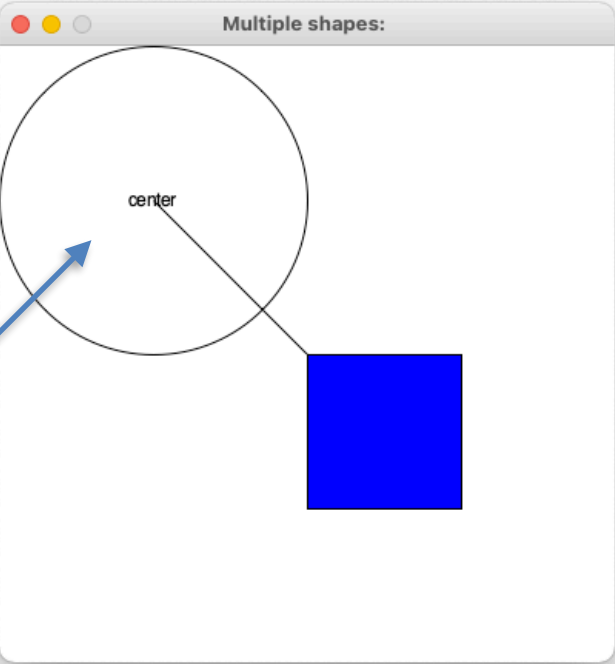
    # create rectangle
    rect_point_a = Point(200, 200)
    rect_point_b = Point(300, 300)
    my_rect = Rectangle(rect_point_a, rect_point_b)
    my_rect.setFill("blue")

    # line connecting circle and rectangle
    my_line = Line(cir_center, rect_point_a)

    # text
    text_point = Point(100, 100)
    my_text = Text(text_point, "center")

    # draw all the components
    my_cir.draw(win)
    my_rect.draw(win)
    my_line.draw(win)
    my_text.draw(win)

    return win
```



Exercise 3

- Write a function that draws a **Triangle** based on
 - challenge 1: find out what how many variables do you need to draw it?
 - challenge 2: then receive that many user inputs
- Hints: read the specification below and try to figure out what might be a useful graphical object for this task

<https://mcsp.wartburg.edu/zelle/python/graphics/graphics/graphref.html>

