

# CS65: Introduction to Computer Science

Continuation of graphics library  
Writing more user-defined functions



Md Alimoor Reza  
Assistant Professor of Computer Science

# Recap

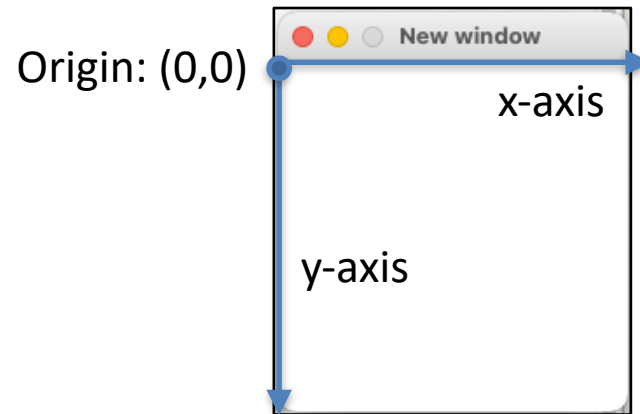
- Graphics library
  - Installation in Thonny
  - Drawing window and a circle using graphics library
- Quiz 1
  - Quick discussion!
  - Expect similar questions for future quizzes/midterm/final
  - Don't panic, your lowest quiz score will be dropped
    - out of 6 quizzes

# Recap: 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

# Recap: graphics window

- *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

# Topics

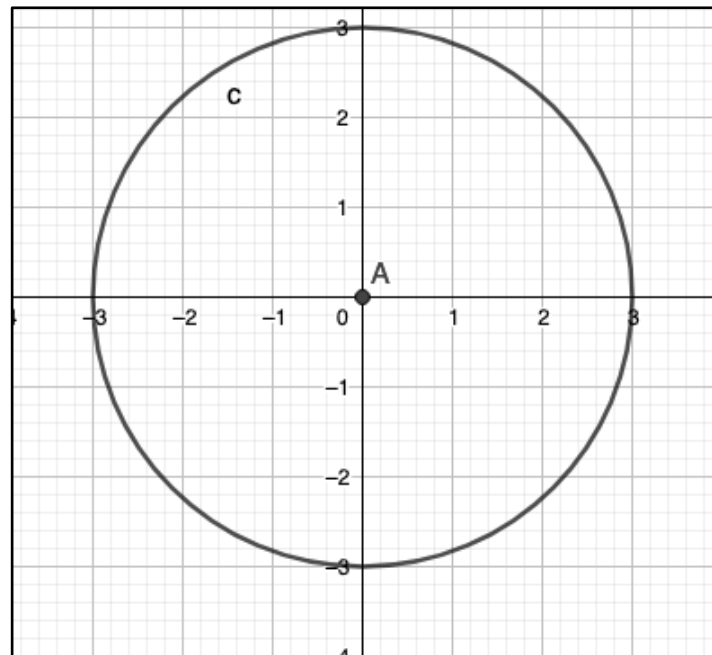
- Drawing shapes inside the window
  - Circle
  - Rectangle
  - Line, Text, and combinations of these shapes
- Changing coordinate system
- Mouse interaction inside graphics window

# 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 —> 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

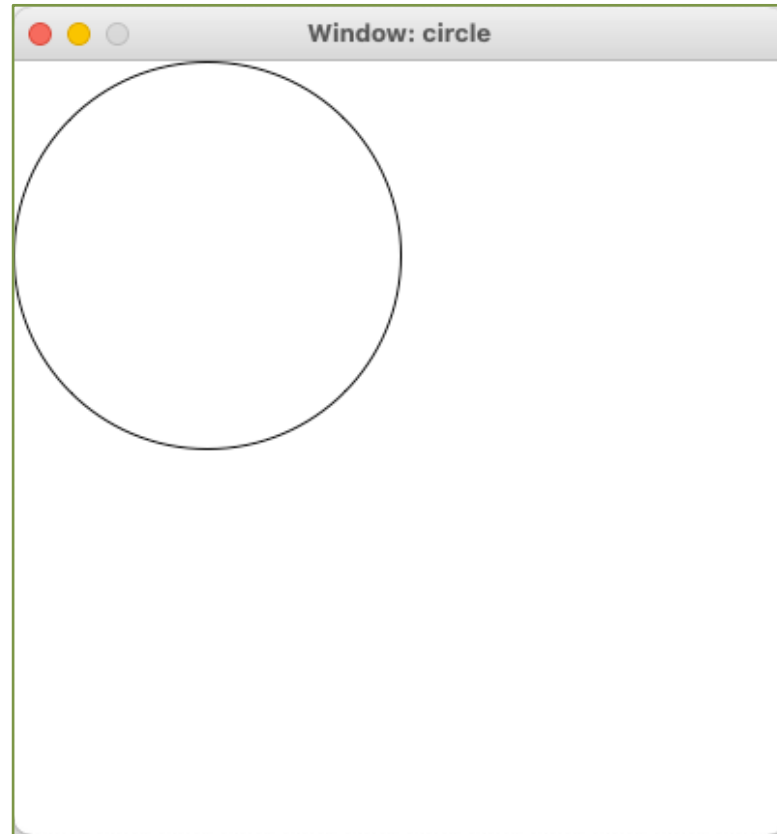
    circle.draw(window)          # step 2

    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()
```

Python 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 *

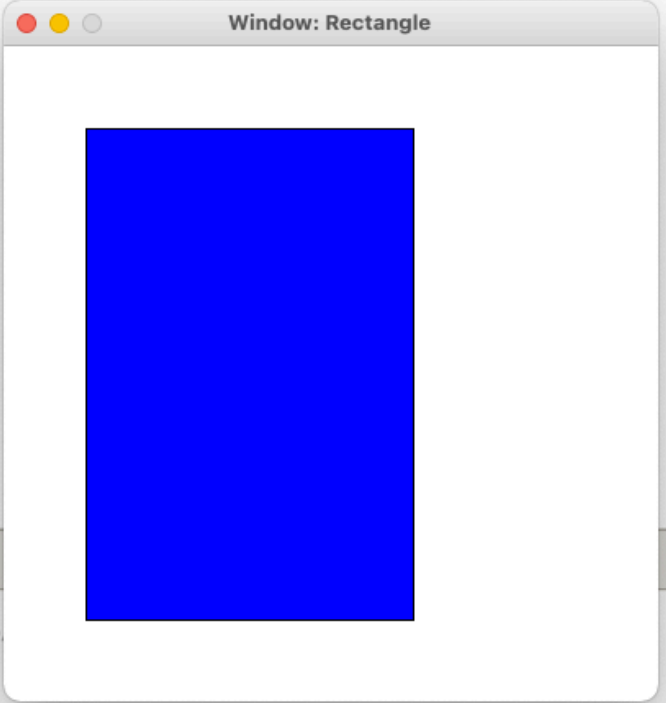
def create_simple_window_w_rect():
    window = GraphWin("Window: Rectangle", 400, 400)

    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()
```



The image shows a code editor window on the left and a graphical window on the right. The code editor contains a Python script that uses the 'graphics' library to create a window titled 'Window: Rectangle' with a width of 400 and a height of 400. Inside this window, a blue rectangle is drawn, starting at point (50, 50) and ending at point (250, 350). The graphical window on the right shows the result of running this code: a white window titled 'Window: Rectangle' containing a solid blue rectangle.

# 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)

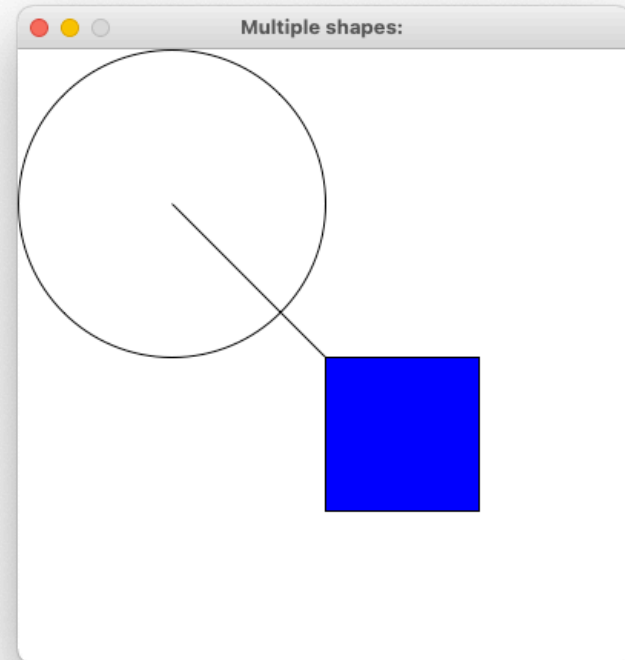
    # create rectangle
    rect_point_a = Point(200, 200)
    rect_point_b = Point(300, 300)
    my_rect = Rectangle(rect_point_a, rect_point_b)
    my_rect.setFill("blue")

    # 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

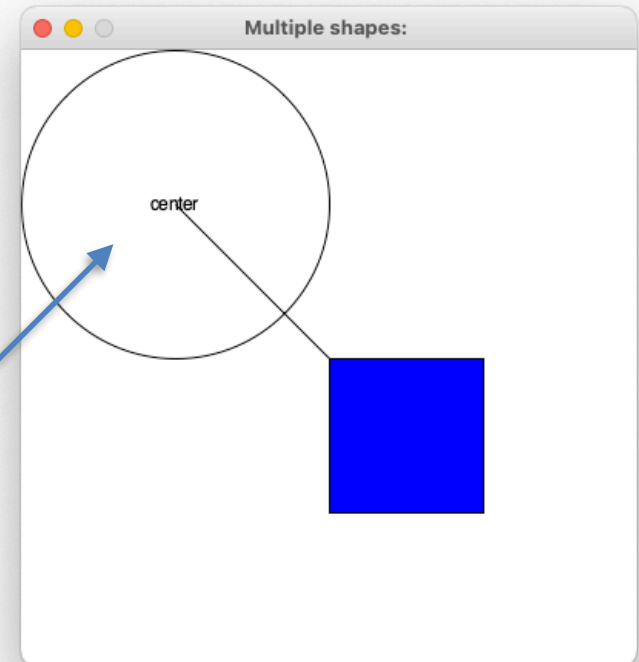
new_window = create_random_shapes()
```



# Draw multiple shapes

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

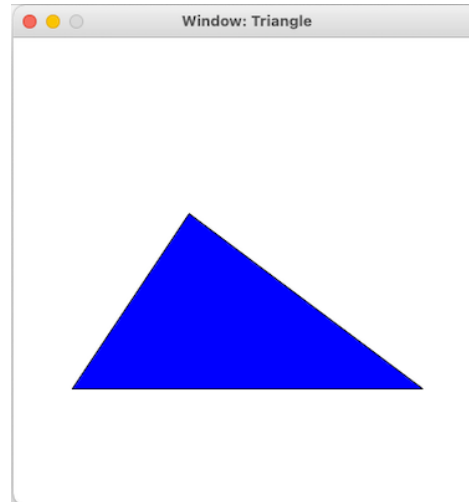
```
from graphics import *  
  
def create_random_shapes():  
  
    win = GraphWin("Multiple shapes: ", 400, 400)  
  
    # create circle  
    cir_center = Point(100, 100)  
    cir_radius = 100  
    my_cir = Circle(cir_center, cir_radius)  
  
    # create rectangle  
    rect_point_a = Point(200, 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>

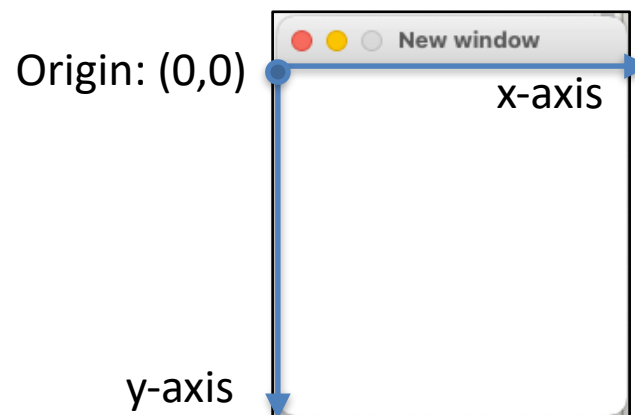




# Changing coordinate system

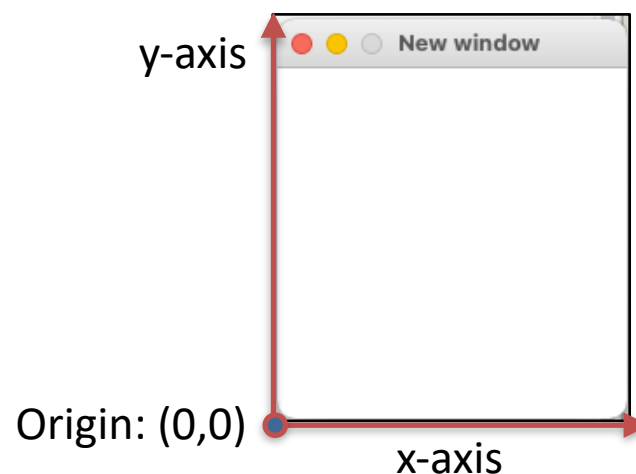
- Default coordinate system:

- x: top-left  $\rightarrow$  top-right
- y: top-left  $\rightarrow$  bottom-left



- Transforming into traditional coordinate system

- x: bottom-left  $\rightarrow$  bottom-right
- y: bottom-left  $\rightarrow$  bottom-up



# Changing coordinate system

```
from graphics import *
```

```
def create_circle_default_coord(px, py, radius):
```

```
    window = GraphWin("Default coord. system", 400, 400)
```

```
    # create circle
```

```
    cir_center = Point(px, py)
```

```
    cir_radius = radius
```

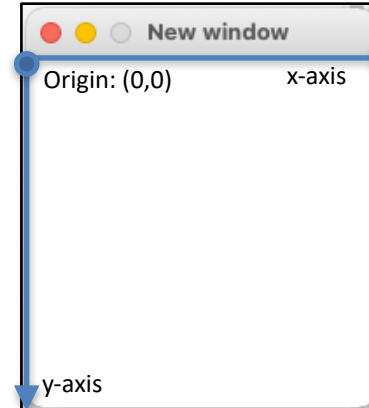
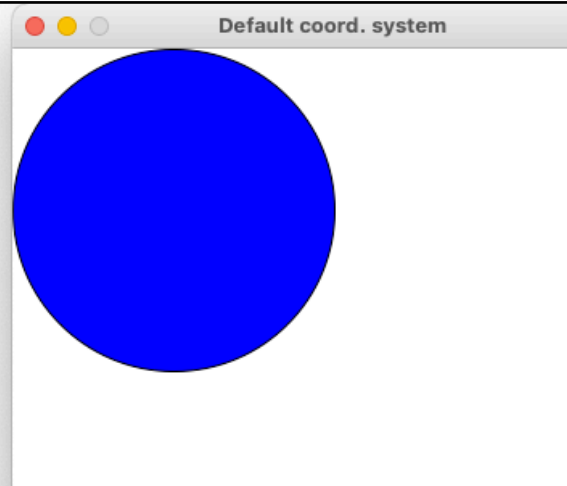
```
    my_cir = Circle(cir_center, cir_radius)
```

```
    my_cir.setFill("blue")
```

```
    my_cir.draw(window)
```

```
    return window
```

```
win_default = create_circle_default_coord(100, 100, 100)
```



```
from graphics import *
```

```
def create_circle_transformed_coord(px, py, radius):
```

```
    window = GraphWin("Transformed coord. system", 400, 400)
```

```
    # ----- changing the coordinate system -----
```

```
    # lower left corner becomes 0,0
```

```
    # upper right corner becomes 400, 400
```

```
    window.setCoords(0,0, 400, 400)
```

```
    # -----
```

```
    # create circle
```

```
    cir_center = Point(px, py)
```

```
    cir_radius = radius
```

```
    my_cir = Circle(cir_center, cir_radius)
```

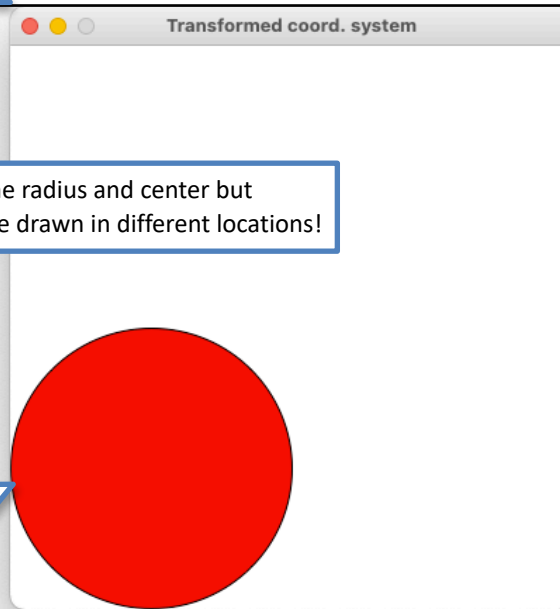
```
    my_cir.setFill("red")
```

```
    my_cir.draw(window)
```

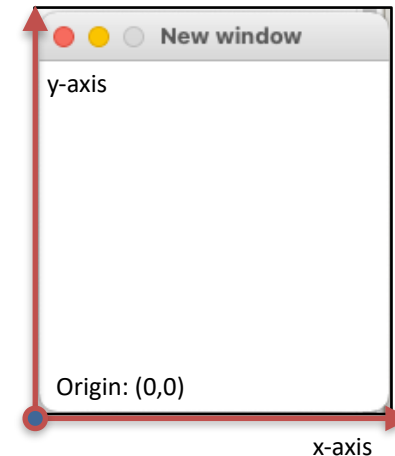
```
    return window
```

```
# ----- lower left corner (0,0) and upper right corner (400, 400) -----
```

```
win_trans = create_circle_transformed_coord(100, 100, 100)
```



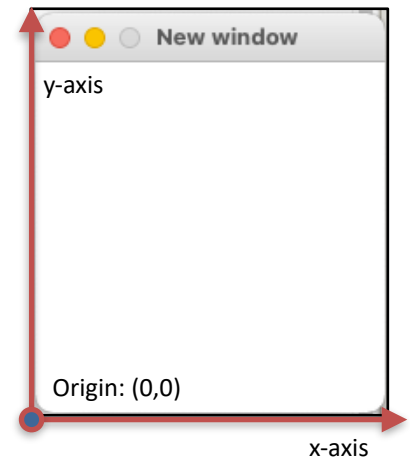
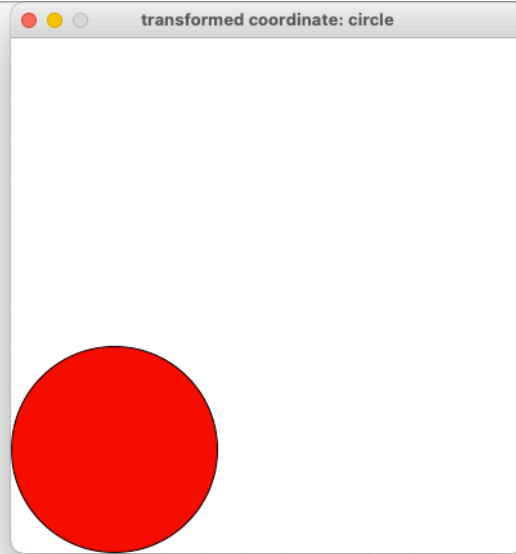
Same radius and center but  
were drawn in different locations!



# Changing coordinate system

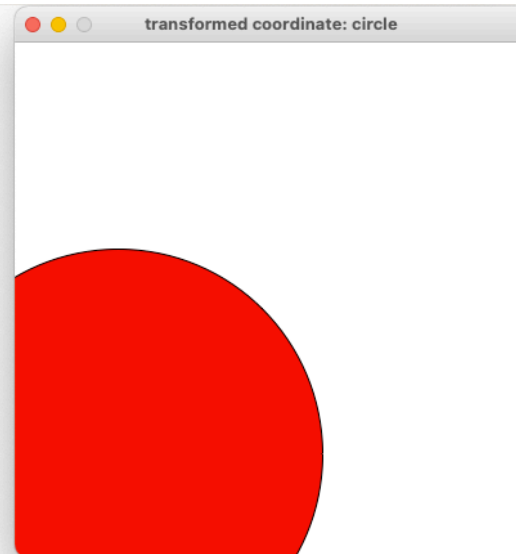
```
from graphics import *  
  
def create_transformed_window_v1(x, y, radius):  
    window = GraphWin("transformed coordinate: circle", 400, 400)  
    window.setCoords(-100, -100, 400, 400)  
  
    circle_center = Point(x, y)  
    circle_radius = radius  
    tr_circle = Circle(circle_center, circle_radius)  
    tr_circle.setFill("red")  
    tr_circle.draw(window)  
  
    return window
```

```
my_win = create_transformed_window_v1(0, 0, 100)
```



```
from graphics import *  
  
def create_transformed_window_v1(x, y, radius):  
    window = GraphWin("transformed coordinate: circle", 400, 400)  
    window.setCoords(-100, -100, 400, 400)  
  
    circle_center = Point(x, y)  
    circle_radius = radius  
    tr_circle = Circle(circle_center, circle_radius)  
    tr_circle.setFill("red")  
    tr_circle.draw(window)  
  
    return window
```

```
my_win = create_transformed_window_v1(0, 0, 200)
```

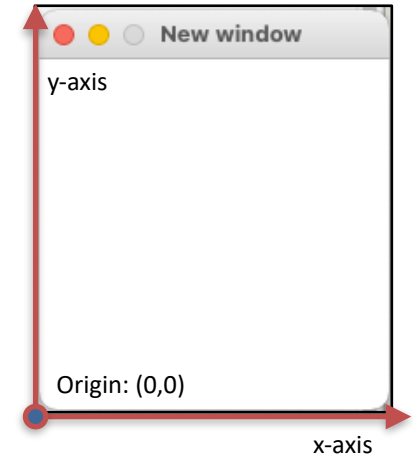
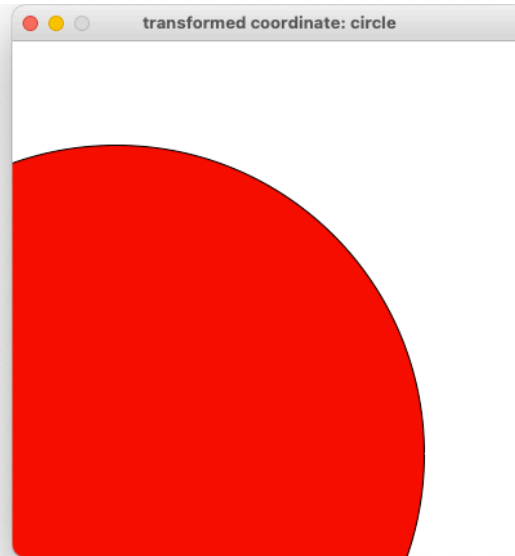


The red circle has been clipped

# Changing coordinate system

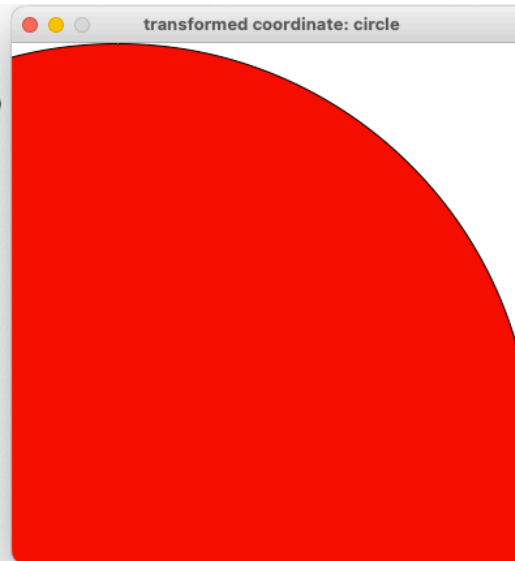
```
from graphics import *  
  
def create_transformed_window_v1(x, y, radius):  
    window = GraphWin("transformed coordinate: circle", 400, 400)  
    window.setCoords(-100, -100, 400, 400)  
  
    circle_center = Point(x, y)  
    circle_radius = radius  
    tr_circle = Circle(circle_center, circle_radius)  
    tr_circle.setFill("red")  
    tr_circle.draw(window)  
  
    return window
```

```
my_win = create_transformed_window_v1(0, 0, 300)
```



```
from graphics import *  
  
def create_transformed_window_v1(x, y, radius):  
    window = GraphWin("transformed coordinate: circle", 400, 400)  
    window.setCoords(-100, -100, 400, 400)  
  
    circle_center = Point(x, y)  
    circle_radius = radius  
    tr_circle = Circle(circle_center, circle_radius)  
    tr_circle.setFill("red")  
    tr_circle.draw(window)  
  
    return window
```

```
my_win = create_transformed_window_v1(0, 0, 400)
```



# Interaction with the user using Mouse

- GraphWin() has a function that allows us to identify the location of your mouse-click
  - 2D coordinate
  - represented by Point object

```
from graphics import *  
  
window = GraphWin("Mouse interaction", 400, 400)  
  
mouse_point1 = window.getMouse()  
mouse_point2 = window.getMouse()  
  
print(mouse_point1)  
print(mouse_point2)|
```

# Summary

- **Takeaway from this lecture**

- Basics shapes are already defined, you just need to draw them according in specific ways
  - Circle, Rectangle, Triangle
- Change of coordinates to draw in a different way
- Mouse interaction with the user

- **To do:**

- Read: <https://mcsp.wartburg.edu/zelle/python/graphics/graphics/index.html>

- **Announcements:**

- Assignment 1 will be out soon! It will be due in 2 weeks.