Chapter 19 — Better Graphics

Introduction

This chapter introduces how to create simple graphics and animations using the "Graphics" library by Professor John M. Zelle at Wartburg College. This library interfaces to and simplifies the built in `tkinter` library.

To use this library you must first download the file graphics.py and place it in the directory where your Python programs will be saved. You may download the open-source file here: http://mcsp.wartburg.edu/zelle/python/ . There is also a manual going into much more detail that can be downloaded here: http://mcsp.wartburg.edu/zelle/python/graphics/graphics.pdf .

The programs and method for using this module, in this chapter, varies from the original author's documentation. To maintain consistency between chapters, the graphics module will be imported using an absolute import and not a relative import. The examples will prefix the modules' classes by the module name 'graphics'. Using modules this way insures that if two modules have classes or properties with the same names, there will not be a conflict.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Use a coordinate system with the upper left corner at origin.
- Use the point class to create shapes and to get the coordinate location of shapes on the screen.
- Draw basic shapes like: circles, squares, rectangles, ovals, and lines.
- Apply modifications to shapes to change the display of their color and borders.
- Draw text on the screen in different fonts and sizes.
- Use a system delay and the move method of shapes to create simple animations.

Prerequisites

The material in this chapter only depends on material from Chapters 1-7.

Window Coordinates

In mathematics...

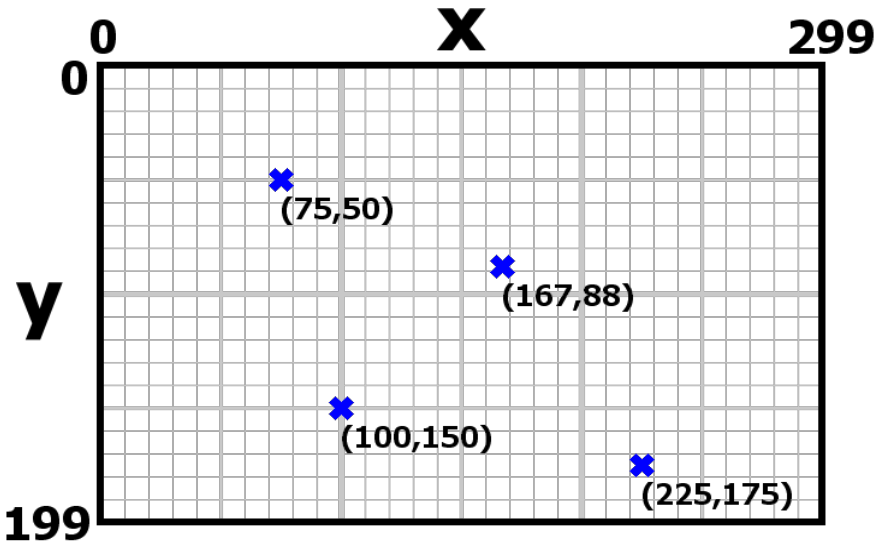


Illustration 28: Window Coordinates

Simple Circle

This first program shows the basic use of the Graphics module. To use this, you must load the module and create a window object that will contain all the drawing objects.

`graphics`	Module
The graphics module includes methods to draw shapes and text on a window.	

`import graphics`	Statement
Load in the "graphics" library of objects. You need to place this statement in your program before you try to draw. Be sure that you have downloaded the file "graphics.py" and have saved a copy	

of it to the same folder that you are saving your program to.	

graphics.GraphWin(*title_expr, width_expr, height_expr***)**	Class in graphics
Create a new object of the GraphWin type. A graphics window will be displayed with a drawing space of the size defined in width and height. A title will also be placed on the top bar of the window.	

All access to the GraphWin is dome using point objects. A point object is defined by the number of pixels over from the left edge and the number of pixels from the top.

graphics.Point(*x_expr, y_expr***)**	Shape class in graphics
Create a new object of the Point type. Points represent locations on the screen and are used to position shapes, query shapes, and determine where the mouse is clicked.	

Now that we have defined a Point, we will use it to define the center of a circle. The second parameter needed is the radius of the circle to draw. Once we have created the shape object, the last step is to tell it to draw.

graphics.Circle(*point, radius_expr***)**	Shape class in graphics
Create a new circle object. When the circle is drawn, place the center at the point and draw with the radius.	

*graphicsobject***.Draw(***graphwin***)**	Method of shape objects
Method used by all graphics objects to actually draw them on the window.	

The last two methods of GraphWin in our first program are getMouse and close. The getMouse method waits until a user clicks with the mouse pointer on the window. It will return the Point where the click happened, but in this program we are just using it to pause, and don't care where the click was made. The last method is close. It closes the graphics window and terminates the Python program.

graphwin.**getMouse()**	Method of GraphWin
Wait for the user to do a mouse click on the window and return the point where they clicked.	

graphwin.**close()**	Method of GraphWin
Close the window and release all of its resources.	

```
1| import graphics
2|
3| win = graphics.GraphWin("My Circle", 100, 100)
4|
5| c = graphics.Circle(graphics.Point(50,50), 10)
6| c.draw(win)
7|
8| print("Click on popup window to close program.")
9| win.getMouse() # Pause to view result
10| win.close()    # Close window when done
```

Illustration 29: A Circle Drawn with the Graphics Module

Shape Attributes

There are several attributes of shapes that you may manipulate. This allows you to draw exactly what you want to. There are additional attributes, beyond those covered here. You will find details about them in the documentation for the Graphics module.

graphicsobject.**setWidth(***width***)**	Method of shape objects

Sets the drawing pen width in pixels.	

graphics.color_rgb(r, g, b)	Class in graphics
Return a color value for use in the setOutline and setFill methods of GraphicsObjects. Pass the three values between 0-255 to mix colors. For example `color_rgb(0,0,0)` is black, `color_rgb(255,255,255)` is white, and `color_rgb(255,100,100)` is pinkish.	

Colors may also be defined using any of the hundreds of named colors that the `tkinter` module supports.

aquamarine	green	orange
	grey	pink
blue	grey33	red
brown	grey66	thistle
coral	indigo	violet
cyan	maroon	wheat

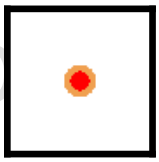
graphicsobject.**setOutline(***color***)**	Method of shape objects
Sets the shape's outline to a specific color. You may use a color returned by the `color_rgb` function, a named tkinter color, or a color in hexadecimal notation. http://effbot.org/tkinterbook/tkinter-widget-styling.htm	

graphicsobject.**setFill(***color***)**	Method of shape objects
Sets the shape's fill to a specific color. You may use a color returned by the `color_rgb` function, a named tkinter color, or a color in hexadecimal notation. http://effbot.org/tkinterbook/tkinter-widget-styling.htm	

graphicsobject.**setWidth(***pixels***)**	Method of shape objects

Sets the width of the outline of a shape, in pixels.	

```
import graphics

win = graphics.GraphWin("My Circle", 100, 100)

c = graphics.Circle(graphics.Point(50,50), 10)
c.setFill("red")
color = graphics.color_rgb(255,255,0)

c.setOutline("#f0a050")
c.setOutline(color)
c.setWidth(3)
c.draw(win)

print("Click on popup window to close program.")
win.getMouse() # Pause to view result
win.close()    # Close window when done
```



Additional Shapes and DrawingText

l = Line(Point(x,y), Point(x,y))

graphics.Line(*point, point***)**	Shape class in graphics
Draw a line from one point to another.	

r = Rectangle(Point(x,y), Point(x,y))

graphics.Rectangle(*point, point***)**	Shape class in graphics
Create a rectangle object with the first point being one corner and the second point being the opposite corner. Typically, the points will be the top-left and bottom right corners.	

o = Oval(Point(x,y), Point(x,y))

graphics.Oval(*point, point*)	Shape class in graphics
Create an oval object with the first point being one corner and the second point being the opposite corner. Typically, the points will be the top-left and bottom right corners.	

t = Text(Point(x,y), string)
t.setSize(pt)
t.setFace(font) - 'helvetica', 'courier', 'times roman', and 'arial'

graphics.Text(*point, string*)	Shape class in graphics
Draw text on the window at the specified point.	

***textobject*.setSize(*point*)**	Method of text objects
Sets the height of the text in point. A point is 1/72 of an inch when printed.	

***textobject*.setFace(*font_name*)**	Method of text objects
Sets font family to be used to draw the text. Fonts include: 'helvetica', 'courier', 'times roman', and 'arial'.	

Getting a Shape's Location

Shapes with a bounding box (line, circle, rectangle, oval) have getP1() and getP2() that returns the bounding box

also has getCenter() that returns the center

***graphicsobject*.getP1()** ***graphicsobject*.getP2()**	Method of shape objects
Gets the coordinates of a box that bounds (surrounds) the shape, as a point object.	

graphicsobject.getCenter()	Method of shape objects
Gets the coordinates of the center of a shape, as a point object.	

text has getAnchor()

textobject.getAnchor()	Method of text objects
Gets the point on the screen where the text has been drawn.	

point.getX(), point.getY() - get the actual location

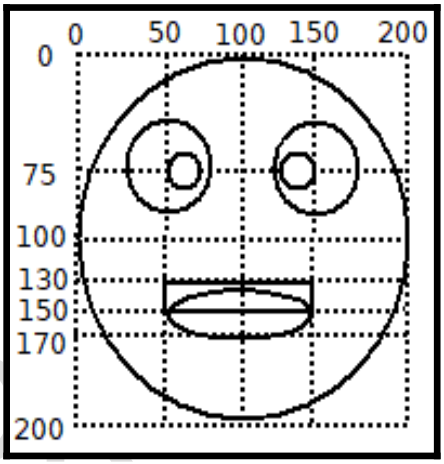
pointobject.getX() pointobject.getY()	Method of point objects
Get the actual x and y location of the point.	

Simple Animation

shape.move(dx,dy) — move a shape by x and y

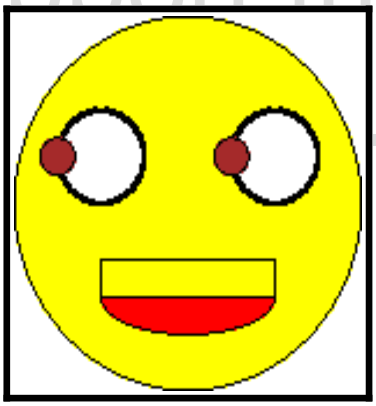
import time
time.sleep(seconds) — May be fractional.

The smiling face is laid out like:




```
import graphics
import time

def main():

    speed = .05

    window = graphics.GraphWin()
    face = graphics.Circle(graphics.Point(100,100),100)
    face.setFill("yellow")
    face.draw(window)
    mouth = graphics.Oval(graphics.Point(50,130),
  graphics.Point(150,170))
    mouth.setFill("red")
    mouth.draw(window)
    lip =
  graphics.Rectangle(graphics.Point(50,130),graphics.Point(150,
  150))
    lip.setFill("yellow")
    lip.setWidth(0)
    lip.draw(window)
    reye = graphics.Circle(graphics.Point(50,75),25)
    reye.setFill("white")
    reye.setWidth(2)
    reye.draw(window)
    leye = graphics.Circle(graphics.Point(150,75),25)
    leye.setFill("white")
    leye.setWidth(2)
    leye.draw(window)
```

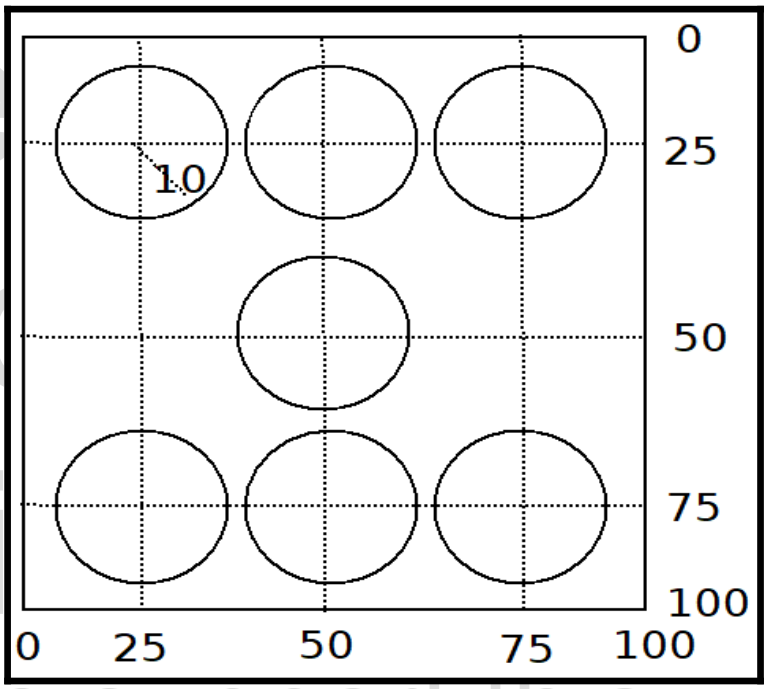
```
    rpupil = graphics.Circle(graphics.Point(25,75), 10)
    rpupil.setFill("brown")
    rpupil.draw(window)
    lpupil = graphics.Circle(graphics.Point(125,75), 10)
    lpupil.setFill("brown")
    lpupil.draw(window)


    while not window.checkMouse():
        for t in range(50):
            rpupil.move(1,0)
            lpupil.move(1,0)
            time.sleep(speed)
        for t in range(50):
            rpupil.move(-1,0)
            lpupil.move(-1,0)
            time.sleep(speed)

    window.close()

main()
```

Sample Program — Rolling a Die

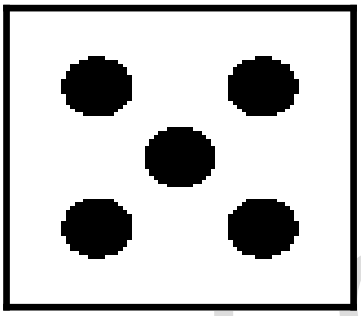
This sample program draws a randomly thrown six sided die face. You will see the loop where randomly flips are shown and the final toss is then displayed. It repeats the roll every time you click on the image.

The die face is laid out like:




```
import graphics
import random
import time


def drawDie(window, n):
    window.delete("all")
    if n == 2 or n == 3 or n == 4 or n == 5 or n == 6:
        pip = graphics.Circle(graphics.Point(25,25),10)
        pip.setFill("black")
        pip.draw(window)
    if n == 6:
        pip = graphics.Circle(graphics.Point(50,25),10)
        pip.setFill("black")
        pip.draw(window)
    if n == 4 or n == 5 or n == 6:
        pip = graphics.Circle(graphics.Point(75,25),10)
        pip.setFill("black")
        pip.draw(window)
    if n == 1 or n == 3 or n == 5:
        pip = graphics.Circle(graphics.Point(50,50),10)
```

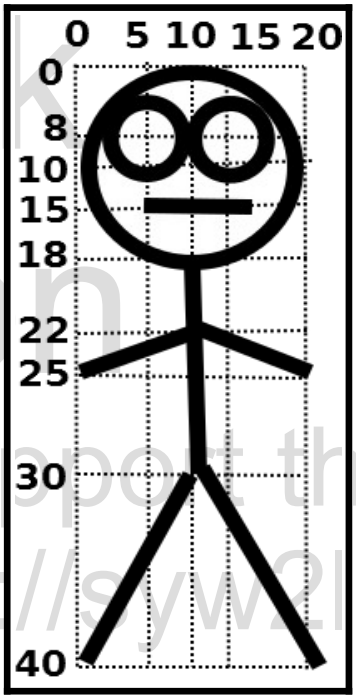
```
22|         pip.setFill("black")
23|         pip.draw(window)
24|     if n == 4 or n == 5 or n == 6:
25|         pip = graphics.Circle(graphics.Point(25,75),10)
26|         pip.setFill("black")
27|         pip.draw(window)
28|     if n == 6:
29|         pip = graphics.Circle(graphics.Point(50,75),10)
30|         pip.setFill("black")
31|         pip.draw(window)
32|     if n == 2 or n == 3 or n == 4 or n == 5 or n == 6:
33|         pip = graphics.Circle(graphics.Point(75,75),10)
34|         pip.setFill("black")
35|         pip.draw(window)
36|
37| def main():
38|
39|     window = graphics.GraphWin("Die",100,100)
40|     while True:
41|         for t in range(15):
42|             n = random.randrange(1,7)
43|             drawDie(window,n)
44|             time.sleep(.1*random.random())
45|
46|         n = random.randrange(1,7)
47|         print(n)
48|         drawDie(window,n)
49|
50|         window.getMouse()
51|
52|     window.close()
53|
54| main()
```



Sample Program — Animate a Stick Human

\

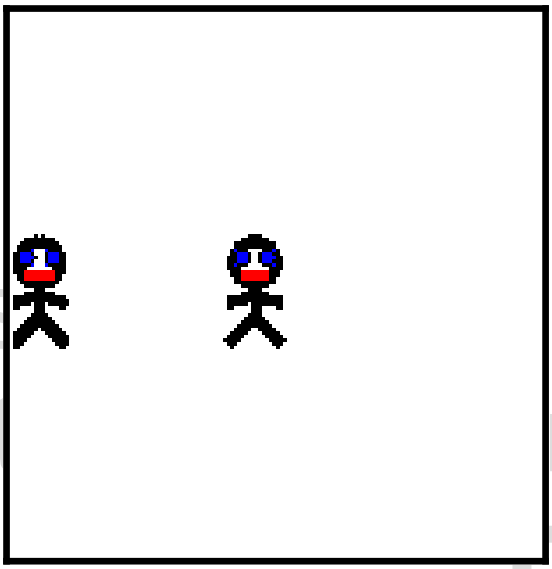
This program creates a couple of classes to draw multiple shape objects and then animates them. The first is called "list_of_shapes" that allows us to make a list of shapes, draw them all at once, and move them all with a single method call. The second method inherits list_of_shapes and then creates several shapes to draw a stick figure.




```
 1| import graphics
 2| import time
 3|
 4| class list_of_shapes:
 5|     def __init__(self):
 6|         self.shapes = []
 7|     def move(self, dx, dy):
 8|         for shape in self.shapes:
 9|             shape.move(dx, dy)
10|     def draw(self, window):
11|         for shape in self.shapes:
12|             shape.draw(window)
13|
14| class stickman(list_of_shapes):
```

```
15|     def __init__(self):
16|         list_of_shapes.__init__(self)
17|         head = graphics.Circle(graphics.Point(10,10),8)
18|         head.setWidth(3)
19|         self.shapes.append(head)
20|         eyer = graphics.Circle(graphics.Point(5,8),3)
21|         eyer.setFill("blue")
22|         self.shapes.append(eyer)
23|         eyel = graphics.Circle(graphics.Point(15,8),3)
24|         eyel.setFill("blue")
25|         self.shapes.append(eyel)
26|         mouth = graphics.Line(graphics.Point(5,15),
    graphics.Point(15,15))
27|         mouth.setWidth(3)
28|         mouth.setOutline("red")
29|         self.shapes.append(mouth)
30|         body =
    graphics.Line(graphics.Point(10,18),graphics.Point(10,30))
31|         body.setWidth(3)
32|         self.shapes.append(body)
33|         armr =
    graphics.Line(graphics.Point(10,22),graphics.Point(0,25))
34|         armr.setWidth(3)
35|         self.shapes.append(armr)
36|         arml =
    graphics.Line(graphics.Point(10,22),graphics.Point(20,25))
37|         arml.setWidth(3)
38|         self.shapes.append(arml)
39|         legr =
    graphics.Line(graphics.Point(10,30),graphics.Point(0,40))
40|         legr.setWidth(3)
41|         self.shapes.append(legr)
42|         legl =
    graphics.Line(graphics.Point(10,30),graphics.Point(20,40))
43|         legl.setWidth(3)
44|         self.shapes.append(legl)
45|
46| def main():
47|     window = graphics.GraphWin()
48|     person = stickman()
49|     person2 = stickman()
50|     person.draw(window)
51|     person2.draw(window)
```

```
    while not window.checkKey():
        for x in range(0,50):
            person.move(2,2)
            person2.move(0,2)
            time.sleep(.05)
        for x in range(0,50):
            person.move(-2,-2)
            person2.move(0,-2)
            time.sleep(.05)
    window.close()

main()
```



Summary

Goes here

Important Terms

- Circle
- GraphWin
- Line
- Oval
- Point
- Rectangle
- Text
-]getP1
- color_rgb

- coordinate
- getAnchor
- getCenter
- getP2
- graphics
- move
- setFace
- setFill
- setOutline
- setSize
- setWidth
- sleep
- time

Exercises

Here

Word Search

```
o r u s a 1 s e t f i l l n o j o
e c b a s e n t y o v a l c t e r
c b t a l t t t o z e g g o t g p
o 1 e n e v k o t h r e l o m r i
l l i i e s p o i n t t s r o a ]
o s t a p w l l e s f c r d v p g
r g e t a n c h o r e e t i e h e
_ o i s e t s i z e e n s n h w t
r e g s e t f a c e s t e a f i p
g r e c t a n g l e a e t t c n 1
b r y g e n · · r q c r w e i v t
t i o s s c l a p e e c i t r m p
i l i n e o h t p l r i d e c t u
g r a p h i c s i k o n t t l t m
e l g t i m e v e r e s h a e c y
] n · d r z g e t p 2 l s n c i x
g t e x t s e t o u t l i n e e i
```

Circle, GraphWin, Line, Oval, Point, Rectangle, Text,]getP1, color_rgb, coordinate, getAnchor, getCenter, getP2, graphics, move, setFace, setFill, setOutline, setSize, setWidth, sleep, time

References