# Appendix I: The Bouncing Ball Program

# 16

```
from turtle import *
import tkinter
import random

screenMaxX = 300
screenMaxY = 300
screenMinX = -300
screenMinY = -300

# This is a example of a class that uses inheritance.
# The Ball class inherits from the RawTurtle class.
# This is indicated to Python by writing
# class Ball(RawTurtle):
# That says, class Ball inherits from RawTurtle, which
# means that a Ball is also a RawTurtle, but it is a
# little more than just a RawTurtle. The Ball class also
# maintains a dx and dy value that is the amount
# to move as it is animated.
class Ball(RawTurtle):
      # The __init__  is the CONSTRUCTOR. Its purpose is to
      # initialize the object by storing data in the object. Anytime
      # self.variable = value is written a value is being stored in
      # the object referred to by self. self always points to the
      # current object.
      def __init__(self,cv,dx,dy):
            # Because the Ball class inherits from the RawTurtle class
            # the Ball class constructor must call the RawTurtle class
            # constructor to initialize the RawTurtle part of the object.
            # The RawTurtle class is called the BASE class. The Ball class
            # is called the DERIVED class. The call to initialize the
            # base class part of the object is always the first thing
            # you do in the derived class's constructor.
            RawTurtle.__init__(self,cv)

            # Then the rest of the object can be initialized.
            self.penup()
            self.shape("soccerball.gif")
            self.dx = dx
            self.dy = dy

      # The move method is a mutator method. It changes the data
      # of the object by adding something to the Ball's x and y
      # position.
      def move(self):
            newx = self.xcor() + self.dx
            newy = self.ycor() + self.dy

            # The if statements below make the ball
```



K.D. Lee, *Python Programming Fundamentals*,
Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-6642-9_16

```
49              # bounce off the walls.
50              if newx < screenMinX:
51                    newx = 2 * screenMinX - newx
52                    self.dx = -self.dx
53              if newy < screenMinY:
54                    newy = 2 * screenMinY - newy
55                    self.dy = - self.dy
56              if newx > screenMaxX:
57                    newx = 2 * screenMaxX - newx
58                    self.dx = - self.dx
59              if newy > screenMaxY:
60                    newy = 2 * screenMaxY - newy
61                    self.dy = -self.dy
62
63              # Then we call a method on the RawTurtle
64              # to move to the new x and y position.
65              self.goto(newx,newy)
66
67  # Once the classes and functions have been defined we'll put our
68  # main function at the bottom of the file. Main isn't necessarily
69  # written last. It's simply put at the bottom of the file. Main
70  # is not a method. It is a plain function because it is not
71  # defined inside any class.
72  def main():
73
74        # Start by creating a RawTurtle object for the window.
75        root = tkinter.Tk()
76        root.title("Bouncing Balls!")
77        cv = ScrolledCanvas(root,600,600,600,600)
78        cv.pack(side = tkinter.LEFT)
79        t = RawTurtle(cv)
80        fram = tkinter.Frame(root)
81        fram.pack(side = tkinter.RIGHT,fill=tkinter.BOTH)
82
83        screen = t.getscreen()
84        screen.setworldcoordinates(screenMinX,screenMinY,screenMaxX,screenMaxY)
85        t.ht()
86        screen.tracer(20)
87        screen.register_shape("soccerball.gif")
88
89        # The ballList is a list of all the ball objects. This
90        # list is needed so the balls can be animated by the
91        # program.
92        ballList = []
93
94        # Here is the animation handler. It is called at
95        # every timer event.
96        def animate():
97              # Tell all the balls to move
98              for ball in ballList:
99                    ball.move()
100
101             # Set the timer to go off again
102             screen.ontimer(animate)
103
104       # This code creates 10 balls heading
105       # in random directions
106       for k in range(10):
107             dx = random.random() * 3 + 1
108             dy = random.random() * 3 + 1
109             # Here is how a ball object is created. We
110             # write ball = Ball(5,4)
111             # to create an instance of the Ball class
112             # and point the ball reference at that object.
113             # That way we can refer to the object by writing
114             # ball.
115             ball = Ball(cv,dx,dy)
116             # Each new ball is added to the Ball list so
117             # it can be accessed by the animation handler.
```

```
118                 ballList.append(ball)
119
120         # This is the code for the quit Button handling. This
121         # function will be passed to the quitButton so it can
122         # be called by the quitButton when it wasPressed.
123         def quitHandler():
124                 # close the window and quit
125                 print("Good Bye")
126                 root.destroy()
127                 root.quit()
128
129         # Here is where the quitButton is created. To create
130         # an object we write
131         # objectReference = Class(<Parameters to Constructor>)
132         quitButton = tkinter.Button(fram, text = "Quit", command=quitHandler)
133         quitButton.pack()
134
135         # This is another example of a method call. We've been doing
136         # this all semester. It is an ontimer method call to the
137         # TurtleScreen object referred to by screen.
138         screen.ontimer(animate)
139
140         tkinter.mainloop()
141
142 if __name__ == "__main__":
143         main()
```