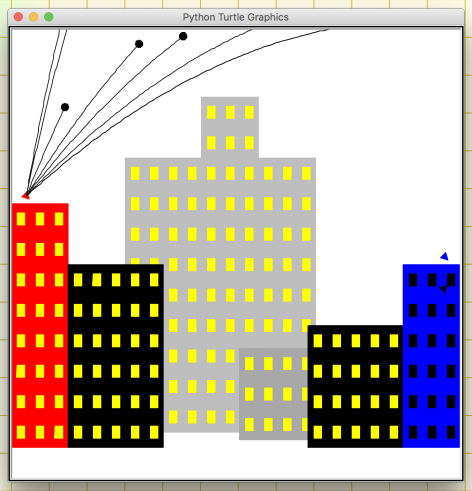
7

Monkey Wars

Introduction

Monkey Wars



This is a very old arcade game, consisting of two monkeys on top of two skyscrapers throwing rocks at each other.
Players race to aim and fire as quickly as possible to hit either the other player or the building that the other player is standing on. As the building take damage the lights on the building go out.

We are going to try something different today. I'm not going to give you any code — well only a little code — but instead I'm just going to say what I want the game to do and your job is to generate the code. I expect you to get stuck in (many) places — and would like you to ask each other and me, and hopefully we can work out the details together.
As before we have broken up the task in separate stages:

① Design the city

The city consists of:

- A red building on the left with random height of 2 to 10 floors and 3 windows wide.
- A blue building on the right with random height of 2 to 10 floors and 3 windows wide.
- Random building in between the red and blue building with taller building in the middle.

② The Players

- Draw two players on top of each building.
- We will create two turtles — one for each player.

③ Throwing Rocks

- To throw a rock we are going to use our projectile motion code from two weeks ago.

CoderDojo, Tramore, Waterford. (kmurphy@wit.ie)

1 Design the City

Creating the city will take us some time, and to simplify development we will change some of the default `turtle` settings.

Create file called `city.py` and insert the following code.

Monkey_Wars.py

```
1  import turtle
2  import math
3  import random
4
5  screen = turtle.Screen()
6  screen.setup (width=600, height=600, startx=0, starty=0)
7  screen.setworldcoordinates(-300,-10,300,500)
8
9  WINDOW_WIDTH = 25
10 FLOOR_HEIGHT = 40
11
12 bob = turtle.Turtle()      # draws the city
```

This code

- Loads in libraries that we will need.

(line 5) Creates a screen in which the turtles can move and draw.

(line 6) Moves the screen to the top left of the computer screen (so hopefully it won't be displayed on top of your code).

(line 7) Changes the coordinates of the screen so that the bottom left corner is $(-300, -10)$ and the top right corner is $(300, 500)$. This will make drawing of the building easier.

- In lines 9 and 10 we create variables that control the height and width of buildings.

(line 12) Finally we create our hard working turtle, `bob`.

Write a function that draws a filled rectangle at a given location

Now to draw a building we need to specify:

- The position of the building — we will use the bottom left corner as the anchor point.
- The height in number of floors.
- The width in number of windows in each floor.
- The colour — defaults to `black`.
- And draw in the windows (in `yellow`).

So we can draw a building by just drawing rectangles — so we will use function `draw_rectangle` for all drawings.

Write a function that draws a building using the following line.

This should draw a building with default colour of `black` consisting of `floors` floors and `windows` windows per floor.

> **What does the parameter `health` do?**
>
> I want to use the parameter `health` to represent the health of the building and to control how many lights as the building sustains more damage. We will talk how this can be implemented during the session.

Write a function that draws the city using the following code. The `screen.tracer(0)` line allows the city to be drawn before displaying it.

```
15 def draw_city():
16     screen.tracer(0)
17     # do work
18     screen.tracer(1,1)
```

2 The Players

The first stage of the game is done, so it is a good idea to save this and continue to work on the game using a different file — save your code and change the file name to `players.py`

The players are both turtles — lets called them `red` and `blue` so we write

```
13 red = turtle.Turtle()      # red player
14 blue = turtle.Turtle()     # blue player
```

- Place the players on top of their buildings:
  - So we need to remember how tall both the red and blue building are! Go back to your `draw_city` function and see where you generated a random number for the height of each building. Save those values to `red.floors` and `blue.floors`.
- Point both players (turtles) facing up — so if they fire they will damage their own building!
- Add event listeners so that "`q`" and "`a`" and used to aim the red player, and "`,`" and "`.`" — these are the "<" and ">"' keys — are used to aim the blue player.
  - The event listening code is the same as we used in the Graphical Take Away game, so look over previous worksheets if — horror of horrors — you have forgotten anything.
- Add events listeners so that "`z`" and "`m`" fire the red and blue player rocks — to avoid duplicate code we are going to have a single fire function for both players as shown in the following code.

Insert the following functions into your program and verify that the key events work as expected. You need a function `fire(player)` to check if your code works at this point.

```
93
94  def redUp():
95      red.left(2)
96      print ("Player red heading is now %s" % red.heading())
97
98  def redDown():
99      red.right(2)
100     print ("Player red heading is now %s" % red.heading())
101
102 def redFire():
103     print ("Player red fired")
104     fire(red)
105
106 def blueUp():
107     blue.left(2)
108     print ("Player blue heading is now %s" % blue.heading())
109
110 def blueDown():
111     blue.right(2)
112     print ("Player blue heading is now %s" % blue.heading())
113
114 def blueFire():
115     print ("Player blue fired")
116     fire(blue)
```

3 Throwing Rocks

Next we need to deal with the rocks flying through the air! This has some tricky bits so I'm not going to do this here but talk about it in the sessions.

Write the skeleton of the function `fire` as given below and we will fill in the details in the session.

```
1  def fire(player):
2      # create a new turtle to represent the rock
3
4      # move rock to player position and heading
5
6      # use projectile logic to get vertical and horizontal speed
7
8      # animate rock movement
9      while 1:
10         # update rock position
11         # check if rocks moved passed city
12         # check if collision with buildings
13         # check if passed screen
```