pyGame Lecture #4

(Examples: rain, rain2)

LISTS IN PYGAME

I. Power of Lists

Games are more fun if there are many objects on the screen. Even some of the more basic games have many objects on the screen at the same time, such as Asteroids. Naturally, it would be very difficult to store each object in a separate variable. In Python, lists are very flexible. Not only can we store a list of integers or Strings, but we can also store a list of lists. This becomes very handy because most objects we want to draw on a pyGame screen consist of multiple attributes. For example, consider a simple circle object. We must know its x and y coordinate, its radius and its dx and dy for movement. We could store these five values in a single list and the list of five values represents a single circle object. If we want several circle objects, then we can simply create a list of lists.

II. Rain Example

In this example we'll simply run a simulation of rain. To simplify things, each rain drop will simply be a randomly colored circle from a random x coordinate falling from the top of the screen. We'll add several rain drops every frame so that we have lots of them!

For each raindrop, we'll store four variables: x, y, dx and dy, in that order in a list. Thus, we have a list called item, then item[0] is the x-coordinate of the rain drop, item[1] is the y-coordinate, item[2] is the change in x between frames, and item[3] is the change in y between frames.

For this example, two functions will be very useful for us:

```
def move(items):
    for item in items:
        item[0] += item[2]
        item[1] += item[3]

def removeUseless(items):
    for item in items:
        if item[1] > SCREEN_H:
            items.remove(item)
```

The first function takes in a list of lists, items. It simply changes the x and y coordinate of each item in the list items by its corresponding dx and dy values.

The second removes each item in the list that has fallen below the bottom of the screen. The for loop that allows us to iterate through each item in a list is very useful in helping us write clean code for these two functions.

Writing our rain simulation becomes much easier with these two functions as utilities. Here is the plan for the code inside our main loop:

1. Take care of the user quitting.

2. Determine the number of new drops to add.

3. Add each drop to our list of drops.

4. Draw each drop in the simulation and update the display.

5. Move the drops.

6. Remove the drops that have gone off the bottom of the screen.

Before our main game loop we initialize our list of drops as follows:

```
rain = []
```

Here is how we add new drops inside the game loop (steps 2 and 3):

```
    numNewRain = random.randint(1, 10)
    xvals = set()
    while len(xvals) < numNewRain:
        x = random.randint(1, SCREEN_W)
        xvals.add(x)
    for val in xvals:
        rain.append([val,0,0,DY])
```

We haven't seen sets yet, but they are useful in this instance to make sure that no two drops start at the identical place. A set is like a list but doesn't store repeats. If you add a number to a set that is already in the set, then no change occurs ot the set. Thus, in the code above, we continue adding random values to the set xvals until the set contains the appropriate number of elements, numNewRain. Once we know the x values of each new raindrop, we add these rain drops. For this simulation, you'll see that all the raindrops fall at the same rate and have no change in x. (This is something very easy to change. You should try it out to see how the rain changes!)

Now, we can draw all the drops (and assign each a random color each frame) as follows:

```
    for item in rain:
        r = random.randint(0,255)
        g = random.randint(0,255)
        b = random.randint(0,255)
        pygame.draw.circle(DISPLAYSURF, pygame.Color(r,g,b),
                           (item[0],item[1]), RADIUS, 0)
```

Each of these codes can be checked for either a KEYDOWN event or a KEYUP event.

Putting it all together, here is the whole program:

```
# Arup Guha
# 7/15/2015
# "Rain" simulation - uses lists

import random
import math
import time
import pygame, sys
from pygame.locals import *

# Useful Constants
SCREEN_W = 1000
SCREEN_H = 600
DY = 5
RADIUS = 10
BLACK = pygame.Color(0,0,0)
BLUE = pygame.Color(0,0,255)


# This function handles moving each item listed in items.
def move(items):
    for item in items:
        item[0] += item[2]
        item[1] += item[3]
```

```
# This function removes all items that will never be visible again.
def removeUseless(items):
    for item in items:
        if item[1] > SCREEN_H:
            items.remove(item)

pygame.init()
DISPLAYSURF = pygame.display.set_mode((SCREEN_W, SCREEN_H))
pygame.display.set_caption("Let it rain!")
clock = pygame.time.Clock()

rain = []

while True:

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    # Calculate number of new drops to add this iteration, then randomly
    # generate that many unique values.
    numNewRain = random.randint(1, 10)
    xvals = set()
    while len(xvals) < numNewRain:
        x = random.randint(1, SCREEN_W)
        xvals.add(x)

    # Add each if these items into the list rain.
    for val in xvals:
        rain.append([val,0,0,DY])

    DISPLAYSURF.fill(BLACK)

    # Draw each raindrop individually.
    for item in rain:
        r = random.randint(0,255)
        g = random.randint(0,255)
        b = random.randint(0,255)
        pygame.draw.circle(DISPLAYSURF, pygame.Color(r,g,b),
                           (item[0], item[1]), RADIUS, 0)

    pygame.display.update()

    # Move the drops for the next iteration and remove useless ones.
    move(rain)
    removeUseless(rain)

    clock.tick(30)
```

III. Rain 2 Example

In this example, we introduce blits. Instead of drawing a circle for each raindrop, we'll use a .png image of a raindrop and draw that instead for each raindrop. Since the image we use will be fairly big, we'll just produce a random raindrop once every 10 frames. In order to accomplish our change, we will make only modest changes to the previous example. First, we have to load the image:

```
drop = pygame.image.load("raindrop.png")
```

To add one random drop every 10 frames we do the following:

```
if step%10 == 0:
    x = random.randint(1, SCREEN_W)
    rain.append([x,0,DX,DY])
```

Finally, the way we draw blits is as follows:

```
for item in rain:
    DISPLAYSURF.blit(drop,(item[0], item[1]))
```

The blit function called on the display surface just takes in the image and its top left corner.

The next lecture, about detecting mouse input will have another example with lists in pyGame. In fact, most of the future examples for pyGame will contain lists.