Python: Animation

Computer Science ICS20

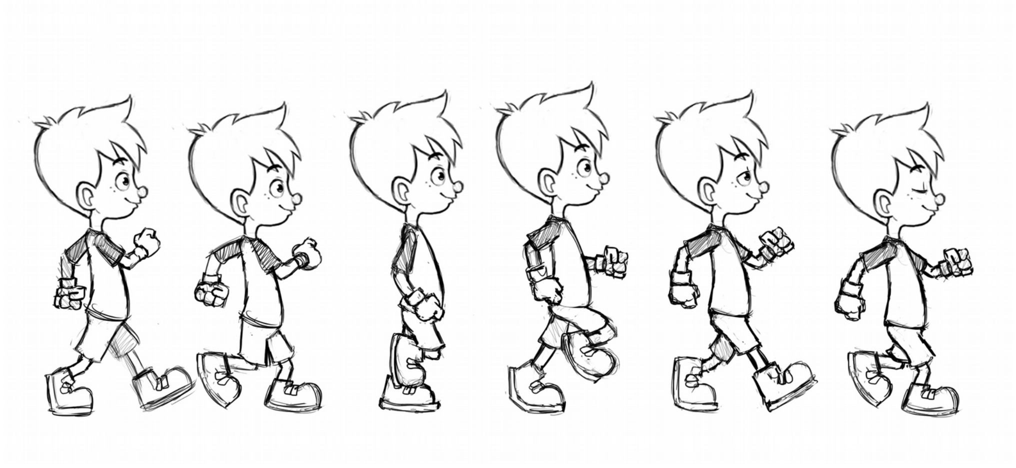
The Illusion of Motion

As we saw in the previous introduction to graphics, we are able to change the colour of certain pixels on the screen through the use of various pygame.draw functions. You will have noticed that it is important to specify the correct location of the objects to be drawn otherwise they may be off the screen and not visible.

You will also recall that we are repeatedly drawing objects in a while loop at a rate of 60 fps. In this section we will use this idea to animate our objects.

Animation is simply the illusion of movement by drawing pictures that change slightly from one frame to the next. This can be achieved in many ways such as by flipping the pages of a book that contains individual hand drawn pictures (cel animation) or taking photos of an object after moving it slightly (stop animation) or taking individual photos very quickly and playing them back just as fast (movies)

All of these techniques employ the same idea of repeatedly displaying an object that is in a slightly different position in each frame.



Animating Using PyGame

Since we already have template code that repeatedly draws basic shapes at 60 fps most of the work is already done for us. All that we need to do is make sure that for every iteration of the loop, we move objects that we want to animate by a small amount.

Let’s take a look at our template code again. Within the loop we’ll add some code to draw a rectangle of width=50 and height=200 at a location of (100, 50) on the screen. We can do this by the following line of code:

```
pygame.draw.rect(window, red, [100, 50, 50, 200])
```

Note that the last parameter can be either a list or a tuple.

If we run this code you should see a stationary red rectangle on the screen even though we are actually redrawing the rectangle 60 time every second. The reason that it is not moving is that we are constantly drawing it at the same position, over and over again.

```
import pygame
pygame.init()     # initializes the graphics module
window = pygame.display.set_mode((800,600))     # define window size
pygame.display.set_caption('PyGame Animation') # title of program that
                                                # appears on window frame
clock = pygame.time.Clock()     # used to track time within the game (FPS)
quit = False
while not quit:                         # main program loop
    for event in pygame.event.get():    # check if there were any events
        if event.type == pygame.QUIT:   # check if user clicked the upper
            quit = True                 # right quit button

      # your code that draws to the window goes here
      # draw a red rectangle
      pygame.draw.rect(window, red, [100, 50, 50, 200])

    pygame.display.update()         # refresh your display
    clock.tick(60)                  # wait a certain amount of time that
                                    # ensures a frame rate of 60 fps
pygame.quit()                       # shutdown module
```

Let's now try to animate the rectangle by having it move horizontally across the screen. As you can probably guess, this means that we have to change the x position of the rectangle's location. But simply changing the x value from 50 to 51 will only repeatedly draw the rectangle at the new x-value. What we need is to make the constant value into a variable and have the variable change on its own when the program is run. Once again, since we are in a while loop already all we need to do is the following:

```
pygame.draw.rect(window, red, [x, 50, 50, 200])
x = x + 1
```

Of course the variable x must be defined and initialized to some starting position before the while loop. After doing this, running the program will smoothly move the rectangle across the screen and eventually off the screen as x becomes larger than the window width.

Speed and Direction

In the example above the rectangle appeared to move across the screen at a given speed. (Can you calculate the speed with which it is moving?)

But what if we wanted to change the speed of the moving rectangle either by slowing it down or speeding it up?

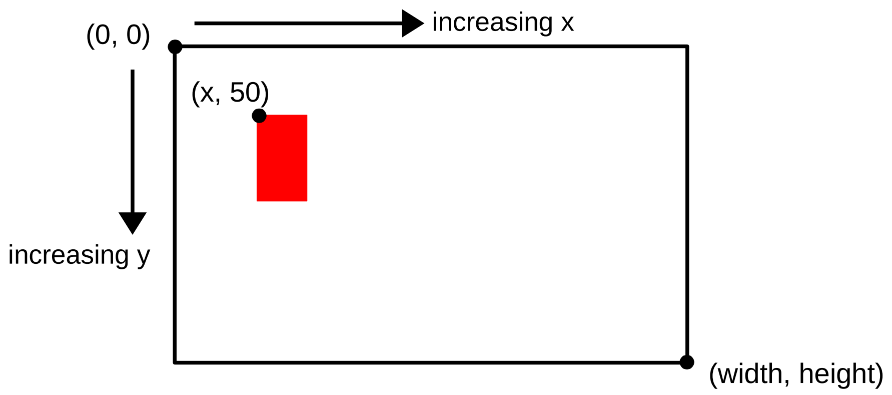
Looking back at our code we see that every frame changes the position of the x-position of the rectangle by exactly one pixel:

```
        x = x + 1
```

So the “1” actually represents the speed of the motion. It would be better to rewrite the above line like this:

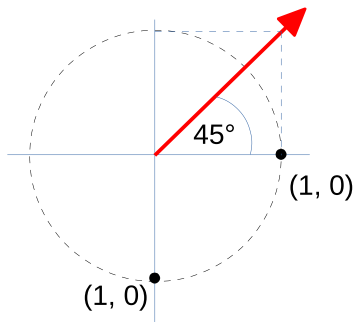
```
        x = x + x_speed
```

where the new variable **`x_speed`** can represent the speed of the object in the x direction. Once again, this variable also needs to be defined and initialized before the start of the while loop. Try to change the value of x_speed to see how this changes the animation. What happens when x_speed is negative? What happens if x_speed is a decimal number?



Similarly we can also move vertically by defining a y_speed variable and adding it to the y-position of the rectangle. If the speed of the vertical component is the same as the horizontal component, the object will move at a 45° angle down and to the right.

By combining the x_speed with the y_speed we can move the object in any direction that we want. So direction can be described in terms of its x and y components or it can be described by an angle. The picture below shows the relationship between the two ways of describing direction:



You can convert between angles and directional components in the following way:

x,y directions to angle:

$$\theta = atan\left(\frac{y}{x}\right)$$

angle to x,y directions:

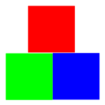
$$x = cos(\theta)$$
$$y = sin(\theta)$$

The sin(), cos() and atan() functions are available in the Python math module.

Exercises

For the following animations your frame rate should be 60 fps.

1. Write a Python program that animates a blue circle of radius 50 pixels so that it starts at location (100, 50) and moves horizontally across and stops when it reaches the right side edge of the window. The circle should move at a rate of 1 pixel per frame.
2. Write a Python program just like in question 1 but this time, instead of a circle, draw and animate the object shown below. It is made from three 100x100 squares in a triangle arrangement.



3. Write a Python program that animates a blue circle of radius 50 pixels so that it starts at location (100, 50) and moves diagonally across the window at a -45° angle.
4. Write a Python program that animates a blue circle of radius 50 pixels so that it starts at location (100, 50) and moves horizontally across the screen.
   a) Extend your program so that when the right edge of the circle hits the right edge of the window, the circle should "bounce" off the edge and move in the opposite direction
   b) Extend your program from part a) so that when the left edge of the circle hits the left edge of the window, the circle should "bounce" off and again move in the opposite direction.
5. Extend question 4 to add a vertical component to your movement so that the circle moves at an angle of 45°. Make sure to "bounce" the circle off the top and bottom edges of the window just like in question 4.
6. Modify question 5 by randomly choosing the initial position, speed and direction of the circle.
7. Add a second circle of a different colour and size with a randomly chosen initial position, speed and direction. You should now have two balls bouncing around the edges of your window.