pygame

Michele Pratusevich

February 24, 2014

1 Pygame

```
import pygame
```

Pygame docs: https://www.pygame.org/docs/index.htmlA Python **module** that is mostly used to make games, but is also a great way to make GUIs in Python. It relies on the ideas of object-oriented programming, classes, and objects.

1.1 A few concepts we need before using PyGame

Sizes and locations

- Everything is measured in pixels. A pixel is a dot on the screen.
- The entire screen is an x, y coordinate system with the TOP LEFT corner being 0, 0

Setting up the screen

```
main_screen = pygame.display.set_mode((400, 400))
main_screen.fill((255,255,255))
```

RGB color

- Computer screen (rendered) color is described by three numbers - Red, Green, and Blue (RGB).
- Triplet (R, G, B) corresponds to the color, where each number is from 0 to 255.
- For example: (255, 0, 0) is red; (0, 255, 0) is green; (255, 255, 255) is white; (0, 0, 0) is black

Event polling / game loop

- Instead of responding immediately when an event happens, hold a master "list of events" and based on what kind it is, respond to it.
- A "game loop" or an "event loop" is used to access events. Basically, it is an infinite loop that keeps going for the entire time the game is running, and only stops when you stop the game.

An example of a game loop:

```
while True:
    ev = pygame.event.poll()
    if ev.type == pygame.QUIT:
        sys.exit()
    if ev.type == pygame.MOUSEBUTTONDOWN:
        x, y = ev.pos
        # do something with the click
    pygame.display.flip()
```

Rectangles

- Everything in pygame is organized by rectangles. You draw shapes and buttons into rectangles, you click on rectangles, you work with rectangles. A rectangle represents a space where you will draw something.

The four numbers represent the (top left x coordinate, top left y coordinate, width, height)

```
button_rec = pygame.Rect(100, 200, 50, 20)
```

Surface

- The type representing "image on the screen." You can make it colorful or actually add an image to it.

For an image:

```
buttonimg = pygame.image.load('baby-bamba.jpg')
```

- The default size will be the size of the image. There is no way to resize it - if you need to resize it, use a photo-editing program like Gimp.

For a square of black color:

```
button_sq = pygame.Surface([20, 20])
```

For a square of another color:

```
button_sq = pygame.Surface([20, 20])
button_sq.fill((255, 0, 0))
```

Blit

- Think of it like "drawing on the screen." What it actually means is "change the pixel color inside a rectangle"

```
button_rec = pygame.Rect(100, 100, 20, 20)
button_sq = pygame.Surface([20, 20])
main_screen.blit(button_sq, button_rec)
```

Flip

- Redraw the entire window. Very costly, so you shouldn't do this very often.

```
pygame.display.flip()
```

A very basic example of a Pygame program is shown here:

In []:
```
import pygame
import sys
```

```
if __name__=="__main__":
    pygame.init()
    main_screen = pygame.display.set_mode((400, 400))
    main_screen.fill((255,255,255))
    button_rec = pygame.Rect(100, 100, 20, 20)
    button_sq = pygame.Surface([20, 20])
    main_screen.blit(button_sq, button_rec)

    while True:
        ev = pygame.event.poll()
        if ev.type == pygame.QUIT:
            sys.exit()
        if ev.type == pygame.MOUSEBUTTONDOWN:
            x, y = ev.pos
            if button_rec.collidepoint(x, y):
                print "you clicked me!"
        pygame.display.flip()
```

And a commented version is here:

In []:
```
import pygame
import sys

# custom classes here


# functions not related to classes here


# main method
if __name__=="__main__":
    # start the pygame module
    pygame.init()

    # the two numbers represent the X and Y window size (in pixels)
    main_screen = pygame.display.set_mode((400, 400))

    # set the background color of the window - remember RGB
    main_screen.fill((255,255,255))

    # make your rectangles and surfaces here
    button_rec = pygame.Rect(100, 100, 20, 20)
    button_sq = pygame.Surface([20, 20])

    # blit the button we created at the top here, because it will not cha
    # if it will change, blit it inside the game loop
    main_screen.blit(button_sq, button_rec)

    # game / event / main loop
    while True:
        # get the last event that happened
        ev = pygame.event.poll()

        # if you clicked the X button to exit the window
        if ev.type == pygame.QUIT:

            # close the window
            sys.exit()

        # if you clicked down with the mouse
        if ev.type == pygame.MOUSEBUTTONDOWN:
            # x, y represents the position of the click in the screen
            x, y = ev.pos

            # do something with the click

            # check if the position of the click is inside the button
```

```
            if button_rec.collidepoint(x, y):
                print "you clicked me!"

        # every game loop, make sure you update the display, but only ONCE
        pygame.display.flip()
```

1.2 More things

Making a text label

```
label_rec = pygame.Rect(50, 50, 200, 30)

# first argument is a filename (none if you want the default)
# second argument is the font size
orderlabel = pygame.font.Font(None, 30)

# Method: render(text, antialias, color, background=None)
label = orderlabel.render("Pillow", 1, (0, 0, 0), (255, 255, 255))

main_screen.blit(label, label_rec)
```

Screens

- You can only have one window at a time in pygame, so everything you do will have to be on the same screen - no popup windows!