Lab 18: PyGame

Apr 5, 2018

Main Event

1. Introduction

Pygame

If you have not done so already, you should first setup and install Pygame. Open terminal and type the following command:

```
$> pip install pygame
```

The Pygame documentation is a good resource for learning and for reference.

Pythonw

For best results, Python programs that import and use Pygame should be run from the command line. In doing so, they should be run using `pythonw` instead of `python`. Thus, non-graphical programs would be run as follows:

```
$> python myprogram.py
```

Graphical programs, on the other hand:

```
$> pythonw myprogram.py
```

Command line basics

Remember the basics of command line interaction:

- `cd` changes your current location
- `ls` lists the files in your current directory (`dir` for Windows users)

To run a program, you should either `cd` to the directory that contains the program, then run it:

```
$> cd a/b/c
$> python d.py
```

or pass the *absolute* path of the program directly to python:

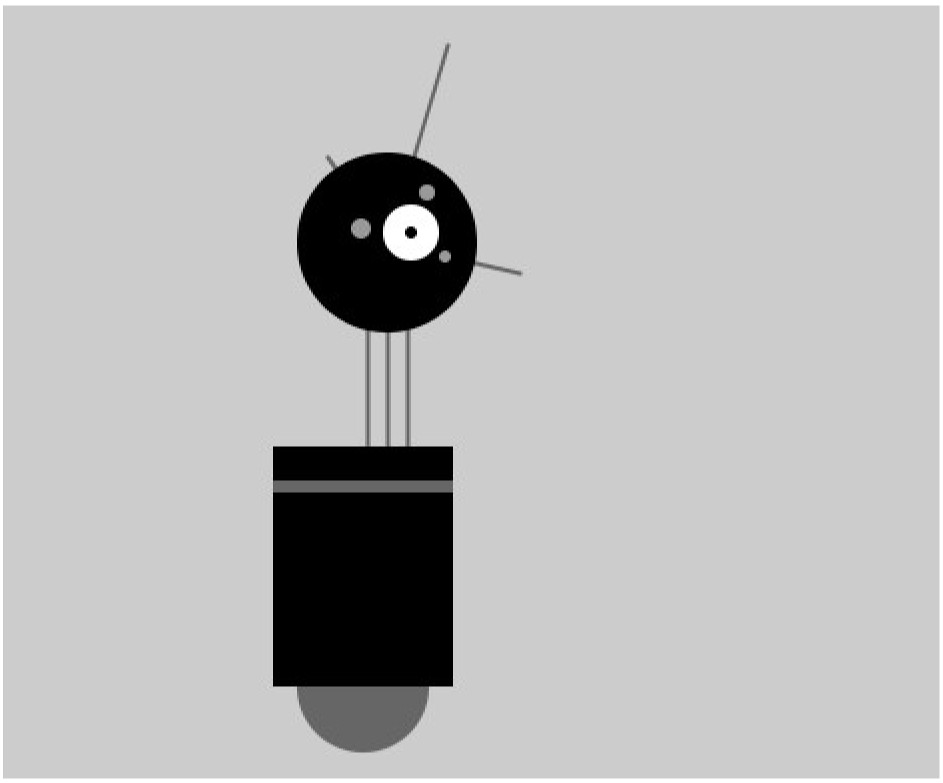
```
$> python a/b/c/d.py
```

If there are spaces in either your directory name or your Python file, you will need to either escape the spaces, or put everything in quotes. These examples are equivalent:

```
$> python "path/to my favourite/directory in the world/myprogram.py"
$> python path/to\ my\ favourite/directory\ in\ the\ world/myprogram.p
```

2. Picasso

Build the following robot:



Notice that it's essentially a series of shapes that we have covered. You can "fake" the overlays by ordering your commands in the correct way. Your result doesn't have to be exactly as shown, the focus here is on writing shapes and getting comfortable with Pygame.

Solution:

```
# https://www.openprocessing.org/sketch/306465

import sys
import pygame

pygame.init()
surface = pygame.display.set_mode((720, 480))

background = (137, 137, 137)
body = (0, 80, 3)
trim = (232, 225, 16)
eyes = (20, 255, 0)
nose = (0, 0, 0)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.display.quit()
            sys.exit()
    surface.fill(background)

    # neck
    pygame.draw.line(surface, trim, (266, 257), (266, 162), 2)
    pygame.draw.line(surface, trim, (276, 257), (276, 162), 2)
    pygame.draw.line(surface, trim, (286, 257), (286, 162), 2)

    # antenae
    pygame.draw.line(surface, trim, (276, 155), (246, 112), 2)
    pygame.draw.line(surface, trim, (276, 155), (306, 56), 2)
    pygame.draw.line(surface, trim, (276, 155), (342, 170), 2)

    # wheel
    pygame.draw.circle(surface, trim, (264,377), 33)

    # trunk
    pygame.draw.rect(surface, body, pygame.Rect(219, 257, 90, 120))
    pygame.draw.rect(surface, trim, pygame.Rect(219, 274, 90, 6))
    # head and nose
    pygame.draw.circle(surface, body, (276, 155), 45)
    pygame.draw.circle(surface, nose, (288, 150), 14)
    pygame.draw.circle(surface, trim, (288, 150), 3)

    # eyes
    pygame.draw.circle(surface, eyes, (263, 148), 5)
    pygame.draw.circle(surface, eyes, (296, 130), 4)
    pygame.draw.circle(surface, eyes, (305, 162), 3)

    pygame.display.update()
```

3. Particle remix

Recall our lab on particles. We essentially built lots of ASCII characters and had them move around the screen. It wasn't pretty. This is our chance to correct that! Give your old particle code a new life in Pygame:

- Make a single particle move diagonally across the screen.
- Generate a set of particles with random colors, random sizes, and random speeds. It may help to create a `display` method within the Particle class that calls Pygame's draw.circle. To do so would require the method take a parameter of type Surface (the type returned by display.set_mode; or have the Particle class itself hold a Surface as an attribute.

  Finally, in event loop (the while-loop in the global context), it is then just a matter of iterating through your list of Particles, and invoking `move` and `display` on each.
- Make the particles stay within the bounds of your box. When particles were ASCII characters on a Board structure, if their coordinates were out of the board bounds we would get an error. Not in Pygame: once they are out of the screen boundaries they disapear. That's not fun at all.

  Instead, make them either "wrap" back around onto the canvas, or bounce off the walls—the latter is probably cooler.
- Get the particles to follow your mouse. There are multiple ways of doing this, but perhaps the best is to update a Particle's position based on some function of its velocity *and* the current mouse position. Remember,

  ```
  pygame.mouse.get_pos()
  ```

returns a tuple containing the *x* and *y* coordinates, respectively, of the current mouse position.

Other than the particles reacting to your mouse, there is no wrong way to tackle this question.

Solution:

```
import sys
import random

import pygame

class Pair:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y

        return Pair(x, y)

    def astuple(self):
        return (self.x, self.y)

class Particle:
    def __init__(self, position, velocity):
        self.position = position
        self.velocity = velocity
        self.color = random.choices(range(256), k=3)
        self.radius = random.randrange(5, 20)

    def move(self):
        self.position += self.velocity

    def display(self, surface):
        pygame.draw.circle(surface,
                           self.color,
                           self.position.astuple(),
                           self.radius)
```

```python
pygame.init()
surface = pygame.display.set_mode((640, 480))

# Create the particles...
particles = []
for i in range(10):
    x = random.randrange(surface.get_width())
    y = random.randrange(surface.get_height())
    position = Pair(x, y)

    while True:
        x = random.choice(range(-1, 2))
        y = random.choice(range(-1, 2))
        velocity = Pair(x, y)
        if velocity.x != 0 or velocity.y != 0:
            break

    particles.append(Particle(position, velocity))

# ... and watch them move!
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    surface.fill((0, 0, 0))
    for p in particles:
        too_right = p.position.x + p.radius > surface.get_width()
        too_left = p.position.x - p.radius < 0
        if too_right or too_left:
            p.velocity.x *= -1

        too_low = p.position.y + p.radius > surface.get_height()
        too_high = p.position.y - p.radius < 0
        if too_low or too_high:
            p.velocity.y *= -1

        p.move()
        p.display(surface)

    pygame.display.update()
```

Introduction to Computer Science

Introduction to Computer Science
jerome.white@nyu.edu

jerome-white

Learning computer science concepts through practice.