Flower Pot Pygame

By Gray and Jacob

Demonstration

Class and Object Design

- Based on the Sprite parent class (given by Pygame module)
- Mouse class derived from sprite class
  - Has image and rect. (rectangle) attributes in order to set the mouse to certain images depending on certain methods
- Each method is a different state for the mouse depending on what the user clicks on in the main game. The state later helps trigger a dictionary connected to images

```
# classes for our game objects

#Setting up mouse sprite
class Mouse(pg.sprite.Sprite):
    def __init__(self):
        pg.sprite.Sprite.__init__(self)
        self.image , self.rect = load_image("Jason.png", -1)
        self.image = pg.transform.scale(self.image, (50,50))
        self.state = "Jason"



#Changing the mouse image based off of collison
    def water_pressed(self):
        self.image , self.rect = load_image("Water.png", -1)
        self.state = "Water"

    def not_pressed(self):
        self.image , self.rect = load_image("Jason.png" , -1)
        self.state = "Jason"

    def fertalize_pressed(self):
        self.image , self.rect = load_image("Fertalize.png", -1)
        self.state = "Fertalizer"

    def seeds_pressed(self):
        self.image, self.rect = load_image("Plant.png", -1)
        self.state = "Seeds"

    def sell_pressed(self):
        self.image, self.rect = load_image("Sell.png", -1)
        self.state = "Sell"
```

```
    def update(self):
        pos = pg.mouse.get_pos()
        self.rect.center = pos


 #Setting up classes that run as objects that players interact with
class Water(pg.sprite.Sprite):

    def __init__(self,image,position):
        pg.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image(image, -1)
        self.rect.center = position

class Fertalize(pg.sprite.Sprite):

    def __init__(self,image,position):
        pg.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image(image, -1)
        self.rect.center = position

class Sell(pg.sprite.Sprite):

    def __init__(self,image,position):
        pg.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image(image, -1)
        self.rect.center = position

class Plant(pg.sprite.Sprite):

    def __init__(self,image,position):
        pg.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image(image, -1)
        self.rect.center = position
```

```
class Pots(pg.sprite.Sprite):

    def __init__(self,position):
        pg.sprite.Sprite.__init__(self)  # call Sprite initializer
        self.image , self.rect = load_image("empty_pot.png" , -1)
        print(self.image)
        self.image = pg.transform.scale(self.image, (300,300))
        self.growth_stage = 0
        self.score = 0
        screen = pg.display.get_surface()
        self.area = screen.get_rect()
        self.rect.center = position
        #Dictonary to make the growth stages line up with an accesable index
        self.growth_dict = {
        0:"empty_pot.png",
        1:"1.png",
        2:"2.png",
        3:"3.png"
        }

class Warning(pg.sprite.Sprite):

    def __init__(self,image,position):
        pg.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image(image, -1)
        self.rect.center = position


def main():
    # Initialize Everything
    pg.init()
    screen = pg.display.set_mode((1000, 750), pg.SCALED)
    pg.display.set_caption("Gay Gardening")
    pg.mouse.set_visible(False)
```

Future Work

- Later on down the line we intend to add a score reset feature as well as accompanying music to go with the game
- We also intend on fixing up the help function to be a pop-up when the user clicks on the help symbol in the top right corner
- We also may hope to be able to add a starting screen to the game
- As a reach, if we have time, we have interest in changing the point system to a currency

Overall Team Dynamics

- Overall the two of us stuck with pair programming during the in-class work sessions (primarily on Gray's MacBook).
- Whenever one would find issues, they'd report it to the other person who would take a look and resolve the issue if possible.
- In the event that neither could solve it, whoever was available would resort to TA assistance.
- Some tasks were delegated to specific people
- For example, Gray was in charge of drawing and inserting the assets used in the game while Jacob primarily worked on debugging select functions

What was Completed vs. What was Proposed

Completed

- The created game is functional and features the warning functions and growth stages, along with a functioning scoreboard that keeps track of how many plants have been sold
- All assets were drawn and added in a timely manner

Proposed

Minimum Functionality

- Score Reset Feature

Extensions

- Music
- Starting Screen
- Help Function
- Timer for Actions
- Sound Effects
- Currency
- Animations
- More Plants

What We Learned/Suggestions for Future Projects

Jacob:

- Pygame and other imported modules can be pretty frustrating to install and work with. Make sure to get help as soon as possible if you also have issues
- Don't underestimate how much time making even a simple game can take, time management is pretty crucial
- Make sure to organize your class and object code to make the programming part easier (it can get pretty complicated to read at times)

Gray:

- ~~Pygame makes me want to switch my major~~
- Understanding how to use pygame does not help that much with coding using pygame
- When programming have a layout in mind to understand where everything will fit
- I understand classes and objects in relation to practical work
- For future projects I will likely make more pseudo code before starting

Thank You!