L12-6 Final Project

In this final project, you will be creating Simon Says. Most of the components used to develop this game have been taught througout the modules. The game is made up of two python files which are the button and main py files. In this section you will be given the code to both files to complete the project. In the next section you will find the instructions to complete the missing pieces of code.

Please make sure to create new files then copy and paste the following code into your code editor.

You will also need to download these sounds for the game.

button.py

```
import random
import time
import pygame
pygame.init()


class Button(pygame.sprite.Sprite):
    def __init__(self): # Add given properties as parameters
        pygame.sprite.Sprite.__init__(self)
        # Initialize properties here



        self.image = pygame.Surface((230, 230))
        self.image.fill(self.color_off)
        self.rect = self.image.get_rect()
        # Assign x, y coordinates to the top left of the sprite
        self.rect.topleft = ()
        self.clicked = False

    '''
    Draws button sprite onto pygame window when called
    '''

    def draw(self, screen):
        # blit image here

    '''
    Used to check if given button is clicked/selected by player
    '''

    def selected(self, mouse_pos):
        # Check if button was selected. Pass in mouse_pos.

    '''
    Illuminates button selected and plays corresponding sound.
    Sets button color back to default color after being illuminated.
    '''
    def update(self, screen):
        # Illuminate button by filling color here

        # blit the image here so it is visible to the player

        # Play sound

        pygame.display.update()
        self.image.fill(self.color_off)
        screen.blit(self.image, (self.rect.x, self.rect.y))
        pygame.time.wait(500)
        pygame.display.update()
```

main.py

```
import pygame
import random
import time
from button import Button # By importing Button we can access methods from
the Button class

pygame.init()

clock = pygame.time.Clock()

# Constants
SCREEN_WIDTH = 500
SCREEN_HEIGHT = 500

SCREEN = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))

GREEN_ON = (0, 255, 0)
GREEN_OFF = (0, 227, 0)
RED_ON = (255, 0, 0)
RED_OFF = (227, 0, 0)
BLUE_ON = (0, 0, 255)
BLUE_OFF = (0, 0, 227)
YELLOW_ON = (255, 255, 0)
YELLOW_OFF = (227, 227, 0)

# Pass in respective sounds for each color
GREEN_SOUND = pygame.mixer.Sound("bell1.mp3") # bell1
RED_SOUND = pygame.mixer.Sound() # bell2
BLUE_SOUND = pygame.mixer.Sound() # bell3
YELLOW_SOUND = pygame.mixer.Sound() # bell4

# Button Sprite Objects
green = Button(GREEN_ON, GREEN_OFF, GREEN_SOUND, 10, 10)
red =
blue =
yellow =

# Variables
colors = ["green", "red", "blue", "yellow"]
cpu_sequence = []
choice = ""

'''
Draws game board
'''


def draw_board():
    # Call the draw method on all four button objects



'''
Chooses a random color and appends to cpu_sequence.
Illuminates randomly chosen color.
'''


def cpu_turn():
    choice = random.choice(colors)  # pick random color
    cpu_sequence.append(choice)  # update cpu sequence
    if choice == "green":
        green.update(SCREEN)
    # Check other three color options



'''
Plays pattern sequence that is being tracked by cpu_sequence
'''


def repeat_cpu_sequence():
    if(len(cpu_sequence) != 0):
        for color in cpu_sequence:
            if color == "green":
                green.update(SCREEN)
            elif color == "red":
                red.update(SCREEN)
            elif color == "blue":
                blue.update(SCREEN)
            else:
                yellow.update(SCREEN)
            pygame.time.wait(500)


'''
After cpu sequence is repeated the player must attempt to copy the same
pattern sequence.
```

```
The player is given 3 seconds to select a color and checks if the selected
color matches the cpu pattern sequence.
If player is unable to select a color within 3 seconds then the game is
over and the pygame window closes.
'''


def player_turn():
    turn_time = time.time()
    players_sequence = []
    while time.time() <= turn_time + 3 and len(players_sequence) <
len(cpu_sequence):
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONUP and event.button == 1:
# button click occured
                # Grab the current position of mouse here
                pos =
                if green.selected(pos): # green button was selected
                    green.update(SCREEN) # illuminate button
                    players_sequence.append("green") # add to player
sequence
                    check_sequence(players_sequence) # check if player
choice was correct
                    turn_time = time.time() # reset timer
                # Check other three options

    # If player does not select a button within 3 seconds then the game
closes
    if not time.time() <= turn_time + 3:
        game_over()


'''
Checks if player's move matches the cpu pattern sequence
'''


def check_sequence(players_sequence):
    if players_sequence != cpu_sequence[:len(players_sequence)]:
        game_over()


'''
Quits game and closes pygame window
'''


def game_over():
    pygame.quit()
    quit()


# Game Loop
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.display.quit()
            pygame.quit()
            quit()
    pygame.display.update()

    draw_board() # draws buttons onto pygame screen
    repeat_cpu_sequence() # repeats cpu sequence if it's not empty
    cpu_turn() # cpu randomly chooses a new color
    player_turn() # player tries to recreate cpu sequence
    pygame.time.wait(1000) # waits one second before repeating cpu
sequence

    clock.tick(60)
```

L12-7 Final Project Continued

button.py

Init Method

The init method is used to initialize properties when a new object is created. In this case, whenever a button is created we want the following properties to be intiailized in this order: color_on, color_off, sound, x and y.

Draw Method

The draw method will be used to draw the buttons onto the pygame screen. Blit self.image using the rect's x and y values onto the pygame screen.

Selected Method

The selected method will be used to check if the player has selected the button. Using self.rect.collidepoint check if button was selected. If so, return True otherwise return False.

Update Method

The update method will be responsible for turning the button on and off when it has been selected.

1. Using self.image.fill pass in the object's color_on property to illuminate the button.
2. Blit self.image using the rect's x and y values onto the pygame screen to make the changes visible to the player.
3. Play the object's sound property

main.py

Sounds and Objects

Each button will have a sound that will be played when it is selected. Before assigning each color a sound make sure you have downloaded the sound assets and added them to your project directory.

Now you will be creating the button objects that will be used to play the game. The green button object has already been created for you as reference. Create the remaining objects by passing in their respective values. Below you will find the x and y coordinates for each button object.

- red = 260, 10
- blue = 260, 260
- yellow = 10, 260

draw_board()

The draw_board function will be used to draw the buttons onto pygame screen. Call the draw method on all four button objects and pass in SCREEN as the argument.

cpu_turn()

Check which color was randomly chosen by the cpu and illuminate that color. The color green has been completed for you as reference.

player_turn()

Check to see if the player selected any of the four buttons by grabbing the mouse position of the button click. If any of the four buttons were selected do the following: illuminate the button, add the color to the player sequence, check if the player was correct and reset the timer. Green has been completed for you as reference.

Stretch Goals

Here are some other ideas that you can try implementing to improve the game:

- Add text to keep track of the player's score
- Add a reset option when player loses game rather than having the window close
- Add a visible timer on the pygame screen