MF_GUI

December 21, 2021

@author: Marcos Tulio Fermin Lopez

[ ]:
```
import time
import pygame
import sys
import plotter
import MF_Simulation_PIR as pir
import MF_Simulation_Camera as cam
import MF_Simulation_Antenna as ant
```

This Module creates a basic GUI to interact with the project files. It allows to run each technology simulation (individually) and plot the generated data from a single window.

[ ]:
```
#################################################################################
# Initial Parameters
#################################################################################
pygame.init()  # Initializes Pygame
FPS = 60  # Simulation Speed

# Screen Dimentions
SCREEN_WIDTH = 1400
SCREEN_HEIGHT = 922

# Loads the GUI backgounds and scales them to the screen dimentions
bg1 = pygame.image.load('images/menu/main1.jpg')
bg1 = pygame.transform.smoothscale(bg1, (1400, 922))
bg2 = pygame.image.load('images/menu/main2.jpeg')
bg2 = pygame.transform.smoothscale(bg2, (1400, 922))

# Set the screen mode
state = 0
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT), 0, 32)
clock = pygame.time.Clock()  # Initializes clock
screen_mode = 'title'   # Modes: title, menu
```

[ ]:
```
#################################################################################
# Title screen components
#################################################################################
```

```
""" Title Screen text """
# Define the font used for the title screen surface
title_font = pygame.font.SysFont('impact', 50)
# Render the simulation title surface
title_surface = title_font.render(
    "Marcos Fermin's Dynamic Traffic Lights Simulator", True, (255, 255, 255))
# Window Caption
pygame.display.set_caption(
    "Marcos Fermin's Dynamic Traffic Lights Simulator - EE Capstone Project -␣
 ↪Fall 2021")

# Create the button boxes font
box_font = pygame.font.SysFont('arial', 50)

# Render the menu surface buttons
menu_surface = box_font.render('Menu', True, (255, 255, 255))
quit_surface = box_font.render('Quit', True, (255, 255, 255))
cam_surface = box_font.render('Camera', True, (255, 255, 255))
PIR_surface = box_font.render('PIR', True, (255, 255, 255))
Antenna_surface = box_font.render('Antenna', True, (255, 255, 255))
Plot_surface = box_font.render('Plot Data', True, (255, 255, 255))

# Create the rings around the buttons
ring1 = pygame.Rect(590, 480, 120, 70)  # Ring position as a rectangle
clr1 = (255, 0, 0)

ring2 = pygame.Rect(590, 820, 120, 70)
clr2 = (255, 0, 0)

ring3 = pygame.Rect(560, 290, 170, 70)
clr3 = (255, 0, 0)

ring4 = pygame.Rect(585, 420, 120, 60)
clr4 = (255, 0, 0)

ring5 = pygame.Rect(560, 530, 190, 70)
clr5 = (255, 0, 0)

ring6 = pygame.Rect(560, 670, 190, 70)
clr6 = (255, 0, 0)

# Colors used in the rings when mouse collides with buttons
green = (0, 255, 0)
red = (255, 0, 0)

# Array of rings and circles
```

```
ringArr = [ring1, ring2, ring3, ring4, ring5, ring6]
clrArr = [clr1, clr2, clr3, clr4, clr5, clr6]
```

[ ]:
```
################################################################################
# Game Loop
################################################################################

run = True
while run:
    # We need to show different things depending on whether or not we're in␣
 ↪'title'
    # or 'menu' mode

    if state == 0:  # Photos
        screen.blit(bg1, (0, 0))
        screen.blit(title_surface, (160, 40))
    elif state == 1:
        screen.blit(bg2, (0, 0))

    for event in pygame.event.get():

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                if state == 0:
                    run = False  # Kill the game on escape key
                elif state == 1:
                    state = 0

        if event.type == pygame.QUIT:
            run = False  # Kill the game
            sys.exit()
            pygame.quit()
            print('\nShutting down the game')

    # Rings
    if state == 0:
        pygame.draw.rect(screen, clrArr[0], ring1, 2, 10)  # Menu
    if state == 1:
        pygame.draw.rect(screen, clr3, ring3, 2, 10)
        pygame.draw.rect(screen, clr4, ring4, 2, 10)
        pygame.draw.rect(screen, clr5, ring5, 2, 10)
        pygame.draw.rect(screen, clr6, ring6, 2, 10)

    pygame.draw.rect(screen, clrArr[1], ring2, 2, 10)  # Quit

    # Detects mouse collitions with the buttons and changes their ring color
    for i in ringArr:
```

```
            if i.collidepoint(pygame.mouse.get_pos()):
                clrArr[ringArr.index(i)] = (0, 255, 0)  # Green
            else:
                clrArr[ringArr.index(i)] = (255, 0, 0)  # Red

    if ring3.collidepoint(pygame.mouse.get_pos()):
        clr3 = green
    else:
        clr3 = red

    if ring4.collidepoint(pygame.mouse.get_pos()):
        clr4 = green
    else:
        clr4 = red

    if ring5.collidepoint(pygame.mouse.get_pos()):
        clr5 = green
    else:
        clr5 = red

    if ring6.collidepoint(pygame.mouse.get_pos()):
        clr6 = green
    else:
        clr6 = red

    # Click detection. Prints which technology simulation was selected
    if pygame.mouse.get_pressed()[0] and ring3.collidepoint(pygame.mouse.
↪get_pos()) and state == 1:
        print('Camera Button Pressed')
        cam.main()

    if pygame.mouse.get_pressed()[0] and ring4.collidepoint(pygame.mouse.
↪get_pos()) and state == 1:
        print('PIR Button Pressed')
        pir.main()

    if pygame.mouse.get_pressed()[0] and ring5.collidepoint(pygame.mouse.
↪get_pos()) and state == 1:
        plotter.plot_all()

    if pygame.mouse.get_pressed()[0] and ring6.collidepoint(pygame.mouse.
↪get_pos()) and state == 1:
        print('Antenna Button Pressed')
        ant.main()

    # Quit if quit button is pressed
```

```
    if pygame.mouse.get_pressed()[0] and ring2.collidepoint(pygame.mouse.
↪get_pos()):
        run = False
        sys.exit()
        pygame.quit()

    # Checks if any button is pressed and waits to update state
    if pygame.mouse.get_pressed()[0] and ring1.collidepoint(pygame.mouse.
↪get_pos()):
        state = 1
        time.sleep(0.5)

    # Blits button text surface
    if state == 0:
        screen.blit(menu_surface, (600, 485))  # Menu
    if state == 1:
        screen.blit(cam_surface, (570, 300))  # Camera
        screen.blit(PIR_surface, (610, 420))  # PIR
        screen.blit(Plot_surface, (570, 540))  # Plot
        screen.blit(Antenna_surface, (575, 680))  # Antenna

    # Blits quit button text surface
    screen.blit(quit_surface, (610, 825))  # Quit

    # Clock tick function for every frame
    clock.tick(FPS)
    # Updates the content of the entire screen
    pygame.display.flip()
```