2023

Curtin University

FOP Assessment Report

STAGE SHOW

AFNAN UR REHMAN

FOP Project Report

Overview

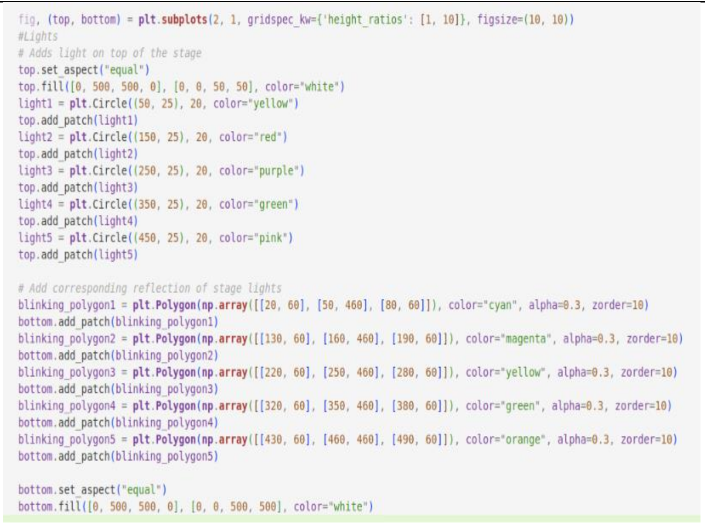
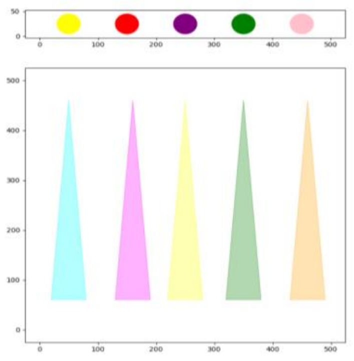
The Purpose of the program is to demonstrate a code showing a visualization of a stage party, indicating lights, stage, audience, people dancing in the background as backdrop, curtains on top of the stage with changing light showing the choreography on the stage. We have used different libraries like mat plot and NumPy in order to get output from the code.

User Guide

How to use your simulation (and parameter sweep code, if applicable)

1. Open python program. Use vim scene1.py for scenario 1.
2. install "Matplotlib.pyplot"," NumPy", "random" and "matplotlib. Image" and after installing, import these libraries.
3. After installing the specific libraries, copy the code provided below and paste it on vim scene1.py.
4. Come back to the terminal and use python3 scene1.py to run the program and display output.
5. Use vim scene2.py and vim scene3.py and repeat the above-mentioned steps.
6. You can run the mentioned codes by returning to the terminal and typing python3 scene2.py and scene3.py.

Traceability Matrix

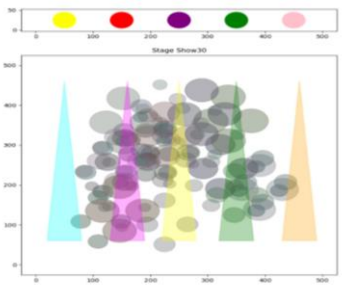
Feature	Code Reference	Test Reference
Lights		

```python
fig, (top, bottom) = plt.subplots(2, 1, gridspec_kw={'height_ratios': [1, 10]}, figsize=(10, 10))
#Lights
# Adds light on top of the stage
top.set_aspect("equal")
top.fill([0, 500, 500, 0], [0, 0, 50, 50], color="white")
light1 = plt.Circle((50, 25), 20, color="yellow")
top.add_patch(light1)
light2 = plt.Circle((150, 25), 20, color="red")
top.add_patch(light2)
light3 = plt.Circle((250, 25), 20, color="purple")
top.add_patch(light3)
light4 = plt.Circle((350, 25), 20, color="green")
top.add_patch(light4)
light5 = plt.Circle((450, 25), 20, color="pink")
top.add_patch(light5)

# Add corresponding reflection of stage lights
blinking_polygon1 = plt.Polygon(np.array([[20, 60], [50, 460], [80, 60]]), color="cyan", alpha=0.3, zorder=10)
bottom.add_patch(blinking_polygon1)
blinking_polygon2 = plt.Polygon(np.array([[130, 60], [160, 460], [190, 60]]), color="magenta", alpha=0.3, zorder=10)
bottom.add_patch(blinking_polygon2)
blinking_polygon3 = plt.Polygon(np.array([[220, 60], [250, 460], [280, 60]]), color="yellow", alpha=0.3, zorder=10)
bottom.add_patch(blinking_polygon3)
blinking_polygon4 = plt.Polygon(np.array([[320, 60], [350, 460], [380, 60]]), color="green", alpha=0.3, zorder=10)
bottom.add_patch(blinking_polygon4)
blinking_polygon5 = plt.Polygon(np.array([[430, 60], [460, 460], [490, 60]]), color="orange", alpha=0.3, zorder=10)
bottom.add_patch(blinking_polygon5)

bottom.set_aspect("equal")
bottom.fill([0, 500, 500, 0], [0, 0, 500, 500], color="white")
```

Description: The code includes the lights on top which are representing the stage lights. It also includes the code for the drop down lights, which I have demonstrated using the polygons function, which we learnt in the PracTest2. The test result has been demonstrated. I have specified colors of the polygon, which could be changed to any color. My goal is to demonstrate a change of color using loop function or read it through the csv file in the later parts of the assignment.

Smoke Machine

```python
#Code to add smoke effect on the stage
def smoke_effect():
    x = np.random.uniform(100, 400, size=50)
    y = np.random.uniform(100, 400, size=50)
    sizes = np.random.uniform(10, 30, size=100)
    clarity = np.random.uniform(0.3, 0.5, size=100)
    colors = np.random.uniform(0.3, 0.5, size=(100, 3))

    # Add smoke patches to the plot Using Von Neumann Neighborhood
    neighbors = [(x[1]-30, y[1]), (x[1]+40, y[1]), (x[1], y[1]-50), (x[1], y[1]+60)]
    for neighbor in neighbors:
        if 0 <= neighbor[0] <= 500 and 0 <= neighbor[1] <= 500:
            circle_neighbor = plt.Circle(neighbor, sizes[1], color=colors[1], alpha=clarity[0])
            bottom.add_patch(circle_neighbor)
```



I have used smoke effect to produce smoke bubble within my grid using Von Neumann Neighbourhood which is showing a diffusion of bubbles in the middle of the grid. The program has been tested and put in loop, which is running the code in a loop for 5 times. However, the number of loops would be increased in the final file to create a better simulation.

Props/Band

For stage:

```python
# lets start with making a stage

stage =  bottom.fill([0, 500, 500, 0], [0, 0, 80, 80], color="wheat")
```

For smoke machine object:

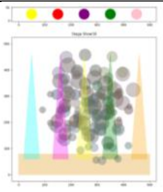
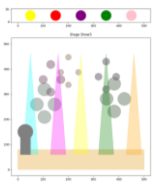
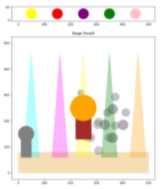
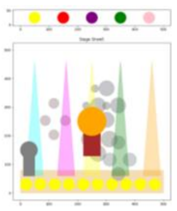
```python
#stage is working, now for props i think lets add an object on stage
#this is visualising smoke machine i guess
rectangle = plt.Rectangle((10, 60), 40, 100, color="gray", zorder=10)
bottom.add_patch(rectangle)

circle7 = plt.Circle((30, 150), 30, color="gray", zorder=10)
bottom.add_patch(circle7)
```

For Person in the middle:

```python
# add another object visualising a person in the middle
rectangle = plt.Rectangle((220, 130), 60, 100, color="brown", zorder=10)
bottom.add_patch(rectangle)
circle7 = plt.Circle((250, 250), 50, color="orange", zorder=10)
bottom.add_patch(circle7)
```

For Audience:

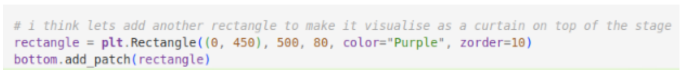
```python
    #lets add circle visualizing audience on the stage

for h in range(10):
    circle = plt.Circle((20 + h * 50, 30), 20, color="yellow", zorder=10)
    bottom.add_patch(circle)
```

I have used rectangle function to create a stage. I have used 2 objects on stage, one is the visualisation of a smoke machine on the side of the stage, other I have visualised as a person in the middle of the stage. I have created audience which is visualising as a crowd sitting and looking at the stage.

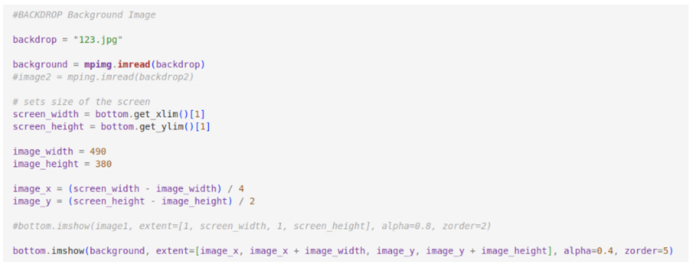
Backdrop

For Curtain on top:

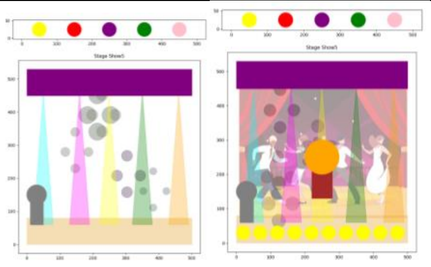


```python
# i think lets add another rectangle to make it visualise as a curtain on top of the stage
rectangle = plt.Rectangle((0, 450), 500, 80, color="Purple", zorder=10)
bottom.add_patch(rectangle)
```

For background Image:

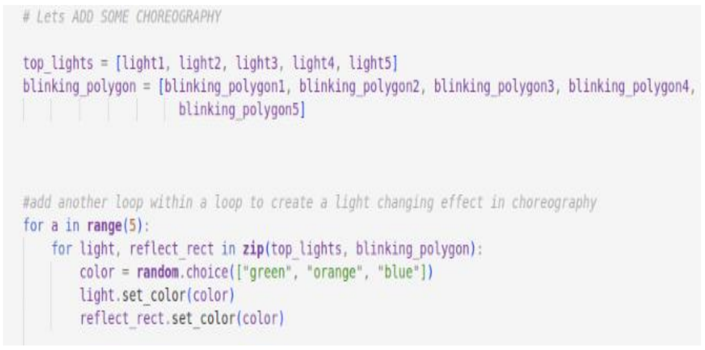


```python
#BACKDROP Background Image

backdrop = "123.jpg"

background = mpimg.imread(backdrop)
#image2 = mping.imread(backdrop2)

# sets size of the screen
screen_width = bottom.get_xlim()[1]
screen_height = bottom.get_ylim()[1]

image_width = 490
image_height = 380

image_x = (screen_width - image_width) / 4
image_y = (screen_height - image_height) / 2

#bottom.imshow(image1, extent=[1, screen_width, 1, screen_height], alpha=0.8, zorder=2)

bottom.imshow(background, extent=[image_x, image_x + image_width, image_y, image_y + image_height], alpha=0.4, zorder=5)
```

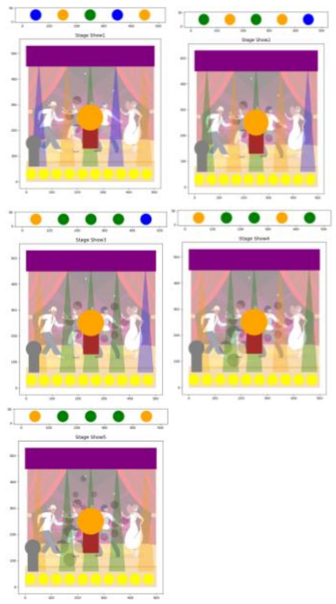


For the backdrop, I have used curtains and background image of people dancing. The code has been tested and the output has been shown.

Choreography



```python
# Lets ADD SOME CHOREOGRAPHY

top_lights = [light1, light2, light3, light4, light5]
blinking_polygon = [blinking_polygon1, blinking_polygon2, blinking_polygon3, blinking_polygon4,
                    blinking_polygon5]



#add another loop within a loop to create a light changing effect in choreography
for a in range(5):
    for light, reflect_rect in zip(top_lights, blinking_polygon):
        color = random.choice(["green", "orange", "blue"])
        light.set_color(color)
        reflect_rect.set_color(color)
```



For choreography, I have used a loop to create a simulation effect, representing change of lights in random order.

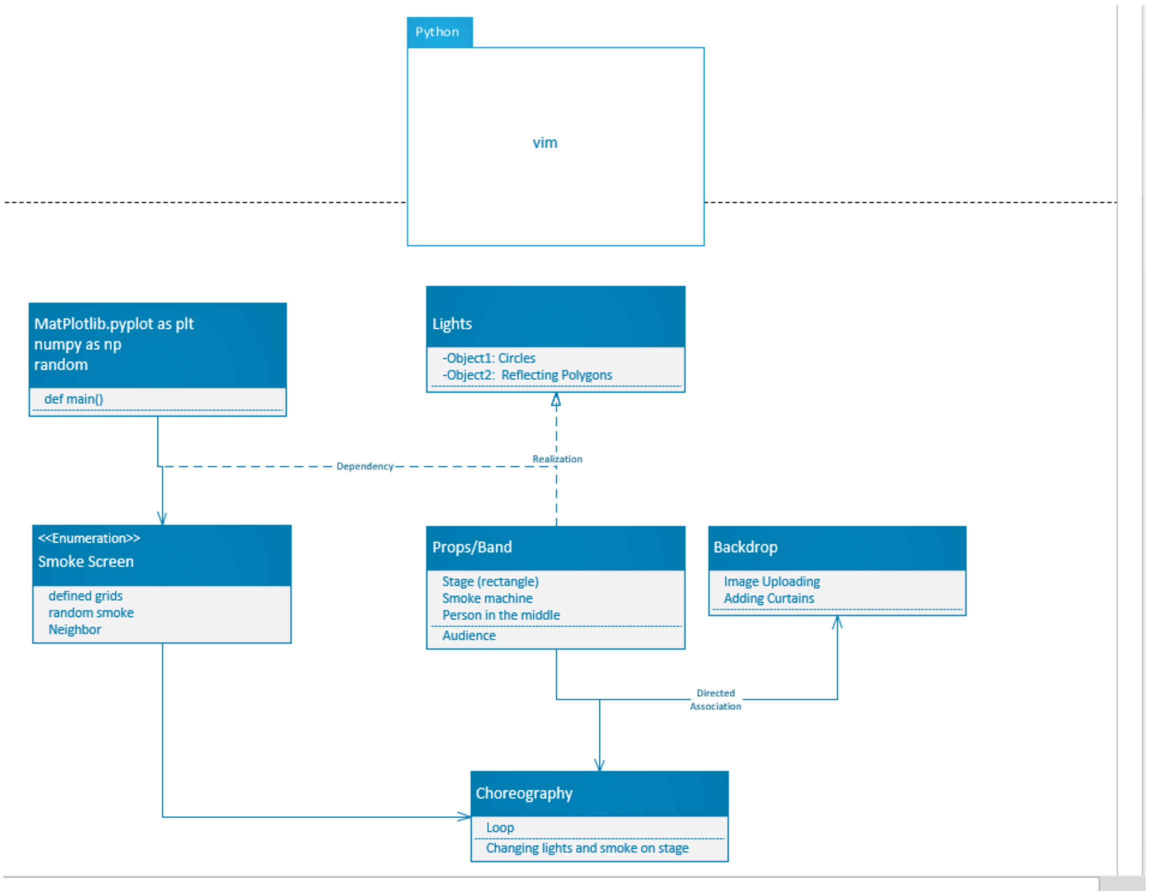
Discussion

In order for the simulations and scripts to work, “matplotlib.pyplot”, “numpy”, “random” and matplotlib.image needs to be imported into the python program. The main function defined in the script is the point where program begins or entry point of the script. A loop function “for a in range(20):” has been used for the program, which would be iterating my program twenty times with a pause of 0.5 seconds in between each iteration.

Program has been defined in such a way that with each iteration, there would be a change in the stage lights which are represented by circles object in the program. There corresponding light beams are created using the Polygon objects, which are represented in such a way that with each iteration they would change colour randomly, demonstrating the choreography at the back of the stage. In each iteration of the main loop, using the function “random.choice()” and specifying “green”, “I have also incorporated a smoke machine, which with change in each defined parameters, creates bubble objects, representing smoke on the stage.

With the change in each loop, the size of smoke particles keeps adding up. In each iteration of the for loop, another for loop is created to iterate a circle, created using the plt.Circle() function. The centre of the circle is defined by the coordinates (20 + i * 50, 30), where i is the current iteration index. This means that the circles will be horizontally spaced apart by 50 units, starting from the x-coordinate 20 and with a fixed y-coordinate of 30. The circle is added to the ax1 plot using the add_patch() method, which displays the circle as part of the audience visualization in the stage party. I have used a loop function to create simulation effect by changing its light in random order.

UML Class Diagram:



Showcase

Introduction:

The aim of this code is to showcase different variations I have applied into this code. I Have used different scenarios and have tried to implement it into the code. For the first part scene1.py I have made a simple code using matplotlib and plot function. I have used circles and polygons to represent lights and reflections. I have made stage using the rectangle function and put object in the middle and side of the stage as props. For second scenario I have used a loop function to change the lights with each iteration using set_colour function and for the last scenario, I have implemented a smoke machine into the code.

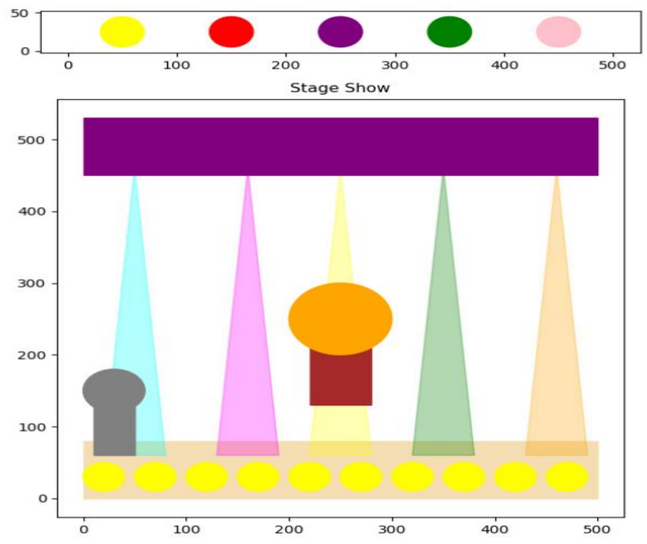
Scenario 1:

File name: scene1.py

Input file:

```
import matplotlib.pyplot as plt
import numpy as np


def main():


    fig, (top, bottom) = plt.subplots(2, 1, gridspec_kw={'height_ratios': [1, 10]}, figsize=(10, 10))
    #Lights
    # Adds light on top of the stage
    top.set_aspect("equal")
    top.fill([0, 500, 500, 0], [0, 0, 50, 50], color="white")
    light1 = plt.Circle((50, 25), 20, color="yellow")
    top.add_patch(light1)
    light2 = plt.Circle((150, 25), 20, color="red")
    top.add_patch(light2)
    light3 = plt.Circle((250, 25), 20, color="purple")
    top.add_patch(light3)
    light4 = plt.Circle((350, 25), 20, color="green")
    top.add_patch(light4)
    light5 = plt.Circle((450, 25), 20, color="pink")
    top.add_patch(light5)

    # Add corresponding reflection of stage lights
    blinking_polygon1 = plt.Polygon(np.array([[20, 60], [50, 460], [80, 60]]), color="cyan", alpha=0.3,
zorder=10)
    bottom.add_patch(blinking_polygon1)
    blinking_polygon2 = plt.Polygon(np.array([[130, 60], [160, 460], [190, 60]]), color="magenta",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon2)
    blinking_polygon3 = plt.Polygon(np.array([[220, 60], [250, 460], [280, 60]]), color="yellow",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon3)
    blinking_polygon4 = plt.Polygon(np.array([[320, 60], [350, 460], [380, 60]]), color="green", alpha=0.3,
zorder=10)
    bottom.add_patch(blinking_polygon4)
    blinking_polygon5 = plt.Polygon(np.array([[430, 60], [460, 460], [490, 60]]), color="orange",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon5)

    bottom.set_aspect("equal")
    bottom.fill([0, 500, 500, 0], [0, 0, 500, 500], color="white")

    #PROPS
    # starting code to make Props and bands, I think object on the middle would act as person standing. I
will make a stage too, and add some audience objects,,letss see
    # lets start with making a stage
    stage =  bottom.fill([0, 500, 500, 0], [0, 0, 80, 80], color="wheat")

    #stage is working, now for props i think lets add an object on stage
    #this is visualising smoke machine i guess
    rectangle = plt.Rectangle((10, 60), 40, 100, color="gray", zorder=10)
    bottom.add_patch(rectangle)

    circle1 = plt.Circle((30, 150), 30, color="gray", zorder=10)
    bottom.add_patch(circle1)


    # add another object visualising a person in the middle
    rectangle = plt.Rectangle((220, 130), 60, 100, color="brown", zorder=10)
    bottom.add_patch(rectangle)
    circle2 = plt.Circle((250, 250), 50, color="orange", zorder=10)
    bottom.add_patch(circle2)

     # i think lets add another rectangle to make it visualise as a curtain on top of the stage
    rectangle = plt.Rectangle((0, 450), 500, 80, color="Purple", zorder=10)
    bottom.add_patch(rectangle)

    #lets add circle visualizing audience on the stage

    for h in range(10):
        circle = plt.Circle((20 + h * 50, 30), 20, color="yellow", zorder=10)
        bottom.add_patch(circle)



    plt.title("Stage Show", fontsize="11")
    plt.savefig("Stage Show"+ '.png')


    plt.show()


if __name__ == "__main__":
    main()
```

Output



Discussion:

For the first scenario, I have created a stage view using matplotlib to create a plot and add various shapes to the plot which includes the lights on top of the stage, represented by the circles of different colours and has been added to the subplot. I have used a rectangle function to plot a stage. Similar function has been used to plot a curtain on top on the second plot. I have used the second plot to represent the reflection of light through there corresponding lights above them. Furthermore, I have used circle as an object and placed it on stage area to represent the audience. I have added props, like smoke machine which is shown on the left corner of output visualised by grey rectangle and circle. Additionally, I have another object placed in the middle represented by a rectangle and circle object visualising the person standing on the stage.

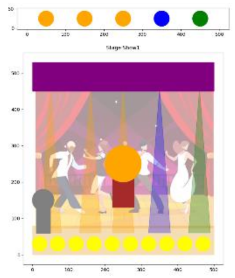
Scenario 2

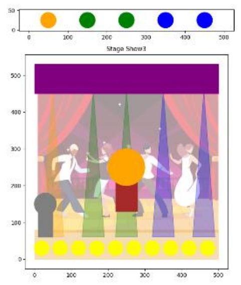
File Name: scene2.py

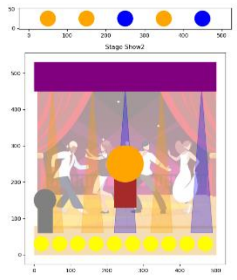
Input

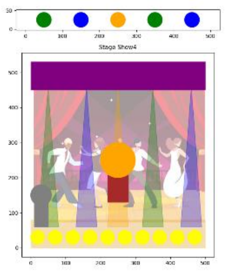
```
import matplotlib.pyplot as plt
import numpy as np
import random
import matplotlib.image as mpimg

def main():

    fig, (top, bottom) = plt.subplots(2, 1, gridspec_kw={'height_ratios': [1, 10]}, figsize=(10, 10))
    #Lights
    # Adds light on top of the stage
    top.set_aspect("equal")
    top.fill([0, 500, 500, 0], [0, 0, 50, 50], color="white")
    light1 = plt.Circle((50, 25), 20, color="yellow")
    top.add_patch(light1)
    light2 = plt.Circle((150, 25), 20, color="red")
    top.add_patch(light2)
    light3 = plt.Circle((250, 25), 20, color="purple")
    top.add_patch(light3)
    light4 = plt.Circle((350, 25), 20, color="green")
    top.add_patch(light4)
    light5 = plt.Circle((450, 25), 20, color="pink")
    top.add_patch(light5)

    # Add corresponding reflection of stage lights
    blinking_polygon1 = plt.Polygon(np.array([[20, 60], [50, 460], [80, 60]]), color="cyan", alpha=0.3,
zorder=10)
    bottom.add_patch(blinking_polygon1)
    blinking_polygon2 = plt.Polygon(np.array([[130, 60], [160, 460], [190, 60]]), color="magenta",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon2)
    blinking_polygon3 = plt.Polygon(np.array([[220, 60], [250, 460], [280, 60]]), color="yellow",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon3)
    blinking_polygon4 = plt.Polygon(np.array([[320, 60], [350, 460], [380, 60]]), color="green", alpha=0.3,
zorder=10)
    bottom.add_patch(blinking_polygon4)
    blinking_polygon5 = plt.Polygon(np.array([[430, 60], [460, 460], [490, 60]]), color="orange",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon5)

    bottom.set_aspect("equal")
    bottom.fill([0, 500, 500, 0], [0, 0, 500, 500], color="white")

    #BACKDROP Background Image

    backdrop = "123.jpg"

    background = mpimg.imread(backdrop)
    #image2 = mping.imread(backdrop2)

    # sets size of the screen
    screen_width = bottom.get_xlim()[1]
    screen_height = bottom.get_ylim()[1]

    image_width = 490
    image_height = 380
    image_x = (screen_width - image_width) / 4
    image_y = (screen_height - image_height) / 2

    #bottom.imshow(image1, extent=[1, screen_width, 1, screen_height], alpha=0.8, zorder=2)

    bottom.imshow(background, extent=[image_x, image_x + image_width, image_y, image_y + image_height],
alpha=0.4, zorder=5)


    #PROPS
    # starting code to make Props and bands, I think object on the middle would act as person standing. I
will make a stage too, and add some audience objects,,letss see
    # lets start with making a stage

    stage =  bottom.fill([0, 500, 500, 0], [0, 0, 80, 80], color="wheat")

    #stage is working, now for props i think lets add an object on stage
    #this is visualising smoke machine i guess
    rectangle = plt.Rectangle((10, 60), 40, 100, color="gray", zorder=10)
    bottom.add_patch(rectangle)

    circle1 = plt.Circle((30, 150), 30, color="gray", zorder=10)
    bottom.add_patch(circle1)





    # i think lets add another rectangle to make it visualise as a curtain on top of the stage
    rectangle = plt.Rectangle((0, 450), 500, 80, color="Purple", zorder=10)
    bottom.add_patch(rectangle)

    # add another object visualising a person in the middle
    rectangle = plt.Rectangle((220, 130), 60, 100, color="brown", zorder=10)
    bottom.add_patch(rectangle)
    circle2 = plt.Circle((250, 250), 50, color="orange", zorder=10)
    bottom.add_patch(circle2)


        #lets add circle visualizing audience on the stage

    for h in range(10):
        circle = plt.Circle((20 + h * 50, 30), 20, color="yellow", zorder=10)
        bottom.add_patch(circle)

    # Lets ADD SOME CHOREOGRAPHY

    top_lights = [light1, light2, light3, light4, light5]
    blinking_polygon = [blinking_polygon1, blinking_polygon2, blinking_polygon3, blinking_polygon4,
                          blinking_polygon5]


    for a in range(5):
        for light, reflect_rect in zip(top_lights, blinking_polygon):
            color = random.choice(["green", "orange", "blue"])
            light.set_color(color)
            reflect_rect.set_color(color)

        plt.title("Stage Show" + str(a + 1), fontsize="11")
        plt.savefig("Stage Show" + str(a + 1)+ '.png')
        plt.pause(0.5)


    plt.show()


if __name__ == "__main__":
    main()
```

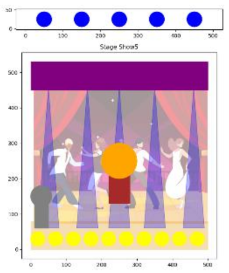
Output:











Discussion:

For this scenario, I have written the code to visually represent a stage show with lights, props, backdrop image and choreographed lights. I have used the loop function to add choreography to the stage lights. The code randomly changes the light colour and their corresponding reflection in loop creating a simulation effect. Each iteration of loop creates a different stage show with random stage light. Furthermore, I have added a picture as a backdrop from the internet showing a dance party going on the stage.

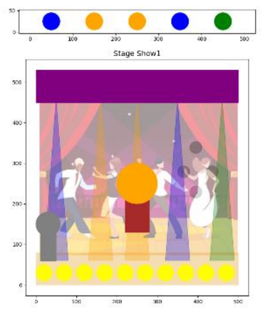
Scenario 3

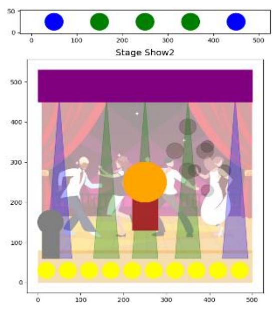
FileName: scene3.py

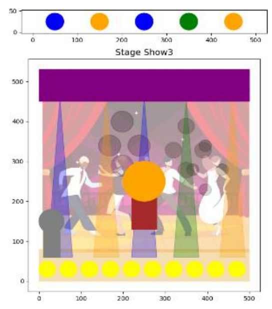
Input:

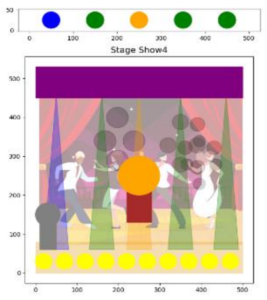
```
import matplotlib.pyplot as plt
import numpy as np
import random
import matplotlib.image as mpimg

def main():

    fig, (top, bottom) = plt.subplots(2, 1, gridspec_kw={'height_ratios': [1, 10]}, figsize=(10, 10))
    #Lights
    # Adds light on top of the stage
    top.set_aspect("equal")
    top.fill([0, 500, 500, 0], [0, 0, 50, 50], color="white")
    light1 = plt.Circle((50, 25), 20, color="yellow")
    top.add_patch(light1)
    light2 = plt.Circle((150, 25), 20, color="red")
    top.add_patch(light2)
    light3 = plt.Circle((250, 25), 20, color="purple")
    top.add_patch(light3)
    light4 = plt.Circle((350, 25), 20, color="green")
    top.add_patch(light4)
    light5 = plt.Circle((450, 25), 20, color="pink")
    top.add_patch(light5)
    # Add corresponding reflection of stage lights
    blinking_polygon1 = plt.Polygon(np.array([[20, 60], [50, 460], [80, 60]]), color="cyan", alpha=0.3,
zorder=10)
    bottom.add_patch(blinking_polygon1)
    blinking_polygon2 = plt.Polygon(np.array([[130, 60], [160, 460], [190, 60]]), color="magenta",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon2)
    blinking_polygon3 = plt.Polygon(np.array([[220, 60], [250, 460], [280, 60]]), color="yellow",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon3)
    blinking_polygon4 = plt.Polygon(np.array([[320, 60], [350, 460], [380, 60]]), color="green", alpha=0.3,
zorder=10)
    bottom.add_patch(blinking_polygon4)
    blinking_polygon5 = plt.Polygon(np.array([[430, 60], [460, 460], [490, 60]]), color="orange",
alpha=0.3, zorder=10)
    bottom.add_patch(blinking_polygon5)

    bottom.set_aspect("equal")
    bottom.fill([0, 500, 500, 0], [0, 0, 500, 500], color="white")

    #BACKDROP Background Image

    backdrop = "123.jpg"

    background = mpimg.imread(backdrop)
    #image2 = mping.imread(backdrop2)

    # sets size of the screen
    screen_width = bottom.get_xlim()[1]
    screen_height = bottom.get_ylim()[1]

    image_width = 490
    image_height = 380

    image_x = (screen_width - image_width) / 4
    image_y = (screen_height - image_height) / 2

    #bottom.imshow(image1, extent=[1, screen_width, 1, screen_height], alpha=0.8, zorder=2)

    bottom.imshow(background, extent=[image_x, image_x + image_width, image_y, image_y + image_height],
alpha=0.4, zorder=5)

    #Code to add smoke effect on the stage
    def smoke_effect():
        x = np.random.uniform(100, 400, size=50)
        y = np.random.uniform(100, 400, size=50)
        sizes = np.random.uniform(10, 30, size=100)
        clarity = np.random.uniform(0.3, 0.5, size=100)
        colors = np.random.uniform(0.3, 0.5, size=(100, 3))


        # Add smoke patches to the plot Using Von Neumann Neighborhood
        neighbors = [(x[1]-30, y[1]), (x[1]+40, y[1]), (x[1], y[1]-50), (x[1], y[1]+60)]
        for neighbor in neighbors:
            if 0 <= neighbor[0] <= 500 and 0 <= neighbor[1] <= 500:
                circle_neighbor = plt.Circle(neighbor, sizes[1], color=colors[1], alpha=clarity[0])
                bottom.add_patch(circle_neighbor)


#PROPS
   # starting code to make Props and bands, I think object on the middle would act as person standing. I
will make a stage too, and add some audience objects,,letss see
    # lets start with making a stage

    stage =  bottom.fill([0, 500, 500, 0], [0, 0, 80, 80], color="wheat")

    #stage is working, now for props i think lets add an object on stage
    #this is visualisation of smoke machine
    rectangle = plt.Rectangle((10, 60), 40, 100, color="gray", zorder=10)
    bottom.add_patch(rectangle)

    circle1 = plt.Circle((30, 150), 30, color="gray", zorder=10)
    bottom.add_patch(circle1)

    # i think lets add another rectangle to make it visualise as a curtain on top of the stage
    rectangle = plt.Rectangle((0, 450), 500, 80, color="Purple", zorder=10)
    bottom.add_patch(rectangle)

    # add another object visualising a person in the middle
    rectangle = plt.Rectangle((220, 130), 60, 100, color="brown", zorder=10)
    bottom.add_patch(rectangle)
    circle2 = plt.Circle((250, 250), 50, color="orange", zorder=10)
    bottom.add_patch(circle2)


        #lets add circle visualizing audience on the stage
    for h in range(10):
        circle = plt.Circle((20 + h * 50, 30), 20, color="yellow", zorder=10)
        bottom.add_patch(circle)


    # Lets ADD SOME CHOREOGRAPHY

    top_lights = [light1, light2, light3, light4, light5]
    blinking_polygon = [blinking_polygon1, blinking_polygon2, blinking_polygon3, blinking_polygon4,
                          blinking_polygon5]



    #add another loop within a loop to create a light changing effect in choreography
    for a in range(10):
        for light, reflect_rect in zip(top_lights, blinking_polygon):
            color = random.choice(["green", "orange", "blue"])
            light.set_color(color)
            reflect_rect.set_color(color)


        smoke_effect()
        plt.title("Stage Show" + str(a + 1), fontsize="14")
        plt.savefig("Stage Show" + str(a + 1)+ '.png')
        plt.pause(0.5)


    plt.show()


if __name__ == "__main__":
    main()
```

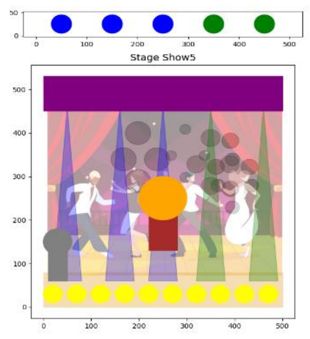
Output:

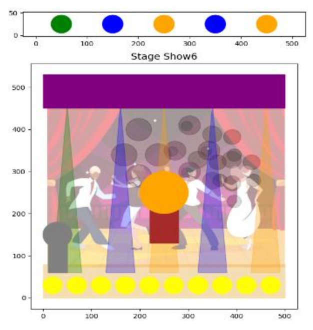


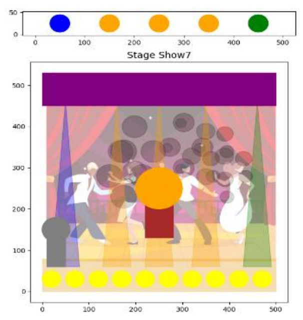


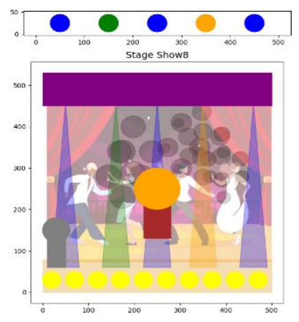


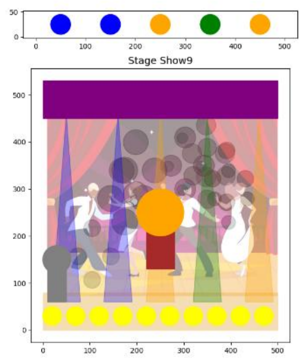


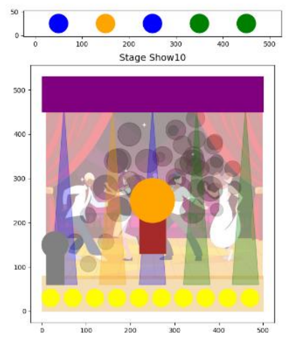












Discussion:

For this scenario, I have added smoke machine to the code. I have used smoke effect for the function. The Function randomly generates coordinates x and y with in the bounded range of 100 to 400 for every 50 smoke patches. These are the coordinates that are determining the smoke patches position on the stage. The code defines neighbours using Von Neumann Neighbourhood concept which help the spread of smoke in a more realistic manner. The smoke patches are used to appear as circles with random position and sizes.

Conclusion

The assignment was very complex as compared to the ideas discussed in the practicals, however, it helped me get into the code and understand the working dynamics of the python codes and different libraries. I had to read through different online resources, match it with the practicals, notes and slides provided and then try to implement it using my own imagination. Considering, I have no basic knowledge of software, computing and programming, I think I have done fairly well in this assignment. I have put my 110% effort and applied everything I have learnt and researched.

Future Work

During my research and study for this assignment, I realised the extensive usage of the python language and how it could be used to web development and simulations. I am very interested in learning MATLAB and pygame and enhance my learning. For this assignment, I would like to learn and apply a 3-D model for this simulation and learn some codes for the animations.

References

Image used:

https://www.vectorstock.com/royalty-free-vector/group-people-on-dance-floor-party-dancing-vector-29105447

Smoke Effect:

https://blender.stackexchange.com/questions/243294/quick-smoke-animation-by-python-invisible-working-by-gui