Exercise Series: Graphical Programming with Graphix.py

1. Hello Graphix

Objective: Create a window that displays a greeting message.

```python
from graphix import Window, Text, Point

def hello_graphix():
    win = Window("Hello Graphix", 400, 400)  # Create a window
    message = Text(Point(200, 200), "Hello, Graphix!")  # Create a text object
    message.draw(win)  # Draw the message
    win.get_mouse()  # Wait for a mouse click
    win.close()  # Close the window

>> hello_graphix()
```

Concepts: Window creation, text display, and user interaction.

Variations: ask for user input in console, ask for user input in text entry

2. Drawing a Point

Objective: Draw a single point on the window and move it.

```python
from graphix import Window, Point

def draw_point():
    win = Window("Draw a Point", 400, 400)
    p = Point(200, 200)  # Create a point
    p.draw(win)          # Draw the point
    win.get_mouse()
    p.move(50, -50)     # Move the point
    win.get_mouse()
    win.close()

>> draw_point()
```

Concepts: Point creation, drawing, and moving points.

Variations: use keyPress to move point up, down, left or right by 10px, use a loop to move the point and change direction with keyPress

---

3. Drawing a Line

Objective: Draw a line between two points.

```
from graphix import Window, Line, Point

def draw_line():
    win = Window("Draw a Line", 400, 400)
    line = Line(Point(50, 50), Point(350, 350))  # Line from (50, 50) to (350, 350)
    line.draw(win)                                # Draw the line on the window
    win.get_mouse()                              # Wait for user to click
    win.close()                                  # Close the window

>> draw_line()
```

Concepts: Line creation and drawing.

Variations: create patterns with lines inside a loop which increments points - to create interesting effects, use move() function on line

---

4. Creating a Circle

Objective: Ask the user for the radius of a circle and draw it.

```
from graphix import Window, Circle, Point

def draw_circle():
    win = Window("Circle Drawer", 400, 400)
    radius = int(input("Enter the radius of the circle: "))  # User input for radius
    circle = Circle(Point(200, 200), radius)  # Create a circle centered in the window
    circle.draw(win)  # Draw the circle
    win.get_mouse()  # Wait for user to click
    win.close()  # Close the window

draw_circle()
```

Concepts: User input, circle creation, and drawing.

Variations: See previous variations

---

5. Drawing a Centered Rectangle

Objective: Draw a rectangle with user-defined dimensions.

```
from graphix import Window, Rectangle, Point

def draw_rectangle():
    win = Window("Centered Rectangle", 400, 400)
    width = int(input("Enter the width of the rectangle: "))
    height = int(input("Enter the height of the rectangle: "))

    # Center the rectangle in the window
    x_left = (400 - width) // 2
    y_top = (400 - height) // 2

    rect = Rectangle(Point(x_left, y_top), Point(x_left + width, y_top +
height))
    rect.draw(win)  # Draw the rectangle
    win.get_mouse()  # Wait for user to click
    win.close()  # Close the window

draw_rectangle()
```

Concepts: User input, rectangle creation, centering shapes.

Variations: loops, autochange points so create shornking but centred rectangles, use different colours for each rectangle to create dynamic pattern.

---

6. Colorful Spirals

Objective: Draw a series of circles in a spiral pattern.

```
from graphix import Window, Circle, Point

def draw_spiral():
    win = Window("Colorful Spirals", 400, 400)
    for i in range(10):
        circle = Circle(Point(200 + (i * 10), 200), 10 + (i * 5))  # Create
circles with increasing size
        if i % 2 == 0:
            circle.fill_colour = "blue"
        else:
            circle.fill_colour = "green"
        circle.draw(win)  # Draw the circle
    win.get_mouse()  # Wait for user to click
    win.close()  # Close the window

>> draw_spiral()
```

Concepts: Looping, dynamic circle creation, and color filling.

---

7. Custom Shapes with Points

Objective: Create a custom shape based on user-defined points.

from graphix import Window, Polygon, Point

```
def draw_custom_shape():
    win = Window("Custom Shape", 400, 400)
    points = []
    for i in range(5):  # Allow user to define 5 points
        x = int(input(f"Enter x-coordinate for point {i + 1}: "))
        y = int(input(f"Enter y-coordinate for point {i + 1}: "))
        points.append(Point(x, y))  # Store the points in a list

    custom_shape = Polygon(points)  # Create a polygon using the points
    custom_shape.fill_colour = "orange"  # Set fill color
    custom_shape.draw(win)  # Draw the custom shape
    win.get_mouse()  # Wait for user to click
    win.close()  # Close the window
```

```
>> draw_custom_shape()
```

Concepts: User-defined shapes, Polygon creation.

Variations: using mathematics create a tiling of the custom_shape

---

8. Interactive Circle Drawing

Objective: Draw a circle at the location of each mouse click.

```
from graphix import Window, Circle

def interactive_circle_drawing():
    win = Window("Draw Circles on Click", 400, 400)
    while True:
        click = win.get_mouse()  # Wait for mouse click
        circle = Circle(click, 20)  # Create a circle at the click position
        circle.fill_colour = "purple"   # Set fill color to purple
        circle.draw(win)            # Draw the circle

>> interactive_circle_drawing()
```

Concepts: Dynamic shape creation based on user input.

Variations: once drawn the circle should move off in 1px steps and bounce off screen

---

9. Text and User Input

Objective: Use an Entry box to get user input and display it as text.

```
from graphix import Window, Entry, Text, Point

def text_input_interaction():
    win = Window("User Input", 400, 200)
    input_box = Entry(Point(200, 100), 20)  # Create an entry box for user
input
    input_box.draw(win)                      # Draw the entry box
    message = Text(Point(200, 50), "Enter your name:")  # Instruction message
    message.draw(win)                                   # Draw the instruction text

    win.get_mouse()                                     # Wait for user to click
    user_input = input_box.text                         # Get the user's input
    message.text = "Hello, " + user_input               # Update the text with the user's
name
    win.get_mouse()                                     # Wait for another click
    win.close()                                         # Close the window

>> text_input_interaction()
```

Concepts: Entry fields for input, handling text dynamically.

---

10. Plotting Clicks

Objective: Capture mouse clicks and display the coordinates on the window.

```python
from graphix import Window, Text, Point

def plot_clicks():
    win = Window("Plot Clicks", 400, 400)
    for _ in range(10):  # Allow user to click 10 times
        p = win.get_mouse()  # Wait for mouse click
        location = Text(Point(p.x, p.y), f"({p.x}, {p.y})")  # Create text
with coordinates
        location.draw(win)  # Draw the coordinates
    win.get_mouse()  # Wait for user to click
    win.close()  # Close the window

plot_clicks()
```

Concepts: Capturing mouse clicks, displaying coordinates.

These exercises introduce new concepts and encourage creativity while still using the graphix.py library's features. They progress from simple tasks to more complex interactions, providing a solid foundation for students to build their graphical programming skills