```
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import tkinter as tk
from tkinter import ttk, colorchooser, messagebox, simpledialog

class DrawingApp:
    def __init__(self):
        pygame.init()
        self.screen_width = 800
        self.screen_height = 600
        pygame.display.set_mode((self.screen_width, self.screen_height), DOUBLEBUF | OPENGL)

        self.rendering_mode = GL_FILL
        gluPerspective(45, (self.screen_width / self.screen_height), 0.1, 50.0)
        glTranslatef(0.0, 0.0, -5)
        self.quad = gluNewQuadric()
        gluQuadricTexture(self.quad, GL_TRUE)
        self.load_texture()



        glViewport(0, 0, self.screen_width, self.screen_height)
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        glOrtho(0, self.screen_width, self.screen_height, 0, -1, 1)
        glMatrixMode(GL_MODELVIEW)

        self.background_color = (0.2, 0.2, 0.2, 1.0)  # Initial background color
        self.object_color = (0.8, 0.8, 0.8, 1.0)  # Initial object color
        self.line_width = 1.0  # Initial line width
        self.line_style = GL_LINES  # Initial line style
        self.line_type = 'Solid'
        self.num_style = 0
        self.scale_factor = 1.0  # Initial scale factor
        self.rendering_mode = GL_FILL
        self.gl_style_options = [GL_LINE, GL_LINE_STRIP, GL_LINE_LOOP, GL_POINTS, GL_LINES, GL_TRIANGLES,
GL_TRIANGLE_FAN]

        self.xy = [(200, 200), (300, 300)]

        self.drawing_function = self.draw_line  # Initial drawing function

        self.set_background_color(self.background_color)  # Set initial background color

        #self.diskRender = opengl_part.DiskRenderer()


    def choose_background_color(self):
        color = colorchooser.askcolor(title="Choose background color")
        background_color = [0.0, 0.0, 0.0, 1.0]
        for i in range(3):
            background_color[i] = color[0][i] / 255
        return tuple(background_color)

    def choose_line_width(self):
        root = tk.Tk()
        root.withdraw()

        number = simpledialog.askfloat("Enter Float Number", "Please enter a float number:")
```

```
    return float(number)

def choose_line_style(self):
    root = tk.Tk()
    root.title("Select Line Style")

    line_styles = ['Dashed', 'Dotted']

    # Create a combobox with the available line styles
    line_style_combo = ttk.Combobox(root, values=line_styles)
    line_style_combo.pack()

    def on_select():
        line_style = line_style_combo.get()
        print("Selected Line Style:", line_style)
        self.num_style = line_style
        self.line_type = line_style
        root.destroy()

    # Create a button to confirm the selection
    select_button = ttk.Button(root, text="Select", command=on_select)
    select_button.pack()

    root.mainloop()
def choose_fill_color(self):
    color = colorchooser.askcolor(title="Choose background color")
    fill_color = [0.0, 0.0, 0.0, 1.0]
    for i in range(3):
        fill_color[i] = color[0][i] / 255
    return tuple(fill_color)

def ask_coordinates(self, num_coordinates):
    root = tk.Tk()
    root.withdraw()

    coordinates = []

    for i in range(num_coordinates):
        x = simpledialog.askinteger("X Coordinate", f"Please enter the X coordinate for point {i + 1}:")
        y = simpledialog.askinteger("Y Coordinate", f"Please enter the Y coordinate for point {i + 1}:")
        coordinates.append((x, y))

    return coordinates

def draw_line(self):
    glColor4fv(self.object_color)
    glLineWidth(self.line_width)

    if self.line_type == "Dashed Line":
        glLineStipple(1, 0x0F0F)
        glEnable(GL_LINE_STIPPLE)
    elif self.line_type == "Dotted Line":
        glLineStipple(1, 0xAAAA)
        glEnable(GL_LINE_STIPPLE)
    else:
        glDisable(GL_LINE_STIPPLE)
    glBegin(self.line_style)

    x0 = self.xy[0][0]
    x1 = self.xy[1][0]
    y0 = self.xy[0][1]
```

```
    y1 = self.xy[1][1]
    glVertex2f(x0 * self.scale_factor, y0 * self.scale_factor)  # Starting point of the line
    glVertex2f(x1 * self.scale_factor, y1 * self.scale_factor)  # Ending point of the line
    glEnd()

def draw_rectangle(self):
    glColor4fv(self.object_color)
    glLineWidth(self.line_width)
    glBegin(GL_QUADS)

    x0 = self.xy[0][0]
    x1 = self.xy[1][0]
    y0 = self.xy[0][1]
    y1 = self.xy[1][1]
    glVertex2f(x0 * self.scale_factor, y0 * self.scale_factor)  # Top-left vertex
    glVertex2f(x1 * self.scale_factor, y0 * self.scale_factor)  # Top-right vertex
    glVertex2f(x1 * self.scale_factor, y1 * self.scale_factor)  # Bottom-right vertex
    glVertex2f(x0 * self.scale_factor, y1 * self.scale_factor)  # Bottom-left vertex
    glEnd()

def draw_polygon(self):
    glColor4fv(self.object_color)
    glLineWidth(self.line_width)
    glBegin(GL_POLYGON)

    for i in range(len(self.xy)):
        x = self.xy[i][0]
        y = self.xy[i][1]
        glVertex2f(x * self.scale_factor, y * self.scale_factor)

    glEnd()

def draw_triangle(self):
    glColor4fv(self.object_color)
    glLineWidth(self.line_width)
    glBegin(GL_TRIANGLES)
    x0 = self.xy[0][0]
    x1 = self.xy[1][0]
    x2 = self.xy[2][0]
    y0 = self.xy[0][1]
    y1 = self.xy[1][1]
    y2 = self.xy[2][1]

    glVertex2f(x0 * self.scale_factor, y0 * self.scale_factor)  # Vertex 1
    glVertex2f(x1 * self.scale_factor, y1 * self.scale_factor)  # Vertex 2
    glVertex2f(x2 * self.scale_factor, y2 * self.scale_factor)  # Vertex 3
    glEnd()

def set_background_color(self, color):
    self.background_color = color

def set_object_color(self, color):
    self.object_color = color

def set_line_width(self, width):
    self.line_width = width

def set_line_style(self, style):
    self.line_style = style

def set_scale_factor(self, factor):
    self.scale_factor = factor
```

```
    def ask_integer(self):
        root = tk.Tk()
        root.withdraw()

        integer = simpledialog.askinteger("Enter an Integer", "Please enter an integer:")

        return integer

    def handle_keypress(self, event):
        if event.key == pygame.K_p:
            self.rendering_mode = GL_POINT  # Switch to point rendering mode
        elif event.key == pygame.K_w:
            self.rendering_mode = GL_LINE  # Switch to wireframe rendering mode
        elif event.key == pygame.K_s:
            self.rendering_mode = GL_FILL  # Switch to solid rendering mode
        glPolygonMode(GL_FRONT_AND_BACK, self.rendering_mode)  # Update the rendering mode

    def load_texture(self):
        textureSurface = pygame.image.load('texture.jpg')
        textureData = pygame.image.tostring(textureSurface, "RGBA", 1)
        width = textureSurface.get_width()
        height = textureSurface.get_height()

        glEnable(GL_TEXTURE_2D)
        texid = glGenTextures(1)

        glBindTexture(GL_TEXTURE_2D, texid)
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA, GL_UNSIGNED_BYTE,
textureData)

        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL)

    def draw_disk(self):
        glRotatef(1, 3, 1, 1)

        slices = 50
        inner_radius = 0.5
        outer_radius = 1.0
        gluDisk(self.quad, inner_radius, outer_radius, slices, 1)

    def see_disk(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            elif event.type == pygame.KEYDOWN:
                self.handle_keypress(event)

        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
        glRotatef(1, 3, 1, 1)
        self.drawing_function = self.draw_disk

    def run(self):
        running = True
        while running:
            for event in pygame.event.get():
                if event.type == QUIT:
```

```
                running = False
            elif event.type == KEYDOWN:
                if event.key == K_1:
                    self.xy = self.ask_coordinates(2)
                    self.drawing_function = self.draw_line
                elif event.key == K_2:
                    self.xy = self.ask_coordinates(2)
                    self.drawing_function = self.draw_rectangle
                elif event.key == K_3:
                    self.xy = self.ask_coordinates(3)
                    self.drawing_function = self.draw_triangle
                elif event.key == K_4:
                    num_of_coord = self.ask_integer()
                    self.xy = self.ask_coordinates(num_of_coord)
                    self.drawing_function = self.draw_polygon
                elif event.key == K_b:
                    self.set_background_color(self.choose_background_color())  # Set background color to white
                elif event.key == K_f:
                    self.set_object_color(self.choose_fill_color())  # Set object color to green
                elif event.key == K_w:
                    self.set_line_width(self.choose_line_width())  # Set line width to 1.0g
                elif event.key == K_r:
                    self.drawing_function = self.draw_disk
                elif event.key == K_s:
                    self.choose_line_style()
                    #self.set_line_style(self.line_style)  #  TODO Set line style to GL_LINES
                elif event.key == K_EQUALS and pygame.key.get_mods() & pygame.KMOD_SHIFT:
                    self.set_scale_factor(self.scale_factor * 1.1)  # Increase scale factor by 10%
                elif event.key == K_MINUS:
                    self.set_scale_factor(self.scale_factor * 0.9)  # Decrease scale factor by 10%

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glLoadIdentity()

        glClearColor(*self.background_color)

        self.drawing_function()

        pygame.display.flip()
        pygame.time.wait(10)

    pygame.quit()

app = DrawingApp()
app.run()
```