CMPE 496 - Project 1

This program is basically a simple paint program which allows you to create, move and delete rectangles, squares, circles and lines.

This program developed on python by using the "tkinter" package and considering object oriented programming principles.

Main Functions

1- Create Object

You can create some object by choosing the object type from the menu and using the left click of the mouse. Once you release the left click, the program will create an object from first clicked point to released point in the shape of selection.

2 - Move Object

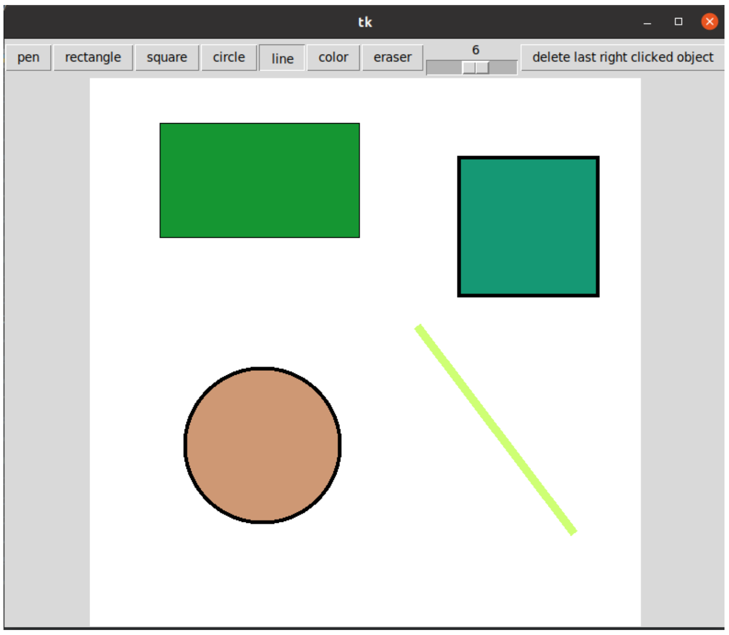
You can use the mouse right click to choose an object and move it. Once you release the right click, it moves the object to the released point.

Moreover, you can move objects by using arrows right after creation of the object.

3 - Delete Object

You can use the mouse right click to choose an object. Once you choose and object you can press the menu button "delete last right clicked object" and the program deletes the object.

Moreover, you can delete objects by using "delete" on the keyboard right after creation of the object.



Code of The Programm

```
from tkinter import *
from tkinter.colorchooser import askcolor


class Paint(object):
   DEFAULT_PEN_SIZE = 5.0
   DEFAULT_COLOR = 'black'

   def __init__(self):

       self.root = Tk()
       self.moving_object = None
       self.old_x = None
       self.old_y = None
       self.x = None
       self.y = None
       self.last_created_object = None

       self.pen_button = Button(self.root, text='pen', command=self.use_pen)
       self.pen_button.grid(row=0, column=0)

       self.brush_button = Button(self.root, text='rectangle',
command=self.use_brush)
       self.brush_button.grid(row=0, column=1)

       self.square_button = Button(self.root, text='square',
command=self.use_square)
       self.square_button.grid(row=0, column=2)

       self.circle_button = Button(self.root, text='circle',
command=self.use_circle)
       self.circle_button.grid(row=0, column=3)

       self.line_button = Button(self.root, text='line',
command=self.use_line)
       self.line_button.grid(row=0, column=4)

       self.color_button = Button(self.root, text='color',
command=self.choose_color)
       self.color_button.grid(row=0, column=5)

       self.eraser_button = Button(self.root, text='eraser',
command=self.use_eraser)
       self.eraser_button.grid(row=0, column=6)

       self.choose_size_button = Scale(self.root, from_=1, to=10,
orient=HORIZONTAL)
       self.choose_size_button.grid(row=0, column=7)
```

```python
        self.delete_button = Button(self.root, text='delete last right clicked object', command=self.delete_object)
        self.delete_button.grid(row=0, column=8)

        self.c = Canvas(self.root, bg='white', width=600, height=600)
        self.c.grid(row=1, columnspan=9)

        self.line_width = self.choose_size_button.get()
        self.color = self.DEFAULT_COLOR
        self.eraser_on = False
        self.active_button = self.pen_button

        self.setup()
        self.root.mainloop()

    def setup(self):
        self.c.bind('<B1-Motion>', self.paint)
        self.c.bind('<ButtonPress-1>', self.button_press)
        self.c.bind('<ButtonRelease-1>', self.button_release)
        self.c.bind('<ButtonPress-3>', self.button_press_right)
        self.c.bind('<ButtonRelease-3>', self.button_release_right)
        self.root.bind("<Left>", self.left)
        self.root.bind("<Right>", self.right)
        self.root.bind("<Up>", self.up)
        self.root.bind("<Down>", self.down)
        self.root.bind("<Delete>", self.remove_object)

    def remove_object(self, event):
        print(event.keysym)
        self.c.delete(self.last_created_object)

    # for motion in negative x direction
    def left(self, event):
        print(event.keysym)
        self.c.move(self.last_created_object, -5, 0)

    # for motion in positive x direction
    def right(self, event):
        print(event.keysym)
        self.c.move(self.last_created_object, 5, 0)

    # for motion in positive y direction
    def up(self, event):
        print(event.keysym)
        self.c.move(self.last_created_object, 0, -5)

    # for motion in negative y direction
    def down(self, event):
        print(event.keysym)
        self.c.move(self.last_created_object, 0, 5)

    def use_pen(self):
        self.activate_button(self.pen_button)
    def use_brush(self):
        self.activate_button(self.brush_button)

    def use_square(self):
        self.activate_button(self.square_button)

    def use_circle(self):
        self.activate_button(self.circle_button)

    def use_line(self):
        self.activate_button(self.line_button)

    def choose_color(self):
        self.eraser_on = False
        self.color = askcolor(color=self.color)[1]

    def use_eraser(self):
        self.activate_button(self.eraser_button, eraser_mode=True)

    def activate_button(self, some_button, eraser_mode=False):
        self.active_button.config(relief=RAISED)
        some_button.config(relief=SUNKEN)
        self.active_button = some_button
        self.eraser_on = eraser_mode

    def paint(self, event):
        self.c.delete("temp_rectangle")
        self.c.delete("temp_square")
        self.c.delete("temp_circle")
        self.c.delete("temp_line")
        self.line_width = self.choose_size_button.get()
        paint_color = 'white' if self.eraser_on else self.color
        if self.old_x and self.old_y and (self.active_button ==
self.pen_button or self.eraser_on):
            self.c.create_line(self.old_x, self.old_y, event.x, event.y,
                               width=self.line_width, fill=paint_color,
                               capstyle=ROUND, smooth=TRUE, splinesteps=36)

        if self.x and self.y and self.active_button == self.brush_button:
            self.c.create_rectangle(self.x, self.y, event.x, event.y,
tag="temp_rectangle",
                                   width=self.line_width)

        elif self.x and self.y and self.active_button == self.square_button:
            diff_x = self.x - event.x
            diff_y = self.y - event.y
            if abs(diff_x) > abs(diff_y):
                edge_length = abs(diff_x)
            else:
                edge_length = abs(diff_y)
            if diff_y > 0:
                if diff_x > 0:
                    coord_x = self.x - edge_length
                    coord_y = self.y - edge_length
                else:
                    coord_x = self.x + edge_length
                    coord_y = self.y - edge_length
            else:
                if diff_x > 0:
                    coord_x = self.x - edge_length
                    coord_y = self.y + edge_length
                else:
                    coord_x = self.x + edge_length
                    coord_y = self.y + edge_length

            self.c.create_rectangle(self.x, self.y, coord_x, coord_y,
                                    tag="temp_square",
                                    width=self.line_width)
        elif self.x and self.y and self.active_button == self.circle_button:
            diff_x = self.x - event.x
            diff_y = self.y - event.y
            if abs(diff_x) > abs(diff_y):
                edge_length = abs(diff_x)
            else:
                edge_length = abs(diff_y)
            if diff_y > 0:
                if diff_x > 0:
                    coord_x = self.x - edge_length
                    coord_y = self.y - edge_length
                else:
                    coord_x = self.x + edge_length
                    coord_y = self.y - edge_length
            else:
                if diff_x > 0:
                    coord_x = self.x - edge_length
                    coord_y = self.y + edge_length
                else:
                    coord_x = self.x + edge_length
                    coord_y = self.y + edge_length
            self.c.create_oval(self.x, self.y, coord_x, coord_y,
tag="temp_circle",
                               width=self.line_width)

        elif self.x and self.y and self.active_button == self.line_button:
            self.c.create_line(self.x, self.y, event.x, event.y,
tag="temp_line",
                               width=self.line_width)

        self.old_x = event.x
        self.old_y = event.y

    def button_press(self, event):
        self.x, self.y = event.x, event.y

    def button_release(self, event):
        if self.x and self.y:
            self.c.delete("temp_rectangle")
            self.c.delete("temp_square")
            self.c.delete("temp_circle")
            self.c.delete("temp_line")
            paint_color = 'white' if self.eraser_on else self.color
            self.line_width = self.choose_size_button.get()
            if self.active_button == self.brush_button:
                self.last_created_object = self.c.create_rectangle(self.x,
self.y, event.x, event.y,

width=self.line_width, fill=paint_color)
            elif self.active_button == self.square_button:
                diff_x = self.x - event.x
                diff_y = self.y - event.y
                if abs(diff_x) > abs(diff_y):
                    edge_length = abs(diff_x)
                else:
                    edge_length = abs(diff_y)
                if diff_y > 0:
                    if diff_x > 0:
                        coord_x = self.x - edge_length
                        coord_y = self.y - edge_length
                    else:
                        coord_x = self.x + edge_length
                        coord_y = self.y - edge_length
                else:
                    if diff_x > 0:
                        coord_x = self.x - edge_length
                        coord_y = self.y + edge_length
                    else:
                        coord_x = self.x + edge_length
                        coord_y = self.y + edge_length
                self.last_created_object = self.c.create_rectangle(self.x,
self.y,
                                                                   coord_x,
coord_y,

width=self.line_width, fill=paint_color)
            elif self.active_button == self.circle_button:
                diff_x = self.x - event.x
                diff_y = self.y - event.y
                if abs(diff_x) > abs(diff_y):
                    edge_length = abs(diff_x)
                else:
                    edge_length = abs(diff_y)
                if diff_y > 0:
                    if diff_x > 0:
                        coord_x = self.x - edge_length
                        coord_y = self.y - edge_length
                    else:
                        coord_x = self.x + edge_length
                        coord_y = self.y - edge_length
                else:
                    if diff_x > 0:
                        coord_x = self.x - edge_length
                        coord_y = self.y + edge_length
                    else:
                        coord_x = self.x + edge_length
                        coord_y = self.y + edge_length
                self.last_created_object = self.c.create_oval(self.x, self.y,
coord_x, coord_y,

width=self.line_width, fill=paint_color)

            elif self.active_button == self.line_button:
                self.last_created_object = self.c.create_line(self.x, self.y,
event.x, event.y,

width=self.line_width, fill=paint_color)

        self.old_x, self.old_y = None, None

    def move_object(self, event):
        self.c.moveto(self.moving_object, event.x, event.y)

    def delete_object(self):
        self.c.delete(self.moving_object)

    def button_press_right(self, event):
        self.moving_object = self.c.find_withtag("current")
        self.c.tag_bind(self.moving_object, '<B3-Motion>', self.move_object)

    def button_release_right(self, event):
        self.c.tag_unbind(self.moving_object, '<B3-Motion>')


if __name__ == '__main__':
    Paint()
```