Pyint

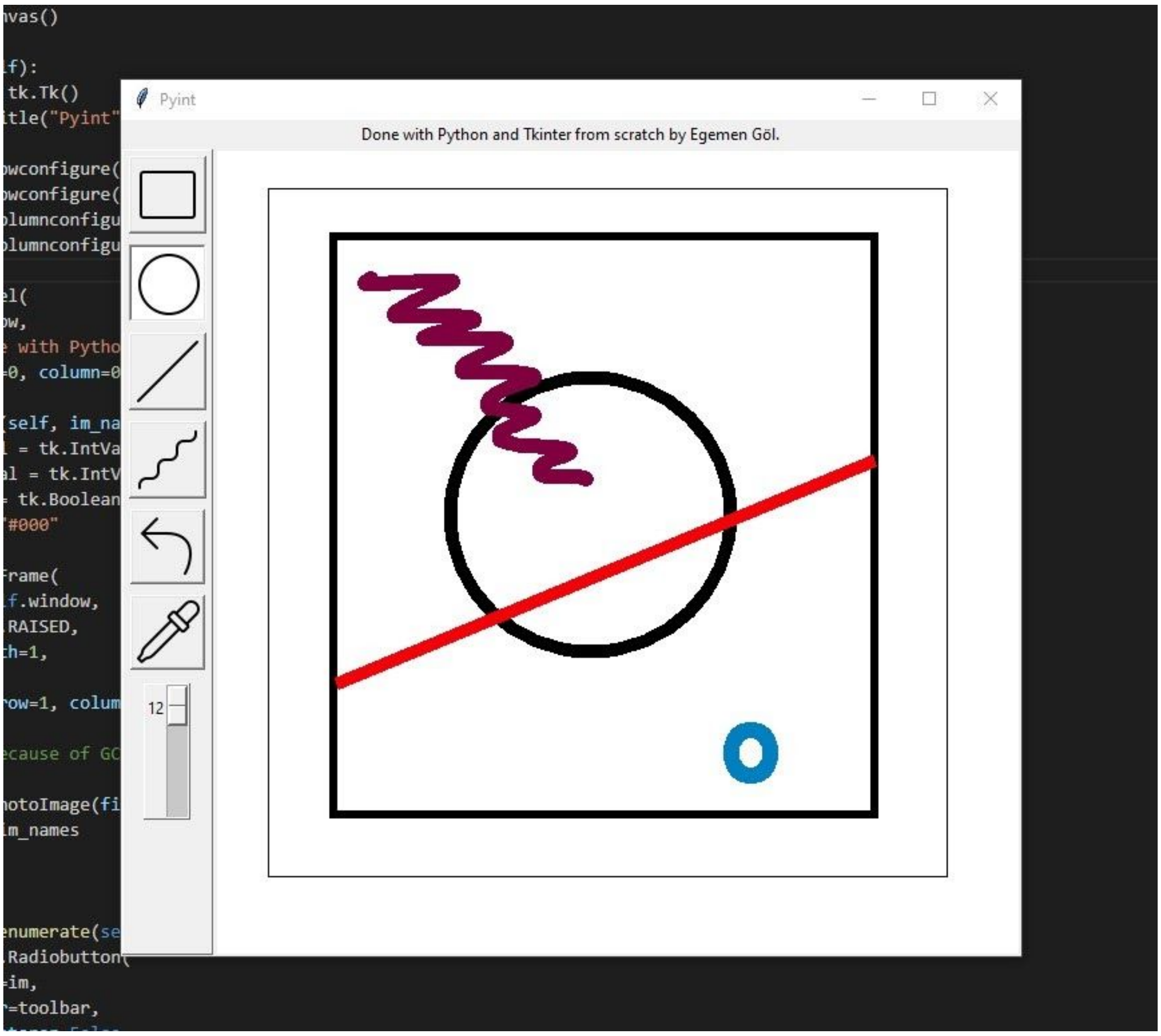
/paɪnt/

A basic, interactive graphics tool

Egemen Göl

https://github.com/egemengol/pyint

In this project, I implemented a Microsoft Paint imitation which can draw rectangles, circles, lines and free-draw shapes to the canvas with varying widths and colors.



```
nvas()

lf):
 tk.Tk()
itle("Pyint"

owconfigure(
owconfigure(
olumnconfigu
olumnconfigu

el(
ow,
e with Pytho
=0, column=0

(self, im_na
l = tk.IntVa
al = tk.IntV
= tk.Boolean
"#000"

Frame(
lf.window,
.RAISED,
th=1,

row=1, colum

ecause of GC

hotoImage(fi
im_names



enumerate(se
.Radiobutton(
=im,
r=toolbar,
```

Done with Python and Tkinter from scratch by Egemen Göl.

Capabilities

- Draw
    - Rectangles
    - Ovals
    - Lines
    - Free-form shapes
- Can vary
    - Line widths
    - Colors
- Also can undo one move when the move is not a free-draw.

Depended on

- Tkinter
- Pillow (for .png files)
- Icons from icons8.com

Implementation Journey

I have chosen Python for my preferred language for this project for its superior development speed and its rich ecosystem in which I can find numerous high-quality libraries and people that uses those.

I also have far superior experience in Python compared to Java, but this has played a lesser role than I envisioned because of the radically different event-based GUI programming paradigm which I was not accustomed before.

I used the Object-Oriented design for my project for mainly encapsulation purposes, both between the program and the class, and between functions where variables make sense only in that function’s scope.

My GUI library of choice is Tkinter because it is said to be one of the simplest GUI libraries which is also cross-platform. Since it is simple, I guessed that the development process should be fast.

It turns out that I have overlooked one important factor, which is the lack of quality documentation, which hindered my attempts of doing this project in a speedy manner. I did not let it bother me, I had done nearly half of the project when the realization hit.

Tkinter looks old, since it uses native GUI backends, it only has support for old windows GUI. I used icons for buttons, so it gave the window a modern, minimalistic look, which I liked immensely.

I searched the Internet and looked around for people's related Tkinter projects for examples on how to use the library. I took inspirations from other (even simpler) paint implementations and a (very simple) text editor implementation. They were very helpful, I love open source! :)

Building free-form drawing was fun, since the library had no support for it, I used small (but width-adjustable) ovals for every pixel the mouse-drag gesture touches.

The library also had no support for on-the-fly representation of shapes, it just drew what four points you gave it. I worked around this issue by deleting the last shape and drawing a new one, until you release the mouse.

The undo feature also works by deleting the last shape, but free-shapes are formed from multiple ovals so this strategy does not work. In the end, I disabled undo whenever a free-shape gets drawn. Implementing undo on that procedure would need much work over the simple interface the library exposes.

There is no eraser, a user can draw with white whenever she likes to. A separate button might be useful, but redundant. I decided an eraser feature needs a better process of erasing, so no eraser until I find a reliable way to erase shapes.

Overall, this was a fun project with a visual output, which is an especially fun and rare treat nowadays. Programming GUIs require manual testing in my limited experience with this library, which is a laborious process. On the other hand, having a visual representation is enjoyable. Would do again.

Pyint

```python
import tkinter as tk
from tkinter.colorchooser import askcolor
from PIL import Image, ImageTk
from enum import Enum


im_names = [
    "rect",
    "circ",
    "line",
    "draw",
    "undo",
    "color",
]


class Radio(Enum):
    NULL = 0
    RECT = 1
    CIRCLE = 2
    LINE = 3
    DRAW = 4


class Paint:
    def __init__(self, im_names):
        self.last_item = None
        self.setup_grid()
        self.setup_toolbar(im_names)
        self.setup_canvas()

    def setup_grid(self):
        self.window = tk.Tk()
        self.window.title("Pyint")

        self.window.rowconfigure(0, weight=0)
        self.window.rowconfigure(1, weight=0, minsize=400)
        self.window.columnconfigure(0, minsize=60, weight=0)
        self.window.columnconfigure(1, minsize=200, weight=1)

        info = tk.Label(
            self.window,
            text="Done with Python and Tkinter from scratch by Egemen Göl.")
        info.grid(row=0, column=0, columnspan=2)

    def setup_toolbar(self, im_names):
        self.radio_val = tk.IntVar()
        self.slider_val = tk.IntVar()
        self.is_undo = tk.BooleanVar()
        self.color = "#000"

        toolbar = tk.Frame(
            master=self.window,
            relief=tk.RAISED,
            borderwidth=1,
        )
        toolbar.grid(row=1, column=0, sticky="ns", padx=1, pady=1)

        # only self because of GC
        self.ims = [
            ImageTk.PhotoImage(file=f"static/{f}.png")
            for f in im_names
        ]


        for i, im in enumerate(self.ims[:-2], 1):
            rdio = tk.Radiobutton(
                image=im,
                master=toolbar,
                indicatoron=False,
                relief=tk.RAISED,
                value=i,
                variable=self.radio_val,
            )
            rdio.pack(padx=4, pady=4)

        self.btn_undo = tk.Button(
            image=self.ims[-2],
            master=toolbar,
            relief=tk.RAISED,
            state=tk.DISABLED,
            command=self.handle_undo
        )
        self.btn_undo.pack(padx=4, pady=4)


        btn_color = tk.Button(
            image=self.ims[-1],
            master=toolbar,
            relief=tk.RAISED,
            command=self.handle_color,
        )
        btn_color.pack(padx=4, pady=4)


        slider = tk.Scale(
            master=toolbar,
            relief=tk.RAISED,
            from_=12,
            to=1,
            orient=tk.VERTICAL,
            variable=self.slider_val,
        )
        slider.pack(padx=4, pady=4)

    def setup_canvas(self):
        self.canvas = tk.Canvas(master=self.window, width=600, height=600, bg="white")
        self.canvas.grid(row=1, column=1, sticky="nw")

        self.canvas.bind("<Button-1>", self.handle_press)
        self.canvas.bind("<ButtonRelease-1>", self.handle_release)
        self.canvas.bind("<B1-Motion>", self.handle_drag)

    def mainloop(self):
        self.window.mainloop()

    def handle_press(self, event):
        self.pressed_x = event.x
        self.pressed_y = event.y
        self.last_item = None

    def handle_release(self, event):
        if not Radio(self.radio_val.get()) == Radio.DRAW:
            self.btn_undo.config(state=tk.NORMAL)

    def handle_drag(self, event):
        if not Radio(self.radio_val.get()) == Radio.DRAW:
            self.canvas.delete(self.last_item)
        self.last_item = self.draw(event.x, event.y)


    def draw(self, x, y): # -> itemid
        radio = Radio(self.radio_val.get())
        if radio == Radio.LINE:
            return self.canvas.create_line(
                self.pressed_x,
                self.pressed_y,
                x, y,
                width=self.slider_val.get(),
                fill=self.color,
            )
        elif radio == Radio.RECT:
            return self.canvas.create_rectangle(
                self.pressed_x,
                self.pressed_y,
                x, y,
                width=self.slider_val.get(),
                outline=self.color,
            )
        elif radio == Radio.CIRCLE:
            return self.canvas.create_oval(
                self.pressed_x,
                self.pressed_y,
                x, y,
                width=self.slider_val.get(),
                outline=self.color,
            )
        elif radio == Radio.DRAW:
            self.btn_undo.config(state=tk.DISABLED)
            width = self.slider_val.get() // 2
            return self.canvas.create_oval(
                x - width,
                y - width,
                x + width,
                y + width,
                fill=self.color,
                outline=self.color,
            )

    def handle_undo(self):
        self.canvas.delete(self.last_item)
        self.btn_undo.config(state=tk.DISABLED)

    def handle_color(self):
        self.color = askcolor(
            self.color,
            title="Please choose a color",
        )[1]



if __name__ == "__main__":
    paint = Paint(im_names)
    paint.mainloop()
```