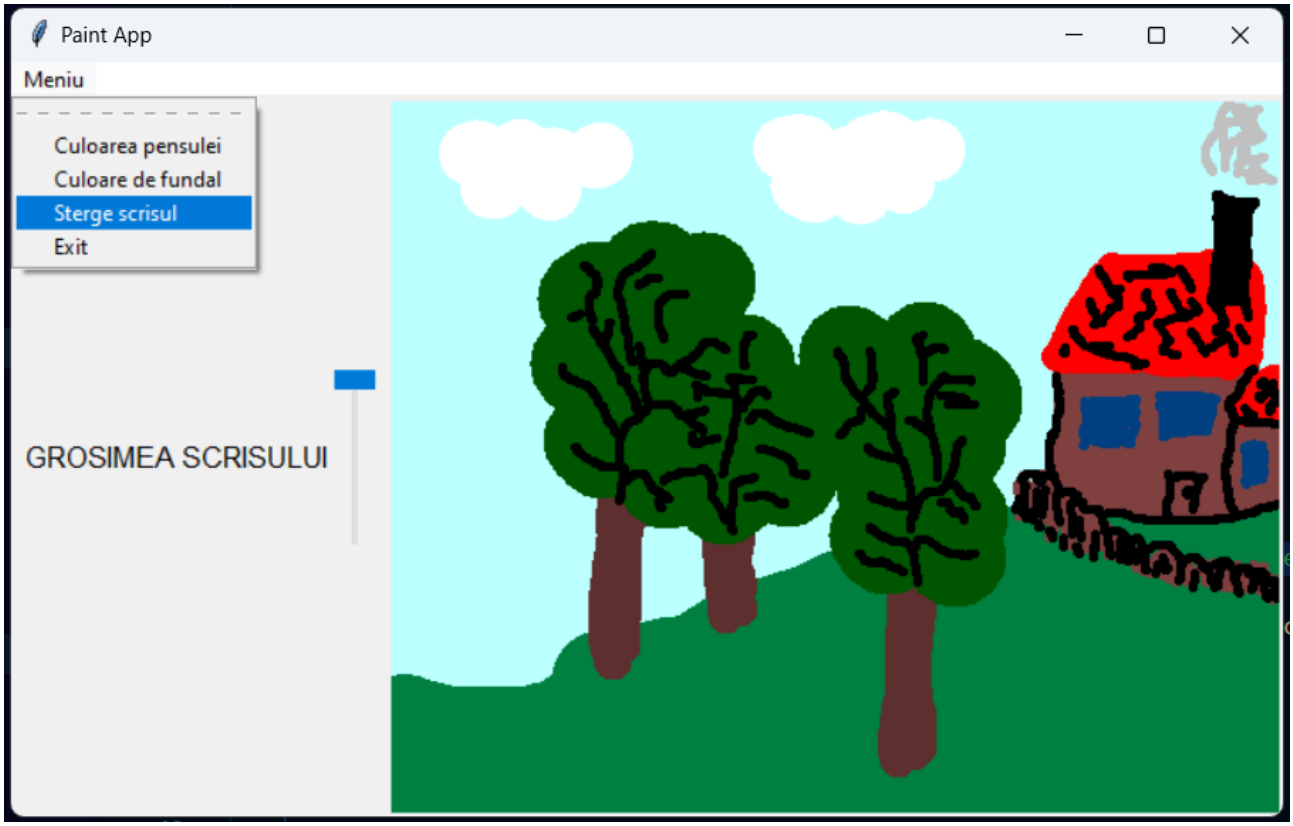
Agapie_Andrei-Cosmin

1. **Cerința**
   Sa se creeze o aplicatie de tip Paint.

2. **Tehnologii utilizate**
   Pentru a edita codul, s-a folosit Visual Studio Code, un editor de cod sursă dezvoltat de Microsoft pentru Windows, Linux si macOS. În aplicație am instalat extensia Python pentru a putea lucra in acest limbaj de programare.
   Pentru a dezvolta platforma pentru Paint App am ales sa utilizam bibloteca tkinter prezenta in Phyton.
3. **Descrierea aplicației**

3.1. Interfața



Interfața programului Paint App creat cu Tkinter include următoarele elemente:

Zona de Desenare :

Este zona principală în care utilizatorul poate desena.

Inițial are o dimensiune de 500x400 de pixeli și un fundal alb ('white'), dar fundalul poate fi schimbat utilizând funcționalitatea din meniu.

Meniul:

Meniul se află în partea de sus a ferestrei și include opțiunile:

"Meniu" - deschide un meniu derulant cu opțiunile:

"Culoarea pensulei" - permite utilizatorului să aleagă o nouă culoare pentru pensulă, folosind o fereastră de selecție a culorii.

"Culoare de fundal" - permite utilizatorului să aleagă o nouă culoare pentru fundalul canvasului, utilizând o fereastră de selecție a culorii.

"Sterge scrisul" - șterge tot conținutul de pe canvas, oferind un mod simplu de a începe o nouă desenare, dar nu si fundalul

"Exit" - închide aplicația.

Panoul de Control :

Afișat în partea stângă a ferestrei.

Include un label pentru grosimea peniței ("GROSIMEA SCRISULUI") și un slider vertical pentru a regla grosimea peniței.

Valoarea inițială a grosimii peniței este 5, dar utilizatorul o poate modifica utilizând sliderul.

Aceste elemente îți oferă o interfață simplă și funcțională pentru desenarea în Tkinter. Utilizatorii pot schimba culorile peniței și fundalului, regla grosimea peniței, șterge canvas-ul și închide aplicația.

3.2.Arhitectura Sistem

Arhitectura utilizată în programul Paint App este cunoscută sub numele de arhitectură orientată pe obiecte (POO), specific în limbajul de programare Python cu Tkinter. În acest program, obiectele sunt definite prin intermediul unei clase principale denumite main. Iată câteva elemente cheie ale acestei arhitecturi:

Clasa Principală (main):

POO este implementată folosind clasa main. Aceasta servește ca țesătură principală a programului și encapsulează toate funcționalitățile și atributele necesare.

Constructorul inițializează obiectul, stabilește valorile implicite și apelează metoda drawWidgets pentru a construi interfața grafică.

Metode ale Clasei (paint, reset, changedW, clearcanvas, etc.)

3.3.Prezentare funcționalități

Funcționalitățile principale ale aplicației sunt separate în metode ale clasei. De exemplu, paint este responsabilă pentru desenarea pe canvas, reset pentru resetarea coordonatelor, changedW pentru actualizarea grosimii peniței, etc.

Interfața Utilizatorului (UI):

Interfața utilizatorului este definită și construită în metoda drawWidgets. Aceasta include canvas-ul pentru desenare, un panou de control cu un slider pentru grosimea peniței, și un meniu cu opțiuni pentru a schimba culorile și a șterge canvas-ul.

Evenimente și Legături :

Evenimentele (cum ar fi mișcarea mouse-ului sau eliberarea butonului) sunt legate de metode specifice ale clasei folosind bind. De exemplu, '<B1-Motion>' și '<ButtonRelease-1>' sunt evenimente care declanșează metodele paint și reset, respectiv.

Utilizarea Obiectelor Tkinter:

Componentele grafice (cum ar fi Canvas, Frame, Label, Scale, etc.) sunt obiecte Tkinter care sunt utilizate pentru a construi interfața.

COD cu explicatii:

from tkinter import * #Importă toate funcțiile și clasele din modulul tkinter.

#Acesta este un mod convențional de a aduce toate funcționalitățile din tkinter în spațiul de nume curent.

from tkinter import colorchooser, ttk #Importă funcționalitatea pentru a alege culori (colorchooser)

#și componente specifice ttk (themed tkinter) care oferă widget-uri îmbunătățite grafic în comparație cu cele tradiționale.

class main: #Definește o clasă numită main.

def __init__(self, master): #Inițializează obiectul main. Constructorul primește un argument master, care ar trebui să fie obiectul părinte în care interfața grafică va fi afișată.

self.master = master #Stochează referința către obiectul părinte în atributul master al obiectului curent.

self.color_fg = 'Black' #Inițializează culoarea liniei de desenare la negru.

self.color_bg = 'white' # Inițializează culoarea fundalului la alb.

self.old_x = None #Inițializează variabilele pentru coordonatele anterioare ale mouse-ului cu None.

self.old_y = None

self.pen_width = 5 #Inițializează grosimea creionului la 5 pixeli.

self.drawWidgets() # Invocă metoda drawWidgets(), care va crea și afișa widget-urile pentru interfața grafică.

self.c.bind('<B1-Motion>', self.paint) #Leagă evenimentul de mișcare a mouse-ului la metoda self.paint, astfel încât să poată desena atunci când butonul stâng al mouse-ului este ținut apăsat și mouse-ul se mișcă.

self.c.bind('<ButtonRelease-1>', self.reset) #Leagă evenimentul de eliberare a butonului stâng al mouse-ului la metoda self.reset, care va actualiza coordonatele anterioare ale mouse-ului.

def paint(self, e):

#Verifică dacă există coordonate anterioare (dacă nu este primul punct desenat)

if self.old_x and self.old_y:# Creează o linie între coordonatele anterioare și cele actuale ale mouse-ului

self.c.create_line(self.old_x, self.old_y, e.x, e.y, width = self.pen_width, fill = self.color_fg, capstyle='round', smooth = True) #Utilizează metoda create_line pentru a desena o linie pe canvas (self.c). Parametrii precum self.old_x și self.old_y reprezintă coordonatele anterioare ale mouse-ului, în timp ce e.x și e.y reprezintă coordonatele curente ale mouse-ului. Alți parametri precum width, fill, capstyle și smooth sunt utilizați pentru a specifica grosimea liniei, culoarea, stilul capului de linie și nivelul de netezire al liniei.

#Actualizează coordonatele anterioare la poziția curentă a mouse-ului pentru a fi utilizate în următoarea iterație a desenării.

self.old_x = e.x

self.old_y = e.y

#Setează coordonatele anterioare la None. Acest lucru indică că următoarea mișcare de desenare începe un nou traseu, astfel încât prima pereche de coordonate să nu fie conectată la traseul anterior. Aceasta este o modalitate de a separa liniile desenate pe canvas.

def reset(self, e):

self.old_x = None

self.old_y = None

#Această metodă presupune că width este o valoare numerică care reprezintă grosimea dorită a creionului. Atunci când este apelată, această metodă actualizează atributul self.pen_width al obiectului curent la valoarea specificată. Grosimea creionului este apoi utilizată în metoda paint pentru a controla grosimea liniilor desenate pe canvas.

def changedW(self, width):

self.pen_width = width

#Metoda delete a canvas-ului (self.c) este utilizată cu argumentul ALL, ceea ce înseamnă că se vor șterge toate obiectele desenate pe canvas. Prin apelul acestei metode, interfața grafică va fi curățată, eliminând toate liniile desenate sau alte obiecte existente pe canvas.

```
    def clearcanvas(self):
        self.c.delete(ALL)
    def change_fg(self):
        #Utilizatorul poate alege o nouă culoare pentru linia de desen
        self.color_fg = colorchooser.askcolor(color=self.color_fg)[1]
    def change_bg(self):
        ## Utilizatorul poate alege o nouă culoare pentru fundal
        self.color_bg = colorchooser.askcolor(color=self.color_bg)[1]
        # Actualizează culoarea de fundal a canvas-ului
        self.c['bg'] = self.color_bg
    def drawWidgets(self):
        # Creează un cadru pentru controale cu un spațiu de margini
        self.controls = Frame(self.master, padx=5, pady=5)
        # Eticheta pentru indicarea grosimii liniei
        textpw = Label(self.controls, text='GROSIMEA SCRISULUI', font='TimesNewRoman 12')
        textpw.grid(row=0, column=0)
         # Creează un obiect de tip Scale pentru a permite utilizatorului să aleagă grosimea liniei
        self.slider = ttk.Scale(self.controls, from_=5, to=100, command=self.changedW,
orient='vertical')
        self.slider.set(self.pen_width)# Setează valoarea inițială a slider-ului la grosimea curentă a
liniei
        self.slider.grid(row=0, column=1)
        # Plasează cadru cu controale pe partea stângă a ferestrei principale
        self.controls.pack(side="left")
        # Creează un canvas pentru desenare cu o dimensiune inițială și culoare de fundal
        self.c = Canvas(self.master, width=500, height=400, bg=self.color_bg)
        self.c.pack(fill=BOTH, expand=True)# Plasează canvas-ul în fereastra principală, lărgindu-l și
umplând spațiul disponibil


        menu = Menu(self.master) # Creează un obiect de tip Menu
        self.master.config(menu=menu)# Configurează meniul pentru fereastra principală
        optionmenu = Menu(menu)# Creează un meniu  pentru opțiuni
```

```
        menu.add_cascade(label='Menu', menu=optionmenu)# Adaugă meniul pentru opțiuni la meniul principal
        # Adaugă opțiuni în meniu
        optionmenu.add_command(label='Culoarea pensulei', command=self.change_fg)
        optionmenu.add_command(label='Culoare de fundal', command=self.change_bg)
        optionmenu.add_command(label='Sterge scrisul', command=self.clearcanvas)
        optionmenu.add_command(label='Exit', command=self.master.destroy)


win = Tk() # Creează o fereastră principală
win.title("Paint App") # Configurează titlul ferestrei
main(win)  # Inițializează clasa main pentru a gestiona interfața de desen
win.mainloop() # Intră în bucla principală de evenimente pentru a afișa și interacționa cu aplicația
```