KEREKES PATRIK (JMN1RF)
SZABÓ PATRIK (DEUHZ9)

NEUMANN JÁNOS EGYETEM
GAMF MŰSZAKI ÉS INFORMATIKAI KAR

Gyakorlatvezető:
AGG PÉTER

Feladat:
KÉPSZERKESZTŐ ALKALMAZÁS KÉSZÍTÉSE


Photo editor v2.2
Photo Editor
Open New Image
Undo
Redo
Rotate
Black and white
Blur
Blur value
0
Flip
Change brightness
Brightness value
1.3
Change sharpness
Sharpness value
3.7
Change contrast
Contrast value
1.9
Save
Select extension
jpg

NEUMANN
JÁNOS
EGYETEM

Tartalom

Fejlesztői dokumentáció

Feladat:

Egyszerű, de több feladatra használható képszerkesztő alkalmazás készítése (PIL és Tkinter segítségével)

Környezet:

Visual Studio 2019

Állományok:

PhotoEditor.py

Főprogram, a program főosztálya és függvényei

ImageHistory.py

A visszavonási műveletekért felelős állomány

Program felépítése:

Felhasznált csomagok

tkinter, PIL

```
from tkinter import *
from tkinter import filedialog as fd
from tkinter import messagebox as ms
import PIL
from PIL import Image, ImageTk, ImageFilter, ImageEnhance
from ImageHistory import ImageHistory
```

A tkinter csomagot használtuk a grafikus megjelenítéshez, a PIL csomagból a különböző képi effekteket (fekete-fehér, homályosítás stb.).

Osztályok

PhotoEditor.py

Főprogram

```
root=Tk()
root.configure(bg='snow3')
root.title('Photo editor v2.2')
application(root)
FullScreenApp(root)
root.resizable(0,0)
root.mainloop()
```

application

Ez a program főosztálya, itt hozzuk létre a felhasználói felületet, a képen való módosításokat, illetve a mentést.

Tulajdonságok:

```
        self.master = master
        self.width = master.winfo_screenwidth() - 375
        self.height = master.winfo_screenheight() - 180
        self.maxwidth = self.width
        self.maxheight = self.height
        self.appHeight = master.winfo_screenheight() - 100
        self.blurvalue = IntVar()
        self.brightnessvalue = DoubleVar()
        self.sharpnessvalue = DoubleVar()
        self.contrastvalue = DoubleVar()
        self.history = ImageHistory(10)
        self.extension = StringVar()
        self.setup_gui(self.width, self.height, self.appHeight)
```

A master a tkinter ablaka, ez a program gyökere.

Különböző méreteket vettünk fel, a width és height megkapják a képernyő méretét, majd ezt átadják a maxheight és maxwidth-nek, ekkorára csökken majd a megnyitott kép, hogy elférjen a programban.

A blur-, brightness-, sharpness- és contrastvalue tárolják a homályosítás, fényesítés, élesítés és a kontraszt értékeit. Az értéket mindegyiknél egy csúszka állítja, ami egy Var típust ad vissza, a blur Int-et vár, a többi Double-t.

A history tárolja a módosításokat.

Az extension az elmentendő kép kiterjesztéséért felelős.

A setup_gui hozza létre a felületet.

Függvények

```
 def setup_gui(self,w, h, appH):

         Label(self.master,text = 'Photo Editor',pady=5,bg='grey',
            font=('Courier new',30)).pack(fill=X)

        txt = "No image"

        # Betűméret beállítása a képernyő méretéhez igazítva
        font_size = IntVar()
        font_size = int(appH/65)

        f = Frame(self.master,bg='grey',padx=10,pady=10, bd = 5) #file megnyitás ás effektek frame-je
        f2 = Frame(self.master,bg='grey',padx=10,pady=10, bd = 5) #undo-redo frame-je
        f3 = Frame(self.master, bg='black', padx=10, pady=10, bd=3) #kép frame-je

        self.canvas = Canvas(f3,height=h ,width=w,
            bg='black',relief='ridge')
        self.wt = self.canvas.create_text(self.width/2, self.height/2 ,text=txt
            ,font=('',30),fill='white')
        self.canvas.pack()

        #file megnyitás gombja
        Button(f,text='Open New Image',bd=2,fg='black',bg='gray80',font=('',font_size) ,command=self.make_image,
pady=font_size * 1.5).pack(side=TOP, fill=X)

        #effketek gombjai
        Button(f,text='Rotate',bd=2,fg='white',bg='black',font=('',font_size)
```

```
            ,command=self.rotate_image).pack(side=TOP, fill=X)
        Button(f,text='Black and white',bd=2,fg='white',bg='black',font=('',font_size)
            ,command=self.make_blacknwhite).pack(side=TOP, fill=X)
        Button(f,text='Blur',bd=2,fg='white',bg='black',font=('',font_size)
            ,command=self.make_blur).pack(side=TOP, fill=X)
        Scale(f, sliderlength = 5, orient = HORIZONTAL, resolution=1, from_=0, to_=5, variable = self.blurvalue,
label='Blur value').pack(side = TOP, fill=X)
        Button(f,text='Flip',bd=2,fg='white',bg='black',font=('',font_size)
            ,command=self.flip).pack(side=TOP, fill=X)
        Button(f,text='Change brightness',bd=2,fg='white',bg='black',font=('',font_size)
            ,command=self.brightness).pack(side=TOP, fill=X)
        scale1 = Scale(f, sliderlength = 5, orient = HORIZONTAL, resolution=0.1, from_=0, to_=5, variable =
self.brightnessvalue, label='Brightness value')
        scale1.set(1)
        scale1.pack(side=TOP, fill=X)
        Button(f,text='Change sharpness',bd=2,fg='white',bg='black',font=('',font_size)
            ,command=self.sharpness).pack(side=TOP, fill=X)
        scale2 = Scale(f, sliderlength = 5, orient = HORIZONTAL, resolution=0.1, from_=0, to_=10, variable =
self.sharpnessvalue, label='Sharpness value')
        scale2.set(0)
        scale2.pack(side=TOP, fill=X)
        Button(f,text='Change contrast',bd=2,fg='white',bg='black',font=('',font_size)
            ,command=self.contrast).pack(side=TOP, fill=X)
        scale3 = Scale(f, sliderlength = 5, orient = HORIZONTAL, resolution=0.1, from_=-10, to_=10, variable =
self.contrastvalue, label='Contrast value')
        scale3.set(0)
        scale3.pack(side=TOP, fill=X)

        #mentés
        self.extension.set("jpg")
        OptionMenu(f, self.extension, "jpg", "png", "bmp").pack(side=BOTTOM, fill=X)
        Label(f, text='Select extension', bg='snow3', font=("Ariel", font_size)).pack(side=BOTTOM, fill=X)
        Button(f,text='Save',bd=2,fg='white',bg='black',font=('',font_size)
            ,command=self.save).pack(side=BOTTOM, fill=X)

        #undo-redo gombok
        Button(f2,text='Undo', bd=2,fg='white',bg='black',font=('',15)
            ,command=self.undo).pack(side=TOP, fill=X)
        Button(f2,text='Redo', bd=2,fg='white',bg='black',font=('',15)
            ,command=self.redo).pack(side=TOP, fill=X)

        #frame-ek elhelyezése
        f.pack(side = LEFT, fill=Y)
        f2.pack(side=LEFT, fill=Y)
        f3.pack(side=RIGHT, expand=1)
```

Ez függvény hozza létre az összes widget-et, amivel a program működik. Az alkalmazás alapvetően két fő részre van bontva, a gombokra amik vezérlik, és egy nagy felületre ahol láthatjuk a képet amit módosítunk.

Minden gomb meghívja a command paraméterével az aktuális függvényt, amit használni kell.

Néhány effekt igényel egy változót is, ezeket a megfelelő csúszkával lehet állítani, az adatot a gomb megnyomásával adja át.

```
    def make_image(self):
        try:
            File = fd.askopenfilename(filetypes=[('Pictures', "*.png | *.PNG | *.jpg | *.JPG |
*.bmp | *.BMP"), ('All files', '*')])
            self.pilImage = Image.open(File)    # eredeti nagy kép betöltése
            self.resizedImage_to_canvas()       # eredeti -> resizedImage -> img -> canvas
            self.history.AddImageToHistory(self.pilImage)   # eredeti kép -> history
        except:
            ms.showerror('Error!','File type is unsupported.')
```

Ez a függvény hozza létre a képet, fájlkezelőből lehet kiválasztani egy képet majd létrehoz egy pilImage változót, amin majd dolgozunk.

```
def resizedImage_to_canvas(self):
    self.resize_pilImage()  # eredeti(nagy) kép -> resizedImage
    self.img = ImageTk.PhotoImage(self.resizedImage)
    self.canvas.delete(ALL)
    self.canvas.config(width=self.width, height=self.height)
    self.canvas.create_image(0, 0, anchor=NW, image=self.img)
```

Itt helyezzük rá a képet a canvas-re, ahol láthatjuk magát a képet, és a módosításainkat.

```
def resize_pilImage(self):
    w, h = self.pilImage.size
    self.width = w
    self.height = h
    while self.width > self.maxwidth or self.height > self.maxheight -1:
        self.width = int(self.width//1.01)
        self.height = int(self.height//1.01)
    self.resizedImage = self.pilImage.resize((self.width, self.height),Image.NEAREST)
```

Mivel bármekkora képet megnyithatunk, így az nem feltétlenül fog rá férni a képernyőnkre (pl 4k-s kép), ezért le kell csökkenteni a kép méretét. A főosztályban definiált maximális méretig csökkenti a képet egészosztással, amíg rá nem fér pontosan a canvas-re.

Az effekt függvények alapja mindegyiknél ugyanaz:

```
def <függvénynév>(self):
    try:
        self.pilImage = <kód>
        self.resizedImage_to_canvas()
        self.history.AddImageToHistory(self.pilImage)
    except:
        ms.showerror('No photo', 'Select something')
```

Az eredeti képünket módosítja, ráteszi a canvas-re átméretezve, majd hozzáadja a history-hoz. Ha nincs betöltve kép hibaüzenetet ad. A különböző effektek kódrészletei:

Forgatás: `self.pilImage.transpose(Image.ROTATE_90)`
Fekete-fehér beállítása: `self.pilImage.convert('L')`
Homályosítás: `self.pilImage.filter(ImageFilter.GaussianBlur(radius = self.blurvalue.get()))`
Tükrözés: `self.pilImage.transpose(Image.FLIP_LEFT_RIGHT)`
Fényerő: `ImageEnhance.Brightness(self.pilImage).enhance(self.brightnessvalue.get())`
Élesítés: `ImageEnhance.Sharpness(self.pilImage).enhance(self.sharpnessvalue.get())`
Kontraszt: `ImageEnhance.Contrast(self.pilImage).enhance(self.contrastvalue.get())`

undo(self):
Meghívja az ImageHistory osztály Undo() függvényét, ennek a visszatérési értékét átadja a pilImage-nek, majd azt méretre szabva a Canvas-re helyezi.

```
# Visszavonás
    def undo(self):
        self.pilImage = self.history.Undo() # előző kép betöltése
        self.resizedImage_to_canvas()  # betöltött kép -> kicsinyített kép -> canvas
```

redo(self):
Meghívja az ImageHistory osztály Undo() függvényét, ennek a visszatérési értékét átadja a pilImage-nek, majd azt méretre szabva a Canvas-re helyezi.

```
    # Újra
    def redo(self):
        self.pilImage = self.history.Redo() # következő kép betöltése
        self.resizedImage_to_canvas()  # betöltött kép -> kicsinyített kép -> canvas
```

FullScreenApp

Ez az osztály felelős hogy az alkalmazás teljes képernyőn jelenjen meg.

```
    def __init__(self, master, **kwargs):
        self.master=master
        pad=40
        master.geometry("{0}x{1}+0+0".format(
            master.winfo_screenwidth(), master.winfo_screenheight()-100))
```

Forrás: https://stackoverflow.com/questions/7966119/display-fullscreen-mode-on-tkinter
Lekéri az éppen használt képernyő méretét, majd a tkinter alkalmazás méretét erre változtatja.

ImageHistory.py

ImageHistory

Tulajdonságok

```
        self.maxNumberOfSteps = maxSteps
        self.images = []
        self.historyIndex = -1
```

maxNumberOfSteps: az alkalmazás ennyi műveletet képes tárolni
images: a módosított kép egyes állapotait tároló tömb
historyIndex: az aktuális állapot indexe

Függvények

AddImageToHistory(self, image):
Egy új képet vesz fel a listába. Ha a historyIndex nem a lista utolsó elemére mutat, akkor az aktuális elem után minden továbbit eltávolítunk, ezután felvesszük az új képet. Ha a lista elérte a megadott maximális hosszt, a legrégebbi elemet töröljük.

```
    # Kép hozzáadása a listához
    def AddImageToHistory(self, image):
        # az aktuális kép után kiveszünk mindent a listából
        if len(self.images) - 1 > self.historyIndex:
            for i in range(self.historyIndex + 1, len(self.images)):
                self.images.pop(self.historyIndex + 1)
```

```
        # kép hozzáfűzése a listához
        self.images.append(image)
        self.historyIndex += 1

        # ha a lista elérte a maximális hosszt
        if len(self.images) > self.maxNumberOfSteps:
            self.images.pop(0)
            self.historyIndex -= 1
```

Undo(self):
Visszaadja a kép előző állapotát.

```
    # Visszavonás
    def Undo(self):
        if self.historyIndex > 0:   # ha elérte a 0 indexet
            self.historyIndex -= 1

        print("historyIndex: ")
        print(self.historyIndex)
        return self.images[self.historyIndex]
```

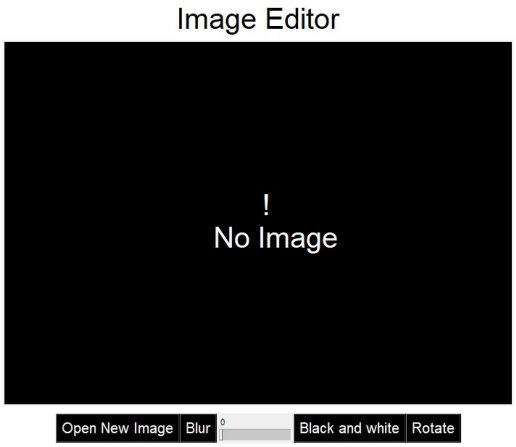
Redo(self):
Visszavonás után lehetőségünk van visszatérni az újabb állapotokhoz. A függvény a kép következő állapotát adja vissza.

```
    # Újra
    def Redo(self):
        self.historyIndex += 1
        if self.historyIndex == len(self.images):   # ha elérte a lista hosszát
            self.historyIndex -= 1

        print("historyIndex: ")
        print(self.historyIndex)
        return self.images[self.historyIndex]
```

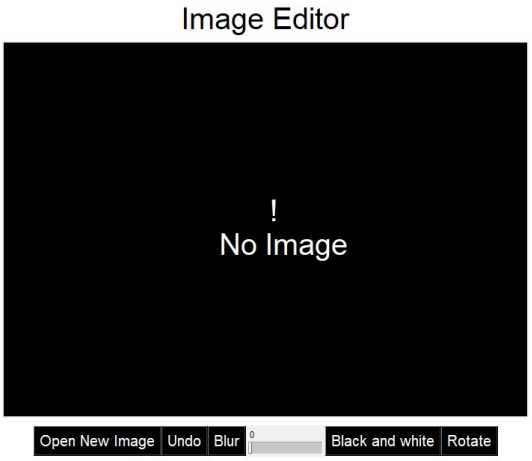
Verziótörténet

Photo Editor 1.1



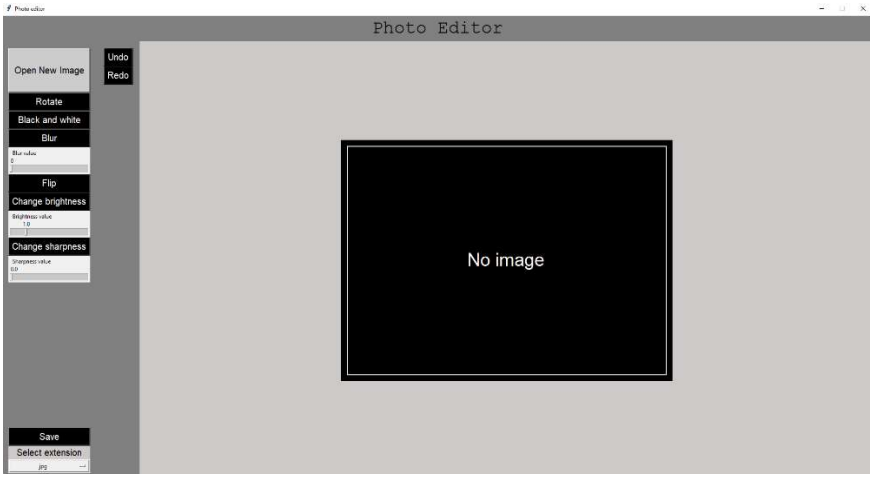
Ez volt az alkalmazás első verziója, még csak egyszerű függvényekkel, nincs mentési lehetőség, csak Full HD képernyőn működött.

Photo Editor 1.2



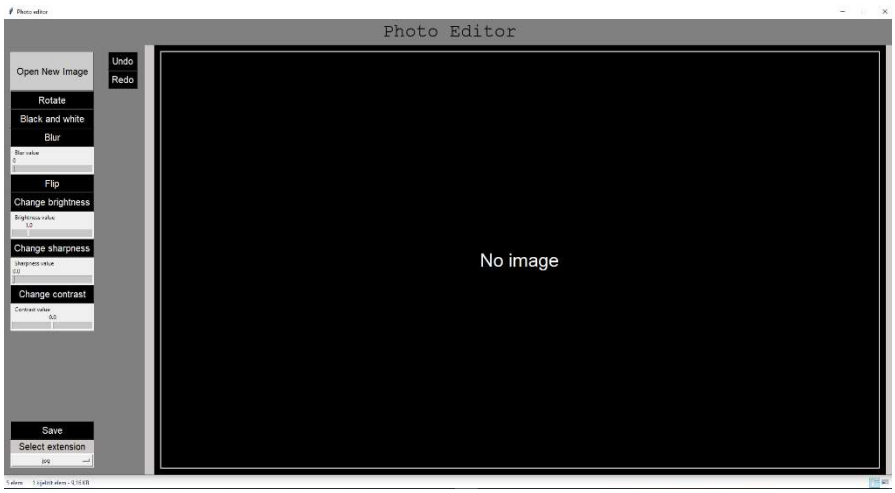
Ebben a verzióban már megjelent a visszavonási lehetőség, habár nem mindig működött jól(forgatás utáni műveleteket elfelejtette).

Photo Editor 2.1



Elkészült a felhasználói felület frissítése, a mentés és plusz effektek.

Photo Editor 2.2 (recent)

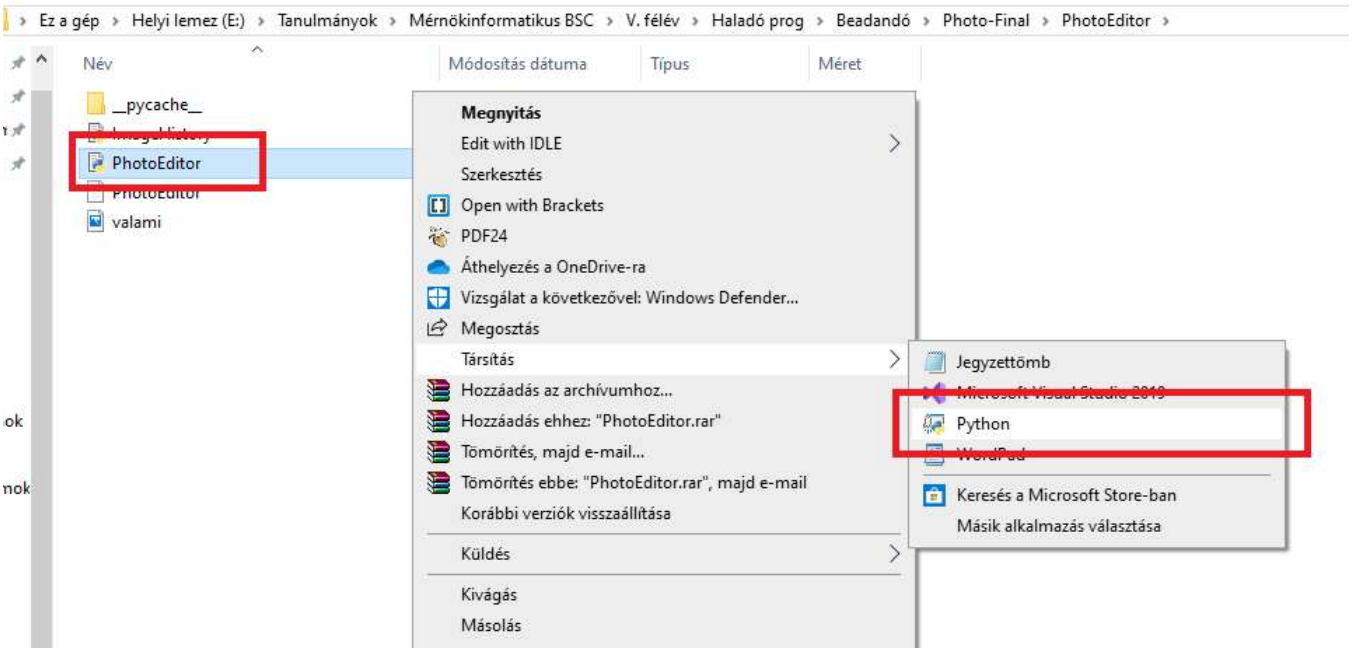


A gombok és a canvas is reszponzívak lettek, bármekkora képernyőn fut az alkalmazás, többféle fájltípust lehet megnyitni, illetve elmenteni, undo-redo bug javítások.

Felhasználói dokumentáció

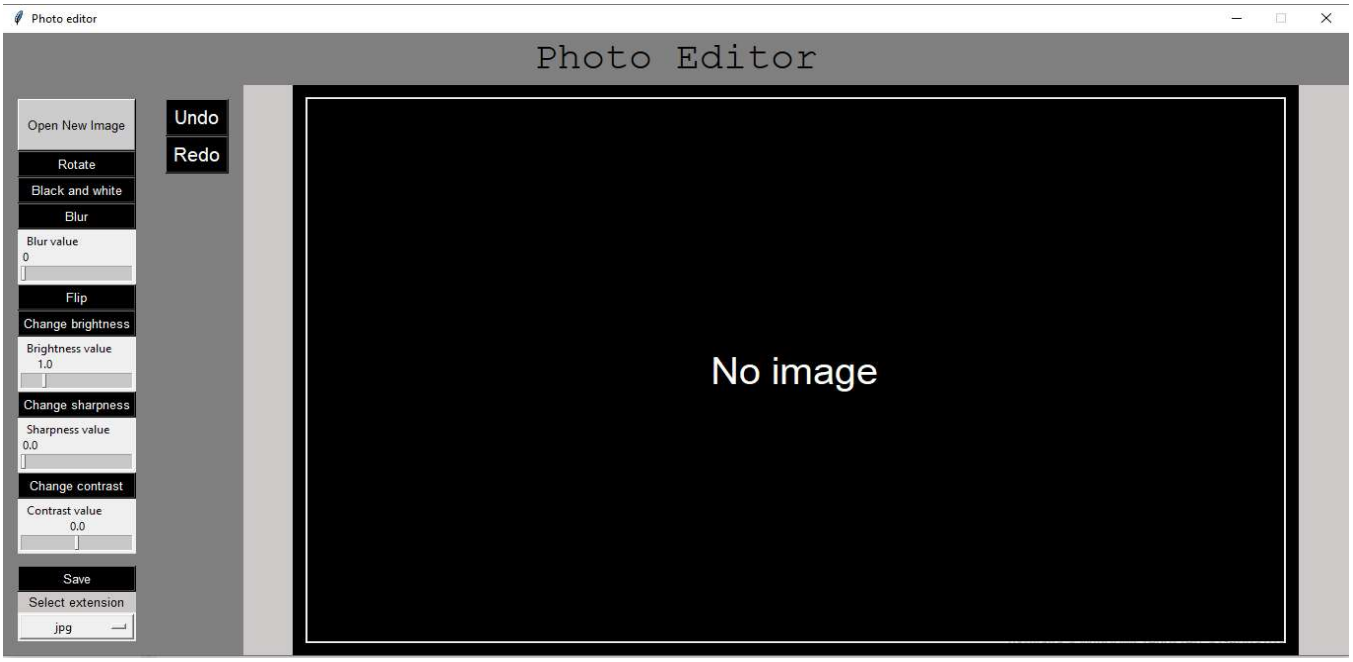
Alkalmazás indítása

Az alkalmazás futtatásához szükség van egy Python kód értelmezőre(interpreterre pl: py.exe). Ezzel kell futtatnunk a PhotoEditor.py állományt.



0.1 ábra: alkalmazás elindítása

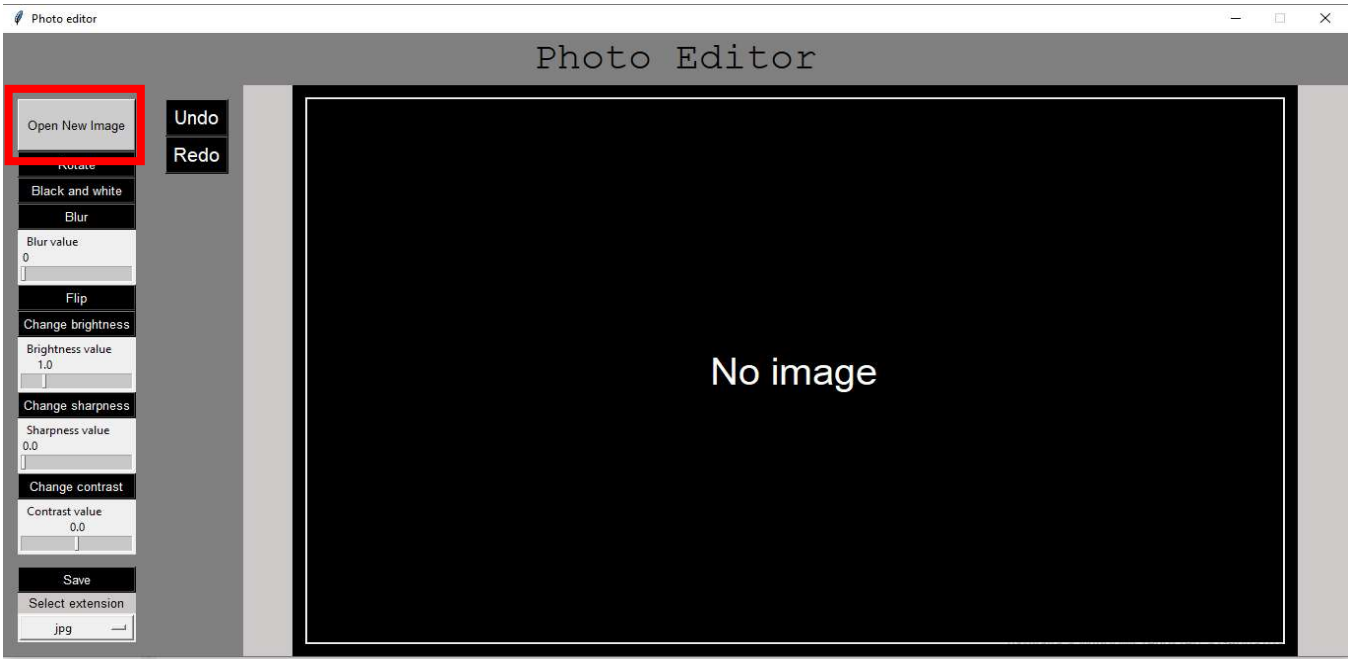
Kép betöltése

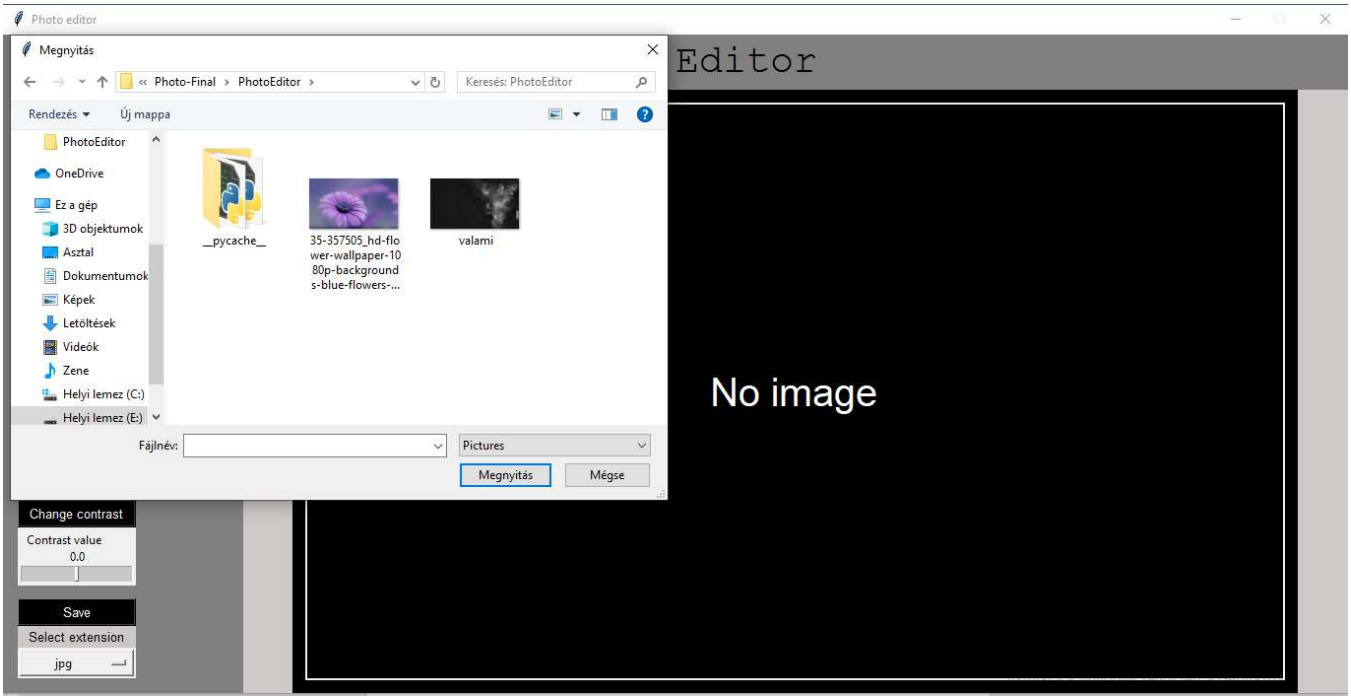


0.1 ábra: felhasználói felület

Az alkalmazás elindításakor a fenti képernyő fogad minket. Itt láthatjuk a felhasználói felületet, amely alapvetően három részre bomlik. Bal szélen találhatóak a különböző effektek gombjai, új kép betöltés gomb és a mentés gomb. Jobb szélen helyezkedik el a majd betöltött képünk, tőle balra pedig a visszavonás és ismét gombok láthatóak.

Új kép megnyitásához kattintsunk az „Open New Image” feliratú gombra. Ezután a felugró ablakban válasszuk ki a szerkeszteni kívánt képet.

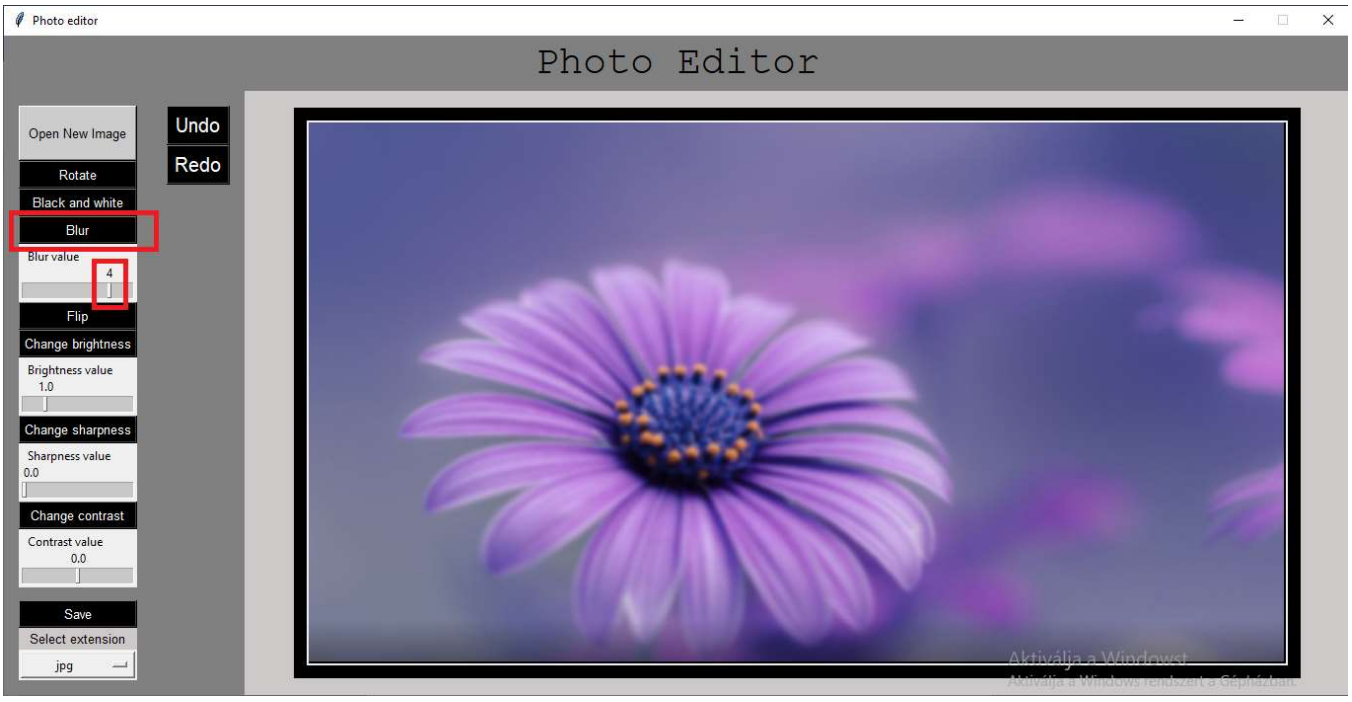




0.2 ábra: kép kiválasztása

Kép szerkesztése

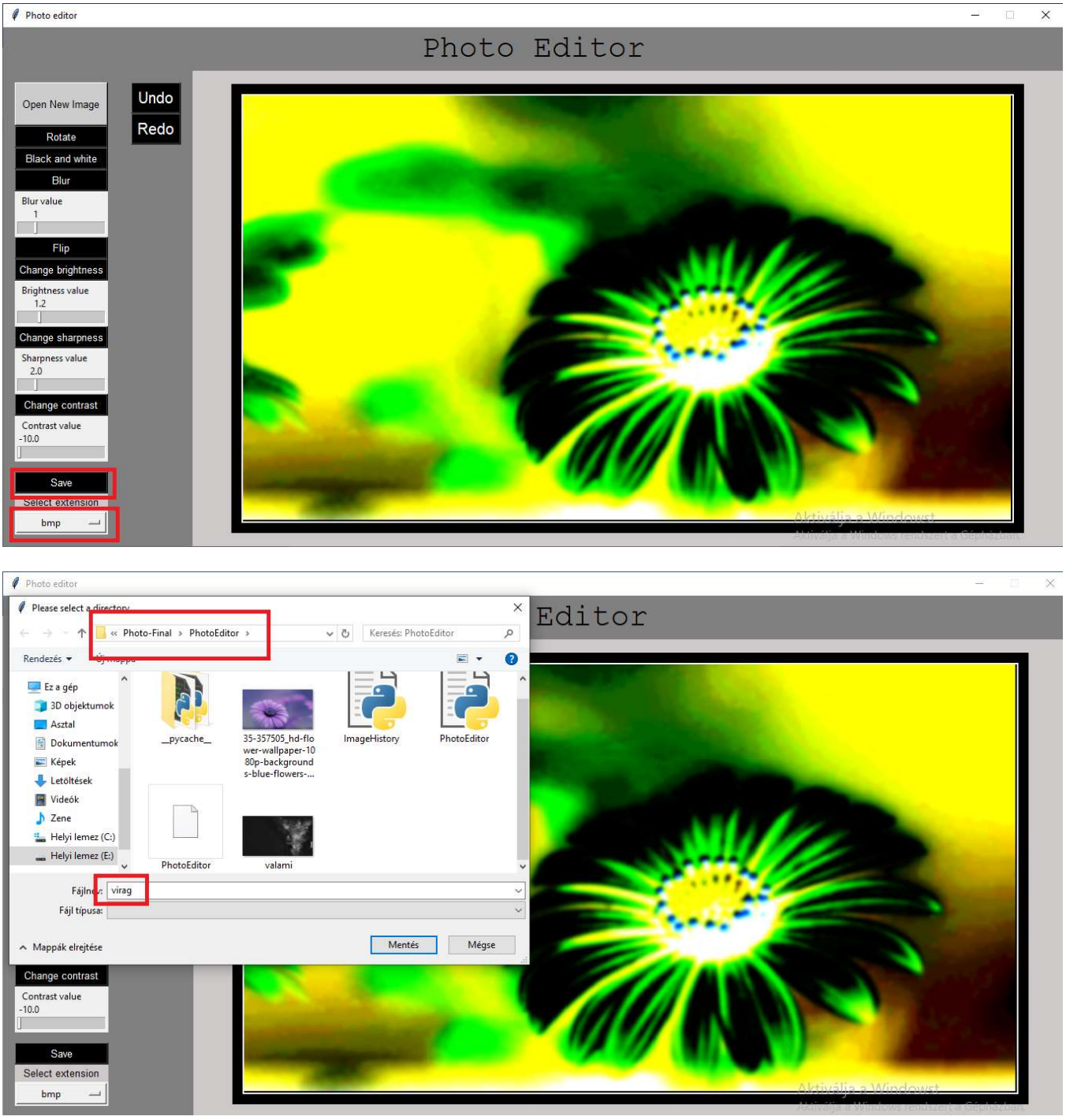
A betöltött kép szerkesztésére a felület bal szélén megtalálható eszközök állnak rendelkezésünkre. Egyes effektekhez értéket is beállíthatunk egy-egy csúszka segítségével.



0.3 ábra: kép homályosítása csúszka használatával

Kép mentése

Ha végeztünk a kép szerkesztésével, a bal alsó sarokban lévő Save gombbal menthetjük azt. A gomb alatt három fajta kiterjesztés között választhatunk: .jpg, .png, .bmp. A gombra rákattintva a felugró ablakon kiválaszthatjuk a fájl mentési helyét és a fájlnevet.



3.1 ábra: kép mentése