**Міністерство освіти та науки України Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра інформаційних систем та технологій**

Звіт з практикуму 3 на тему

«**Оцінка алгоритмів обробки растрових графічних зображень**»

**Перевірив**: **Виконав**:

cтудент групи

доц. Резніков С. А. ІК-91 Тяпухін Д. Г.

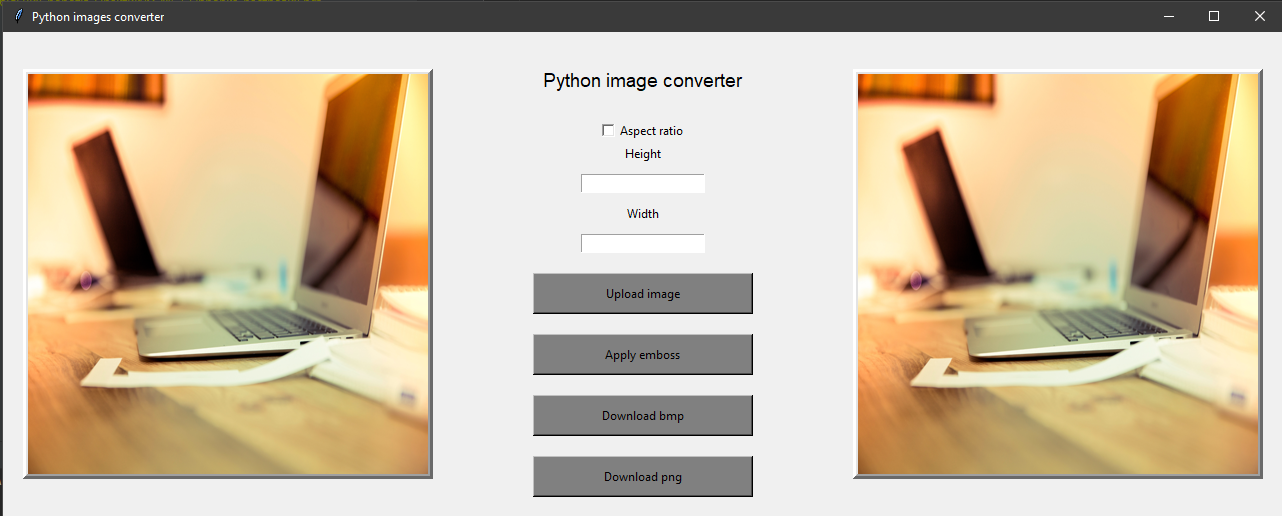
Київ — 2021

* **Тема**: Обробка растрових графічних зображень з допомогою мови Python та бібліотеки Python Imaging Library(PIL - PILLOW)
* **Мета**: Вивчення та здобуття практичних навичок використання бібліотеки PIL та конвертування форматів зображень у Python.

**Варіант 5**

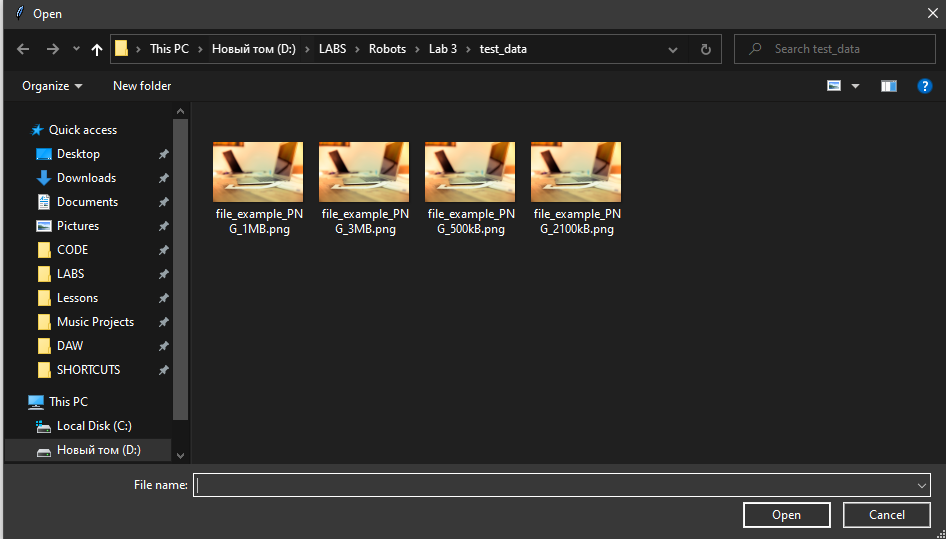
**Виконання роботи**

1. Розпочали роботу з інтерфейсу користувача (**View**) для того, щоб легше було визначити потрібний функціонал. Головне вікно програми виглядає наступним чином.



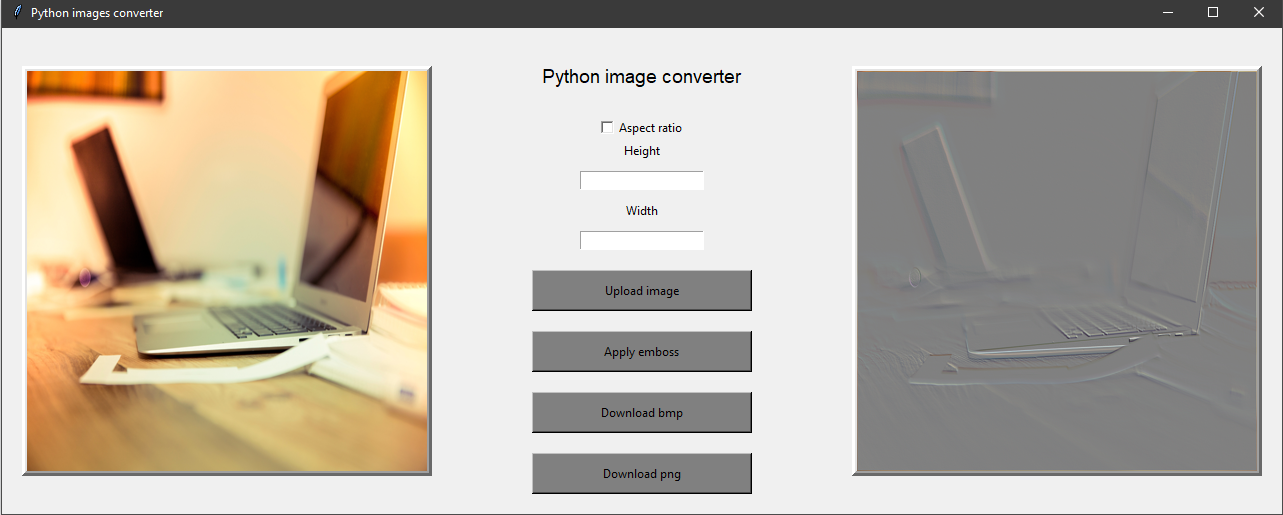
**Рис. 1 Інтерфейс програми**

1. Можна побачити чотири головні кнопки, 2 поля введення для користувача та один чекбокс.
2. Зліва розташоване початкове зображення, а справа те, як воно буде виглядати після всіх маніпуляцій.
3. Для того, щоб завантажити зображення в програму, користувач натискає на кнопку “**Upload image**”, відкривається наступне вікно:



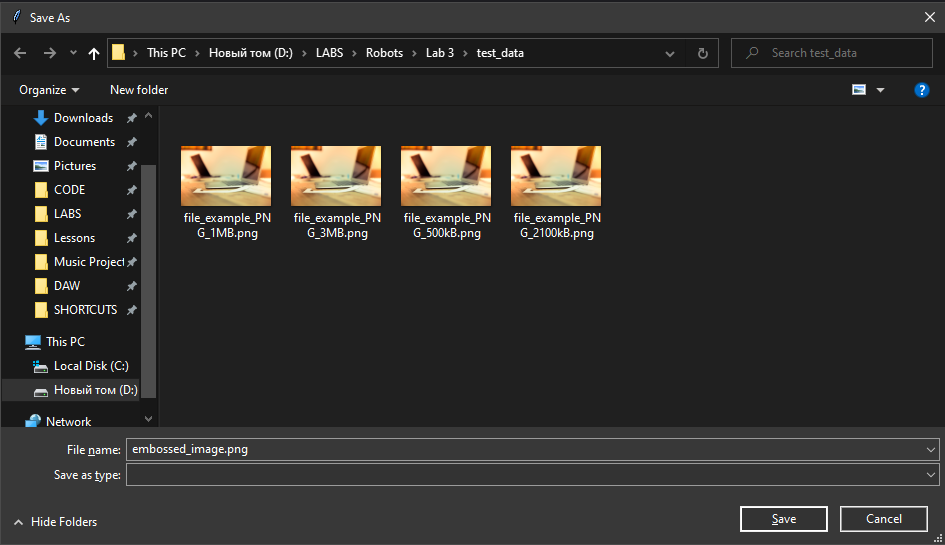
**Рис. 2 Вікно вибору потрібного зображення**

1. Заздалегідь було підготовлено декілька зображень різної роздільної здатності. Виберемо будь-яке з них, після цього воно відкриється в самій програмі (**рис. 1**)
2. Натиснемо на кнопку “**Apply emboss**” для того, щоб застосувати фільтр. Результат на рисунку 3.

****

**Рис. 3 Результат після застосування фільтра Emboss**

1. Якщо користувачу не потрібно змінювати роздільну здатність чи шастабування він вже може натиснути на “**Download png**” або “**Download bmp**”, щоб завантажити зображення у відповідному форматі. Після натискання відривається наступне вікно:



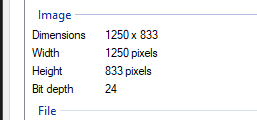
**Рис. 4 Вікно вибору шляху для нового зображення**



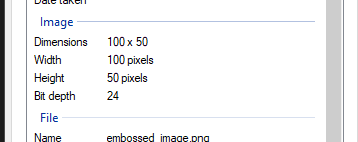
**Рис. 5 Зображення після застосування фільтра**

1. Для того щоб змінити роздільну здатність, достатньо внести потрібні значення у поля **Width**(Ширина) та **Height**(Висота)та після цього також завантажити нове зображення. Якщо користувач хоче змінити масштабування, він може залишити значення Ширини та Висоти, але

увімкнути опцію **Aspect ratio**,тоді зображення змінить своє масштабування (Рис. 8)



**Рис. 6 Початкові характеристики вхідного зображення**

****

**Рис. 7 Хар-ки вихідного зображення**



**Рис. 8 Вихідне зображення**

**Далі можна побачити вихідний код**

**main\_window.py**

**from tkinter import \***

**from ctypes import windll**

**from tkinter import filedialog**

**from model.picture\_manager import PictureManager**

**class MainWindow:**

**def \_\_init\_\_(self, picture\_manager: PictureManager):**

**window = Tk()**

**windll.shcore.SetProcessDpiAwareness(1)**

**window.title('Python images converter')**

**window.geometry('1280x720')**

**window.resizable(width=TRUE, height=TRUE)**

**self.\_window = window**

**self.\_picture\_manager = picture\_manager**

**self.\_emboss\_applied = False**

**# main frame located in window**

**self.\_program\_frame = Frame(window)**

**# frame with buttons and dropdowns**

**self.\_controls\_frame = Frame(self.\_program\_frame)**

**# main label**

**# packing means attaching to the frame**

**self.\_main\_label = Label(self.\_controls\_frame, font=('Arial', 14), text='Python image converter')**

**self.\_main\_label.pack(fill="both", expand=1)**

**# keep aspect ration flag**

**self.\_aspect\_ration\_value = IntVar()**

**self.\_aspect\_ratio = Checkbutton(self.\_controls\_frame, text="Aspect ratio", variable=self.\_aspect\_ration\_value)**

**self.\_aspect\_ratio.pack()**

**# user's height entry**

**self.\_height\_entry\_label = Label(self.\_controls\_frame, text="Height")**

**self.\_height\_entry\_label.pack(side=TOP)**

**self.\_height\_entry = Entry(self.\_controls\_frame)**

**self.\_height\_entry.pack(side=TOP, pady=10)**

**# user's width entry**

**self.\_width\_entry\_label = Label(self.\_controls\_frame, text="Width")**

**self.\_width\_entry\_label.pack(side=TOP)**

**self.\_width\_entry = Entry(self.\_controls\_frame)**

**self.\_width\_entry.pack(side=TOP, pady=10)**

**self.register\_download\_png\_button(self.\_controls\_frame)**

**self.register\_download\_bmp\_button(self.\_controls\_frame)**

**self.register\_emboss\_button(self.\_controls\_frame)**

**# modified picture**

**self.\_modified\_picture = Label(self.\_program\_frame, borderwidth=5, fg="gray", relief="raised")**

**self.\_modified\_picture.pack(side=RIGHT, padx=20)**

**# Create uploaded picture Image**

**self.\_uploaded\_picture = Label(self.\_program\_frame, borderwidth=5, fg="gray", relief="raised")**

**self.\_uploaded\_picture.pack(side=LEFT, padx=20)**

**# mount mane frame**

**self.\_program\_frame.pack(fill="both", expand=1, pady=10)**

**self.\_program\_frame.focus\_set()**

**self.register\_upload\_button(self.\_controls\_frame)**

**# mount controls frame**

**self.\_controls\_frame.pack(fill=Y, expand=1)**

**def open(self):**

**self.\_window.mainloop()**

**def upload\_button\_click\_handler(self):**

**self.\_file\_path = filedialog.askopenfilename()**

**if not getattr(self, '\_file\_path', None):**

**return**

**self.\_image = self.\_picture\_manager.open(self.\_file\_path)**

**resized\_image = self.\_picture\_manager.resize\_image(self.\_image)**

**photo\_image = self.\_picture\_manager.convert\_to\_photo\_image(resized\_image)**

**# original\_image = Image**

**self.\_uploaded\_picture.config(image=photo\_image)**

**self.\_uploaded\_picture.image = photo\_image**

**self.\_modified\_picture.config(image=photo\_image)**

**self.\_modified\_picture.image = photo\_image**

**self.\_emboss\_applied = False**

**def download\_base\_handler(self, image\_format='png'):**

**if not getattr(self, '\_file\_path', None):**

**return**

**save\_path = filedialog.asksaveasfilename()**

**if not save\_path:**

**return**

**height = self.\_height\_entry.get()**

**width = self.\_width\_entry.get()**

**if height and width:**

**base\_image = self.\_image**

**if self.\_aspect\_ration\_value.get():**

**self.\_picture\_manager.thumbnail\_image(base\_image, int(width), int(height))**

**else:**

**base\_image = self.\_picture\_manager.resize\_image(base\_image, int(width), int(height))**

**base\_image.save(save\_path, format=image\_format)**

**return**

**self.\_image.save(save\_path, format=image\_format)**

**def download\_button\_png\_handler(self):**

**self.download\_base\_handler('png')**

**def download\_button\_bmp\_handler(self):**

**self.download\_base\_handler('bmp')**

**def emboss\_button\_handler(self):**

**if not getattr(self, '\_file\_path', None):**

**return**

**self.\_image = self.\_picture\_manager.emboss\_image(self.\_image)**

**resized = self.\_picture\_manager.resize\_image(self.\_image)**

**pi = self.\_picture\_manager.convert\_to\_photo\_image(resized)**

**self.\_modified\_picture.config(image=pi)**

**self.\_modified\_picture.image = pi**

**self.\_emboss\_applied = True**

**def register\_download\_png\_button(self, frame):**

**self.\_download\_png\_button = Button(frame, text='Download png',**

**command=self.download\_button\_png\_handler, height=2, width=30, bg='gray')**

**self.\_download\_png\_button.pack(side=BOTTOM, pady=10)**

**def register\_emboss\_button(self, frame):**

**self.\_emboss\_button = Button(frame, text='Apply emboss',**

**command=self.emboss\_button\_handler, height=2, width=30, bg='gray')**

**self.\_emboss\_button.pack(side=BOTTOM, pady=10)**

**def register\_download\_bmp\_button(self, frame):**

**self.\_download\_bmp\_button = Button(frame, text='Download bmp',**

**command=self.download\_button\_bmp\_handler, height=2, width=30, bg='gray')**

**self.\_download\_bmp\_button.pack(side=BOTTOM, pady=10)**

**def register\_upload\_button(self, frame):**

**self.\_upload\_button = Button(frame, text='Upload image',**

**command=self.upload\_button\_click\_handler, height=2, width=30, bg='gray')**

**self.\_upload\_button.pack(side=BOTTOM, pady=10)**

**Picture\_manager.py**

**from PIL import ImageTk, Image, ImageFilter**

**class PictureManager:**

**def resize\_image(self, image: Image.Image, width=400, height=400) -> Image.Image:**

**return image.resize((width, height))**

**def open(self, file\_path):**

**return Image.open(file\_path)**

**def convert\_to\_photo\_image(self, image):**

**return ImageTk.PhotoImage(image)**

**def emboss\_image(self, image: Image.Image) -> Image.Image:**

**ImageFilter.EMBOSS.filterargs = ((3, 3), 1, 128, (-1, 0, 0, 0, 1, 0, 0, 0, 0))**

**return image.filter(ImageFilter.EMBOSS)**

**def thumbnail\_image(self, image: Image.Image, width=400, height=400) -> None:**

**image.thumbnail((width, height), Image.ANTIALIAS)**

**main.py**

**from view import main\_window**

**from model import picture\_manager**

**def main():**

**pic\_manager = picture\_manager.PictureManager()**

**w = main\_window.MainWindow(pic\_manager)**

**w.open()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**main()**