**Макойда Максим**

Для реалізації нашого проєкту і програмування зовнішнього вигляду програми скористаємося бібліотеко PyQT. Тому імпортуємо основні віджети які будуть використані в програмі: QApplication-Додаток, QWidget- Вікно додатка, QLabel – Напис, QPushButton-Кнопка, Вертикальна і горизонтальна направляюча лінія- QHBoxLayout, QVBoxLayout, QListWidget - Список з можливістю вибору елементів(клікабельний список)

Також оскільки робота програми передбачає вибір зображень з комп’ютера скористаємося новим віджетом бібліотеки PyQt5 - QFileDialog.

З його допомогою викликається вікно вибору папки (Провідник або Finder)

import os - Підключення модуля os, що **містить функції для роботи з операційною системою**

import os

from PyQt5.QtWidgets import (

   QApplication, QWidget,

   QFileDialog,

   QLabel, QPushButton, QListWidget,

   QHBoxLayout, QVBoxLayout

)

**Кучак Ярослав**

from PyQt5.QtCore import Qt # потрібна константа Qt.KeepAspectRatio для зміни розмірів із збереженням пропорцій

from PyQt5.QtGui import QPixmap # оптимізована для показу на екрані картинка

Клас **Image**:властивості і методи роботи з картинкою як з файлом, Модуль **ImageFilter**:

Набір констант для накладення фільтрів, повороту і відображення картинки

from PIL import Image

from PIL.ImageQt import ImageQt # Для перенесення графіки з Pillow до QT

from PIL import ImageFilter

from PIL.ImageFilter import (

   BLUR, CONTOUR, DETAIL, EDGE\_ENHANCE, EDGE\_ENHANCE\_MORE,

   EMBOSS, FIND\_EDGES, SMOOTH, SMOOTH\_MORE, SHARPEN,

   GaussianBlur, UnsharpMask

)

**Черчель Юрій**

Створимо інтерфейс програми: app = QApplication([])- **Конструктор**, створює об'єкт типу «Додаток»

win = QWidget()- **Конструктор**, створює об'єкт типу «Вікно». win.resize -розмір вікна програми, win.setWindowTitle -встановлює заголовок вікна, а саме 'Easy Editor'

lb\_image = QLabel("Картинка") -конструктор, створює об’єкт типу напис, в подальшому тут буде завантажена картинка

btn\_dir = QPushButton("Папка")- конструктор, створює об’єкт типу кнопка для вибору зображень з комп’ютера, QListWidget() - конструктор, створює об’єкт типу клікабельний список; наспупні елементи QPushButton -кнопки

app = QApplication([])

win = QWidget()

win.resize(700, 500)

win.setWindowTitle('Easy Editor')

lb\_image = QLabel("Картинка")

btn\_dir = QPushButton("Папка")

lw\_files = QListWidget()

btn\_left = QPushButton("Вліво")

btn\_right = QPushButton("Вправо")

btn\_flip = QPushButton("Відзеркалити")

btn\_sharp = QPushButton("Різкість")

btn\_bw = QPushButton("Ч/Б")

**Влад І Руслан Керчуни**

row = QHBoxLayout()          # Головна лінія

**Розташування віджетів по лінії**

row = QHBoxLayout()         головна горизонтальна лінія, що ділиться на 2 стовпці

# в першому - кнопка вибору каталогу і список файлів

# в другому – картинка і ряд кнопок

addWidget - Метод, додає віджет до лінії і розташовує по центру.

Команда setLayout -додати отриману лінію і її об'єкти у вікно програми

main\_win.show() - Зробити вікно видимим

app.exec()- Залишати додаток відкритим, поки не буде натиснута кнопка виходу

col1 = QVBoxLayout()         # ділиться на два стовпці

col2 = QVBoxLayout()

col1.addWidget(btn\_dir)      # в першому - кнопка вибору каталогу

col1.addWidget(lw\_files)     # і список файлов

col2.addWidget(lb\_image, 95) # в другому - картинка

row\_tools = QHBoxLayout()    # і ряд кнопок

row\_tools.addWidget(btn\_left)

row\_tools.addWidget(btn\_right)

row\_tools.addWidget(btn\_flip)

row\_tools.addWidget(btn\_sharp)

row\_tools.addWidget(btn\_bw)

col2.addLayout(row\_tools)

row.addLayout(col1, 20)

row.addLayout(col2, 80)

win.setLayout(row)

win.show()

**Тут треба запустити код на виконання і показати зовнішній вигляд програми**

**Ясніковський Святослав**

workdir = ''

Наступним ми запрограмовували кнопку **Папка**, при натисненні на яку зявлятиметься діалогове вікно і можна буде вибрати на комп’ютері папку із файлами та завантажити список імен файлів у ListWidget.

Створимо глобальну змінну workdir = ' ' . Її значення можна отримувати та змінювати з будь-якої частини програми, тут буде зберігатися шлях до вибраної папки з вікна QFileDialog

Виберемо файли лише з графічними розширеннями.

* Усі допустимі розширення зручно розмістити у списку **extensions**. Для цього створимо функцію def filter у якій необхідно Створити пустий список result для імен файлів.
* *Для кожного імені файлу зі списку filenames*:
  + *І кожного розширення зі списку extensions*:

**якщо** ім'я закінчується на це розширення,

**то** додати його до списку результатів.

* Повернути список result.

Метод **endswith(ext)** Поверне **True**, якщо ім'я файлу закінчується на ext і **False**, якщо ні.

def filter(files, extensions):

   result = []

   for filename in files:

       for ext in extensions:

           if filename.endswith(ext):

               result.append(filename)

   return result

**Марцинюк Михайло**

def chooseWorkdir():

   global workdir

   workdir = QFileDialog.getExistingDirectory()

**Запрограмуємо відображення списку імен у віджеті в функції** def showFilenamesList():

Вибираємо робочу папку (workdir).

Задаємо список допустимих розширень.

Завантажуємо імена файлів папки та залишаємо лише з розширеннями extensions.

Очищаємо віджет-список (у випадку, якщо там залишилися імена файлів іншої папки).

По одному додаємо у віджет відібрані імена файлів.

clicked.connect – обробка події натиснення на кнопку

def showFilenamesList():

   extensions = ['.jpg','.jpeg', '.png', '.gif', '.bmp']

   chooseWorkdir()

   filenames = filter(os.listdir(workdir), extensions)

   lw\_files.clear()

   for filename in filenames:

       lw\_files.addItem(filename)

btn\_dir.clicked.connect(showFilenamesList)

**Зауваження**:

Для роботи знадобиться не лише список файлів (є у віджеті), а й робоча папка.

Наразі ім'я папки видаляється після завершення роботи showFilenamesList()

Щоб зберегти папку, визначимо її як

***глобальну змінну***.

**Тут треба запустити код на виконання, натиснути на кнопку Папка, вибрати папку з графічними файлами та показати, що їх назви відображаються в списку**

**Булавков Олександр**

Наступне завдання запрограмувати попередній перегляд картинки, тобто щоб при натисненні на картинку у списку назв файлів, вона відображалася у вікні програми.

* Це завдання складніше ніж може здатися , тому що Зображення має адаптуватися до розміру вікна програми.
* При перемиканні між картинками прев'ю має змінюватися.

Під час обробки має з'являтися прев'ю обробленої копії.

* Для цього запрограмуємо клас ImageProcessor() в конструкторі якого запишемо поточне **зображення** (за замовчуванням None);
* поточне **ім'я файлу** (за замовчуванням None);

**ім'я підпапки** для збереження змінених картинок;

class ImageProcessor():

   def \_\_init\_\_(self):

       self.image = None

       self.dir = None

       self.filename = None

       self.save\_dir = "Modified/"

**Метод loadImage() — завантаження зображення**

**Збережемо filename** у властивості filename екземпляра класу ImageProcessor

Зі шляху до робочої папки та імені файлу **сформуємо шлях до картинки**

**Відкриємо картинку** (об'єкт Image), звернувшись за повним шляхом

fullname = os.path.join(workdir, filename) - Отримання повного шляху до файлу об'єднанням шляху до папки та імені файлу

   def loadImage(self, filename):

       ''' під час завантаження запам'ятовуємо шлях та ім'я файлу '''

       self.filename = filename

       fullname = os.path.join(workdir, filename)

       self.image = Image.open(fullname)

**Строкош Євген**

**Метод showImage() — показати зображення**

Приховаємо віджет на час «технічних робіт».

Повним шляхом до файлу створюємо об'єкт QPixmap спеціально для відображення графіки в інтерфейсі.

Повним шляхом до файлу створюємо об'єкт QPixmap спеціально для відображення графіки в інтерфейсі.

Адаптуємо картинку під розміри поля.

**Відображаємо віджет!**

def showImage(self, path):

       lb\_image.hide()

       pixmapimage = QPixmap(path)

       w, h = lb\_image.width(), lb\_image.height()

       pixmapimage = pixmapimage.scaled(w, h, Qt.KeepAspectRatio)

       lb\_image.setPixmap(pixmapimage)

       lb\_image.show()

Щоб застосувати написаний функціонал у програмі, створимо функцію-обробник **showChosenImage** натискання по елементу списку імен картинок.

Якщо вибрано елемент списку lw\_files:

**filename** = вибраний рядок віджету-списку

**Завантажуємо** картинку методом loadImage()

**Формуємо** шлях для відображення картинки

**Відображаємо** картинку методом showImage()

Далі команда **lw\_files.currentRowChanged.connect(showChosenImage) підключення функції до віджета QListWidget**

def showChosenImage():

    if lw\_files.currentRow() >= 0:

       filename = lw\_files.currentItem().text()

       workimage.loadImage(filename)

       workimage.showImage(os.path.join(workdir, workimage.filename))

**Тут треба клацнути по картинках і показати що вони відображаються у вікні програми**

**Сверида Тимур**

   def saveImage(self):

**Метод saveImage() — зберегти картинку**

**Формуємо шлях** до папки для збереження

Якщо цей шлях **не існує**?

(Папки для збереження ще немає?) За сформованим шляхом **створити нову папку**

**Формуємо шлях до** картинки, що зберігається (з ім'ям файлу картинки!)

**Зберігаємо картинку** вбудованим методом класу Image

       ''' зберігає копію файлу у підпапці '''

       path = os.path.join(workdir, self.save\_dir)

       if not(os.path.exists(path) or os.path.isdir(path)):

           os.mkdir(path)

       fullname = os.path.join(path, self.filename)

       self.image.save(fullname)

**Кучак Ярослав**

**Переходимо до довгоочікуваної**

**обробки зображень!**

**Тут розповісти про імпорт бібліотеки Pillow, код на початку**

**Функція опрацювання події натиснення на кнопку ч/б**

**-** convert("L")- перетворює зображення вбудованим методом у відтінки сірого (чорно-біле)

- **Зберегти** змінений об'єкт Image як файл методом save()

- **Формуємо** шлях для відображення зміненої картинки

**- Відображаємо** змінену картинку методом showImage()

+ сказати за btn\_bw.clicked.connect(workimage.do\_bw)

   def do\_bw(self):

       self.image = self.image.convert("L")

       self.saveImage()

       image\_path = os.path.join(workdir, self.save\_dir, self.filename)

       self.showImage(image\_path)

**Левицький Дмитро**

**Функції опрацювання події натиснення на кнопки вліво-вправо**

**Обробляємо** поточну картинку вбудованим методом (ROTATE\_90) – праворуч; (ROTATE\_270) – лівоворуч

**Зберігаємо** змінений об'єкт Image як файл методом *saveImage*()

**Формуємо** шлях для відображення зміненої картинки

**Відобразимо** змінену картинку

методом *showImage*()

+ сказати за

btn\_left.clicked.connect(workimage.do\_left)

btn\_right.clicked.connect(workimage.do\_right)

   def do\_left(self):

       self.image = self.image.transpose(Image.ROTATE\_90)

       self.saveImage()

       image\_path = os.path.join(workdir, self.save\_dir, self.filename)

       self.showImage(image\_path)

   def do\_right(self):

       self.image = self.image.transpose(Image.ROTATE\_270)

       self.saveImage()

       image\_path = os.path.join(workdir, self.save\_dir, self.filename)

       self.showImage(image\_path)

**Корнієнко Олександр**

**Функції опрацювання події натиснення на кнопки дзеркало і різкість**

**Обробляємо** поточну картинку вбудованим методом

transpose(Image.FLIP\_LEFT\_RIGHT)-дзеркальневідображення

filter(SHARPEN)-для різкості

**Зберігаємо** змінений об'єкт Image як файл методом *saveImage*()

**Формуємо** шлях для відображення зміненої картинки

**Відобразимо** змінену картинку

методом *showImage*()

+ сказати за

btn\_sharp.clicked.connect(workimage.do\_sharpen)

btn\_flip.clicked.connect(workimage.do\_flip)

   def do\_flip(self):

       self.image = self.image.transpose(Image.FLIP\_LEFT\_RIGHT)

       self.saveImage()

       image\_path = os.path.join(workdir, self.save\_dir, self.filename)

       self.showImage(image\_path)

   def do\_sharpen(self):

       self.image = self.image.filter(SHARPEN)

       self.saveImage()

       image\_path = os.path.join(workdir, self.save\_dir, self.filename)

       self.showImage(image\_path)

workimage = ImageProcessor() #поточне робоче зображення для роботи

lw\_files.currentRowChanged.connect(showChosenImage)

btn\_bw.clicked.connect(workimage.do\_bw)

btn\_left.clicked.connect(workimage.do\_left)

btn\_right.clicked.connect(workimage.do\_right)

btn\_sharp.clicked.connect(workimage.do\_sharpen)

btn\_flip.clicked.connect(workimage.do\_flip)

app.exec()