

Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення цифрової обробки зображень»

Студента 4 курсу, 7 групи,
спеціальності 121 «Інженерія
програмного забезпечення»

Силенка Олександра
Олеговича

підпис студента

Науковий керівник
кандидат технічних наук,
доцент

Рзаєва Світлана
Леонідівна

підпис керівника

Гарант освітньої програми
кандидат технічних наук,
доцент

Цензура Микола
Олександрович

підпис керівника

КИЇВ – 2020

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1 СТРУКТУРА КОМП'ЮТЕРНОГО ДОДАТКУ ДЛЯ РОБОТИ З ЗОБРАЖЕННЯМИ ТА ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	5
1.1 Загальні відомості про додатки для роботи з зображеннями	5
1.2. Опис та характеристика бібліотек для обробки зображень на мові Python	6
1.3. Технічне завдання	8
1.4 Висновки до розділу 1	11
РОЗДІЛ 2	12
АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ГРАФІЧНОГО РЕДАКТОРА НА МОВІ PYTHON	12
2.1. Стандарти розробки та життєвого циклу програмного продукту.	12
2.2 Фреймворк для створення користувацького інтерфейсу	18
2.3 Інструментарій розробки системи	22
2.4 Висновки до розділу 2.....	25
РОЗДІЛ 3	26
РОЗРОБКА ГРАФІЧНОГО РЕДАКТОРА НА PYTHON	26
3.1 Процес розробки додатку на мові Python.....	26
3.2 Висновки до розділу 3	35
ВИСНОВКИ ТА ПРОПОЗИЦІЇ.....	36
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	38
ДОДАТКИ	39

					КНТЕУ 121 07-15.БР			
Зм.	Аркуш	№ докум	Підпис	Дата	Розробка програмного забезпечення цифрової обробки зображень	Стадія	Аркуш	Аркушів
Зав. кафедри	Криворучко О.В.					3	2	38
Керівник	Рзаєва С.Л.					Факультет інформаційних технологій, 4 курс, 7 група		
Гарант	Цензура М.О.							
Розроб.	Силенко О.О.				Зміст			

ВСТУП

Актуальність. Сьогодні зображення та відео є скрізь. Інтернет-сайти для обміну фотографіями та соціальні мережі мають їх у мільярдах штук. Пошукові системи будуть знаходити зображення майже на будь-який можливий запит. Практично всі телефони та комп'ютери оснащені вбудованими камерами. Не рідкість що у багатьох людей на своїх пристроях мають багато гігабайтів фотографій та відео.

Багато галузей техніки, що мають відношення до отримання, обробки, зберігання та передачі інформації, в значній мірі орієнтуються в даний час на розвиток систем, в яких інформація має характер зображень. Зображення, яке можна розглядати як двовимірний сигнал, є значно більш ємкісним носієм інформації, ніж звичайний одновимірний (часовий) сигнал. Разом з тим, рішення наукових та інженерних задач при роботі з візуальними даними вимагає особливих зусиль, що спираються на знання специфічних методів, оскільки традиційна ідеологія одновимірних сигналів і систем мало придатна в цих випадках. В особливій мірі це проявляється при створенні нових типів інформаційних систем, які вирішують такі проблеми, які до сих пір в науці і техніці не вирішувалися, і які вирішуються зараз завдяки використанню інформації візуального характеру.

Вже багато років стрімко розвивається ринок пристроїв із графічним інтерфейсом. У розробці комп'ютерної графіки, наприклад, при розташуванні об'єктів, потрібно пропонувати досить великі і складні розробки для аналізу та періодизації інформації за обраною обставиною. На сьогоднішній день існує безліч різних методів, отриманих на обробку зображень.

На даний час розробка програм з графічним інтерфейсом є однією з найприбутковіших галузей, а також однією з найактуальніших. Бо усі хочуть

					КНТЕУ 121 07-15.БР			
Зм.	Аркуш	№ докум	Підпис	Дата	Алгоритми та методи цифрової обробки зображень	Стадія	Аркуш	Аркушів
Зав. кафедри	Криворучко О.В.					В	3	38
Керівник	Рзаєва С.Л.					Факультет інформаційних технологій, 4 курс, 7 група		
Гарант	Цензура М.О.							
Розроб.	Силенко О.О.							
					Вступ			

бачити яскравий інтерфейс на своєму екрані або моніторі. На меті данного дослідження буде створення програми з графчним інтерфейсом у мінімалістичному стилі, що буде здатна до редагування фото. А також дати визначення що собою являє графічний інтерфейс та бібліотеки з якими надалі планується робота.

Метою дослідження випускної кваліфікаційної роботи є аналіз функціональних характеристик цифрової обробки зображень та розробка програмного забезпечення цифрової обробки зображень мовою Python.

Об'єкт дослідження – цифрова обробка зображень.

Предмет дослідження – розробка програмного забезпечення цифрової обробки зображень.

Задачи дослідження

- зробити аналіз відомостей про цифрову обробку зображень;
- описати можливості та характеристики цифрової обробки зображень мовою Python;
- розробити технічне завдання;
- проаналізувати Стандарти розробки та життєвого циклу програмного продукту;
- здійснити опис функціональних характеристик Інструментарій розробки програмного забезпечення цифрової обробки зображень;
- створити програмну розробку графічного редактора цифрової обробки зображень мовою Python.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		4

РОЗДІЛ 1

СТРУКТУРА КОМП'ЮТЕРНОГО ДОДАТКУ ДЛЯ РОБОТИ З ЗОБРАЖЕННЯМИ ТА ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні відомості про додатки для роботи з зображеннями

Як правило, комп'ютерну графіку ділять на растрову і векторну, але іноді виділяють ще фрактальну графіком. Крім того, за типом уявлення говорять про двовірної і тривимірної графіки.

Растрова графіка. Зображення представляється як матриця пікселів, де кожного пікселя задано певне значення кольору, яскравості, прозорості. Мінус представлення зображень в растровому вигляді полягає в тому, що при збільшенні картинка втрачає свою красу через появу гігантських квадратів, які раніше були пікселями. Цього не станеться лише тоді, коли зображення має дуже великий дозвіл (більша кількість пікселів).

Векторна графіка представляє зображення як набір геометричних фігур (точки, прямі, прямокутники, кола, криві). Для кожної фігури задані товщина ліній, колір, координати. На відміну від растрової графіки, зображення векторної графіки масштабуються, повертаються без втрати якості.

У тривимірній графіці об'єкти представляється в тривимірному просторі. Широко використовується в кіно, тривимірних комп'ютерних мультфільмах, комп'ютерних іграх.

У реальності, будь-яке зображення, яке користувач убачає на екрані монітору. Монітор або екран у баченні користувача плоский, але на ньому видно об'ємне зображення.

					КНТЕУ 121 07-15.БР			
Зм.	Аркуш	№ докум	Підпис	Дата	Розробка програмного забезпечення цифрової обробки зображень	Стадія	Аркуш	Аркушів
Зав. кафедри	Криворучко С.Р.					Р1	5	38
Керівник	Рзаєва С.Л.							
Гарант	Цензура М.О.							
Розроб.	Силенко О.О.				Структура комп'ютерного додатку для роботи з зображеннями та технічне завдання на розробку програмного	Факультет інформаційних технологій, 4 курс, 7 група		

Все це завдяки тому що монітор це матриця пікселів. На моніторі зображено лише проекцію тривимірної фігури, але завдяки просторовій уяві людина може розрізнити трьох вимірні фігури на екрані.

1.2. Опис та характеристика бібліотек для обробки зображень на мові Python

1. Scikit-image - це пакет Python з відкритим кодом, який працює з масивами NumPy. Він реалізує алгоритми та програми для використання в дослідницьких, освітніх та промислових додатках. Ця постійна проста і зрозуміла бібліотека навіть для новичків в екосистемі Python. Данна бібліотека містить високоякісний та рецензований код, написаний активними добровольцями.

2. NumPy - це одна з основних Python-бібліотек з підтримкою масивів. Ілюстрація представляє собою стандартний масив NumPy, що містить пікселі точних даних. Таким чином, у виконанні основних NumPy-операцій ми можемо змінити піксельні значки зображення. Саме зображення можна завантажувати через skimage і показувати за допомогою Matplotlib.

3. SciPy - це також важливий науковий модуль в Python, як і NumPy. Він підходить для вирішення основних завдань з обробки та проробляє роботу із зображеннями. У приватності, в підмодулі scipy.ndimage доступні функції, які працюють у n-мерних масивах NumPy. Даний пакет включає в себе функції для лінійної та нелінійної фільтрації, бінарної морфології, інтерполяції B-спланованих та вимірювальних об'єктів.

4. PIL (Python Imaging Library) - це безкоштовна Python-бібліотека для відкриття, роботи і збереження різних форматів зображень.

На жаль, її розробка остаточно зупинилася, а останнє оновлення вийшло в 2009. На щастя, є Pillow - активно розвивається форк PIL з простою установкою. Він працює на всіх основних операційних системах і підтримує Python 3.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		6

Бібліотека містить базовий функціонал для обробки зображень, включаючи точкові операції, фільтри з набором вбудованих ядер згортки і перетворення кольорового простору.

5. OpenCV (Open Source Computer Vision Library) - одна з найпопулярніших бібліотек для додатків з комп'ютерного бачення. OpenCV-Python - це Python-версія інтерфейсу для OpenCV. Наявність коду на C / C ++ в бекенд гарантує швидкість бібліотеки, а Python-обгортка у фронтенді забезпечує простоту настройки і розгортання. Завдяки цьому OpenCV-Python є відмінним рішенням для високонавантажених обчислювальних програм з комп'ютерного зору.

6. SimpleCV - це ще один фреймворк з відкритим кодом для створення додатків з комп'ютерного зору. З ним у вас з'являється доступ до кількох потужним бібліотекам комп'ютерного зору (наприклад, OpenCV) без необхідності вивчення глибини кольору, файлових форматів, кольорних просторів і т.д. Крива навчання куди менше, ніж в OpenCV, і, як то кажуть в їх слогані, «комп'ютерний зір стає простіше».

7. Mahotas також є Python-бібліотекою для комп'ютерного зору і обробки зображень. Вона містить стандартні функції по обробці зображень (фільтри і морфологічні операції), а також сучасні можливості комп'ютерного зору для обчислення ознак (виявлення особливих точок і локальні дескриптори). Швидкість розробки забезпечується Python-інтерфейсом, а плюсом для швидкості служать алгоритми на C ++. Mahotas - це швидка бібліотека з мінімалістичним кодом і залежностями.

8. TK або Insight Segmentation and Registration Toolkit - це крос-платформна система з відкритим кодом, що надає розширений набір інструментів для аналізу зображень. Так само як і SimpleITK - спрощений шар, «надбудований» поверх ITK. Даний шар полегшує роботу з бібліотекою при швидкому прототіпірованні, навчанні і інтерпретованих мовах.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		7

SimpleITK - це набір інструментів для аналізу зображень з великою кількістю компонентів, що підтримують загальну фільтрацію, сегментацію і реєстрацію зображень. Сам SimpleITK написаний на C ++, але доступний для багатьох мов програмування, включаючи Python.

9. PyCairo являє собою набір прив'язок Python-коду для графічної бібліотеки Cairo. Cairo - це 2D-бібліотека для відтворення векторної графіки. Векторна графіка цікава тим, що не втрачає своєї чіткості при зміні розмірів або трансформації. PyCairo - це набір прив'язок для Cairo, за допомогою яких можна викликати Cairo-команди з Python.

1.3. Технічне завдання

1. Загальні відомості

1.1. Найменування додатку – «I Am Editor».

1.1.1. Повне найменування додатку – «I Am Editor»

1.1.2. Скорочене найменування додатку – «I Am Editor»

1.2. Планові терміни початку та закінчення робіт

1.2.1. Початок робіт – 01.03.2020

1.2.2. Закінчення робіт – 01.05.2020

1.3. Порядок та пред'явлення результатів робіт – пред'явити готовий додаток керівнику

1.4. Потенційні користувачі системи – користувачі віком від 13 років, які бажають отримати досвід роботи з растровими графічними редакторами.

2. Мета та призначення створення додатку

2.1. Призначення додатку – надання можливості роботи з зображеннями

2.2. Мета створення додатку – створити додаток для надання широкому загалу користувачів досвіду роботи з графікою

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		8

3. Вимоги до додатку

3.1. Вимоги до додатку в цілому – у додатку має бути реалізована можливість редагувати зображення на їх розсуд та бути зрозумілим тим людям користувачам, хто вперше користується додатком.

3.1.1. Вимоги до структури та функціонування додатку, перелік підсистем – додаток базується на мові програмування Python виконаного у Pycharm з використанням бібліотеки PILLOW.

3.1.1.1. Вимоги до режимів функціонування додатку – додаток повинен функціонувати у режимі оффлайн.

3.1.1.2. Вимоги до діагностування додатку – додаток повинен працювати безперебійно та успішно виконувати усі процеси.

3.1.1.3. Вимоги до режимів управління додатком – повинно бути реалізовано режим користувача.

3.1.2. Показники призначення

3.1.2.1. Параметри, що характеризують ступінь відповідності системи призначенням – безперебійність роботи системи, відсутність помилок.

3.1.2.2. Вимоги до пристосованості додатку до змін . Буде створено меню налаштувань, яке буде змінювати роботу всієї системи. Вимоги до збереження працездатності додатку в різних ймовірних умовах – додаток буде питати про збереження даних перед виходом, а також має можливість налаштування збереження по таймеру. Налаштування після виходу з додатку зберігаються, при зверненні додатку він стає на «паузу» а не вимикається.

3.1.2.3. Вимоги до надійності

3.1.2.4. Вимоги до методів оцінки і контролю показників надійності на різних стадіях створення додатку – на стадії тестування дозволена поява помилок, на стадії здачі додатку не дозволена поява помилок.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		9

3.1.3. Вимоги до ергономіки та технічної естетики – зі знарядь праці на одне робоче місце необхідний комп'ютер.

3.1.4. Вимоги до захисту інформації від несанкціонованого доступу – доступ до інформації надається тільки користувачу додатку.

3.1.5. Вимоги до захисту від впливу зовнішніх факторів – додаток приймає зміни тільки безпосередньо від операційної системи (встановлення системної теми).

3.1.6. Вимоги безпеки – додаток повинен бути безпечним для його користувачів.

3.2. Вимоги до видів забезпечення:

3.2.1. Вимоги до інформаційного забезпечення – інформаційне забезпечення повинне бути цілісним і захищеним від несанкціонованого доступу.

3.2.1.1. Вимоги до складу, структури і способів організації даних в додатку – усі дані повинні зберігатись за допомогою баз даних і управлятись СУБД. Вимоги до інформаційного обміну між компонентами системи - передача інформації між компонентами системи має виконуватись стандартними протоколами на рівні програмного забезпечення або на рівні платформи (системи керування БД).

3.2.1.2. Вимоги щодо застосування систем управління базами даних – потрібно створити адаптери для кожної таблиці бази даних, з яких будуть вноситись дані.

3.2.1.3. Вимоги до структури процесу збору, обробки, передачі даних в системі представлення даних – процес збору, обробки та передачі даних в системі повинен бути безпечним та швидким.

3.2.1.4. Вимоги до захисту даних від руйнувань при аваріях і збоях в електроживленні системи – аварійний вихід з додатку зі збереженням даних в пам'яті комп'ютера.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		10

4. За допомогою додатку користувачі повинні мати змогу вибрати зображення та його редагувати. Також користувачі повинні мати змогу використовувати бокове меню й усі засоби, що у ньому запропоновані. Користувач зможе в меню налаштувань налагоджувати систему під себе, змінювати тему а також розмір шрифту у всьому додатку, та скидати налаштування до початкових. Програмне забезпечення повинно базуватись на використанні:

4.1.Pycharm 2020.1.

4.2.PyQT5.

4.3.Python.

4.4.SQL.

4.5.Для технічного забезпечення рекомендовано використовувати пристрій з такими характеристиками: Linux Ubuntu.

4.6.Intel celeron g4930.

4.7. AMD Radeon R7 M340.

4.8.1 гігабайт оперативної пам'яті.

4.9. 2 гігабайти відопам'яті.

5. Вимоги до методичного забезпечення – не передбачаються, адже використання системи буде інтелектуально простим та зрозумілим.

1.4 Висновки до розділу 1

У першому розділі було проаналізовані основні види графіки, а також бібліотеки для роботи з ними на мові Python. У доповнення було створене та узагальнене технічне завдання.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		11

РОЗДІЛ 2

АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ГРАФІЧНОГО РЕДАКТОРА НА МОВІ PYTHON

2.1. Стандарти розробки та життєвого циклу програмного продукту.

Під моделлю життєвого циклу ПО розуміється структура, що визначає послідовність виконання і взаємозв'язку процесів, дій і завдань протягом ЖЦ. Модель ЖЦ залежить від специфіки, масштабу і складності проекту і специфіки умов, в яких система створюється і функціонує.

Стандарт ISO / IEC 12207 не пропонує конкретні моделі життєвого циклу і методи розробки ПП. Положення стандарту є загальними для будь-яких моделей життєвого циклу, методів і технологій розробки ПП.

Стандарт описує структуру процесів життєвого циклу ПП, але не уточнює, як виконати дії і завдання, включені в ці процеси.

Модель життєвого циклу будь-якого конкретного ПП визначає характер процесу його створення, який являє собою сукупність упорядкованих у часі, взаємопов'язаних і об'єднаних в етапи робіт, виконання яких необхідне і досить для створення ПП, відповідного заданим вимогам.

Найбільшого поширення набули такі моделі життєвого циклу розробки ПП: Каскадна модель, або «водоспад» (Waterfall model). В однорідних інформаційних системах 1970-х і 1980-х років прикладні програмні продукти представляли собою єдине ціле. Для розробки такого типу програмних продуктів застосовувалася каскадна модель, або «водоспад» (waterfall)

Каскадна модель. Принципова особливість каскадного підходу: перехід на наступний етап здійснюється тільки після того, як буде повністю завершена робота на поточному етапі, і повернень на пройдені етапи не передбачається. Кожен етап

					КНТЕУ 121 07-15.БР			
Зм.	Аркуш	№ докум	Підпис	Дата	Розробка програмного забезпечення цифрової обробки зображень	Стадія	Аркуш	Аркушів
Зав. кафедри	Криворучко С.Р.					P2	12	38
Керівник	Рзаєва С.Л.							
Гарант	Цензура М.О.							
Розроб.	Силенко О.О.				Аналіз програмного забезпечення графічного редактора на мові Python	Факультет інформаційних технологій, 4 курс, 7 група		

закінчується отриманням деяких результатів, які служать вихідними даними для наступного етапу.

Каскадний підхід добре зарекомендував себе при побудові інформаційних систем, для яких на самому початку розробки можна досить точно і повно сформулювати всі вимоги з метою надати розробникам свободу реалізувати їх технічно якнайкраще. У цю категорію потрапляють складні системи з великим числом завдань обчислювального характеру, системи реального часу та ін.

У той же час даний підхід має ряд істотних недоліків, обумовлених перш за все тим, що реальний процес розробки ПП ніколи повністю не вкладається в таку жорстку схему. Цей процес носить, як правило, ітераційний характер: результати чергового етапу часто викликають зміни в проектних рішеннях, вироблених на більш ранніх стадіях. Таким чином, постійно виникає потреба в поверненні до попередніх етапів і уточнення або перегляд раніше прийнятих рішень. В результаті реальний процес розробки набирає вигляду моделі, званої моделлю з проміжним контролем.

V-образна модель (V-shaped model). Ця модель була розроблена як різновид каскадної моделі, в якій особлива увага приділяється верифікації та атестації ПП. Модель показує, що тестування продукту обговорюється, проектується і планується, починаючи з ранніх етапів життєвого циклу розробки (на рис. Цей процес позначений штриховими стрілками).

Від каскадної моделі V-образна модель успадкувала послідовну структуру, відповідно до якої кожна послідовна фаза починається тільки після успішного завершення попередньої фази.

Дана модель заснована на систематичному підході до проблеми, для вирішення якої визначено чотири базові кроки: аналіз, проектування, розробка та огляд. При виконанні аналізу здійснюються планування проекту та складання вимог. Проектування поділяється на високорівневе і детальне (низькорівневе).

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		13

Розробка включає в себе кодування, а огляд - різні види тестування. Дану модель доцільно використовувати при розробці програмних продуктів, головною вимогою для яких є висока надійність.

Модель прототипування (Prototype model). Модель прототипування дозволяє створити прототип програмного продукту до або протягом етапу складання вимог до програмного продукту.

Потенційні користувачі працюють з цим прототипом, визначаючи його сильні і слабкі сторони, про результати повідомляють розробникам програмного продукту. Таким чином, забезпечується зворотний зв'язок між користувачами і розробниками, яка використовується для зміни або коригування специфікації вимог до ПП. В результаті такої роботи продукт буде відображати реальні потреби користувачів.

Життєвий цикл розробки ПП починається з розробки плану проекту потім виконується швидкий аналіз, після чого створюються база даних (якщо, звичайно, вона використовується в ПП), призначений для користувача інтерфейс і виконується розробка необхідних функцій. В результаті цієї роботи виходить документ, що містить часткову специфікацію вимог до додатку. Даний документ надалі є основою для ітераційного циклу швидкого прототипування.

В результаті прототипування розробник демонструє користувачам готовий прототип, а користувачі оцінюють його функціонування. Після цього визначаються проблеми, над усуненням яких спільно працюють користувачі і розробники. Цей процес триває до тих пір, поки користувачі не будуть задоволені ступенем відповідності додатку, поставленим перед ним вимогам. Потім прототип демонструють користувачам з метою отримання пропозицій щодо його удосконалення, які включаються в послідовні ітерації до тих пір, поки робоча модель не виявиться задовільною. Після цього отримують від користувачів офіційне схвалення (затвердження) функціональних можливостей прототипу і виконують його остаточне перетворення в готовий додаток.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		14

Модель прототипування рекомендується застосовувати в наступних випадках:

- Вимоги до ПП заздалегідь невідомі;
- Вимоги не постійні або невдало сформульовані;
- Вимоги необхідно уточнити;
- потрібно перевірка концепції;
- Існує потреба в інтерфейсі;
- виконуємо нова, яка не має аналогів розробка;
- розробник не впевнені в тому, яке рішення слід вибрати

Модель швидкої розробки додатків, або RAD-модель (RAD - Rapid Application Development model). У RAD-моделі кінцевий користувач грає вирішальну роль. У тісній взаємодії з розробниками він бере участь у формуванні вимог і апробації їх на працюючих прототипах. Таким чином, на початку життєвого циклу на кінцевого користувача випадає велика частина роботи, але в результаті цього створювана система формується більш швидко.

У традиційному життєвому циклі розробки більшу частину роботи складають програмування і тестування. При автоматизації програмування і повторне використання коду, що застосовуються в RAD-моделі, більшу частину роботи складають планування і проектування.

Модель включає в себе наступний фази:

- Складання вимог і планування - здійснюються з використанням так званого методу спільного планування вимог (планування робіт зі створення ПП і складання вимог до ПП виконуються одночасно), який полягає в структурному аналізі і обговоренні вирішуваних завдань;
- Опис користувача - проектування ПП, яке виконує при безпосередній участі замовника;
- створення - детальне проектування, кодування і тестування ПП, а також постачання його замовнику;

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		15

-супровід - приймальні випробування, установка ПП і навчання користувачів.

Розглянуту RAD-модель можна застосовувати при розробці ПП, які добре піддаються моделюванню, коли вимоги до ПП добре відомі, а замовник може взяти безпосередню участь в процесі розробки.

Багатопохідна модель (Incremental model). Багатопохідний модель - це кілька ітерацій процесу побудови прототипу ПП з додаванням на кожній наступній ітерації нових функціональних можливостей або підвищенням ефективності додатку.

Передбачається, що на ранніх етапах життєвого циклу розробки (планування, аналіз вимог і розробка проекту) виконується конструювання ПП в цілому. Тоді ж визначається і число необхідних інкрементів та належних до них функцій. Кожен інкремент потім проходить через остаточні фази життєвого циклу (кодування і тестування). Спочатку виконуються конструювання, тестування і реалізація базових функцій, що становлять основу ПП. Наступні ітерації спрямовані на поліпшення функціональних можливостей ПП.

Багатопохідна модель може бути застосована, якщо більшість вимог до ПП будуть сформульовані заздалегідь, а для виконання проекту буде виділено великий період часу.

Спіральна модель (Spiral model). Для подолання проблем, пов'язаних з використанням багатопохідної моделі, в середині 1980-х років була запропонована спіральна модель життєвого циклу.

Її принципова особливість полягає в тому, що прикладної додатку створюється не відразу, як в разі каскадного підходу, а по частинах з використанням методу прототипування. Під прототипом розуміється чинний програмний компонент, який реалізує окремі функції і зовнішні інтерфейси розробляється ПП. Створення прототипів здійснюється за декілька ітерацій, або витків спіралі. Кожна ітерація відповідає створенню фрагмента, або версії ПП, на ній уточнюються цілі і характеристики проекту, оцінюється якість отриманих

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		16

результатів та плануються роботи наступної ітерації. На кожній ітерації проводиться ретельна оцінка ризику перевищення термінів і вартості проекту з метою визначення необхідності виконання ще однієї ітерації, ступеня повноти і точності розуміння вимог до системи, а також доцільності припинення проекту.

Спіральна модель позбавляє користувачів і розробників ПП від повного і точного формулювання вимог до системи на початковій стадії, оскільки вони уточнюються на кожній ітерації. Таким чином, поглиблюються і послідовно конкретизуються деталі проекту і в результаті вибирається обґрунтований варіант, який доводиться до реалізації.

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи, дозволяючи переходити на наступну стадію, не чекаючи повного завершення роботи на поточній стадії, оскільки при ітеративному способі розробки відсутню роботу можна виконати на наступній ітерації. Головне завдання такої розробки - якомога швидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог.

В якості моделі життєвого циклу розробки програмного продукту велике поширення отримала поліпшена спіральна модель. На відміну від раніше розглянутої спіральної моделі ця модель використовує каскадний підхід на завершальних етапах розробки ПП.

Використання спіральної моделі доцільно, якщо існує хоча б одна з наступних причин:

- доцільність створення прототипу;
- організація володіє навичками, необхідними для адаптації моделі;
- вимагайте виконувати проекти з середнім і високим ступенями ризику;
- замовниками не впевнені в своїх потребах;
- Вимоги занадто складні;
- проект дуже великий.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		17

2.2 Фреймворк для створення користувацького інтерфейсу

Python ([paɪθən], - скриптована мова програмування, створена Гвідо ван Россум (нині співробітник Dropbox). Названий так, за словами самого Гвідо, в честь шоу Monty Python. Найбільш знаменита відмінна риса мови - використання відступів для виділення блоків коду і керуючих структур.

Щоб його вивчити, досить ознайомитися з туторіали і погортати код паритрійки опенсорсних програм. Мова вкрай проста і скромна на виразні засоби порівняно с Ruby або Perl (один з елементів ідеології - треба програми писати, а не самовиражатися вигадливо). Але насправді надані можливості досить широкі, щоб надовго захопитися цією мовою. При достатньому бажанні і відсутності перманентного заперечення є ймовірність навчитися писати придатний код що можна супроводжувати та продавати. В силу свого нульового порогу входу, вкрай популярний серед шкіл та людей що хочуть показати що вони вміють програмувати.

Основною причиною вищезгаданого заперечення зазвичай буває «магія відступів»: на відміну від багатьох мов, в яких можна, але не обов'язково робити відступи, в Пітоні написати код в рядок неможливо в принципі - відступи прямо впливають на вкладеність виразів.

Прихильники вважають, що це зручніше, ніж фігурні дужки (не кажучи вже про begin ... end), а заодно позитивно впливає на читабельність коду, дисциплінує програмістів та допомагає саморозвитку як людині.

Його не дуже любляють люди що все життя програмували мовою Джава, бо лише обмежено підтримує деякі просунуті технології програмування:

Анонімні функції (лямбда) можуть складатися тільки з одного виразу;

«Reduce ()» (аналог foldl з Haskell) прибраний в засіки стандартної бібліотеки і не рекомендується до застосування через «погану читабельність коду»;

Відсутня Pattern Matching: все-таки, це імперативна мова;Гвідо ван Россум вважає непотрібною підтримку хвостовій рекурсії, бо вже є цикли;

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		18

Неможливо потоками паралелі програму по декільком процесорам / ядер: замість потоків треба використовувати модуль multiprocessing або інші інтерпретатори пітона (наприклад, PyPy-STM).

Глобальний стан: навіть можна міняти поля в об'єктах зовні (пітон спочатку не діє);

Відступи згідно стандарту PEP8 повинні бути в 4 пробілу – а Python, навіть третій, приймає будь-яке число.

Python 3 це нова версія того ж мови від того ж автора. виправляє деякі недоліки попередньої версії, наприклад: будь-які спроби конвертувати байти в юнікод і назад без вказівки кодування викликать виняток, та й взагалі робота з юнікодом стала набагато зручніше. Не дуже зручний GIL (Global Interpreter Lock) залишився, але у версії 3.2 його трішечки покращили, тому тепер він працює нормально, хоч він все ще у пріоритеті вибирає тяжкі процеси що виконуються півроку процес натомість хорошим легким процесам. З'явилася можливість вказувати підказки для типів аргументів функцій, а з версії 3.6 - для звичайних змінних і атрибутів класу.

Поліпшена бібліотека колекцій. Строкові шаблони використовують зручний синтаксис в як у C # замість старого C-подібного синтаксису. Доданий новий шар I / O, що дозволяє працювати з файлами так само, як в Java і C #. Був доданий новий фреймворк asyncio, так що тепер можна писати асинхронний код, що дозволяє не використовувати послуги таких гігантів, як Twisted або Tornado.

На даний час усі патчі робляться для трійки, і лише третина на двійку (можна спокійно наштовхнутися на баг 2007 року, який на 3 виправлений, а на 2 - ні). Однак не зрозуміло чому, що хоча Python3 відрізали від другої версії в 2008 році, багато хто продовжує писати на гілці 2. А суть тут в порушенні зворотної сумісності між 2. і 3. версіями, особливо з-за рядків. Під 3 треба писати купу нових бібліотек, або переробляти старі. А під 2 все вже і так є, і в порівнянні з цією майже рабською працею по перенесенню всіх напрацювань спільноти, плюси від зміни декількох

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		19

строк коду старої версії - неочевидний. Насправді, перехід між версіями не так уже й складний, більшість старих бібліотек вже успішно переписали під 3-ку, а багато нові тільки під неї і виходять. Останній плюс на користь третьої версії є офіційна заява розробників про завершення підтримки другої в 2020 році. Так що нові проекти на другій версії ніхто не пише, а старі на трійку не перекладають хіба що ліниві, та ще хлопці, у яких все зав'язано на Twisted або якого-небудь схожого монстра, що не має повної сумісності з Python3.

Крім того, будучи простим в освоєнні інтерпретатором, Python став скриптовою мовою в багатьох бізнес-додатках, потіснивши BASIC та багато інших скриптових мов. Python зараз майже усяди в CVS, і в Blender, і в Techlog, а також у Open / LibreOffice, і навіть у додатках Microsoft. Однак, не у всіх розробників вистачає грошей і сміливості поміняти в своєму додатку милий серцю 2.7 на Python3.

Qt дозволяє змінювати та розробляти написане за допомогою його допомогою ПЗ в більшості сучасних операційних систем шляхом простої компіляції програми для кожної системи без зміни вихідного коду.

Включає в себе усе необхідне для написання програмного забезпечення включаючи класи та починаючи від елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних і XML. Є повністю об'єктно-орієнтованим, розширюваним і підтримує техніку компонентного програмування.

Відмітна особливість - використання мета-об'єктного компілятора - попередньої системи обробки вихідного коду. Розширення можливостей забезпечується системою плагінів, які можливо розміщувати безпосередньо в панелі візуального редактора. Також існує можливість розширення звичної функціональності віджетів, пов'язаної з розміщенням їх на екрані, відображенням, перемальовуванням при зміні розмірів вікна.

Його складовим є візуальне середовище розробки графічного інтерфейсу Qt Designer, що дозволяє створювати діалоги і форми в режимі WYSIWYG. У

					КНТЕУ 121 07-15.БР	Аркуш
						20
Зм.	Аркуш	№ докум	Підпис	Дата		

постачання Qt є Qt Linguist - графічна утиліта, що дозволяє спростити локалізацію і переклад програми на багато мов; і Qt Assistant - довідкова система Qt, що спрощує роботу з документацією по бібліотеці, а також дозволяє створювати кроссплаформену довідку для розроблювального на основі Qt програмного забезпечення. Починаючи з версії 4.5.0 в комплект включена середовище розробки Qt Creator, яка включає редактор коду, довідку, графічні засоби Qt Designer і можливість налагодження додатків. Qt Creator може використовувати GCC або Microsoft VC ++ як компілятора і GDB як відладчика. Для Windows-версій бібліотека комплектується компілятором, заголовними і об'єктними файлами MinGW.

Існують версії бібліотеки для Microsoft Windows, систем класу UNIX з графічною підсистемою X11, Android, iOS, Mac OS X, Microsoft Windows CE, QNX [10], вбудованих Linux-систем і платформи S60. Йде портирование на Windows Phone і Windows RT.

Також йде портирование на Haiku і Tizen. Деякий час бібліотека також поширювалася ще в версії Qt / Embedded, призначеної для застосування на вбудованих і мобільних пристроях, але починаючи з середини 2000-х років вона виділена в самостійний продукт Qtopia.

Приклади використання Qt5 цьому програмному продукті.

@classmethod

def newImage(cls, w, h, bg, context):

image = QtGui.QImage(w, h, QtGui.QImage.Format_ARGB32_Premultiplied)

image.fill(bg)

return cls("", image, bg, context)

Даний код слугує для виклику нового зображення у робоче середовище Програмного продукту.

def save(self):

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		21

```
self.image.save(self.fileName)

self.modified = False
```

А цей приклад слугує для збереження цього файлу.

2.3 Інструментарій розробки системи

PyCharm - це найінтелектуальніша Python IDE з повним набором засобів для ефективної розробки на мові Python. Випускається в двох варіантах - безкоштовна версія PyCharm Community Edition і підтримує більший набір можливостей PyCharm Professional Edition. PyCharm виконує інспекцію коду на льоту, автодоповнення, в тому числі ґрунтуючись на інформації, отриманої під час виконання коду, навігацію по коду, забезпечує безліч рефакторингов.

Редактор коду PyCharm надає першокласну підтримку для Python, JavaScript, CoffeeScript, TypeScript, CSS, популярних мовних і багато чого іншого.

Оцініть переваги можливості підтримки декількох мов, виявлення помилок і їх виправлення!

Розумна Навігація. Скористайтеся розумним пошуком для переходу до будь-якого класу, файлу, символу, або навіть події IDE і вікна інструментів. Всього один клік, і ви переходите до супер-методу, тестування, використання, реалізації, декларації і це як мінімум.

Швидкий і Безпечний Рефакторинг. З PyCharm вам відкривається розумне оновлення коду з безпечним видаленням і перейменуванням, методом вилучення, вступної змінної, вбудованої змінної або методом, і іншими видами рефакторінгу. Орієнтовані на фреймворк і мову програмування рефакторинг допоможуть виконати будь-яка зміна в рамках цілого проекту.

Вбудовані Інструменти Розробки. Велика колекція інструментів PyCharm включає в себе інтегрований відладчик і запуск тестування• профайлер Python;

- вбудований термінал;
- інтеграцію з великими VCS і вбудованими інструментами баз даних;

2					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		22

- можливість віддаленої розробки з віддаленими інтерпретаторами;
- інтегрований термінал ssh;
- інтеграція з Docker і Vagrant.

Налагодження, Тестування і Профілювання

Використовуйте потужний відладчик з графічним інтерфейсом під Python і JavaScript. Створюйте і проводите тестування з підтримкою коду і драйвером текстів на основі графічного інтерфейсу. Отримайте повний контроль над своїм кодом завдяки інтеграції Python Profiler.

VCS, Розгортання і Дистанційна Розробка. Збережіть свій час завдяки уніфікованому призначеному для користувача інтерфейсу для роботи з Git, SVN, Mercurial і іншими системами контролю версій. Запускайте і проводьте налагодження свого застосування в віддаленому режимі.

Вам доступна проста конфігурація автоматичного розгортання для віддаленого хоста або VM і управління вашої інфраструктурою за допомогою Vagrant і Docker.

Інструменти Бази Даних

- Access Oracle
- SQL Server
- PostgreSQL
- MySQL

І інші бази даних прямо з IDE. Ви можете розраховувати на допомогу PyCharm при редагуванні коду SQL, виконання запитів, перегляді даних і внесення змін до схеми. Веб розробка. У доповненні до Python, PyCharm надає першокласну підтримку різних фреймворків веб розробки від Python, окремих мов, JavaScript, CoffeeScript, TypeScript, HTML / CSS, AngularJS, Node.js та інших.

Веб Фреймворки Python. PyCharm надає відмінну підтримку окремих фреймворків для сучасних фреймворков веб розробки, таких як Django, Flask,

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		23

Google App Engine, Pyramid, і web2py, включаючи відладчик шаблонів Django, інструменти manage.py і appcfg.py, автозаповнення і навігацію.

JavaScript і HTML. PyCharm надає першокласну підтримку для JavaScript, CoffeeScript, TypeScript, HTML і CSS, а також їх сучасних наступників. Отладчик JavaScript також включений в PyCharm і є інтегрованим з конфігурацією запуску сервера Django.

Live Edit. Live Editing Preview дозволяє вам відкривати сторінку в редакторі і браузері для пошуку і миттєвого знаходження недавно внесених змін в браузері. PyCharm зберігає ваші зміни в автоматичному режимі, в той час як браузер сам оновлює сторінку, показуючи результати змін. Наукові інструменти. PyCharm інтегрується з IPython Notebook, має інтерактивну консоль Python і

підтримує як Anaconda, так і безліч інших пакетів, включаючи Matplotlib і NumPy.

Інтеграція IPython Notebook. PyCharm інтегрується з IPython Notebook і забезпечує рішення, яке комбінує можливості IPython Notebook з додатковими перевагами, які може запропонувати велика частина інтелектуального середовища розробки Python, включаючи автоматичне завершення, навігацію, перевірку помилок, і так далі.

Інтерактивна Консоль Python. Ще одна причина скачати PyCharm полягає в тому, що ви можете запустити консоль Python REPL в PyCharm, що дає масу переваг: миттєва перевірка синтаксису з додатковими перевітками, зіставлення дужок і лапок, і, звичайно, завершенням коду.

Підтримка Наукового Стека. PyCharm надає спеціальну підтримку наукових бібліотек. Він підтримує Anaconda, Numpy, Matplotlib і інші наукові бібліотеки, надаючи користувачу глибоке розуміння коду, інтерактивні графіки, перегляд масивів і багато іншого. Налаштовується і Крос-платформна IDE. Ви можете завантажити PyCharm і встановити на Windows, Mac OS і Linux використовуючи один ліцензійний ключ. Насолоджуйтесь відмінно налаштованим робочим

2					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		24

простором з налаштованими кольоровими схемами та гарячими клавішами, з доступною емуляцією VIM.

Інтерфейс що підлаштовується під користувача. Чи існує хоча б один розробник, який не любить налаштовувати свої інструменти? Тому до послуг користувача - зручне налаштування UI від PyCharm. Насолоджуйтесь зручним робочим простором з налаштованим колірними схемами і гарячими клавішами.

Модулі. Більше 10 років розробки платформи IntelliJ дає PyCharm більше 50 плагінів на різний смак і колір, включаючи підтримку додаткових VCS, інтеграції з різними інструментами і фреймворками, редактором оновлень, таким як емуляція Vim.Крос-платформне середовище розробки. Ви можете завантажити PyCharm і встановити його на Windows, Mac OS або Linux.

Крім цього, ви можете встановити його на таку кількість комп'ютерів, яке самі побажаєте, користуючись тією ж середовищем і функціоналом на будь-якому комп'ютері.

2.4 Висновки до розділу 2

У заключенні розділу 2 неможливо не відмітити корисність розробки та зручність IDE PyCharm. Також важливо зазначити перелічені моделі розробки програмного забезпечення що також являється добрим нагадуванням яким складним та різноплановим може бути програмування та його компоненти.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		25

РОЗДІЛ 3

РОЗРОБКА ГРАФІЧНОГО РЕДАКТОРА НА PYCHARM

Інструмент Pycharm надає початківцям розробникам можливість негайно приступити до роботи, але перед цим необхідно все ретельно продумати і обговорити всі нюанси. Наявність такого гнучкого інструменту як Pycharm не вирішує всі проблеми і тому необхідно чітко уявляти, що саме повинно вийти в результаті.

3.1 Процес розробки додатку на мові Python

Встановити PyCharm на робочий комп'ютер доволі просто. Для даного проекту досить безкоштовної версії. Після натискання на кнопку «безкоштовне завантаження» відбувається скачування установника PyCharm. Цей інсталятор важить менше 1 Мб і всі необхідні файли скачує після початку установки.

Для початку створимо GUI для зручнішого та зрозумілішого користування.

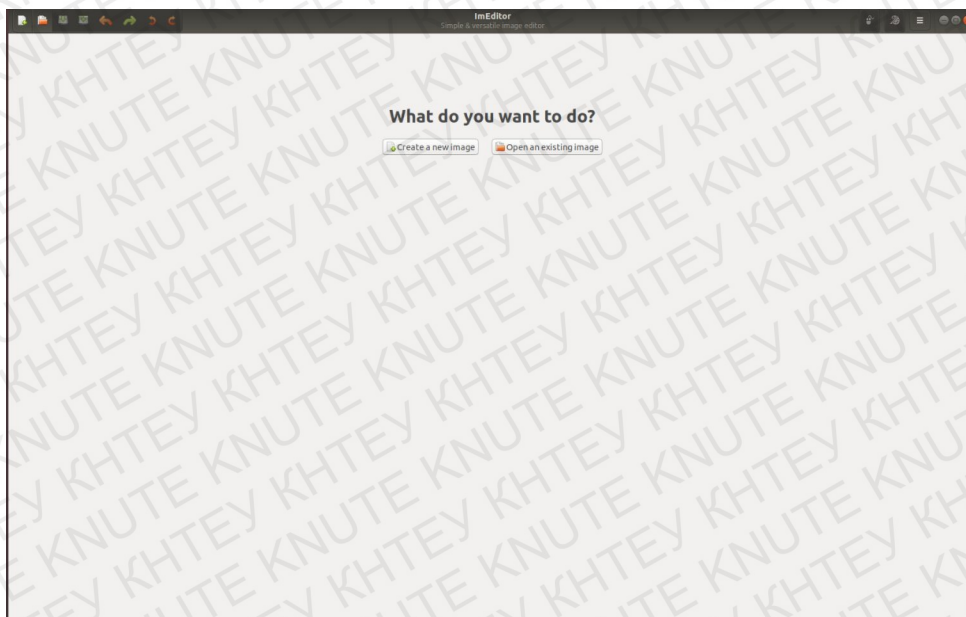


Рис. 3.1. Основне меню для роботи із зображенням

					КНТЕУ 121 07-15.БР			
Зм.	Аркуш	№ докум	Підпис	Дата	Розробка програмного забезпечення цифрової обробки зображень	Стадія	Аркуш	Аркушів
Зав. кафедри	Криворучко О.В.					РЗ	26	38
Керівник	Рзаєва С.Л.					Факультет інформаційних технологій, 4 курс, 7 група		
Гарант	Цензура М.О.							
Розроб.	Силенко О.О.							
					Розробка графічного редактора на PyCharm			

```
super(NewFileDialog, self).__init__(Parent)
self.context = context
self.parent = Parent

dimensionGroup = QtWidgets.QGroupBox(self.context.getText("dialog_new_image", "dimension"))
dimensionLayout = QtWidgets.QVBoxLayout()

self.width = QtWidgets.QSpinBox(dimensionGroup)
self.width.setMinimum(1)
self.width.setMaximum(4096)
self.width.setValue(self.context.getIntDefault("new", "width", 32))
self.height = QtWidgets.QSpinBox(dimensionGroup)
self.height.setMinimum(1)
self.height.setMaximum(4096)
self.height.setValue(self.context.getIntDefault("new", "height", 32))

clipboard = QtWidgets.QApplication.clipboard()
print("Accesses clipboard")
im = clipboard.image()
if not im.isNull():
    print("Image not null")
    self.width.setValue(im.width())
    self.height.setValue(im.height())

dimensionLayout.addWidget(self.width)
dimensionLayout.addWidget(self.height)
dimensionGroup.setLayout(dimensionLayout)

backgroundGroup = QtWidgets.QGroupBox(self.context.getText("dialog_new_image", "background"))
backgroundLayout = QtWidgets.QVBoxLayout()
self.r1 = QtWidgets.QRadioButton(self.context.getText("dialog_new_image", "transparent"))
self.r1.setChecked(True)
self.r2 = QtWidgets.QRadioButton(self.context.getText("dialog_new_image", "color"))
self.cButton = QtWidgets.QPushButton()
self.cButton.clicked.connect(self.getColor)
self.cButton.setSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)
self.color = QtGui.QColor(255, 255, 255)
self.cButton.setStyleSheet("background-color: " + self.color.name() + ";")
self.cButton.setText(self.color.name())
colorLayout = QtWidgets.QHBoxLayout()
```

Рис. 3.2. Розробка основного меню

					Аркуш
					КНТЕУ 121 07-15.БР
Зм.	Аркуш	№ докум	Підпис	Дата	27

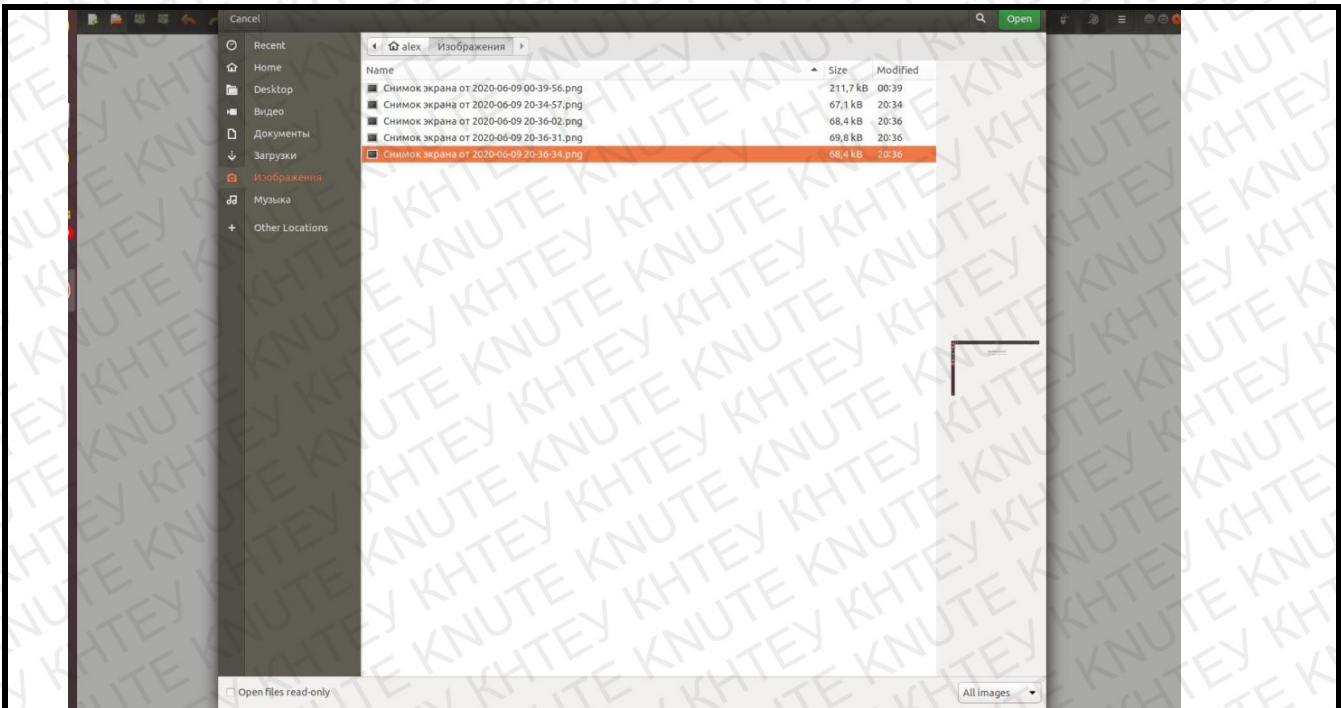


Рис. 3.3. Меню вибору зображення

```
class Image:
    def __init__(self, fileName, image, bg, context):
        self.fileName = fileName
        self.image = image
        self.bgColor = bg
        self.context = context

        self.zoom = 1
        self.selection = None
        self.history = [QtGui.QImage(self.image)]
        self.posHistory = 0
        self.modified = False

    @classmethod
    def fromFile(cls, fileName, context):
        image = QtGui.QImage(fileName).convertToFormat(QtGui.QImage.Format_ARGB32_Premultiplied)

        if image.hasAlphaChannel():
            bgColor = QtGui.QColor(0, 0, 0, 0)
        else:
            bgColor = QtGui.QColor(255, 255, 255)

        return cls(fileName, image, bgColor, context)

    @classmethod
    def newImage(cls, w, h, bg, context):
        image = QtGui.QImage(w, h, QtGui.QImage.Format_ARGB32_Premultiplied)
        image.fill(bg)

        return cls("", image, bg, context)

    def save(self):
        self.image.save(self.fileName)
        self.modified = False
```

Рис. 3.2. Код для вибору та збереження зображення

Також у додатку присутні кнопки Undo та Redo Їх реалізація приведена на зображенні нижче.

					КНТЕУ 121 07-15.БР		Аркуш
							28
Зм.	Аркуш	№ докум	Підпис	Дата			

```

self.image.save(self.fileName)
self.modified = False

def addHistoryStep(self):
    if self.posHistory != len(self.history)-1:
        self.history = self.history[:self.posHistory+1]
        self.history.append(QtGui.QImage(self.image))

    self.posHistory += 1
    self.modified = True

def paintPoint(self, x, y, color, size):
    painter = QtGui.QPainter(self.image)
    painter.setPen(color)
    painter.setCompositionMode(QtGui.QPainter.CompositionMode_Source)
    painter.drawPixmap(QCore.QPoint(x-size+1, y-size+1), self.context.brushes[size-1])

def createMaskFromArea(self, point, threshold=0):
    mask = self.recursiveFill(QtGui.QImage(self.image), point, self.image.pixel(point))
    return mask

def recursiveFill(self, mask, point, color, threshold=0):
    x, y = point.x(), point.y()

    if x < 0 or y < 0 or x > mask.width()-1 or y > mask.height()-1:
        return

    elif mask.pixel(x,y) == color:
        mask.setPixel(x, y, QtCore.Qt.color0)
        self.recursiveFill(_mask, QtCore.QPoint(x+1, y), color)
        self.recursiveFill(_mask, QtCore.QPoint(x, y+1), color)
        self.recursiveFill(_mask, QtCore.QPoint(x-1, y), color)
        self.recursiveFill(_mask, QtCore.QPoint(x, y-1), color)

```

Рис. 3.5. Реалізація кнопок головного меню

					КНТЕУ 121 07-15.БР		Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата			29

Також реалізовано можливість зміни розміру форми та кольору кисті для малювання що зображено на зображеннях нижче.

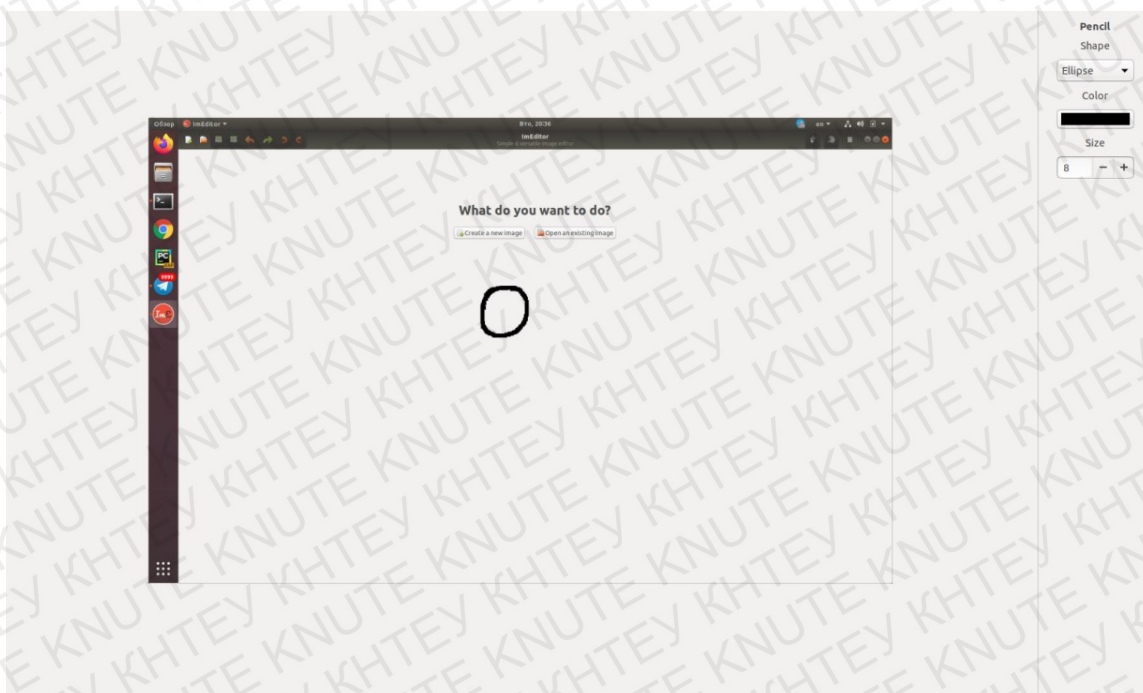


Рис. 3.6. Приклад зображення круглого пензля



Рис. 3.7. Приклад зображення квадратного пензля – зображено кисть

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		30

```
def createBrushes():

    circles = []
    brushes = []

    for i in range(9):
        m, im = createCircle(i)
        circles.append(m)
        pm = QtGui.QBitmap().fromImage(im)
        brushes.append(pm)

    return circles, brushes
```

Рис. 3.8. Клас для створення пензлей

```
createCircle(radius):
    im = QtGui.QImage(QCore.QSize(radius*2+1, radius*2+1), QtGui.QImage.Format_Mono)
    im.fill(QCore.Qt.color0)
    x0 = radius
    y0 = radius
    f = 1 - radius
    ddf_x = 1
    ddf_y = -2 * radius
    x = 0
    y = radius
    l = []
    for i in range(2*radius+1):
        l.append([False]*(2*radius+1))
    l[y0+radius][x0] = True
    im.setPixel(x0, y0+radius, QtCore.Qt.color0)
    l[y0-radius][x0] = True
    im.setPixel(x0, y0-radius, QtCore.Qt.color0)
    l[y0][x0+radius] = True
    im.setPixel(x0+radius, y0, QtCore.Qt.color0)
    l[y0][x0-radius] = True
    im.setPixel(x0-radius, y0, QtCore.Qt.color0)
    while x < y:
        if f >= 0:
            x += 1
            ddf_x += 2
            f += ddf_x
        for i in range(x0-x, x0+x+1):
            l[y0+y][i] = True
            im.setPixel(i, y0+y, QtCore.Qt.color0)
        for i in range(x0-x, x0+x+1):
            l[y0-y][i] = True
            im.setPixel(i, y0-y, QtCore.Qt.color0)
        for i in range(x0-y, x0+y+1):
            l[y0+x][i] = True
            im.setPixel(i, y0+x, QtCore.Qt.color0)
        for i in range(x0-y, x0+y+1):
            l[y0-x][i] = True
            im.setPixel(i, y0-x, QtCore.Qt.color0)
        for i in range(x0-radius, x0+radius+1):
            l[y0][i] = True
            im.setPixel(i, y0, QtCore.Qt.color0)
    return l, im
```

Рис. 3.9. Зображено код який застосовується для зміни розміру пензлей

					КНТЕУ 121 07-15.БР	Аркуш
						31
Зм.	Аркуш	№ докум	Підпис	Дата		

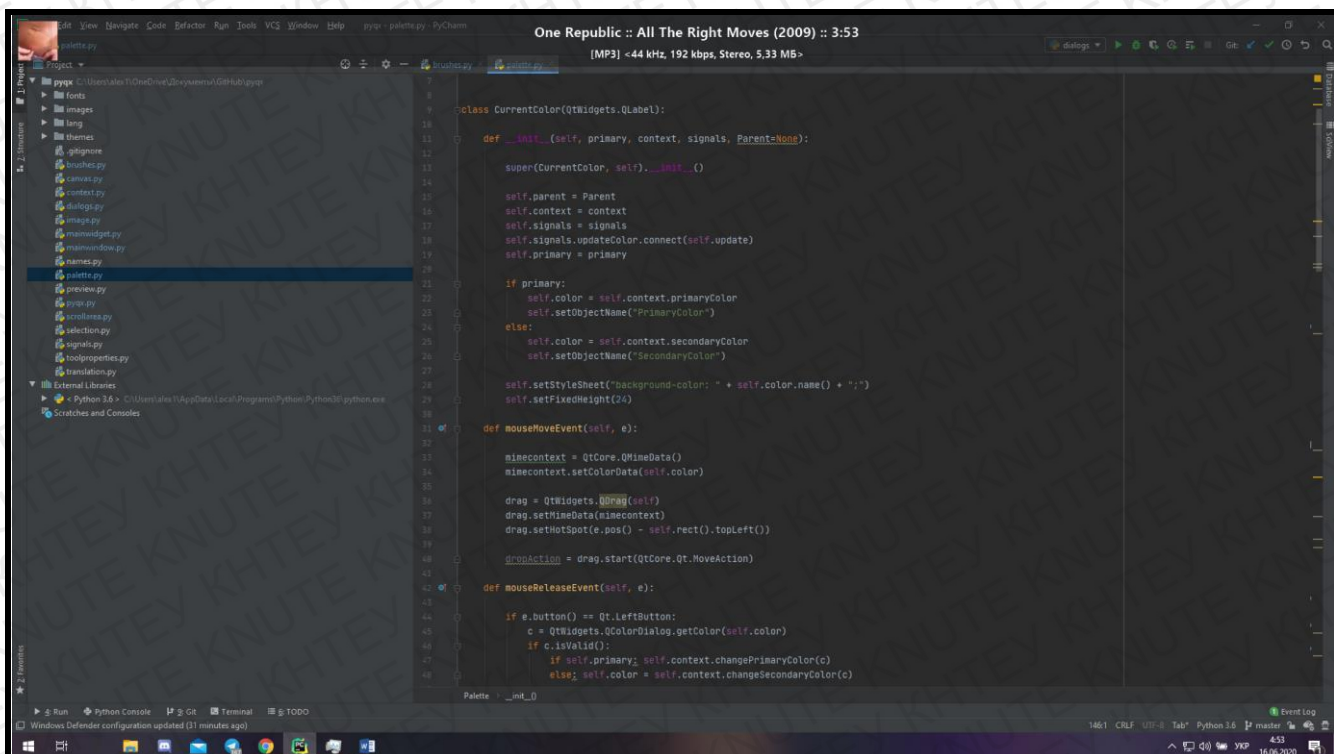


Рис. 3.10. Клас який визначає яким кольором зараз буде малювати пензель

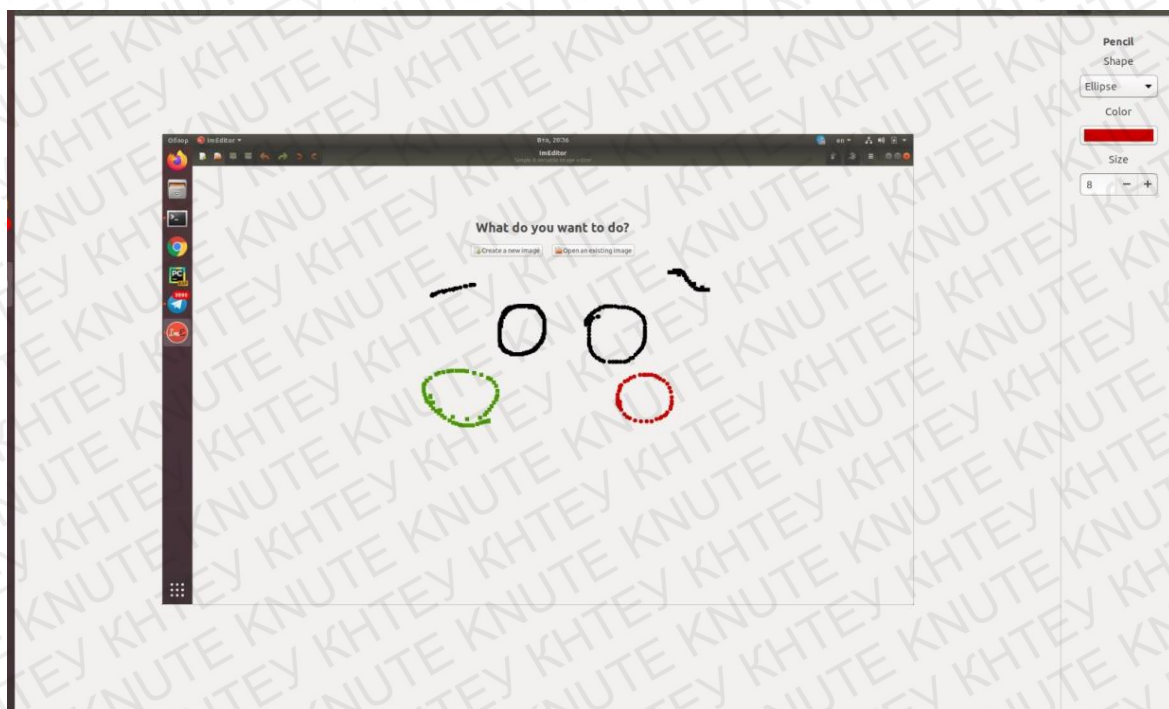


Рис. 3.11. Зображена зміна кольору пензлей для малювання

					Аркуш
					32
Зм.	Аркуш	№ докум	Підпис	Дата	

КНТЕУ 121 07-15.БР

```

def __init__(self, title, context, signals, Parent=None):
    super(Preview, self).__init__(title, Parent)

    self.context = context
    self.signals = signals
    self.parent = Parent
    self.setAllowedAreas(Qt.RightDockWidgetArea)
    self.setFeatures(QtWidgets.QDockWidget.NoDockWidgetFeatures)

    self.label = QtWidgets.QLabel()
    if self.context.getCurrentImagePos() != -1:
        self.label.setPixmap(QtWidgets.QPixmap.fromImage(self.context.currentQImage()))
    self.label.setObjectName("Preview")
    self.label.setSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Maximum)
    self.label.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
    self.setWidget(self.label)
    self.update()

    self.signals.updateCanvas.connect(self.update)
    self.signals.imageChanged.connect(self.update)

def update(self):
    super(Preview, self).update()
    if self.context.currentImage() != None:
        if self.context.currentQImage().width() > 128 or self.context.currentQImage().height() > 128:
            image = self.context.currentQImage().scaled(128, 128, Qt.KeepAspectRatio)
            self.label.setPixmap(QtWidgets.QPixmap.fromImage(image))
        else:
            self.label.setPixmap(QtGui.QPixmap.fromImage(self.context.currentQImage()))
    else:
        self.label = QtWidgets.QLabel()
        self.label.setSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Maximum)
        self.label.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
        self.setWidget(self.label)

```

Рис. 3.12. Приклад коду, що дозволяє при виборі зображення побачити його превью у меню вибору

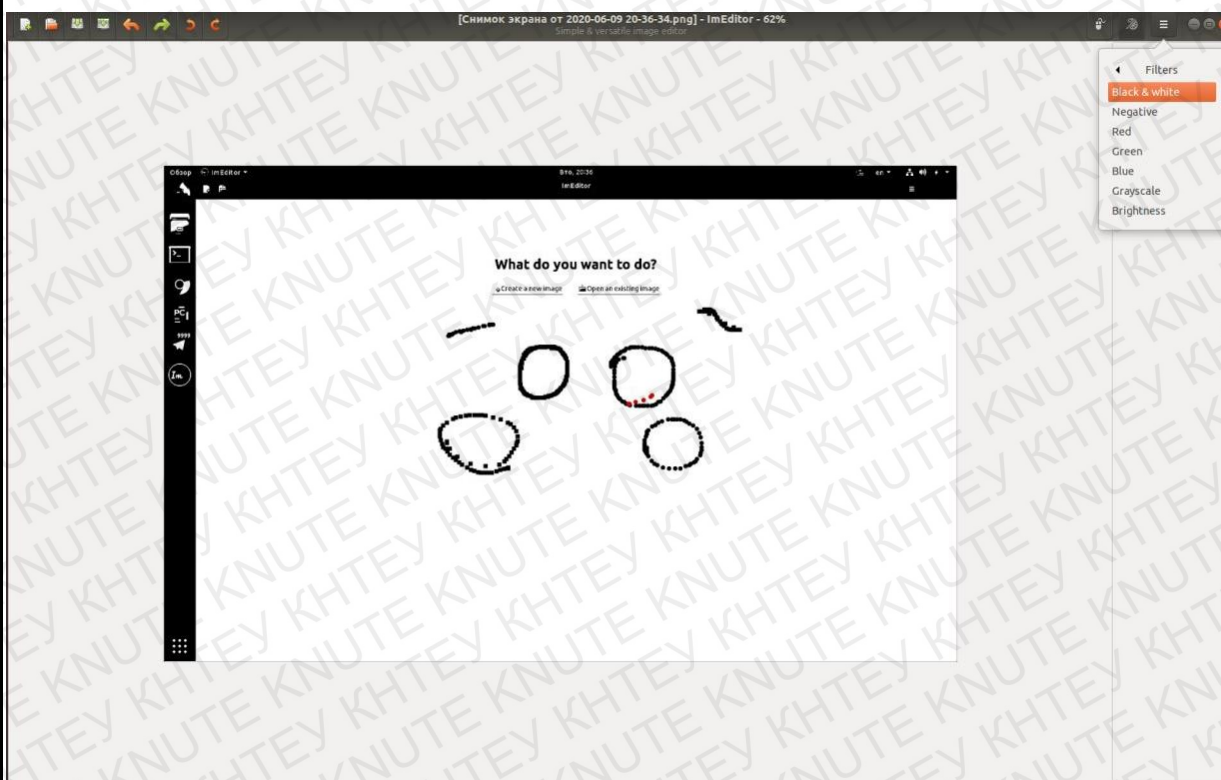


Рис. 3.13. Велика варіативність фільтрів, що має даний додаток

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		33

Також даний додаток має скорочені комбінації клавіш які зображені на зображеннях нижче.

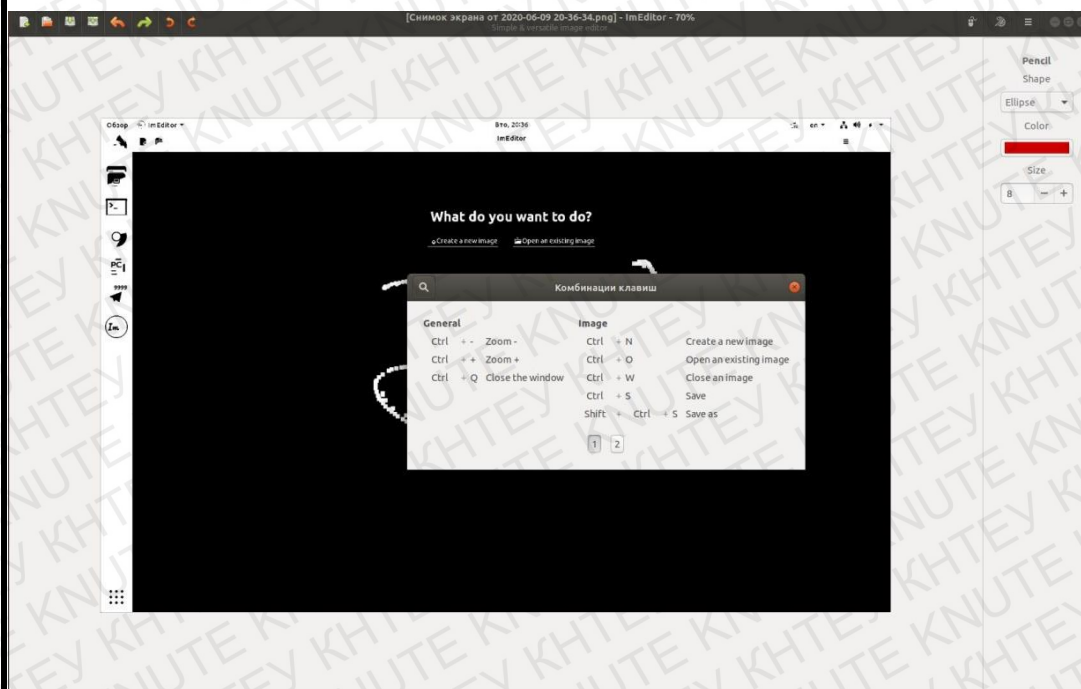


Рис. 3.14. Перша частина клавіш швидкого доступу

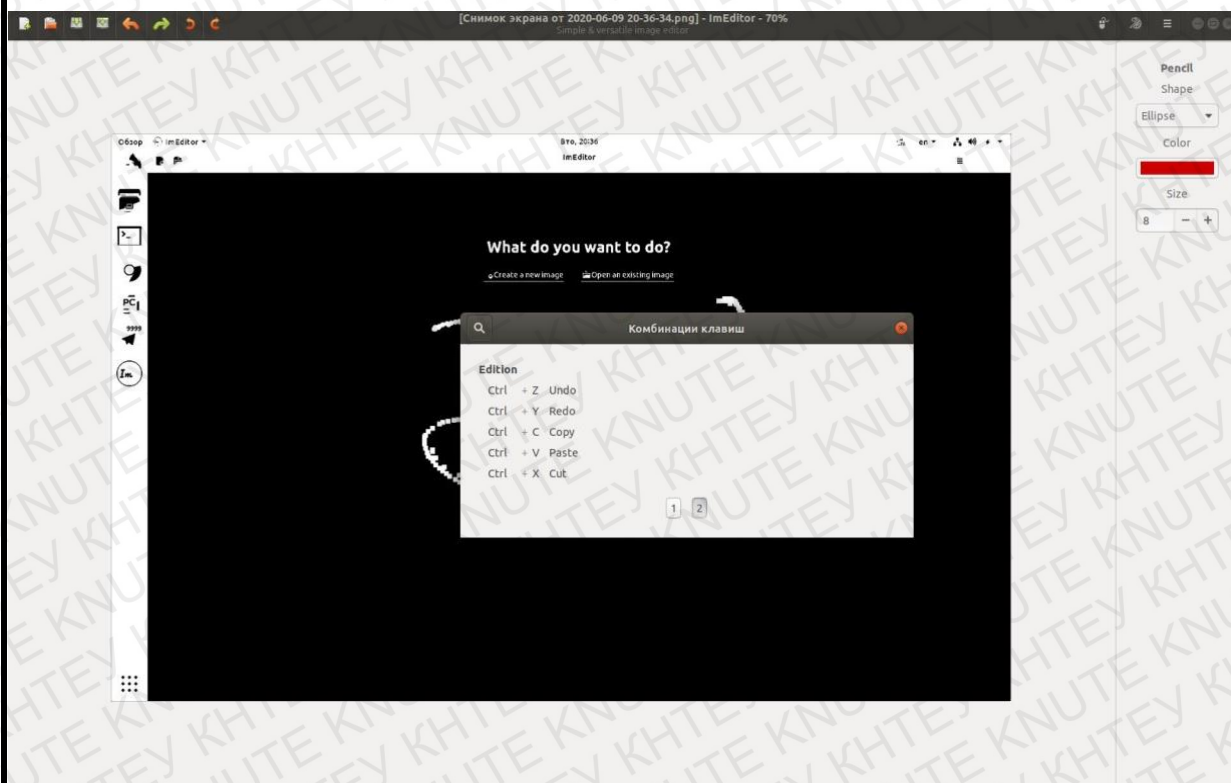


Рис. 3.15. Друга частина клавіш швидкого доступу

					Аркуш
					34
Зм.	Аркуш	№ докум	Підпис	Дата	

КНТЕУ 121 07-15.БР

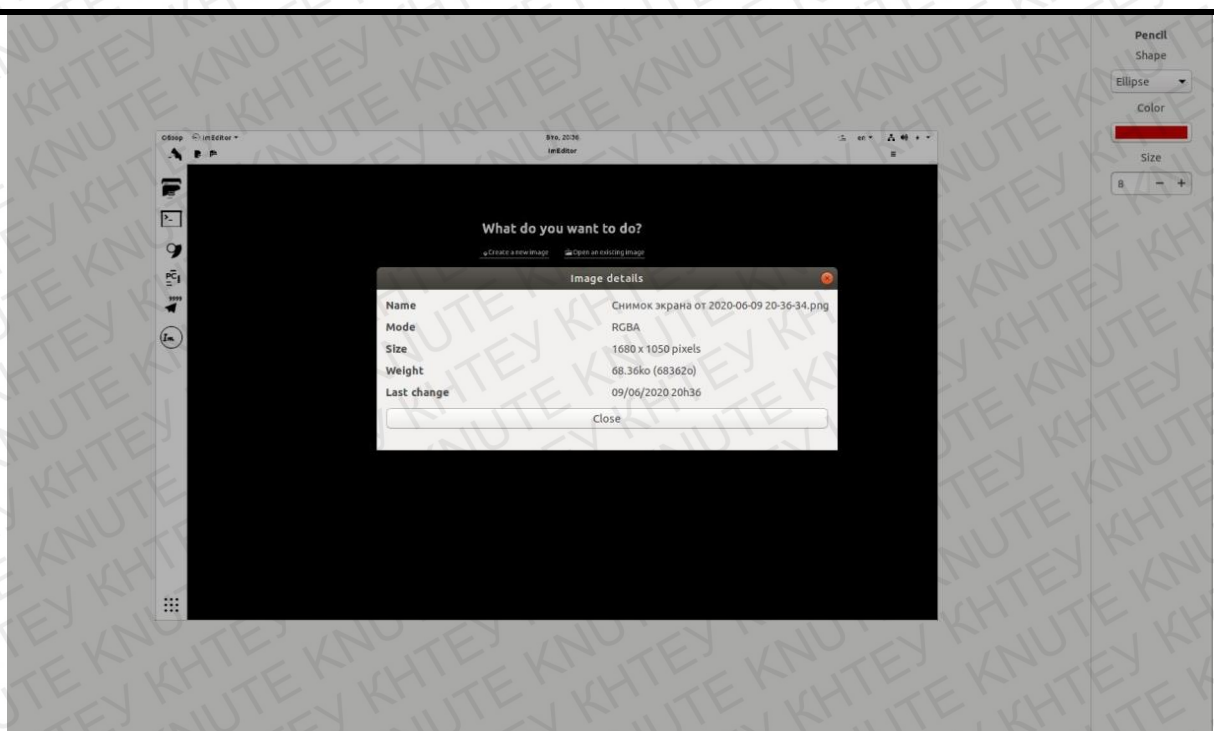


Рис. 3.1 Вивід інформації про характеристики зображення

3.2 Висновки до розділу 3

Розроблено додаток що дозволяє редагувати та видозмінювати зображення а також малювати на ньому.

У мові Python та інструментарії PyCharm було виконано додаток та завдяки цьому досконало вивчено роботу з графічним інтерфейсом та графічними редакторами зазначеної вище мови.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		35

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У ході розробки додатку було проаналізовано вже існуючі програми для видозмінювання зображень та виявлено, що розробці такого роду програмам приділяється багато уваги.

Отож, було вирішено створити проект що з графічним інтерфейсом та здатності до редагування зображень, простий у використанні зручний інтерфейс, розібратися з яким буде не складно будь-кому, а також с великим функціоналом.

Оскільки програмний продукт вирішено створювати для операційної системи Linux, після аналізу популярних мов програмування, було вирішено обрати мову програмування Python та середовище розробки Pycharm, як найкращий варіант для даного проекту. Також було використано SqlLite, як спосіб управління базою даних, оскільки він повністю відповідає вимогам проекту та ідеально інтегрується в обране середовище розробки.

Було проведено системний аналіз, визначено цілі даного продукту, була виконана постановка завдання та розроблено алгоритм створення програми.

База даних є важливою складовою даної програми, тож було спроектовано та розроблено базу даних SQLLite. Після чого було розроблено користувацький інтерфейс та створено, в середовищі розробки PyCharm, проект, який містить методи взаємодії між програмою та базою даних.

Оскільки розробка даного програмного продукту є досить актуальною, розробник має пропозицію покращити дану систему та почати реалізовувати її для розповсюдження мережеюінтернет. У наступній версії планується додати зміну мови, а також теми додатку, також у планах покращити користувацький інтерфейс та додати більшу можливість для редагування фото. Розробка програма була доволі успішною. Було реалізовано значну частину ідей та вимог. Розробка проводилася

					<i>КНТЕУ 121 07-15.БР</i>			
					<i>Розробка програмного забезпечення цифрової обробки зображень</i>	<i>Стадія</i>	<i>Аркуш</i>	<i>Аркушів</i>
<i>Зм.</i>	<i>Аркуш</i>	<i>№ докум</i>	<i>Підпис</i>	<i>Дата</i>		<i>ВП</i>	<i>36</i>	<i>38</i>
<i>Зав. кафедри</i>		<i>Криворучко О.В.</i>				<i>Факультет інформаційних технологій, 4 курс, 7 група</i>		
<i>Керівник</i>		<i>Рзаєва С.Л.</i>						
<i>Гарант</i>		<i>Цензура М.О.</i>						
<i>Розроб.</i>		<i>Сипенко О.О.</i>			<i>Висновки та пропозиції</i>			

поетапно: створено технічне завдання, проведено маркетингове дослідження, вибрано мови програмування та платформи для реалізації, розроблено дизайну та, власне, реалізовано задум продукту.

Результатом роботи стала повністю функціонуюча програма «IAmEditor», яка отримала досить привабливий інтерфейс та вражаючий функціонал. Усі задачі поставлені до розробки були виконані.

					КНТЕУ 121 07-15.БР	Аркуш
Зм.	Аркуш	№ докум	Підпис	Дата		37

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Основний

1. Solem J.E. Programming Computer Vision with Python: Tools and algorithms for analyzing images/O'Reilly Media,— 264 с.
2. Цифрова обробка сигналу на мові Python/O`Reilly Media, Аллен Б.Дауни 2017 — 152с.
3. Документація Stack Overflow.Python, 2018 – 286 с.
4. Python module of the week Автор — Дуг Хеллман.
5. «Python. Докладний довідник» Автор — Девід Бізлі.

Інтернет-ресурси

6. Мова програмування Python. [Електронний ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/Python>
7. Документація PyCharm [Електронний ресурс] – Режим доступу: <https://www.jetbrains.com/ru-ru/pycharm/documentation/>
8. Документація Python. [Електронний ресурс] – Режим доступу: <https://www.python.org/doc/>.
9. Список бібліотек Python. [Електронний ресурс] – Режим доступу: <https://github.com/vinta/awesome-python>
10. Стиль написання коду Python. [Електронний ресурс] – Режим доступу: <http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html>
11. Книга у свободному доступі. [Електронний ресурс] – Режим доступу: <http://learnpythonthehardway.org/book>

					<i>КНТЕУ 121 07-15.БР</i>			
					<i>Розробка програмного забезпечення цифрової обробки зображень</i>	<i>Стадія</i>	<i>Аркуш</i>	<i>Аркушів</i>
<i>Зм.</i>	<i>Аркуш</i>	<i>№ докум</i>	<i>Підпис</i>	<i>Дата</i>		<i>ВД</i>	<i>38</i>	<i>38</i>
<i>Зав. кафедри</i>	<i>Криворучко О.В.</i>					<i>Факультет інформаційних технологій, 4 курс, 7 група</i>		
<i>Керівник</i>	<i>Рзаєва С.Л.</i>							
<i>Гарант</i>	<i>Цензура М.О.</i>							
<i>Розроб.</i>	<i>Силенко О.О.</i>				<i>Список використаних джерел</i>			

ДОДАТКИ

Додаток А

Розробка користувацького інтерфейсу

```
import os

from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtCore import Qt
import names as Pixeler
from mainwidget import MainWidget
from palette import Palette
from toolproperties import ToolProperties
from preview import Preview
from dialogs import NewFileDialog, ResizeImageDialog, ResizeCanvasDialog,
Preferences

class MainWindow(QtWidgets.QMainWindow):

    def __init__(self, context, signals):
        super(MainWindow, self).__init__()
        self.signals = signals
        self.context = context
        self.resize(800,480)
        self.setWindowTitle(self.context.getText("pyqx", "title"))
        self.statusBar = self.statusBar()
        self.menuBar = self.createMenuBar()
        self.toolBar = self.createToolBar()
        self.createDockWidgets()
        self.ctrlPressed = False
        self.mainWidget = MainWidget(context, signals, self)
        self.setCentralWidget(self.mainWidget)
        self.imagePosLabel = QtWidgets.QLabel()
        self.imagePosLabel.setObjectName("ImagePosLabel")
```

```

self.signals.autoUpdateTool.connect(self.setCurrentTool)
self.signals.enterCanvas.connect(self.showImagePosition)
self.signals.leaveCanvas.connect(self.hideImagePosition)
self.signals.overCanvas.connect(self.setImagePosition)
self.show()

def createPopupMenu(self):
    pass

def createToolBarActions(self):
    l = []
    self.tools = QtWidgets.QActionGroup(self)
    tools = ["selection", "magicwand", "pencil", "eraser", "colorpicker",
"fill", "gradient", "exchange"]
    connects = [lambda:
self.context.changeCurrentTool(Pixeler.Tools.Selection),
lambda:
self.context.changeCurrentTool(Pixeler.Tools.MagicWand),
lambda:
self.context.changeCurrentTool(Pixeler.Tools.Pencil),
lambda:
self.context.changeCurrentTool(Pixeler.Tools.Eraser),
lambda:
self.context.changeCurrentTool(Pixeler.Tools.ColorPicker),
lambda:
self.context.changeCurrentTool(Pixeler.Tools.Fill),
lambda:
self.context.changeCurrentTool(Pixeler.Tools.Gradient),
lambda:
self.context.changeCurrentTool(Pixeler.Tools.Exchange)]
    shortcuts = ['Z', ' ', 'X', 'C', 'A', 'S', 'D', '']
    for i in range(len(tools)):

```

```

        a = QtWidgets.QAction(QtGui.QIcon( os.path.join("themes",
self.context.theme, tools[i] + ".png") ), self.context.getText("tools", tools[i]) + " (" +
shortcuts[i] + ")", self.tools)
        a.setCheckable(True)
        a.setShortcut(shortcuts[i])
        if connects[i] != 0: a.toggled.connect(connects[i])
        l.append(a)
    a = QtWidgets.QAction(QtGui.QIcon( os.path.join("themes",
self.context.theme, "zoomin.png") ), self.context.getText("tools", "zoomin"), self.tools)
    a.setShortcut("Ctrl++")
    a.triggered.connect(self.zoomIn)
    l.append(a)
    a = QtWidgets.QAction(QtGui.QIcon( os.path.join("themes",
self.context.theme, "zoomout.png") ), self.context.getText("tools", "zoomout"),
self.tools)
    a.setShortcut("Ctrl+-")
    a.triggered.connect(self.zoomOut)
    l.append(a)
    l[self.context.currentTool].setChecked(True)
    return l

def createToolBar(self):
    toolBar = QtWidgets.QToolBar()
    l = self.createToolBarActions()
    j = 0
    for i in l:
        toolBar.addAction(i)
        if j == 7:
            toolBar.addSeparator()
        j += 1
    toolBar.setMovable(False)

```

```

        toolBar.setOrientation(Qt.Vertical)
        self.addToolBar(Qt.LeftToolBarArea, toolBar)
        return toolBar

    def createFileActions(self):
        ids = ["new", "open", "save", "saveas", "exit"]
        icons = ["document-new.png", "document-open.png", "document-
save.png", "document-save-as.png", "application-exit.png"]
        shortcuts = ['Ctrl+N', 'Ctrl+O', 'Ctrl+S', 'Ctrl+Shift+S', 'Ctrl+Q']
        connects = [self.newFile, self.openFile, self.saveFile, self.saveFileAs,
self.close]
        l = []
        for i in range(len(ids)):
            a = QtWidgets.QAction(QtGui.QIcon("images/" + icons[i]),
self.context.getText("menu_file_labels", ids[i]), self)
            a.setShortcut(shortcuts[i])
            a.triggered.connect(self.restoreFocus)
            a.setStatusTip(self.context.getText("menu_file_status_tips",ids[i]))
            if connects[i] != 0: a.triggered.connect(connects[i])
            l.append(a)
        l.insert(4,0)
        return l

    def createEditActions(self):
        ids = ["undo", "redo", "selectall", "deselect", "cut", "copy", "paste",
"clear", "preferences"]
        icons = ["edit-undo.png", "edit-redo.png", "", "", "edit-cut.png", "edit-
copy.png", "edit-paste.png", "edit-clear.png", "document-properties.png"]
        shortcuts = ['Ctrl+Z', 'Ctrl+Y', "Ctrl+A", "Ctrl+Shift+A", 'Ctrl+X',
'Ctrl+C', 'Ctrl+V', 'Del', ""]
        connects = [self.undo, self.redo, self.selectAll, self.deselect, self.cut,
self.copy, self.paste, self.clear, self.showPreferences]

```

```

l = []
for i in range(len(ids)):
    a = QtWidgets.QAction(QtGui.QIcon("images/" + icons[i]),
self.context.getText("menu_edit_labels", ids[i]), self)
    a.setShortcut(shortcuts[i])
    a.triggered.connect(self.restoreFocus)
    a.setStatusTip(self.context.getText("menu_edit_status_tips",
ids[i]))
    if connects[i] != 0: a.triggered.connect(connects[i])
    l.append(a)
l.insert(2,0)
l.insert(5,0)
l.insert(10,0)
return l

```

```

def createViewActions(self):
    ids = ["pixel_grid", "matrix_grid"]
    icons = ["", ""]
    shortcuts = ['Ctrl+G', 'Ctrl+M']
    connects = [self.setPixelGrid, self.setMatrixGrid]
    l = []
    for i in range(len(ids)):
        a = QtWidgets.QAction(QtGui.QIcon("images/" + icons[i]),
self.context.getText("menu_view_labels", ids[i]), self)
        a.setShortcut(shortcuts[i])
        a.triggered.connect(self.restoreFocus)
        a.setStatusTip(self.context.getText("menu_view_status_tips",
ids[i]))
        if connects[i] != 0: a.triggered.connect(connects[i])
        a.setCheckable(True)

```

```

        l.append(a)
    l.insert(2,0)
    l[0].setCheckable(True)
    if self.context.grid: l[0].setChecked(True)
    l[1].setCheckable(True)
    if self.context.matrixGrid: l[1].setChecked(True)
    return l

    def createTransformActions(self):
        ids = ["flip_hor", "flip_ver", "rotate_cw", "rotate_ccw", "rotate_180",
"resize", "resize_canvas"]
        icons = [ "", "", "", "", "", "", "" ]
        shortcuts = [ "", "", "", "", "", "" ]
        connects = [
[self.flipHorizontally,self.flipVertically,self.rotate90CW,self.rotate90CCW,self.rotate180,self.showResizeImageDialog,self.showResizeCanvasDialog]
        l = []
        for i in range(len(ids)):
            a = QtWidgets.QAction(QtGui.QIcon("images/" + icons[i]),
self.context.getText("menu_transform_labels", ids[i]), self)
            a.setShortcut(shortcuts[i])
            a.triggered.connect(self.restoreFocus)
            a.setStatusTip(self.context.getText("menu_transform_status_tips", ids[i]))
            if connects[i] != 0: a.triggered.connect(connects[i])
            l.append(a)

        l.insert(2,0)
        l.insert(6,0)
        return l

    def createHelpActions(self):

```

```

ids = ["contents", "about"]
icons = ["help-contents.png", "help-about.png"]
shortcuts = ['F1', 'Ctrl+B']
connects = [self.showHelp, self.showAboutDialog]
l = []
for i in range(len(ids)):
    a = QtWidgets.QAction(QtGui.QIcon("images/" + icons[i]),
self.context.getText("menu_help_labels", ids[i]), self)
    a.setShortcut(shortcuts[i])
    a.triggered.connect(self.restoreFocus)
    a.setStatusTip(self.context.getText("menu_help_status_tips",
ids[i]))

    if connects[i] != 0: a.triggered.connect(connects[i])
    l.append(a)
l.insert(1,0)
return l

def createMenuBar(self):
    menubar = self.menuBar()
    fileMenu = menubar.addMenu(self.context.getText("menu", "file"))
    editMenu = menubar.addMenu(self.context.getText("menu", "edit"))
    viewMenu = menubar.addMenu(self.context.getText("menu",
"view"))
    transformMenu = menubar.addMenu(self.context.getText("menu",
"transform"))
    helpMenu = menubar.addMenu(self.context.getText("menu", "help"))
    fileActions = self.createFileActions()
    editActions = self.createEditActions()
    viewActions = self.createViewActions()
    transformActions = self.createTransformActions()
    helpActions = self.createHelpActions()

```

```

        for i in fileActions:
            if i == 0: fileMenu.addSeparator()
            else: fileMenu.addAction(i)
        for i in editActions:
            if i == 0: editMenu.addSeparator()
            else: editMenu.addAction(i)
        for i in viewActions:
            if i == 0: viewMenu.addSeparator()
            else: viewMenu.addAction(i)
        for i in helpActions:
            if i == 0: helpMenu.addSeparator()
            else: helpMenu.addAction(i)
        for i in transformActions:
            if i == 0: transformMenu.addSeparator()
            else: transformMenu.addAction(i)
        return menubar

    def createDockWidgets(self):
        self.palette =
QtWidgets.QDockWidget(self.context.getText("dock_widgets", "palette"), self)
        self.palette.setAllowedAreas(Qt.RightDockWidgetArea)
        self.palette.setFeatures(QtWidgets.QDockWidget.NoDockWidgetFeatures)
        paletteWidget = Palette(self.context, self.signals)
        self.palette.setWidget(paletteWidget)
        self.palette.setSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Minimum)
        self.addDockWidget(Qt.RightDockWidgetArea, self.palette)
        # Tool Properties widget
        self.toolProperties =
ToolProperties(self.context.getText("dock_widgets", "tool_properties"), self.context,
self.signals)

```

```

self.addDockWidget(Qt.RightDockWidgetArea, self.toolProperties)
self.toolProperties.setSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Expanding)

# Preview
self.preview = Preview(self.context.getText("dock_widgets",
"preview"), self.context, self.signals, self)

self.addDockWidget(Qt.RightDockWidgetArea, self.preview)

def restoreFocus(self):
    print("Restoring Focus")
    self.ctrlPressed = False
    self.releaseMouse()
    self.releaseKeyboard()
    QtCore.QCoreApplication.instance().restoreOverrideCursor()

def setCurrentTool(self, index):
    self.tools.actions()[0].setChecked(True)
    self.signals.updateTool.emit(0)

def zoomIn(self):
    if self.context.currentImage().zoom < 25:
        self.context.currentImage().zoom += 1
        self.signals.zoom.emit()

def zoomOut(self):
    if self.context.currentImage().zoom > 1:
        self.context.currentImage().zoom -= 1
        self.signals.zoom.emit()

def newFile(self):
    d = NewFileDialog(self.context, self)

def openFile(self):
    fileName =
QtWidgets.QFileDialog.getOpenFileName(self,
self.context.getText("dialog_open", "title"),

```

```

        "/home",
        self.context.getText("dialog_open", "images") +
u" (*.bmp *.gif *.png *.xpm *.jpg);;" + self.context.getText("dialog_open", "all_files")
+ u" (*)")

        if fileName:
            self.context.loadImage(fileName)
        def saveFile(self):
            if self.context.currentImage().fileName == "":
                self.saveFileAs()
            else:
                self.context.currentImage().save()
        def saveFileAs(self):
            d = QtWidgets.QFileDialog()
            fileName, filterName = d.getSaveFileName(self,
                self.context.getText("dialog_save", "title"),
                "",
                "*.bmp;;*.gif;;*.png;;*.xpm;;*.jpg")
            if fileName.split(".")[1] in ["bmp", "gif", "png", "xpm", "jpg"]:
                self.context.currentImage().fileName = fileName
            self.signals.fileNameChanged.emit(self.context.getCurrentImagePos(),
os.path.basename(str(fileName)))
            else:
                self.context.currentImage().fileName = fileName +
filterName[1:]

            self.signals.fileNameChanged.emit(self.context.getCurrentImagePos(),
os.path.basename(str(fileName + filterName[1:])))
            self.context.currentImage().save()
        def close(self):
            pass

```

```

def undo(self):
    if self.context.currentImage().posHistory > 0:
        self.context.currentImage().posHistory -= 1
        self.context.currentImage().image=
QtWidgets.QImage(self.context.currentImage().history[self.context.currentImage().pos
History])

        self.signals.updateCanvas.emit()
        self.signals.resizeCanvas.emit()

def redo(self):
    if self.context.currentImage().posHistory <
len(self.context.currentImage().history)-1:
        self.context.currentImage().posHistory += 1
        self.context.currentImage().image =
QtWidgets.QImage(self.context.currentImage().history[self.context.currentImage().pos
History])

        self.signals.updateCanvas.emit()
        self.signals.resizeCanvas.emit()

def selectAll(self):
    self.mainWidget.currentWidget().canvas.selectAll()
def deselect(self):
    self.mainWidget.currentWidget().canvas.applySelection()
def cut(self):
    self.signals.cutImage.emit()
def copy(self):
    self.signals.copyImage.emit()
def paste(self):
    clipboard = QtWidgets.QApplication.clipboard()
    if not clipboard.image().isNull():
        self.signals.pasteImage.emit()

```

```

        self.signals.updateCanvas.emit()

    def clear(self):
        self.signals.clearImage.emit()

    def showPreferences(self):
        d = Preferences(self.context, self.signals, self)

    def setPixelGrid(self):
        self.context.grid = not self.context.grid
        self.signals.updateCanvas.emit()
        self.context.setDefault("grid", "grid", self.context.grid)

    def setMatrixGrid(self):
        self.context.matrixGrid = not self.context.matrixGrid
        self.signals.updateCanvas.emit()
        self.context.setDefault("grid", "matrix_grid", self.context.matrixGrid)

    def flipHorizontally(self):
        pass

    def flipVertically(self):
        pass

    def rotate90CW(self):
        pass

    def rotate90CCW(self):
        pass

    def rotate180(self):
        pass

    def showResizeImageDialog(self):
        pass

    def showResizeCanvasDialog(self):
        pass

    def showHelp(self):
        pass

    def showAboutDialog(self):

```

```

        pass
    def showImagePosition(self):
        if self.imagePosLabel.isHidden():
            self.statusBar.insertWidget(0, self.imagePosLabel, 0)
            self.imagePosLabel.show()
    def hideImagePosition(self):
        self.statusBar.removeWidget(self.imagePosLabel)
    def setImagePosition(self, x, y):
        self.imagePosLabel.setText(" Pos: (" + str(x) + ", " + str(y) + ")")
    def keyPressEvent(self, event):
        super(MainWindow, self).keyPressEvent(event)
        if event.key() == Qt.Key_Control:
            print("Control Pressed")
            self.ctrlPressed = True
        QtCore.QCoreApplication.instance().setOverrideCursor(self.context.colorP
ickerCur)
            self.signals.ctrlPressed.emit()
            self.grabMouse()
            self.grabKeyboard()
        elif event.key() == Qt.Key_Plus:
            if self.context.currentTool == 1:
                self.context.setPencilSize(self.context.pencilSize+1)
            elif self.context.currentTool == 2:
                self.context.setEraserSize(self.context.eraserSize+1)
        elif event.key() == Qt.Key_Minus:
            if self.context.currentTool == 1:
                self.context.setPencilSize(self.context.pencilSize-1)
            elif self.context.currentTool == 2:
                self.context.setEraserSize(self.context.eraserSize-1)
        else:

```

```

QtCore.QCoreApplication.instance().restoreOverrideCursor()
self.releaseMouse()
self.releaseKeyboard()
def keyPressEvent(self, event):
    super(MainWindow, self).keyPressEvent(event)
    if event.key() == Qt.Key_Control:
        self.ctrlPressed = True
        QtCore.QCoreApplication.instance().restoreOverrideCursor()
        self.releaseMouse()
        self.releaseKeyboard()
def mousePressEvent(self, event):
    super(MainWindow, self).mousePressEvent(event)
    if self.ctrlPressed:
        print("Picking Desktop Color")
        widget =
QtCore.QCoreApplication.instance().desktop().screen()
        im = QtWidgets.QPixmap.grabWindow(widget.winId()).toImage()
        c = QtWidgets.QColor(im.pixel(QtWidgets.QCursor.pos()))
        if event.button() == Qt.LeftButton:
            self.context.changePrimaryColor(c)
        elif event.button() == Qt.RightButton:
            self.context.changeSecondaryColor(c)
def mouseMoveEvent(self, event):
    super(MainWindow, self).mouseMoveEvent(event)
    if self.ctrlPressed:
        widget = QtCore.QCoreApplication.instance().desktop().screen()
        im = QtWidgets.QPixmap.grabWindow(widget.winId()).toImage()
        c = QtWidgets.QColor(im.pixel(QtWidgets.QCursor.pos()))
        if event.buttons() == Qt.LeftButton:
            self.context.changePrimaryColor(c)

```

```

        elif event.buttons() == Qt.RightButton:
            self.context.changeSecondaryColor(c)

    def wheelEvent(self, event):

        if self.ctrlPressed:
            if event.delta() > 0:
                self.zoomIn()
            else:
                self.zoomOut()

        super(MainWindow, self).wheelEvent(event)

    def closeEvent(self, event):
        l = []
        for i in range(len(self.context.images)):
            if self.context.images[i].modified: l.append(i)
        if len(l) > 0:
            reply= QtWidgets.QMessageBox.warning(self,      self.context.getText("dialog_exit",
            "title"),

                self.context.getText("dialog_exit", "message"),
                QtWidgets.QMessageBox.SaveAll
                |
                QtWidgets.QMessageBox.Discard | QtWidgets.QMessageBox.Cancel,
                QtWidgets.QMessageBox.Cancel)
            if reply == QtWidgets.QMessageBox.Discard:event.accept()
            elif reply == QtWidgets.QMessageBox.Cancel:event.ignore()
                return
            elif reply == QtWidgets.QMessageBox.SaveAll:for i in l:
                self.mainWidget.setCurrentIndex(i)
                self.context.setCurrentImagePos(i)self.saveFile()event.accept()
                return
            self.context.saveDefaults()
            super(MainWindow, self).closeEvent(event)

```