НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис)

“___” _____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

по спеціальності **123 «Комп'ютерна інженерія»**

на тему: ЗАСОБИ ДЛЯ СТВОРЕННЯ І РЕДАГУВАННЯ ГРАФІЧНИХ ОБ’ЄКТІВ

Виконав: студент IV курсу, групи КВ-74

Шепель Костянтин Святославович _____
(підпис)

Керівник ст. викл. Дробязко І. П. _____
(підпис)

Консультант з нормоконтролю доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(підпис)

Рецензент _____ _____
(підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
«___» 2021 р.

ЗАВДАННЯ
на дипломний проект студента

Шепеля Костянтина Святославовича

1. Тема проекту Засоби для створення і редагування графічних об’єктів, керівник проекту Дробязко І.П., ст. викл. кафедри СПКС, затверджені наказом по університету від «25» травня 2021р. №1331-С

2. Термін подання студентом проєкту « 9 » червня 2021р.

3. Вихідні дані до проекту: див. Технічне завдання

4. Зміст пояснювальної записки:

–Аналіз існуючих рішень та обґрунтування теми дипломного проекту;
–Вибір технологій і програмних засобів для реалізації інтерфейсу додатку;
–Вибір алгоритмів для редагування растрових графічних зображень;
–Опис розроблених засобів редагування.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо) :

- Схема взаємодії компонентів інтерфейсу редактору.
- Схема взаємодії функціональних елементів редактору.

– Алгоритм побудови довільної кривої інструментами малювання.
– Схема взаємодії користувача з додатком.
– Презентація дипломного проєкту

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент.		

7. Дата видачі завдання 30 жовтня 2020р.

Календарний план

№ з/п	Назва етапів виконання дипломного прокту	Термін виконання етапів
1.	Видача завдання на дипломне проєктування	30.10.2020
2.	Вивчення літератури за тематикою проєкту	17.11.2020
3.	Розроблення та узгодження технічного завдання	02.12.2021
4.	Аналіз існуючих рішень	13.12.2021
5.	Підготовка матеріалів першого розділу дипломного проєкту	05.02.2021
6.	Підготовка матеріалів другого розділу дипломного проєкту	17.03.2021
7.	Підготовка матеріалів третього розділу дипломного проєкту	12.04.2021
8.	Підготовка матеріалів четвертого розділу дипломного проєкту	01.05.2021
9.	Підготовка графічної частини дипломного проєкту	12.05.2021
10.	Оформлення документації дипломного проєкту	15.05.2021
11.	Попередній огляд матеріалів диплому на кафедрі	27.05.2021

Студент _____ Костянтин Шепель

Керівник проєкту _____ Ірина ДРОБЯЗКО

АНОТАЦІЯ

Дипломний проєкт включає пояснювальну записку (57 с., 29 рис., 4 додатки).

Об'єкт розробки – засоби для створення і редагування графічних об’єктів.

Метою дипломного проєкту є створення ефективних і зручних у використанні інструментів для побудови та редагування растрових графічних зображень.

У ході розробки:

- проведено аналіз сучасних алгоритмів і програмних засобів побудови растрових зображень;
- визначено алгоритми та розроблено структуру і функціональні компоненти редактора;
- забезпечено підтримку використання графічного планшету Wacom;
- розроблено зручний GUI для взаємодії з користувачем.

Розроблені засоби редагування дозволяють:

- відтворювати ефект малювання різними художніми інструментами;
- обирати необхідний колір на палітрі RGB відтінків;
- будувати примітивні растрові об’єкти (окружність, лінію, прямокутник та ін.);
- коректно зафарбовувати визначену користувачем зону.

Для розробки графічного інтерфейсу користувача використано фреймворк pyGame. Функціональні компоненти написані мовою Python.

Використання створених засобів дозволить користувачу підвищити якість та швидкість створення растрових графічних зображень.

Ключові слова: графічний редактор, Wacom, GUI, RGB, pyGame, Python.

ABSTRACT

The diploma project includes an explanatory note (Article 57, Fig. 29, 4 appendices).

Object of development - tools for creating and editing graphic objects.

The aim of the diploma project is to create effective and easy-to-use tools for building and editing raster graphics.

During development:

– analysis of modern algorithms and software of construction of raster images is carried out;

– algorithms are defined and the structure and functional components of the editor are developed;

– support for the use of Wacom graphics tablet is provided;

– developed a user-friendly GUI for user interaction.

Developed editing tools allow:

– reproduce the effect of drawing with various artistic tools;

– choose the desired color on the palette of RGB shades;

– build primitive raster objects (circle, line, rectangle, etc.);

– correctly paint the user-defined area.

pyGame framework was used to develop the graphical user interface. Functional components have been written on Python.

Using this application will allow the user to improve the quality and speed of raster graphics.

Keywords: graphic editor, Wacom, GUI, RGB, pyGame, Python.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.002 ТЗ	Засоби для створення	4		
			і редагування			
			графічних об’єктів.			
			Технічне завдання			
	А4	ІАЛЦ.045480.003 ТП	Засоби для створення	3		
			і редагування			
			графічних об’єктів.			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045480.004ПЗ	Засоби для створення	57		
			і редагування			
			графічних об’єктів.			
			Пояснювальна записка			
	А4	ІАЛЦ.045480.005Д1	Засоби для створення			
			і редагування			
			графічних об’єктів.			
			Взаємодія компонентів			
			інтерфейсу редактора.			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045480.001 ОА		
Розробив		Шепель К.С.			Засоби для створення і редагування графічних об’єктів **Опис альбому**	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Дробязко І.П.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-74	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.006Д2	Засоби для створення	1		
			і редагування			
			графічних об’єктів.			
			Взаємодія			
			функціональних			
			елементів редактора.			
			Схема структурна			
	А4	ІАЛЦ.045480.007Д3	Засоби для створення	1		
			і редагування			
			графічних об’єктів.			
			Процедура побудови			
			довільної кривої.			
			Схема алгоритму			
	А4	ІАЛЦ.045480.008Д4	Засоби для створення	1		
			і редагування			
			Графічних об’єктів			
			Взаємодія			
			користувача з додатком			
			Схема алгоритму			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал.			
			Фрагменти програмного			
			коду.			
			Презентація дипломного			
			проєкту			

						Арк.
					ІАЛЦ.045480.001 ОА	3
Змін.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: Засоби для створення і редагування графічних об’єктів.

Галузь застосування: будь-яка галузь, для якої потрібно наочне подання інформації (реклама, поліграфія, моделювання та ін.).

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення ефективних і зручних у використанні інструментів для побудови та редагування растрових графічних зображень.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь-якою операційною системою;
- наявність системи розмітки графічних компонент;
- можливість створення нового графічного зображення;
- можливість редагування графічного зображення;

- наявність зручної системи розташувань віджетів;
- можливість динамічної зміни розміру графічного пензла;
- можливість одночасного редагування групи пікселів;
- наявність системи швидкого доступу до графічних компонент;
- зберігання растрових графічних зображень;
- наявність системи вибору відтінків.

5.2. Вимоги до апаратного забезпечення

- Стандартний дисплей, побудований на рідких кристалах;
- Оперативна пам’ять 1024 Mb;
- Наявність встановленого драйвера Wacom 6.4.43-3;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows 7 та вище;
- Наявність встановленого драйвера Wacom 6.4.43-3;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного прокту	Термін виконання етапів
1.	Видача завдання на дипломне проєктування	30.10.2020
2.	Вивчення літератури за тематикою проєкту	17.11.2020
3.	Розроблення та узгодження технічного завдання	02.12.2021
4.	Аналіз існуючих рішень	13.12.2021
5.	Підготовка матеріалів першого розділу дипломного проєкту	05.02.2021
6.	Підготовка матеріалів другого розділу дипломного проєкту	17.03.2021
7.	Підготовка матеріалів третього розділу дипломного проєкту	12.04.2021
8.	Підготовка матеріалів четвертого розділу дипломного проєкту	01.05.2021
9.	Підготовка графічної частини дипломного проєкту	12.05.2021
10.	Оформлення документації дипломного проєкту	15.05.2021
11.	Попередній огляд матеріалів диплому на кафедрі	27.05.2021

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.004	Засоби для створення	57		
			і редагування			
			графічних об’єктів.			
			Пояснювальна			
			записка			
	А4	ІАЛЦ.045480.005Д1	Засоби для створення			
			і редагування	1		
			графічних об’єктів.			
			Взаємодія компонентів			
			інтерфейсу редактора.			
			Схема структурна			
	А4	ІАЛЦ.045480.006Д2	Засоби для створення			
			і редагування			
			графічних об’єктів.			
			Взаємодія			
			функціональних			
			елементів редактора.			
			Схема алгоритму			

					ІАЛЦ.045480.003ТП		
Змін	Арк.	№ докум.	Підпис	Да			
Розробив		Шепель К.С.			Засоби для створення і редагування графічних об’єктів	Літ.	Аркуш 1 / Аркушів 2
Перевірила		Дробязко І.П.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-74	
Н. контроль		Клятченко Я.М.			**Відомість технічного проєкту**		
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.007Д3	Засоби для створення	1		
			і редагування			
			графічних об’єктів.			
			Процедура побудови			
			довільної кривої.			
			Схема алгоритму			
	А4	ІАЛЦ.045440.008Д4	Засоби для створення	1		
			і редагування			
			графічних об’єктів.			
			Схема взаємодії			
			користувача з додатком.			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної			
			записки. Графічний	1		
			матеріал. Фрагменти			
			програмного коду.			
			Презентація дипломного			
			проєкту			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045480.003ТП	Арк. 2

Пояснювальна записка
до дипломного проекту

на тему: Засоби для створення і редагування графічних об’єктів

Київ–2021 року

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Затравлюючий піксель – піксель усередині області, з якого починає роботу алгоритм зафарбовування.

Кубічний сплайн – ділянка траєкторії, що являє собою окрему криву Без'є, побудовану по чотирьом точкам [1].

Aliasing – візуальний сходовий дефект контуру растрових графічних об'єктів.

dpi – кількість точок в одному дюймі зображення.

GUI – графічний інтерфейс користувача.

Pressure level – рівень, що позначає силу тиску графічного пера на поверхню.

Python shell – інтерактивний інтерпретатор Python.

RGB – поширена адитивна модель відтінків, яка показує спосіб побудови та встановлення кольору.

ВСТУП

На сьогоднішній день у більшості галузей по всьому світу можна зустріти приклади графічних зображень: починаючи від різноманітних дизайнів веб-додатків або реклам і закінчуючи зображеннями в новинах. Поширеність використання пояснюється тим, що новий матеріал легше сприймається користувачем у випадку візуалізації поданої інформації.

Зображення – це результат виконання операцій, орієнтованих на створення або редагування певних графічних об’єктів. Його кінцева якість залежить від алгоритмів відображення та побудови елементарних складових. Оскільки існуючі рішення створення зображень мають певні недоліки, виникає необхідність у розробці набору засобів для редагування, який би відповідав потребам більшості клієнтів.

Користувачу некомфортно використовувати усі засоби редагування окремо один від одного. Оскільки такі інструменти направлені на побудову зображення, доцільно об’єднати їх в одному додатку. Набір програмних засобів для створення, обробки та редагування об’єктів являє собою графічний редактор.

Всі функції, що містить такий додаток, пов’язані між собою та відображені за допомогою графічного інтерфейсу. Розміщення елементів впливає не тільки на зовнішній вигляд редактора, але й на швидкість взаємодії користувача з інструментами редагування.

Саме тому пропонується розробка засобів, яких достатньо для зручного створення та редагування графічних об’єктів, а також загального інтерфейсу, який надає швидкий доступ до кожного із зазначених елементів.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Загальна класифікація графічних редакторів

Графічний редактор – це програма, основною функцією якої є створення або редагування об’єктів та їх візуалізація у вигляді зображення на моніторі пристрою, з яким взаємодіє користувач. Результат обробки зберігається у вигляді файлу [2].

Додатки для редагування або обробки двовимірних зображень являють собою набір засобів та, в залежності від галузі застосування, орієнтовані на растрову або векторну графіку, хоча існують і універсальні, що мають обидві категорії інструментів. Результати, отримані за допомогою растрової графіки, частіше за все використовується для візуального подання зображень. Багато програм може працювати з растровими файлами, а також якість, отримана при обробці растровими засобами редагування, набагато вища ніж векторними. Мова йде не про чіткість побудованих об’єктів, а про їх деталізованість та загальну привабливість. Якщо найменші векторні об’єкти являють собою прості математичні фігури, то для растрових – це завжди один піксель [3]. Кожен такий компонент даних зберігає у собі інформацію про колір, місце розташування його на екрані та в результаті формує графічний файл. Особливістю зображення, обробленого растровими засобами редагування, є можливість забезпечення близької до оригіналу реалістичності, яку неможливо отримати, використовуючи векторні примітиви.

Переважна більшість зображень, отриманих за допомогою растрових засобів редагування, використовуються в галузях, для яких потрібне наочне подання інформації, наприклад в рекламі, поліграфії, для побудови сайтів та ін. [2]. Художники-ілюстратори не обмежують себе набором математичних об’єктів, віддаючи перевагу растровій графіці, а ні ж векторній.

1.2 Основні проблеми растрових графічних редакторів

Для сучасного користувача, котрий хоче працювати з універсальним інструментом створення та редагування графічних об’єктів, існує велика кількість растрових редакторів. Одні більше спеціалізуються на редагуванні вже існуючих зображень, інші – на створенні з нуля. Логічно припустити, що найкращим варіантом будуть такі додатки, що мають найбільш широкий функціонал. Однак більшу частину засобів у великому середовищі розробки займають інструменти, їх параметри та налаштування, котрими звичайний користувач може ніколи не скористуватися. Додаткові зміни у додатку найчастіше не покращують результат, який можна отримати інструментами, встановленими за замовчуванням, а лише дозволяють виконувати вузько направлені завдання. Більш того, велика кількість можливостей, де не встановлено пріоритет між основними (базовими) та додатковими можливостями, ускладнює інтерфейс користувача. Процес вивчення основних функціональних особливостей займає багато часу та врешті решт може відвернути користувача.

Окрім складного функціоналу, існує такий важливий критерій як розмір файлу установки. В таких програмах, як Creative Cloud та CS6, розмір коливається від 300 Мбайт до 2 Гбайт [4]. Оскільки, за замовчуванням, розробники багатьох таких додатків рекомендують для коректної роботи встановлювати програму на системний диск, можливі зависання та уповільнення виконання деяких функцій. При створенні зображення базовими інструментами досить часто з’являється необхідність користувачу виконати тонку та плавну дію, тому навіть незначна затримка може повністю зіпсувати результат.

У переважної більшості графічних редакторів з обмеженим набором засобів створення зображень, проміжок часу між “дією” та “відгуком” є настільки малим, що можна говорити про “миттєвий відгук”. Частіше за все,

розмір таких додатків не перевищує 30 Мбайт. Ймовірність затримки в таких програмах є, але значно менша, ніж у великих, та не через системні складнощі, а через погану реалізацію алгоритмів роботи відповідних функцій. Більшість таких додатків дозволяють однаково змінювати властивість тільки певної групи пікселів, що псує головну перевагу растрової графіки над векторною, а саме – можливість варіювати кольором зображення у кожному пікселі.

Існує велика кількість алгоритмів реалізації кожного функціонального елементу графічного редактора. Однак найголовнішими з цих елементів є ті, що впливають на візуальне сприйняття результату та відображення його на моніторі комп'ютера. Деякі з алгоритмів малювання – найважливішої частини растрового графічного редактора – змінювалися вже після релізу програм. Через це результат виконання однієї й тієї ж операції малювання в різних додатках оброблюється по-різному. З'являються незначні похибки, які можна побачити та які будуть мати значний вплив на кінцеве зображення через постійне використання неоптимізованого інструменту. "Еталоном" результату роботи є форма руху звичайного пензля по паперу.

Під час роботи у додатку користувач виконує безліч змін на графічному холсті за допомогою відповідних інструментів. Очевидно, що зміна опції кожен раз "вручну", тобто при наведенні курсору на відповідний елемент, буде не тільки займати багато зайвого часу, але й дратувати користувача. Багато редакторів не спроможні на отримання складного результату растрового зображення, тому опція швидкого доступу до базових віджетів або "гарячих клавіщ" в них не доступна.

Користуючись мишкою або тачпадом при пересуванні курсору по екрану, коли відбувається подія зміни кольору певних пікселів на полотні, неможливо отримати плавний результат. В деяких місцях результат буде викривленим, а також з'являться "стрибки" при нерівномірному руху унаслідок тертя [5]. Розмір групи пікселів, колір яких буде змінюватися під час пересування курсору залишиться таким, яким він був встановлений перед

натисканням відповідного віджету. Це обмежує користувача та погіршує кінцевий результат. Для управління розміром редагованої групи пікселів використовується графічний планшет, підтримка якого є далеко не в усіх редакторах. Навіть у тих додатках, де підтримка графічного пензла присутня, є проблеми з плавним натиском. На початку зміщення курсору з'являються так звані "нерівномірності", коли перші декілька пікселів за напрямком руху пензла зафарбовуються нерівномірно. Такий візуальний дефект з'являється у наслідок погано оптимізованих алгоритмів взаємодії між графічним обладнанням.

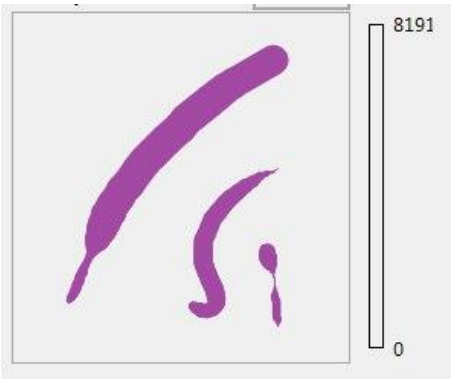

Рисунок 1 – Приклад дефекту графічного пера

Отже, підсумовуючи всю інформацію, наведену вище, можна виділити список критеріїв, яким повинен відповідати растровий редактор для задоволення загальних потреб користувача:

1) простота інтерфейсу;

2) невеликий розмір файлу установки;

3) відсутність будь-яких затримок під час використання редагування растрового полотна;

4) наявність оптимізованих алгоритмів;

5) можливість доступу до базового функціоналу за допомогою гарячих клавіш;

6) можливість обробки подій від графічного планшету.

1.3 Аналіз існуючих графічних редакторів

Редактор Microsoft Paint

Серед усіх растрових редакторів Microsoft Paint відомий кожному, хто користується Windows, оскільки цей додаток вбудовано в операційну систему за замовчуванням [6]. Після кількох переробок редактор отримав зручний та зрозумілий інтерфейс, який представлено на рис. 1. До складу базового функціоналу Paint для Windows 7 входить цифровий пензл, що дозволяє робити лінії однакової товщини та прозорості, олівець, гумка та заливка, які в сукупності дають змогу користувачу створювати досить непогані зображення. Окрім того, програма має 57600 різних кольорів та елемент “піпетка”, який, при наведенні курсору на піксель на холсті, дозволяє зіставити його відтінок з переліком можливих та надалі користуватися ним, зберігши цей новий колір. Інструмент виділення прямокутної області змінює обрану групу пікселів, вільно пересуваючи її екраном.

Усі функції, які надає дана програма, працюють швидко та без затримок. Лінія, отримана інструментом пензель, не шорстка, під час швидкого руху не побачимо окремі мазки [7], текстура має однакову щільність по всьому шляху пересування курсору – отже алгоритми працюють коректно. Однак суттєвий мінус полягає у тому, що від середини сліду і до границь прозорість залишається незмінною. Під час малювання на реальному папері, чим далі поширюється фарба від місця торкання пензла, тим менш чіткий слід. Paint не надає змоги змінювати прозорість зафарбованих пікселів, отже елемент “пензл” працює майже так само, як і “олівець”.

Рисунок 1 – Інтерфейс додатку Microsoft Paint

Редактор KolourPaint

Даний редактор повністю безкоштовний та надає можливості дещо вищі, ніж Microsoft Paint. Додаток має набір базових функцій, які, перш за все, потрібні користувачу, а саме [8]:

- вибір потрібного кольору з палітри у нижній частині екрану або знаходженням відтінку на полотні;
- малювання пензлем, гумкою та олівцем;
- заповнення області одним кольором;
- створення простих фігур.

На відміну від Microsoft Paint, окрім базового функціоналу, KolourPaint також може маніпулювати прозорістю пензла, тому при створенні зображень можна використовувати олівець для контуру, та пензл для плавних переходів між кольорами, а також маніпулювання відтінками зображення (рис. 2). Крім того, KolourPaint надає можливість регулювання інтенсивності кольорів, що також покращує кінцевий результат при роботі з зображенням.

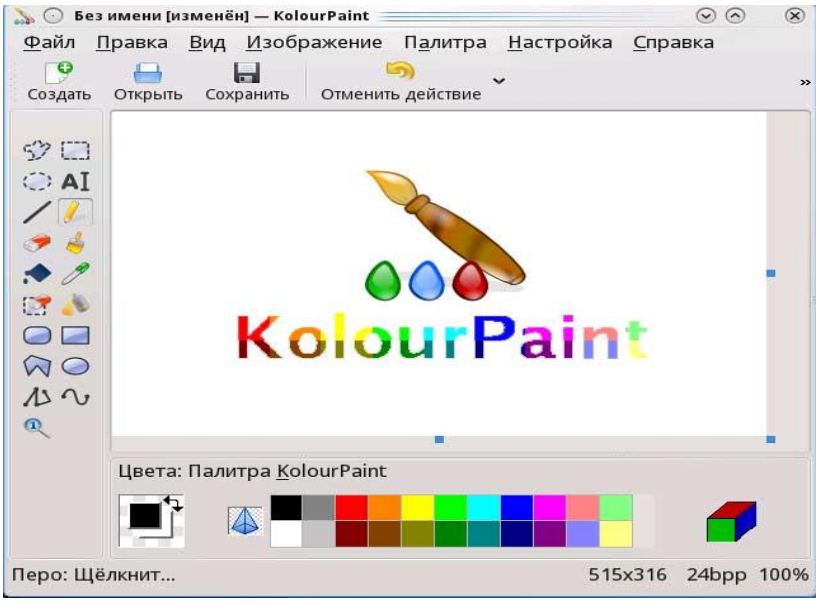

Рисунок 2 – Інтерфейс додатку KolourPaint

Недоліком програми є те, що підтримується вона тільки у UNIX-подібних операційних системах. При використанні елементу “пензл” та швидкому пересуванні курсору можна побачити “нерівності” (рис. 3), які можуть зіпсувати кінцевий результат. При підключенні графічного планшету до комп’ютера та роботі у додатку, рух пера іноді не сприймається, а реакція на тиск користувача відсутня зовсім.

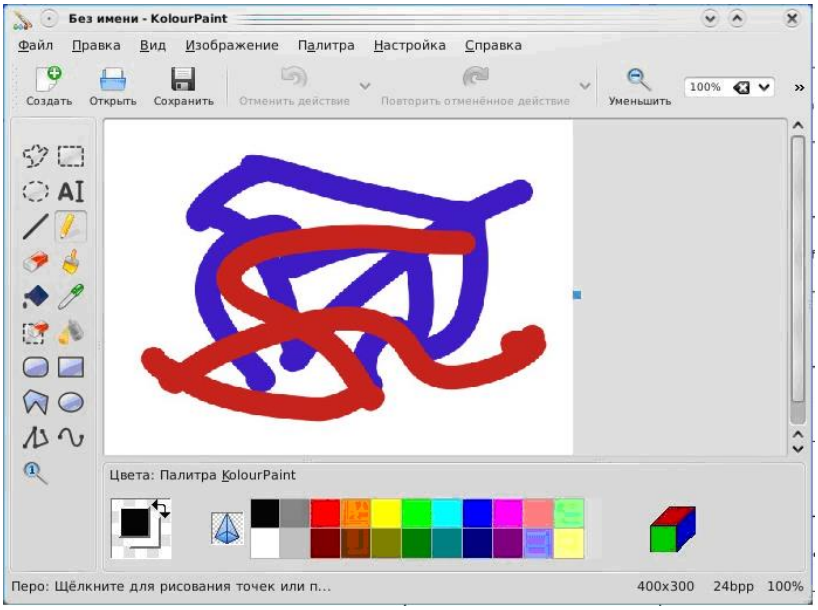

Рисунок 3 – Малювання пензлем у додатку KolourPaint

EasyPaint

Програма EasyPaint – це QT растровий редактор, який має простий та доступний інтерфейс користувача та набір функцій для малювання (рис. 4). В лівій частині екрану знаходиться панель доступу до всього функціоналу. Перевагою додатку є те, що він надає можливість працювати одразу з декількома зображеннями. Для роботи з новим зображенням спочатку потрібно за допомогою пункту меню створити новий файл. На панелі зверху можуть бути відкриті декілька файлів водночас. Таким чином, програма дозволяє копіювати зображення з файлу-чернетки до основного. У налаштуваннях в додатку можна побачити усі допустимі поєднання клавіш для швидкого виклику багатьох функцій [9].

EasyPaint може використовуватися тільки у UNIX-подібних операційних системах. Головним недоліком програми є те, що окрім інструменту “олівець” з постійною товщиною та прозорістю, інших засобів для редагування зображення там немає. Існує функція побудови кривої, але на практиці її використання займає значно більше часу, аніж використання простого пензла.

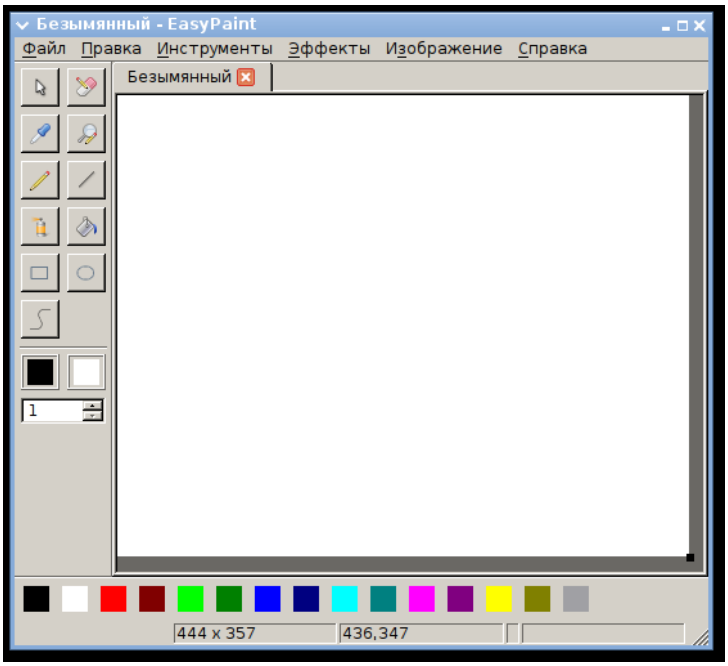

Рисунок 4 – Інтерфейс додатку EasyPaint

Редактор mtPaint

Цей редактор представляє доволі функціональний і гнучкий набір інструментів для роботи з растровою графікою (рис. 5). Займаючи близько 700 Кбайт вільного місця та не потребуючи великий об'єм системних ресурсів, mtPaint може не тільки будувати нове зображення, але й обробляти існуюче, користуючись бібліотекою ефектів.

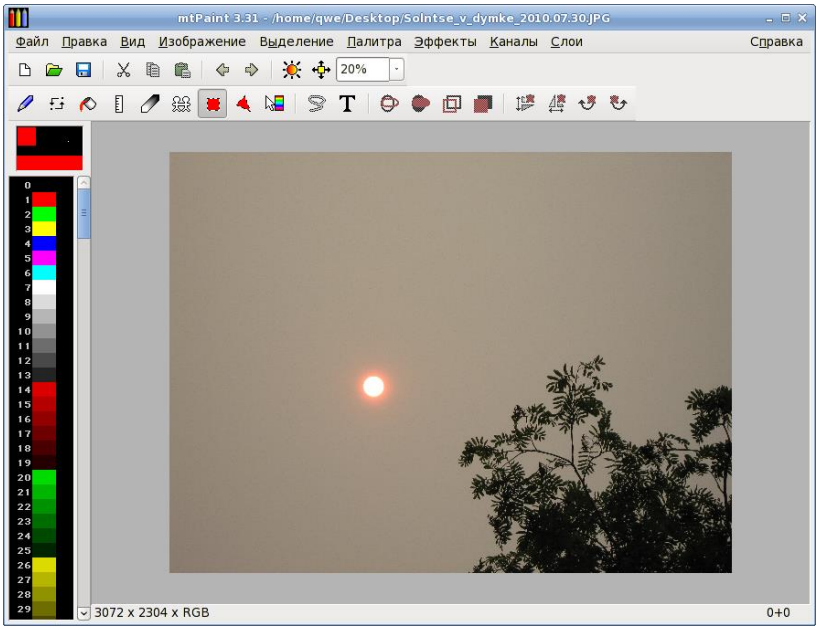

Рисунок 5 – Інтерфейс додатку mtPaint

У даному додатку відсутня система гарячих клавіш, тому процес швидкої зміни з олівця на ластик є досить незручним, адже обидва інструменти знаходяться разом з іншими, далеко один від одного. Проте у програмі є система повернення на крок вперед або назад [10]. Через це, навіть при невдалій дії можливо повернутися до попереднього стану холста. Максимальна кількість повернень сягає 1000, що є доволі суттєвою перевагою у порівнянні з іншими редакторами. mtPaint підтримує обробку подій від графічного планшету, але прозорість в додатку ніяк не відображується (рис. 6).

Головним недоліком програми є невдале розташування головних функціональних компонентів, адже при створюванні зображення до панелі

інструментів потрібно звертатися безліч разів, і така реалізація інтерфейсу користувача може бути незручною. Швидкодія додатку невисока, і обробка зображення може зайняти певний час. Крім того, на головній панелі знаходиться багато інструментів, не потрібних для нового користувача. Надлишкова інформація ускладнить використання програми.

Рисунок 6 – Малювання пензлем у додатку mtPaint

1.4 Обґрунтування теми дипломного проекту

Проведений вище аналіз вже існуючих рішень графічних редакторів дозволив визначити основні проблеми сучасних додатків, доцільність створення нових програмних засобів для створення та редагування графічних зображень, а також сформулювати основні вимоги до них.

Метою даного дипломного проєкту є розробка програмних засобів для створення та редагування графічних зображень. Також постає необхідність створення відповідних алгоритмів. Визначено, що набір таких засобів для створення, обробки та редагування зображень представляє собою графічний редактор.

Розроблювані засоби мають являти собою середовище, у котрому користувач зможе взаємодіяти з графічним полотном так, як він взаємодіє у

реальному житті з звичайним аркушем під час малювання [6]. Холст – це вся область, що доступна для редагування. За замовчуванням, користувач може змінити колір будь-якого пікселю на холсті. Для редагування обмеженої групи пікселів, зміни їх положення на екрані, а також розміру повинен бути інструмент “виділення” довільної області.

Основним стандартним засобом створення нового зображення, мають бути інструменти “пензл” та “олівець”. При роботі у додатку з мишкою або тачпадом натиск на відповідний інструмент завжди сталий та максимальний. При роботі з графічним планшетом натиск динамічно змінюється від 0 до максимуму, де нижня границя – це відривання пера від поверхні пристрою, а верхня – максимальний натиск при триманні пера під кутом 90^0 до планшету. Різниця між “пензлом” та “олівцем” в обох режимах полягає в тому, що при взаємодії із “пензелем” прозорість пікселів збільшується від центру до краю, а при взаємодії з “олівцем” залишається незмінною. Концептуально така робота кожного із інструментів правильна, бо олівець перш за все використовується для створення чіткого контуру, а пензл більш “м’який” та потрібен для створення плавних та привабливих переходів між відтінками [11].

Для видалення зафарбованих пікселів використовується “ластик”, який так само підтримує режими з мишкою або планшетом та має сталу непрозорість усього сплайну.

Окрім пензля та олівця, ще одним стандартним засобом взаємодії з графічним холстом повинна бути “заливка”. Так, просто обравши ппіксель, можна зафарбувати одним кольором усю область, в якій він знаходиться.

При роботі з багатофункціональними растровими графічними редакторами, окрім пензла та олівця, доволі часто можна зустріти інші інструменти редагування, такі як акварель або розмиваючий пензл [9]. Але усі ці засоби являють собою лише похідні від стандартних інструментів, що надаватиме розроблюваний редактор. А при зміні щільності або прозорості початкового сплайну можна отримати безліч різноманітних ефектів.

2 ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНИХ ЗАСОБІВ РОЗРОБЛЕННЯ ГРАФІЧНОГО РЕДАКТОРА

2.1 Операційна система

Перше, що потрібно обрати при написанні програмного забезпечення – це засоби, за допомогою котрих буде виконана розробка. Інструментарій обмежений мовою програмування або оболонкою, вибір яких, у свою чергу, залежить від використовуваної операційної системи.

Основною причиною, через яку більшість звичайних користувачів, яким потрібно не розроблювати новий продукт, а взаємодіяти з вже готовим, працюють у Windows, є простота використання. З ранніх версій, представлення файлової системи, а також базовий зовнішній вигляд усіх основних програм цієї операційної системи залишається майже незмінним [12]. З кожним оновленням робота користувача не ускладнюється – весь функціонал інтуїтивно зрозумілий.

Відомим фактом є те, що компанія Microsoft домінує на світовому ринку. Тому логічно припустити, що затребуваність додатку, який підтримує Windows, буде найвищою серед інших операційних систем.

Більш того, у разі написанні додатку для поточної, більш старої версії операційної системи, виникає проблема сумісності, адже якщо постане питання щодо оновлення системи – користувач навряд чи захоче встановлювати додаток заново. Windows характеризується як система з зворотною сумісністю, тому більшість програм будуть коректно працювати і з новими версіями ОС.

При використанні зовнішнього обладнання, такого як графічний планшет, графічний пензл тощо, необхідно мати версію драйверів, яка підходить до поточної операційної системи. Без драйверів подія від обладнання або буде оброблюватися, але не в повній мірі (наприклад графічний пензл без встановлених драйверів працює так само, як комп’ютерна

мишка, але не фіксує тиск користувача на поверхню або кут тримання пристрою), або зовсім не буде. Знову ж таки, домінування Microsoft на світовому ринку гарантує, що виробники не проігнорують Windows [13].

До того ж ця операційна система доволі добре розпізнає нове обладнання. Функція Plug & Play, у разі встановлення всіх необхідних драйверів, дозволяє користувачу одразу використовувати підключений пристрій у той час як інші операційні системи, частіше за все, потребують налаштування вручну.

2.2 Мова програмування

Для розробки додатку, який би мав простий інтерфейс та можливості створення растрового графічного зображення, необхідно обрати мову програмування. Кінцева програма обов'язково буде містити так званий графічний інтерфейс користувача (GUI), котрий надає доступ до основного функціоналу через візуальні елементи, такі як вікна, кнопки, панелі та віджети (репрезентативні зображення). У результаті взаємодії користувача з елементами інтерфейсу повинні виконуватись алгоритми створення растрових графічних об'єктів, а також оброблятись події від зовнішніх пристроїв.

Через знання, гнучкість, широкий спектр застосування, популярність та досить велику кількість потужних бібліотек та фреймворків, орієнтованих на взаємодію з растровими графічними об'єктами, вирішено обрати мовою для написання додатку Python.

Python – це об'єктно-орієнтована, інтерпретована та високорівнева мова програмування, яка має строгу динамічну семантику. Існують п'ять парадигм програмування, які підтримуються у Python: імперативна, структурна, об'єктно-орієнтовна, функціональна та аспектно-орієнтовна. Кожне значення в Python являє собою об'єкт. Типи даних усіх цих значень – це класи. Змінні, у свою чергу, є екземплярами відповідних класів [14].

Перевагами цієї мови є:

- простота та швидкість, через що значно спрощується програмування. Мова чудово підходить для написання сценаріїв.

- продуктивність – новому користувачу навчатися досить просто, оскільки є весь необхідний набір методів для пришвидшення процесу розробки, що також сприяє підвищенню продуктивності. Майже усі базові потреби користувач може втілити, користуючись функціями з бібліотеки за замовчуванням.

- доволі легкий та простий синтаксис покращує читабельність та зрозумілість коду, що в результаті зменшує додаткові витрати на обслуговування програмного забезпечення. Програма, написана на Python, компілюється порядково, через що процес налагодження помилок є доволі простим та ефективним. У разі пошкодження програмного забезпечення ініціювання помилки сегментації у додатку не відбувається;

- портативність та кросплатформність – Python це кросплатформна мова. На сьогоднішній день її інтерпретатори можна знайти майже для будь-якої операційної системи. Сам інтерпретатор та інші ресурси, такі як бібліотеки, доступні як у вихідній формі, так і у скомпільованій.

- відкритий код – Python розроблено з урахуванням OSL ліцензії (Open source licence). Таким чином, навіть у комерційних цілях, код можна використовувати абсолютно безкоштовно. Так само Python можна як завгодно модифікувати, включати до свого додатку та потім повторно розповсюджувати. Завантажити Python можливо з офіційного веб-сайту;

- бібліотека управління пам’яттю містить приватну множину об’єктів та структур даних Python, а також вбудований менеджер пам’яті для обробки даних;

- універсальність – через значну кількість бібліотек та фреймворків Python можна назвати найбільш універсальною мовою програмування. Не дивлячись на те, що будь-яке питання може бути вирішено різними

способами, і деякі з них покращують кінцевий результат, Python завжди виконує задачу коректно.

Проте, мова програмування Python дещо обмежена з точки зору безпеки та продуктивності, що спричиняє ряд недоліків:

- швидкість виконання операцій відносно менша ніж у таких мовах, як C, C++ та Java через використання інтерпретатора замість компілятора;

- велике споживання пам’яті, оскільки структури даних мови програмування Python займають доволі багато вільного місця у пам’яті. Через це мова не підходить для розробки додатків з обмеженням пам’яті;

- оскільки Python виконується не через компілятор, а через інтерпретатор, розробник не може знайти певні недоліки та помилки під час компіляції. Виявити їх можливо лише під час роботи, що спричиняє великі труднощі при тестуванні.

Кожен раз читати документації відповідної мови, для того щоб вирішити, які засоби існують для вирішення поточного питання займає багато часу що не є ефективнтим. Більш того, доступ до деяких засобів можливий тільки у випадку використання сторонніх бібліотек. Існує набір пакетів, згрупованих для полегшення розробки програмного забезпечення. Фреймворк – це середовище з набором інструментів для вирішення специфічних проблем та комфортного створення або редагування різних додатків.

Однією з областей, в якій на сьогоднішній день широко використовуються мова програмування Python, є розробка програм на основі графічного інтерфейсу. Такі інструменти, як PyQt, PyGame, PyGtr, PyTrinker та ін., дозволяють розробникам маніпулювати GUI та створювати власні додатки.

2.3 Фреймворк для розробки додатку

Фреймворк PyQt

Фреймворк PyQt – це набір інструментів та бібліотек C++, якими розробник може користуватися при проектуванні нового програмного забезпечення майже для будь-яких існуючих платформ (Windows, Linux, MacOS, Android та ін.). Розробник RiverBank презентував прив'язку мови програмування Python до середовища Qt.

Існують дві версії PyQt. Перша побудована на основі версій Qt4 та Qt5, друга – тільки на Qt5. Остання має більш широкий функціонал, що надає такі можливості:

-розробка графічних інтерфейсів користувачів за допомогою відповідних незалежних абстракцій (Qt, QtGui);

- взаємодія з базами данних SQL, SVG, OpenGL та XML (QtSql);

- робота з регулярними виразами (QtRegExp);

- організація потоків (QtThread);

- відслідковування мережевої взаємодії (QtNetwork);

- перегляд веб-сторінок (QtMulimedia).

PyQT – дуже потужний багатофункціональний фреймворк. Компанія Qt розробила свій власний аналог з прив'язкою до мови Pyhton. Офіціальний пакет Python для Qt має назву PySide2. Оскільки обидва фреймворки побудовані на Qt, вони мають майже однаковий інтерфейс без особливих відмінностей. Існує можливість перенесення коду від одного середовища до іншого без суттєвих змін. На офіційних сайтах розробників можна знайти повну документацію, яка уможливлює вивчення усіх особливостей фрейморку з мінімальними зусиллями [17].

Не дивлячись на те, що PyQt реалізує декілька тисяч класів та модулів Python та Qt v5, що підтверджує універсальність фреймворку, доступ до нього

можливий тільки із урахуванням приватної ліцензії розробника (Riverbank Commercial License) або стандартної (General Public Licence v3).

Виникає потреба в сумісності ліцензії розроблюваного продукту із ліцензією PyQt. Тому при створенні комерційних додатків обов'язково потрібно мати відповідну RCL ліцензію.

Основним класом при розробці графічного інтерфейсу користувача в PyQt є QWidget. Це стандартний клас, за допомогою якого розробник може створювати усі компоненти інтерфейсу. За замовчуванням, усі складові прямокутної форми, з параметром, який відповідає за розміщення їх у головному вікні. Однак при зміні певних методів або атрибутів можна змінити стандартний вигляд компонент або їх функціонал. Обробка подій виконується від будь-яких зовнішніх пристроїв і, якщо віджет фіксує подію натискання, він змінює свій початковий стан та надає інформацію про виконану зміну [18]. Колекція віджетів у PyQt досить велика, але найбільш розповсюдженими компонентами є:

- кнопки – компоненти інтерфейсу, які потрібні для подання вказівок програмі, виконання різноманітних команд, а також для дозволу на зміну певних компонент користувачем;
- ярлики – компоненти інтерфейсу, які містять посилання на запитувану користувачем інформацію;
- рядкове поле – однорядкове текстове поле, з котрим може взаємодіяти користувач. Актуально, коли є необхідність редагування даних у вигляді тексту;
- поля із списком – віджет, за допомогою якого розробник може надати користувачу багато необхідної інформації, у той час, як сама інформація повинна займати як можна менше місця на головному екрані (список, що розгортається);

- перемикач – кнопка вибору, розповсюджена у графічних додатках. За замовчуванням, може бути навіть ввімкненою і потрібна для надання вибору користувачу.

Фреймворк wxPython

wxPython – це безкоштовний кросплатформний фреймворк, в якому за основу взято популярну міжплатформну графічну бібліотеку wxWidgets, що написана на C++ [19]. Ця оболонка використовується програмістами, перш за все, для розробки графічного інтерфейсу користувача мовою Python. Він застосовується, коли постає питання створення додатку з високо функціональним та надійним графічним інтерфейсом. Python являє собою той інструмент, що згортає всі модулі бібліотеки wxWidgets, тому написані додатки досить зрозумілі та прості.

На відміну від PyQt, wxPython має відкритий вихідний код, доступний на офіціальній сторінці виробника. Кожен програміст може користуватися кодом безкоштовно, модифікувати його та викладати у мережу для подальшого перегляду та внесення змін.

Стиль віджетів, що використовуються при розробці додатку, однаковий на будь-якій платформі. На даний момент, у wxPython є доступ до розробки додатків на таких платформах як Windows, Linux та Mac OS X.

Головна бібліотека містить велику кількість засобів для створення необхідних компонентів графічного інтерфейсу та їх вказівок. Розробник створив багато різноманітних віджетів. Однак, окрім цього, існує механізм створення власного елементу керування, що відповідає індивідуальним потребам користувача, для подальшого використання у розроблюваному додатку. Оскільки wxWidgets написано на C++, можливо просто отримати власний клас з модулів Control, Window тощо.

На офіційному сайті даного фреймворку можна знайти докладну документацію відносно засобів програмування, використання, особливостей

бібліотеки wxWidgets та поширених помилок про базові моделі розробки елементів GUI [20].

Мають місце декілька недоліків, пов’язаних із швидкодією даного фреймворку, однак нові модулі ядра wxPython не викликають ніяких порушень при роботі.

Ще однією особливістю wxPython є реалізація MVC (Module-View-Controller). MVC відповідає за управління приватними та клієнтськими даними програми (API) та забезпечує зовнішню взаємодію з користувачем за допомогою інтерфейсу (GUI). На відміну від PyQt v5, основний модуль wx.lib.pubsub підтримується тільки для зворотної сумісності. Саме тому іноді можливе порушення цілісності даних програми.

При розробці GUI можна побачити, що майже усі компоненти, які може переглядати користувач у готовому додатку, мають індивідуальну систему налаштувань. MVC вимагає, щоб дані користувача зберігалися на верхньому (модельному) рівні. Через це інші дані, які зберігаються нижче (для відображення або зміни ознаки), мають бути копією основних. Після зміни оригіналу дані нижчих рівнів не будуть точною копією. У разі незнання або необережного ставлення, можлива поява помилок при подальшій розробці.

Фреймворк PyTrinket

Не дивлячись на те, що для мови програмування Python написано дуже багато різноманітних фреймворків для роботи з графічним інтерфейсом, існує бібліотека, яка надає не менш широкий простір для створення GUI додатків. PyTkinter повністю відрізняється від інших – він є вбудованим фреймворком Python, що знаходиться в стандартній бібліотеці. Це одна з значних переваг PyTkinter над PyQt, оскільки перший має більшу швидкодію через розташування всіх необхідних модулів у вбудованій бібліотеці Python [21].

Як і в wxPython, основні елементи інтерфейсу відображуються за допомогою власних компонентів операційної системи, тому всі додатки, що

написані на PyTkinter, коректно відображаються на тій платформі, на котрій запущені.

За допомогою пакету Tkinter, а також стандартних розширень tkinter.tik та tkinter.ttk програміст отримує повний доступ до широкого набору інструментів для роботи з вікнами. Для вивчення PyTkinter достатньо прямого звернення до Tk/Tcl документацій.

Класи Tk та Tcl призначені не для створення нових екземплярів явно, адже середовище створює іх автоматично після появи нового екземпляру окремого класу. Їх функція полягає лише в управлінні стандартними функціями у доменному просторі.

Віджет в Tkinter являє собою клас, в котрому знаходяться необхідні параметри, які вказує розробник, та інструкції, які виконуються після підтвердження взаємодії. З розширеннями, PyTkinter має більш широкий набір віджетів порівняно з фреймворком PyQt (рис. 7).

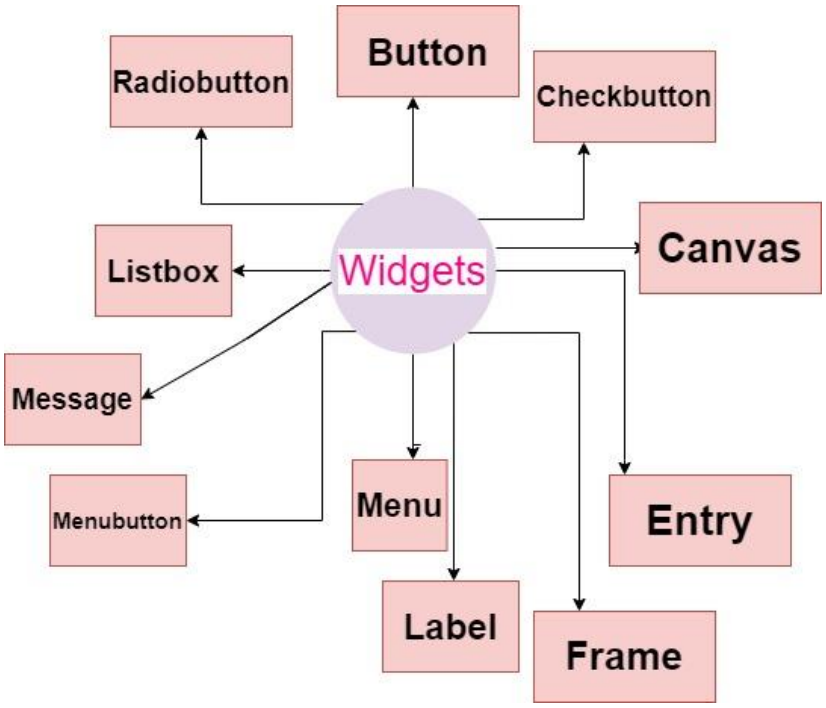

Рисунок 7 – Стандартні віджети, що доступні у PyTkinter

Особливістю фреймворку PyTkinter є також наявність трьох менеджерів геометрії:

- Використовуючи абсолютне позиціювання, програміст задає розмір та величину компонентів, що залишаються незмінними при збільшенні або зменшенні головного вікна. Багато сучасних GUI фреймворків здібні коректно працювати на різних платформах. При розробці графічних інтерфейсів для додатків, програмісти часто розміщують компоненти програмного забезпечення “абсолютно”, тобто відносно однієї точки відліку. Але при запуску додатку на іншій платформі, кожен раз виникає проблема з розташуванням віджетів. У приватних додатках це може призвести до повної зіпсованості інтерфейсу. Для абсолютного розміщення візуальних компонентів у додатку PyTkinter використовується менеджер геометрії place.

- Іншим варіантом розміщення віджетів у вікні є позиціювання за вертикаллю та горизонталлю. Кожний з компонентів при такому розміщенні обов’язково заповнює частину області, в якій знаходиться, згідно вказівок розробника. Для створення панелі з набором віджетів необхідно встановити усі компоненти в рамку (яка знаходиться в стандартній бібліотеці у класі Frame). Так само для складніших розміток використовується декілька рамок. Щоб розмістити візуальні компоненти по нормалям у додатку PyTkinter, використовується менеджер геометрії pack;

- Хоча за допомогою методу pack можливо отримати складний інтерфейс додатку, розробник повинен у такому разі використовувати багато фреймів, запам’ятовуючи позицію та розмір кожного, а також параметри компонент. Даний підхід не є ефективним, оскільки при потребі швидкої перестановки віджетів, програмісту буде необхідно змінювати позицію кожного прилеглого фрейму вручну. Менеджер геометрії grid дозволяє розміщувати віджети у контейнері, що візуально можна порівняти з таблицею. Такий спосіб більш зручний та гнучкий.

Оскільки PyTkinter є вбудованим у мову програмування фреймворком, для знаходження необхідної інформації розробник може звернутися до документації (підрозділу Tk стандартної бібліотеки Python) на офіційному сайті [22].

Фреймворк PyGame

PyGame – це безкоштовний та популярний кросплатформний фреймворк, написаний поверх бібліотеки SDL, що складається з модулів Python та дозволяє створювати різні мультимедійні додатки [23].

Незважаючи на те, що PyGame не володіє таким широким спектром можливостей як PyTkinter, даний фреймворк є дуже корисним при написанні саме графічних додатків. Велика частина модулів орієнтована на забезпечення комфортної взаємодії користувача з програмним забезпеченням (рис. 8).

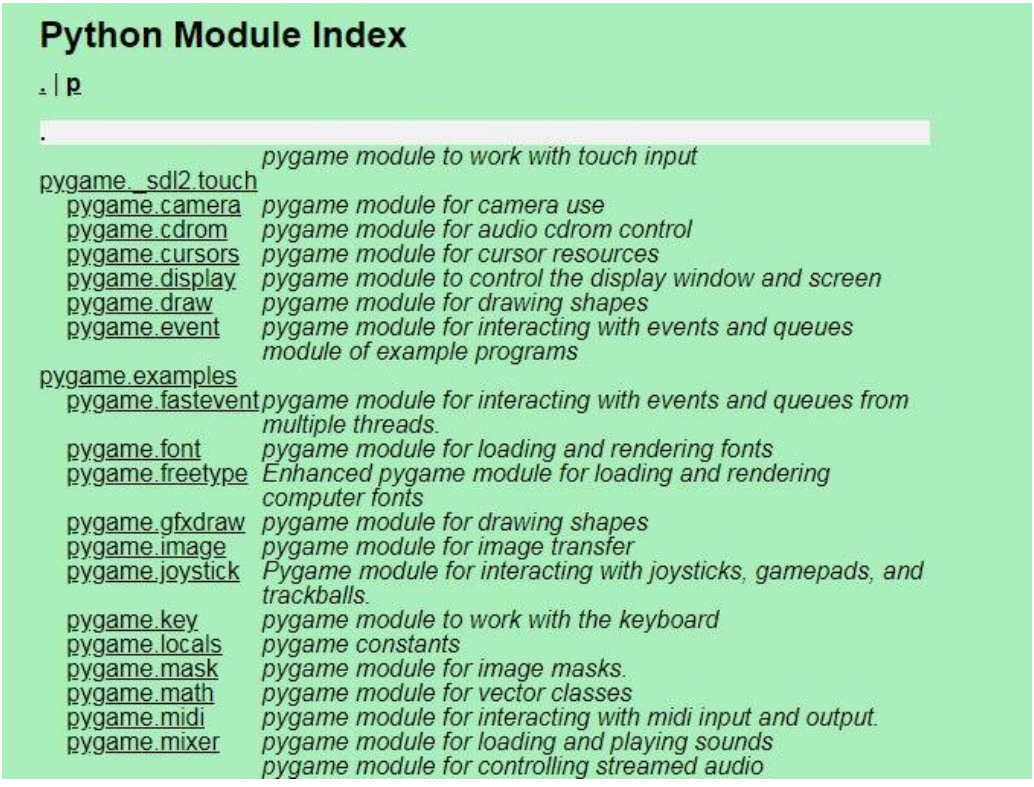

Рисунок 8 - Основні модулі PyGame

Серед складових фреймворку PyGame окремо потрібно виділити модуль tablet. Компонент був розроблений LCS Telegraphics у вигляді оболонки wintab, котра підтримується планшетами Wacom та відповідає їх стандартам [24]. Даний модуль дозволяє відслідковувати події від графічного планшета та графічного пензла. Рухаючи пензл, у кожен момент часу запитується копія повернутих пакетів. Якщо перо знаходиться на великій відстані від поверхні, повернеться “None”, оскільки ніяких подій не повинно відбуватися. Інакше, інформація повертається, і серед усіх пакетних даних ізолюється останній пакет у черзі. Об’єкт brushPressure вбудованого класу Tablet повертає інформацію про рівень натискання користувача на поверхню (pressure), а також абсолютне положення курсору відносно лівого верхнього кута (xpos та ypos).

Основні компоненти графічного інтерфейсу користувача, а також набір базових вказівок, таких як pygame.draw, pygame.sprite, pygame.surfarray та ін. мають оптимізовані вбудовані алгоритми, написані такими мовами програмування як асемблер та C. Хоча інші фреймворки також використовують Python для C-бібліотек, вказівки розробнику все ж таки придется писати саме на Pyhton. Такий код при збірці буде працювати у сотні разів повільніше. У PyGame не потрібно встановлювати додаткові розширення, навіть з ctypes типами. Для динамічних елементів відмальовування можна використовувати швидкі вбудовані функції. Це не означає, що PyGame містить вже готові інструменти для динамічної взаємодії з растровим зображенням, але, будуючи власний алгоритм розробника на основі вбудованих, можна суттєво покращити швидкодію. Крім того, вбудовані алгоритми відмальовування суттєво спрощують кінцевий код. У разі необхідності, модулі можна ініціалізувати та використовувати окремо [25].

На офіційному сайті PyGame можна зайти усі необхідну документацію для подальшого зручного використання фреймворку.

2.4 Середовище для розробки додатку

В результаті аналізу існуючих інструментів для створення графічного інтерфейсу користувача та різних фреймворків, в якості середовища для розробки програмного продукту обрано PyCharm. PyCharm надає повну підтримку розробки за допомогою мови програмування Python майже на усіх існуючих платформах. PyCharm надає велику кількість можливостей для тестування додатків [26]. Використовуючи вбудовані модулі та стандартні пакети, написання програми значно спрощується, а ефективність значно збільшується. Декілька списків налаштувань роблять можливим редагування середовища для задоволення необхідних функціональних вимог або власних уподобань програміста. Інтерфейс середовища розробки PyCharm наведено на рис.10.

Переваги PyCharm:

- автодоповнення коду (краще заповнення шаблонів, ніж у багатьох подібних середовищах);
- існує можливість відлагодження або тестування програми;
- впровадження системи взаємодії з багатьма базами даних;
- робота у середовищі на іншому комп’ютері та зміни коду;
- поточне виправлення усіх програмних рішень;
- shell вбудований в середовище;

Недоліки:

- Більшість функцій недоступна в безкоштовній версії;
- іноді виникають проблеми при відлагодженні.

3. АЛГОРИТМИ ТА ІНСТРУМЕНТИ РЕДАГУВАННЯ

3.1 RGB модель

Для отримання переважної більшості відтінків кольорів світлового спектру на випромінюючих пристроях використовуються адитивна кольорова модель RGB. Відтінки складаються зі змішаних основних компонент моделі (червоного, зеленого та синього кольорів).

У випадку, коли усі три джерела мають однакову інтенсивність випромінювання світла, можливо проаналізувати існуючий набір відтінків у тривимірній RGB системі координат з масштабними коефіцієнтами. Чорному кольору відповідає початок координат (0,0,0), білому – протилежна точка (1,1,1), а відтінки сірого належать основній діагоналі. Результуючий колір, котрий користувач може отримати шляхом змішування базових кольорів, буде вектором з координатами (i, j, k) у розглянутій системі (рис. 9). При зміні співвідношення буде змінюватися і кінцевий колір [27].

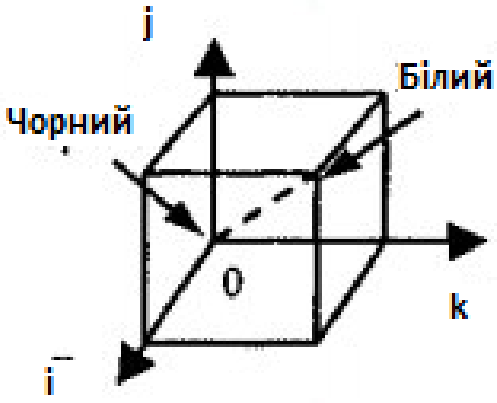

Рисунок 9 – Система координат RGB

Тобто:

result_color = iR + jG + kB (1)

Якщо кожен коефіцієнт інтенсивності розписати через суму інших, то отримаємо:

$$i' = \frac{i}{i+j+k},\ j' = \frac{j}{i+j+k},\ k' = \frac{k}{i+j+k} \quad (2)$$

Після перетворення (1), було отримано рівняння для одиничної площини, замість тривимірної системи:

$$\text{result\_color} = i' R + j' G + k' B \quad (3)$$

Дана площина також має назву “трикутник Максвела” (рис. 10).

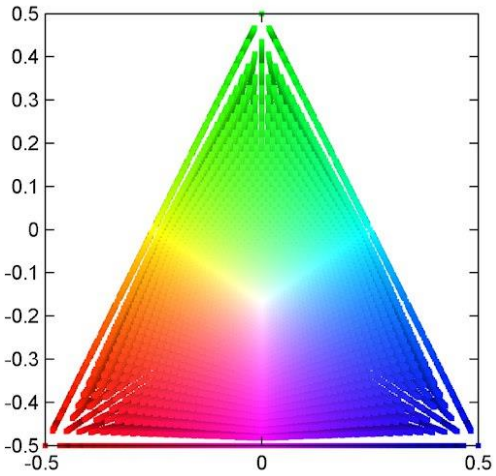

Рисунок 10 – Трикутник Максвела

Оскільки головні кольори знаходяться у кутах площини, модель забезпечує комфортну роботу користувача.

Для обробки кольорових зображень постає питання у кодуванні кожного кольору RGB моделі. Для цього усі три компоненти набувають 256 рівнів інтенсивності. Розмір, який вони займають, дорівнює трьом байтам. Кількість відтінків при такій градації дорівнює $256^3 = 16{,}7$ млн. Представлення кожного кольору в програмі буде у вигляді бінарного числа, що складається з трьох октетів. Кожен октет відповідає за рівень інтенсивності відповідного спектру. Приклад кодування чорного відтінку:

result_color = 00000000 00000000 00000000.

3.2 Алгоритм функціонування інструменту “графічний пензель”

Під час роботи у растрових графічних редакторах найбільш важливим для користувачів засобом для створення та редагування зображень є

інструмент малювання довільних кривих. Можливо досягти багатьох варіацій цього компоненту через те, що він не має прив’язки до певної форми відбитку. Під час взаємодії з інструментом можливе статичне або динамічне змінення товщини або прозорості лінії. Лінія складається з множини відбитків, тип та колір яких було задано на початку роботи користувачем. Саме через використовуваність графічного пензла, потрібно сформувати алгоритм, котрий буде мати досить високу швидкодію, не заважати роботі додатку в інтерактивному режимі, раціонально споживати пам’ять робочого пристрою та коректно виконувати необхідну клієнту задачу [27].

Узагальнений алгоритм функціонування даного засобу складається з:

- зчитування відбитків руху графічного пензла кожні t мілісекунд;
- встановлення усередненої траєкторії руху графічного пензла, використовуючи набір міток, отриманих з попереднього руху, або встановлення послідовної неперервної групи кривих (сплайнів) між кожними n мітками початкового набору;
- заповнення усієї траєкторії або окремих частин об’єктами, які являють собою стиль обраного інструменту (олівець, акварель, розпилювач тощо).

Малювання растрового відбитку

В деяких графічних редакторах існує власна бібліотека стилів пензлу, яка надає користувачу широкі можливості під час роботи у додатку. Однак кожен стиль таких непрозорих кривих відрізняється тільки траєкторією промальовування об’єктів, а також алгоритмами побудови власне головної траєкторії (рис. 11).

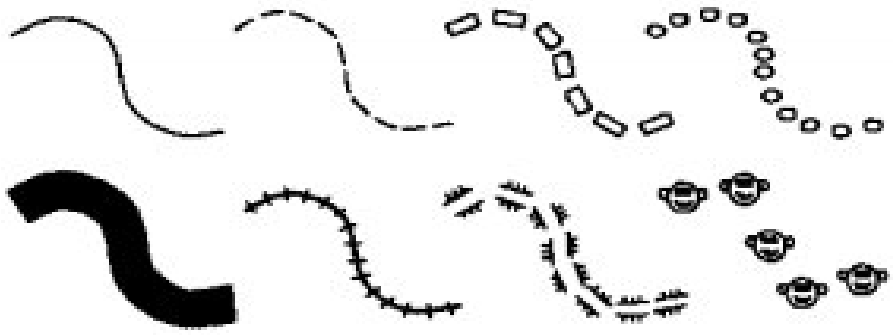

Рисунок 11 – Стилі ліній при однаковій траєкторії

Стилізацію кривих можна назвати штампуванням форми растрового графічного об’єкту упродовж всієї траєкторії лінії (рис 12, а). Об’єкт, у свою чергу, є простим зображенням або текстурою. Візуалізація у PyGame зображення кожного компоненту на дисплеї виконується за допомогою вбудованого методу draw(). Відстань між кожними двома мазками взято як 25% від діаметру окружності, описаної навколо мазку, для запобігання занадто великих відступів (рис. 12, б). Дана модель заповнення кривої набором графічних примітивів є дуже простою у впровадженні, має високу швидкодію, але не може правильно намалювати лінії з коефіцієнтом непрозорості менше одиниці, оскільки прозорі штрихи, перекриваючись, стають більш інтенсивними (рис. 12, в).

Рисунок 12 – Побудова довільної кривої:

а) – нормальне штампування упродовж усієї траєкторії;

б) – штампування з великими відступами; в) – перекриття прозорих відбитків під час штампування

Для стандартного округлого мазку динамічним пензлем, кількість пікселів, котрі потрібно обробити, буде дорівнювати:

$$s = \sum_{1}^{n} R^2 F(i) \quad (4)$$

де n – кількість обробок руху графічного пера, R – максимальне значення пензла, визначене користувачем, і – індекс обробки події від планшету, F(i) – значення тиску користувача на перо під час обробки і-ої події [28].

Додати налаштування прозорості пензля під час редагування у растрі можливо, якщо промальовування окремих перекриваючих об'єктів замінити на зв'язані геометричні фігури. Такі елементи складаються з двох горизонтальних або вертикальних та двох похилих ліній, в залежності від попереднього побудованого об'єкту (рис.17). При зміні нахилу початкової траєкторії, будова окремого відрізку продовжиться до тих пір, поки кут між дотичною до кривої у місці поточної обробки події від зовнішнього пристрою та частиною лінії, що будується, не стане більше певного α, після чого почнеться побудова нового відрізку апроксимуючої ламаної. Даний алгоритм уникає накладання частин різних складових об'єктів кривої, а також використовує алгоритми промальовування вертикальних та горизонтальних відрізків, що значно збільшує швидкість виведення. Такий алгоритм можна використовувати для малювання прозорих кривих. Величина проценту збігання ламаної до початкової лінії буде тим більша, чим менший пороговий кут побудови ділянок об'єктів.

Навіть при значному зменшенні граничного кута α, після побудови кривої можна побачити характерні злами (рис. 13). Оскільки ламана складається з однакових за формою фігур, місця дотику неможливо заповнити.

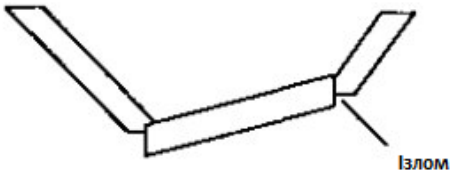

Рисунок 13 – Дефекти при побудові ламаної

Даний недолік було усунено шляхом перетворення попереднього алгоритму шляхом заповнення вільних місць полігонами (рис. 14, а). Для видалення зайвих частин граней, найбільш простим за своєю реалізацію та найбільш швидким є алгоритм z-буферизації. Кожному пікселю з окремими координатами відведено його відповідну нормаль до растрової площини за напрямком побудови глибини (тобто буферу z_start). Після визначення масиву глибин, грань конвертувалася в своє растрове уявлення на площині, і для кожного складового пікселю визначався елемент масиву, тобто глибина. Якщо вона менше початкової глибини z-буферу, піксель відображався в растрі, та його глибина переносилася в z-буфер.

Даний алгоритм однаково коректно обробляє лінію з будь-яким коефіцієнтом непрозорості. Для досить малих значеннях α, можна сказати, що (4):

$$\lim_{A \longrightarrow 0} \frac{f(A)}{L} = 1; \qquad (5)$$

Недоліком даного алгоритму є низька швидкодія, оскільки операція заповнення полігону займає значно більше часу, а ні ж виведення відрізків, але вона все ж таки дозволяє працювати користувачу з додатком в інтерактивному режимі. Більш того, під час швидкого пересування графічного пера при великих значеннях за замовчуванням α, у разі невисокої швидкості обробки подій від зовнішнього обладнання, можлива поява ефекту ламаної лінії, замість округлої (рис.14, б).

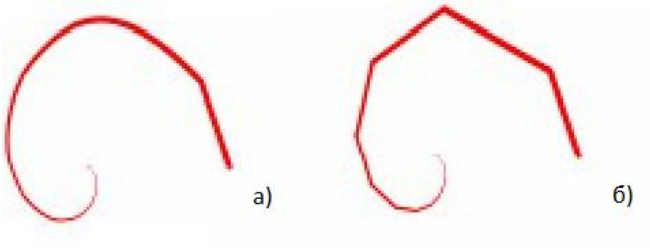

Рисунок 14 – Заповнення лінії полігонами:

а) – середня швидкість побудови; б) – швидка побудова з ізломами

Побудова траєкторії

Для візуалізації складових елементів кісті під час малювання важливим є також побудова траєкторії руху. Відображення кожного окремого мазку тільки під час обробки подій від графічного пристрою потребує багато пам’яті, а також містить певну хаотичність та “тремтіння” проміжків лінії. Для усунення даних недоліків та створення вектору руху доцільно використовувати алгоритм побудови апроксимуючої кривої, яка б усереднювала зміщення графічного пера користувача по растру.

Замість того, щоб обробляти усю ламану, вона розбивається на менші, більш прості множини кривих (сплайни), які підпорядковувались однаковій процедурі. Будова наступного сплайну, таким чином, пов’язувалась з координатами попереднього, що запобігало появі “стрибків” або зміщень пікселів при продовженні руху графічного пера. Для побудови сплайну вирішено використати кубічний алгоритм кривої Без’є [29].

Без’є описується функцією залежності координат від параметру “a”, а також від кількості опорних точок проміжної кривої: для m = 3 кількість опорних точок дорівнює 4 (рис.15). Координати опорних точок являють собою місця відгуку на подію, згенеровану зовнішнім керуючим пристроєм.

Під час руху пера такі криві, накладаючись, утворювали неперервну згладжену траєкторію растеризації мазку.

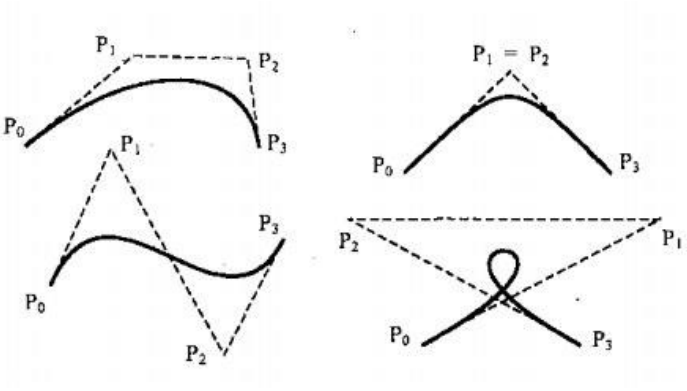

Рисунок 15 – Кубічні криві Без’є

Оскільки сплайни можна моделювати дискретно, вони дозволяють досить просто підрахувати візуальні зіткнення, петлі та вигони. Для отримання якісного сліду від пензла, початкову криву розбито на велику кількість дрібних сегментів. У такому випадку, результуюча форма є не тільки гладкою, але і без дефектів.

Даний алгоритм є оптимальним при побудові апроксимуючої до початкової кривої. Попередні обчислення мають середню складність. Кількість пікселів, оброблених даним алгоритмом, буде дорівнювати добутку кількості точок апроксимації до кількості оброблень пікселів для кожного сплайну.

3.3 Алгоритм функціонування інструменту “зафарбовування”

Інструмент зафарбовування довільних областей є невід’ємним компонентом при редагуванні майже кожного графічного зображення. При натисканні користувачем на довільний піксель растру, автоматично зафарбовується область навколо початкового пікселю до тих пір, поки не дійде границь об’єкту або всього холсту [31].

Найпростіший алгоритм роботи змінює колір обраного пікселю на встановлений користувачем, після чого проводить аналіз сусідніх пікселів

(рис.16). Для редагування групи пікселів було обрано чотирьохзв’язний алгоритм замість восьмизв’язного, оскільки він має більшу швидкодію, хоча і не може пройти крізь діагональні піксельні лінії. Функція paintbucket зафарбовування викликає себе під час роботи, тобто є рекурсивною. Повернення на верхній рівень рекурсії виконується тільки тоді, коли усі граничні пікселі до поточного вже зафарбовані або якщо їх колір дорівнює кольору границі об’єкту [30]. Під час тестування даного алгоритму виявлено, що через значний розмір початкових редагуємих зображень, рекурсивний алгоритм викликав переповнення стеку. У дипломному проєкті цю проблему вирішено після взяття в якості буферу даних масив line_buff.

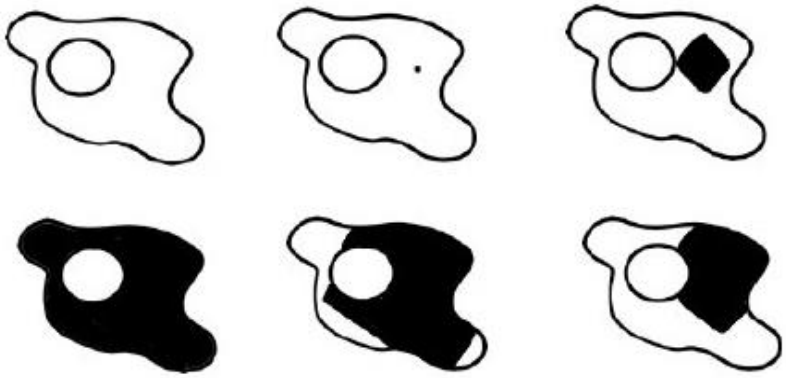

Рисунок 16 – Порядок зафарбовування довільної області рекурсивним алгоритмом

Для вдосконалення початкового алгоритму з використанням рекурсії, застосовано рекурсивне зафарбовування горизонтальної лінії замість пікселю. На початку роботи, ряд, якому належить піксель зафарбовування, змінював свій колір (рис.17). Далі функція викликалась для усієї лінії. Глибина рекурсії залежала напряму від довжини лінії. Даний факт означає, що для деяких об’єктів із значною довжиною об’єм займаного місця у пам’яті комп’ютера може збільшитися.

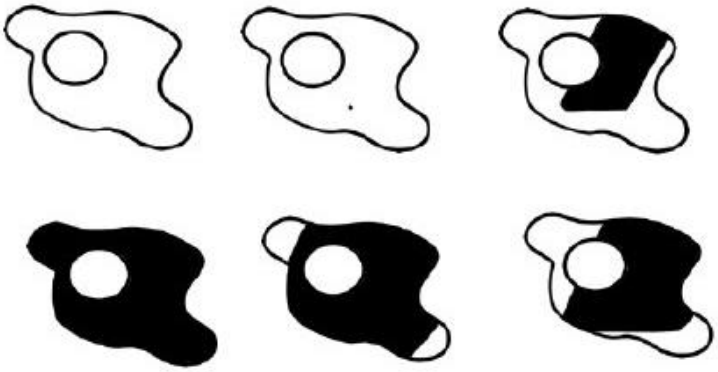

Рисунок 17 – Порядок зафарбовування довільної області модифікованим рекурсивним алгоритмом

4. ОПИС РОЗРОБЛЕНОГО РЕДАКТОРА

4.1 Інтерфейс графічного редактора

Інтерфейс додатку представлено у файлі MainRasterEditor. В головному циклі drawScene в режимі реального часу перевіряється наявність активованих компонентів редактора, а також відображається зміни поточного кольору засобів редагування. Вікно додатку заповнюється панелями tools, options, highlighttoolbox, colours, кнопками clear, load та save, а також основним растром, в якому буде виконуватись редагування певного графічного зображення (рис. 18). Кожна з панелей являє собою графічний примітив та візуалізується за допомогою вбудованого методу draw().

Дані про розташування кожного з віджетів зберігаються у списках LeftToolBarY та RightToolBarY. При обробці програмою, кожен з елементів розташовується в залежності від ширини вікна. Точне розташування визначається у списку SizeManyBarY, який являє собою набір координат пар компонент (тобто кожного лівого та правого віджету на панелі).

Візуалізація елементів load/save виконується у функції defaultsaveloadbox. Об’єкти loadbox та savebox слугують для встановлення кнопок зберігання або завантаження растрових файлів, але не містять ніякої прив’язки до вказівок. Кожен момент часу перевіряється успішність виконання функцій checksave та checkload. Під час взаємодії з областю, на якій знаходяться дані компоненти, виконується відповідна функція loadimage або saveimage, після чого програма повертається до зображень цих елементів, вказаних за замовчуванням. Clearbox слугує для очищення растру. Після активації даного компоненту виконується метод draw початкового вікна растру.

Панель з highlights має функцію швидкого доступу до основних кольорів, з котрими в даний момент працює користувач. У загальному циклі за замовчуванням активний верхній highlight. Інформація про це вказана у модулі

checkoptionswitch (optionswitch встановлено в 1). Його відтінок співпадає з початковим кольором пензля користувача. Нижній другорядний highlight має відтінок головного вікна додатку. Для коректного функціонування інших алгоритмів, в colourswitch записується інформація, в залежності від активованого головного або другорядного кольору. Зміна основного кольору виконується за допомогою функції CIrect(). Зчитування поточного кольору, встановленого у highlighttoolbox виконується за допомогою функцій selectcolour1 та selectcolour2.

Панель RGB являє собою прямокутну область з палітрою кольорів. Відтінки на панелі описані в файлі colours.py. Алгоритм замальовування набору пікселів панелі для надання максимально-можливого спектру відтінків реалізовано в функції checkcolour(). Побудова панелі виконується у функції defaultcolorbox.

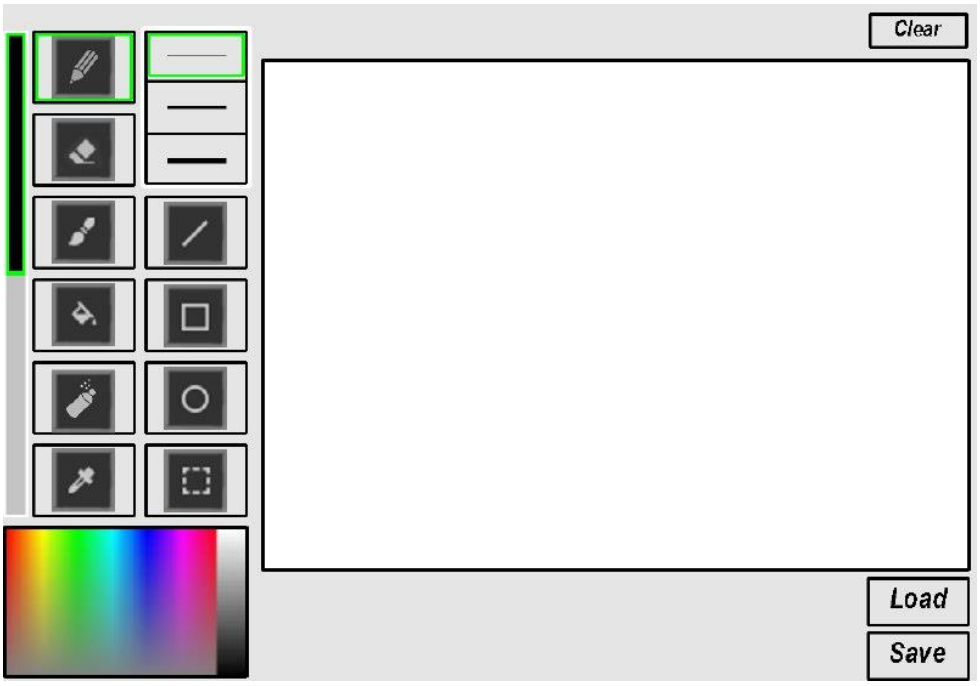

Рисунок 18 – Інтерфейс розробленого додатку

Користувачу надається можливість обрати необхідні засоби для обробки графічних об’єктів. Під час активації відповідний елемент буде виділено

(рис. 19). Наявність активації перевіряється у функції picktool, після чого у highlightoptions виконується побудова рамки активації. У випадку використання графічного планшету, достатньо торкнутися пером області поверхні, яка відповідає розміщенню інструмента у вікні додатка. Активація доступна тільки в області панелі інструментів, через використання вбудованого методу set_clip тільки для SizeManyBarY, а не для всього вікна.

Рисунок 19 – Позначення активації обраного користувачем компоненту

Деякі графічні засоби редагування доцільно змінювати під час роботи з зображенням. Тому можливе налаштування інструментів “пензель”, “олівець” та “ластик” (рис.20).

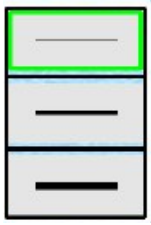

Рисунок 20 – Опції налаштування розміру активного інструменту

У додатку впроваджена система гарячих клавіш. Вони слугують для швидкого доступу до засобів редагування (0-9), а також для послідовної зміни налаштувань обраного інструменту (Left Shift). Зміна відбудеться в активному елементі у випадку, якщо вона можлива. Перевірка активації клавіш швидкого доступу виконується у функції checkkeys в головному циклі побудови зображення. Оскільки швидкий доступ реалізовано до основних інструментів редагування, checkeys може або запустити виконання функції picktool, або змінити основний колір на другорядний у selectcolour.

Окрім налаштувань та активованих елементів, головний цикл програми також перевіряє колір інструментів. Для зміни відтінку віджету, встановленого на панелі highlights, а також для зміни основного кольору усіх використовуваних інструментів користувач може звернутися до RGB моделі. Динамічна зміна поточного відтінку буде виконуватись тільки після наведення курсору на кольорову палітру. checkcolour перевіряє параметри m_hold та pen_touch, і, у випадку встановлення одного з них в “down”, змінює інтенсивність пікселя на відтінок з поточними координатами курсору.

Взаємодію компонентів інтерфейсу для визначення та коректного відображення змін, що внесені користувачем, представлено на рис. 21.

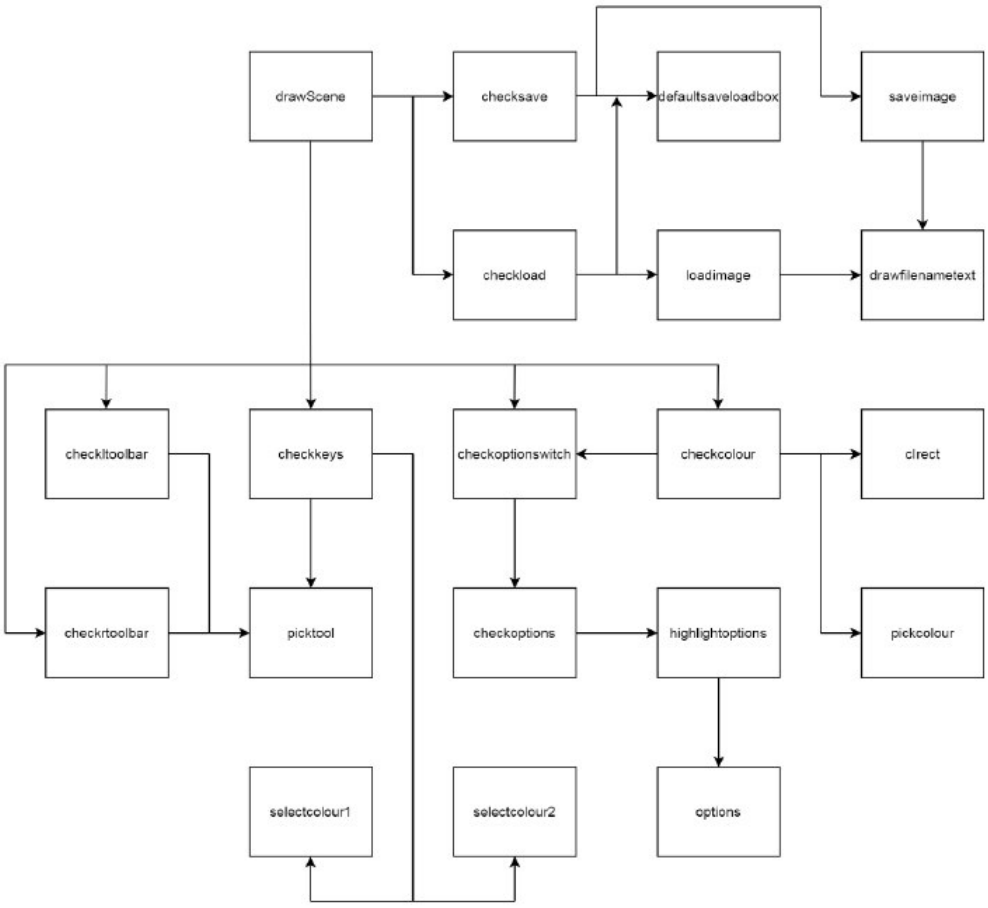

Рисунок 21 – Взаємодія компонентів інтерфейсу додатку

4.2 Взаємодія функціональних компонентів графічного редактора

Опис основних алгоритмів, необхідних для редагування растрового зображення, наведено у третьому розділі. Для прив’язки цих алгоритмів до віджетів функціональних елементів, замість стандартної бібліотеки pyGame, використовувалась бібліотека pgu з власним модулем графічного інтерфейсу. Оскільки створена система є інтерактивною, дане впровадження не тільки полегшило інтегрування значної кількості алгоритмів у розроблений код, але й зменшило середній час відгуку на звернення користувача.

В головній функції usetool виконується зчитування поточних даних про позицію курсору, а також про рівень тиску пера на поверхню у разі його активації. Перевірка наявності підключених пристроїв виконується у checkmhold та checkpentouch. У випадку, коли рівень тиску більше нуля, або при натисканні лівої кнопки миші, проводиться перевірка поточного обраного інструмента редагування.

Промальовування лінії, за допомогою інструменту «олівець» виконується в функції Pencil() головного файлу MainRasterEditor. Метод set_clip дозволяє редагування тільки пікселів растру. Для визначення сплайну апроксимуючої траєкторії, в буфер pencil_line_data, обмежений чотирма комірками, кожен раз записуються дані (поки параметр m_hold дорівнює “down”) про два останніх зчитування події зовнішнього пристрою для побудови попереднього сплайну, а також інформація про два нових зчитування. Після побудови траєкторія заповнюється зразками круглих відбитків олівця, розмір яких залежить від параметру pressure у pencil_line_data (рис.22).

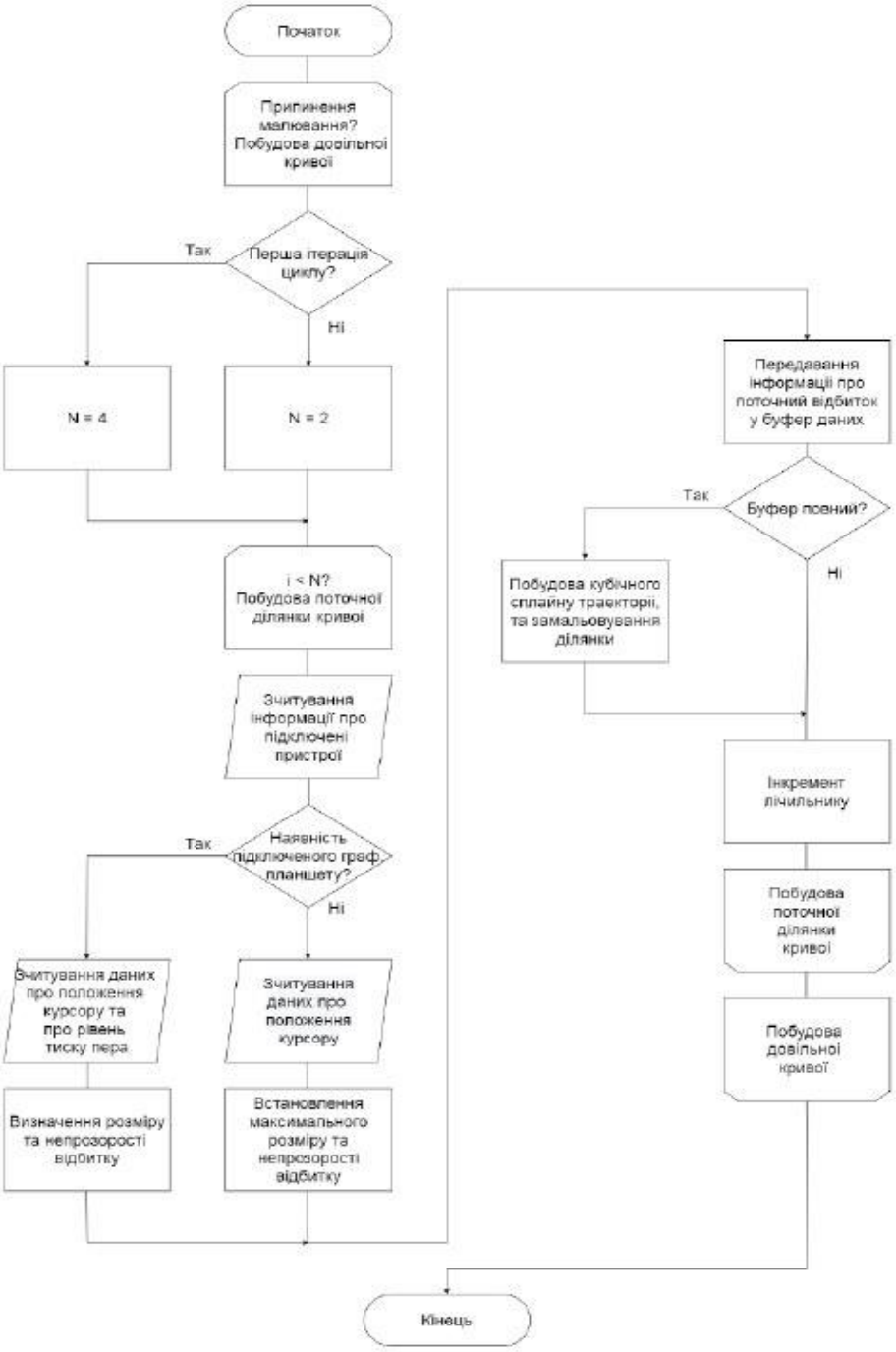

Рисунок 22 – Алгоритм побудови довільної кривої інструментом малювання

Алгоритм роботи ластику виконується у Eraser та працює майже так само, як і у олівця. Різниця полягає в тому, що на місце області виділених пікселів підставляється набір початкових пікселів згідно їх глобальних координат (навколо області, що являє собою описаний прямокутник, ведеться

порядкове зчитування пікселів та заміна тільки відредагованих на відтінок початкових). Не дивлячись на те, що такий підхід має меншу швидкодію, він надає можливість внесення змін у початкове зображення. Якщо було завантажено довільне зображення та проведено редагування групи його пікселів, ластик не замінить видалені пікселі на фонові, а поверне початковий варіант растру.

Редагування графічних об’єктів за допомогою інструменту «пензель» відрізняється від першого алгоритму. Оскільки промальовування олівцем має перекривання мазків, напівпрозорі пікселі будуть накладатися та руйнувати плавність по контуру відбитку. Функція Paint_Brush() використовує другий алгоритм заповнення кривої. Кожен раз при зміні параметру line_angle під час малювання перевіряється перевищення його граничного значення, що вказано у lim_angle. У разі успіху перевірки, завершується будова об’єкту, виконується заповнення проміжків між попереднім відбитком та поточним, зменшення інтенсивності пікселю в залежності від відстані до основної траєкторії.

Зафарбовування однорідної області однаковім кольором виконується у модулі PaintBucket. Дана функція є рекурсивною. До тих пір, поки не з’явиться ліва та права границі у рядку ypos початкового пікселю, або доки set_clip повертає true, виконується зміна відтінку та пошук сусідніх негранічних пікселів, а також записування даних у буфер bucket_list, після чого знову викликається початкова функція.

Для зміни поточного кольору використовується інструмент «піпетка». Цей елемент описано в функції EyeDropper. Окрім стандартних даних (позиція курсору), на вхід також поступає значення colourswitch. Алгоритм визначення поточного кольору описано у Pickcolour(). checkcolourswitch перевіряє, який з двох індикаторів панелі highlights активовано в даний момент. Зміна кольору відбувається тільки в активованому highlight. Доступ до зчитування відтінку пікселю можливий не тільки у растрі, але і на панелі RGB, та виконується за допомогою вбудованого методу get_at.

Для геометричних фігур є своя функція LineTool, RectTool та OvalTool, кожна з яких приймає на вхід параметри кольору, поточної позиції курсору, дані про зовнішні пристрої, а також ширину об’єкту. У випадку, якщо значення m_hold дорівнює up, або pen_tough дорівнює нулю, виконується заміна растрового зображення поточною копією. Інакше, будується відповідний геометричний об’єкт за алгоритмом Брезенхема.

Взаємодію функціональних елементів розроблених засобів представлено на рис. 23.

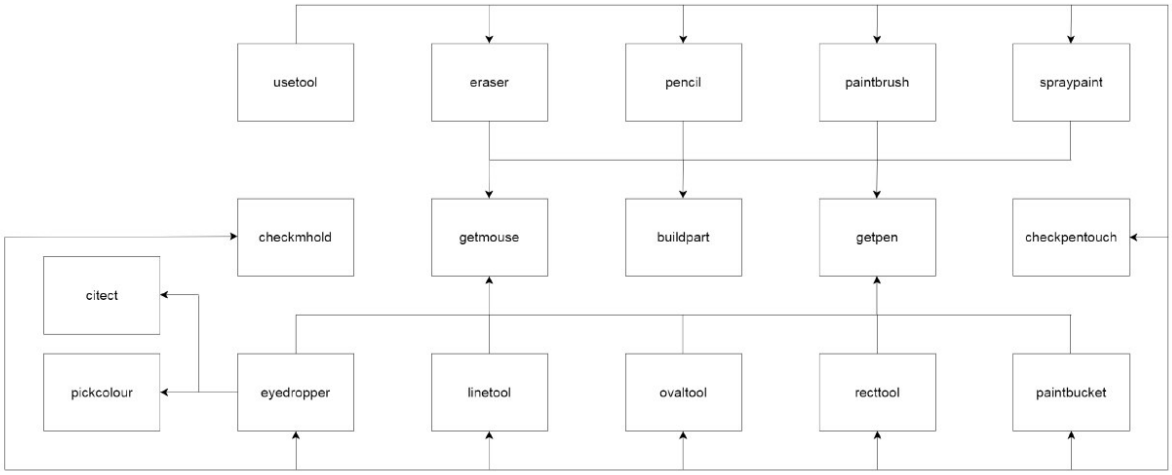

Рисунок 23 – Взаємодія функціональних елементів

Нижче подається призначення функціональних модулів розробленого додатку:

Pencil() – динамічне малювання олівцем;

Eraser() – динамічне видалення пікселів;

Paint_brush() – динамічне малювання пензлем;

Paint_bucket() – зафарбовування довільної області, обмеженої контуром;

Eye_dropper() – швидке визначення довільного пікселю зображення та встановлення його, як основного, для подальшої роботи;

Line_tool() – побудова довільного прямого відрізку;

Rect_tool() – побудова довільного прямокутнику;

Oval_tool() – побудова довільного еліпсу;

Selector() – виділення прямокутної області для одночасної зміни положення групи пікселів;

check_keys() – перевірка активації гарячих клавіш для обирання відповідних компонент;

get_mouse() – визначення поточної позиції миші;

get_Tablet() – визначення поточного положення грічного пера відносно поверхні, а також тиску;

check_m_hold() – перевірка натискання лівої кнопки миші. Повертає “down” на період фіксації та “up” після його завершення;

Pick_colour() – встановлення кольору пікселя;

check_colour() – перевірка будь-якої спроби змінити колір;

select_colour1() – виділення для активованого основного кольору поля-індикатора на панелі highlight;

select_colour2() – виділення для активованого другорядного кольору поля-індикатора на панелі highlight;

check_colour_switch() – очищення другого кольору поля з пам’яті;

C_Irect() – замальовування панелі highlight обраним кольором;

default_size() – повернення поточного розміру інструменту до розміру, встановленого за замовчуванням;

no_options() – заповнення пустої області після переходу від інструментів малювання до інших засобів;

options() – зміна розміру поточного інструменту малювання;

highlight_options() – виділення поля під час активації відповідної опції;

check_option_switch() – перевірка натискання shift для зміни розміру поточного інструменту;

check_r_toolbar() – перевірка вибору на правій панелі інструментів;

check_l_toolbar() – перевірка вибору на лівій панелі інструментів;

highlight_tool_box() – виділення поля-індикатора для активованої RGB моделі;

pick_tool() – активація необхідного інструменту;

use_tool() – ініціалізація кожного інструменту;

draw_l_toolbar() – будування лівої панелі з віджетами;

draw_r_toolbar() – будування правої панелі з віджетами;

check_save() – перевірка активації кнопки “save”;

check_load() – перевірка активації кнопки “load”;

save_image() – обирання директорії для зберігання зображення на комп’ютері

load_image() – обирання файлу для завантаження у програмі;

draw_Scene() – головний цикл для інтерактивного внесення змін у поточне зображення.

4.3. Тестування інструментів редагування

Під час роботи у редакторі користувачу надається набір засобів для створення нових графічних об’єктів або для редагування вже існуючих.

Перелік інструментів, доступних для використання:

- олівець
- пензель
- ластик
- зафарбовування
- розпилювач фарби
- піпетка
- довільна пряма
- довільний прямокутник
- довільне коло

Програма надає доступ до інструментів «олівець» (рис. 24) та «пензель» (рис. 25). Перший використовується при створенні растрових зображень для будування чіткого контуру. Щільність лінії та інтенсивність пікселів при використанні такого засобу, однакова на всіх ділянках. На відміну від олівця, пензель має різну непрозорість пікселів, що зменшується, при досяганні границь мазку. Це обумовлено тим, що пензель імітує художній ефект акварелі на полотні. Фарба, найбільш чітка на місці торкання пензля та аркуша але розмита ближче до країв, оскільки аркуш вбирає воду. Оскільки малювання кривої відбувається не по поточним зчитуванням курсору, а за допомогою сплайнів, під час використання інструменту не має ефекту тремтіння. Через побудову апроксимуючої кривої незначний зсув не вплине на загальну траєкторію.

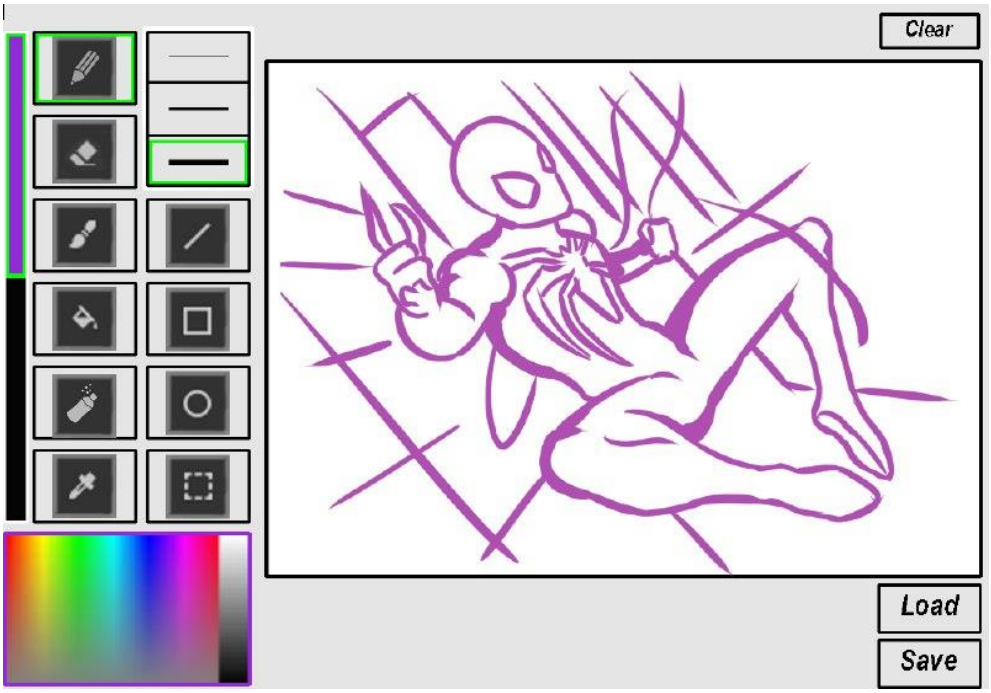

Рисунок 24 – Приклад малювання за допомогою інструменту «олівець»

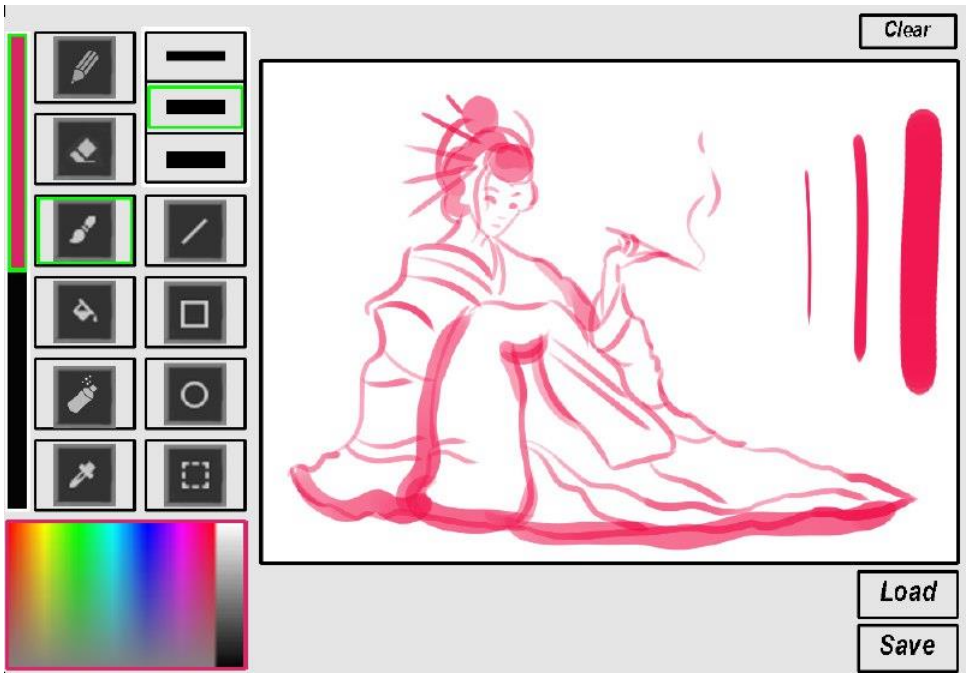

Рисунок 25 – Приклад малювання за допомогою інструменту «пензель»

Для зміни кольору групи пікселів у довільному околі користувач може використати інструмент «зафарбовування» (рис.26). Після виділення активного компоненту, достатньо навести курсор в область та натиснути ліву кнопку миші або торкнутися пером поверхні граф планшету. Якщо область обмежена замкненою кривою, то зафарбується тільки внутрішня множина пікселів. Інакше буде зафарбовано увесь растр (окрім інших областей, обмежених замкненими кривими). Оскільки використовується чотирьохзв'язний модифікований алгоритм, гранична крива може бути досить тонкою та запобігати "проливанню фарби" за межі контуру. Через те, що програма виконує порядкове рекурсивне зафарбовування, програма нормально функціонує в інтерактивному режимі навіть під час фарбування великих зображень.

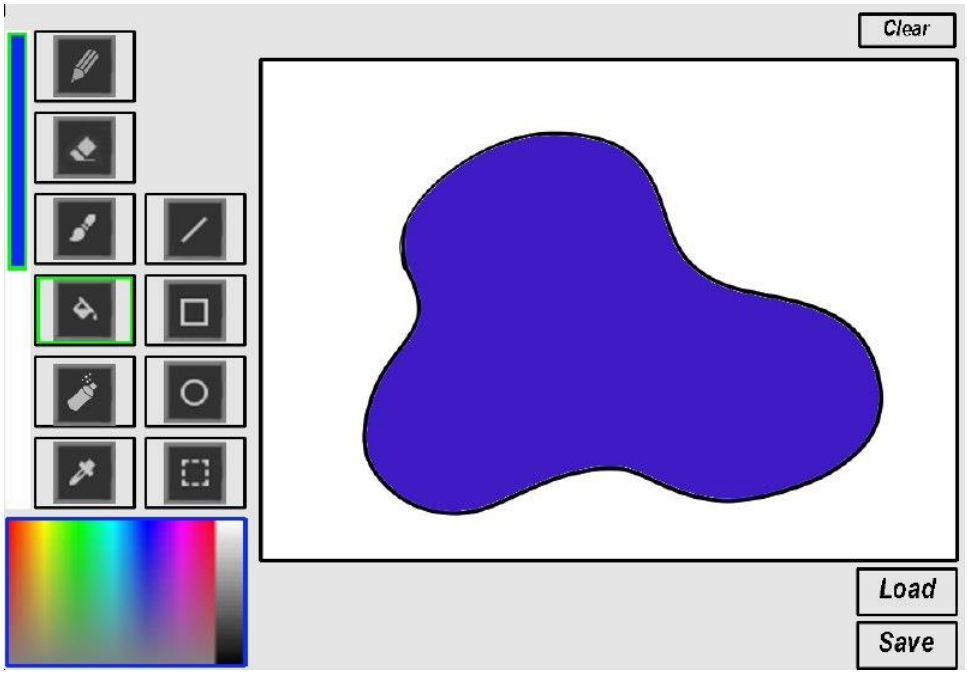

Рисунок 26 – Приклад використання інструменту «зафарбовування»

У додатку існує можливість побудови простих графічних об’єктів (рис. 27). Алгоритм створення нових об’єктів однаковий для всіх трьох компонент редактора. Малювання починається після натискання лівої кнопки миші та закінчується, коли користувач перестає виводити кінцеву точку побудованої фігури.

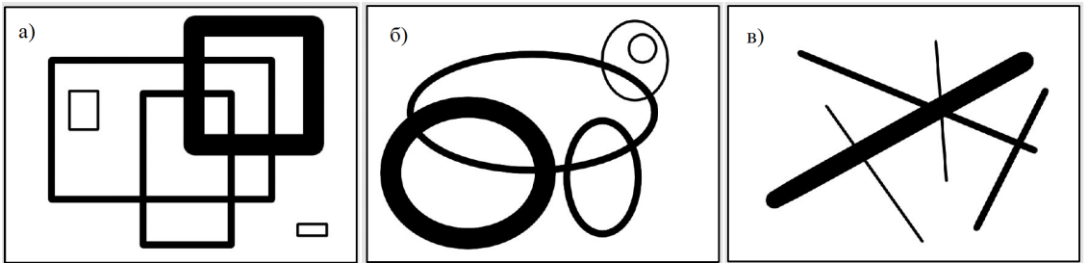

Рисунок 27 – Приклад побудови графічних об’єктів:
а) – прямокутників; б) – окружності; в) – ліній

Інструмент «ластик» працює за таким самим алгоритмом, як і олівець. Тому при їх використанні відсутній плавний перехід між головним кольором та фоном (рис.28).

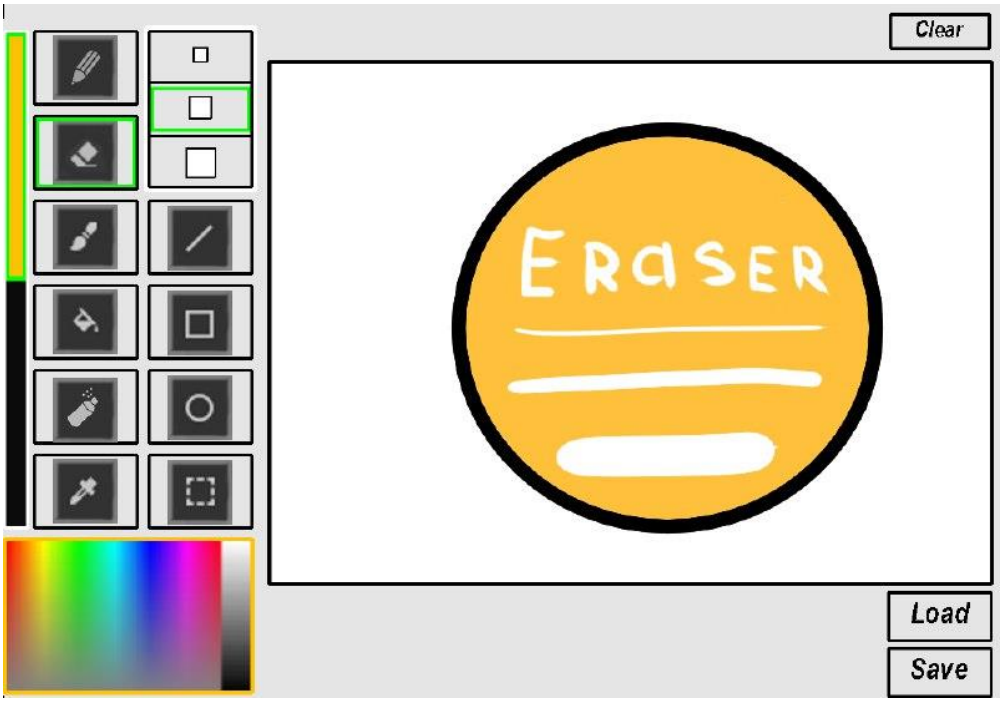

Рисунок 28 – Приклад використання інструменту «ластик»

Приклад створення зображення за допомогою розроблених засобів редагування представлено на рис. 29.

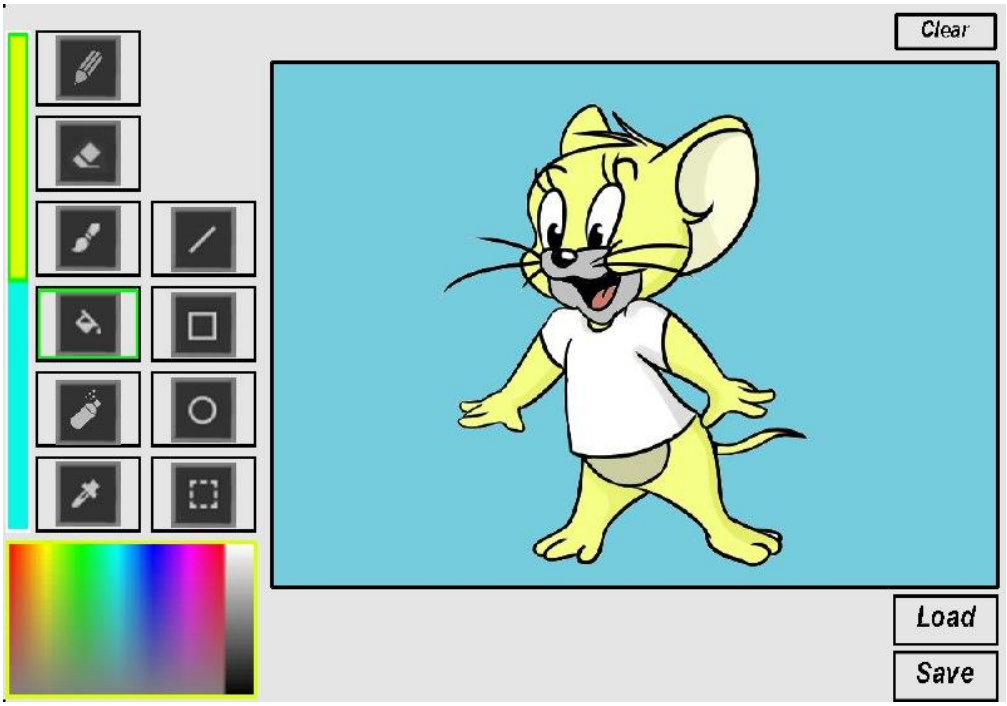

Рисунок 29 – Приклад побудови зображення у редакторі

4.4 Рекомендації щодо подальшого вдосконалення

Оскільки основною метою розробленого редактора було покращення результатів створення та редагування графічних об’єктів, нові можливості, перш за все, повинні підвищити якість отриманих зображень. Визначено три основні напрямки покращення якості:

1) За рахунок нових засобів редагування

- створення бібліотеки кістей;
- додавання інструменту для швидкого виділення області;
- розробка текстурного та градієнтного зафарбовування;

2) За рахунок модифікації існуючих алгоритмів редактору

- збільшення швидкодії алгоритмів заповнення апроксимуючої траєкторії, для усіх інструментів малювання;
- зменшення інтенсивності пікселів по контуру відбитку під час малювання;
- розробка та впровадження фільтра antialising для видалення сходового ефекту контуру при побудові нових графічних об’єктів;

3) Додавання системи загального підвищення якості растру, після завершення використання поточного інструменту;

- додавання системи шарів;
- візуальне збільшення кількості відтінків кольорів у зображенні за рахунок растрування усієї області або дизерингу;
- використання системи локальної цифрової фільтрації кінцевого зображення.

ВИСНОВКИ

Даний дипломний проєкт присвячено розробці зручних у використанні та ефективних засобів створення і редагування графічних зображень.

Під час вирішення поставленої задачі було визначено і проаналізовано набір програмних засобів для реалізації інтерфейсу графічного редактора та операцій редагування. Обрано мову програмування Python та фреймворк PyGame, що має вбудовані оптимальні алгоритми побудови графічних примітивів завдяки використанню мови C++ та асемблеру.

В дипломному проєкті було обґрунтовано доцільність покращення роботи компонентів сучасних редакторів, визначено основні вимоги до результатів при управлінні графічними інструментами, забезпечено підтримку використання графічного планшету.

Розроблено GUI для взаємодії з користувачем, а також набір засобів, що відтворюють ефект малювання художніми інструментами. Доступна побудова простих растрових об’єктів.

Тестування роботи редактора показало відповідність розроблених засобів поставленим вимогам.

Впровадження розробки сприятиме підвищенню якості створюваних растрових зображень та редагування вже існуючих.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Julien Lagarde, Benoit Bardy, Manfred Nussek. Perception and prediction of simple object interactions. 2007. – 7 р.(дата звернення 20.03.2021).
2. Vector vs Raster Graphics. URL: https://www.geeksforgeeks.org/vector-vs-raster-graphics/#:~:text=The main difference between vector,which together form an image(дата звернення 20.03.2021).
3. Free dictionary. Raster graphics editor. URL: https://encyclopedia2.thefreedictionary.com/Raster+graphics+editor(дата звернення 20.03.2021).
4. Adobe Learn & Support. URL: https://helpx.adobe.com/support/creative-cloud.html(дата звернення 21.03.2021).
5. Modern Computer Graphics. URL: https://brasil.cel.agh.edu.pl/~12sustrojny/en/grafika-rastrowa-a-grafika-wektorowa/index.html(дата звернення 21.03.2021).
6. Micrisoft documentation: Painting and drawing. URL: https://docs.microsoft.com/en-us/windows/win32/gdi/painting-and-drawing (дата звернення 27.03.2021).
7. Microsoft Paint. URL: https://en.wikipedia.org/wiki/Microsoft_Paint(дата звернення 27.03.2021).
8. Kolour Paint documentation. URL: http://kolourpaint.org/about.html(дата звернення 27.03.2021).
9. Easy Paint. URL: https://zenway.ru/page/easypaint(дата звернення 27.03.2021).
10. Mark Tyler’s Paint Program. URL: http://mtpaint.sourceforge.net(дата звернення 27.03.2021).

11. Curve brush. URL: https://docs.krita.org/en/reference_manual/brushes/brush_engines/curve_engine.html(дата звернення 18.04.2021).
12. Features of Windows OS and Advantages and Disadvantages of Windows OS. URL: http://cybercomputing.blogspot.com/2014/08/Features-of-Windows-OS-and-Advantages-and-Disadvantages-of-Windows-OS.html (дата звернення 18.04.2021).
13. Main features of modern Operation Systems. URL: https://www.guru99.com/operating-system-tutorial.html(дата звернення 18.04.2021).
14. Official Python documentation. URL: https://www.python.org/doc/(дата звернення 18.04.2021).
15. Python features. URL: https://www.geeksforgeeks.org/python-features/(дата звернення 18.04.2021).
16. Python programming language. URL: https://en.wikipedia.org/wiki/Python_(programming_language)(дата звернення 18.04.2021).
17. PyQt – wiki. URL: https://ru.wikipedia.org/wiki/PyQt(дата звернення 20.04.2021).
18. Python GUI с PyQt и QtDesigner. URL: https://tproger.ru/translations/python-gui-pyqt (дата звернення 20.04.2021).
19. wxPython documentation. URL: https://wiki.wxpython.org(дата звернення 20.04.2021).
20. wxPython – руководство. URL: https://coderlessons.com/tutorials/python-technologies/uchim-wxpython/wxpython-kratkoe-rukovodstvo(дата звернення 20.04.2021).
21. tkinter – Python interface to Tcl/Tk URL: https://docs.python.org/3/library/tkinter.html(дата звернення 20.04.2021).

22. Введение в pytkinter. URL: https://habr.com/ru/post/133337/(дата звернення 20.04.2021).
23. Python GUI. URL: https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter(дата звернення 23.04.2021).
24. pyGame official documentation. URL: https://www.pygame.org/wiki/about (дата звернення 23.04.2021).
25. pyGame – wiki. URL: https://ru.wikipedia.org/wiki/Pygame(дата звернення 23.04.2021).
26. PyCharm tools. URL: https://www.jetbrains.com/pycharm/(дата звернення 23.04.2021)
27. Fishkin, Kenneth P. Algorithms for brush movement /Kenneth P. Fishkin, Brian A. Barsky // The Visual Computer. — 1985р. 221– 230. URL: http://dx.doi.org/10.1007/BF02021811(дата звернення 27.04.2021)
29. Базовые растровые алгоритмы: Растеризаций линий. Алгоритмы закрашивания.ТурлаповВ.Е.URL:http://www.graph.unn.ru/rus/materials/CG/CG07_RasterAlgorithms.pdf(дата звернення 29.04.2021)
30. Порєв. В. І. Компьютерная графика. Питер, 2001.– 47 с.
31. Гантмахер Ф.Р. Теория матриц. Питер, 1967. – 423 с.