МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ФАХОВИЙ КОЛЕДЖ ІНЖЕНЕРІЇ ТА УПРАВЛІННЯ

НАЦІОНАЛЬНОГО АВІАЦІЙНОГО УНІВЕРСИТЕТУ»

ДОПУСТИТИ ДО ЗАХИСТУ

Заступник директора з НР

_____ О.В. Родіонова

«____»_____ 2021 р.

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ

«БАКАЛАВР»

Тема: Програмний модуль для роботи з зображенням

Автор: Давидова М.В.

Керівник проєкту: Рябчук Н.А.

Нормконтролер: Кругляк В.М.

Київ 2021

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНЖЕНЕРІЇ ТА УПРАВЛІННЯ
НАЦІОНАЛЬНОГО АВІАЦІЙНОГО УНІВЕРСИТЕТУ»

Циклова комісія: Інженерії програмного забезпечення

Освітнього ступеня: «Бакалавр»

Спеціальність: 123 Комп'ютерна інженерія

Освітньо-професійна програма: «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Голова циклової комісії

_____ Н.А. Рябчук

«____»_____ 2021 р.

ЗАВДАННЯ

на виконання дипломного проєкту

Давидової Маргарити Владиславівни

1. Тема роботи: «Програмний модуль для роботи з зображенням»

затверджена наказом від «15» березня 2021 року № 28-Ст.

2. Термін виконання: з 12.04.2021р. по 20.06.2021р.

3. Вихідні дані: розробити найпростіший графічний редактор з допомогою об'єктно-орієнтованої мови програмування, з використанням його графічних функцій і методів.

4. Зміст пояснювальної записки:

У 1 розділі проаналізувати завдання на проєкт та основні методи його розв'язання.

У 2 розділі навести опис основних етапів розробки.

У 3 розділі описати основні вимоги охорони праці.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	12.04.2021	Виконано
2.	Аналіз літературних джерел	15.04.2021	Виконано
3.	Обґрунтування рішення	21.04.2021	Виконано
4.	Збір інформації	01.05.2021	Виконано
5.	Аналіз існуючих методів. Обгрунтування вибору мови програмування	06.05.2021	Виконано
6.	Виконання проєкту	08.06.2021	Виконано
7.	Оформлення і друк пояснювальної записки	15.06.2021	Виконано
8.	Оформлення презентації	17.06.2021	Виконано
9.	Отримання рецензій	20.06.2021	Виконано
10.	Захист проєкту		

Дипломник Давидова М.В.

(підпис, дата) (П.І.Б.)

Дипломний керівник Рябчук Н.А.

(підпис, дата) (П.І.Б.)

Консультант з охорони праці Пешков І.В.

(підпис, дата) (П.І.Б).

Зміст

Вступ

Представлення даних на моніторі комп'ютера в графічному вигляді вперше було реалізовано в середині 50-х років для великих ЕОМ, що застосовувалися в наукових і військових дослідженнях. З тих пір графічний спосіб відображення даних став невід'ємною частиною переважної більшості комп'ютерних систем, особливо персональних.

Комп'ютерна графіка – це спеціальна область інформатики, що вивчає методи і засоби створення та обробки зображень за допомогою програмно-апаратних обчислювальних комплексів. Вона охоплює всі види і форми представлення зображень, доступних для сприйняття людиною або на екрані монітора, або у вигляді копії на зовнішньому носії (папір, кіноплівка, тканина і інше). Без комп'ютерної графіки неможливо уявити собі не тільки комп'ютерний, але і звичайний, цілком матеріальний світ.

На сьогоднішній день комп'ютери і комп'ютерна графіка невід'ємна частина життя сучасного суспільства. Для прикладу назвемо медицину (комп'ютерна томографія), наукові дослідження (візуалізація будови речовини, векторних полів і інших даних), моделювання тканин та одягу, дослідно-конструкторські розробки, рекламні щити, кольорові журнали, спецефекти у фільмах - все це в тій чи іншій мірі має відношення до комп'ютерної графіки. Тому створені програми для створення і редагування зображень, тобто графічні редактори.

Комп'ютерною графікою останнім часом займаються багато, що обумовлено високими темпами розвитку обчислювальної техніки. Більше 90% інформації здорова людина отримує через зір або асоціює з геометричними просторовими уявленнями. Комп'ютерна графіка має величезний потенціал для полегшення процесу пізнання і творчості.

В даний час ринок програмного забезпечення переповнений різними програмами і редакторами, що дозволяють обробляти і редагувати цифрові фото. Людині, яка не дуже добре розуміє особливості тих чи інших програмних засобів,

часом дуже складно розібратися в цьому різноманітності софту. Однак, правильний вибір програмних засобів для вирішення конкретного завдання з обробки фотознімків є одним із запорук успіху отримання закінчених фотографій. Кажучи про графічні редактори, перш за все, необхідно відзначити, що всі цифрові зображення поділяються на векторні і точкові.

Графічний редактор - програма (або пакет програм), що дозволяє створювати і редагувати двовимірні зображення за допомогою комп'ютера Актуальність дипломного проекту на тему «Програмний модуль для роботи з зображенням» обумовлена тим, що будь-який школяр, студент і викладач в даний час повинен володіти комп'ютерними технологіями на високому рівні.

«Програмний модуль для роботи з зображенням» – це програмний засіб обробки растрових зображень. Метою виконання дипломного проекту є створення програмного засобу обробки елементарних растрових зображень. Створений програмний засіб є зручний та зрозумілий у використанні.

Дипломний проект складається з загальної частини, спеціальної частини, охорони праці та висновків.

Загальна частина описує основні вимоги до програмного засобу, основні теоретичні відомості про растрові зображення, вибір середовища розробки для створення програми та мови програмування, за допомогою якої засіб буде реалізовано.

Спеціальна частина описує життєвий цикл розробки програмного засобу, який включає в себе: алгоритми та блок-схеми, структуру програми, основні засоби реалізації, функціональні та нефункціональні вимоги, огляд та порівняння засобу із ідентичними, інструкція користувача програмного засобу та тестування.

Розділ Охорона праці аналізує основні вимоги до приміщень, до робочого місця, рівня шуму та освітленості, а також основні засоби та заходи протипожежного захисту.

1 Загальна частина

1.1 Постановка задачі

Необхідно розробити найпростіший графічний редактор з допомогою об'єктно-орієнтованої мови програмування, з використанням його графічних функцій і методів.

Реалізований графічний редактор дозволяє здійснювати роботу з як уже наявними найпростішими зображеннями, так і створювати власні за допомогою різних можливостей, таких як:

1) створення растрових об'єктів (квадрата) і їх різні модифікації (розтягування) в прямокутник;
2) створення малюнків за допомогою курсора;
3) можливість завдання розмірності;
4) можливість зміни розташування зображення;
5) стирання об’єкту повторним натисненням на нього;
6) інвертування, поворот зображення на 90^{o}, 180^{o};
7) також реалізувати відкриття і збереження картинки в форматі * .maz / *. png / *. hex;
8) дзеркальне відображення наявного зображення;
9) зміна розташування об’єкту на листі;
10) зміна розташування зображення на сторінці.

Інтерфейс програми складається з основного вікна редагування з набором наявних примітивів і робочої області. Управління програмою проводиться мишею.

У процесі розробки програми також необхідно виконати додаткові поставлені завдання:

1) створення файлу;
2) відкриття файлу:
3) збереження файлу:

4) друк файлу;

5) попередній перегляд файлу.

До вхідної інформації відносяться елементи зображення, які вводить користувач.

Вихідною інформацією є графічне представлення створених об’єктів у вигляді зображення.

Платформа розробки:

1) процесор: Intel (R) Core ™ i5;

2) оперативна пам'ять: 6Gb;

3) жорсткий диск: HDD 500Gb;

4) монітор;

5) клавіатура;

6) миша;

Мінімальні вимоги коректного функціонування програмного засобу:

1) програмна платформи: *Windows XP*/ 7/ 8/ 8.1/ 10;

2) 32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1 ГГц;

3) відеокарта з 1 Гб відеопам'яті;

4) процесор з частотою 2,0 ГГц і 2 ядрами;

5) 1 Гб оперативної пам'яті;

6) 1 Гб місця на жорсткому диску.

1.2 Теоретичні відомості

Графічні програми - програмне забезпечення, що дозволяє створювати, редагувати або переглядати графічні файли.

Графічний редактор - програма (або пакет програм), що дозволяє створювати і редагувати двовимірні зображення за допомогою комп'ютера.

Комп'ютерну графіку можна розділити на три категорії:

1) растрова графіка;

2) векторна графіка;

3) тривимірна графіка.

Растрове зображення - це файл даних або структура, що представляє прямокутну сітку пікселів. Піксель - найменша одиниця двомірного цифрового зображення в растровій графіці. Піксель є неподільний об'єкт прямокутної (зазвичай квадратної) форми, що володіє певним кольором. Растрове комп'ютерне зображення складається з пікселів, розташованих по рядках і стовпцях. На комп'ютерному моніторі, папері та інших відображають пристроях і матеріалах.

При використанні растрової графіки важливим елементом є:

1) розмір полотна (canvas);

2) колірний простір (наприклад, RGB);

3) кількість використовуваних квітів.

Растрову графіку редагують за допомогою растрових графічних редакторів. Створюється растрова графіка фотоапаратами, сканерами, безпосередньо в растровому редакторі, також шляхом експорту з векторного редактора або у вигляді скріншотів.

Растровий графічний редактор - спеціалізована програма, призначена для створення і обробки зображень. Подібні програмні продукти знайшли широке застосування в роботі художників-ілюстраторів, при підготовці зображень до друку друкарським способом або на фотопапері, публікації в Інтернеті.

Растрові графічні редактори дозволяють користувачеві малювати і редагувати зображення на екрані комп'ютера, а також зберігати їх в різних растрових форматах.

На противагу векторних редакторів, растрові використовують для представлення зображень матрицю крапок (bitmap). Однак, більшість сучасних растрових редакторів містять векторні інструменти редагування як допоміжні.

Растрові формати:

GIF (Graphics Interchange Format) - формат для обміну зображеннями, зберігання графічних зображень. Формат GIF здатний зберігати стислі без втрат

зображення у форматі до 256 квітів з палітрою, і призначений, в основному, для креслень, графіків.

Незалежний від апаратного забезпечення формат GIF був розроблений в 1987 році (GIF87a) фірмою CompuServe для передачі растрових зображень по мережах. У 1989-му формат був модифікований (GIF89a), були додані підтримка прозорості і анімації. GIF використовує LZW-компресію, що дозволяє добре стискати файли, в яких багато однорідних заливок (логотипи, написи, схеми). GIF широко використовується на сторінках Всесвітньої Павутини.

PNG (Portable Network Graphics) - растровий формат зберігання графічної інформації, що використовує стиснення без втрат. PNG був створений як для поліпшення, так і для заміни формату GIF графічним форматом, що не вимагає ліцензії для використання.

BMP (Bitmap - бітова карта) - формат зберігання растрових зображень. Зазвичай використовується без стиснення.

З форматом BMP працює величезна кількість програм, так як його підтримка інтегрована в операційні системи Windows і OS / 2. Файли формату BMP можуть мати розширення .bmp, .dib і .rle. Крім того, дані цього формату включаються в двійкові файли ресурсів RES і в PE-файли.

Глибина кольору в даному форматі може бути від 1 до 48 біт на піксель, максимальні габарити зображення 65535 на 65535 пікселів. У форматі BMP є підтримка стиснення по алгоритму RLE, але, незважаючи на це, через великий обсяг він рідко використовується в Інтернеті.

Растрове зображення являє собою безліч кольорових крапок (зазвичай прямокутне) на моніторі, папері та інших відображають пристроях і матеріалах.

Кожна точка растрового зображення мала настільки, що не сприймається оком як окремий об'єкт, але сукупність точок сприймається як єдине зображення. Така технологія побудови зображень дуже нагадує мозаїку.

Приклад растрового зображення можна побачити в газеті чи журналі: будь-яка фотографія в них складається з масиву непомітних на перший погляд точок різного кольору і розміру. Телевізійне зображення і вид на екрані монітора - це теж

растр, тільки, на відміну від друку на папері, растрова крапка комп'ютерного зображення має квадратну форму (рис. 1.1).

Рисунок 1.1 – Приклад растрового зображення

Термін "піксель" ("піксел") (від англ. Pixel, pel - скорочення від англ. Pix element, picture cell, picture element - елемент зображення) має два значення:

1) найменший елемент двовимірного цифрового зображення в растровій графіці;

2) "фізичний" елемент матриці дисплеїв, що формують зображення. Піксель є неподільний об'єкт прямокутної або круглої форми, що характеризується певним кольором (стосовно до плазмових панелей).

Растрове комп'ютерне зображення складається з пікселів, розташованих по рядках і стовпцях. Чим більше пікселів на одиницю площі містить зображення, тим воно детальней.

Кожен піксель растрового зображення - об'єкт, що характеризується певним кольором, яскравістю і, можливо, прозорістю, або комбінацією цих значень. Один

піксель може зберігати інформацію тільки про один колір, який і асоціюється з ним (в деяких комп'ютерних системах колір і пікселі представлені у вигляді двох роздільних об'єктів).

Піксель - це також найменша одиниця растрового зображення, одержуваного за допомогою графічних систем виведення інформації (комп'ютерні монітори, принтери і т.д.). Дозвіл такого пристрою визначається горизонтальним і вертикальним розмірами виведеного зображення в пікселях. Колір пікселів, що відображаються на кольорових моніторах, складається з тріад (субпікселів червоного, зеленого і синього кольорів, розташованих поруч в певній послідовності).

Якість растрового зображення залежить від кількості растрових точок, з яких воно складається (рис. 8.4). Основним показником якості є дозвіл зображення, тобто кількість точок на одиницю довжини (дюйм, мм, см). Найчастіше вимірюється кількість точок на один дюйм (англ. Dpi - dots per inch). Більша кількість точок дає більшу деталізацію зображення, однак при збереженні такого зображення необхідно зберегти інформацію про колір кожної точки, а оскільки точок може бути до декількох мільйонів, то розмір збереженого файлу теж буде великим.

Максимальна деталізація растрового зображення задається при його створенні і не може бути збільшена. Без особливих втрат растрові зображення можна тільки лише зменшувати. При збільшенні растрових зображень пікселі перетворюються на збільшені квадрати того чи іншого кольору.

Створюється растрова графіка фотоапаратами, сканерами, безпосередньо в растровому редакторі, а також шляхом експорту з векторного редактора або у вигляді знімків екрану.

1.3 Аналіз засобів реалізації та обґрунтування вибору мови програмування

Обрано декілька основних об’єктно-орієнтованих мов програмування, щоб порівняти, яка найбільше підходить для реалізації програмного засобу.

До таких, зокрема, належать C++, C# та Java. Розглянемо детальніше особливості кожної із них.

Мова програмування C ++.

Особливості мови: C ++ може багато чого. Занадто багато. Це спроба стати універсальним, при цьому не будучи хорошим в чомусь одному. У мові є: оператор goto, покажчики, посилання, ООП, перевантаження операторів і інші не особливо корисні фічі.

Мова була створена в далекому 1979 році, коли його творці не знали на чому потрібно фокусуватися. Додавати в мову більше можливостей вважалося хорошою ідеєю, адже це збільшувало область застосування.

Швидкість: C ++ славиться довгої компіляцією. Вона значно довше ніж у Java. З іншого боку продуктивність вже скомпільованих додатків і час їх запуску досить гарні.

Екосистема/інструментарій: опис помилок - не сильне сторона C ++.

Прибирання сміття: цієї фічі ніколи не було в C ++. Ручне управління складанням сміття - джерело безлічі помилок.

ООП, яке не вийшло: за часів створення C ++ ООП була крутий новою парадигмою, але при її реалізації було допущено кілька критичних помилок. Принаймні в C ++ використовувати ООП не обов'язково (На відміну від Java).

Складність вивчення: C ++ - складний низькорівнева мова без автоматичного управління пам'яттю. Його складно вивчати новачкам через надмірну кількість функцій.

Паралелізм: мова створювався за часів одноядерних обчислень, тільки в останні десятиліття в нього була додана рудиментарна підтримка паралелізму.

Обробка помилок: кращий механізм обробки - викид і обробка виключень.

Іммутабельність: підтримка відсутня.

Підтримка NULL: всі посилання можуть бути NULL.

Вердикт: невдала спроба перевершити мову Сі. Ймовірно, варто використовувати тільки для системного програмування. Однак і тут є кращі альтернативи.

Мова програмування Java.

Прибирання сміття: це одне з ключових переваг Java над C ++, що дозволяє уникнути безлічі багів.

Екосистема: Java існує досить довго, тому вона має значну екосистемою для бекенд-розробки. Це значно зменшує витрати на розробку.

ООП: Детальніше моя думка про ООП можна дізнатися в статті ООП - катастрофа на трильйон доларів. Замість цього я процитую більш видатної людини: «Мені шкода, що я придумав для цього термін «об'єкти», і люди сфокусувалися на побічної ідеї. Головна ідея - повідомлення» - Алан Кей, винахідник ООП.

Швидкість: Java запускається на JVM, що уповільнює час старту. Я бачив програми які запускалися за 30 секунд і довше, що неприйнятно для сучасних додатків. Час компіляції зростає на великих проектах, що впливає на продуктивність розробників. Однак продуктивність JVM під час виконання програми дійсно хороша.

Складність вивчення: Хоча Java досить просту мову, писати на ньому хороший об'єктно-орієнтована код дійсно складно.

Паралелізм: Java також як і C ++ був створений в еру одноядерних обчислень, і має лише рудиментарну підтримку паралелізму.

Підтримка NULL: всі посилання можуть бути NULL.

Обробка помилок: кращий механізм обробки - викид і обробка виключень.

Іммутабельність: підтримка відсутня.

Вердикт: Java був непоганим сучасною мовою програмування в момент своєї появи. Його псує зосередженість на ООП. Мова дуже багатослівний і страждає від шаблонного коду.

Мова програмування C #.

Синтаксис: синтаксис C # завжди трохи випереджав Java. Він менше страждає від шаблонного коду. Але хоч C # і об'єктно-орієнтована мова, він теж страждає від багатослівності. Приємно бачити як синтаксис C # поліпшується з кожним релізом, додаються: зіставлення зі зразком, кортежі і інші можливості.

ООП: C #, як і Java більше зосереджений на ООП. І знову, замість того щоб розповідати про недоліки ООП, я процитую більш видатної людини: «Я вважаю, що недостатнє перевикористання більше відноситься до ООП мов, ніж функціональним. Тому що проблема ООП мов в неявній середовищі, яку вони тягають за собою. Ви хочете банан, але отримуєте горилу, що тримає банан і цілі джунглі» - Джо Армстронг, творець мови Erlang.

Мультипарадигменість: розробники стверджують, що C# - мультипарадигмена мова. Зокрема, кажуть що C # підтримує функціональне програмування. Я вважаю, що підтримки функцій першого класу не достатньо для того, щоб вважати мову функціональним. Що для цього потрібно? Як мінімум вбудована підтримка іммутабельних структур даних, зіставлення зі зразком, конвеєрний оператор для створення ланцюжків функцій і алгебраїчні типи даних.

Паралелізм: аналогічно з C ++ і Java.

Підтримка NULL: аналогічно з C ++ і Java.

Обробка помилок: аналогічно з C ++ і Java.

Іммутабельність: аналогічно з C ++ і Java.

Вердикт: як і у випадку з Java я б порекомендував більш сучасні мови програмування.

C# під капотом - та ж Java, з більш сучасним синтаксисом.

Враховуючи те, що перелічені властивості використовуватимуться для створення програмного засобу, для програмування було обрано мову C#.

Середовищем розробки програмного додатку обрано інтегроване середовище розробки програмного забезпечення Microsoft Visual Studio.

Дане середовище дозволяє створювати різноманітні програмні продукти: консольні програми, програми з графічним інтерфейсом, наприклад віконні додатки Windows Forms, а також Web-додатки тощо.

Середовище Visual Studio дозволяє розробляти додатки, використовуючи різні мови програмування: Visual C#, Visual Basic, Visual F#, Visual C++, Python і т.д. Також існує можливість розробляти додатки не тільки під Windows, а і під інші популярні платформи: Android, iOS.

Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Вбудований відладчик працює як відладчик на рівні вихідного рівня, так і відладчик на рівні машини. Інші вбудовані інструменти включають програму для кодування, конструктор форм для побудови графічних інтерфейсів, веб-дизайнер, дизайнер класів та дизайнер схеми баз даних. Він приймає плагіни, які покращують функціональність практично на всіх рівнях, включаючи підтримку систем керування джерельними ресурсами (наприклад, Subversion та Git) та додавання нових наборів інструментів, таких як редактори та візуальні розробники для мов або наборів інструментів для інших аспектів розробки програмного забезпечення життєвий цикл (наприклад, клієнт Team Foundation Server: Team Explorer).

Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Вбудований відладчик працює як відладчик на рівні вихідного рівня, так і відладчик на рівні машини. Інші вбудовані інструменти включають програму для кодування, конструктор форм для побудови графічних інтерфейсів, веб-дизайнер, дизайнер класів та дизайнер схеми баз даних. Він приймає плагіни, які покращують функціональність практично на всіх рівнях, включаючи підтримку систем керування джерельними ресурсами (наприклад, Subversion та Git) та додавання нових наборів інструментів, таких як редактори та візуальні розробники для мов або наборів інструментів для інших аспектів розробки програмного забезпечення життєвий цикл (наприклад, клієнт Team Foundation Server: Team Explorer).

Visual Studio підтримує 36 різних мов програмування і дозволяє редакторові коду та відладчику підтримувати (в тій чи іншій мірі) майже будь-яку мову програмування, якщо існує певна мова-служба. Вбудовані мови включають C, C ++, C ++ / CLI, Visual Basic. NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML та CSS. Підтримка інших мов, таких як Python, Ruby, Node.js та M, серед інших, доступна через плагіни.

Visual Studio дозволяє підключити функціональність, кодовану як VSPackage. Після встановлення функціональність доступна як Сервіс. IDE надає три сервіси: SVsSolution, що забезпечує можливість переліку проектів та рішень; SVsUIShell, що забезпечує вікна та функціональність інтерфейсу користувача (включаючи вкладки, панелі інструментів та вікна інструментів); і SVsShell, що займається реєстрацією VSPackages. Крім того, IDE також відповідає за координування та забезпечення зв'язку між службами. Всі редактори, дизайнери, типи проектів та інші інструменти реалізовані як VSPackages. Visual Studio використовує COM для доступу до VSPackages. SDK Visual Studio також включає керовану пакувальну структуру (MPF), яка являє собою набір керованих обгорток навколо COM-інтерфейсів, які дозволяють писати пакунки на будь-якій сумісній мові CLI. Проте MPF не забезпечує всі функціональні можливості, що висуваються через інтерфейси Visual Studio COM. Ці послуги можуть бути використані для створення інших пакетів, які додають до Visual Studio IDE функціональність.

2 Спеціальна частина

2.1 Опис алгоритму створення програмного засобу

Алгоритм – точно визначений план дій виконавця, спрямований на розв'язання якоїсь задачі.

Для більш наочного уявлення алгоритму використовується графічний спосіб. Існує кілька способів графічного опису алгоритмів. Найбільш широко використовуваним на практиці графічним описом алгоритмів є використання блок-схем. Безсумнівна перевага блок-схем – наочність і простота запису алгоритму.

Кожній дії алгоритму відповідає геометрична фігура (блоковий символ).

На рис. 2.1 зображений алгоритм растрового графічного редактора.

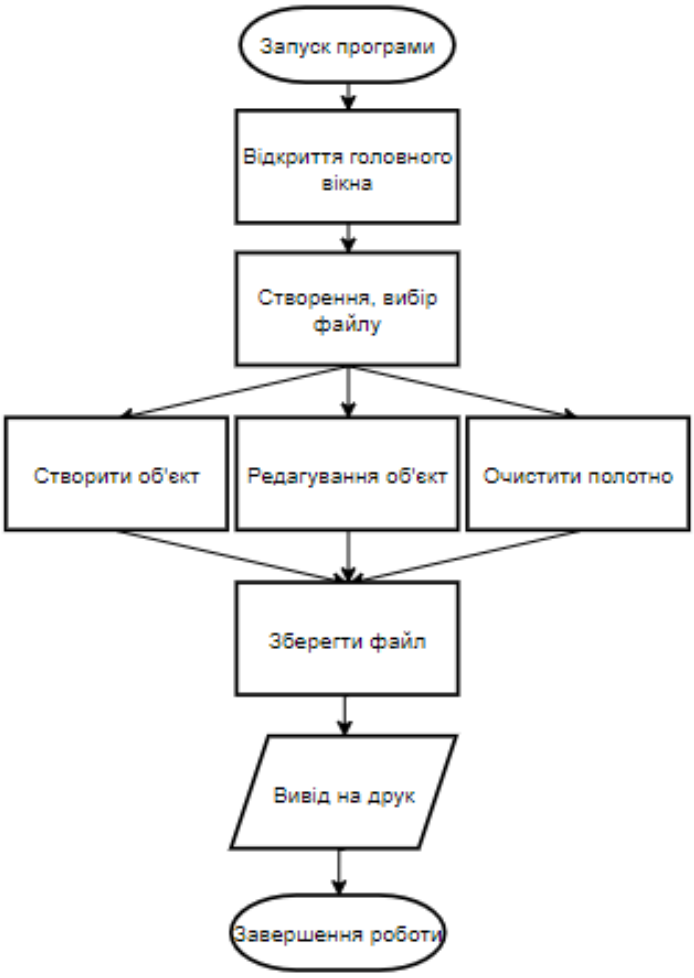

Рисунок 2.1 – Блок-схема алгоритму програми «Програмний модуль для роботи з зображенням»

Алгоритм описує процес роботи програмного засобу, послідовність створення/редагування зображень.

Алгоритм, зображений на рис. 2.1, містить наступні основні процеси: відкриття головного вікна, створення/вибір файлу, створення, редагування або очищення полотна, збереження файлу та вивід файлу на друк, після чого можна або завершити роботу з програмою.

2.2 Функціональні та нефункціональні вимоги

Функціональні вимоги (functional requirements) визначають функціональність ПЗ, яку розробники повинні побудувати, щоб користувачі змогли виконати свої завдання в рамках бізнес-вимог. Іноді вони називаються вимогами поведінки (behavioral requirements), вони містять положення з традиційним «повинен» або «повинна»: «Система повинна по електронній пошті відправляти користувачеві підтвердження про замовлення».

Функціональні вимоги документуються в специфікації вимог до ПЗ (software requirements specification, SRS), де описується так повно, як необхідно, очікуване поведінка системи.

Функціональні вимоги. Це перелік сервісів, які повинна виконувати система, причому має бути вказано, як система реагує на ті чи інші вхід-ні дані, як вона поводиться в певних ситуаціях і т.д. У деяких випадках вказується, що система не повинна робити.

Ці вимоги описують поведінку системи і сервіси (функції), які вона виконує, і залежать від типу системи, що розробляється і від потреб користувачів. Якщо функціональні вимоги оформлені як призначені для користувача, вони, як правило, описують системи в узагальненому вигляді. На противагу цьому функціональні вимоги, оформлені як системні, описують систему максимально детально, включаючи її вхідні і вихідні дані, виключення і т.д.

Функціональні вимоги для програмних систем можуть бути описані різними способами.

Функціональні вимоги програмного засобу «Програмний модуль для роботи з зображенням» наступні:

1. Користувач повинен мати можливість вибору розмірності поля.

2. Можливість виконання елементарних операцій із об’єктом.

3. Можливість експорту даних.

До нефункціональних вимог програмного додатку відносять атрибути якості та зовнішній інтерфейс.

Атрибутами якості є наступні характеристики програмного продукту:

1) легкість;

2) простота використання;

3) простота переміщення;

4) цілісність;

5) ефективність;

6) стійкість до збоїв.

Вимоги до зовнішнього інтерфейсу:

1) сумісний з потребами та можливостями користувача;

2) реакція системи на всі типи запитів також повинна бути однозначною і зрозумілою, простою;

3) максимальна простота його використання і готовність в повній мірі задовольнити запити користувача;

4) вивід різних компонентів системи повинен здійснюватися в новому вікні.

Системні вимоги:

1) сумісний з операційними системами Windows 7, Windows 8 та Windows 10;

2) портативний модуль (з вихідними кодами) повинен займати не більше 200 МБ дискового простору;

3) модуль повинен коректно працювати на апаратному забезпеченні з обмеженими ресурсами.

2.3 Структура програмного модуля

Процес проектування складного програмного забезпечення починають з уточнення його структури, тобто визначення структурних компонент та зв'язків між ними. Результат уточнення структури може бути представлений у вигляді структурної та / або функціональної схем і опису (специфікацій) компонентів.

Структурна схема розроблюваного програмного забезпечення. Структурною називають схему, яка відображає склад і взаємодію з управління частин розроблюваного програмного забезпечення.

Структурні схеми пакетів програм не інформативні, оскільки організація програм в пакети не передбачає передачі управління між ними. Тому структурні схеми розробляють для кожної програми пакету, а список програм пакету визначають, аналізуючи функції, зазначені в технічному завданні.

Найпростіший вид програмного забезпечення - програма, яка в якості структурних компонентів може включати тільки підпрограми та бібліотеки ресурсів. Розробку структурної схеми програми зазвичай виконують методом покрокової деталізації.

Структурними компонентами програмної системи або програмного комплексу можуть бути програми, підсистеми, бази даних, бібліотеки ресурсів.

Структурна схема програмного комплексу демонструє загальну роботу програмного засобу створення елементарних растрових зображень (рис 2.2).

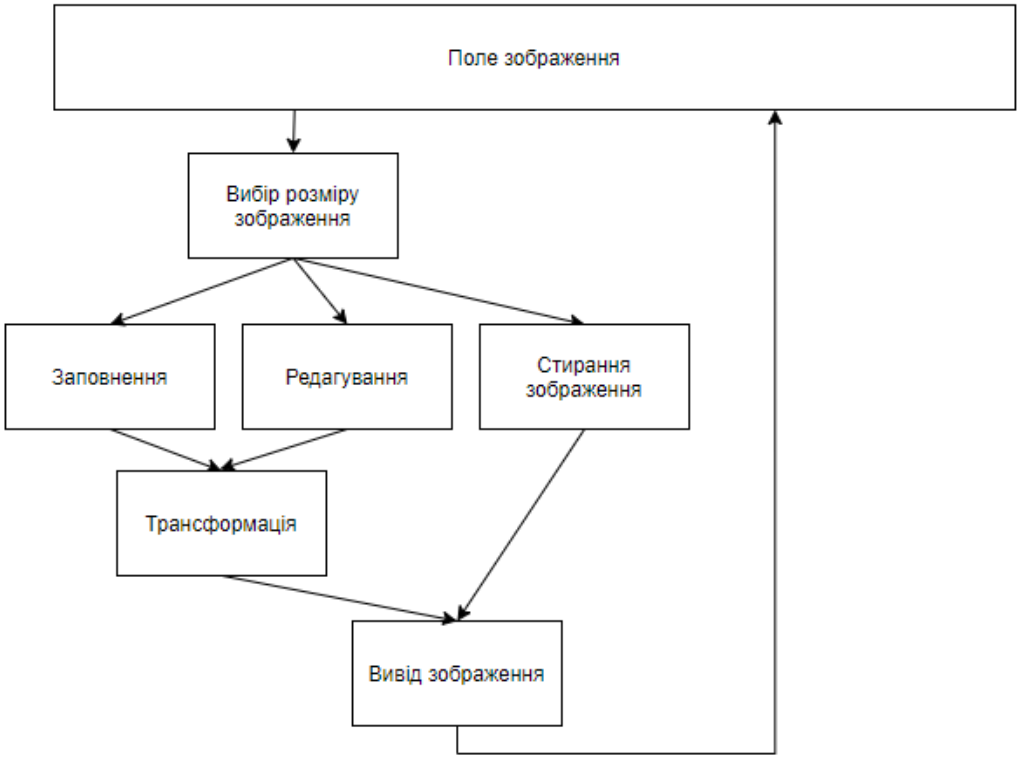

Рисунок 2.2 – Структурна схема програмного засобу

Схема складається з наступних компонентів:

1) поле зображення, доступне для змін;
2) введення діапазону поля зображення;
3) заповнення поля;
4) редагування частини поля;
5) очищення поля;
6) трансформація зображення;
7) вивід зображення на екран.

UML (англ. Unified Modeling Language - уніфікована мова моделювання) - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

UML забезпечує три представлення моделі системи:

Подання функціональних вимог (функціональні вимоги системи з точки зору користувача, включаючи варіанти використання);

Статична структурне уявлення (об'єкти, атрибути, відносини і операції, включаючи діаграми класів)

Подання динамічної поведінки (взаємодія об'єктів і зміни внутрішнього стану об'єктів, включаючи діаграми послідовностей, діяльностей і станів).

Діаграма класів показують набір класів, інтерфейсів, а також їх зв'язку. Діаграми цього виду найчастіше використовуються для моделювання об'єктно-орієнтованих систем. Вони призначені для статичного представлення системи.

Більшість елементів UML мають унікальну і пряму графічну нотацію, яка дає візуальне уявлення найбільш важливих аспектів елемента.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

Діаграма класів системи управління персоналом представлена на рис. 2.3.

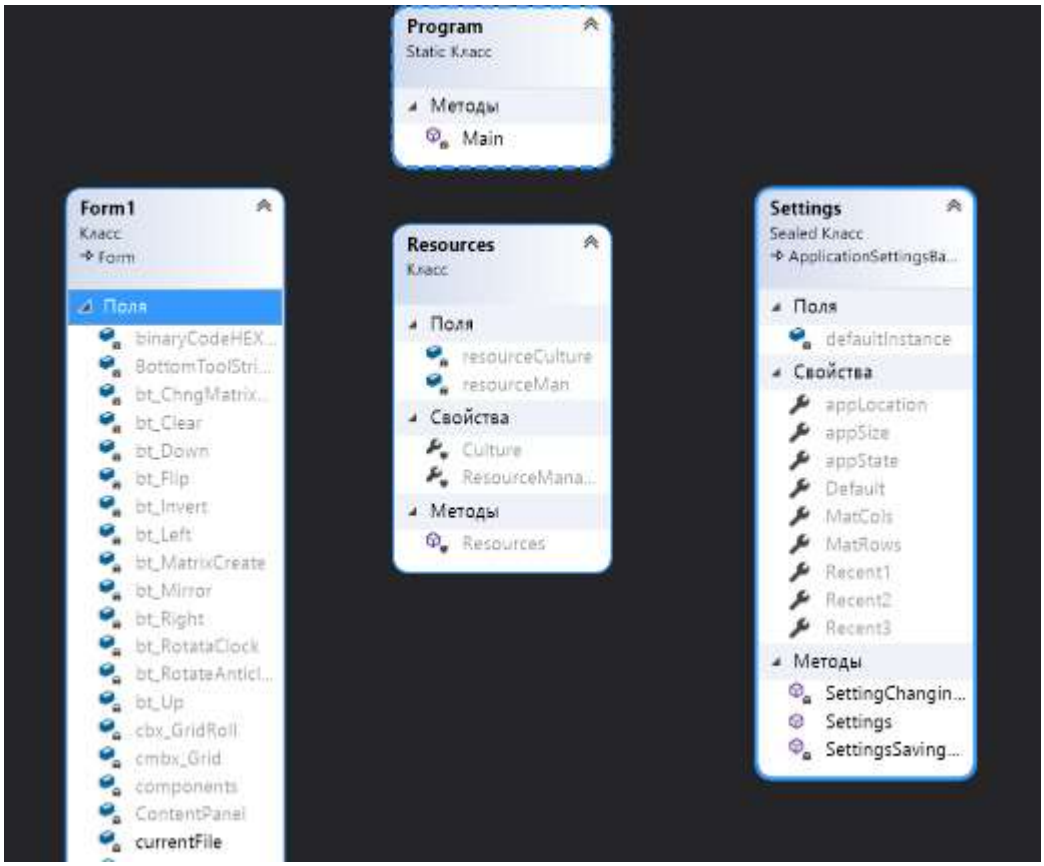

Рисунок 2.3 – Діаграма класів

Суть діаграми варіантів використання полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, які взаємодіють з системою за допомогою, так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на модельовану систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає акторові. Іншими словами, кожен варіант використання визначає певний набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодія акторів із системою. Діаграма варіантів використання програмного засобу зображена на рисунку 2.4.

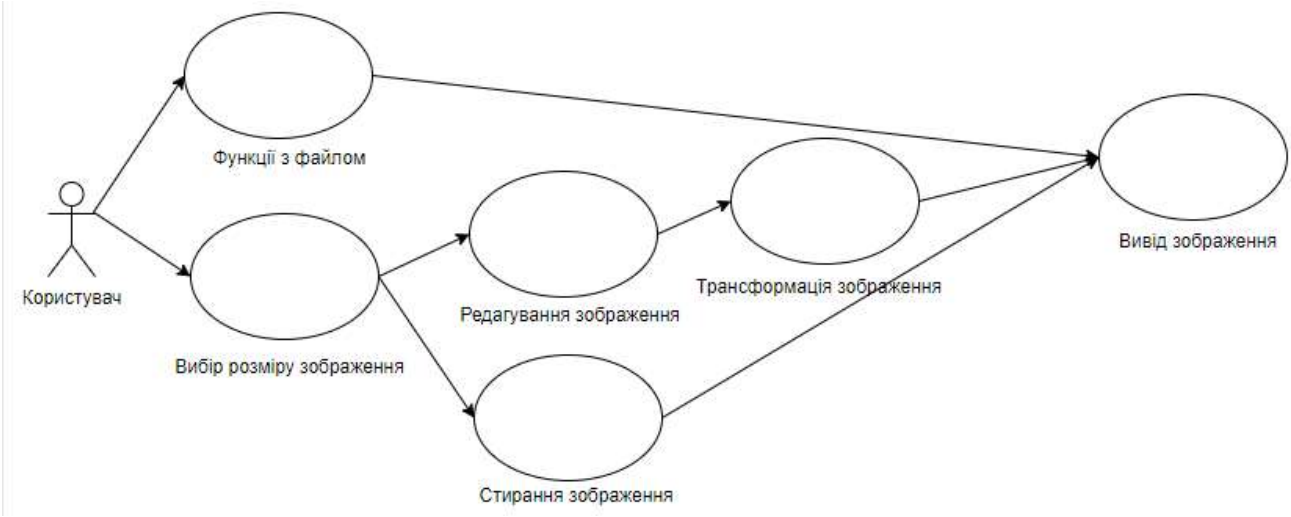

Рисунок 2.4 – Діаграма прецедентів програмного засобу

Діаграма послідовності програмного засобу створення растрових зображень відображає події об’єктів, впорядковані за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.

Іншими словами, діаграма послідовностей відображає часові особливості передачі і прийому повідомлень об'єктами.

Діаграми послідовностей можна використовувати для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію,

повідомлення, якими вони обмінюються, і які повертаються результати, які пов'язані з повідомленнями.

В UML взаємодія елементів розглядається в інформаційному аспекті їх комунікації, тобто взаємодіючі об'єкти обмінюються між собою деякою інформацією. При цьому інформація приймає форму закінчених повідомлень. Іншими словами, хоча повідомлення і має інформаційний зміст, воно набуває додаткове властивість надавати направлений вплив на свого одержувача.

Це повністю узгоджується з принципами ООАП, коли будь-які види інформаційної взаємодії між елементами системи повинні бути зведені до відправки і прийому повідомлень між ними. Для моделювання взаємодії об'єктів у мові UML використовуються відповідні діаграми взаємодії. Говорячи про ці діаграмах, мають на увазі два аспекти взаємодії.

По-перше, взаємодії об'єктів можна розглядати в часі, і тоді для подання часових особливостей передачі і прийому повідомлень між об'єктами використовується діаграма послідовності. Цей вид канонічних діаграм є предметом вивчення цієї глави. Раніше, при вивченні діаграм стану та діяльності, було відзначено одна важлива обставина. Хоча розглянуті діаграми і використовуються для специфікації динаміки поведінки систем, час в явному вигляді в них не присутня. Проте часовий аспект поведінки може мати суттєве значення при моделюванні синхронних процесів, що описують взаємодії об'єктів. Саме для цієї мети в мові UML використовуються діаграми послідовності.

По-друге, можна розглядати структурні особливості взаємодії об'єктів. Для представлення структурних особливостей передачі і прийому повідомлень між об'єктами використовується діаграма кооперації.

На діаграмі послідовності зображуються виключно ті об'єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні асоціації з іншими об'єктами. Для діаграми послідовності ключовим моментом є саме динаміка взаємодії об'єктів у часі. При цьому діаграма послідовності має як би два виміри. Одне - зліва направо у вигляді вертикальних ліній, кожна з яких зображає лінію життя окремого об'єкта, який бере участь у взаємодії. Графічно

кожен об'єкт зображується прямокутником і розташовується у верхній частині своєї лінії життя. Усередині прямокутника записуються ім'я об'єкта і ім'я класу, розділені двокрапкою. При цьому вся запис підкреслюється, що є ознакою об'єкта, який, як відомо, представляє собою екземпляр класу.

Діаграма послідовності програмного засобу представлена на рисунку 2.5.

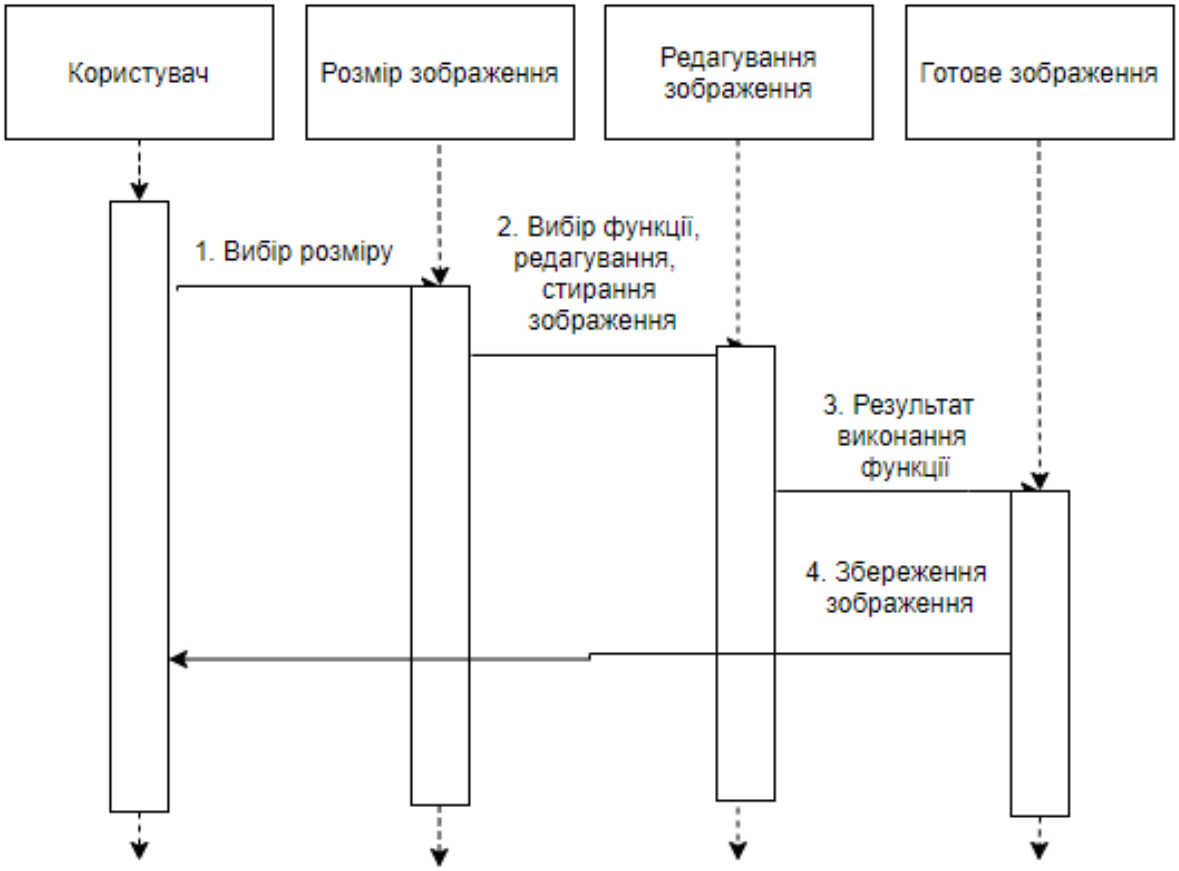

Рисунок 2.5 – Діаграма послідовності програми

2.4 Опис засобів реалізації

Для реалізації системи використовуються такі основні класи, як Button, Label, menuStrip, toolTip, NumericUpDown, ComboBox, dataGridView.

Button - можна клацнути мишею, ввести клавішу або пробіл, якщо кнопка має фокус.

Встановіть AcceptButton властивість або CancelButton об'єкта, Form щоб дозволити користувачам натиснути кнопку, натиснувши клавіші Enter або ESC, навіть якщо кнопка не має фокусу. Це дає формі поведінку діалогового вікна.

При відображенні форми за допомогою ShowDialog методу можна використовувати DialogResult властивість кнопки, щоб вказати значення, що повертається ShowDialog.

Ви можете змінити вигляд кнопки. Наприклад, щоб зробити його плоским для веб-пошуку, задайте FlatStyle для властивості значення FlatStyle.Flat. FlatStyleСвойству також можна привласнити значення FlatStyle.Popup, яке відображається плоским до тих пір, поки курсор не знайде над кнопкою; потім кнопка приймає стандартний вид кнопки Windows.

Label елементи управління зазвичай використовуються для надання описового тексту для елемента управління. Наприклад, можна використовувати, Label щоб додати описовий текст для TextBox елемента управління, щоб інформувати користувача про тип даних, очікуваних в елементі управління. Label елементи управління також можна використовувати для додавання тексту опису в, Form щоб надати користувачеві корисну інформацію. Наприклад, можна додати Label в початок Form, що надає користувачеві інструкції щодо введення даних в елементи управління у формі. Label елементи управління можна також використовувати для виведення відомостей про час виконання в стані додатки. Наприклад, можна додати в Label форму елемент управління для перегляду стану кожного файлу у вигляді списку оброблюваних файлів.

MenuStrip - це контейнер верхнього рівня, який замінюється MainMenu. Він також забезпечує обробку ключів і можливості багатодокументного інтерфейсу (MDI). Функціонально ToolStripDropDownItem і ToolStripMenuItem працює разом з MenuStrip, хоча вони є похідними від ToolStripItem.

Наступні елементи спеціально призначені для ефективної роботи з обома ToolStripSystemRenderer ToolStripProfessionalRenderer орієнтаціями.

TextBox представляє елемент управління "текстове поле" Windows.

За допомогою TextBox елемента управління користувач може вводити текст в додатку. Цей елемент управління має додаткові функціональні можливості, які не знайдені в стандартному елементі управління "текстове поле Windows", включаючи многострочное редагування і маскування символів пароля.

За допомогою ToolTip класу можна надати вказівки користувачеві, коли користувач поміщає курсор миші на елемент керування. ToolTipКласс зазвичай використовується для оповіщення користувачів про передбачуване використанні елемента керування. Наприклад, можна вказати текст підказки для TextBox елемента управління, який приймає ім'я, вказавши формат імені, вводиться в елемент керування. Крім вказівки, можна також використовувати ToolTip клас для надання відомостей про стан часу виконання. Наприклад, можна використовувати ToolTip клас для відображення швидкості підключення і даних якості лінії, коли користувач переміщує курсор миші на PictureBox елемент управління, що відображає стан підключення до Інтернету.

NumericUpDownЕлемент управління містить одне числове значення, яке можна збільшити або зменшити, натиснувши кнопки зі стрілками вгору або вниз. Користувач також може ввести значення, якщо тільки ReadOnly властивість не має значення true.

Щоб додати або видалити об'єкти в списку під час виконання, використовуйте методи ComboBox.ObjectCollection класу (за допомогою Items властивості ComboBox). Можна призначити масив посилань на об'єкт за допомогою AddRange методу. Потім в списку відображається значення рядка за замовчуванням для кожного об'єкта. Можна додати окремі об'єкти за допомогою Add методу. Можна видалити елементи за допомогою Remove методу або очистити весь список за допомогою Clear методу.

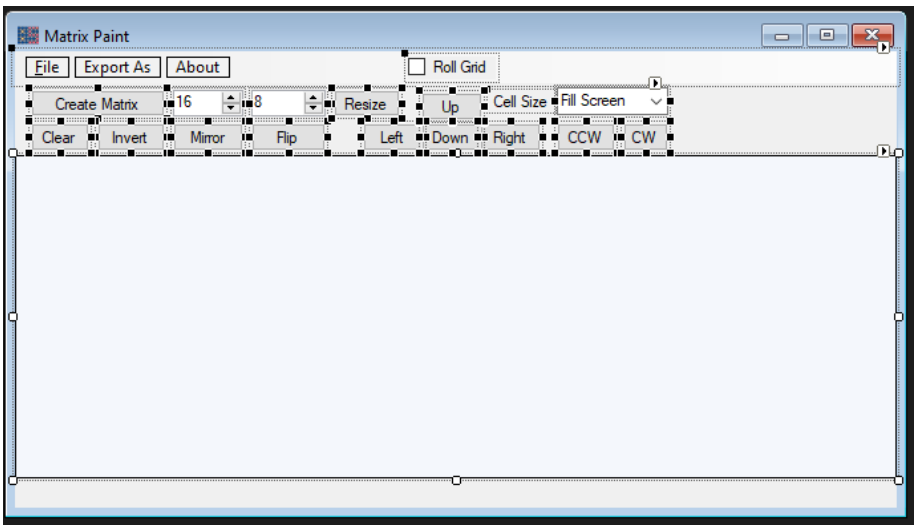

Рисунок 2.6 – Конструктор з основними засобами реалізації

Приклади основних методів, які використовуються для реалізації функцій програмного засобу описані нижче.

Даний метод дозволяє здійснити ініціалізацію основних даних.

```
    private void Form1_Load(object sender, EventArgs e)
{
    this.Location = Properties.Settings.Default.appLocation;
    this.WindowState = Properties.Settings.Default.appState;
    this.Size = Properties.Settings.Default.appSize;
    num_Col.Value = Properties.Settings.Default.MatCols;
    num_Row.Value = Properties.Settings.Default.MatRows;
    recent1toolStripMenuItem.Text = Properties.Settings.Default.Recent1;
    recent2toolStripMenuItem.Text = Properties.Settings.Default.Recent2;
    recent3toolStripMenuItem.Text = Properties.Settings.Default.Recent3;
    bt_RotataClock.Enabled = false;
    bt_RotateAnticlock.Enabled = false;
    if (matrixFileName != null)
    {
        OpenMatrix(matrixFileName);
        currentFile = FileStatus.Open;
        isChanged = false;
    }
    else
    {
        currentFile = FileStatus.New;
        NewMatrix();
    }
    UpdateMatrixCellSize();

    // Tooltip
    System.Windows.Forms.ToolTip ToolTip1 = new System.Windows.Forms.ToolTip();
    ToolTip1.SetToolTip(this.num_Col, "Select the Width");
    ToolTip1.SetToolTip(this.num_Row, "Select the Height");
    ToolTip1.SetToolTip(this.cmbx_Grid, "Select the size of the block");
    ToolTip1.SetToolTip(this.bt_MatrixCreate, "To create and display the matrix of provided parameters");
    ToolTip1.SetToolTip(this.bt_Clear, "Clear the matrix data");
    ToolTip1.SetToolTip(this.bt_Invert, "Invert the matrix");
    ToolTip1.SetToolTip(this.bt_Mirror, "Create the mirror of the matrix");
    ToolTip1.SetToolTip(this.bt_Flip, "Turn over the matrix");
    ToolTip1.SetToolTip(this.bt_RotateAnticlock, "Rotate anticlockwise by 90 deg");
    ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
    ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
```

```
        ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
        ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
        ToolTip1.SetToolTip(this.cbx_GridRoll, "Choose whether to rotate/roll the grid");
        ToolTip1.SetToolTip(this.bt_Up, "Move up in the grid");
        ToolTip1.SetToolTip(this.bt_Down, "Move down in the grid");
        ToolTip1.SetToolTip(this.bt_Left, "Move left in the grid");
        ToolTip1.SetToolTip(this.bt_Right, "Move right in the grid");
    }
```

Наступний описаний метод виконується при закритті вікна, він виконує перевірку, чи збережені дані, та завершує усі поточні процеси.

```
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (isChanged == true && currentFile == FileStatus.New)
        {
            DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
            if (z == DialogResult.Yes)
                SaveMatrix();
            else if (z == DialogResult.Cancel)
                e.Cancel = true;
        }
        else if (isChanged == true && currentFile == FileStatus.Open)
        {
            DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
            if (z == DialogResult.Yes)
                SaveOpenMatrix();
            else if (z == DialogResult.Cancel)
                e.Cancel = true;
        }
        Properties.Settings.Default.appLocation = this.Location;
        Properties.Settings.Default.appState = this.WindowState;
        Properties.Settings.Default.appSize = this.Size;
        Properties.Settings.Default.MatCols = Convert.ToInt16(num_Col.Value);
        Properties.Settings.Default.MatRows = Convert.ToInt16(num_Row.Value);
        Properties.Settings.Default.Recent1 = recent1toolStripMenuItem.Text;
        Properties.Settings.Default.Recent2 = recent2toolStripMenuItem.Text;
        Properties.Settings.Default.Recent3 = recent3toolStripMenuItem.Text;
        Properties.Settings.Default.Save();
    }
```

Метод здійснює зміну розміру зображення.

```
        private void Form1_Resize(object sender, EventArgs e)
    {
      if (this.WindowState == FormWindowState.Minimized)
      {
        notifyIcon1.Visible = true;
        this.ShowInTaskbar = false;
      }
      else
      {
        notifyIcon1.Visible = false;
        this.ShowInTaskbar = true;
      }
    }
```

Метод дозволяє відкрити або створити матрицю зображення.

```
        private void bt_MatrixCreate_Click(object sender, EventArgs e)
    {
      if (isChanged == true && currentFile == FileStatus.New)
      {
        DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
        if (z == DialogResult.Yes)
        {
          SaveMatrix();
          NewMatrix();
        }
        else if (z == DialogResult.No)
          NewMatrix();
      }
      else if (isChanged == true && currentFile == FileStatus.Open)
      {
        DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
        if (z == DialogResult.Yes)
        {
          SaveOpenMatrix();
          NewMatrix();
        }
        else if (z == DialogResult.No)
          NewMatrix();
      }
      else if (isChanged == false)
        NewMatrix();      }
```

Метод дозволяє заповнити матрицю білим або чорним кольором.

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
  if (dataGridView1.CurrentCell.Style.BackColor != Color.Black)
    dataGridView1.CurrentCell.Style.BackColor = Color.Black;
  else
    dataGridView1.CurrentCell.Style.BackColor = Color.White;
  dataGridView1.CurrentCell.Selected = false;
  isChanged = true;
}
```

Наступний метод виконує вивід зображення певного розміру (в залежності від обраного) на екран.

```
private void UpdateMatrixCellSize()
{
  int RowHeight = 10, ColumnWidth = 10;
  switch (cmbx_Grid.SelectedIndex)
  {
    case 0:
      // Fill Screen
      if (dataGridView1.RowCount > 0)
        RowHeight = dataGridView1.Height / dataGridView1.RowCount;
      if (dataGridView1.ColumnCount > 0)
        ColumnWidth = dataGridView1.Width / dataGridView1.ColumnCount;
      break;
    case 1:
      // Tiny
      RowHeight = 12;
      ColumnWidth = 12;
      break;
    case 2:
      // Medium
      RowHeight = 22;
      ColumnWidth = 22;
      break;
    case 3:
      // Large
      RowHeight = 42;
      ColumnWidth = 42;
      break;
    case 4:
      // HUGE
      RowHeight = 62;
      ColumnWidth = 62;
      break;
    default:
```

```
            break;
        }

        for (int z = 0; z < dataGridView1.RowCount; z++)
            dataGridView1.Rows[z].Height = RowHeight;
        for (int z = 0; z < dataGridView1.ColumnCount; z++)
            dataGridView1.Columns[z].Width = ColumnWidth;
    }
```

2.5 Порівняльний аналіз реалізованого програмного засобу та програм-аналогів

Існує велика кількість програм для роботи з зображеннями, двомірної і тривимірною графікою, кресленнями, як для професіоналів, так і для звичайних користувачів. Нижче представлені найбільш популярні з них.

ibis Paint X - це фантастичне мобільний додаток, який включає в себе все необхідне для малювання (рис.2.7).

Функції:

1) понад 140 кистей;
2) кисть, ластик, функція розмиття, палітра кольорів і інші інструменти;
3) завантажуйте відео з вашими роботами, адже вони автоматично записуються;
4) сумісність з стилусами;
5) регульований стабілізатор лінії;
6) інструмент Шар;
7) діліться своїми роботами з іншими користувачами ibisPaint.

Рисунок 2.7 – Програмний продукт ibisPaint

Clip Studio Paint, раніше Manga Studio або ComicStudio в Японії, являє собою сімейство програмних додатків для Mac OS X і Microsoft Windows, які використовуються для цифрового створення коміксів і манги.

Він надає потужні передові інструменти для малювання і розфарбовування, що робить його незамінним для професійних художників коміксів і манги.

Незважаючи на те, що він має загальні функції з графічним програмним забезпеченням загального призначення, таким як Adobe Photoshop, набір інструментів Clip Studio Paint орієнтований і оптимізований для використання при створенні коміксів і манги. У ньому є інструменти для створення макетів панелей,

розмітки перспективи, створення ескізів, малювання, застосування тонів і текстур, розмальовки та створення спливаючих підказок і написів (рис. 2.8).

Підтримує створення растрових і векторних зображень, а також імпорт 3D-моделей, введення за допомогою миші або графічного планшета.

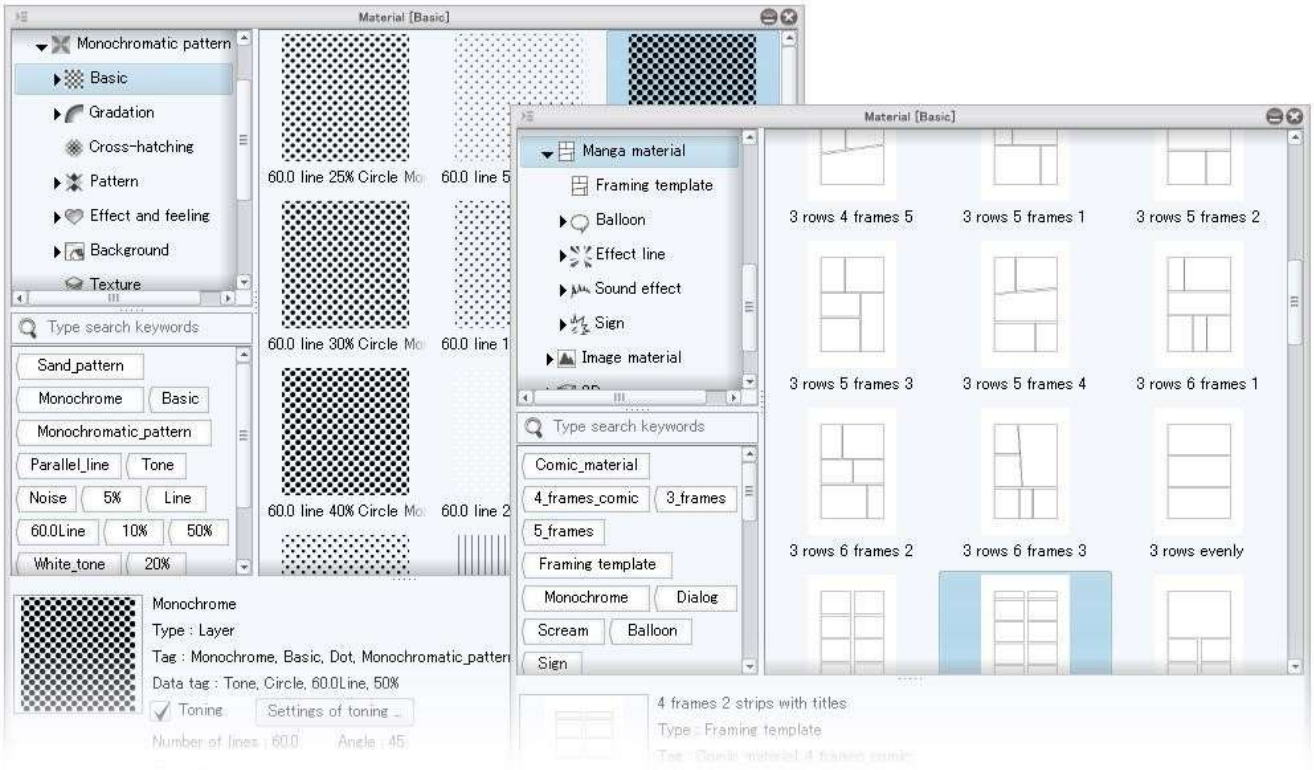

Рисунок 2.8 – Програмний засіб Clip Studio Paint

FireAlpaca - це редактор зображень, який можна легко і просто використовувати. Ми зробили можливою безкоштовне завантаження, щоб наші продукти могло випробувати як можна більшу кількість користувачів з усього світу.

Він доступний для користувачів Mac і Windows! Ми хотіли б, щоб ви вперше відчули прекрасні можливості, пропоновані FireAlpaca (рис. 2.9).

Рисунок 2.9 – Програмний засіб FireAlpaca

Програмний засіб «Програмний модуль для роботи з зображенням» – програмний засіб створення елементарних растрових зображень. Програмне забезпечення дозволяє створювати та редагувати найелементарніші растрові зображення за допомогою найпростіших функцій, таких як: повернути, пересунути вверх, вниз, направо, наліво, змінити розмір, заповнити, очистити, інвертувати, відобразити дзеркально. Також програмний засіб має зручний та зрозумілий інтерфейс (рис. 2.10).

Програмний засіб може використовуватися школярами та студентами для навчання, вчителями та викладачами у професійній діяльності, а також усіма зацікавленими особами для саморозвитку. Програмний засіб доступний для завантаження для усіх користувачів, є абсолютно безкоштовним та простим у використанні.

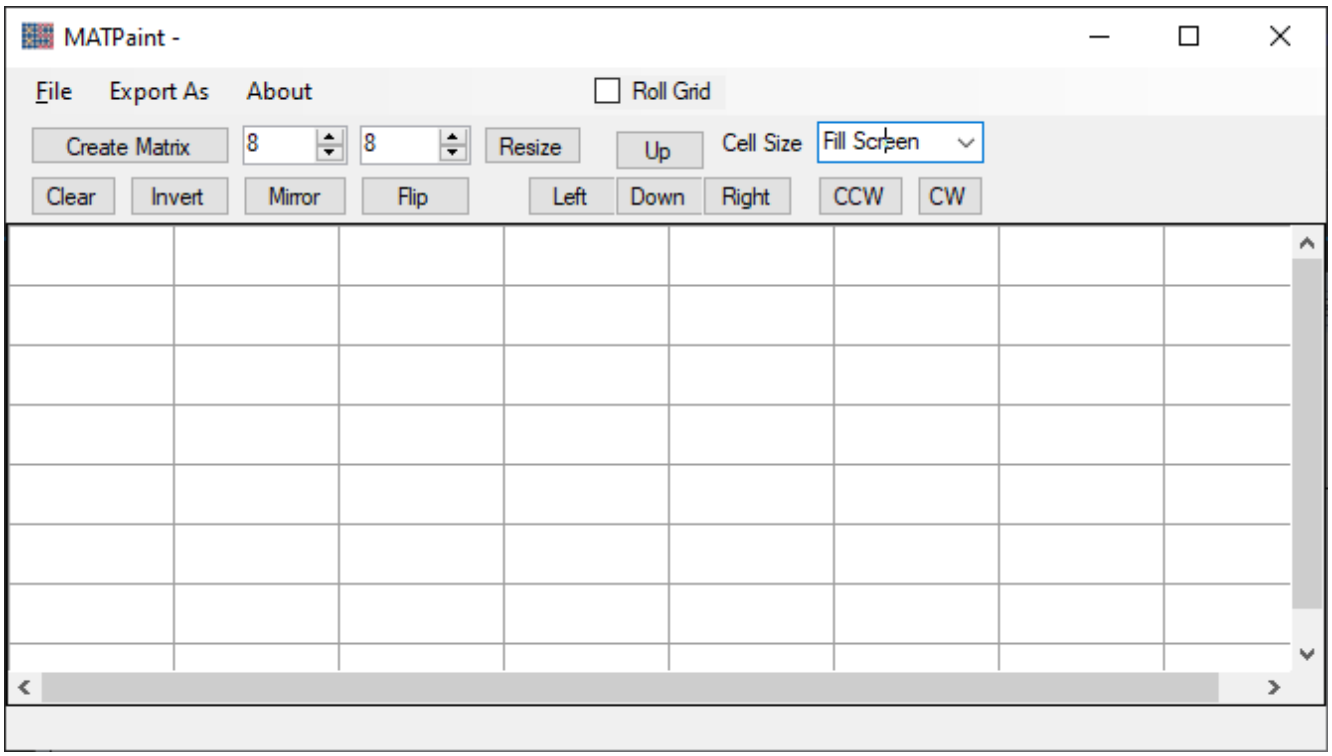

Рисунок 2.10 – Програмний засіб «Програмний модуль для роботи з зображенням»

Програма створення зображень «Програмний модуль для роботи з зображенням» має основні особливості та переваги:

1) вибір розмірності зображення;
2) заповнення матриці;
3) очищення матриці;
4) переміщення матриці;
5) дзеркальне відображення;
6) поворот;
7) інвертування;
8) зрозумілий інтерфейс;

Після огляду та порівняння програмних засобів різних програмних засобів для роботи із растровими зображеннями можна стверджувати, що найбільш простим та зручним у використанні є програмний засіб «Програмний модуль для роботи з зображенням».

Даний програмний засіб дозволяє використовувати вбудовані функції та обирати необхідний розмір зображення.

Програмний засіб може бути застосований у будь-якій галузі для полегшення роботи із зображеннями.

«Програмний модуль для роботи з зображенням» є абсолютно безкоштовним та знаходиться у вільному доступі в мережі Інтернет.

2.6 Керівництво користувача програмного модуля

Програмний засіб побудови графіків «Програмний модуль для роботи з зображенням» запускається подвійним кліком миші на файл MatPaint.exe (рисунок 2.11).

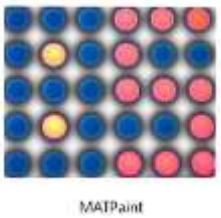

Рисунок 2.11 – Файл запуску програмного засобу

Після чого виконується відкриття головного вікна програми. За замовчуванням у верхній частині вікна розташована область меню, та панель з основними функціями. Основна частина вікна містить розмірну сітку певного розміру за замовчуванням (рисунок 2.12).

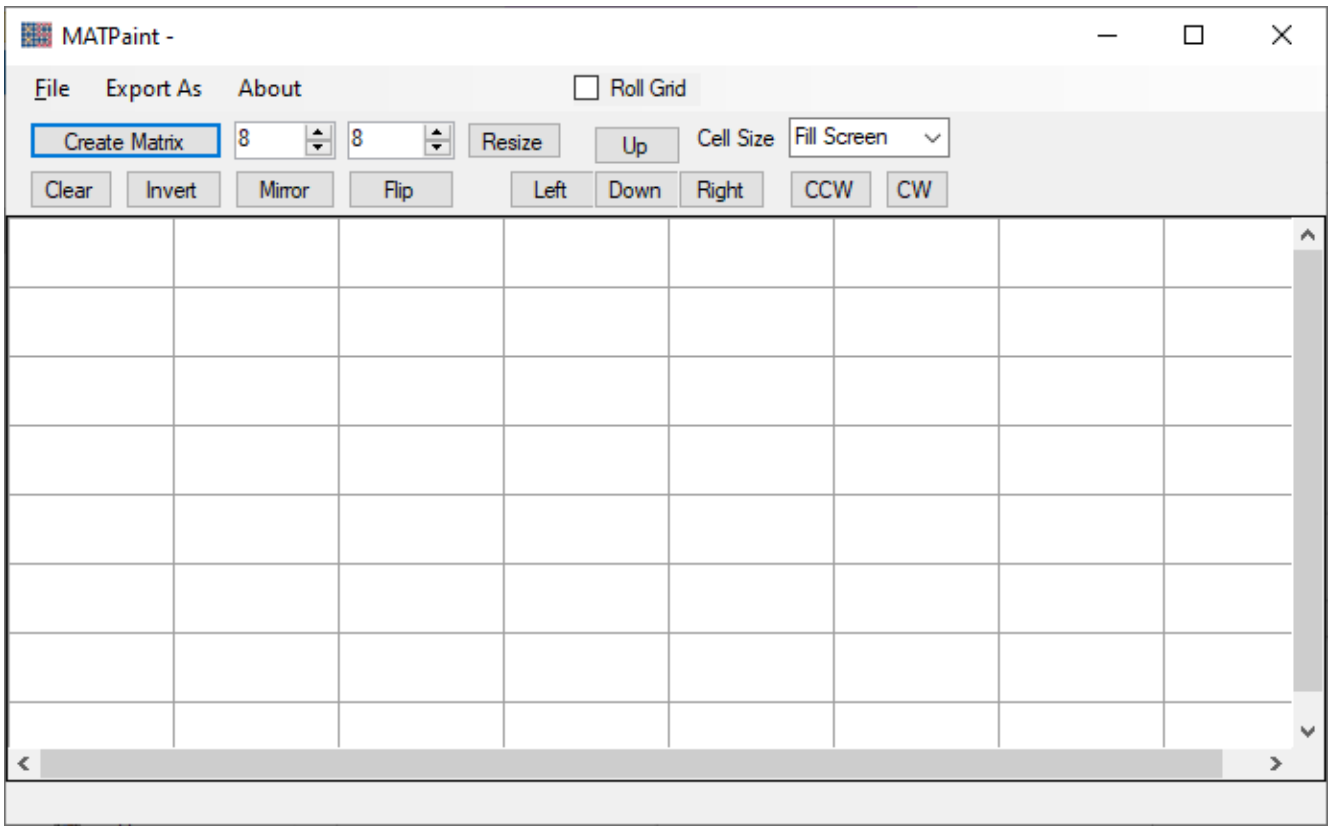

Рисунок 2.12 – Вікно програми

За допомогою кнопки «Створити матрицю» та заданих розмірів матриці виконується її відображення на екрані (рисунок 2.13).

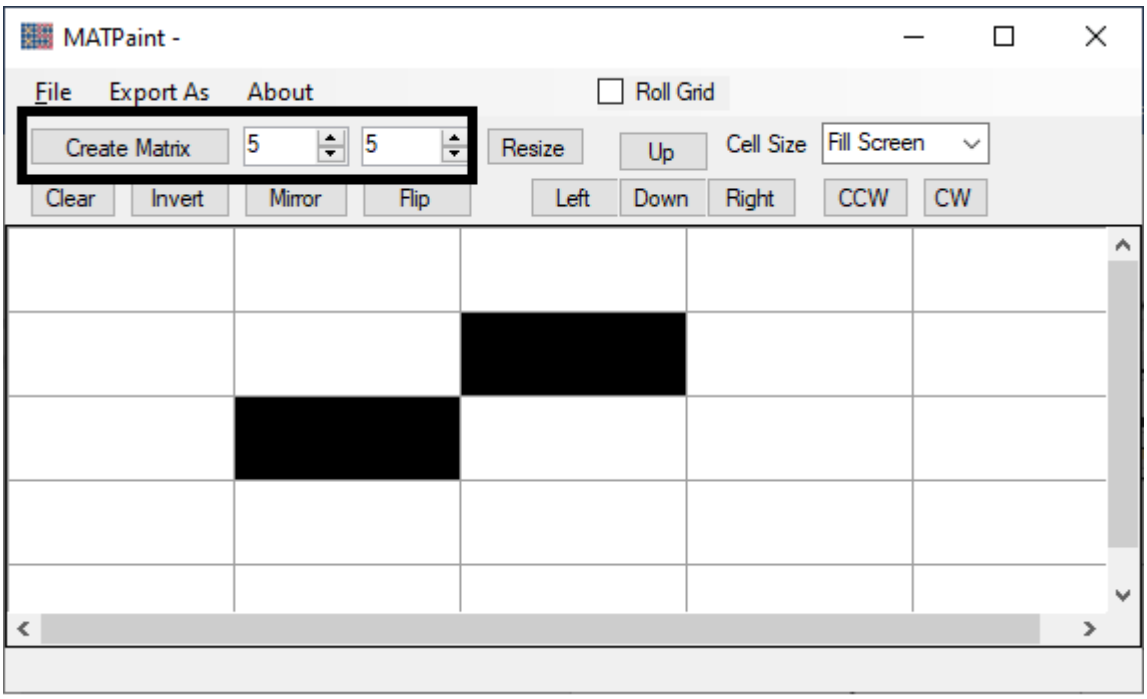

Рисунок 2.13 – Матриця із заданим розміром

Кнопки які знаходяться нижче дозволяють виконати очищення матриці, інвертування (рис. 2.14), дзеркальне відображення по вертикалі (рис. 2.15) та горизонталі відповідно (рис. 2.16).

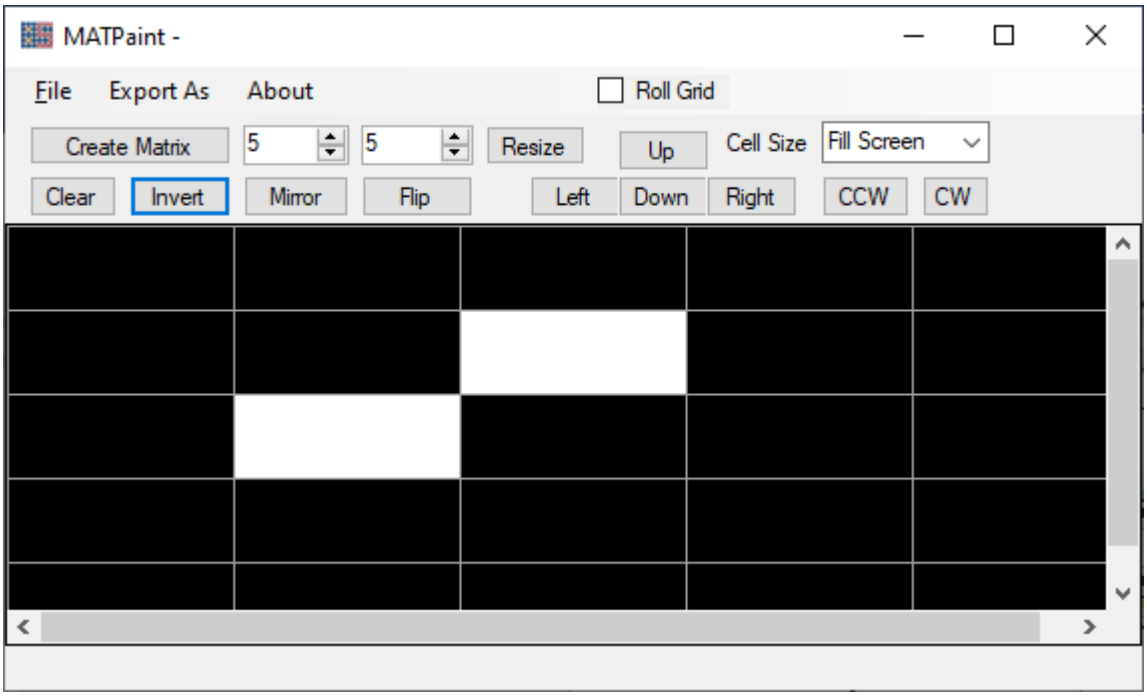

Рисунок 2.14 – Інвертування матриці

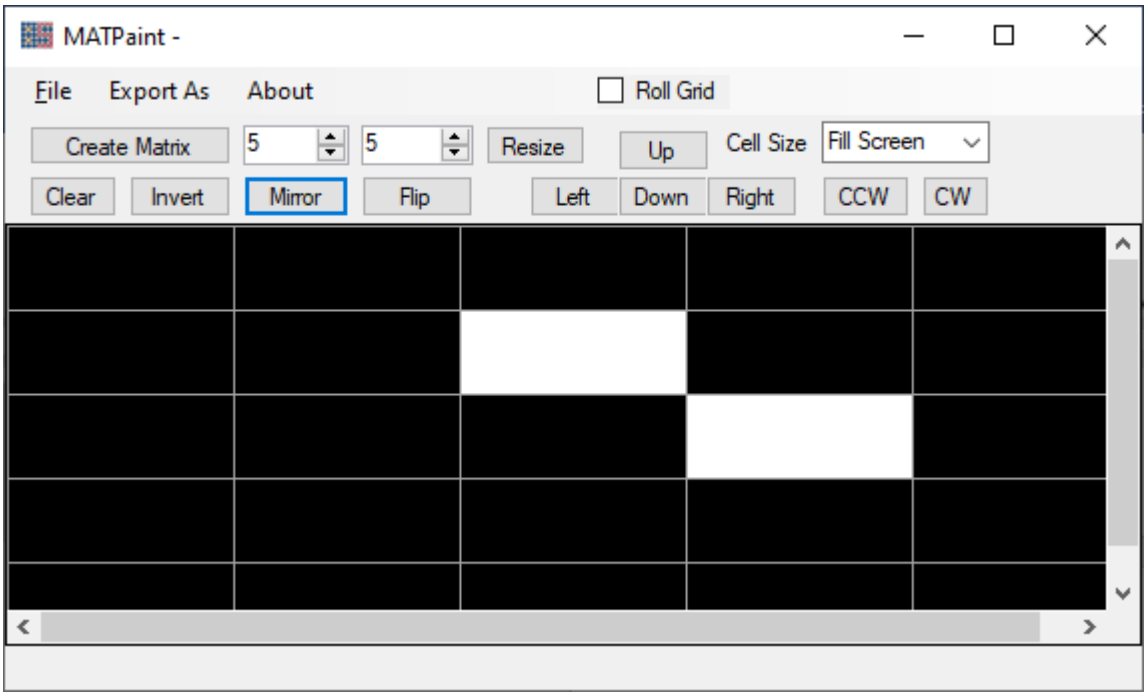

Рисунок 2.15 – Вертикальне відображення

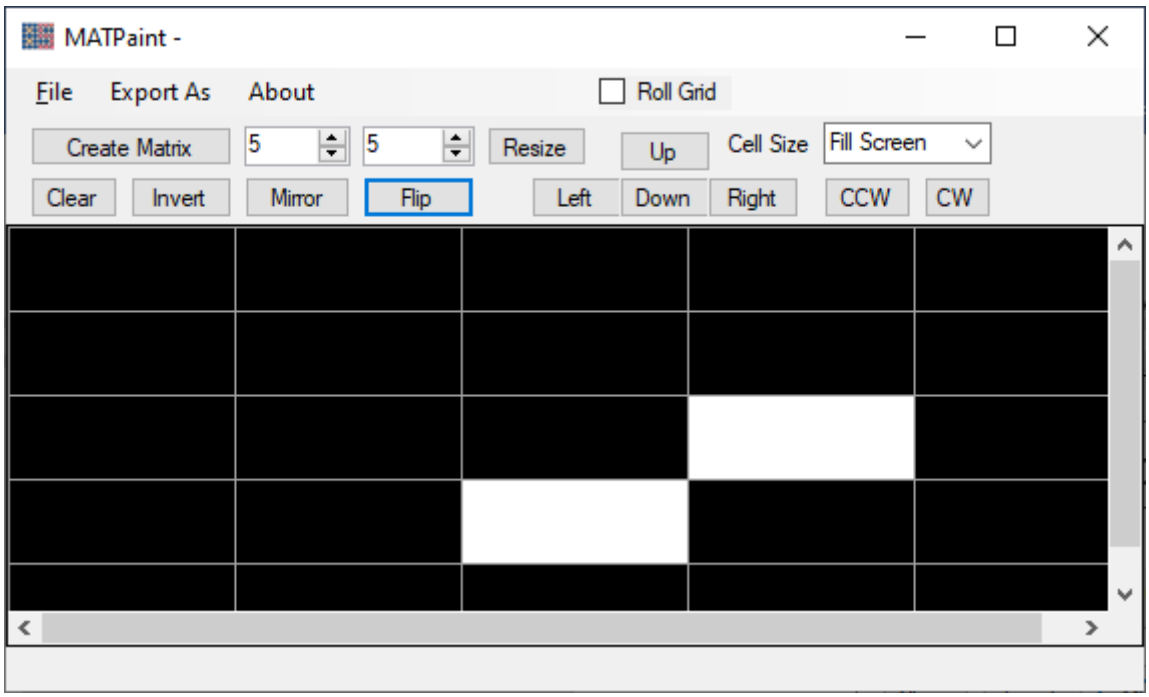

Рисунок 2.16 – Горизонтальне відображення

Кнопка «Змінити розмір» виконує додавання або стирання лінійок чи стовпців відповідно (рисунок 2.17).

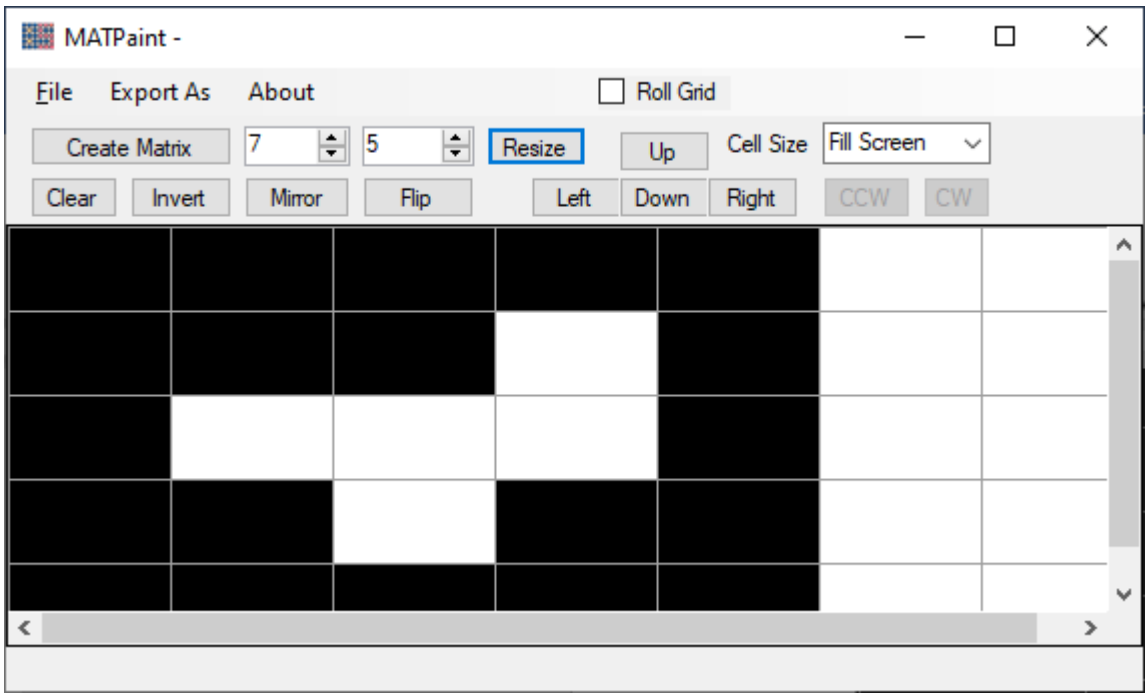

Рисунок 2.17 – Зміна розміру

Кнопки «Вверх», «Вниз», «Вліво», «Вправо» дозволяють пересунути зображення в певну сторону відповідно (рисунок 2.18).

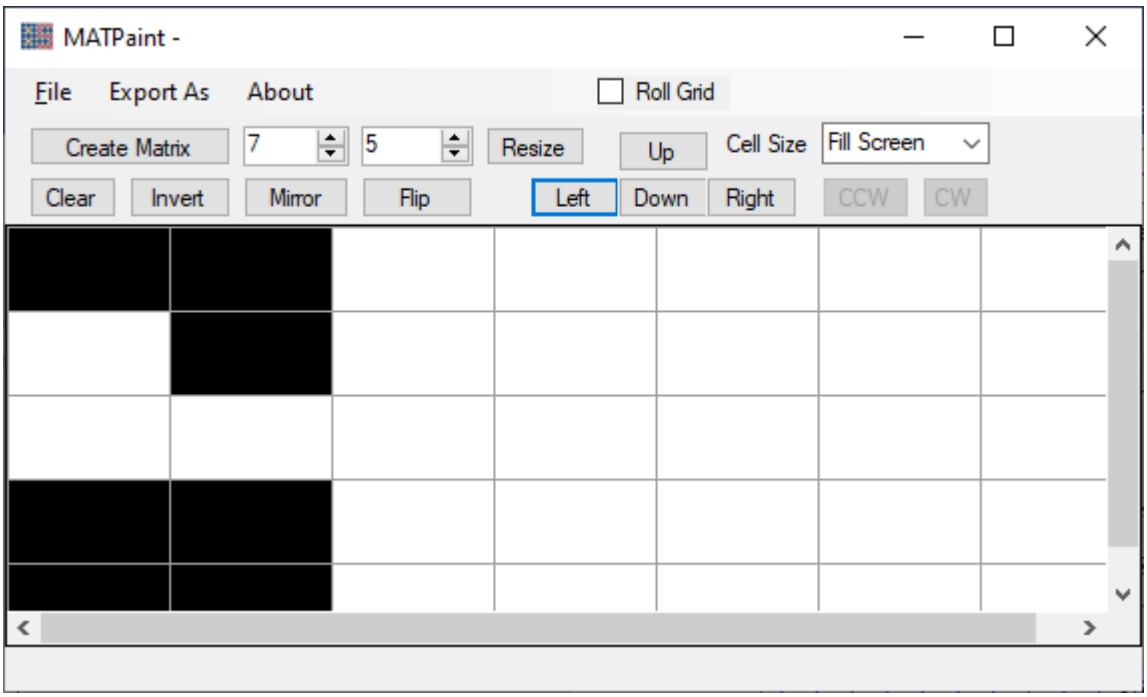

Рисунок 2.18 – Переміщення рисунку

Кнопка «Завернути таблицю» дозволяє при переміщенні рисунку робити зациклення (рисунок 2.19).

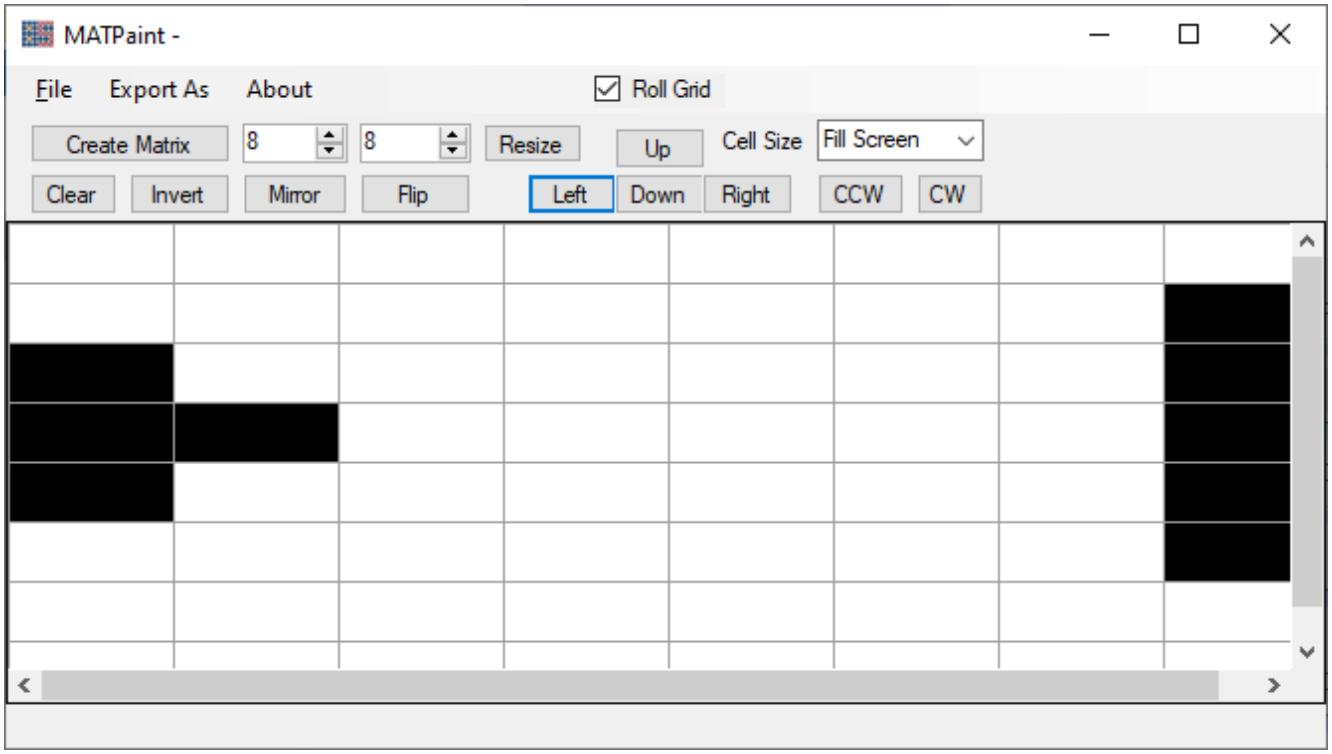

Рисунок 2.19 – Зациклення таблиці

Далі можна обрати режим заповнення комірками: заповнення екрану, мінімальний розмір (рис. 2.20), середній (рис. 2.21), великий (рис. 2.22), крупний (рис. 2.23). За замовчуванням встановлений розмір заповнення екрану.

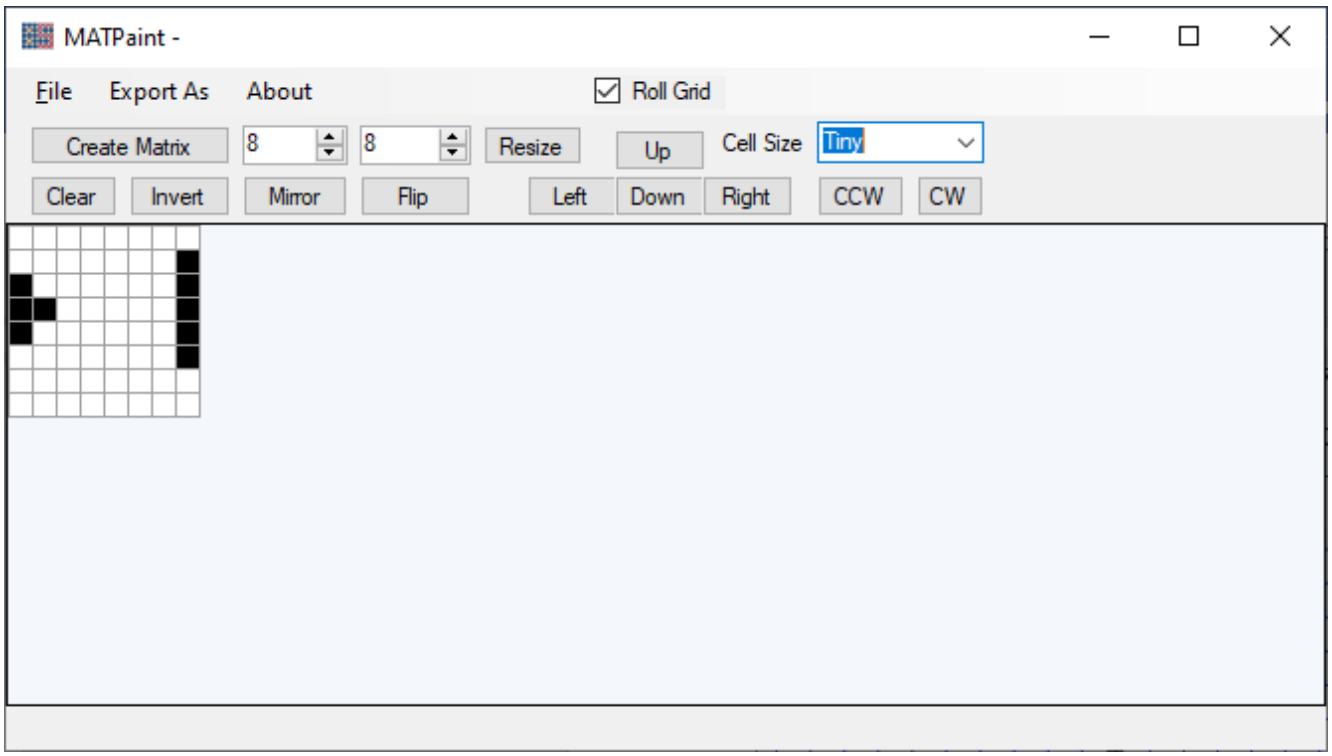

Рисунок 2.20 – Мінімальний розмір

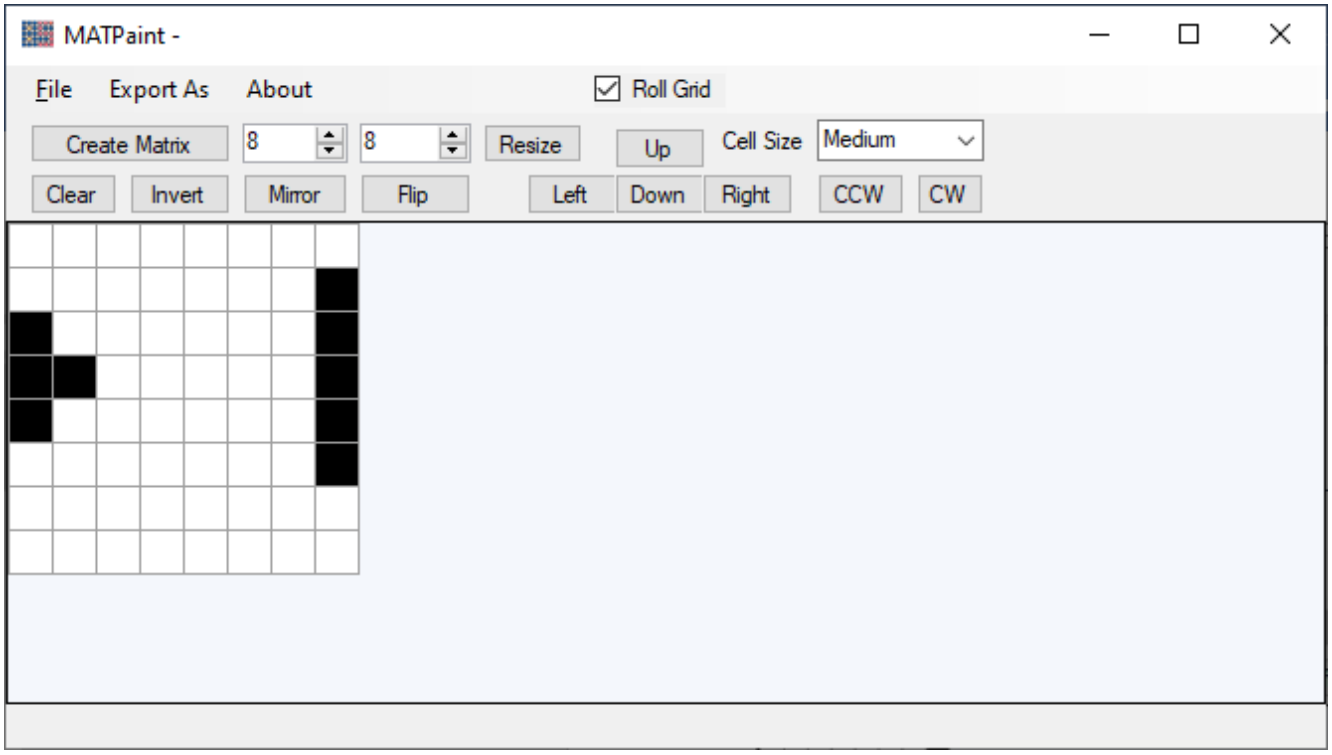

Рисунок 2.21 – Середній розмір

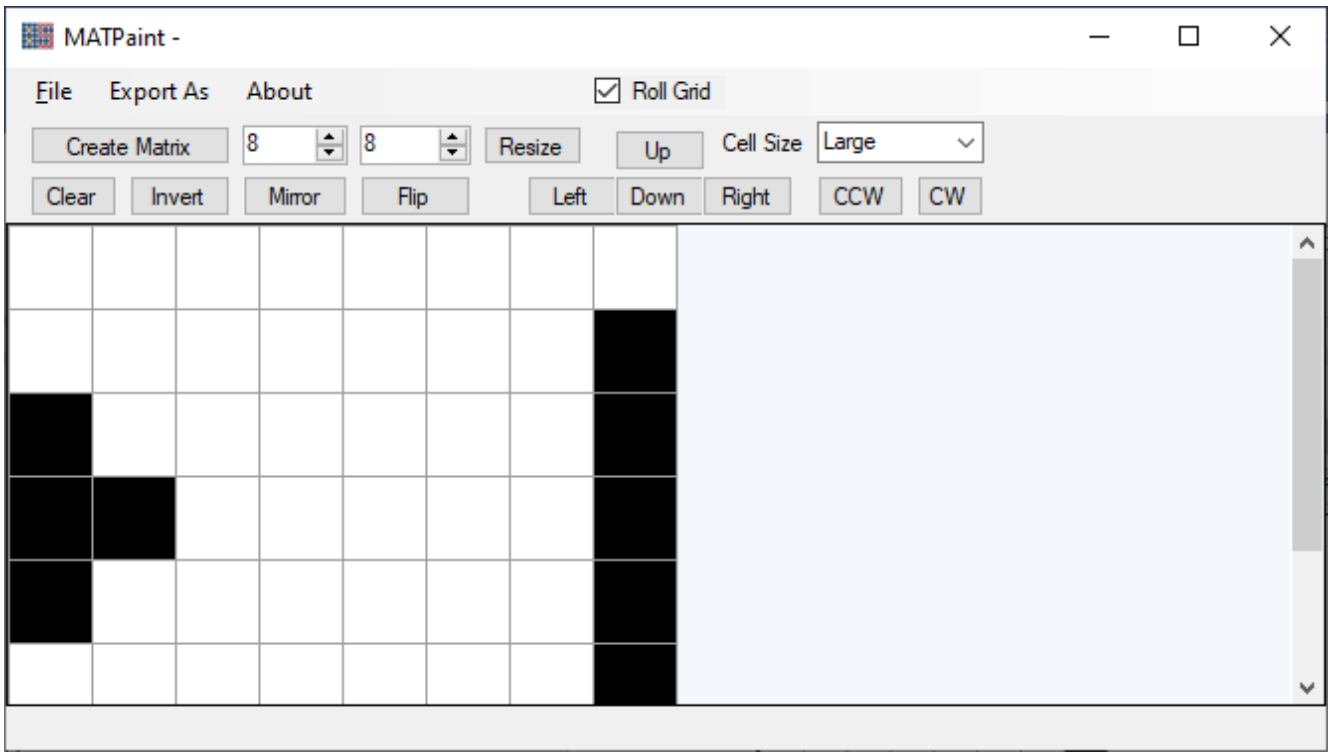

Рисунок 2.22 – Великий розмір

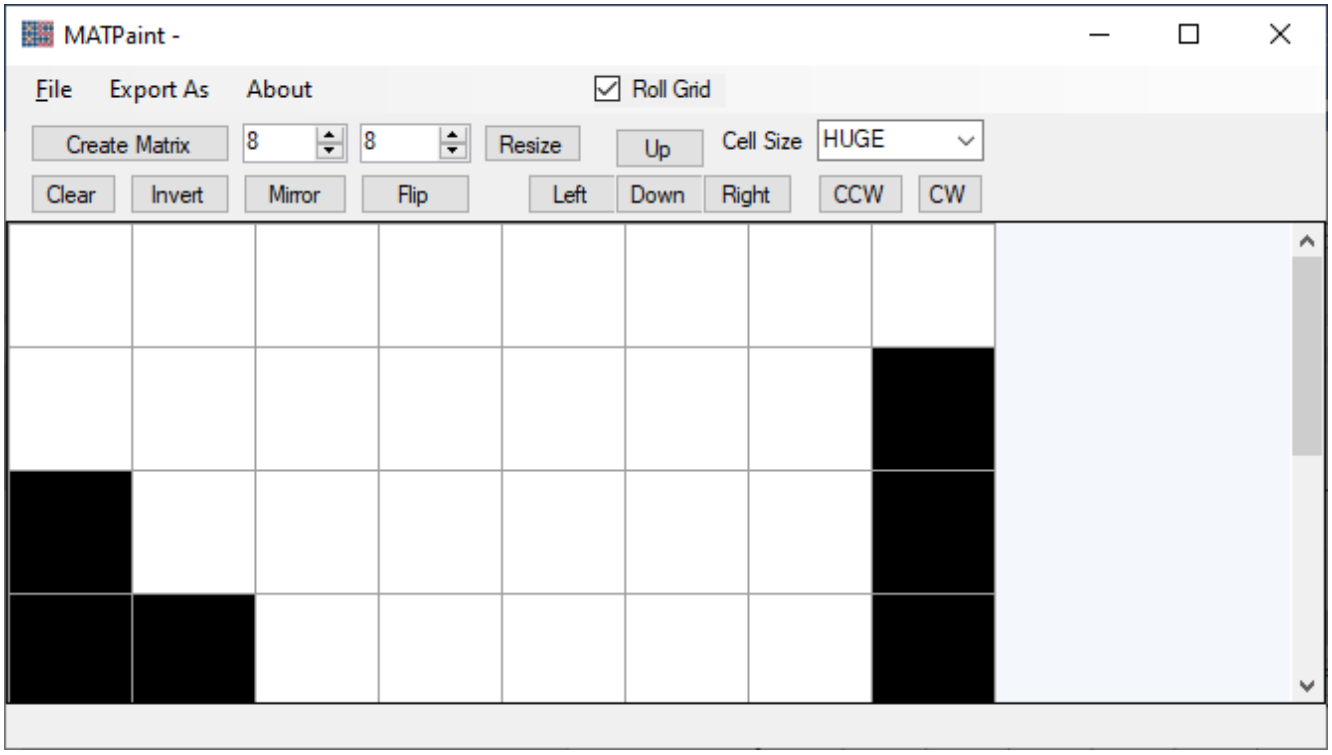

Рисунок 2.23– Крупний розмір

Кнопки «Повернути навпроти годинникової стрілки на 90°» та «Повернути за годинниковою стрілкою на 90°» виконують відповідні повороти (рис.2.24).

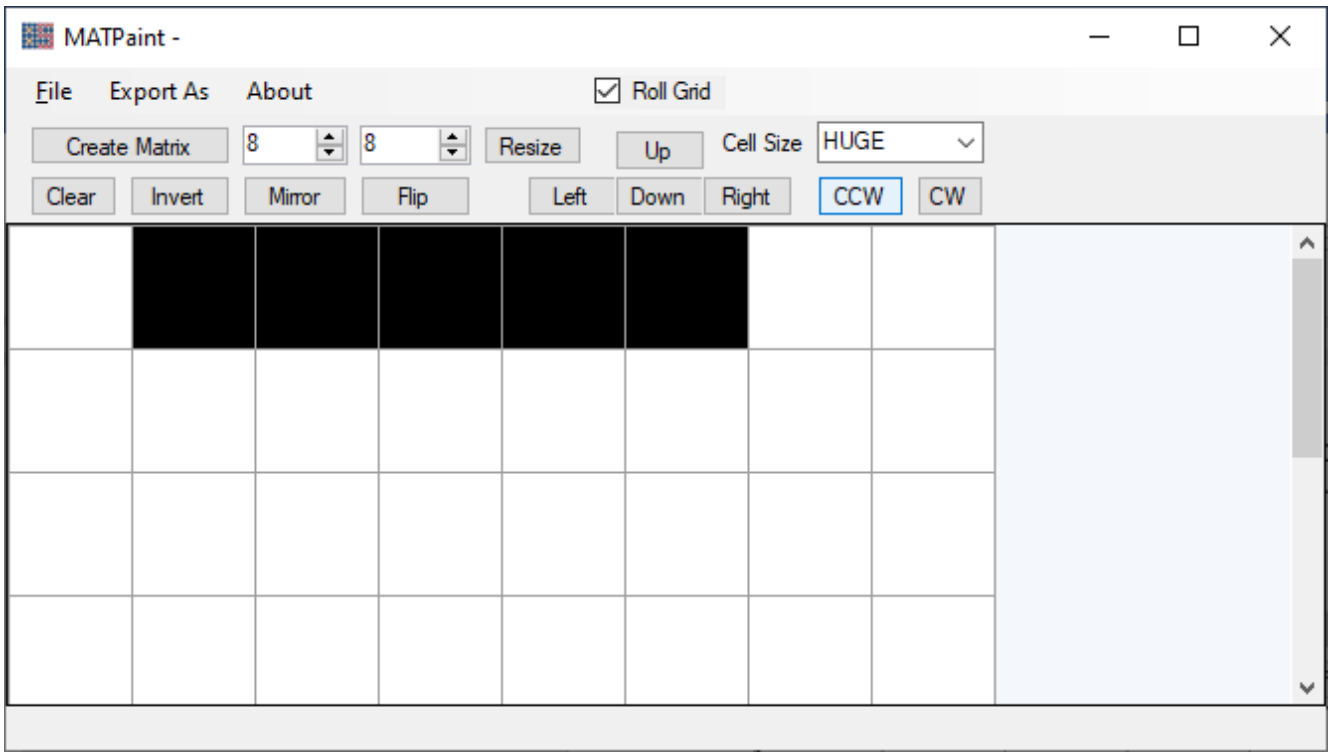

Рисунок 2.24– Поворот навпроти годинникової стрілки

Перші кнопка панелі управління дозволяє виконати наступні дії з файлом: створити, відкрити, переглянути нещодавно відкриті, зберегти, зберегти як, друкувати, попередній перегляд, вихід (рис.2.25).

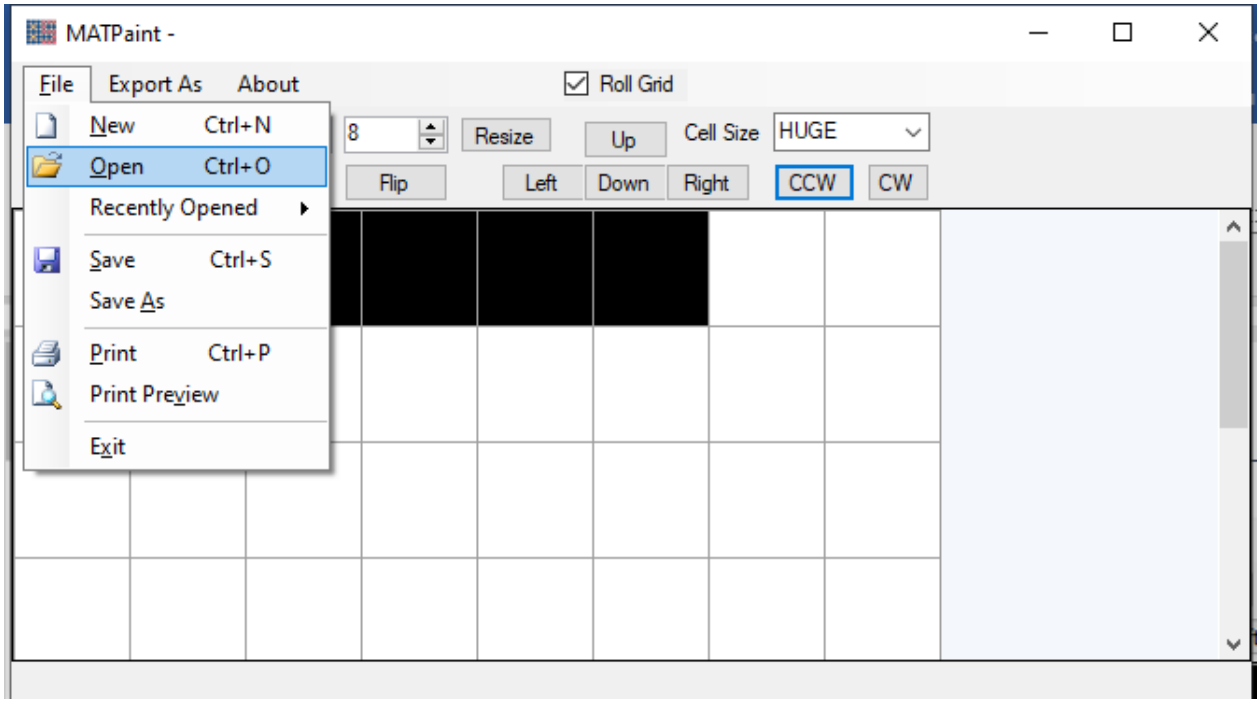

Рисунок 2.25– Робота з файлом

Наступна кнопка панелі управління дозволяє виконати експорт зображення типу .PNG та як бінарний код (рис. 2.26).

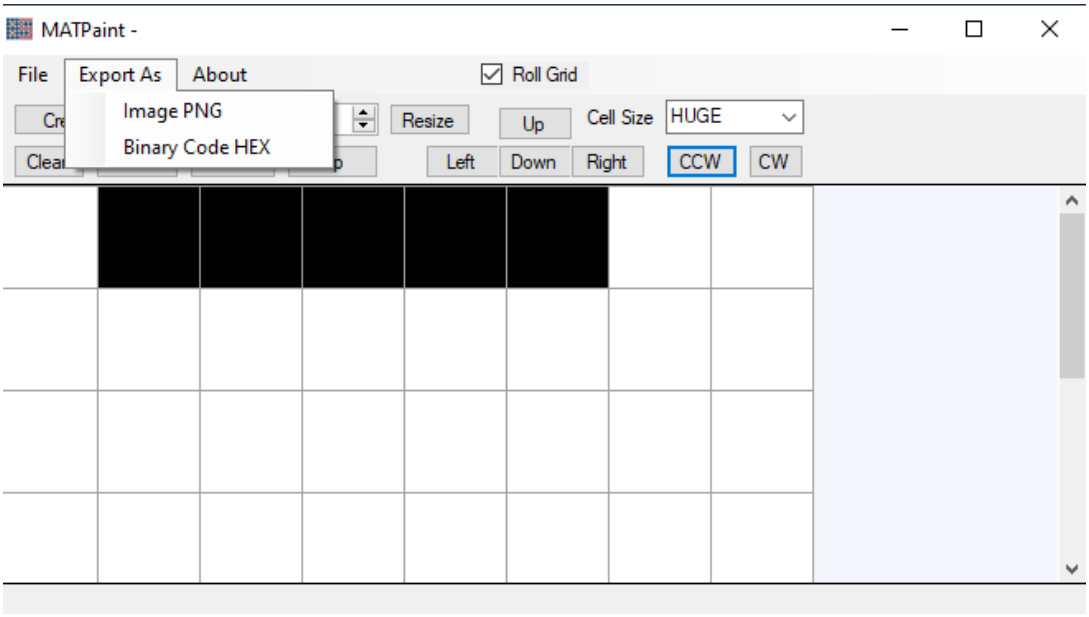

Рисунок 2.26– Експорт файлу

Наступна кнопка панелі управління відкриває вікно з відомостями про програмний засіб (рис. 2.27).

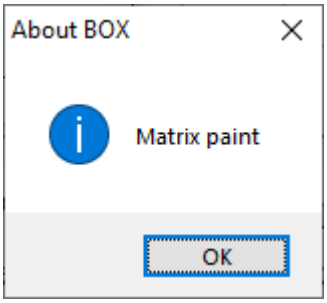

Рисунок 2.27– Про програмний засіб

Після виконання необхідних функцій потрібно коректно завершити роботу програмного засобу, натиснувши на відповідну кнопку панелі управління або на хрестик в правому верхньому куті вікна.

2.7 Тестування роботи програмного модуля

Мета: тестування якості програмного продукту, коректності виконання потрібних функцій програми.

Рівень плану: Тест План (*Test Plan*).

Склад документа: опис обсягу робіт з тестування, опис об'єкта тестування, стратегії, розкладу, критеріїв початку і закінчення тестування.

План проекту: тестування програмного продукту.

Опис тестованого продукту: програмний засіб створення та редагування елементарних растрових зображень. Засіб написаний мовою C# в середовищі Microsoft Visual Studio.

Відповідний стандарт: *IEEE 829-1998 Format*.

Тестові завдання. Контроль якості програмного засобу та виявлення дефектів.

Повинно бути проведено тестування таких частин:

1. Програмний засіб.
2. Створення/редагування зображення.
3. Виконання дій з файлом.
4. Коректність функціонування інтерфейсу.

Функціональне тестування перевіряє, чи реалізовані функціональні вимоги, тобто можливості ПЗ в певних умовах вирішувати завдання, потрібні користувачам. Функціональні вимоги визначають, що саме робить продукт, які завдання вирішує.

Функціональні вимоги включають в себе:

1) функціональна придатність;
2) точність;
3) можливість до взаємодії;
4) відповідність стандартам та правилам;
5) захищеність.

Нефункціональне тестування описує тести, необхідні для визначення характеристик ПЗ, які можуть бути виміряні різними величинами. В цілому, це тестування того, «як» система працює. Далі перелічені основні види нефункціональних тестів:

1) всі види тестування продуктивності:
    1) навантажувальне тестування;
    2) стресове тестування;
    3) тестування стабільності та надійності;
2) об'ємне тестування;
3) інсталяційне тестування;
4) тестування зручності користування;
5) тестування на «відмову» та відновлення;
6) конфігураційне тестування.

Підхід до тестування. Рівень тестування: системне, тестується інтегрована система на її відповідність вимогам з точки зору кінцевого користувача.

Спеціальні засоби тестування: відсутні, тестування буде виконуватися вручну.

Метрики: в рамках даного плану передбачається створити комплект тестів, повний щодо метрики за тестовими випадками (*Test Cases*) (відповідно до міжнародного стандарту *ISO 14598*).

Особливі вимоги до тестування: відсутні, звичайний режим тестування.

Сегмент компонентів: певний сегмент компонентів повинен бути протестований разом.

Обмеження для тестування: проведення тестування без спеціалізованих засобів.

Рекомендована методика тестування: ручна, тому що зменшує матеріальні та програмні витрати на проведення тестування.

Критерії успішності та припинення тестування. Критерії успішності тестування:

1) Розроблені тести виконуються без помилок, система віддажться в експлуатацію;

2) Вдале тестування на виправлену помилку при повторній передачі на тестування.

Критерій виходу дозволяє встановити, який обсяг тестування слід вважати достатнім.

Тестування може бути завершено коли:

1) всі 100% вимог враховані;

2) дефекти встановлені / очікуване число дефектів виявлено;

3) всі дефекти, що відносяться до класу Show Stopper або Blocker, усунуті, ні у одного з критичних дефектів немає статусу «відкритий»;

4) всі дефекти з високим пріоритетом ідентифіковані і виправлені;

5) Defect Rate (швидкість дефектообразования) нижче встановленого допустимого рівня;

6) дуже невелика кількість дефектів середнього рівня критичності «відкриті», їх розбір проведений;

7) число «відкритих» дефектів середнього рівня, які не впливають на користування системою, дуже невелика;

8) всі дефекти з високим рівнем пріоритету закриті і відповідні регресивні сценарії успішно проведені.

Тест кейси зображені на таблицях 2.1, 2.2, 2.3.

Таблиця 2.1 – Тест кейс для створення/відкриття зображення

Дія	**Очікуваний результат**	**Фактичний результат**
Передумова		
Відкрити програмний засіб	Програмний засіб створення зображень відкритий та доступний для використання	Пройдено

Кроки тесту		
Створити новий файл	Успішно створений файл	Пройдено
Відкрити новий файл	Успішно відкритий файл	Пройдено
Зберегти файл	Успішно збережений файл	Пройдено
Друкувати файл	Успішно надрукований файл	Пройдено
Постумова		
Завершити роботу програмного засобу	Робота програмного засобу завершена	Пройдено

Таблиця 2.2 – Тест кейс для основних функцій модуля

Дія	**Очікуваний результат**	**Фактичний результат**
Передумова		
Відкрити програмний засіб	Програмний засіб створення зображень відкритий та доступний для використання	Пройдено
Кроки тесту		
Зміна розміру зображення	Розмір зображення змінено	Пройдено
Виконання переміщення зображення	Зображення переміщено у потрібний бік	Пройдено
Вибір крупності зображення	Зміна крупності зображення	Пройдено
Поворот, дзеркальне зображення,	Зображення повернуто,	Пройдено

інверсія	відображене дзеркально, інвертоване	
Заповнення комірок	Всі комірки зображення заповнені	Пройдено
Очищення комірок	Всі комірки зображення очищені	Пройдено
Постумова		
Завершити роботу програмного засобу	Робота програмного засобу завершена	Пройдено

Таблиця 2.3 – Тест кейс додаткових функцій користувача

Дія	**Очікуваний результат**	**Фактичний результат**
Передумова		
Відкрити програмний засіб	Програмний засіб створення зображень відкритий та доступний для використання	Пройдено
Кроки тесту		
Попередній перегляд	Зображення доступне для перегляду	Пройдено
Експорт файлу в різних форматах	Зображення доступне для експорту	Пройдено
Постумова		
Завершити роботу програмного	Робота програмного засобу	Пройдено

засобу	завершена	

Висновком даного тестового випадку (табл. 2.1) є те, що програмний засіб виконує коректно роботу з файлом, наприклад, відкриття, створення, збереження, друк.

Висновком даного тестового випадку (табл. 2.2) є те, що усі основні функції програмного засобу виконуються коректно, а саме: поворот, зміна розміру зображення, віддзеркалення, інвертування зображення, вибір крупності комірок, автоматичне заповнення усіх комірок та очищення усіх комірок.

Висновком даного тестового випадку (табл. 2.3) є те, що програмний засіб дозволяє виконувати такі додаткові функції, як: експорт зображення у відповідному форматі та попередній перегляд перед друком зображення.

3 Охорона праці

З розвитком науково-технічного прогресу важливу роль грає можливість безпечного виконання людьми своїх трудових обов'язків. У зв'язку з цим була створена і розвивається наука про охорону праці і життєдіяльності людини.

Безпека життєдіяльності (БЖД) - це комплекс заходів, спрямованих на забезпечення безпеки людини в середовищі проживання, збереження його здоров'я, розробку методів і засобів захисту шляхом зниження впливу шкідливих і небезпечних факторів до допустимих значень, вироблення заходів по обмеженню збитку в ліквідації наслідків надзвичайних ситуацій мирного і воєнного часу.

Охорона здоров'я трудящих, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства. Звертається увага на необхідність широкого застосування прогресивних форм наукової організації праці, зведення до мінімуму ручної, малокваліфікованої праці, створення обстановки, що виключає професійні захворювання і виробничий травматизм.

Даний розділ дипломного проекту присвячений розгляду наступних питань:

1) визначення оптимальних умов праці техніка-програміста;
2) розрахунок освітленості;
3) розрахунок рівня шуму.

3.1 Характеристика умов праці програміста

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової праці. Їх праця стала більш інтенсивним, напруженим, які вимагають значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку.

В даний час комп'ютерна техніка широко застосовується у всіх областях діяльності людини. При роботі з комп'ютером людина піддається дії ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики і ін.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

У процесі роботи з комп'ютером необхідно дотримувати правильний режим праці та відпочинку. В іншому випадку у персоналу наголошуються значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, втому і хворобливі відчуття в очах, в поясниці, в області шиї і руках.

3.2 Вимоги до виробничих приміщень

Забарвлення і коефіцієнти віддзеркалення. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою.

Джерела світла, такі як світильники і вікна, які дають віддзеркалення від поверхні екрану, значно погіршують точність знаків і тягнуть за собою перешкоди фізіологічного характеру, які можуть виразитися в значній напрузі, особливо при тривалій роботі. Віддзеркалення, включаючи віддзеркалення від вторинних джерел світла, повинне бути зведено до мінімуму. Для захисту від надмірної яскравості вікон можуть бути застосовані штори і екрани.

Освітлення. Правильно спроектоване і виконане виробниче освітлення покращує умови зорової роботи, знижує стомлюваність, сприяє підвищенню продуктивності праці, благотворно впливає на виробниче середовище, надаючи

позитивну психологічну дію на працюючого, підвищує безпеку праці і знижує травматизм.

Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань, тому такий важливий правильний розрахунок освітленості.

Існує три види освітлення - природне, штучне і поєднане (природне і штучне разом).

Природне освітлення - освітлення приміщень денним світлом, що потрапляє через світлові прорізи в зовнішніх огороджуючих конструкціях приміщення. Природне освітлення характеризується тим, що змінюється в широких межах залежно від часу дня, пори року, характеру області і ряду інших чинників.

Штучне освітлення застосовується при роботі в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день). Освітлення, при якому недостатнє за нормами природне освітлення доповнюється штучним, називається змішаним освітленням.

Згідно СНіП II-4-79 в приміщень обчислювальних центрів необхідно застосувати систему комбінованого освітлення.

При виконанні робіт категорії високої зорової точності (найменший розмір об'єкту розрізнення 0,3 ... 0,5 мм) величина коефіцієнта природного освітлення (КЕО) повинна бути не нижче 1,5%, а при зоровій роботі середньої точності (найменший розмір об'єкту розрізнення 0,5 ... 1,0 мм) КЕО повинен бути не нижче 1,0%. В якості джерел штучного освітлення звичайно використовуються люмінесцентні лампи типа ЛБ, або ДРЛ, які попарно об'єднуються в світильники, які повинні розташовуватися рівномірно над робочими поверхнями.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, наступні: при виконанні зорових робіт високої точності загальна освітленість повинна

складати 300лк, а комбінована - 750лк; аналогічні вимоги при виконанні робіт середньої точності - 200 і 300лк відповідно.

Параметри мікроклімату. Параметри мікроклімату можуть мінятися в широких межах, у той час як необхідною умовою життєдіяльності людини є підтримка постійності температури тіла завдяки терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище. Принцип нормування мікроклімату - створення оптимальних умов для теплообміну тіла людини з навколишнім середовищем.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури і зниження відносної вологості в приміщенні. У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (табл. 3.1).

Об'єм приміщень, в яких розміщені працівники обчислювальних центрів, не повинен бути меншим 19,5 м ^{3} / людини з урахуванням максимального числа одночасно працюючих в зміну. Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери, приведені в табл. 3.2.

Таблиця 3.1 - Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	**Параметр мікроклімату**	**Величина**
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22 ... 24 ° С 40 ... 60% до 0,1 м / с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23 ... 25 ° С 40 ... 60% 0,1 ... 0,2 м / с

Таблиця 3.2 - Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери

Характеристика приміщення	**Об'ємна витрата подається в приміщення свіжого повітря, м ^{3} / на одну людину в годину**
Об'єм до 20м^3 на особу 20 ... 40м ^{3} на особу Більш 40м ^{3} на особу	Не менше 30 Не менше 20 Природна вентиляція

Для забезпечення комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року і доби, чергування праці і відпочинку), так і технічні засоби (вентиляція, кондиціювання повітря, опалювальна система).

Шум і вібрація. Шум погіршує умови праці надаючи шкідливу дію на організм людини. Працюючі в умовах тривалої шумової дії випробовують дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену

стомлюваність, зниження апетиту, біль у вухах і т.д. Такі порушення в роботі ряду органів і систем організму людини можуть викликати негативні зміни в емоційному стані людини аж до стресових. Тривала дія інтенсивного шуму [вище 80 дБ (А)] на слух людини приводить до його часткової або повної втрати.

У табл. 3.3 вказані граничні рівні звуку залежно від категорії тяжкості і напруженості праці, що є безпечними відносно збереження здоров'я і працездатності.

Таблиця 3.3 - Граничні рівні звуку, дБ, на робочих місцях.

Категорія напруженості праці	**Категорія важкості праці**			
	I. Легка	**II. Середня**	**III. Важка**	**IV. Дуже важка**
I. Мало напружений	80	80	75	75
II. Помірно напружений	70	70	65	65
III. Напружений	60	60	-	-
IV. Дуже напружений	50	50	-	-

Ергономічні вимоги д робочого місця. Проектування робочих місць, забезпечених відеотерміналами, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки.

Робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи.

Ергономічними аспектами проектування відеотермінальних робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до розташування документів на робочому місці (наявність і розміри підставки для

документів, можливість різного розміщення документів, відстань від очей користувача до екрану, документа, клавіатури і т.д.), характеристики робочого крісла, вимоги до поверхні робочого столу, регульованість елементів робочого місця.

Головними елементами робочого місця програміста є стіл і крісло. Основним робочим положенням є положення сидячи.

Оптимальне розміщення предметів праці і документації в зонах досяжності:

1) дисплей розміщається в зоні а (у центрі);
2) системний блок розміщається в передбаченій ніші столу;
3) клавіатура – у зоні г/д;
4) «миша» – в зоні в справа;
5) сканер в зоні а/б (зліва);
6) принтер знаходиться в зоні а (праворуч);
7) документація: необхідна при роботі – в зоні легкої досяжності долоні – в, а у висувних ящиках столу – література, невикористовувана постійно.

Для комфортної роботи стіл повинен задовольняти наступним умовам:

1) висота столу повинна бути вибрана з урахуванням можливості сидіти вільно, в зручній позі, при необхідності спираючись на підлокітники;

2) нижня частина столу повинна бути сконструйована так, щоб програміст міг зручно сидіти, не був змушений підбирати ноги;

3) поверхня столу повинна мати властивості, що виключають появу відблисків у поле зору програміста;

4) конструкція столу повинна передбачати наявність висувних ящиків (не менше 3 для зберігання документації, лістингів, канцелярських приналежностей);

5) висота робочої поверхні рекомендується в межах 680-760мм. Висота поверхні, на яку встановлюється клавіатура, повинна бути близько 650мм.

Велике значення надається характеристикам робочого крісла. Так, рекомендована висота сидіння над рівнем підлоги перебуває в межах 420-550мм. Поверхня сидіння м'яка, передній край закруглений, а кут нахилу спинки - регульований.

Велике значення також надається правильній робочій позі користувача. При незручній робочій позі можуть з'явитися болі в м'язах, суглобах і сухожиллях. Вимоги до робочої пози користувача відеотермінала наступні:

1) голова не повинна бути нахилена більш ніж на 20^0;

2) плечі повинні бути розслаблені;

3) лікті - під кутом 80 ° ... 100^0;

4) передпліччя і кисті рук - в горизонтальному положенні.

Причина неправильної пози користувачів обумовлена наступними чинниками: немає хорошої підставки для документів, клавіатура знаходиться дуже високо, а документи - низько, нікуди покласти руки і кисті, недостатній простір для ніг.

3.3 Розрахунок освітленості і рівня шуму

Розрахунок освітленості робочого місця зводиться до вибору системи освітлення, визначенню необхідного числа світильників, їхнього типу і розміщення. Виходячи з цього, розрахуємо параметри штучного освітлення.

Зазвичай штучне освітлення виконується за допомогою електричних джерел світла двох видів: ламп накалювання і люмінесцентних ламп. Використовуватимемо люмінесцентні лампи, які порівняно з лампами розжарювання мають ряд істотних переваг:

1) за спектральним складом світла вони близькі до денного, природного світла;

2) володіють більш високим ККД (у 1,5-2 рази вище, ніж ККД ламп розжарювання);

3) мають підвищену світловіддачею (в 3-4 рази вище, ніж у ламп розжарювання);

4) більш тривалий термін служби.

Розрахунок освітлення проводиться для кімнати площею 15м 2, ширина якої 5м, висота - 3 м. Для визначення кількості світильників визначється світловий потік, де

F - розраховується світловий потік, Лм;

E - нормована мінімальна освітленість, Лк (визначається за таблицею). Роботу програміста, відповідно до цієї таблиці, можна віднести до розряду точних робіт, отже, мінімальна освітленість E = 300лк;

S - площа освітлюваного приміщення (у нашому випадку S = 15м 2);

Z - відношення середньої освітленості до мінімальної (звичайно приймається рівним 1,1 ... 1,2, нехай Z = 1,1);

K - коефіцієнт запасу, враховує зменшення світлового потоку лампи в результаті забруднення світильників у процесі експлуатації (його значення залежить від типу приміщення й характеру проведених у ньому робіт і в нашому випадку K = 1,5);

n - коефіцієнт використання, (виражається відношенням світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в частках одиниці; залежить від характеристик світильника, розмірів приміщення, фарбування стін і стелі, які характеризуються коефіцієнтами відображення від стін (P C) і стелі (P П)), значення коефіцієнтів P C і P П були зазначені вище: P C = 40%, P П = 60%. Значення n визначимо по таблиці коефіцієнтів використання різних світильників. Для цього обчислимо індекс приміщення:

S - площа приміщення, S = 15 м 2;

h - розрахункова висота підвісу, h = 2.92 м;

A - ширина приміщення, A = 3 м;

B - довжина приміщення, B = 5 м.

Знаючи індекс приміщення I, за таблицею 7 [23] знаходимо n = 0,22

Для освітлення вибираємо люмінесцентні лампи типу ЛБ40-1, світловий потік яких F = 4320 Лк.

Розрахуємо необхідну кількість ламп:

N - обумовлений число ламп;

F - світловий потік, F = 33750 Лм;

F л - світловий потік лампи, F л = 4320 Лм.

При виборі освітлювальних приладів використовуємо світильники типу ОД. Кожен світильник комплектується двома лампами.

Розрахунок рівня шуму.

Одним з несприятливих факторів виробничого середовища в ІОЦ є високий рівень шуму, створюваний друкованими пристроями, обладнанням для кондиціонування повітря, вентиляторами систем охолодження в самих ЕОМ.

Для вирішення питань про необхідність і доцільність зниження шуму необхідно знати рівні шуму на робочому місці оператора.

Рівень шуму, що виникає від декількох некогерентних джерел, що працюють одночасно, підраховується на підставі принципу енергетичного підсумовування випромінювань окремих джерел:

L_i - рівень звукового тиску і-го джерела шуму;

n - кількість джерел шуму.

Отримані результати розрахунку порівнюється з допустимим значенням рівня шуму для даного робочого місця. Якщо результати розрахунку вище допустимого значення рівня шуму, то необхідні спеціальні заходи щодо зниження шуму. До них відносяться: облицювання стін і стелі залу звукопоглинальними матеріалами, зниження шуму в джерелі, правильне планування обладнання і раціональна організація робочого місця оператора.

Рівні звукового тиску джерел шуму, що діють на оператора на його робочому місці представлені в табл. 3.4.

Таблиця 3.4 - Рівні звукового тиску різних джерел.

Джерело шуму	**Рівень шуму, дБ**
Жорсткий диск	40

Вентилятор	45
Монітор	17
Клавіатура	10
Принтер	45
Сканер	42

Зазвичай робоче місце оператора оснащене наступним обладнанням: вінчестер в системному блоці, вентилятор (и) систем охолодження ПК, монітор, клавіатура, принтер і сканер.

Підставивши значення рівня звукового тиску для кожного виду обладнання у формулу, отримаємо:

$L_{\Sigma} = 10 \cdot \lg (10^{4} + 10^{4,5} + 10^{1,7} + 10^{1} + 10^{4,5} + 10^{4,2)} = 49,5$ дБ

Отримане значення не перевищує допустимий рівень шуму для робочого місця оператора, рівний 65 дБ (ГОСТ 12.1.003-83). І якщо врахувати, що навряд чи такі периферійні пристрої як сканер і принтер будуть використовуватися одночасно, то ця цифра ще нижчою. Крім того при роботі принтера безпосередню присутність оператора необов'язково, тому що принтер обладнаний механізмом автоподачі аркушів.

3.4 Заходи та засоби протипожежного захисту

Під пожежною безпекою розуміють такий стан промислового або цивільного об’єкта, за якого з регламентованою ймовірністю виключається можливість виникнення і розвитку пожеж та впливу на людей небезпечних чинників пожежі, а також забезпечується захист матеріальних цінностей та довкілля.

Пожежна безпека об’єкта – доволі складне і багатоаспектне завдання, тому для його вирішення потрібно підходити комплексно. Комплекс заходів та засобів щодо пожежної безпеки складається із відповідних систем, зокрема:

1) системи запобігання пожеж, що містить підсистеми запобігання утворенню горючого середовища та виникненню в горючому середовищі джерела запалювання;

2) системи протипожежного захисту, що, своєю чергою, містить такі підсистеми: підсистема обмеження розвитку пожежі; підсистема забезпечення безпечної евакуації людей та майна; підсистема створення умов для успішного гасіння пожежі;

3) системи організаційно-технічних заходів, що передбачає організаційні, технічні, режимні та експлуатаційні заходи.

Організаційні заходи пожежної безпеки передбачають організацію пожежної охорони на об’єкті, проведення навчань з питань пожежної безпеки (інструктажі та пожежно-технічні мінімуми), застосування наочних засобів протипожежної пропаганди та агітації, організацію ДПД та ПТК, проведення перевірок, оглядів стану пожежної безпеки приміщень, будівель, об’єкта загалом та ін.

До технічних заходів належать суворе дотримання правил і норм, визначених чинними нормативними документами при реконструкції приміщень, будівель та об’єктів, технічному переоснащенні виробництва, експлуатації чи можливому переобладнанні електромереж, опалення, вентиляції, освітлення тощо.

Заходи режимного характеру передбачають заборону куріння та застосування відкритого вогню у недозволених місцях, недопущення появи сторонніх осіб у вибухонебезпечних приміщеннях чи об’єктах, регламентацію пожежної безпеки при проведенні вогневих робіт тощо.

Експлуатаційні заходи передбачають своєчасне проведення профілактичних оглядів, випробувань, ремонтів технологічного та допоміжного устаткування, а також інженерного господарства (електромереж, електроустановок, опалення, вентиляції).

Система запобігання пожежі – це комплекс заходів і технічних засобів, які запобігають виникненню пожежі.

Керівники підприємств повинні визначити обов'язки посадових осіб (у тому числі заступників керівника) з забезпечення пожежної безпеки, призначити

відповідальних за пожежну безпеку окремих будівель, споруд, приміщень, дільниць, технологічного та інженерного обладнання, а також за зберігання та експлуатацію технічних засобів протипожежного захисту.

Обов'язки осіб, відповідальних за забезпечення пожежної безпеки, утримання та експлуатації засобів протипожежного захисту слід відобразити у відповідних документах.

Керівник підприємства зобов'язаний вживати (в межах наданих йому повноважень) відповідних заходів реагування на факти порушень чи невиконання іншими працівниками підприємства встановленого протипожежного режиму, вимог правил пожежної безпеки та нормативних актів, що діють у цій сфері.

Керівники підприємств повинні:

1) організувати розроблення комплексних заходів для забезпечення пожежної безпеки, впроваджувати на підприємстві досягнення науки і техніки, позитивний досвід;

2) відповідно до нормативних актів з пожежної безпеки розробляти і затверджувати положення, інструкції та інші нормативні акти, що діють у межах підприємства, здійснювати постійний контроль за їх додержанням;

3) забезпечувати додержання протипожежних вимог стандартів, норм, правил, а також виконання вимог приписів і постанов органів держпожнагляду;

4) організовувати навчання працівників правилам пожежної безпеки та пропаганду заходів для їх забезпечення;

5) в разі відсутності в нормативних актах вимог, потрібних для гарантування пожежної безпеки - вживати відповідних заходів, узгоджуючи їх з органами держпожнагляду;

6) тримати у справному стані засоби протипожежного захисту і зв'язку, пожежну техніку, обладнання та інвентар, не допускати їх використання не за призначенням;

7) створювати в разі потреби відповідно до встановленого порядку підрозділи пожежної охорони та потрібну для їх функціонування матеріально-технічну базу;

8) подавати на вимогу Державної пожежної охорони відомості та документи про стан пожежної безпеки підприємства (об'єкта) і продукції, яку підприємство виробляє;

9) вживати заходів з впровадження автоматичних засобів виявлення і гасіння пожеж та використання з цією метою виробничої автоматики;

10) своєчасно інформувати пожежну охорону про несправність пожежної техніки, систем протипожежного захисту, водопостачання, а також завчасно інформувати про закриття доріг і проїздів на своїй території;

11) проводити службове розслідування випадків пожеж.

До первинних засобів пожежогасіння належать:

1) вогнегасники;
2) пожежні крани-комплекти, ручні насоси
3) лопати, ломи, сокири, гаки, пили, багри;
4) ящики з піском, бочки з водою;
5) азбестові полотнища, повстяні мати та ін.

Первинні засоби пожежогасіння розміщують на пожежних щитах, які встановлюють на території об'єкта з розрахунку один щит на 5000 м^2. Вони мають бути пофарбовані у червоний колір, а пожежний інструмент у чорний.

Серед первинних засобів пожежогасіння найважливішу роль відіграють вогнегасники різних типів: водяні, водо-пінні, порошкові, вуглекислотні, газові.

Залежно від способу транспортування вони бувають: переносні (до 20 кг) та пересувні (до 450 кг).

Залежно від об'єму вогнегасники бувають малолітражні (до 5л), ручні (до 10 л), пересувні (понад 10л).

Вогнегасники маркують буквами, що означає їх вид та цифрами, що визначають їх об'єм.

Найбільш перспективними є порошкові вогнегасники, які застосовують для гасіння лужних металів, ЛЗР і ТР, електрообладнання, що горить під напругою до 1000В, твердих та газоподібних речовин.

Найбільш розповсюдженими є:

1) ОП-1, ОП-2, ОП-9, ОП-10 — переносні;

2) ОПА-50, ОПА-100 — пересувні.

Вони відрізняються між собою лише складом порошку та пристроєм для його подачі.

Вуглекислотні вогнегасники застосовуються для гасіння загорянь на машинах, автомобілях і для невеликих об'ємів нафтопродуктів, а також електроустановок під напругою до 1000В.

У корпусі вогнегасника міститься вуглекислий газ у рідкому стані під високим тиском 6мПа (ручні) і 15 мПА (переносні). У горловині балону змонтований спеціальний пусковий пристрій із сифонною трубкою, який приводиться у дію за допомогою вентильного або пістолетного пристрою. Виходячи з балону назовні, зріджений двооксид вуглецю перетворюється на снігоподібну масу за температури - 80°С.

Вибір типу вогнегасника визначається розмірами загоряння і можливих осередків пожеж.

Утворенню горючого середовища запобігають застосуванням герметичного виробничого устаткування, максимально можливою заміною в технологічних процесах горючих речовин та матеріалів негорючими, обмеженням кількості пожежо-, вибухонебезпечних речовин та матеріалів під час використання та зберігання, а також правильним їх розміщенням, ізоляцією горючого та вибухонебезпечного середовища, організацією контролю за складом повітря в приміщенні та контролю за станом середовища в апаратах, застосуванням робочої та аварійної вентиляції, відведенням горючого середовища в спеціальні пристрої та безпечні місця, застосуванням в установках з горючими речовинами пристроїв від пошкодження та аварій, використанням інгібувальних (хімічно активні компоненти, що сприяють припиненню пожеж) та флегматизаційних (інертні компоненти, що роблять середовище негорючим) речовин.

Виникненню в горючому середовищі джерела запалювання запобігають використанням устаткування та пристроїв, при роботі яких не виникає джерел запалювання, використанням електроустаткування, що відповідає за

досягнення класу пожежо- та вибухонебезпеки приміщеннями та зонами груп і категорій вибухонебезпечної суміші, виконанням вимог щодо сумісного зберігання речовин та матеріалів, використанням устаткування, що задовольняє вимоги електростатичної іскробезпеки, улаштуванням блискавкозахисту, організацією автоматичного контролю параметрів, що визначають джерела запалювання, використанням швидкодіючих засобів захисного вимкнення, заземленням устаткування, видовжених металоконструкцій, використанням при роботі з ЛЗР інструментів, що не допускають іскроутворення, ліквідацією умов для самоспалахування речовин і матеріалів, усуненням контакту з повітрям пірофорних речовин, підтриманням температури нагрівання поверхні устаткування, пристроїв, речовин та матеріалів, які можуть контактувати з горючим середовищем нижче гранично допустимої (80 %) температури займання.

У даному розділі дипломного проекту були викладені вимоги до робочого місця техніка-програміста. Створені умови повинні забезпечувати комфортну роботу. На підставі вивченої літератури з даної проблеми, були зазначені оптимальні розміри робочого столу і крісла, робочої поверхні, а також проведено вибір системи і розрахунок оптимального освітлення виробничого приміщення, а також розрахунок рівня шуму на робочому місці. Дотримання умов, що визначають оптимальну організацію робочого місця інженера - програміста, дозволить зберегти гарну працездатність протягом усього робочого дня, підвищить як в кількісному, так і в якісному відносинах продуктивність праці програміста, що в свою чергу сприятиме якнайшвидшій розробці і налагодженню програмного продукту.

Висновки

Метою даного дипломного проекту була розробка програми «Програмний модуль для роботи з зображенням», яка дозволяє користувачеві створювати, переглядати і редагувати елементарні растрові графічні зображення.

В дипломному проекту проведено аналіз пропонованих графічних редакторів, а також розглянуті теоретичні матеріали з тематики «Графічні редактори», розглянуті їхні особливості.

У першому розділі проекту були розглянуті основні поняття комп'ютерної графіки, а також їх види, поставлено завдання для реалізації програмного продукту, обрано засоби для створення програми

У другому розділі роботи представлений конкретний поетапний алгоритм розробки програмного продукту, способи реалізації основних функцій, детальна інструкція користувача програмного продукту, тестування програми.

У розділі «Охорона праці» описуються загальні вимоги до робочого місця програміста. У вимогах до виробничих приміщень були зазначені оптимальні розміри робочого столу і крісла, робочої поверхні, а також проведено вибір системи і розрахунок оптимального освітлення виробничого приміщення, а також розрахунок рівня шуму на робочому місці.

В результаті виконання дипломного проекту було розроблено програмний засіб створення та редагування зображень, який виконує основні функції, відповідно до вимог, а саме: створення, трансформація, інверсія, зміна розміру, зміна масштабу зображення, а також реалізовано функції виконання збереження, друку, експорту зображення.

Доволі часто графічні редактори використовуються в побуті. І володіння ними потрібно для того, щоб обробляти як свої особисті фотографії, так само і різні друковані роботи, дипломи та інші роботи. Дизайнерські програми відкривають нові можливості, перспективи, розвиваються різнобічне мислення, що доводитьх актуальність сьогодні.

Перелік використаних джерел

1. Аббасов, И.Б. Двухмерное и трехмерное моделирование в *3ds MAX* / И.Б. Аббасов. - М.: ДМК, 2012. - 176 с.

2. Ганеев, Р.М. 3D-моделирование персонажей в *Maya*: Учебное пособие для вузов / Р.М. Ганеев. - М.: ГЛТ, 2012. - 284 с.

3. Климачева, Т.Н. *AutoCAD.* Техническое черчение и 3D-моделирование. / Т.Н. Климачева. - СПб.: BHV, 2008. - 912 с.

4. Пекарев, Л. Архитектурное моделирование в *3ds Max* / Л. Пекарев. - СПб.: BHV, 2007. - 256 с.

5. Петелин, А.Ю. 3D-моделирование в *Google Sketch Up* - от простого к сложному. Самоучитель / А.Ю. Петелин. - М.: ДМК Пресс, 2012. - 344 с.

6. Погорелов, В. *AutoCAD* 2009: 3D-моделирование / В. Погорелов. - СПб.: BHV, 2009. - 400 с.

7. Полещук, Н.Н. *AutoCAD* 2007: 2D/3D-моделирование. / Н.Н. Полещук. - М.: Русская редакция, 2007. - 416 с.

8. Тозик, В.Т. 3ds Max Трехмерное моделирование и анимация на примерах / В.Т. Тозик. - СПб.: *BHV,* 2008. - 880 с.

9. Швембергер, С.И. *3ds Max*. Художественное моделирование и специальные эффекты / С.И. Швембергер. - СПб.: *BHV,* 2006. - 320 с.

10. Еріх Гамма, Річард Хелм, Джон Вліссідес, Ральф Джонсон. Паттерны проектирования. – Питер, 2001, – 395 с.

11. А. Пол. Объектно-ориентированое программирование в действии.– СПБ.: Питер, 1997, 447 с.

12. Дж. Ликнесс Приложения для *Windows 8* на *C#* и *XAML*. – Питер, 2013, – 368 с.

13. Святослав Куликов – Тестирование программного обеспечения Базовый курс (2-е издание), 2018.

14. Сью Блэкман, *Beginning 3D Game Development with Unity/* Сью Блэкман; *Apress*, 2011, 992 с.

15. Фохт, Д. Проектирование и программная реализация систем на персональных ЭВМ/ Д. Фохт СПб.: БХВ - Петербург, 2005. - 96 с.

16. Крейтон, *P.X. Unity Game Development Essentials / P.X.* Крейтон *Packt Publishing*, 2010, 83 с.

17. Мартынов, Н.Н. *C#* для начинающих / Н.Н. Мартынов. - Москва; Кудиц-пресс, 2007. - 272 с.

18. Павловская, Т.А. *C#.* Программирование на языке высокого уровня / Т.А. Павловская - Спб; ПИТЕР, 2009 432 с.

19. Фленов, М. Библия *C#*/Фленов М. СПб.: БХВ - Петербург, 2011. -560с.

20. Фленов, М. *C#.* Секреты программирования / М. Фленов - СПб, ПИТЕР, 2006. - 457с

21. Дубовці В.А. Безпека життєдіяльності. / Учеб. посібник для дипломників. - К.: вид. Кірпи, 1992.

22. Мотузко Ф.Я. Охорона праці. - М.: Вища школа, 1989. - 336с.

23. Безпека життєдіяльності. / Под ред. Н.А. Бєлова - М.: Знання, 2000 - 364с.

24. Самгін Е.Б. Освітлення робочих місць. - М.: МІРЕА, 1989. - 186с.

25. Довідкова книга для проектування електричного освітлення. / Под ред. Г.Б. Кнорринга. - Л.: Енергія, 1976.

26. Боротьба з шумом на виробництві: Довідник / Є.Я. Юдін, Л.А. Борисов; За заг. ред. Є.Я. Юдіна - М.: Машинобудування, 1985. - 400с., Іл.Зінченко В.П. Основи ергономіки. - М.: МГУ, 1979. - 179с.

27. Методичні вказівки з виконання дипломної роботи (проекту) для студентів спеціальності 5.05010301«Розробка програмного забезпечення».

Додаток А – Код модуля MainWindow

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace MATPaint
{
   public partial class Form1 : Form
   {
      // for Pattern Status SAVE/Open functionality
      // to BE DONE....
      enum FileStatus { New, Open};
      FileStatus currentFile = FileStatus.New;
      bool isChanged = false;
      string matrixFileName = null;

      public Form1()
      {
         InitializeComponent();
      }
      public Form1(string matrixFileNameARG) {
         InitializeComponent();
         matrixFileName = matrixFileNameARG;
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         this.Location = Properties.Settings.Default.appLocation;
         this.WindowState = Properties.Settings.Default.appState;
         this.Size = Properties.Settings.Default.appSize;
         num_Col.Value = Properties.Settings.Default.MatCols;
         num_Row.Value = Properties.Settings.Default.MatRows;
         recent1toolStripMenuItem.Text = Properties.Settings.Default.Recent1;
         recent2toolStripMenuItem.Text = Properties.Settings.Default.Recent2;
```

```
    recent3toolStripMenuItem.Text = Properties.Settings.Default.Recent3;
    bt_RotataClock.Enabled = false;
    bt_RotateAnticlock.Enabled = false;
    if (matrixFileName != null)
    {
        OpenMatrix(matrixFileName);
        currentFile = FileStatus.Open;
        isChanged = false;
    }
    else
    {
        currentFile = FileStatus.New;
        NewMatrix();
    }
    UpdateMatrixCellSize();

    // Tooltip
    System.Windows.Forms.ToolTip ToolTip1 = new System.Windows.Forms.ToolTip();
    ToolTip1.SetToolTip(this.num_Col, "Select the Width");
    ToolTip1.SetToolTip(this.num_Row, "Select the Height");
    ToolTip1.SetToolTip(this.cmbx_Grid, "Select the size of the block");
    ToolTip1.SetToolTip(this.bt_MatrixCreate, "To create and display the matrix of provided parameters");
    ToolTip1.SetToolTip(this.bt_Clear, "Clear the matrix data");
    ToolTip1.SetToolTip(this.bt_Invert, "Invert the matrix");
    ToolTip1.SetToolTip(this.bt_Mirror, "Create the mirror of the matrix");
    ToolTip1.SetToolTip(this.bt_Flip, "Turn over the matrix");
    ToolTip1.SetToolTip(this.bt_RotateAnticlock, "Rotate anticlockwise by 90 deg");
    ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
    ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
    ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
    ToolTip1.SetToolTip(this.bt_RotataClock, "Rotate clockwise by 90 deg");
    ToolTip1.SetToolTip(this.cbx_GridRoll, "Choose whether to rotate/roll the grid");
    ToolTip1.SetToolTip(this.bt_Up, "Move up in the grid");
    ToolTip1.SetToolTip(this.bt_Down, "Move down in the grid");
    ToolTip1.SetToolTip(this.bt_Left, "Move left in the grid");
    ToolTip1.SetToolTip(this.bt_Right, "Move right in the grid");

}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (isChanged == true && currentFile == FileStatus.New)
```

```
            {
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                    SaveMatrix();
                else if (z == DialogResult.Cancel)
                    e.Cancel = true;
            }
            else if (isChanged == true && currentFile == FileStatus.Open)
            {
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                    SaveOpenMatrix();
                else if (z == DialogResult.Cancel)
                    e.Cancel = true;
            }

            Properties.Settings.Default.appLocation = this.Location;
            Properties.Settings.Default.appState = this.WindowState;
            Properties.Settings.Default.appSize = this.Size;
            Properties.Settings.Default.MatCols = Convert.ToInt16(num_Col.Value);
            Properties.Settings.Default.MatRows = Convert.ToInt16(num_Row.Value);
            Properties.Settings.Default.Recent1 = recent1toolStripMenuItem.Text;
            Properties.Settings.Default.Recent2 = recent2toolStripMenuItem.Text;
            Properties.Settings.Default.Recent3 = recent3toolStripMenuItem.Text;
            Properties.Settings.Default.Save();
        }

        private void Form1_Resize(object sender, EventArgs e)
        {
            if (this.WindowState == FormWindowState.Minimized)
            {
                notifyIcon1.Visible = true;
                this.ShowInTaskbar = false;
            }
            else
            {
                notifyIcon1.Visible = false;
                this.ShowInTaskbar = true;
            }
        }
```

```
        private void notifyIcon1_MouseDoubleClick(object sender, MouseEventArgs e)
        {
            this.WindowState = FormWindowState.Normal;
            this.ShowInTaskbar = true;
        }

        private void bt_MatrixCreate_Click(object sender, EventArgs e)
        {
            if (isChanged == true && currentFile == FileStatus.New)
            {
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                {
                    SaveMatrix();
                    NewMatrix();
                }
                else if (z == DialogResult.No)
                    NewMatrix();
            }
            else if (isChanged == true && currentFile == FileStatus.Open)
            {
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                {
                    SaveOpenMatrix();
                    NewMatrix();
                }
                else if (z == DialogResult.No)
                    NewMatrix();
            }
            else if (isChanged == false)
                NewMatrix();
        }

        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (dataGridView1.CurrentCell.Style.BackColor != Color.Black)
                dataGridView1.CurrentCell.Style.BackColor = Color.Black;
            else
```

```
        dataGridView1.CurrentCell.Style.BackColor = Color.White;
    dataGridView1.CurrentCell.Selected = false;
    isChanged = true;
}

private void cmbx_Grid_SelectedIndexChanged(object sender, EventArgs e)
{
    UpdateMatrixCellSize();
}

private void dataGridView1_SizeChanged(object sender, EventArgs e)
{
    UpdateMatrixCellSize();
}
private void UpdateMatrixCellSize()
{
    int RowHeight = 10, ColumnWidth = 10;
    switch (cmbx_Grid.SelectedIndex)
    {
        case 0:
            // Fill Screen
            if (dataGridView1.RowCount > 0)
                RowHeight = dataGridView1.Height / dataGridView1.RowCount;
            if (dataGridView1.ColumnCount > 0)
                ColumnWidth = dataGridView1.Width / dataGridView1.ColumnCount;
            break;
        case 1:
            // Tiny
            RowHeight = 12;
            ColumnWidth = 12;
            break;
        case 2:
            // Medium
            RowHeight = 22;
            ColumnWidth = 22;
            break;
        case 3:
            // Large
            RowHeight = 42;
            ColumnWidth = 42;
            break;
        case 4:
```

```
                // HUGE
                RowHeight = 62;
                ColumnWidth = 62;
                break;
            default:
                break;
        }
        for (int z = 0; z < dataGridView1.RowCount; z++)
            dataGridView1.Rows[z].Height = RowHeight;
        for (int z = 0; z < dataGridView1.ColumnCount; z++)
            dataGridView1.Columns[z].Width = ColumnWidth;
    }
    private void bt_Invert_Click(object sender, EventArgs e)
    {
        for (int z = 0; z < dataGridView1.RowCount; z++)
            for (int y = 0; y < dataGridView1.ColumnCount; y++)
                if (dataGridView1.Rows[z].Cells[y].InheritedStyle.BackColor != Color.Black)
                    dataGridView1.Rows[z].Cells[y].Style.BackColor = Color.Black;
                else
                    dataGridView1.Rows[z].Cells[y].Style.BackColor = Color.White;
        isChanged = true;
    }
    private void bt_Clear_Click(object sender, EventArgs e)
    {
        for (int z = 0; z < dataGridView1.RowCount; z++)
            for (int y = 0; y < dataGridView1.ColumnCount; y++)
                dataGridView1.Rows[z].Cells[y].Style.BackColor = Color.White;
    }
    private void bt_Mirror_Click(object sender, EventArgs e)
    {
        int x = dataGridView1.ColumnCount - 1;
        for (int z = 0; z < dataGridView1.RowCount; z++)
            for (int y = 0; y < dataGridView1.ColumnCount / 2; y++)
            {
                Color temp = dataGridView1.Rows[z].Cells[y].Style.BackColor;
                dataGridView1.Rows[z].Cells[y].Style.BackColor = dataGridView1.Rows[z].Cells[x -
y].Style.BackColor;
                dataGridView1.Rows[z].Cells[x - y].Style.BackColor = temp;
            }
        isChanged = true;
    }
    private void bt_Flip_Click(object sender, EventArgs e)
```

```
        {
            int x = dataGridView1.RowCount - 1;
            for (int z = 0; z < dataGridView1.RowCount / 2; z++)
                for (int y = 0; y < dataGridView1.ColumnCount; y++)
                {
                    Color temp = dataGridView1.Rows[z].Cells[y].Style.BackColor;
                    dataGridView1.Rows[z].Cells[y].Style.BackColor = dataGridView1.Rows[x -
z].Cells[y].Style.BackColor;
                    dataGridView1.Rows[x - z].Cells[y].Style.BackColor = temp;
                }
            isChanged = true;
        }
        private void bt_RotateAnticlock_Click(object sender, EventArgs e)
        {
            int x = dataGridView1.ColumnCount - 1;
            Color[,] GridArray = new Color[dataGridView1.RowCount, dataGridView1.ColumnCount];
            // Saving grid data in a 2D array
            for (int z = 0; z < dataGridView1.RowCount; z++)
                for (int y = 0; y < dataGridView1.ColumnCount; y++)
                    GridArray[z,y] = dataGridView1.Rows[z].Cells[y].Style.BackColor;
            // Rotating anticlockwise by 90 deg
            for (int z = 0; z < dataGridView1.RowCount; z++)
                for (int y = 0; y < dataGridView1.ColumnCount; y++)
                    dataGridView1.Rows[z].Cells[y].Style.BackColor = GridArray[y, x - z];
            isChanged = true;
        }
        private void bt_RotataClock_Click(object sender, EventArgs e)
        {
            int x = dataGridView1.ColumnCount - 1;
            Color[,] GridArray = new Color[dataGridView1.RowCount, dataGridView1.ColumnCount];
            // Saving grid data in a 2D array
            for (int z = 0; z < dataGridView1.RowCount; z++)
                for (int y = 0; y < dataGridView1.ColumnCount; y++)
                    GridArray[z, y] = dataGridView1.Rows[z].Cells[y].Style.BackColor;
            // Rotating clockwise by 90 deg
            for (int z = 0; z < dataGridView1.RowCount; z++)
                for (int y = 0; y < dataGridView1.ColumnCount; y++)
                    dataGridView1.Rows[z].Cells[y].Style.BackColor = GridArray[x - y, z];
            isChanged = true;
        }
        private void newToolStripMenuItem1_Click(object sender, EventArgs e)
        {
```

```
            if (isChanged == true && currentFile == FileStatus.New)
            {
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                {
                    SaveMatrix();
                    NewMatrix();
                }
                else if (z == DialogResult.No)
                    NewMatrix();
            }
            else if (isChanged == true && currentFile == FileStatus.Open)
            {
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                {
                    SaveOpenMatrix();
                    NewMatrix();
                }
                else if (z == DialogResult.No)
                    NewMatrix();
            }
            else if (isChanged == false)
                NewMatrix();
        }
        private void saveToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            if (currentFile == FileStatus.New)
                SaveMatrix();
            else if (currentFile == FileStatus.Open)
                SaveOpenMatrix();
        }
        private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveMatrix();
        }
        private void openToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            if (isChanged == true && currentFile == FileStatus.New)
            {
```

```
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                {
                    SaveMatrix();
                    OpenMatrix();
                }
                else if (z == DialogResult.No)
                    OpenMatrix();
            }
            else if (isChanged == true && currentFile == FileStatus.Open)
            {
                DialogResult z = MessageBox.Show("Do you want to SAVE the current Pattern", "Save Pattern", MessageBoxButtons.YesNoCancel);
                if (z == DialogResult.Yes)
                {
                    SaveOpenMatrix();
                    OpenMatrix();
                }
                else if (z == DialogResult.No)
                    OpenMatrix();
            }
            else
                OpenMatrix();
        }

        private void exitToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void NewMatrix()
        {
            dataGridView1.ColumnCount = Convert.ToInt16(num_Col.Value);
            dataGridView1.RowCount = Convert.ToInt16(num_Row.Value);
            for (int z = 0; z < dataGridView1.RowCount; z++)
                for (int y = 0; y < dataGridView1.ColumnCount; y++)
                    dataGridView1.Rows[z].Cells[y].Style.BackColor = Color.White;

            dataGridView1.CurrentCell.Selected = false;
            dataGridView1.DefaultCellStyle.BackColor = Color.White;
            UpdateMatrixCellSize();
```

```
            if (dataGridView1.RowCount == dataGridView1.ColumnCount)
            {
                bt_RotateAnticlock.Enabled = true;
                bt_RotataClock.Enabled = true;
            }
            UpdateMatrixCellSize();
            currentFile = FileStatus.New;
            isChanged = false;
            this.Text = "MATPaint - ";
        }
        private void SaveMatrix()
        {
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                string matrixCode = "";
                matrixCode = dataGridView1.ColumnCount.ToString() + "," + dataGridView1.RowCount.ToString() +
"\r\n";
                int row = 0, col = 0, page = 0, z = 0;
                for (page = 0; page <= dataGridView1.RowCount / 8; page++)
                {
                    for (col = 0; col < dataGridView1.ColumnCount; col++)
                    {
                        for (row = 0; row < 8; row++)
                        {
                            if (dataGridView1.RowCount > (page * 8) + row)
                                if (dataGridView1.Rows[(page * 8) + row].Cells[col].Style.BackColor == Color.Black)
                                    z = z + Convert.ToInt16(Math.Pow(2.0, row));
                            z += 0;
                        }
                        matrixCode = matrixCode + z.ToString() + ",";
                        z = 0;
                    }
                    matrixCode = matrixCode + "\r\n";
                }
                TextWriter tw = new StreamWriter(saveFileDialog1.FileName);
                tw.Write(matrixCode);
                tw.Close();
                matrixFileName = saveFileDialog1.FileName;
                isChanged = false;
                currentFile = FileStatus.Open;
                this.Text = "MATPaint - " + matrixFileName;
                saveFileDialog1.FileName = "";
```

```
        }
    }
        private void SaveOpenMatrix()
    {
        string matrixCode = "";
        matrixCode = dataGridView1.ColumnCount.ToString() + "," + dataGridView1.RowCount.ToString() +
"\r\n";
        int row = 0, col = 0, page = 0, z = 0;
        for (page = 0; page <= dataGridView1.RowCount / 8; page++)
        {
            for (col = 0; col < dataGridView1.ColumnCount; col++)
            {
                for (row = 0; row < 8; row++)
                {
                    if (dataGridView1.RowCount > (page * 8) + row)
                        if (dataGridView1.Rows[(page * 8) + row].Cells[col].Style.BackColor == Color.Black)
                            z = z + Convert.ToInt16(Math.Pow(2.0, row));
                    z += 0;
                }
                matrixCode = matrixCode + z.ToString() + ",";
                z = 0;
            }
            matrixCode = matrixCode + "\r\n";
        }
        TextWriter tw = new StreamWriter(matrixFileName);
        tw.Write(matrixCode);
        tw.Close();
        isChanged = false;
        currentFile = FileStatus.Open;
        saveFileDialog1.FileName = "";
    }
    private void OpenMatrix(string matrixFileNameToOpen = null)
    {
        bool readFile = false;
        openFileDialog1.FileName = matrixFileNameToOpen;
                if(matrixFileNameToOpen != null)
            readFile = true;
        else if (openFileDialog1.ShowDialog() == DialogResult.OK)
            readFile = true;

        if (readFile == true)
        {
```

```
                try
                {
                    TextReader tr = new StreamReader(openFileDialog1.FileName);
                    string matrixCode = tr.ReadToEnd();
                    tr.Close();
                    string[] cellDataS = matrixCode.Split(new char[] { ',', '\r', '\n' },
StringSplitOptions.RemoveEmptyEntries);
                    int[] cellData = new int[cellDataS.Length];
                    int count = 0;
                    foreach (string s in cellDataS)
                        cellData[count++] = Convert.ToInt16(s);
                    int row = cellData[1], col = cellData[0], page = 0, z = 0;
                    count = 1;
                    dataGridView1.ColumnCount = col;
                    dataGridView1.RowCount = row;
                    bt_Clear_Click(this, EventArgs.Empty);
                    try
                    {
                        for (page = 0; page <= row / 8; page++)
                            for (col = 0; col < dataGridView1.ColumnCount; col++)
                            {
                                z = cellData[++count];
                                for (row = 0; row < 8; row++)
                                    if (dataGridView1.RowCount > (page * 8) + row)
                                        if ((z & Convert.ToInt16(Math.Pow(2.0, row))) > 0)
                                            dataGridView1.Rows[(page * 8) + row].Cells[col].Style.BackColor = Color.Black;
                            }
                    }
                    catch { }
                    dataGridView1.ClearSelection();
                    matrixFileName = openFileDialog1.FileName;
                    isChanged = false;
                    currentFile = FileStatus.Open;
                    num_Col.Value = dataGridView1.ColumnCount;
                    num_Row.Value = dataGridView1.RowCount;

                    if (dataGridView1.RowCount == dataGridView1.ColumnCount)
                    {
                        bt_RotateAnticlock.Enabled = true;
                        bt_RotataClock.Enabled = true;
                    }
                    else
```

```
                {
                    bt_RotateAnticlock.Enabled = false;
                    bt_RotataClock.Enabled = false;
                }
                if (matrixFileName != recent1toolStripMenuItem.Text
                    && matrixFileName != recent2toolStripMenuItem.Text
                    && matrixFileName != recent3toolStripMenuItem.Text)
                {
                    recent3toolStripMenuItem.Text = recent2toolStripMenuItem.Text;
                    recent2toolStripMenuItem.Text = recent1toolStripMenuItem.Text;
                    recent1toolStripMenuItem.Text = matrixFileName;
                }
                this.Text = "MATPaint - " + matrixFileName;
            }
            catch { MessageBox.Show("Cannot Read File", "File Read Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }
    }
    private void bt_ChngMatrixSize_Click(object sender, EventArgs e)
    {
        int tmpCol = Convert.ToInt16(num_Col.Value);
        int tmpRow = Convert.ToInt16(num_Row.Value);
        if ((tmpCol < dataGridView1.ColumnCount) || (tmpRow < dataGridView1.RowCount))
        {
            if (MessageBox.Show("The New Size is Smaller than current Size, some Clipping may occur, \nContinue ?", "Resize Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation) == DialogResult.Yes)
            {
                dataGridView1.ColumnCount = tmpCol;
                dataGridView1.RowCount = tmpRow;
                UpdateMatrixCellSize();
                isChanged = true;
            }
        }
        else
        {
            dataGridView1.ColumnCount = tmpCol;
            dataGridView1.RowCount = tmpRow;
            UpdateMatrixCellSize();
            isChanged = true;
        }
        if (dataGridView1.RowCount == dataGridView1.ColumnCount)
        {
```

```
            bt_RotateAnticlock.Enabled = true;
            bt_RotataClock.Enabled = true;
        }
        else {
            bt_RotateAnticlock.Enabled = false;
            bt_RotataClock.Enabled = false;
        }
    }
    private void bt_Up_Click(object sender, EventArgs e)
    {
        // Storing the data of the critical Row 0 - the first row
        Color[] temp = new Color[dataGridView1.ColumnCount];
        for (int y = 0; y < dataGridView1.ColumnCount; y++)
            temp[y] = dataGridView1.Rows[0].Cells[y].Style.BackColor;

        // Moving the rest of the grid UP
        for (int z = 0; z < dataGridView1.RowCount - 1; z++)
            for (int y = 0; y < dataGridView1.ColumnCount; y++)
                dataGridView1.Rows[z].Cells[y].Style.BackColor = dataGridView1.Rows[z + 1].Cells[y].Style.BackColor;
        // Checking whether to roll or shift the grid UP
        if (cbx_GridRoll.Checked == true)
            for (int y = 0; y < dataGridView1.ColumnCount; y++)
                dataGridView1.Rows[dataGridView1.RowCount - 1].Cells[y].Style.BackColor = temp[y];
        else
            for (int y = 0; y < dataGridView1.ColumnCount; y++)
                dataGridView1.Rows[dataGridView1.RowCount - 1].Cells[y].Style.BackColor = Color.White;
    }
    private void bt_Down_Click(object sender, EventArgs e)
    {
        // Storing the data of the critical Row - the last row
        Color[] temp = new Color[dataGridView1.ColumnCount];
        for (int y = 0; y < dataGridView1.ColumnCount; y++)
            temp[y] = dataGridView1.Rows[dataGridView1.RowCount - 1].Cells[y].Style.BackColor;
        // Moving the rest of the grid DOWN
        for (int z = dataGridView1.RowCount - 1; z > 0; z--)
            for (int y = 0; y < dataGridView1.ColumnCount; y++)
                dataGridView1.Rows[z].Cells[y].Style.BackColor = dataGridView1.Rows[z - 1].Cells[y].Style.BackColor;
        // Checking whether to roll or shift the grid DOWN
        if (cbx_GridRoll.Checked == true)
            for (int y = 0; y < dataGridView1.ColumnCount; y++)
```

```
                dataGridView1.Rows[0].Cells[y].Style.BackColor = temp[y];
            else
                for (int y = 0; y < dataGridView1.ColumnCount; y++)
                    dataGridView1.Rows[0].Cells[y].Style.BackColor = Color.White;
        }
        private void bt_Left_Click(object sender, EventArgs e)
        {
            // Storing the data of the critical Column 0 - the first column
            Color[] temp = new Color[dataGridView1.RowCount];
            for (int x = 0; x < dataGridView1.RowCount; x++)
                temp[x] = dataGridView1.Rows[x].Cells[0].Style.BackColor;
            // Moving the rest of the grid to the LEFT
            for (int y = 0; y < dataGridView1.ColumnCount - 1; y++)
                for (int x = 0; x < dataGridView1.RowCount; x++)
                    dataGridView1.Rows[x].Cells[y].Style.BackColor = dataGridView1.Rows[x].Cells[y + 1].Style.BackColor;
            // Checking whether to roll or shift the grid to the LEFT
            if (cbx_GridRoll.Checked == true)
                for (int x = 0; x < dataGridView1.RowCount; x++)
                    dataGridView1.Rows[x].Cells[dataGridView1.ColumnCount - 1].Style.BackColor = temp[x];
            else
                for (int x = 0; x < dataGridView1.RowCount; x++)
                    dataGridView1.Rows[x].Cells[dataGridView1.ColumnCount - 1].Style.BackColor = Color.White;
        }
        private void bt_Right_Click(object sender, EventArgs e)
        {
            // Storing the data of the critical Column 0 - the last column
            Color[] temp = new Color[dataGridView1.RowCount];
            for (int x = 0; x < dataGridView1.RowCount; x++)
                temp[x] = dataGridView1.Rows[x].Cells[dataGridView1.ColumnCount - 1].Style.BackColor;
            // Moving the rest of the grid to the LEFT
            for (int y = dataGridView1.ColumnCount - 1; y > 0; y--)
                for (int x = 0; x < dataGridView1.RowCount; x++)
                    dataGridView1.Rows[x].Cells[y].Style.BackColor = dataGridView1.Rows[x].Cells[y - 1].Style.BackColor;
            // Checking whether to roll or shift the grid to the RIGHT
            if (cbx_GridRoll.Checked == true)
                for (int x = 0; x < dataGridView1.RowCount; x++)
                    dataGridView1.Rows[x].Cells[0].Style.BackColor = temp[x];
            else
                for (int x = 0; x < dataGridView1.RowCount; x++)
                    dataGridView1.Rows[x].Cells[0].Style.BackColor = Color.White;
```

```
}
private void recent1toolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenMatrix(recent1toolStripMenuItem.Text);
}
private void recent2toolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenMatrix(recent2toolStripMenuItem.Text);
}
private void recent3toolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenMatrix(recent3toolStripMenuItem.Text);
}
private void printPreviewToolStripMenuItem_Click(object sender, EventArgs e)
{
    printPreviewDialog1.Document = printDocument1;
    printPreviewDialog1.ShowDialog();
}
private void printToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (printDialog1.ShowDialog() == DialogResult.OK)
        printDocument1.Print();
}
private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    Bitmap matrixPNG = new Bitmap(dataGridView1.Width, dataGridView1.Height);
    dataGridView1.DrawToBitmap(matrixPNG, dataGridView1.ClientRectangle);
    e.Graphics.DrawImage(matrixPNG, new Point(0,0));
}

private void helpToolStripMenuItem1_Click(object sender, EventArgs e)
{
    MessageBox.Show(" Matrix paint","About BOX",MessageBoxButtons.OK,MessageBoxIcon.Information);
}
private void imagePNGToolStripMenuItem_Click(object sender, EventArgs e)
{
    Bitmap matrixPNG = new Bitmap(dataGridView1.Width, dataGridView1.Height);
    dataGridView1.DrawToBitmap(matrixPNG, dataGridView1.ClientRectangle);

    saveFileDialog1.Filter = "PNG Image File|*.png";
    saveFileDialog1.Title = "Save PNG Screenshot";
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
```

```
        matrixPNG.Save(saveFileDialog1.FileName, System.Drawing.Imaging.ImageFormat.Png);
    saveFileDialog1.Filter = "Matrix Pattern Files|*.maz";
    saveFileDialog1.Title = "Save Pattern File";
    saveFileDialog1.FileName = "";
}
private void binaryCodeHEXToolStripMenuItem_Click(object sender, EventArgs e)
{
    saveFileDialog1.Filter = "Binary Code File |*.hex";
    saveFileDialog1.Title = "Save Pattern Hex Code";
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        using (FileStream hex = new FileStream(saveFileDialog1.FileName, FileMode.Create))
        {
            using (BinaryWriter writer = new BinaryWriter(hex))
            {
                int row = 0, col = 0, page = 0;
                byte z=0;
                for (page = 0; page <= dataGridView1.RowCount / 8; page++)
                    for (col = 0; col < dataGridView1.ColumnCount; col++)
                    {
                        for (row = 0; row < 8; row++)
                        {
                            if (dataGridView1.RowCount > (page * 8) + row)
                                if (dataGridView1.Rows[(page * 8) + row].Cells[col].Style.BackColor == Color.Black)
                                    z = Convert.ToByte(z + Math.Pow(2.0, row));
                            z += 0;
                        }
                        writer.Write(z);
                        z = 0;
                    }
                writer.Close();
            }
        }
    saveFileDialog1.Filter = "Matrix Pattern Files|*.maz";
    saveFileDialog1.Title = "Save Pattern File";
    saveFileDialog1.FileName = "";
}   }}
```