Міністерство освіти і науки україни
Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів оптимізації передачі графічної інформації в системах віддаленого моніторингу та управління
(тема)

Виконав:
Випускник 6 курсу, групи ІПЗздм-19-1
Коломійцев О.С.
(прізвище, ініціали)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Турута О.П.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав.кафедри _____
(підпис)

« 26 » березня 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студент Коломійцев Олег Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів оптимізації передачі графічної інформації в системах віддаленого моніторингу та управління
затверджена наказом університету від 26.03.2021 № 34Стз
2. Термін подання роботи до екзаменаційної комісії 22.05.2021р.
3. Вихідні дані до роботи Мова програмування C++, інтегрована система розробки Microsoft Visual Studio 2019, платформа Windows. Бібліотеки: wxWidgets, Boost, Poco, Google Protocol Buffers. Система автоматизації побудови проектів CMake.
4. Перелік питань, що потрібно опрацювати в роботі Проаналізувати методи оптимізації передачі зображення робочого столу віддаленої робочої станції. Розробити алгоритм знаходження різниці двох зображень. Розробити бібліотеку з високорівневим інтерфейсом для передачі інформації по мережі інтернет. Розробити програмну систему з можливостями: збору графічної інформації віддаленої робочої станції, передачі її по мережі, відображення в додатку для віддаленого перегляду.
5. Перелік графічного матеріалу із зазначенням креслеників, схем, слайдів, ілюстрацій
Слайд-презентація

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Інформаційно тематичний пошук	25.01.21 – 10.02.21	Виконано
2.	Систематизація і аналіз зібраної інформації	11.02.21 – 15.02.21	Виконано
3.	Реалізація алгоритму знаходження різниці двох зображень	16.02.21 – 21.02.21	Виконано
4.	Розробка бібліотеки для передачі інформації по мережі інтернет	22.02.21 – 31.03.21	Виконано
5.	Розробка програмної систему	1.04.21 – 20.04.21	Виконано
6.	Оформлення пояснювальної записки	21.04.21 – 10.05.21	Виконано
7.	Нормоконтроль	16.05.21 – 18.05.21	Виконано
8.	Перевірка на академічний плагіат	19.05.21 – 21.05.21	Виконано
9.	Підготовка доповіді до захисту	22.05.21 – 25.05.21	Виконано

Дата видачі завдання 1 квітня 2021р.

Студент _____
(підпис)

Керівник роботи _____ Турута О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 58 с., 27 рис., 1 табл., 20 джер.

АЛГОРИТМ, БІБЛІОТЕКА, ПРОГРАМНА СИСТЕМ, C++, ПРОТОКОЛ, СЕРІАЛІЗАЦІЯ, КЛІЄНТ-СЕРВЕР.

Об’єктом дослідження є методи та алгоритми оптимізації передачі графічної інформації в мережі інтернет. Мультиплатформенні бібліотеки для розробки графічного інтерфейсу користувача, бібліотеки серіалізації на базі яких можливо побудувати протокол передачі даних, бібліотеки для написання програм, які взаємодіють з іншими програмами за допомогою комп’ютерних мереж.

Метою роботи є дослідження методів оптимізації та розробка алгоритму пошуку участків зображень в яких вони різняться. Відокремлення таких участків та передача комп’ютерною мережею. Розробка високорівневої бібліотеки для передачі інформації в комп’ютерній мережі. Розробка клієнт-серверного додатку для захвату, передачі та відображення графічної інформації на програмному вікні.

Методи розробки базуються на використанні мови програмування C++. Бібліотек wxWidgets, Boost, Poco та Google Protocol Buffers.

ALGORITHM, LIBRARY, SOFTWARE SYSTEM, C++, PROTOCOL, SERIALIZATION, CLIENT-SERVER.

The object of study are methods and algorithms to optimize graphic data transfer through internet. Multi-platform libraries to develop graphical user interface, serialization libraries to develop data transfer protocol, libraries to develop application which are able to communicate with each other through internet.

The aim of the work is to analyze methods and develop algorithm to find parts of the images where they are different. Capture of such pieces of image and transfer them through computer network. Development of client-server application to capture image, transfer and show in graphical user interface on opposite side of such application.

Development methods are based on using C++ programming language. Libraries wxWidgets, Boost, Poco and Google Protocol Buffers.

Я, Коломійцев Олег Сергійович, студент групи ІПЗздм-19-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів оптимізації передачі графічної інформації в системах віддаленого моніторингу та управління», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу Elar KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагуату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

PNG – Portable Network Graphics;

JPEG – Joint Photographic Experts Group;

TIFF – Tagged Image File Format.

ВСТУП

Сьогодення вже важко уявити без комп’ютерів та різного роду комп’ютерних систем. Зараз важко уявити хоча б якусь галузь, яка б їх не використовувала, вони проникають буквально в кожну нішу нашого з вами життя. Починаючи з космічних апаратів, медицини й закінчуючи газонокосарками – всюди комп’ютери відіграють якщо не одну з важливих то саму головну роль.

Зважаючи на всюдипроникність такого роду обладнання його потрібно налаштовувати, обслуговувати й проводити оновлення, як технічної частини так і програмного забезпечення. Саме цим й займаються адміністратори та постачальники послуг, які в своїй практиці проводять цілодобове спостереження не за одним таким пристроєм, а за сотнями, а то й тисячами. Найчастіше ці пристрої не знаходяться в одній кімнаті, місті чи навіть країні, а розосереджені по місцях, які знаходяться на значній відстані один від одного й для того, щоб отримати безпосережній доступ до них треба в кращому випадку подолати значну відстань, що само собою потребуватиме багато часу, а інколи й значних матеріальних витрат. Навіть якщо для когось вищезгадані факти не являються проблемою, тобто він готовий витрачати скільки завгодно часу й грошей на те, щоб просто отримати фізичний доступ до обладнання то отримати одночасний доступ в такому випадку не вийде бо фізично ви будете знаходитись в різних місця, як вихід можна збільшити штат найманих працівників, які будуть за вас виконувати таку роботу. Однак така ідея саме по собі теж являється затратною бо кожному з них потрібно платити гороші. Саме для спрощення роботи в такого роду середовищі й існують системи віддаленого моніторингу та управління.

Системи віддаленого моніторингу та управління – це системи, які створені в поміч системному адміністратору, який в свою чергу у віддаленому режимі здатний проводити спостереження й обслуговування серверів, мобільних пристроїв, настільних комп’ютерів в офісах різного роду компаній. За допомогою спеціально встановленого програмного запезпечення він здатен в режимі реального часу,

віддалено отримувати інформацію про життєздатність системи, встановлене обладнання, виконувати оновлення програмного забезпечення, виконувати обмін файлами між своїм комп'ютером та віддаленим, виконувати віддалене керування та багато іншого.

Завдяки реалізації в таких системах функції передачі через комп'ютерну мережу зображення робочого столу та його відображення у вікні спеціально розробленої програми на стороні адміністратора можна в режимі реального часу спостерігати за процесами, які відбуваються на віддаленій машині, якщо на додаток до цього ще й передавати коди клавіш, які натискаються, положення й команди курсору мишки ми отримаємо повноцінний доступ до віддаленого комп'ютеру. З таким набором функцій ми отримуємо ефект повної присутності наче б то ми дійсно сидимо перед екраном компьютера, хоча насправні він знаходиться десь далеко й фізичного контакту з ним не відбувається.

Як виявляється ефективна передача зображення являється досить не тривіальною задачею. Проблема яка як виявилось впливає на продуктивність передачі зображення й яку ми будемо аналізувати й спробуємо вирішити в цій роботі – це зменшення трафіку при передачі зображення. Так як дані картинки робочого столу є досить великими й якщо передавати їх часто й з великою швидкістю канал зв'язку просто буде вимушений відказати в обслуговуванні й належної якості зображення, в додатку для перегляду, отримано не буде.

Для зменшення даних, які являють собою зображення робочого столу чи частини такого зображення, ми спробуємо дослідити методи та розробити алгоритм пошуку участків двох зображеннь у яких вони різняться. Наш додаток буде використовувати алгоритм для відокремлення таких частин картинки та за допомогою, одночасно розробленої, високорівневої бібліотеки для передачі інформації по комп'ютерній мережі, передавати їх для вставки в потрібне місце у вікні для перегляду.

Таким чином ми розробимо клієнт-серверний додаток для захвату графічної інформації на стороні сервера, передачі та її відображення на стороні клієнта.

1 МЕТОДИ ОПТИМІЗАЦІЇ

Методи оптимізації в даній роботі будуть спрямовані на зменшення об’єму інформації, яка передається по каналу зв’язку. Так як комп’ютерною мережею передається зображення робочого столу віддаленого комп’ютера має сенс кожного разу передавати не все зображення, а лише ті частини зображення, які змінилися в порівнянні з попереднім зображенням отриманим на протилежному кінці розподіленої системи та відображеним в програмі перегляду.

Тобто в першому наближенні потрібно вирішити задачу знаходження частин зображення, які змінилися, ізолювати їх, передати та протилежну сторону по каналу зв’язку й замінити ними відповідні частини зображення у вікні програми перегляду. Для коректної заміни необхідно правильно визначити координати й разом з зображенням передати його координати для правильної вставки.

На розмір мережевих пакетів в нашому випадку також буде впливати формат зображення, яке передається. Зображення, яке ми будемо отримувати представляє собою платформо залежне растрове зображення й може бути конвертоване в низку форматів таких як JPEG, TIFF та PNG. Виходячи з вище сказаного має сенс проводити конвертацію початкового зображення в формат, який займає на жорсткому диску найменше місця, як наслідок з таким зображенням отримаємо найменший розмір мережевого пакету.

То ж далі проведемо аналіз методів знаходження різниці між двома зображеннями й як результат визначимо, який метод найкраще використовувати при реалізації програмної системи. Також порівняємо вище згадані формати зображення для вибору оптимальних з точки зору розміру вихідної картинки та як наслідок навантаження на мережу передачі даних.

1.1 Координатна система вікна

Для початку аналізу методів знаходження різниці двох зображень розглянемо, як влаштована координатна система вікна. Аналіз будемо проводити базуючись на операційній системі Microsoft Windows так як подальшу розробку програмної системи будемо проводити саме на цій платформі.

Координатна система вікна базується на координатній системі встановленого дисплею. Одиницями вимірювання являються пікселі. Будь-яка точка на екрані описується за допомогою пар x та y-координат [1] – [3]. Координата x збільшується зліва на право, y-координата збільшується зверху вниз.

Система та програмні додатки визначають позицію вікна на екрані в екранних координатах. Початок екранних координат розташований в лівому верхньому куті екрану. Позиція точок у вікні визначається в межах клієнтських координат, які починаються в лівому верхньому куті клієнтської області. Всі графічні операції у вікні чи в клієнтській області обмежені верхнім лівим та нижнім правим кутом.

Тепер коли ми розуміємо, як влаштована система координат проведемо мірковування та тему того, як же знайти участки зображення, які різняться й головне, як вставити їх в потрібну позицію після передачі для відтворення актуального зображення.

1.2 Простий метод знаходження різниці двох зображень

Даний метод заключається в послідовному порівнянні всіх пікселів двох зображень з метою знаходження положення в якому значення двох пікселів будуть відрізнятися. Як тільки такий піксель знайдений, в залежності від того в якому напрямку виконувався обхід масиву пікселів зображення, визначається одна з

чотирьох координат. За цими координатами в подальшому буде формуватися прямокутник по периметру якого буде виконуватися вирізання зображення з метою подальшого надсилання по каналу зв'язку та вставки в відповідну позицію результуючого зображення.

Вище згадані координати представляють собою лівий верхній та правий нижній кут прямокутника, якщо провести через ці координати чотири прямі в результаті їх перетину буде сформовано прямокутник, який і вкаже на ту область зображення, яку слід вирізати й вставити в застаріле зображення.

Розглянемо ідеальну ситуацію коли вихідне зображення пусте, а на новому зображенні з'являється фігура показана на рисунку 1.1.

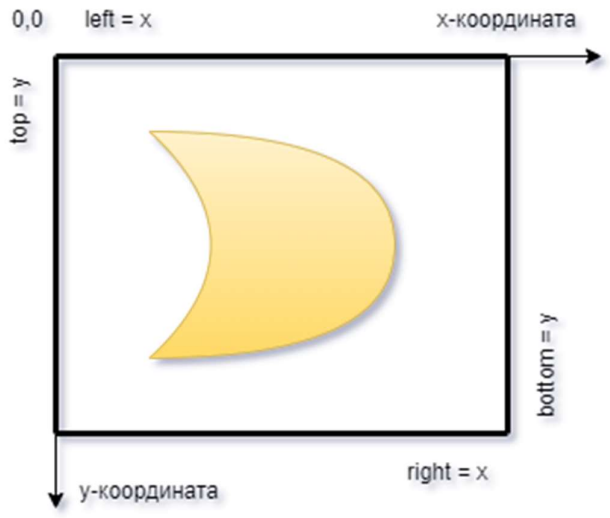

Рисунок 1.1 – Визначення координат результуючого прямокутника

На рисунку 1.1 показано нове зображення, яке поступило на обробку й буде використане для порівняння із попередньою версією. У якості попередньої версії в цьому випадку виступає пусте зображення. В верхньому лівому куті знаходиться початок системи координат. Зліва на право направлена x-координата, зверху вниз направлена y-координата. Координати про які ми згадували раніше також показані на цьому рисунку й називаються вони відповідно для верхнього лівого кута top та left, а для нижнього правого bottom та right. Координата top визначає y-координату

верхнього лівого кута, а left відповідно x-координату цього ж кута. Координата bottom вказує на y-координату нижного правого кута, за x-координату даного кута відповідає right [4] – [5]. Отримувати значення даних координат будемо в результаті обходу й порівняння пікселів прямокутного зображення з чотирьох напрямків (рисунок 1.2).

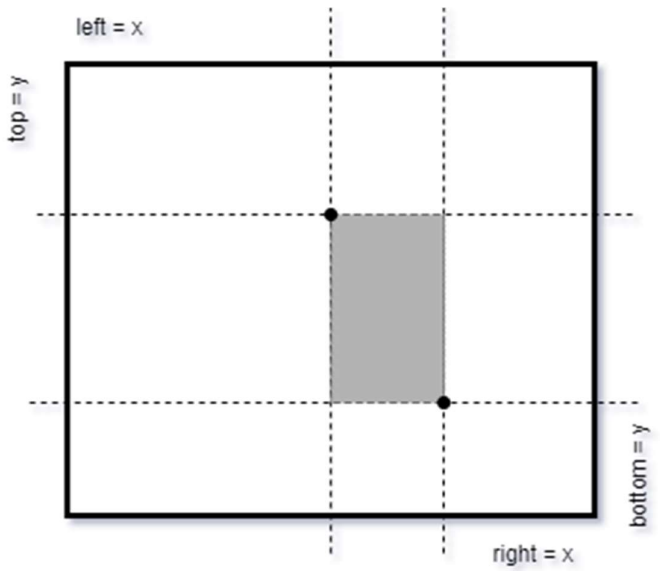

Рисунок 1.2 – Напрямки обходу зображення для визначення координат прямокутника зі змінами

З рисунку 1.2 можна бачити, що якщо через кожну знайдену координату top, left, bottom та right провести пряму лінію перпендикулярно осі відповідної координати в результаті перетинів таких ліній утвориться прямокутник який і буде обмежувати участок зображення зі змінами [6] – [8]. Саме цей прямокутник ми і будемо вирізати з нового зображення й передавати для вставки в результуюче зображення.

Маючи координати лівого верхнього та нижнього правого кутів ми можемо виділити прямокутник для вирізання й з відповідними координатами вставити його в результуюче зображення (рисунок 1.3).

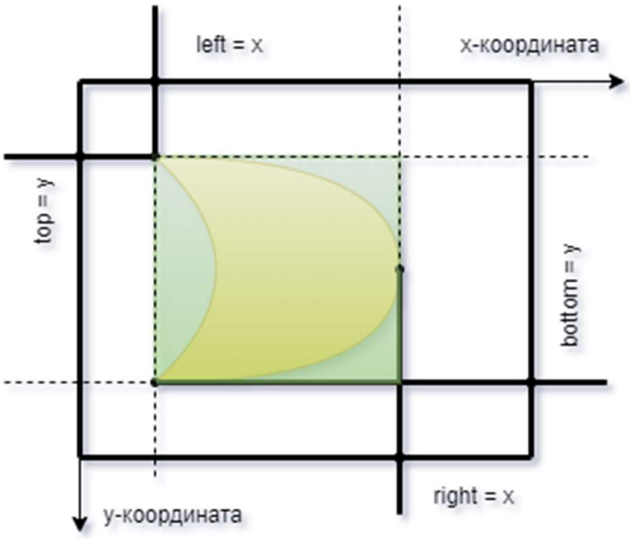

Рисунок 1.3 – Ілюстрація виділення участку зображення з визначеними координатами

На рисунку 1.3 показано, як за допомогою раніше знайдених координат виконано виділення участку зображення, в якому відбулися зміни у порінянні з попедньою версією.

На цьому реалізацію простого методу можна вважати закінченою. Слід також вказати на перевеги та недоліки даного методу.

У порівнянні з ситуацією коли передається все зображення без виділення областей, які змінилися цей алгоритм дає відчутні переваги у зменшенні об’єму інформації, яка передається по каналу зв’язку. Так як він дозволяє працювати в більшості своїх випадків з корисною інформацією.

Однак у даного методу є і свої недоліки до який можна віднести недостатню ефективність при виділенні змінених участків в ситуації коли доприкладу такі зміни відбулися в двох різних кінцях зображення й вони є незначними в плані площі, яку вони покривають (рисунок 1.4).

З рисунку 1.4 видно, що об’єм корисної інформації яка дійсно підлягає передачі, вона представлена кругами в верхньому лівому на нижньому правому кутах, є не великим й значну кількість даних вихідного пакету будуть становити участки зображення в яких змін не відбулося взагалі [8] – [10]. Саме в таких

ситуаціях даний алгоритм веде себе не ефективно. Логічним тут була б поведінка алгоритму, як показано на рисунку 1.5.

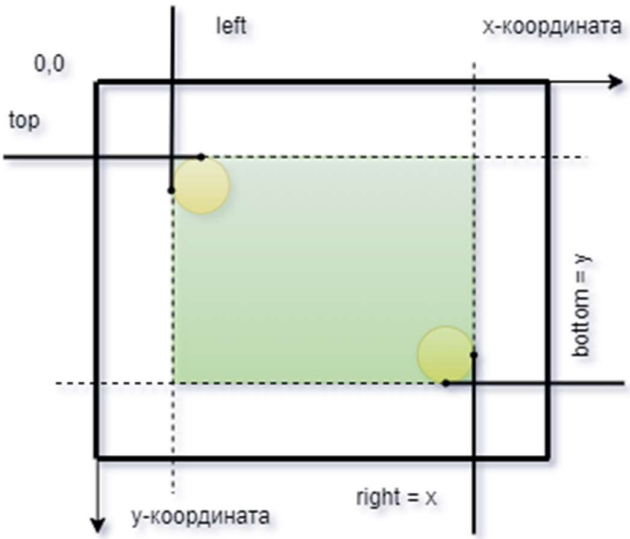

Рисунок 1.4 – Не ефективна локалізація знім

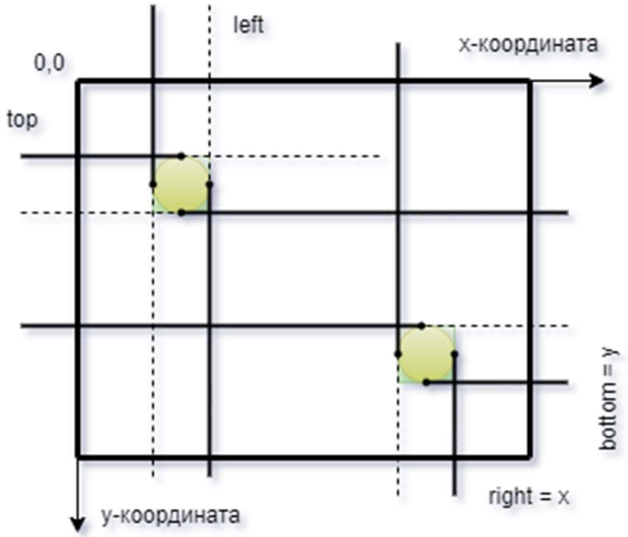

Рисунок 1.5 – Ефективна локалізація змін

В даному випадку (рисунок 1.5) виділення участків, які зазнали змін відбувається максимально ефективно й участки в яких не відбулося зміни зображення не великі. Нажаль такої поведінки не вдасться досягти з цим

алгоритмом, але його можна модифікувати так, щоб корисна інформація виділялася значно ефективніше ніж це відбувається зараз. Як цього досягти ми розглянемо в наступному підрозділі.

1.3 Вдосконалений метод знаходження різниці двох зображень

Для покращення попереднього алгоритму виділення участків зображення зі змінами виконаємо розбиття початкового зображення на зображення меншого розміру (рисунок 1.6).

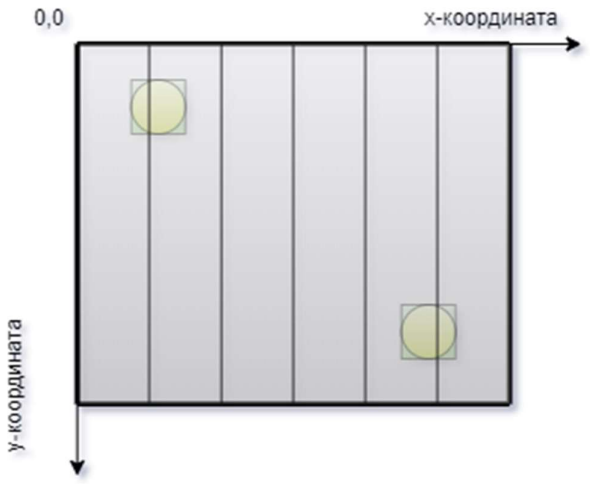

Рисунок 1.6 – Розбиття початкового зображення на зображення меншого розміру

В результаті розбиття показаного на рисунку 1.6 отримаємо так би мовити контейнери менших розмірів й тепер будемо виконувати пошук пікселів, які різняться окремо в кожному контейнері. Сам алгоритм пошуку залишається не змінним. Результати роботи по кожному контейнеру будемо відправляти окремо один від одного. У випадку показаному на рисунку 1.6 замість одного здорового пакету буде відправлено чотири малельних так як обидва участки зі змінами лежать на границі двох контейнерів.

За допомогою використання такого прийому ми зможемо досягти більше ефективної передачі графічної інформації за рахунок того, що пакети якими ми оперуємо не містять зайвої інформації, за розмірами вони є мінімальними, що дозволить передати й обробити їх максимально швидко.

1.4 Формат зображення

Як виявилось формат зображення відіграє досить важливу роль так як різні формати зображення займають різний об’єм пам’яті на жорсткому диску, що в нашому випадку суттєво впливає на розмір даних для передачі по комп’ютерній мережі. Зображення, яке ми можемо отримати використовуючи інтерфейс операційної системи представляє собою платформо залежне растрове зображення й може бути конвертоване в такі формати як JPEG, TIFF та PNG [11].

Для порівняння розміру зображень у вище згаданих форматах проведемо експеримент. Використовуючи бібліотеку wxWidgets розробимо функцію показану на рисунку 1.7, зробимо знімки екрану та виконаємо збереження отриманого зображення на жорсткий диск в різних форматах й порівняємо розмір отриманих файлів.

```
wxImage viewport::make_screenshot_image() {
    wxBitmap screenShot(
        primary_display_.GetWidth(),
        primary_display_.GetHeight(),
        SCREEN_DEPTH);
    wxMemoryDC memDC;
    memDC.SelectObject(screenShot);
    wxScreenDC dcScreen;
    memDC.Blit(
        0,
        0,
        screenShot.GetWidth(),
        screenShot.GetHeight(),
        &dcScreen,
        0,
        0,
        wxRasterOperationMode(wxCOPY),
        true);
    memDC.SelectObject(wxNullBitmap);
    return screenShot.ConvertToImage();
}
```

Рисунок 1.7 – Функція збереження зображення екрану комп’ютера

В результаті використання програмного інтерфейсу представленого на рисунку 1.7 було виконано конвертацію зображення в вибрані формати й збережено на жорсткий диск. Результати дослідження показані в таблиці 1.1.

Таблиця 1.1 – Розмір файлу з зображенням екрану в залежності від формату

Назва формату	Розмір файлу на жорсткому диску, KB
PNG	207
JPEG	157
TIFF	3080

Аналізуючи результати представлені в таблиці 1.1 можна з впевненістю сказати, що зображення у форматі PNG має найменший розмір серед обраних для тесту. Тому конвертацію саме в цей формат рекомендується використовувати при розробці програмної системи.

1.5 Постановка задачі

Опираючитись на проведений теоретичний та практичний аналіз в ході виконання кваліфікаційної роботи планується розробити програмну систему на мові програмування C++ з використанням бібліотек Boost та Poco. Дана система повинна реалізувати клієнт-серверну архітектуру й має виконувати такі задачі як захват, передача та відображення графічної інформації екрану комп’ютера у вікні програмного додатку розробленого з використанням бібліотеки wxWidgets.

Для цілей комунікації розроблених програмних додатків ставиться задача з розробки бібліотеки, яка інкапсулює в собі високорівневі інтерфейси для передачі через комп’ютерну мережу інформації різного типу. Для реалізації функції

передачі графічної інформації планується розробити протокол передачі даних з використання бібліотеки Google Protocol Buffers.

З метою найшвидшої обробки графічної інформації захваченої з екрану комп’ютера ставиться задача з розробки алгоритму пошуку участків двох зображень в яких вони різняться.

2 РЕАЛІЗАЦІЯ АЛГОРИТМІВ ПОШУКУ РІЗНИЦІ ДВОХ ЗОБРАЖЕНЬ

Як було сказано в попередніх розділах релізація алгоритму пошуку різниці двох зображень зводиться до визначення координат прямокутника, який обмежує область зі змінами. Нам достатньо визначити координати верхнього лівого на нижнього правого кутів такого прямокутника. Координати цих кутів будемо позначати як left, top та bottom, right відповідно [12].

Далі будемо розглядати так званий простий та покращений алгоритм пошуку таких координат. Загальний підхід у визначенні вище згаданих координат однаковий для обох алгоритмів однак дещо різнить в тому, які області зображення скануються та в якій послідовності. Тож розглянемо програмну реалізацію цих алгоритмів.

2.1 Простий алгоритм знаходження різниці двох зображень

Ми вже ввели позначення для координат тож розглянемо, як їх розрахувати. Для отримання чотирьох координат нам необхідно виконати чотири ітерації циклів, після кожної з них буде остаточно визначено один з невідомих параметрів.

Ознакою того що відповідна координата знайдена являється той факт, що при скануванні пікселів двох зображень знайдено таку позицію в якій значення пікселів різне, в цей момент ітерація циклу може бути прервана й в залежності від того в якому напрямку було проведено сканування потрібно зберегти значення відповідної координати.

У випадку коли під час сканування зображення на першій ітерації не знайдено пікселів, які б відрізнялися – запуск наступних трьох ітерацій є необов’язковим так як вже на цьому етапі достоменно відомо, що два зображення є абсолютно ідентичними.

Далі розглянемо, як виконується сканування на кожному з чотирьох напрямків й який напрямок відповідає якій координаті.

Першою розглянемо координату top. Дана координата може бути отримана в результаті обходу зображення зверху вниз та зліва на право (див. рис. 2.1). Якщо говорити мовою алгоритмів та циклів то ми отримуємо цикл який виконується в циклі.

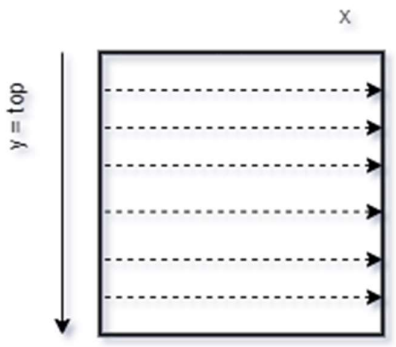

Рисунок 2.1 – Визначення координати top

На рисунку 2.1 горизонтальними лініями показано ітерації внутрішнього циклу, а вертикальна стрілка асоціюється з зовнішнім циклом. Так в момент коли буде знайдено позицію в якій значення пікселів виявиться різним, значення змінної y буде вказувати на значення координати top. Покажимо, як даний цикл реалізується в термінах мови програмування C++ (див. рис. 2.2).

```
wxRect rect;
for (int y = 0; y < current.GetHeight(); ++y) {
    for (int x = 0; x < current.GetWidth(); ++x) {
        if (!are_pixels_equal(current, previous_, x, y)) {
            rect.SetTop(y);
            done = true;
            break;
        }
    }
    if (done) {
        break;
    }
}
```

Рисунок 2.2 – Цикл з визначення координати top

На рисунку 2.2 можна бачити, що зовнішній цикл ітерується вздовж висоти зображення тоді як внутрішній вздовж ширини зображення. В обох випадках ітерації починаються з нуля й проходять відповідно вздовж висоти та ширини зображення. На наступному кроці розглянемо процедуру визначення координати left.

Координату left будемо знаходити обходячи зображення зліва направо та зверху вниз (див. рис. 2.3).

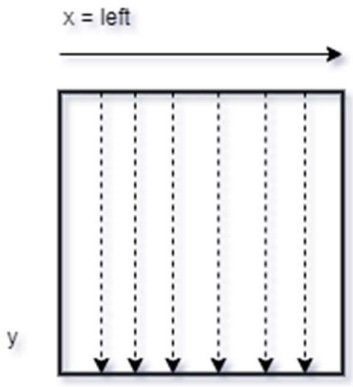

Рисунок 2.3 – Визначення координати left

Вертикальними лініями на рисунку 2.3 показані ітерації внутрішнього циклу, а горизонтальною стрілкою показана ітерація зовнішнього циклу. Так в момент коли буде знайдено позицію в якій значення пікселів різне – значення змінної x буде вказувати на координату left. Реалізація показан на рисунку 2.4.

```
for (int x = 0; x < current.GetWidth(); ++x) {
    for (int y = 0; y < current.GetHeight(); ++y) {
        if (!are_pixels_equal(current, previous_, x, y)) {
            rect.SetLeft(x);
            done = true;
            break;
        }
    }
    if (done) {
        break;
    }
}
```

Рисунок 2.4 – Цикл з визначення координати left

Реалізація циклу показаного на рисунку 2.4 мало чим відрізняється від реалізації циклу для визначення координати top. Головною відмінністю являється те що внутрішній та зовнішній цикли помінялися місцями. Зовнішній цикл тепер ітерує вздов ширини зображення, а внутрішній вздовж висоти зображення. Так само в обох випадках ітерації починаються з нуля.

Коли значення координат top та left розраховано верхній лівий кут прямокутника зі змінами в зображенні остаточно встановлено, лишається визначити координати правого нижнього кута.

Значення координати bottom може бути встановлене в результаті обходу масиву пікселів знизу вверх та справа наліво (див. рис. 2.5).

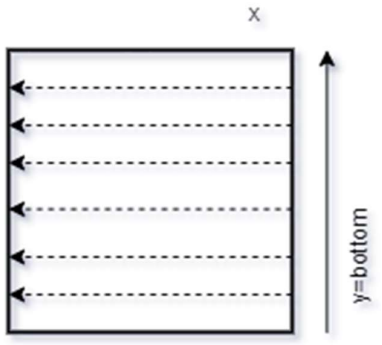

Рисунок 2.5 – Визначення координати bottom

Горизонтальними стрілками на рисунку 2.5 показані ітерації внутрішнього циклу, а вертикальна стрілка вказує на напрям ітерації зовнішнього циклу. В момент коли буде знайдено піксель, який буде різним для двох зображень, значення змінної y буде вказувати на значення параметру bottom. Тут слід зауважити, що початок ітерації для зовнішнього та внутрішнього циклів починається з кінця.

Реалізація на мові програмування C++ показан на рисунку 2.6. Ітерація зовнішнього циклу відбувається вздовж висоти зображення й починається з кінця, ітерація внутрішнього циклу відбувається вздовж ширини зображення й також відбувається в зворотньому напрямку – починається з кінця. Даний цикл чимось

сходий на цикл для визначення координати top з єдиною відмінністю в тому що напрямок ітерацій змінено на протилежний.

```
done = false;
for (int y = current.GetHeight() - 1; y >= 0; --y) {
    for (int x = current.GetWidth() - 1; x >= 0; --x) {
        if (!are_pixels_equal(current, previous_, x, y)) {
            rect.SetBottom(y);
            done = true;
            break;
        }
    }
    if (done) {
        break;
    }
}
```

Рисунок 2.6 – Цикл з визначення координати bottom

І нарешті останній параметр right може бути знайдено порівнюючи значення пікселів двох зображень в таких напрямках, як справа на ліво на знизу вверх (див. рис. 2.7).

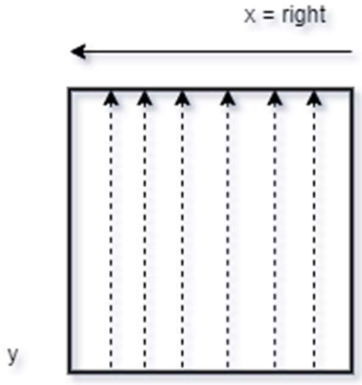

Рисунок 2.7 – Визначення координати right

На рисунку 2.7 вертикальні стрілки вказують напрямок ітерацій внутрішнього циклу, а горизонтальна стрілка вказує напрямок для ітерації зовнішньо циклу. Відповідно значення параметру right стане відомим в той момент коли на черговій ітерації циклу буде знайдено піксель значення якого в ітерованих

масивах буде різним. Реалізацію даного підходу на мові програмування C++ показано на рисунку 2.8.

```
done = false;
for (int x = current.GetWidth() - 1; x >= 0; x--) {
    for (int y = current.GetHeight() - 1; y >= 0; --y) {
        if (!are_pixels_equal(current, previous_, x, y)) {
            rect.SetRight(x);
            done = true;
            break;
        }
    }
    if (done) {
        break;
    }
}
```

Рисунок 2.8 – Цикл з визначення координати right

Як можна бачити з рисунку 2.8 ітерація зовнішнього циклу відбувається вздовж ширини зображення тоді як внутрішній цикл ітерує вздовж висоти зображення. Обидві ітерації відбуваються в зворотньому напрямку. Даний цикл сходий на цикл з визначення координати top з єдиною відмінністю в тому що ітерації виконуються в протилежному напрямку.

На даному етапі координати, як верхнього лівого так і правого нижнього кута зображення зі змінами, нам стають відомі тому можна приступати до вирізання даного прямокутника й відправки (рисунок 2.9) по каналу зв’язку для вставки у відповідну позицію в вікні перегляду на приймаючій стороні.

```
void viewport::screen_data_pack_and_send(const wxImage& img, const wxRect& rect) const {
    viewport::bytes container;
    viewport::save_image_to_container(img, container);
    netlib::viewport::Frame frame;
    frame.set_width(rect.GetWidth());
    frame.set_height(rect.GetHeight());
    frame.set_x(rect.GetX());
    frame.set_y(rect.GetY());
    frame.set_data(&container[0], container.size());
    frame.set_type(netlib::viewport::Frame_Type_Data);
    LOG_INF_FMT("Send image width:%d height:%d", rect.GetWidth(), rect.GetHeight());
    send_handler_(id_, frame);
}
```

Рисунок 2.9 – Процедура упаковки й відправки зображення

Операція вирізання прямокутника заданого знайденими в результаті проведених вище розрахунків координат виконується засобами бібліотеки wxWidgets та класу wxImage додаткових розрахунків та маніпуляцій тут нам робити не потрібно. На рисунку 2.9 в тілі процедуки виконується упаковна зображення в масив байтів, ініціалізація протокольного фрейму та виклик функції для розміщення його в черзі на відправку по каналу зв'язку. Повна реалізація даного алгоритму представлена в додатку В.

2.2 Вдосконалений алгоритм знаходження різниці двох зображень

Даний алгоритму за рахунок розбиття вихідного зображення на частини виконує більш ефективне виділення й локалізацію участків зображення зі змінами, які підлягають перенесенню на результуюче зображення.

В результаті розбиття початкового зображення, як показано на рисунку 2.10, відбувається ітерація не всього зображення відразу, а окремо ітерується кожна частина, яка утворилася після розбиття й поступово аналізується все зображення. Також слід відмітити, що відправка інформації відбувається після закінчення ітерації кожної частини зображення, а не так як у випадку з простим алгоритмом після закінчення аналізу всього зображення. Завдяки цьому оновлення результуючого зображення відбувається значно швидше.

В результаті такого підходу вдається ефективніше виділяти участки для передачі й головне чим цінний цей підхід – це те що сумарний об'єм участків без змін, які так чи інакше потрапляють в корисне навантаження, значно знижується в порівнянні з простим алгоритмов. Таким чином розмір пакетів, які передаються ком'ютерною мережею, зменшується. Водночас з цим збільшується їхня кількість. Однак сумарний об'єм інформації, яка підлягає передачі значно зменшується, що позитивно впливає на продуктивність системи в цілому й оновлення зображення у вікні перегляду відбувається значно швидше.

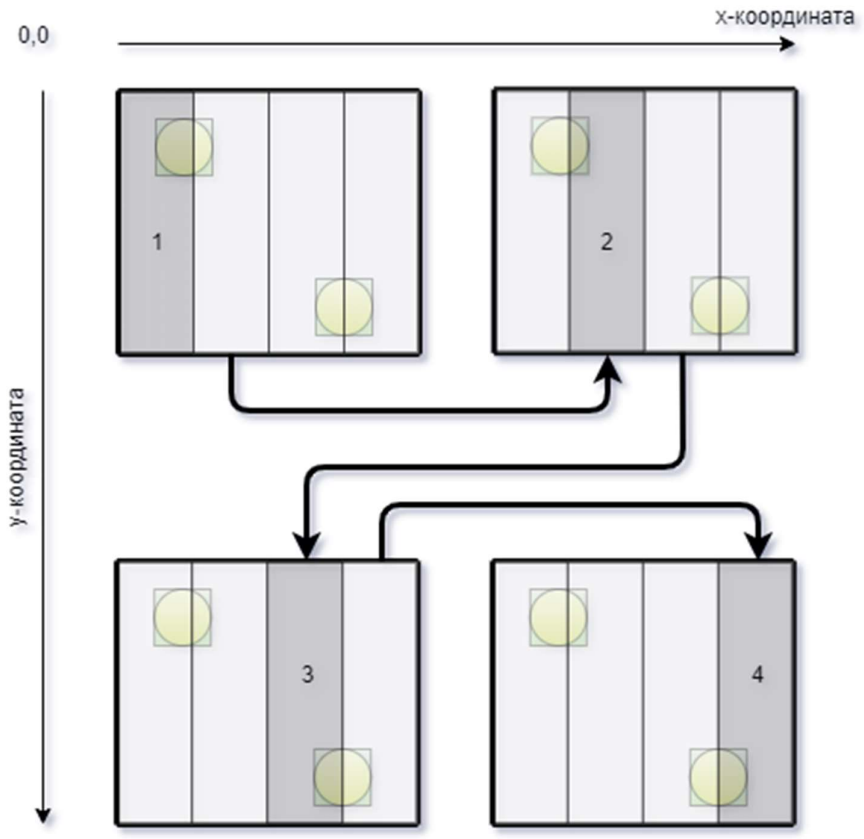

Рисунок 2.10 – Розбиття зображення на частини та обробка кожної із частин

На рисунку 2.10 графічно зображено, як відбувається розбиття початкового зображення на частини й послідовність у якій відбувається ітерація кожної частини. Кожна частина ітерується одна за одною доки не буде оброблене все зображення. Також тут можна бачити що має місце ситуація коли участок зі змінами розділяється між декількома частинами на які розбите початкове зображення й таким чином одна область може бути передана для відтворення в декількох мережевих пакетах.

Цикли за допомогою яких бідвувається пошук участків, які зазнали змін, аналогічні циклам в простому алгоритмі за виключенням того факту, що для виділення кожної наступної частини зображення, яка підлягає аналізу додається здвиг по координаті x, який дорівнює ширині кожної частини для прикладу приведемо програмну реалізацію циклу для визначення координати top в межах однієї частини (див. рис. 2.11).

```
bool is_diff = false;
for (int i = 0; i < parts; ++i) {
    int offset = part_size * i;
    int part_width = (residue && (i == parts - 1)) ? part_size + residue : part_size;

    bool done = false;
    wxRect rect;
    for (int y = 0; y < current.GetHeight(); ++y) {
        for (int x = 0; x < part_width; ++x) {
            if (!are_pixels_equal(current, previous_, x + offset, y)) {
                rect.SetTop(y);
                done = true;
                break;
            }
        }
        if (done) {
            break;
        }
    }
    if (!done) {
        LOG_INF_FMT("Nothing to send in part:%d ...", i);
        continue;
    }
```

Рисунок 2.11 – Цикл з визначення координати top та початкове розбиття зображення на частини

З рисунку 2.11 видно, що для виділення кожної наступної частини зображення використовується здвиг по x-координаті. Здвигу по y-координаті не відбувається. Слід також відмітити, що в ситуації коли різниці в пікселях на цій ітерації не було виявлено визначення наступних координат не відбувається й програма переходить до обробки наступної частини зображення, якщо така є або виходить з циклу обробки, якщо це була остання частина. Вихідний код для реалізації вдосконаленого алгоритму представлений в додатку Г.

3 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Протокол передачі даних

В рамках виконання кваліфікаційної роботи було розроблено бібліотеку для взаємодії програмний додатків в комп’ютерній мережі. Дана бібліотека базується на використанні популярної мереженої бібліотеки Boost Asio, яка предоставляє високорівневі інтерфейси для взяємодії з комп’ютерною мережею [13] - [15].

Для цілей передачі графічної інформації за допомогою розробленої бібліотеки також було розроблено протокол передачі даних з використанням бібліотеки серіалізації Google Protocol Buffers. Дана бібліотка представляє собою зручний інструмент, який можна використовувати на багатьох мовах програмування й різноманітних платформах в тому числі Windows, Linux та MacOS [16]. За допомогою даного інструменту можна задати те як дані повинні бути структуровані один раз, а потім за допомогою спеціально згенерованого коду можна зчитувати та записувати ці структури даних в різноманітні потоки даних використовуючи різноманітні мови програмування.

Для реалізації протоколу передачі даних з використання Google Protocol Buffers необхідно створити так званий proto-файл в якому необхідно описати ті структури даних, які планується використовувати при розробці [17]. Далі за допомогою спеціального консольного додатку, який поставляється з вихідними кодами бібліотеки, треба згенерувати описані структури даних. Згенеровані структури даних представляють собою класи мови програмування C++, які й потрібно використовувати в програмі для реалізації протоколу передачі даних.

При розробці нашої програмної системи будемо використвувати протокол який описаний на рисунку 3.1. Як можна бачити з рисунку протокол складається всього з одного повідомлення – тобто буде згенерований лише один класа C++. Одак не дивлячить на те, що ми описали такий маленький протокол він повністю задовольняє нашим потребам з передачі гарафічної інформації комп’ютерною мережею.

```
package netlib.viewport;

message Frame {
        enum Type {
                Start = 1;
                Algorith_Diff_Simple = 2;
                Algorith_Diff_Chunked = 3;
                Data = 4;
        }
        optional bool fullscreen = 1 [default = false];
        optional uint32 x = 2 [default = 0];
        optional uint32 y = 3 [default = 0];
        optional uint32 width = 4 [default = 0];
        optional uint32 height = 5 [default = 0];
        optional bytes data = 6;
        optional Type type = 7;
}
```

Рисунок 3.1 – Вихідний код розробленого протоколу передачі даних

В результаті обробки proto-файлу, показаного на рисунку 3.1, за допомогою вищезгаданої бібліотеки сериалізації буде згенерований клас C++. Використовуючи цей клас ми матимемо змогу передавати графічну інформацію по комп’ютерній мережі.

Згідно розробленого протоколу ми зможемо передавати інформацію про координати в які необхідно вставити зображення, яке передається, а саме x-координату, y-коорлинату, висоту та ширину зображення. Звісно ми можено передавати дані самого зображення, як байтовий масив. Функція, яка виконує упаковку повідомлення й ставить його в чергу на відправлення показана на рисунку 3.2.

```
template<typename protobuf_t>
void netlib_client::send(
    const netlib_session::sessionid_t& id,const protobuf_t& protobuf_msg) {
    std::string data_msg;
    if (!protobuf_msg.SerializeToString(&data_msg)) {
        throw std::runtime_error("Failed to serialize with protobuf");
    }
    protocol::Payload payload;
    payload.set_data(data_msg);
    netlib_sender_.send(id, payload);
}
```

Рисунок 3.2 – Функція упаковки повідомлення згідно розробленого протоколу

Інтерфейс, який показаний на рисунку 3.2 являється шаблонною функцією тому може працювати не лише з протоколом, який ми розробили, а й з протоколами, які ми розробимо в майбутньму чи які були розроблені раніше.

Для виконання зворотньої операції по розпаковці байтового масиву в C++ клас потрібно виконати кроки показані на рисунку 3.3.

```
netlib::viewport::Frame frame;
if (frame.ParseFromArray(data, bytes)) {
    if (frame.type() == netlib::viewport::Frame_Type_Data) {
        netlib_event e(NETLIB_EVENT_TYPE, netlib_event_t::NETLIB_SUB_IMAGE);
        if (frame.fullscreen()) {
            e.SetId(netlib_event_t::NETLIB_FULLSCREEN);
        }
        e.width_ = frame.width();
        e.height_ = frame.height();
        e.x_ = frame.x();
        e.y_ = frame.y();

        wxMemoryInputStream is(
            reinterpret_cast<const void*>(frame.data().data()),
            frame.data().size());
        e.img_ = wxImage(is, wxBITMAP_TYPE_PNG);
        wxPostEvent(this, e);
    }
}
```

Рисунок 3.3 – Розпаковка байтового масиву згідно розробленого протоколу

На рисунку 3.3 показана операція розпаковки байтового масиву згідно розробленого протоколу, передача інформації для подальної оброки й вставки зображення у вікні перегляду.

Згідно розробленого протоколу також можна відсилати та приймати службові команди типу запустити демонстрацію екрану чи змінити алгоритм згідно якого відбувається пошук частин зображення, які змінилися.

3.2 Діаграма станів об’єкту Viewport

В розробленій програмній системі, яка побудована згідно архітектури клієнт-сервер в якості сервера виступає додаток який запущений на тій стороні звідки графічна інформація передається для відображення. В якості клієнта виступає програмний додаток з графічним інтерфейсом користувача, який займається відображенням графічної інформації отриманої з серверної частини.

Клієнт та сервер спілкуються по каналу зв’язку в комп’ютерній мережі. В момент коли клієнтський додаток остаточно підключився до серверу в роботу вступає об’єкт під назвою Viewport. Головна задача, яка стоїть перед цим об’єктом це збір та аналіз графічної інформації отриманої при захваті зображення екрану. Дану роботу він виконує асинхронно – запускається в окремому потоці. Діаграма станів даного об’єкта показана на рисунку 3.4.

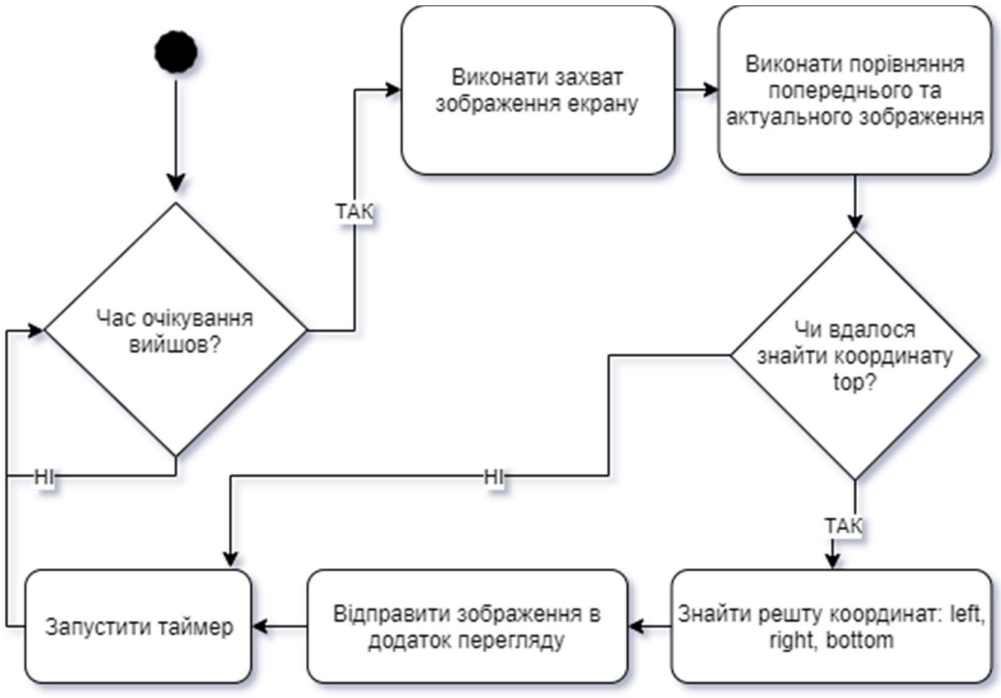

Рисунок 3.4 – Діаграма станів об’єкта Viewport

На рисунку 3.4 показана діаграма станів об’єкту Viewport аналізуючи яку можна бачити, що відразу після встановлення зв’язку між клієнтом та сервером даний об’єкт запускає таймер. В момент коли час таймеру сплинув на виконання запускається алгоритм пошуку різниці між двома зображеннями, який в першу чергу виконує захват зображення екрану, щоб було з чим порівнювати зображення на минулій ітерації. Наступним кроком виконується порівняння двох зображень й в першу чергу виконуются активності по пошуку координати top. У випадку коли алгоритму вдається знайти координату top він продовжує пошук решти координат, таких як left, right та bottom й в результаті відправляє зображення на обробку на сторону клієнта, який представляє собою програму перегляду зображень. Якщо алгоритм не знаходить координати top він повторно запускає таймер на виконання.

Даний об’єкт являється своєрідним сховищем алгоритмів пошуку різниці двох зображень. Згідно розробленого протоколу зі сторони клієнта можна ініціювати зміну алгоритму в режимі реального часу.

3.3 Діаграма взаємодій компонентів програмної системи

Як раніше було сказано розроблена програмна система складається з двох пограмних додатків, які разом реалізують клієнт-серверну архітектуру й спілкуються між собою через комп’ютерну мережу за допомогою розробленого з використанням бібліотеки Google Protocol Buffers протоколу. За допомогою діаграми взаємодій між двома цими об’єктами можна легко пояснити, які процеси тут мають місце [18]. Дана діаграма показана на рисунку 3.5.

З діаграми видно, що підключення ініціює клієнтська частина в той час, як серверна працює в режимі очікування нових підключень.

В момент запуску клієнт формує запрос на підключення згідно розробленого протоколу. Сервер зі свої сторони приймає це підключення, створює нову сесію й висилає підтвердження про те що даний клієнт був зареєстрований на сервері.

Після отримання підтвердження на клієнтській стороні формується й висилається запрос на запуск сесії з моніторингу зображення екрану комп’ютеру. Серверна частина після отримання запросу на запуск функцій моніторингу робить знімок екрану монітора, висилає його на клієнтську частину, виконує інінціалізацію об’єкта типу Viewport й запускає таймер оновлення зображення. В цій точці на діаграмі всі сервіси ініціалізовано та запущено.

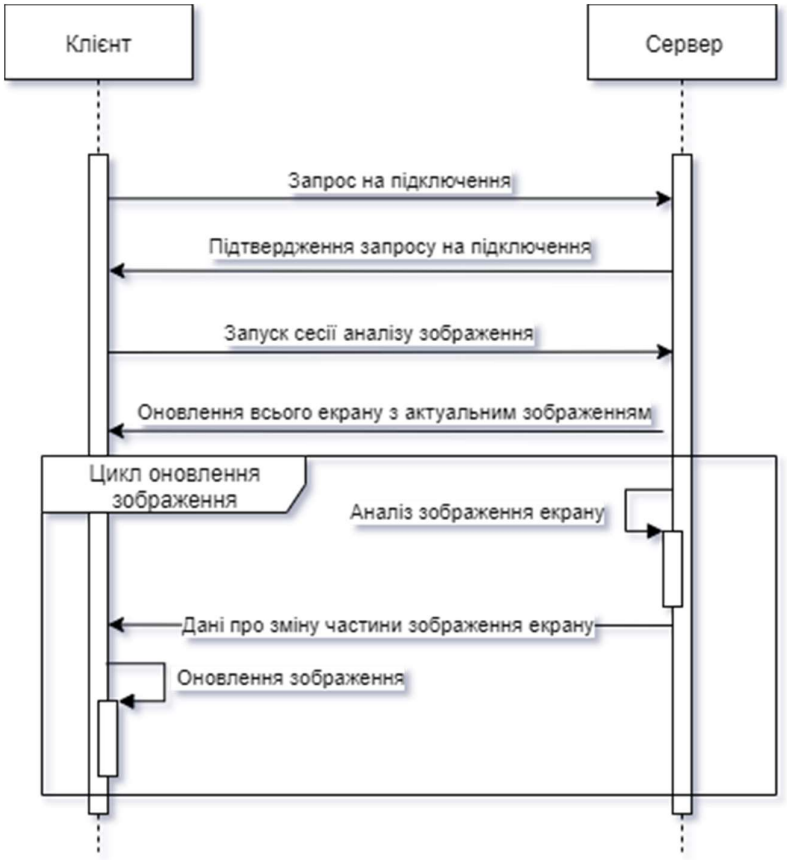

Рисунок 3.5 – Діаграма взаємодій клієнтської та серверної частин розробленої програмної системи

Сервер переходить в режим постійного спостереження й моніторингу за зображенням екрану комп’ютера. За таймером з періодичністю в 300 мілісекунд виконується аналіз змін зображення екрану й у разі якщо такі зміни мають місце бути на клієнтську сторону висилаються відповідні пакети з інформацією про ці зміни. Клієнт у разі отримання пакетів зі змінами зображення виконує оновлення

вікна перегляду зі своєї сторони. Таким чином у вікні перегляду зображення можна спостерігати, в режимі реального часу, за тим які зміни відбуваються на екрані віддаленого комп’ютера.

4 СПЕЦИФІКАЦІЯ РОЗРОБЛЕНОЇ ПРОГРАМИ

4.1 Огляд компонентів розробленої системи

Розроблена система складається з двох додатків так званої серверної та клієнтської частини, які спілкуються між собою за допомогою розробленої в ході кваліфікаційної роботи бібліотеки та протоколу передачі даних.

Серверна частини не має графічного інтерфейсу користувача й запускається, як консольний додаток з командної строки операційної системи Windows. Запущений серверний додаток показано на рисунку 4.1.

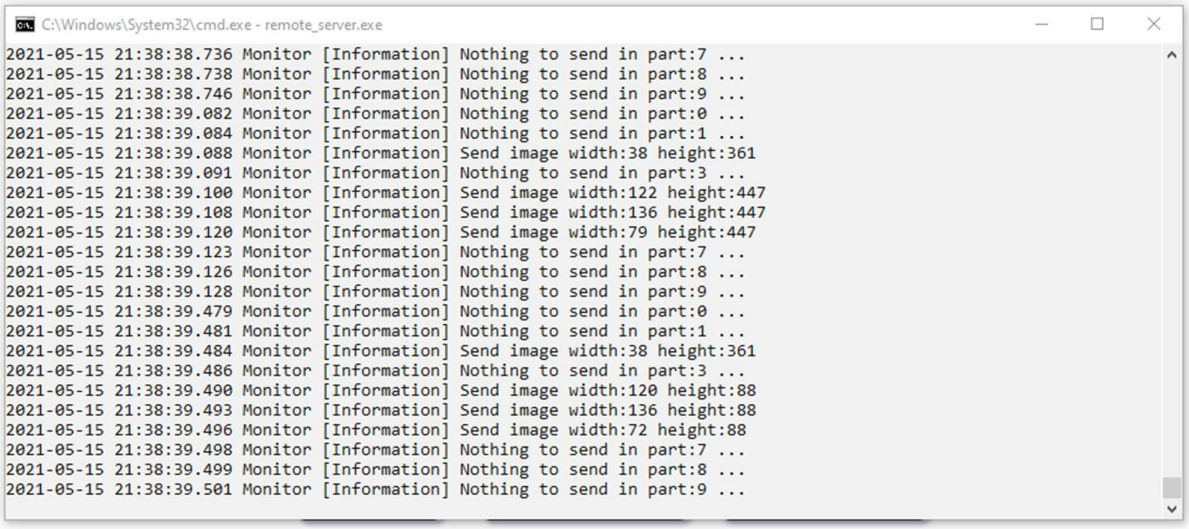

Рисунок 4.1 - Запущений серверний програмний додаток

Не дивлячись на те, що розробка програмного додатку представленого в кваліфікаційній роботі велась на операційній системі Windows за бажанням розроблені компоненти можуть бути портовані з невеликими змінами на інші платформи [19] – [20].

З рисунку 4.1, видно що в запущеному вигляді сервер логує роботу, яку він виконує. Зокрема тут можна спостерігати за тим чи відсилаються якість зміни зображення екрану на сторону клієнта. У випадку коли пакети зі змінами

надсилаються на сервері логується дата, час та розмір зображення, яке було запаковане в такий пакет.

Клієнтська частина розробленої програмної системи наділена графічним інтерфейсом користувача, який представлений на рисунку 4.2.

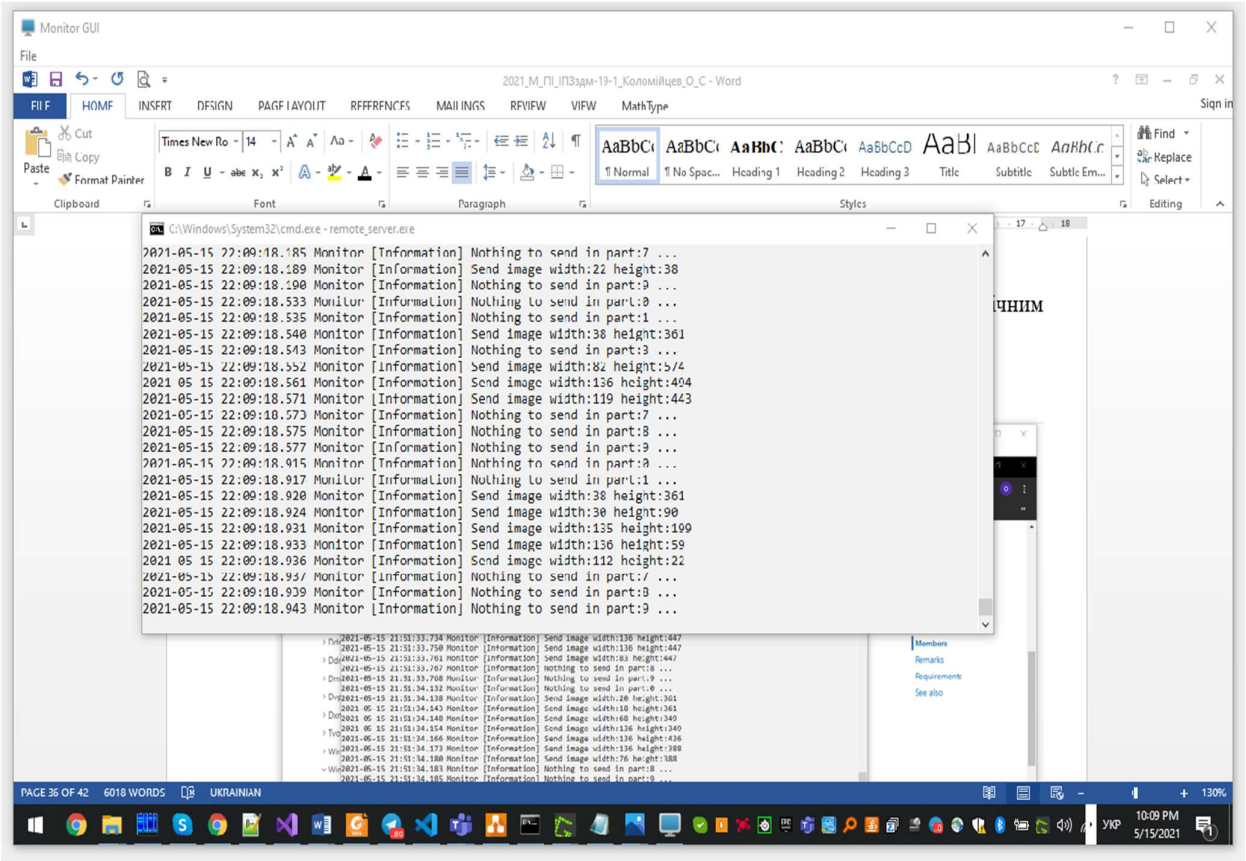

Рисунок 4.2 – Графічний інтерфейс користувача програми перегляду віддаленого екрану

Хоча інтерфейс показаний на рисунку 4.2 не наділений великою кількістю функцій для розробленого прототипу їх достатньо. Окрім вікна перегляду зображення віддаленого монітору тут є меню, за допомогою якого можна виконати віддалену зміну алгоритму з яким працює сервер. Так в режимі реального часу на серверну сторону можна відправити команду зміни алгоритму на вдосконалений чи простий. На серверній частині результат виконання такої команди можна спостерігати за допомогою інформації, яку він логує в консоль (див. рис. 4.3).

```
1-05-15 22:02:02.652 Monitor [Information] Data received...
1-05-15 22:02:02.654 Monitor [Information] Change algorithm to diff simple
1-05-15 22:02:02.656 Monitor [Information] Nothing to send in part:0 ...
1-05-15 22:02:02.661 Monitor [Information] Send image width:20 height:361
1-05-15 22:02:02.665 Monitor [Information] Send image width:18 height:361
1-05-15 22:02:02.667 Monitor [Information] Nothing to send in part:3 ...
1-05-15 22:02:02.673 Monitor [Information] Nothing to send in part:4 ...
1-05-15 22:02:02.678 Monitor [Information] Send image width:23 height:233
1-05-15 22:02:02.680 Monitor [Information] Nothing to send in part:6 ...
1-05-15 22:02:02.682 Monitor [Information] Nothing to send in part:7 ...
1-05-15 22:02:02.686 Monitor [Information] Nothing to send in part:8 ...
1-05-15 22:02:02.688 Monitor [Information] Nothing to send in part:9 ...
1-05-15 22:02:03.119 Monitor [Information] Send image width:1093 height:438
1-05-15 22:02:03.506 Monitor [Information] Send image width:1093 height:441
1-05-15 22:02:03.882 Monitor [Information] Send image width:478 height:362
1-05-15 22:02:04.280 Monitor [Information] Send image width:478 height:364
1-05-15 22:02:04.664 Monitor [Information] Send image width:479 height:361
1-05-15 22:02:05.031 Monitor [Information] Send image width:479 height:362
1-05-15 22:02:05.399 Monitor [Information] Send image width:479 height:362
1-05-15 22:02:05.801 Monitor [Information] Send image width:479 height:362
1-05-15 22:02:06.237 Monitor [Information] Send image width:993 height:660
1-05-15 22:02:06.622 Monitor [Information] Send image width:479 height:361
1-05-15 22:02:07.001 Monitor [Information] Send image width:479 height:361
1-05-15 22:02:18.236 Monitor [Information] Data received...
1-05-15 22:02:20.633 Monitor [Information] Change algorithm to diff chunked
1-05-15 22:02:20.999 Monitor [Information] Send image width:623 height:380
```

Рисунок 4.3 – Логування зміни алгоритмів

На рисунку 4.3 показано логування зміни алгоритму знаходження різниці між зображеннями з покращеного на простий й в зворотньому напрямку. Пункти меню за допомогою яких це можна зробити показані на рисунку 4.4.

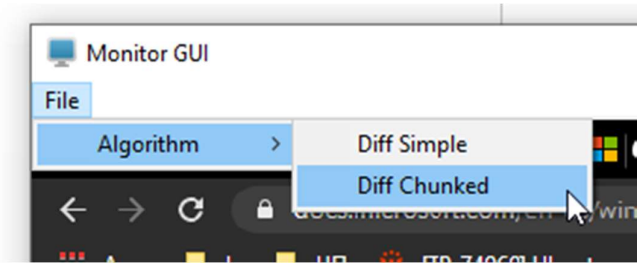

Рисунок 4.4 – Пункти меню для зміни алгоритму знаходження різниці двох зображень

За допомогою пунктів меню показаних на рисунку 4.4 досить зручно переключатися між різними алгоритмами й спостерігати за тим як кожен з них працює в різних умовах. Так в розробленому прототипі їх було додано лише два, але в майбутньому з розвитком даного продукту цей список неодмінно розшириться. Так в перспективі можна додати конвертацію зображення в різні формати, чи переведення режиму перегляду в монохромний.

ВИСНОВКИ

В ході виконання магістерської роботи були проаналізовані й імплементовані методи оптимізації передачі графічної інформації в розподілених системах. Розроблено два алгоритми пошуку різниці двох зображень, які успішно інтегровані й протестовані в розробленій програмній системі. З використанням розроблених алгоритмів вдалося досягти прийнятної швидкості роботи програмного додатку з функцією спостереження за віддаленим робочим столом.

В рамках кваліфікаційної роботи було розроблено розподілену систему, яка складається з двох додатків: серверної та клієнтської частини. Сервер представлений у вигляді консольного додатку з функцією логування роботи системи, він виконує захват зображення екрану віддаленого комп'ютеру та формує й передає інформацію про зміну зображення екрану на клієнтську сторону. Клієнтська частина представляє собою додаток з графічним інтерфейсом користувача головною функцією якої є відображення екрану віддаленого комп'ютеру й коректна обробка пакетів з інформацією про зміни в зображенні віддаленого екрану.

Так як клієнт-серверні додатки спілкуються через комп'ютерну мережу для цих цілей була розроблена бібліотека й протокол передачі графічного зображення через таку мережу. Дані компоненти успішно інтегровані й використовуються в розробленій програмній системі.

Питання розглянуті в магістерській роботі є доволі актуальними для сьогодення, тому розроблений прототип неодмінно знайде застосування в реальному житті й буде розвиватися з плином часу. На базі розробленого прототипу можна побудувати системи для спостереження за роботою працівників які в своїй роботі використовують комп'ютери і є нагальна потреба спостереження за їхньою роботою. Іншим напрямком розвитку даного проекту є віддалений моніторинг та управлінні робочими станціями. Якщо розширити даний прототит функціями передачі натискання клавіш та курсору мишки він неодмінно знайде

застовування у системних адміністраторів, які дуже часто використовують такого робу системи для доступу до віддалених робочих станцій з метою налаштування програмного забезпечення та іншого.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Павловская, Т. А. C/C++. Программирование на языке высокого уровня [Текст] : учеб. / Т. А. Павловская. – СПб. : Питер, 2003. – 461 с.

2. Попов, И. И. Языки программирования [Текст] : учеб. пособие / И. И. Попов, Т. Л. Партыка. – М. : ФОРУМ, 2008. – 400 с.

3. Голицина, О. Л. Основы алгоритмизации и программирования [Текст] : учеб. пособие / О. Л. Голицин, И. И. Попов. – М. : ФОРУМ, 2008. – 432 с.

4. Свердлов, С. З. Языки программирования и методы трансляции [Текст] : учеб. пособие / С. З. Свердлов. – СПб. : Питер, 2007. – 638 с.

5. Камаев, В. А. Технологии программирования [Текст] : учеб. / В. А. Камаев, В. В. Костерин. – М. : Высш. шк., 2006. – 454 с.

6. Крылов, Е. В. Техника разработки программ [Текст]. В 2 кн. Кн.1. Програмирование на языке высокого уровня : учеб. / Е. В. Крылов, В. А. Острейковский, Н. Г. Типикин. – М. : Высш. шк., 2007. – 375 с.

7. Крылов, Е. В. Техника разработки программ [Текст]. В 2 кн. Кн.2. Технология, надежность и качество програмного обеспечения: учеб. / Е. В. Крылов, В. А. Острейковский, Н. Г. Типикин. – М. : Высш. шк., 2008. – 469 с.

8. Лаптев, В. В. C++. Объектно-ориентированное программирование [Текст] : учеб. пособие / В. В. Лаптев, А. В. Морозов, А. В. Бокова. – СПб. : Питер, 2008. – 464 с.

9. Хорев, П. Б. Объектно-ориентированное программирование [Текст] : учеб. пособие / П. Б. Хорев. – М. : Академия, 2011. – 448 с.

10. Буч, Г. Язык UML. Руководство пользователя [Текст] / Г. Буч. – М. : ДМК Пресс, 2006. – 248 с.

11. Фаулер, М. Рефакторинг. Улучшение существующего кода [Текст] / М. Фаулер, К. Бек, Д. Брант, У. Апдайк, Д. Робертс. – СПб. : Символ-Плюс, 2010. – 225 с.

12. Мандел, Т. Разработка пользовательского интерфейса [Текст] / Т. Мандел. – М. : ДМК Пресс, 2008. – 416 с.

13. Стефенс, Д. Р. C++. Сборник Рецептов [Текст] / Д. Р. Стефенс, К. Диггинс, Д. Турканис, Д. Когсуэлл. – М. : Кудиц-Пресс, 2007. – 624 с.

14. Романов, Е. Л. Практикум по програмированию на C++ [Текст] : учеб. пособие / Е. Л. Романов. – СПб. : БХВ-Петербург, 2004. – 432 с.

15. Страуструп, Б. Дизайн и эволюция C++ [Текст] / Б. Страуструп. – М. : ДМК Пресс, 2011. – 488 с.

16. Ерохин А. Л. Моделирование и управление трафиков при решении задачи повышения эффективности обслуживания в web / А. Л. Ерохин, О. П. Турута // Системи обробки інформації. - 2014. - Вип. 2. - С. 159-162.

17. Bielievtsov S. Network technology for transmission of visual information / Bielievtsov S., Ruban I., Smelyakov K., Sumtsov D. // CEUR Workshop Proceedings, 2018, vol. 2318. – P. 161-175.

18. Кормен, Т. Алгоритмы. Вводный курс [Текст] / Т. Кормен. – М. : ООО И. Д. Вильямс, 2014. – 208 с.

19. Кормен, Т. Алгоритмы. Построение и анализ [Текст] / Т. Кормен. – М. : ООО И. Д. Вильямс, 2013. – 1328 с.

20. Yerokhin, A. Information model for heat and mass transfer processes evaluation / A. Yerokhin, H. Zatserklyanyi, A. Babii, O. Turuta, O. Zolotukhin // Management Information System and Devices. All-Ukr. Sci. Interdep. Mag. – 2019. – № 176. – P. 4-9.