

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ВІДОБРАЖЕННЯ РЕЄСТРАЦІЇ ПОВІТРЯНОЇ
ОБСТАНОВКИ У 3D
DISPLAY OF AIR SITUATION REGISTRATION IN 3D

Виконав(ла): студент(ка) 4 курсу, групи КНТ-110
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КУЧЕР А.О.

(ПРІЗВИЩЕ та ініціали)

Керівник ТАБУНЩИК Г.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КУЧЕРА Андрія Олеговича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Відображення реєстрації повітряної обстановки у3D
Display of Air Situation Registration in 3D

керівник проєкту (роботи) к.т.н., професор, ТАБУНЩИК Галина Володимирівна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вхідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір програмних засобів. 3. Основні рішення з реалізації компонентів системи. 4. Експлуатація та тестування програми. Висновки. Перелік джерел посилання. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ТАБУНЩИК Г.В, професор		
Нормоконтролер	ДЕЙНЕГА Л.Ю, ст. викладач		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-5 тижні	Розділ 1
3	Розробка архітектури системи	6-8 тиждень	Розділ 2
4	Розробка системи	9-11 тижні	Розділ 3
5	Тестування та експериментальне дослідження системи	12 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї	13 тиждень	Додатки
7	Нормоконтроль та рецензування	13 тиждень	
8	Захист роботи	14 тиждень	

Студент(ка)

Підпис

(підпис)

Андрій КУЧЕР

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Підпис

(підпис)

Галина ТАБУНЩИК

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 118 сторінок, 63 рисунки, 10 таблиць, 4 додатки, 11 джерел.

ТРЬОХ ВИМІРНЕ ВІДОБРАЖЕННЯ , СТРУКТУРИЗАЦІЯ ДАНИХ, OPENGL, ГРАФІКА, ГЕОГРАФІЧНІ ПЕРЕТВОРЕННЯ, ФАЙЛИ.

Об'єкт дипломної роботи – системи моніторингу, аналізу, «прослуховування» та реєстрації повітряної обстановки.

Предмет дослідження – реалізація та розробка методів відображення результатів реєстрації повітряної обстановки.

Мета роботи – покращення методів аналізу повітряної обстановки за рахунок створення програмного забезпечення, що може зчитувати попередньо записані файли реєстрації руху повітряних тіл у просторі, організовувати та групувати зчитану інформацію у зручний для навігації, користування і аналізу вигляд та відобразити подану інформацію у двох та трьох вимірному просторі.

Результати. Розроблено програмний застосунок що дозволяє відображати реєстрацію повітряної обстановки у двох та трьох вимірному просторі.

Матеріали, методи та технічні засоби: мова програмування C++, середовище програмування IDE “Qt Creator”, OpenGL, додаткові бібліотеки.

Висновки. Створено програмний засіб який дозволяє зчитувати записану у файлі інформацію повітряної обстановки, з подальшим відображенням її у двох та трьох вимірному вигляді. Реалізовано функції встановлення та змінення конкретного проміжку часу для відображення, вибірки маршруту по заданим критеріям.

Галузь використання – аналіз файлів реєстрації руху повітряних тіл для розслідування нещасних випадків, аналізу стану повітряного простору.

ABSTRACT

Explanatory note to the bachelor's thesis: 118 pages, 63 figures, 10 tables, 4 appendixes, 11 sources.

THREE-DIMENSIONAL DISPLAY, DATA STRUCTURING, OPENGL, GRAPHICS, GEOGRAPHIC TRANSFORMATIONS, FILES.

The object of the thesis is systems for monitoring, analyzing, “listening” and recording the air situation.

The subject of the research is the implementation and development of methods for displaying the results of air situation registration.

The aim of the work is to improve the methods of air situation analysis by creating software that can read pre-recorded files of airborne movement registration in space, organize and group the read information into a convenient form for navigation, use and analysis, and display the information in two and three dimensions.

Results. A software application has been developed that allows to display the air situation registration in two and three dimensions.

Materials, methods and techniques: C++ programming language, IDE “Qt Creator”, OpenGL, additional libraries freeGlut.

Conclusions. A software tool has been created that allows you to read the air situation information recorded in the file, with its subsequent display in two and three dimensions. The functions of setting and changing a specific time interval for display, route selection according to specified criteria are implemented.

Application area - analysis of air traffic files for accident investigation,
 airspace state analysis.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Передмова	11
1.2 Огляд існуючих рішень	12
1.2.1 Функціонал програми Google Earth	12
1.2.2 Функціонал програми FlightRadar24.....	17
1.2.3 Порівняння існуючих продуктів	22
1.3 Постановка завдання.....	23
1.3.1 Функціональність системи.....	23
1.3.2 Не функціональні вимоги до системи.....	24
1.4 Висновок за розділом 1.....	25
2 Вибір програмних засобів	26
2.1 Вибір мови програмування	26
2.2 Вибір середовища розробки.....	31
2.3 Вибір інтерфейсу для роботи з графікою	38
2.4 Висновок за розділом 2.....	43
3 Основні рішення з реалізації компонентів системи	44
3.1 Реалізація інтерфейсу користувача	44
3.2 Структура файлів реєстрації	49
3.3 Алгоритм читання збереженої реєстрації з файлу	50
3.4 Організація та структура зберігання зчитаної інформації.....	51

3.5 Сімейство класів роботи з OpenGL	53
3.6 Алгоритми.....	57
3.7 Висновок за розділом 3.....	62
4 Експлуатація та тестування програми	63
4.1 Опис застосунку	63
4.2 Умови виконання програми	63
4.3 Інструкція по експлуатації	64
4.3.1 Характеристики програми	64
4.3.2 Перелік файлів що потрібні для функціонування програми	65
4.3.3 Звернення до програми.....	65
4.3.4 Вихідні й вхідні дані.....	66
4.4 Методика та результати тестування.....	66
Висновки	73
Перелік джерел посилання	74
Додаток А Технічне завдання	75
Додаток Б Опис програми	80
Додаток В Текст програми.....	85
Додаток Г	111

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧОК

GUI – Graphical User Interface (графічний інтерфейс користувача)

IDE – Integrated Drive Electronics (інтегроване середовище розробки)

ДІО – додаткова інформація про об'єкт

ІО – інформація про об'єкт

ІСЛ – інформація про стан локатора

ІТ – інформаційні технології

ОЗП – оперативна пам'ять

ПЗ – програмний засіб

ПО – повітряна обстановка

ЧВ – часова відмітка

ВСТУП

У сучасному світі, використання сучасних комп'ютерних технологій у нашому житті росте з кожним днем. Комп'ютерні технології кожен день проникають у різні галузі нашого життя. Також з появою та розвитком ІТ сфери, було створено багато різних професій та галузей пов'язаних з управлінням, вдосконаленням, розвитком та аналізом комп'ютерної індустрії та сфер діяльності загалом.

При теперішньому рівні розвитку та впроваджені ІТ технологій у сфери життєвої діяльності, вони дозволяють виконувати дії, які раніше були недоступні. Зокрема ІТ технології дозволяють реєструвати великі потоки та об'єми різноманітної інформації про виконання основних задач, роботу та стан окремих систем та іншу корисну інформацію. Без використання ІТ велика частина цієї інформації була б недоступна. Ця інформація має велику цінність для розвитку сфер діяльності, оскільки вона надає можливість для аналізу її людиною. Подальшого налагодження процесів, розробки нових та покращення старих алгоритмів, створення нових систем базуючись на результатах аналізу записаної інформації. Завдяки цим діям ефективність процесів збільшується.

На сьогодні не існує універсального формату реєстрації або програмного засобу, який би охоплював усі сфери діяльності людини і у той самий час міг би подавати інформацію у зручному для читання та аналізу вигляді згідно вимог користувача. Тому існує потреба реалізації спеціальних інструментів, які могли б враховувати специфіку сфери діяльності та реалізовувати всі вимоги кінцевого користувача. Оскільки розробка таких інструментів виконується в загалом для масштабних систем, часто вимоги користувача можуть змінюватися упродовж користування, під час розробки або удосконалення основного продукту.

Кожна із сфер діяльності потребує аналізу різних систем. Кожна система має свої нюанси обробки, реєстрації та відображення інформації. Також для кожної з систем існує різне подання інформації, з перевагами й недоліками, які у свою чергу грають важливу роль у зручності аналізу поданої інформації.

Однією зі сфер що вимагає створення додаткових інструментів для аналізу реєстрації, є сфера моніторингу руху об'єктів у повітряному просторі. Розглядаючи тенденцію росту зацікавленості суспільства у повітряній індустрії, ми можемо спостерігати значний зріст числа авіакомпаній, які використовують літаки для пасажирських та вантажних перевезень. Крім того, приватних підприємств з власними гвинтокрилами та іншими літаючими апаратами стає все більше, а військова промисловість, зокрема, нахилена до використання безпілотних літальних апаратів (БПЛА) у своїх операціях.

З цього можна зробити висновок, що зацікавленість у моніторингу повітряного простору буде лише зростати в майбутньому. Цей розвиток викликає необхідність удосконалення систем моніторингу та аналізу повітряної обстановки. Виникає потреба у точному та правильному відображенні зареєстрованої інформації.

У зв'язку з цим, дослідження та впровадження новітніх методів аналізу повітряної обстановки та нових методів подання та відображення цієї інформації має сприяє прискоренню удосконалення, розвитку та поліпшенню цієї сфери.

У дипломній роботі буде розроблено програму що забезпечує зручне, точне, швидке та зрозуміле відображення результатів реєстрації руху об'єктів у повітряному просторі - файлів реєстрації вихідної інформації радіолокаційних систем.

Для вирішення поставленого завдання треба проаналізувати готові рішення, виділити функціональні вимоги та нефункціональні вимоги. Вибрати середу розробки, мову програмування, інтерфейс для роботи з

графікою та протокол обміну інформації. Розробити рішення з реалізації компонентів системи. Протестувати та увести застосунок у експлуатацію.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Передмова

У світі використовується велика кількість систем для отримання інформації про стан повітряного простору. Деякі системи використовують сенсори для отримання інформації про об'єкт у повітрі, деякі спілкуються з літаками за допомогою каналів зв'язку. Такі системи мають можливість реєстрації своєї роботи у бінарному файлі які містять у собі інформацію про рух повітряних об'єктів. Ця інформація зберігається у вигляді точок з координатами та прив'язкою до часу.

Оскільки аналізувати такі файли у їх початковому вигляді не зручно, було прийнято рішення розробити програмне забезпечення, яке зможе швидко зчитати файл реєстрації, відобразити інформацію без спотворень, а також у зручному та зрозумілому для користувача вигляді.

Є декілька сервісів які надають потрібний для користувача функціонал. Але вони мають певні недоліки, що не дозволяють використовувати ці сервіси для повноцінного аналізу реєстрації.

Google Earth- програмний засіб для відображення географічної інформації у трьох-вимірній проекції.

Flight Radar- веб-застосунок для відображення руху літаків у реальному часі.

Для розуміння який саме інтерфейс буде зручним та інформативним для користувача, які функції будуть корисні, яке подання і групування інформації буде найкращими та який саме вигляд повинна мати програма для зручного користування, було прийнято рішення проаналізувати подібні вище зазначені сервіси. Функціонал та зовнішній вигляд кінцевого продукту буде створено, беручи до уваги результати аналізу вибраних сервісів.

1.2 Огляд існуючих рішень

1.2.1 Функціонал програми Google Earth

Як приклад гарної побудови інтерфейсу, реалізації відображення місцевості у трьох-вимірному просторі та управлінням відображення мапи було вибрано один із найпопулярніших сервісів перегляду мап-Google Earth.

Google Earth – це програмний засіб який дозволяє використовувати хмарні сервіси Google для відображення супутникових мап та маніпуляції з ними. Головне вікно програми представляє собою GUI розбитий на декілька панелей (рис. 1.1).

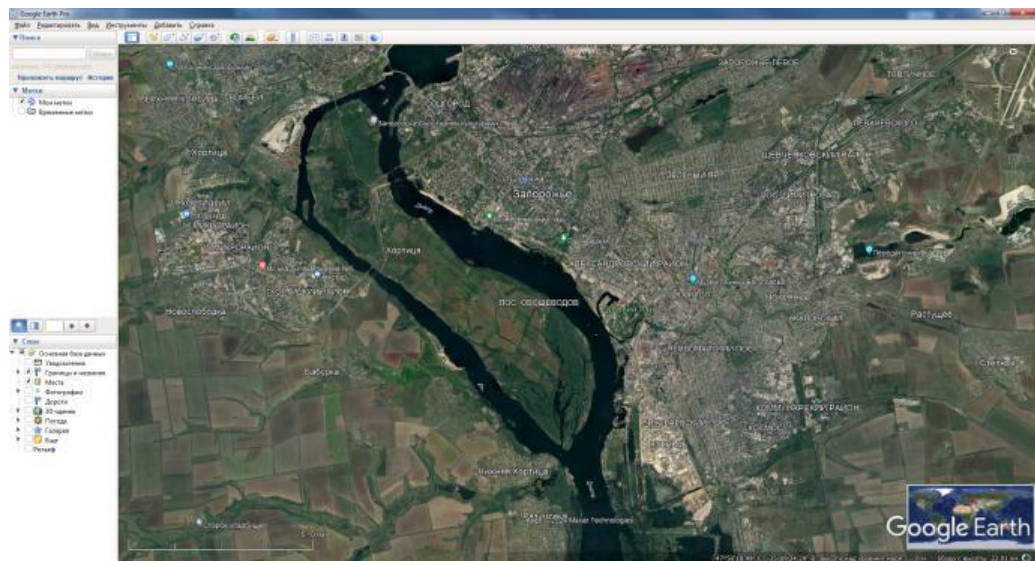


Рисунок 1.1- Вигляд головного вікна програми Google Earth

Більшу частину екрану займає вікно перегляду що дозволяє відобразити більшу кількість графічної інформації. Відображення будується у 3Д вигляді, тим самим воно не спотворює уявлення про розмір та відстань.

Панель інструментів знаходиться над вікном перегляду, що надає швидкий та зручний доступ до неї (рис. 1.2).



Рисунок1.2- Вигляд панель інструментів

Google Earth дозволяє додавати користувацькі маркери, фігури, лінії, шари для мапи та маршрути. Користувач має гнучкий інструмент для виміру дистанції, який дозволяє дізнатися дистанцію по лінії, шляху, має можливість рахувати периметр та область прямокутників та кругів (рис. 1.3-рис. 1.6).

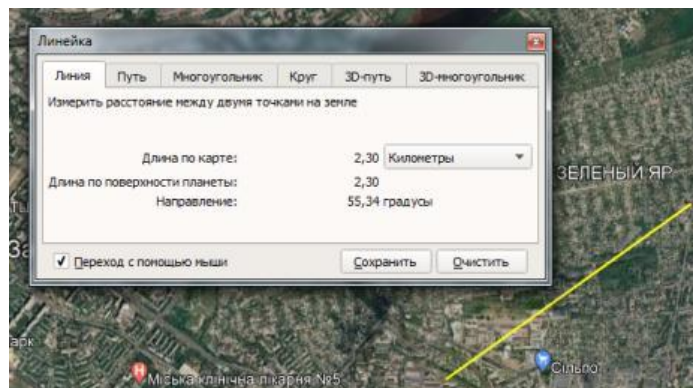


Рисунок 1.3-Вимірювання методом «Лінії»

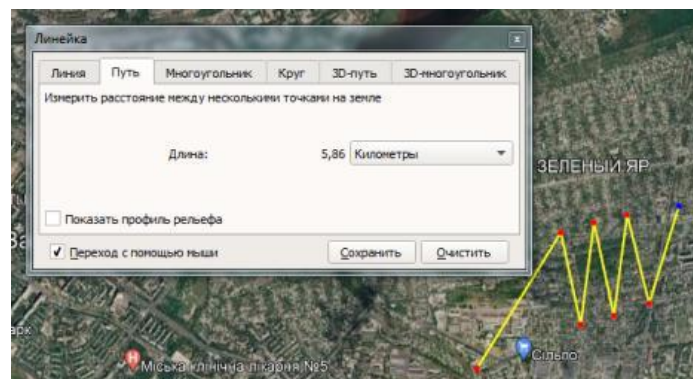


Рисунок 1.4-Вимірювання методом «Шлях»

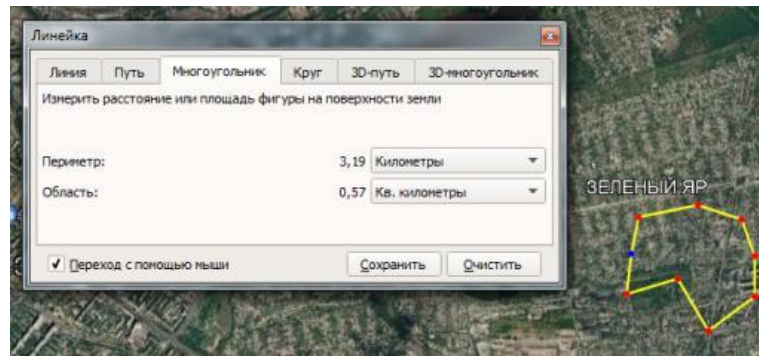


Рисунок 1.5-Вимірювання методом «Многокутник»

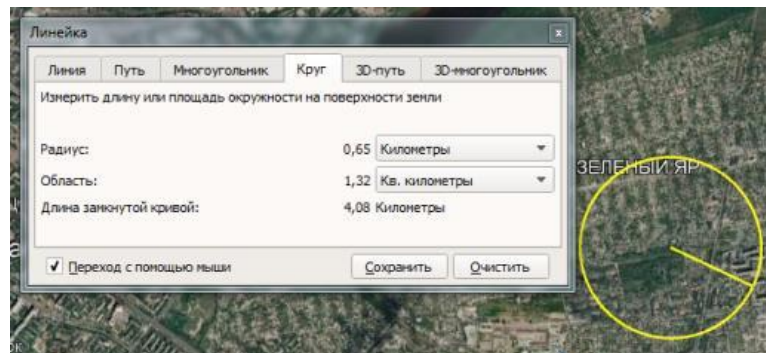


Рисунок 1.6- Вимірювання методом «Круг»

Панель пошуку знаходиться у лівому верхньому куту вікна. Займаючи небагато місця, користувач може швидко знайти потрібний йому об'єкт, місце або координати. Також панель має вікно перегляду історії пошуку та можливість прокласти маршрут до зазначеного місця (рис. 1.7).

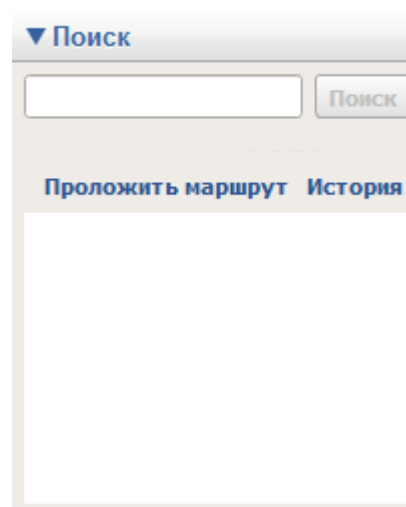


Рисунок 1.7-Вигляд панелі рядка пошуку

Панель міток відображає усі елементи які були додані користувачем. Також структура панелі міток дозволяє групувати користувацькі елементи у папки, дозволяючи створювати комплексні маршрути (рис. 1.8).

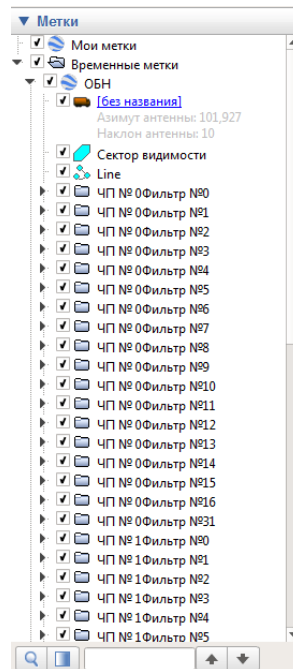


Рисунок 1.8- Вигляд панелі міток

Google Earth має дуже зручну реалізацію вибірки часового відрізка у вигляді повзунка, який дозволяє точно задати часовий діапазон відображення, прокручувати заданий діапазон. Має налаштування прискорення часу та повторного відтворення (рис. 1.9).

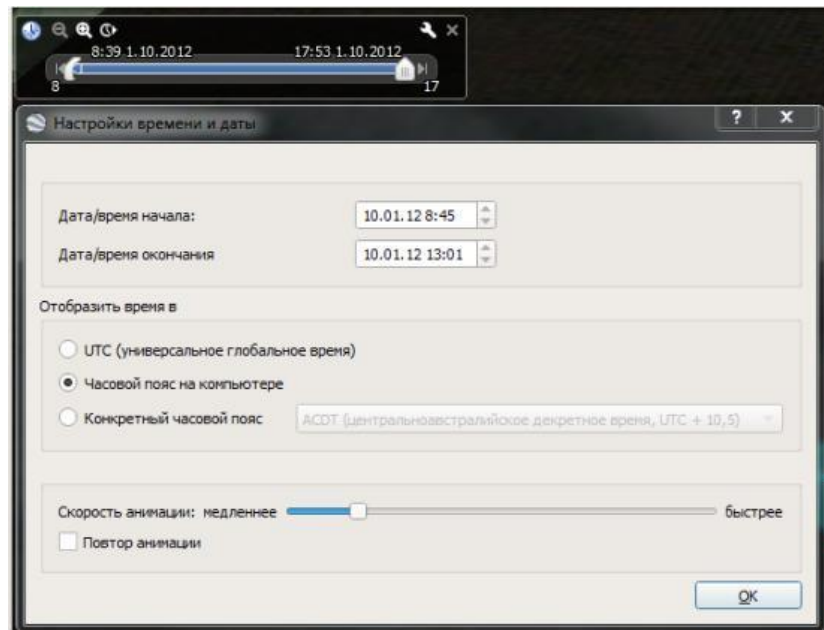


Рисунок 1.9- Вигляд повзунка часу та вікна налаштувань цього повзунка

Однак у Google Earth немає функціоналу який дозволяв би відкривати реєстрацію руху повітряних суден. Для реалізації подібного функціоналу Google Earth підтримує формат файлів типу “.kml” які зберігають у собі інформацію щодо користувацьких елементів які були додані на мапу. Приблизний вигляд однієї з реєстрацій наведено на рисунку 1.10.

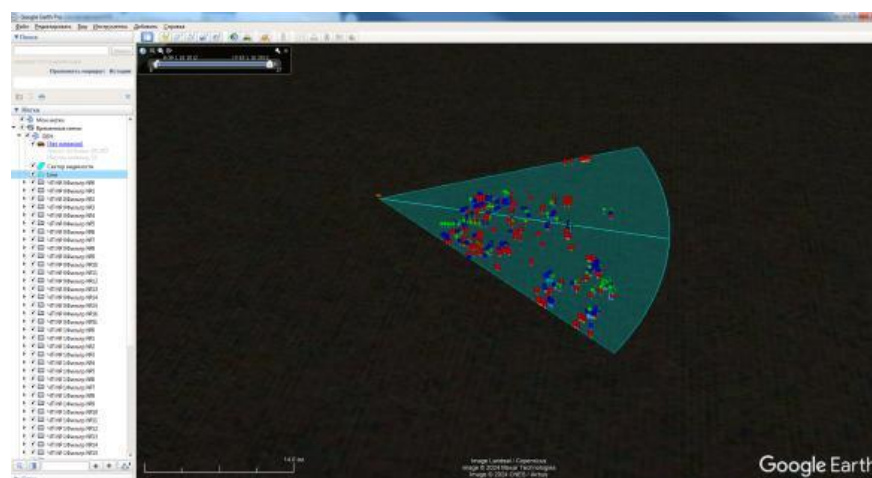


Рисунок 1.10- Результат парсингу файлу реєстрації у формат .kml, з подальшим відкриттям його у Google Earth

Отже, проаналізувавши застосунок Google Earth, мною було зазначено наступні переваги:

- зручний та простий для користувача інтерфейс;
- завжди остання версія мап;
- багато стандартних інструментів;
- гнучкість налаштування середовища (накладання додаткових шарів на мапу, вибір версії мапи, відображення погоди та часу доби);
- можливість маніпулювати часовим відрізком;
- можливість перегляду тривимірних зображень, що дозволяє користувачам отримати реалістичне представлення про об'єкти та їх оточення;
- велика кількість географічних даних, які охоплюють практично всю планету.

Але основними недоліками цієї програми є:

- підтримка тільки формату .kml розміром не більше 100 Мб;
- програма є платною.

1.2.2 Функціонал програми FlightRadar24

Як приклад гарної реалізації структурування інформації, зручності пошуку, вибірки та сортування польотів по заданим критеріям було вибрано сервіс реєстрації та відображення FlightRadar24.

FlightRadar24 – це веб-сервіс та мобільний додаток який надає користувачам можливість відстежувати рух повітряних суден у режимі реального часу. Інтерфейс головної веб сторінки представляє собою мапу з зображеними на ній літаками які оновлюються у реальному часі (рис. 1.11). Елементів керування виконані у вигляді «островів». Це дозволило використати всю область сторінки для відображення корисної інформації.

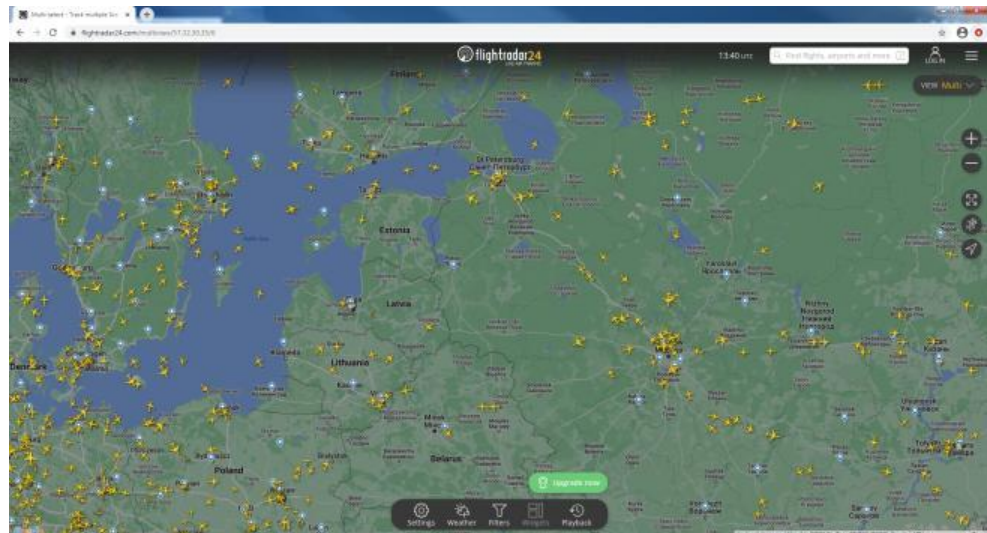


Рисунок 1.11- Вигляд головної сторінки FlightRadar24

FlightRadar включає в себе кілька основних функцій: по перше користувачі можуть переглядати інтерактивну мапу, на якій відображені всі активні повітряні судна. Мапу можна переміщати та масштабувати що забезпечує зручне користування нею. Користувач може вибрати певний літак просто клацнувши по ньому мишею. При виборі літака користувач отримує детальну інформацію про нього, таку як:

- тип, назву та номер судна;
- час польоту;
- траєкторію який судно вже пролетіло;
- номер рейсу;
- поточне місцезнаходження (широта, довгота та висота);
- швидкість судна;
- напрямок руху;
- відстань до пункту призначення та багато іншого.

Цей сервіс також надає можливість фільтрування, вибірки та пошуку повітряного судна по наступним критеріям: назва авіакомпанії, назва судна, тип судна, маршрути, висота, швидкість тощо. Це дозволяє користувачам швидко знаходити та відстежувати судна або польоти, які їх цікавлять.

FlightRadar24 також дозволяє подивитись історію польотів

конкретних літаків, або подивитись загальну історію усіх зафіксованих літаків упродовж тижня.

Щоб отримати додаткові функції сервісу, наприклад: відображення додаткових слоїв мапи(погода, опади, хмарність, швидкість вітру та інші), використання сервісу більш ніж години поспіль, додаткові фільтри для вибору літаків та польотів, відстеження літаку у трьох-вимірному просторі треба оформити підписку на даний сервіс.

Для перегляду зареєстрованого польоту достатньо вибрати літак на мапі, після чого інформація по літаку з'явиться зліва (рис. 1.12).

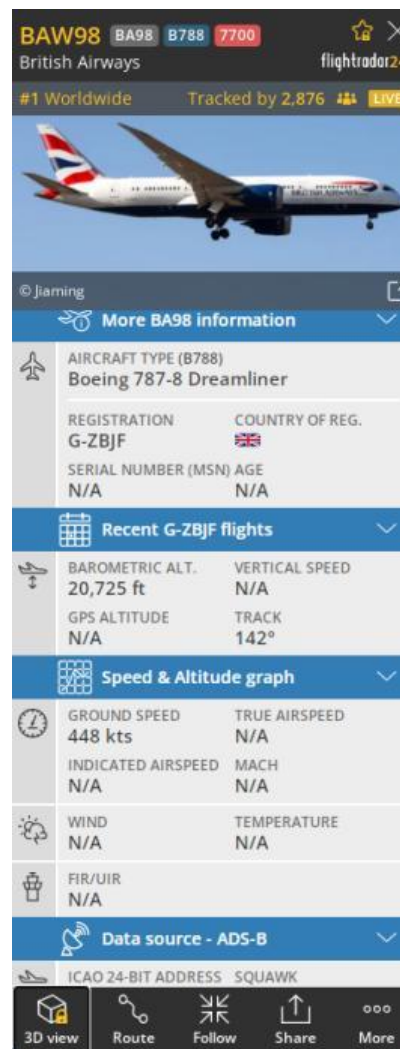


Рисунок 1.12- Інформація стосовно вибраного літака

Після чого натиснути кнопку «Route» для перегляду поточного

польоту літака, або натиснути «More» та вибрати пункт «Playback» для перегляду історії польотів літака(рис. 1.13-1.14). Вікно перегляду історії дозволяє подивитись записані польоти за останній тиждень(рис. 1.15). Після вибору польоту, що цікавить користувача він відобразиться на мапі, та користувач зможе подивитись де був літак у конкретний момент часу використовуючи повзунок(рис. 1.14).

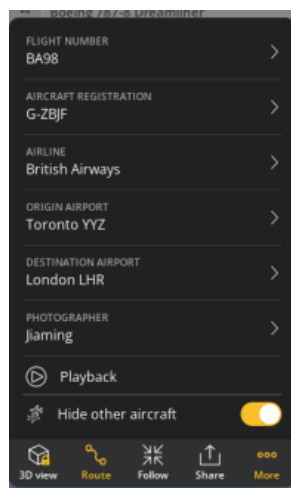


Рисунок 1.13- Меню кнопки «More»

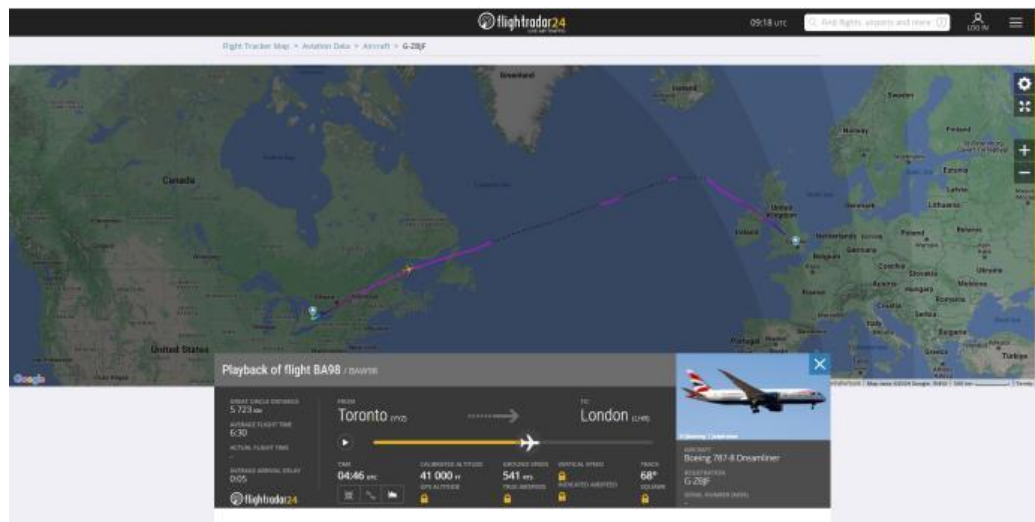


Рисунок 1.14- Вид вікна перегляду історії зареєстрованих польотів
вибраного літака

DATE	FROM	TO	FLIGHT	FLIGHT TIME	STD	ATD	STA	STATUS			
20 Mar 2024	Toronto (YYZ)	London (LHR)	BA98	—	23:05	23:14	10:10	Estimated 09:30			
20 Mar 2024	London (LHR)	Toronto (YYZ)	BA99	7:30	17:05	17:29	21:15	Landed 21:00			
19 Mar 2024	Houston (IAH)	London (LHR)	BA196	8:15	22:40	22:49	12:35	Landed 12:04			
19 Mar 2024	London (LHR)	Houston (IAH)	BA197	9:36	15:05	15:36	20:55	Landed 20:12			
18 Mar 2024	Houston (IAH)	London (LHR)	BA196	8:18	22:40	23:14	12:35	Landed 12:32			
18 Mar 2024	London (LHR)	Houston (IAH)	BA197	9:31	15:05	15:56	20:55	Landed 20:27			
17 Mar 2024	Nashville (BNA)	London (LHR)	BA222	7:09	19:45	20:01	08:45	Landed 08:10			
17 Mar 2024	London (LHR)	Nashville (BNA)	BA223	8:40	13:10	13:36	17:35	Landed 17:17			
15 Mar 2024	Chennai (MAA)	London (LHR)	BA36	10:40	07:50	08:38	13:50	Landed 13:47			
14 Mar 2024	London (LHR)	Chennai (MAA)	BA35	9:13	14:10	16:01	05:45	Landed 06:44			

More than 7 days of G-ZBJF history is available with an upgrade to a Silver (90 days), Gold (1 year), or Business (3 years) subscription.

7-day FREE trial | [Learn more](#)

Рисунок 1.15- Таблиця з інформацією про польоти вибраного літака за останній тиждень

Проаналізувавши застосунок FlightRadar24, мною було зазначено що інформація у цьому застосунку реалізована дуже зручно для користування. Користувач має потужний інструментарій для сортування по наступним критеріям:

- по часу ;
- по номеру літака;
- по імені літака;
- по типу літака;
- по номеру польоту;
- по назві авіакомпанії;
- по назві аеропортів.

Також FlightRadar24 дозволяє переглянути глобальну історію польотів на мапі або історії польотів конкретного літака з детальною інформацією про те, де знаходився літак у заданий користувачем час.

Отже, мною було виявлено наступні переваги FlightRadar24:

- зручний вибір літака;
- можливість відображення великої кількості літаків одночасно;
- можливість надання детальної інформації щодо вибраного користувачем літака;

- база даних яка зберігає повну історію польотів по всій землі за останній тиждень;
- можливість відтворити політ конкретного літака, з додатковою інформацією про позицію, висоту і стан літака у зазначений користувачем час польоту.

З недоліків можна визначити:

- основний функціонал заблоковано, і потребує підписки;
- основна мапа подається у двох-вимірному вигляді (чим далі від екватору, тим більше відбувається спотворення рельєфу та відстані), тому не дає змоги оцінити дійсну відстань та траєкторію яку подолав літак.

1.2.3 Порівняння існуючих продуктів

Проаналізувавши програму для відображення мап у трьох-вимірному просторі, та програму для відображення руху повітряних суден, мною було виявлено переваги й недоліки кожної з цих програм. Проведемо порівняння за певними критеріями, хід якого представимо у вигляді таблиці (1.1). Із критеріїв виділимо наступне:

- доступність – чи потребує продукт активації за рахунок коштів;
- інформативність – можливість побачити докладну інформацію про повітряні об'єкти;
- зручність інтерфейсу- наскільки зручним є керування програмою;
- можливість роботи офлайн -наскільки програма є дієздатною при відключенні від Інтернету;
- додатковий інструментарій – чи має програма додатковий інструментарій для зручної роботи з відображенням(наприклад лінійка);
- оптимізація – чи здатна програма відображати велику кількість повітряних тіл за один раз;
- підтримка формату реєстрації – чи має програма підтримку

читання файлів реєстрації повітряної обстановки.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерії порівняння	Google Earth	Fright Radar 24
Доступність	Є повністю платною	Є платні послуги
Інформативність	Користувач сам визначає інформацію для відображення	Середня
Зручність інтерфейсу	Дуже зручний	Середня
Додатковий інструментарій	Є	Немає
Підтримка формату реєстрації	Так, але з перетворенням у внутрішній формат	Не має можливості завантажувати файли
Оптимізація	Погана	Висока
Можливість роботи офлайн	Є	Немає

З результатів порівняння можна зробити висновок, що сервіси не надають можливості працювати офлайн, не мають нативної підтримки завантаження файлів реєстрації, та є платними (частково або повністю). Тим самим під час роботи, програма може втратити зв'язок з інтернетом, запросити оплату для подальшої роботи, або просто перестати працювати за відсутності мережі. Також жодна з програм не надає можливості нативного читання файлів реєстрації.

1.3 Постановка завдання

1.3.1 Функціональність системи

Розроблений програмний продукт має забезпечувати користувача наступним функціоналом:

- програмний засіб повинен забезпечувати завантаження файлів реєстрації радіолокаційних систем з розширенням «.reg»;

- програмний засіб повинен забезпечити відображення зареєстрованих координат повітряних об'єктів у 2D та 3D форматах;
- програмний засіб повинен бути доступною, та не обмежувати свій функціонал;
- програмний засіб повинен працювати локально(без підключення до інтернету);
- програмний засіб повинен мати можливість відображення результатів реєстрації на встановленому часовому інтервалу;
- програмний засіб повинен забезпечувати можливість налаштування зовнішнього вигляду окремих елементів відображення результатів реєстрації (колір, крок сітки і тд);
- програмний засіб повинен забезпечувати відображення інформації по вибраному об'єкту;
- програмний засіб повинен забезпечити вибірку потрібної користувачеві інформації;
- програмний засіб повинен зберігати налаштування користувача;
- програмний засіб повинен забезпечувати дружній інтерфейс для перегляду та налаштування вигляду повітряної обстановки за вимогами користувача.

1.3.2 Не функціональні вимоги до системи

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування графічної частини застосунку:

- операційна система (ОС)- Unix системи чи Windows;
- відеокарта з підтримкою OpenGL 3.0 або вище;
- процесор Intel Core i3 (або аналогічний);
- 1 ГБ ОЗП або більше;
- місце на жорсткому диску 70 мб.

1.4 Висновок за розділом 1

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.

Система потрібна для швидкого та зручного відображення реєстрації повітряної обстановки.

Система повинна мати можливість відкривати файли реєстрації повітряної обстановки без додаткової конвертації, з подальшим відображенням зчитаної інформації у двох та трьох вимірному просторі.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір мови програмування

При обранні мови програмування важливо враховувати ряд факторів. Найбільш значущими серед них є продуктивність, швидкодія, доступність потрібних бібліотек та фреймворків, а також масштабованість та підтримка специфічних функцій для візуалізації тривимірних даних.

C++ є загальноприйнятою мовою програмування, яка використовується для створення широкого спектру програм, від вбудованих систем до великих додатків та ігор. Вона є потужною і масштабованою мовою, яка володіє великою кількістю функцій і можливостей. Ось деякі ключові аспекти мови програмування C++:

- швидкодія: C++ відома своєю високою продуктивністю та швидкодією. Вона набагато ефективніша за інші мови програмування, оскільки має прямий доступ до апаратного забезпечення комп'ютера, що дозволяє оптимізувати виконання програм;
- об'єктно-орієнтований підхід: C++ підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розбивати програму на окремі об'єкти, що спрощує розробку, підтримку та розширення програм;
- низькорівневий доступ до пам'яті: у C++ є можливість прямого керування пам'яттю, що дозволяє розробникам оптимізувати використання ресурсів та працювати з апаратним забезпеченням на низькорівневому рівні;
- переносимість коду: код, написаний на C++, може бути легко перенесений між різними операційними системами та архітектурами;
- розширюваність: C++ дозволяє розробникам створювати бібліотеки та модулі, що можуть бути повторно використані в різних програмах, що сприяє розширюваності та підтримці коду;
- широке застосування: C++ використовується для розробки

високоєфективних програм, включаючи операційні системи, додатки для вбудованих систем, ігри, алгоритми шифрування, веб-розробку та багато іншого.

C++ залишається однією з найпотужніших та найвикористовуваніших мов програмування у світі, особливо в сферах, де вимагається висока продуктивність та ефективність програм.

Мова програмування Python є однією з найпопулярніших і широко використовуваних мов програмування у світі. Вона була створена Гвідо ван Россумом і вперше випущена у 1991 році. Python відзначається своєю простотою, читабельністю та ефективністю, що робить її ідеальним вибором для початківців у програмуванні, а також для професіоналів.

Ось деякі ключові характеристики Python:

- простота вивчення і читання коду: синтаксис Python дуже чистий і зрозумілий, що робить його ідеальним для новачків. Відсутність фігурних дужок або інших складних конструкцій дозволяє швидко розібратися в коді;
- мультипарадигмальність: Python підтримує різні стилі програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування. Це дозволяє програмістам використовувати той стиль, який найбільше підходить для їх завдань;
- багата екосистема бібліотек: Python має велику кількість бібліотек для різних цілей, що дозволяє вирішувати широкий спектр завдань, починаючи від наукових обчислень до веб-розробки і штучного інтелекту;
- широке застосування: Python застосовується у багатьох галузях, таких як веб-розробка (за допомогою фреймворків, таких як Django і Flask), наукові обчислення (за допомогою бібліотеки NumPy і SciPy), штучний інтелект та машинне навчання (за допомогою бібліотеки TensorFlow та PyTorch), аналіз даних (за допомогою бібліотеки pandas) і багато іншого;
- підтримка спільноти: Python має велику та активну спільноту користувачів, що сприяє швидкому розвитку мови, поширенню знань та допомозі одне одному;

- переносимість коду: код, написаний на Python, може бути легко перенесений між різними операційними системами, оскільки Python є крос-платформеним.

Python продовжує збільшувати свою популярність і застосування, завдяки своїм простоті, гнучкості та потужній функціональності. Вона є потужним інструментом як для початківців у програмуванні, так і для досвідчених розробників.

Мова програмування C# є однією з найпопулярніших мов програмування, розробленою корпорацією Microsoft. Вона є частиною платформи розробки Microsoft .NET і широко використовується для створення різноманітних додатків, від веб-сайтів до настільних програм і мобільних додатків. Ось деякі ключові характеристики мови C#:

- об'єктно-орієнтоване програмування: C# підтримує концепції об'єктно-орієнтованого програмування (ООП), що дозволяє створювати програми, які складаються з об'єктів, які взаємодіють один з одним;
- мультиплатформенність: за допомогою .NET Framework і .NET Core, розробники можуть створювати програми на C#, які працюють на різних платформах, включаючи Windows, macOS і Linux;
- безпека: C# має вбудовану систему безпеки, включаючи контроль типів, що допомагає уникнути багатьох типових помилок програмування, таких як використання неініціалізованих змінних;
- синтаксис: синтаксис C# подібний до синтаксису мови Java, що робить його зрозумілим для програмістів, які вже знайомі з цією мовою;
- підтримка Visual Studio: для розробки програм на C# використовується середовище розробки Visual Studio, яке надає розширені інструменти для підтримки розробки, налагодження та створення проектів;
- широке застосування: C# використовується для створення різноманітних програм, від веб-додатків і сервісів до ігор і мобільних додатків для платформи Windows, Android і iOS;
- асинхронність і паралелізм: C# надає зручні засоби для роботи з

асинхронним кодом і паралелізмом, що дозволяє створювати швидкодіючі та ефективні програми.

Узагальнюючи, мова програмування C# - це потужний інструмент для розробки різноманітних програм, який відрізняється зручним синтаксисом, широкими можливостями і підтримкою відомої розробницької платформи Microsoft.

Мова програмування Java є однією з найпопулярніших і широко використовуваних мов програмування у світі. Вона була створена компанією Sun Microsystems (зараз частина Oracle Corporation) і випущена у 1995 році. Ось деякі ключові характеристики мови Java:

- переносимість: Java є платформою-незалежною мовою програмування. Код, написаний на Java, може бути виконаний на будь-якій платформі, яка має відповідний віртуальний машину Java (Java Virtual Machine, JVM). Це робить Java ідеальним вибором для розробки крос-платформових програм;
- об'єктно-орієнтоване програмування: Java підтримує концепції об'єктно-орієнтованого програмування (ООП), що дозволяє створювати програми, які складаються з об'єктів, що взаємодіють один з одним;
- безпека: Java має вбудовані механізми безпеки, які допомагають уникнути таких загроз, як віруси та зловмисний код. Вона використовує систему безпеки на основі підпису цифрового коду та інші механізми для захисту даних та систем;
- широке використання: Java використовується в різних галузях, включаючи розробку мобільних додатків (за допомогою платформ Android), веб-розробку (за допомогою фреймворків, таких як Spring та Hibernate), корпоративний розвиток програмного забезпечення та багато іншого;
- велика екосистема: Java має велику кількість бібліотек та фреймворків, що полегшують розробку різноманітних додатків. Вона також має активну спільноту розробників, яка постійно розширює та покращує її функціональність.

Загалом, Java є потужним інструментом для розробки програм, який відрізняється своєю переносимістю, безпекою, масштабованістю та широким спектром застосувань.

Отже проаналізувавши мови програмування, можна зробити таблицю важливих факторів (табл 2.1), які будуть впливати на ефективність розробки та зручність написання коду, а саме таких як:

- наявність потрібних бібліотек суттєво спростить написання коду;
- простота написання – різні мови програмування мають різний підхід для реалізації одного й того-ж функціоналу;
- зручна документація допоможе при написанні коду;
- швидкодія відповідає скільки часу потрібно для виконання однієї операції (менше часу- більша швидкодія);
- крос-платформність означає, що для того, щоб випустити програму під іншу операційну систему, код не потребує рефакторингу.

Таблиця 2.1- Порівняння мов програмування

Критерії порівняння	C++	Python	C#	Java
1	2	3	4	5
Швидкодія	Висока	Низька	Висока	Середня
Наявність потрібних бібліотек	Так	Часткова	Так	Так
Простота написання коду	Частково	Дуже просто	Частково	Складно
Зручна документація	На рівні IDE	Частково	Зручна	Не зручно
Крос-платформність	Так	Так	Так	Так

Згідно таблиці 2.1 можна зробити висновок що кращою мовою програмування для реалізації застосунку є C++.

2.2 Вибір середи розробки

Для початку роботи треба вирішити, у якій середі розробки буде зручніше всього досягти поставленої мети. Існує багато IDE для написання програмних застосунків. Кожна з них має свої недоліки та переваги.

Qt Creator - це інтегроване середовище розробки (IDE), призначене для роботи з фреймворком Qt. Створений компанією Qt, цей інструмент спеціалізується на розробці програмного забезпечення з використанням мов програмування C++ та Python, а також дозволяє використовувати QML для розробки додатків з інтерфейсом користувача. Вигляд програми наведено на рисунку 2.1.

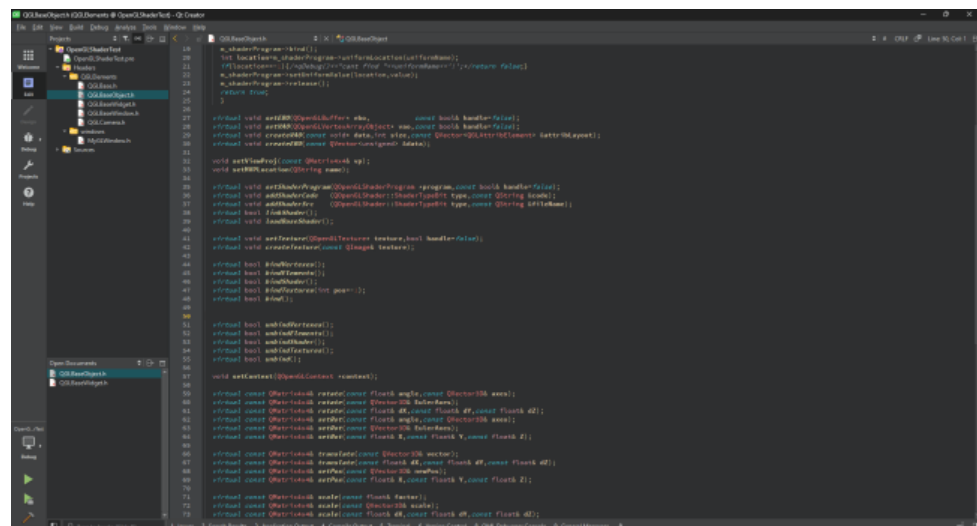


Рисунок 2.1- Вигляд QtCreator

QML можна використовувати як у стандартному редакторі Qt Creator, але краще використовувати додаткову програму Qt Design Studio(рис 2.2).

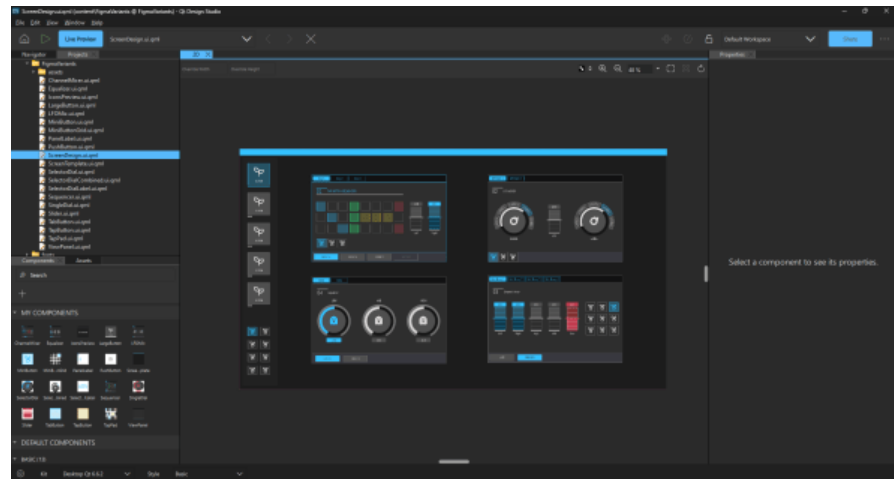


Рисунок 2.2- Вигляд Qt Design Studio

Особливостями Qt Creator є потужний редактор коду з функціями автодоповнення, підказками до синтаксису та рефакторингу, вбудований довідник. Він також має інтеграцію з системами керування версіями, такими як Git, що спрощує роботу в команді над проектами.

Qt Creator надає інструменти для розробки графічних інтерфейсів користувача, включаючи візуальний дизайнер, який дозволяє створювати і налаштовувати віджети та QML-елементи. Крім того, він має вбудовані засоби для налагодження коду та відлагоджування додатків, що допомагають виявляти та виправляти помилки.

Завдяки своїй простоті використання та потужному функціоналу, Qt Creator є популярним інструментом для розробки програмного забезпечення на мовах C++ і Python з використанням фреймворку Qt. Він підходить як для початківців, так і для досвідчених розробників, які працюють над проектами будь-якої складності.

У підсумку, Qt Creator є потужним та дружнім інструментом для розробки програмного забезпечення на мові C++ з використанням фреймворку Qt. Він надає розробникам зручний інтерфейс, багатий набір функцій та інструментів, які допомагають збільшити продуктивність та якість розробки. Безумовно, Qt Creator варто розглядати як вибір для тих, хто працює з Qt або розглядає можливість використання цього фреймворку у

своїх проектах.

Visual Studio - це інтегроване середовище розробки (IDE), розроблене компанією Microsoft для створення різноманітних програмних продуктів. Воно підтримує розробку на різних мовах програмування, таких як C#, C++, JavaScript та інші, і призначене для роботи на платформі Windows та інших платформах. Вигляд програми наведено на рисунку 2.3.

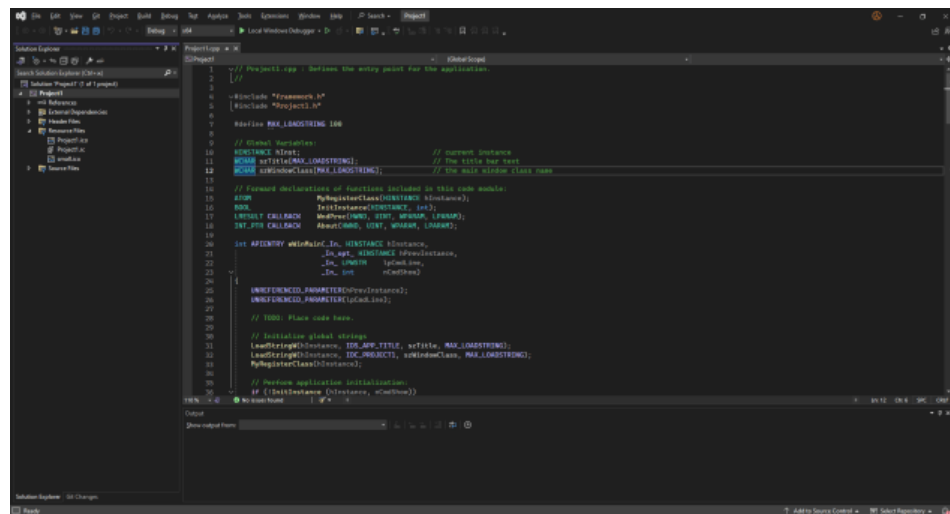


Рисунок 2.3- Вигляд Visual Studio

Основні особливості Visual Studio включають в себе вбудовані редактори коду та форм, розширені можливості для роботи з мовами програмування, велику кількість аддонів та плагінів, довідкову систему та контекстно-залежну допомогу, а також вбудовану консоль та інтеграцію з різноманітними сервісами, такими як Azure DevOps.

Visual Studio підтримує різні типи розробки, включаючи розробку веб-додатків з використанням фреймворків, таких як ASP.NET та ASP.NET Core, розробку мобільних додатків для платформ Android, iOS та Xamarin, а також розробку настільних програм та служб. Він також надає інструменти для налагодження коду, аналізу продуктивності та профілювання, що дозволяє розробникам покращувати якість своїх програм.

Узагальнюючи, Visual Studio є високопродуктивним та розширеним

середовищем розробки, яке дозволяє розробникам створювати якісне програмне забезпечення для різних платформ з використанням різних мов програмування. Його багатифункціональність та потужність роблять його популярним вибором серед розробників по всьому світу.

VSCode, або Visual Studio Code, є популярним, простим, інтегрованим середовищем розробки (IDE), що розроблено компанією Microsoft. Це зручний інструмент для роботи з різними мовами програмування, такими як JavaScript, TypeScript, Python, і багатьма іншими. Він відомий своєю легкістю використання та гнучкістю налаштувань, що робить його популярним серед розробників різного рівня досвіду. Вигляд програми наведено на рисунку 2.4.

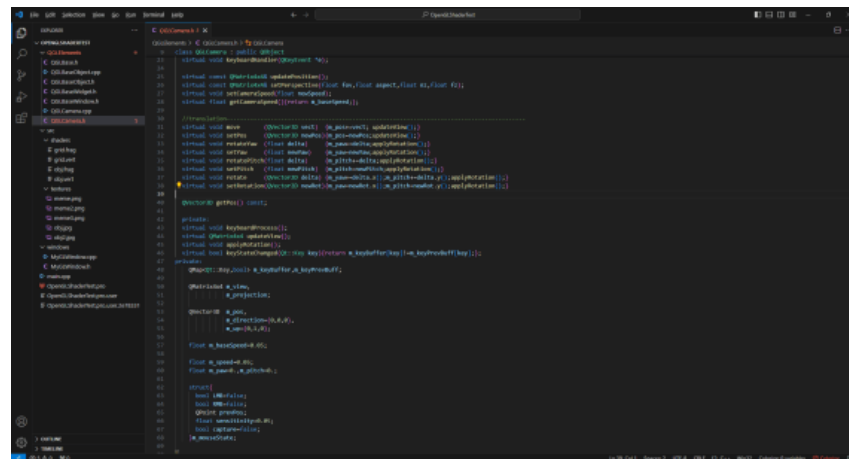


Рисунок 2.4- Вигляд Visual Code

Основні особливості VSCode включають в себе потужний редактор коду з різноманітними функціями, такими як автодоповнення, відступи, кольорове виділення синтаксису і багато інших. Він також має вбудовану систему розширень, що дозволяє розширити його функціонал до найбільш специфічних потреб розробника.

Крім того, VSCode має інтеграцію з системами керування версіями, такими як Git, що спрощує роботу в команді над проектами. Він підтримує роботу з різними системами збірки і може бути легко налаштований для роботи з різними мовами програмування.

VSCode також відомий своєю активною спільнотою користувачів і постійними оновленнями та покращеннями з боку розробників. Це робить його відмінним вибором для розробників, які шукають легке, гнучке та потужне середовище розробки для своїх проектів.

Заключно можна сказати, що Visual Studio Code є вражаючим інтегрованим середовищем розробки, яке забезпечує зручний та потужний інструментарій для програмістів будь-якого рівня досвіду. Його легкість використання, гнучкість налаштувань та розширюваність роблять його ідеальним вибором для широкого спектру завдань, починаючи від розробки веб-додатків до роботи з великими проектами на різних мовах програмування. З активною спільнотою користувачів і постійними оновленнями, Visual Studio Code залишається однією з найпопулярніших та найбільш цінних інструментальних платформ для розробників у сучасній індустрії програмування.

CLion - інтегроване середовище розробки, призначене для програмування на мовах C і C++. Воно надає зручний інтерфейс для роботи з кодом, включаючи можливості автодоповнення, рефакторингу та налагодження. CLion також підтримує інтеграцію з різними системами керування версіями та засобами збірки, що робить його універсальним інструментом для розробки програмного забезпечення на мовах C і C++.

CLion є популярним вибором серед розробників, що працюють з мовами програмування C і C++. Він забезпечує широкі можливості для комфортної та продуктивної роботи над проектами будь-якої складності. Вигляд програми наведено на рисунку 2.5.

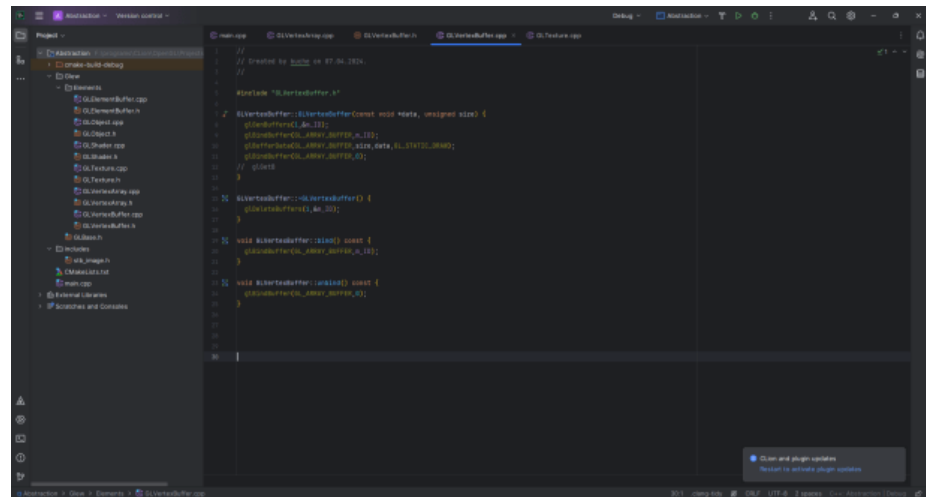


Рисунок 2.5- Вигляд CLion

Однією з ключових особливостей CLion є його потужний редактор коду, який надає функції автодоповнення, підказки та рефакторингу, що допомагають швидко і ефективно писати код. Вбудована система аналізу коду допомагає виявляти помилки та непотрібний код, забезпечуючи високу якість розробки.

Крім того, CLion інтегрується з популярними системами керування версіями, такими як Git, що дозволяє зручно працювати в команді над проектами. Він також підтримує різні системи збірки, включаючи CMake, що робить процес розробки більш гнучким та налаштовуваним.

CLion має вбудовані інструменти для налагодження коду, які дозволяють розробникам ефективно відлагоджувати свої програми та виявляти та виправляти помилки. Крім того, CLion активно оновлюється та підтримується розробниками, що гарантує актуальність та надійність інструменту для користувачів.

Узагальнюючи, CLion є потужним та зручним середовищем розробки для мов програмування C і C++, яке допомагає розробникам створювати якісне програмне забезпечення швидко та ефективно.

Отже проаналізувавши найпопулярніші IDE, можна зробити таблицю важливих факторів (табл. 2.2), які на мою думку є важливими для ефективного та якісного написання коду, а саме таких як:

- графічний інтерфейс користувача (GUI) – комплексна оцінка зрозумілості та інтуїтивності GUI середовища розробки для користувача;
- продуктивність – навантаження на ОС під час розробки (% навантаження на ЦП та обсяг зайнятої ОЗП);
- служба підтримки – комплексна оцінка для визначення рівня технічної підтримки середовища, його розробників та спільноти, що використовує середовище;
- підтримка української мови в GUI;
- підсвічування синтаксису;
- рефакторинг;
- автодоповнення коду;
- пошук по проєкту;
- налагодження;
- тестування програм;
- мови, які підтримуються – визначає які мови підтримує середовище розробки, що в майбутньому дозволить розширювати функціонал при оновленні застосунку із використанням стороннього коду;
- доступність – чи являється програмний засіб безкоштовним, має додаткові платні послуги(безкоштовна з обмеженнями) чи є повністю платним.

Таблиця 2.2- Порівняння IDE

	Qt Creator	Visual Studio	Visual Code	CLion
1	2	3	4	5
GUI	++	++	++	++
Продуктивність	3-5% 300 МБ	4-6% 400 МБ	0-3% 350 МБ	2-4% 2 ГБ
Служба підтримки	++	+	+	+

Продовження таблиці 2.2

1	2	3	4	5
Українська мова	++	+	+	-
Підсвічування синтаксису	++	++	++	++
Рефакторинг	+	+	++	++
Автодоповнення	+	++	++	++
Пошук по проєкту	+	++	+	+
Налагодження	++	+	+	+
Тестування програм	++	++	+	++
Мови, які підтримуються	C++, QML, Python	C++, C#	Будь-яка через доповнення	C++
Доступність	Безкоштов на з обмеження ми	Безкоштов на з обмеження ми	Безкоштовн а	Повністю платна

2.3 Вибір інтерфейсу для роботи з графікою

Для реалізації трьох-вимірного відображення існує потрібно реалізувати зв'язок між процесором та відеокартою. Є два варіанти як можна реалізувати цей зв'язок. Перший- написати реалізацію самому, що потребує глибоких знань у сфері низько-рівневого програмування, розуміння ідеї передачі команд та інформації до GPU. Це потребує багато часу для вивчення та написання свого «прошарку». Другий варіант- використання вже готових рішень, що значно спростить написання застосунку. Із готових рішень можна виділити OpenGL, DirectX та Vulkan.

DirectX- це інтерфейс (API) розроблений Microsoft для створення мультимедійних та графічних (двох та трьох вимірних) додатків на платформі Windows. DirectX включає набір інструментів для роботи зі

звуком, відео, графікою, обробкою пристроїв введення а також мережевими функціями. Він широко використовується в ігровій індустрії та мультимедійних застосунках.

До переваг DirectX належать:

- інтеграція з Windows: оскільки DirectX був створений спеціально для роботи з Windows, він гарантує більшу продуктивність, кращу сумісність та оптимізацію на Windows відносно інших API;
- широкий функціонал: як було зазначено вище, DirectX надає можливість не тільки працювати GPU, а також з пристроями введення, звуком та ін. Завдяки цьому API є все-об'ємною та оптимізованою;
- підтримка нових технологій: Microsoft регулярно оновлюють DirectX, тим самим забезпечуючи розробників доступом до новітніх технологій.

Недоліками DirectX є:

- обмеженість платформи: так як DirectX розроблявся тільки для Windows, він не працює на інших системах;
- залежність від «заліза»: не всі відеокарти підтримують останню версію DirectX, тому для коректної роботи програм повинні бути задані мінімальні системні потреби.;
- залежність від виробника: оновлення DirectX випускає тільки компанія Microsoft та звичайні користувачі не мають доступ до коду. Тим самим звичайний користувач не може змінити API під свої потреби;
- менша гнучкість у порівнянні з іншими API.

OpenGL (Open Graphics Library)- це інтерфейс (API) для програмування двох-вимірної та трьох-вимірної графіки. Він надає низькорівневий доступ до апаратних можливостей графічної карти. OpenGL широко використовується в індустрії відеоігор, візуалізації даних, комп'ютерної графіки, наукових досліджень і багатьох інших сферах. Його можна використовувати на різних платформах, включаючи Windows, macOS та Linux.

До переваг OpenGL належать:

- переносимість: OpenGL підтримується на різних платформах, що дозволяє писати крос-платформні застосунки;
- висока швидкість: оскільки OpenGL використовує апаратні можливості GPU, це дозволяє досягти високої продуктивності;
- розширюваність: для OpenGL існує багато сторонніх бібліотек, що спрощують процес написання програми, створюють більш-високий рівень абстракції для зручної роботи, додають новий функціонал. Наприклад (GLUT, GLEW, GLFW та інші.);
- відкритість: як стандартний інтерфейс, OpenGL має відкритий код який кожен може подивитись і модифікувати за потребою.

Недоліками OpenGL є:

- обмеження функціональності: не дивлячись на додаткові бібліотеки, OpenGL не має деяких функцій, які присутні в інших API;
- низький рівень абстракції: так як OpenGL використовує апаратні можливості GPU, ця API є низькорівневою. Тому для зручної реалізації додатків може знадобитись реалізовувати свої методи та класи щоб узагальнити та спростити написання коду;
- підтримка деяких платформ: через те що OpenGL є крос-платформною бібліотекою та узагальнює свій функціонал щоб забезпечити стабільну роботу на всіх підтримувальних платформах, він не може реалізувати повний потенціал відносно однієї системи.

Vulkan- це інтерфейс (API) розроблений для роботи з двох та трьох вимірною графікою та обчисленнями. Він був розроблений як альтернатива існуючому API OpenGL для забезпечення кращої продуктивності і роботи. Vulkan надає більший контроль над апаратними ресурсами, дозволяє оптимізувати роботу з графікою на різних платформах: настільних комп'ютерах, ноутбуках, телефонах та інші.

Також Vulkan надає можливість крос-платформного програмування, оскільки він підтримується на Windows, macOS, Linux.

Оскільки Vulkan це покращена версія OpenGL, він має такі-ж самі переваги й недоліки що й OpenGL, але через низький рівень абстракції Vulkan становиться більш тяжким для опанування API.

Проаналізувавши інтерфейси побудуємо таблицю 2.3 за наступними факторами:

- простота використання;
- функціональність;
- швидкодія;
- спільнота;
- документація;
- кросплатформеність.

Таблиця 2.3- Порівняння інтерфейсів роботи з графікою

Критерий	OpenGL	Vulkan	DirectX
Простота використання	++	-	+
Функціональність	+	++	+
Швидкодія	+	++	+
Спільнота	+	+	++
Документація	+	-	++
Кросплатформеність	+	-	-

Отже судячи з таблиці можна зробити наступні висновки

а) простота використання:

1) OpenGL: вимагає певного рівня досвіду та може бути менш інтуїтивним для початківців;

2) Vulkan: складний API, який потребує глибокого розуміння для ефективного використання;

3) DirectX: часто вважається більш простим у використанні, особливо для розробників Windows;

б) функціональність:

1) OpenGL: забезпечує достатньо широкий набір функцій,

хоча не настільки потужний, як Vulkan;

2) Vulkan: дуже потужний та гнучкий API з прямим доступом до апаратних ресурсів;

3) DirectX: пропонує широкий набір функцій, особливо для роботи під Windows;

в) швидкодія:

1) OpenGL: не так ефективний у плані продуктивності, як Vulkan або DirectX;

2) Vulkan: забезпечує високу продуктивність та ефективне використання апаратних ресурсів;

3) DirectX: добра продуктивність на платформі Windows;

г) спільнота:

1) OpenGL: має активну та обширну спільноту розробників;

2) Vulkan: ростуча спільнота, але поки не така велика, як у OpenGL;

3) DirectX: обширна спільнота розробників, особливо в галузі ігрової індустрії;

д) документація:

1) OpenGL: добре задокументований, але деякі аспекти можуть бути складними для розуміння;

2) Vulkan: документація може бути складною через складність самого API;

3) DirectX: зазвичай добре задокументований, особливо для розробників Windows;

е) кроссплатформеність:

1) OpenGL: широко підтримується на різних платформах;

2) Vulkan: також підтримується на різних платформах, але потребує більше зусиль для забезпечення кроссплатформеності;

3) DirectX: обмежений в основному платформою Windows.

2.4 Висновок за розділом 2

Як основну мову програмування було вибрано C++, оскільки вважається що C++ є швидкою мовою програмування відносно Python. Також однією з головних причин вибору цієї мови послугувало її парадигми, а саме те що вона включає в себе не тільки ООП, що надає переваги над Java.

Як середу розробки було вибрано Qt Creator, оскільки для створення віконних програм, Qt Creator є набагато зручнішим ніж Visual Studio та CLion. Хоча Visual Studio має в арсеналі розробку GUI за допомогою графічного редактору, але це поширюється тільки на програми написані на C#, та у елементах які доступні у редакторі Visual Studio немає наявних елементів для відображення графіки, тим самим перевага переходить до QtCreator. Також завдяки вбудованому довіднику, Qt Creator надає можливість підвищити продуктивність роботи, не відволікаючись на браузер та пошук потрібної інформації. Відносно VSCode, Qt Creator є більш зручною та простою IDE для налаштування. Без усяких питань VSCode набагато гнучкіший за Qt Creator оскільки має велику спільноту користувачів, яка у свою чергу генерує величезну кількість аддонів для зручної роботи у будь-якій сфері програмування, але через велику кількість тих самих аддонів, дуже складно знайти потрібний функціонал. Також Qt Creator має безкоштовний план для open-source проектів.

Для побудови графічного зображення мною було вибрано OpenGL. Відносно DirectX, OpenGL є кросплатформним, також OpenGL має велику кількість додаткових бібліотек, які розширюють, доповнюють функціонал, та спрощують роботу з OpenGL. Оскільки Vulkan є відносно новою та доволі тяжкою для використання платформою, може виникнути проблеми з вирішенням деяких задач, в той же час OpenGL має велику спільноту, відео уроків та інструкцій, де можливо знайти рішення на будь-яке питання. Тому я вирішив що буду використовувати OpenGL.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Реалізація інтерфейсу користувача

Для розробки інтерфейсу програми були використані стандартні функції Qt, Qt Designer та створено власні віджети.

Для реалізації підбирання кольору, за основу було використано стандартну кнопку QPushButton.

Методи та поля класу наведені на таблиці 3.1

Таблиця 3.1- Методи і поля Color Picker

Поля та методи класу	Опис
1	2
public:	
ColorButton(QWidget *parent)	Конструктор
QColor getColor()	Поточний колір кнопки
public slots:	
setColor(QColor color)	Встановлення кольору кнопки
private:	
QColor curColor	Поточний колір кнопки

Щоб змінити колір кнопки, було використано стандартний метод «setStyleSheet» з параметром «background-color». Вигляд кнопки наведено на рисунку 3.1.



Рисунок 3.1- Вигляд кнопки з різними кольорами.

Для реалізації діапазонного слайдери, було розроблено новий клас RangeSlider, та додано його у клас віджет TimeSlider що додатково реалізує відображення часу у зручному для читання форматі. Методи і поля класу

RangeSlider наведені у таблиці 3.2

Таблиця 3.2- Методи і поля класу RangeSlider

Поля та методи класу	Опис
1	2
public:	
RangeSlider(QWidget*)	Стандартний конструктор
RangeSlider(Qt::Orientation, Options, QWidget*)	Конструктор з додатковими параметрами відображення
QSize minimumSizeHint() const	Мінімальний розмір
int GetMinimun() const	Мінімальне значення
SetMinimum(int)	Встановлення мінімального значення
int GetMaximun() const	Максимальне значення
void SetMaximum(int)	Встановлення максимального значення
int GetLowerValue() const	Значення лівого або нижнього повзунка
SetLowerValue(int)	Встановлення значення нижнього повзунка
int GetUpperValue() const	Значення правого або верхнього повзунка
SetUpperValue(int)	Встановлення значення верхнього повзунка
SetRange(int,int)	Встановити проміжок нижнє-верхнє значення
protected:	
paintEvent(QPaintEvent*) override	Відображення
mousePressEvent(QMouseEvent*) override	Обробка подій миші
mouseMoveEvent(QMouseEvent*) override	
mouseReleaseEvent(QMouseEvent*) override	
mouseDoubleClickEvent (QMouseEvent*)	
changeEvent(QEvent*)	Обробка зміни стану
QRectF firstHandleRect() const	Область першого повзунка
QRectF secondHandleRect() const	Область другого повзунка
QRectF handleRect(int)	Область повзунка по номеру
signals:	
lowerValueChanged(int)	Змінилось нижнє значення
upperValueChanged(int)	Змінилось верхнє значення

Продовження таблиці 3.2

1	2
rangeChanged(int,int)	Змінився проміжуток
<i>public slots:</i>	
setLowerValue(int)	Встановлення значення нижнього повзунка
setUpperValue(int)	Встановлення значення верхнього повзунка
setMinimum(int)	Встановлення мінімального значення
setMaximum(int)	Встановлення максимального значення
<i>private:</i>	
int validLength()	Максимальне значення положення повзунка в пікселях
int mMinimum	Мінімальне значення
int mMaximum	Максимальне значення
int mLowerValue	Нижнє значення
int mUpperValue	Верхнє значення
int mBuffUpperValue	Буфер верхнього значення
bool mFirstHandlePressed	Флаг активності нижнього повзунка
bool mSecondHandlePressed	Флаг активності верхнього повзунка
int mInterval	Інтервал між значеннями
Int mDelta	Дельта відносно попереднього положення повзунка
QColor mBackgroundColorEnabled	Колір зони між повзунками при активному слайдері
QColor mBackgroundColorDisabled	Колір зони між повзунками при не активному слайдері
QColor mBackgroundColor	Задній колір слайдера
Qt::Orientation mOrientation	Орієнтація слайдера
Options type	Тип слайдера

Обернувши цей клас у віджет, для керування часом, отримаємо наступний результат (рис 3.2).

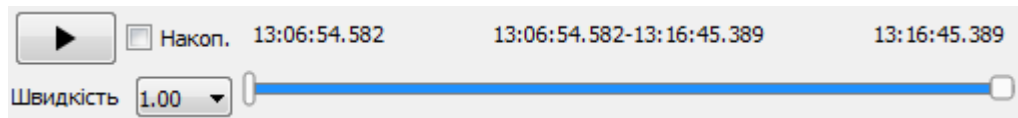


Рисунок 3.2- Вигляд віджету керування часом

Вікно програми поділене на декілька динамічних секторів, що дозволяє користувачеві швидко та легко керувати програмою, а також приховувати деякі сектори за бажанням. Загальний вигляд програми наведено на рисунку 3.3. Із секторів можна визначити:

- а) таблиця траєкторій;
- б) вікно відображення;
- в) керування часом;
- г) налаштування.

Також у програмі є довідка, яка пояснює як керувати відображенням(рис 3.3).

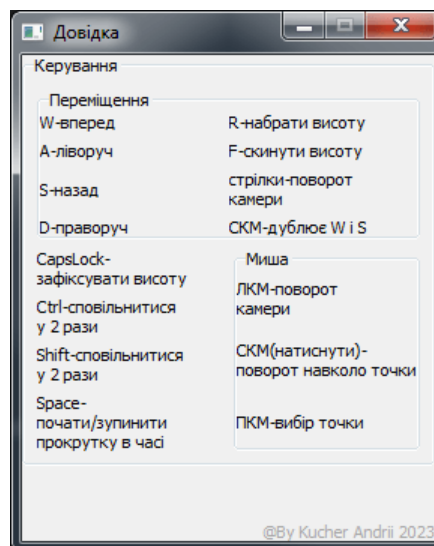


Рисунок 3.3- Вигляд довідки

Варіанти вигляду програми наведені на рисунках 3.4-3.6.

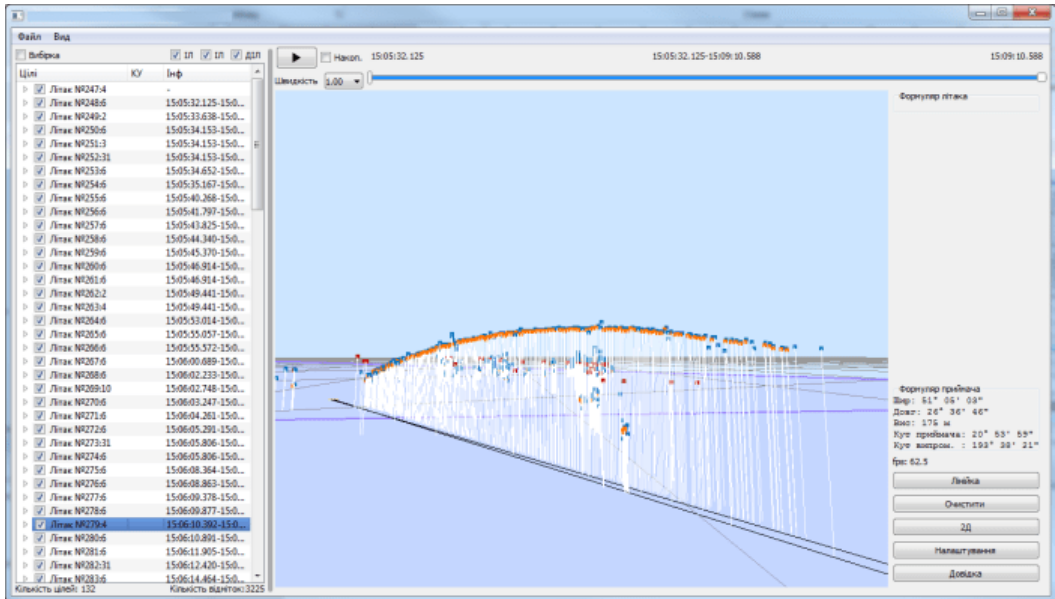


Рисунок 3.4- Загальний вигляд програми

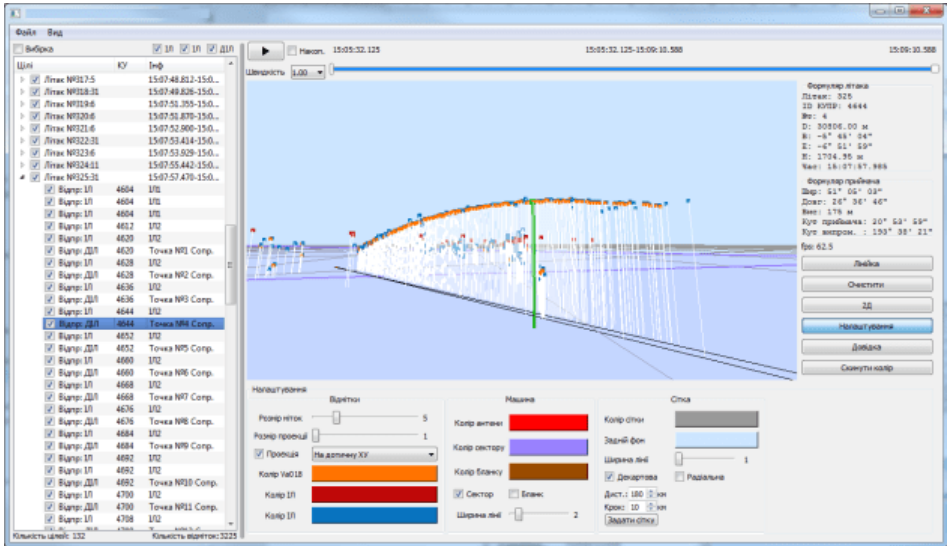


Рисунок 3.5- Один із варіантів компоновки секцій
(Таблиця об'єктів, загальні налаштування, вікно відображення)

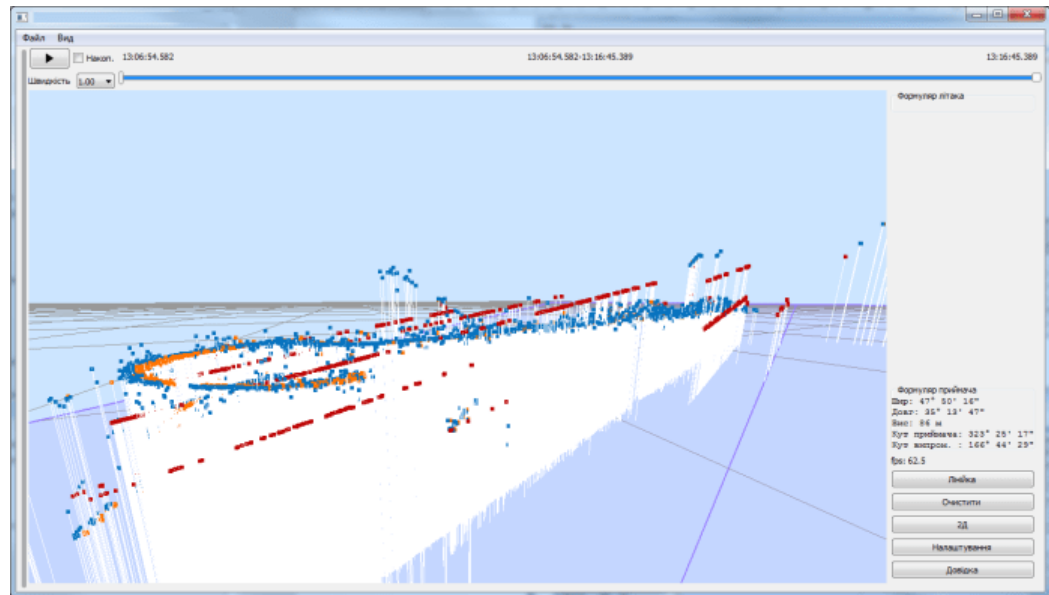


Рисунок 3.6- Один із варіантів компоновки секцій
(Вікно відображення)

3.2 Структура файлів реєстрації

У записаних файлах реєстрації вхідної інформації радіолокаційних систем передається наступна інформація:

- часова відмітка. Містить у собі поточно-zareєстрований час;
- службова інформація про стан локатора (ІСЛ). Містить у собі координати локатора, кут повороту антени, та іншу службову інформацію;
- інформація про об'єкт (ІО, ДІО). Містить у собі інформацію про повітряний об'єкт. ІО містить у собі координати та номер об'єкта, ДІО містить у собі номер об'єкта, швидкість та іншу службову інформацію.

Отже кодограми йдуть по трьом основним каналам: ІСЛ, ІО, ДІО.

Для того щоб правильно зчитати дані з файлу, потрібно розуміти який формат використовувався для запису інформації до файлу. Моя програма, зчитує бінарні файли у яких записані кодограми.

Кодограми –повідомлення яке має певну визначену структуру, що складається з заголовку тіла та кінця.

Заголовком кодограми вважається службова інформація, яка включає

в себе:

- флаг початку кодограми. У нашому випадку це два або чотири байти 0xFF;
- довжина кодограми в байтах. Довжина повинна враховувати заголовок також(2 байти);
- тип кодограми. Інформує яка інформація знаходиться у тілі кодограми(1 байт).

Отже заголовок займає 5-6 байт (залежно від флагу початку).

Тіло кодограми зберігає користувацьку інформацію. Завдяки типу записаному у заголовку, ми маємо можливість визначити яка саме структура даних знаходиться у тілі. Тіло має не визначену довжину і може змінюватись за потреби.

Кінцем кодограми вважається службова інформація, яка відповідає за перевірку цілісності всієї кодограми. У кінці зазвичай зберігається контрольна сума всієї кодограми, завдяки чому можна переконатись, чи прийшла кодограма цілою. Зазвичай для перевірки цілісності кодограми вистачає одного байту.

Вигляд однієї кодограми приведено у таблиці 3.3:

Таблиця 3.3- Структура кодограми

Назва розділу	Заголовок			Тіло	Кінець
Назва змінної	Флаг початку	Довжина	Тип	...	КС
Розмір	2-4 байти	2 байти	1 байт	динамічна	1 байт

3.3 Алгоритм читання збереженої реєстрації з файлу

Оскільки у файлі лежать данні які можуть мати змінну довжину, розробимо базову структуру яка буде зберігати інформацію стосовно однієї кодограми(рис 3.3.1) для реалізації масиву кодограм, та блок-схему на якій

покажемо процес читання кодограм з файлу(рис 3.7)

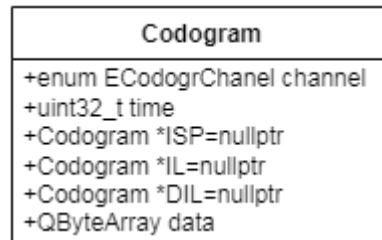


Рисунок 3.7- Базова структура кодограми

Треба звернути увагу на покажчики ІСЛ, ІЛ, ДІЛ. Вони потрібні для подальшого зв'язку між кодограмами.

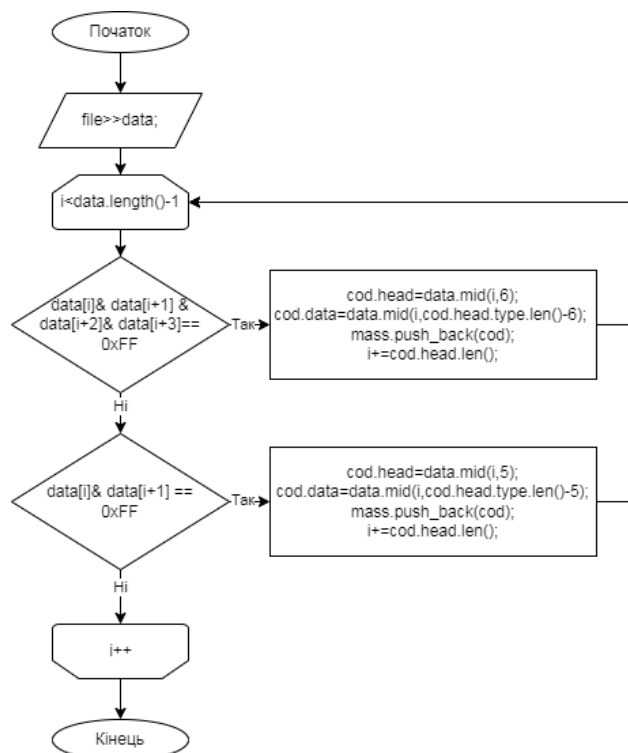


Рисунок 3.8- Алгоритм читання файлу реєстрації

3.4 Організація та структура зберігання зчитаної інформації

В результаті попереднього кроку, отримаємо масив кодограм, які ніяк не організовані.

Для початку треба зрозуміти яким чином та по яким критеріям повинні бути організувати дані, щоб полегшити роботу з ними у майбутньому. Мною було виділено наступні критерії, по яким будуть сортуватись дані:

- групування кодограм по типу каналу;
- групування кодограм по часу;
- групування точок у траєкторії об'єкта, по номеру об'єкту.

Для економії обсягу пам'яті та підвищення продуктивності організації інформації, мною було вирішено зберігати вказівник на відповідну кодограму у основному масиві.

Також для реалізації «розгортання» польотів об'єкту у часі, потрібен масив який буде зберігати в собі кодограми відсортовані по часу. Цього можна досягти завдяки хеш-таблиці. В QtCreator є стандартний шаблонний клас QMap, що дозволяє зберігати пару значень(ключ, значення). Саме його я і буду використовувати.

Так як кожна кодограма ДІЛ зв'язана з кодограмою ІЛ через номер об'єкту, це надає можливість групувати кодограми ІЛ та ДІЛ у масиви, тим самим створювати траєкторії польоту об'єктів. Для подальшого сортування, мною було організовано структуру Point3D(рис 3.9) та клас Trace (таб. 3.4).

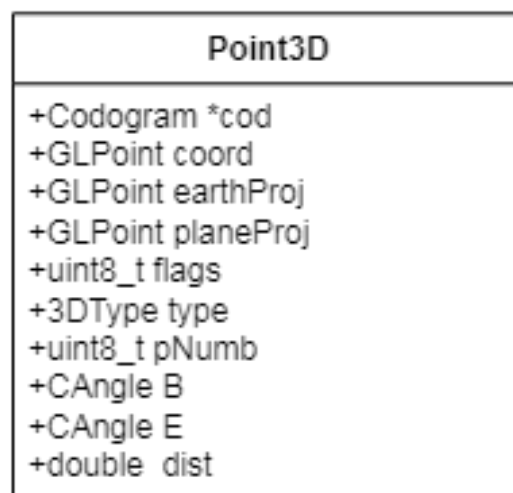


Рисунок 3.9- Структура Point3D

Таблиця 3.4- Методи і поля Trace

Поля та методи класу	Опис
<i>public:</i>	
PointPair startEnd	Пара точок (виліт, приземлення)
GLColor lineColor	Колір проєкцій точок
int startTime	Час початку стеження за об'єктом
int endTime	Час кінця стеження за об'єктом
int pointCount	Кількість точок
int startTimeIndex	Індекс першої видимої точки у заданому часовому проміжку
int endTimeIndex	Індекс останньої видимої точки у заданому часовому проміжку
QVector<Point3D> points	Масив точок
QVector<double> coords	Масив координат для передачі у OpenGL
QVector<double> colors	Масив значень кольору для передачі у OpenGL
uint8_t flags	Флаги відображення
calculateArray(int,int)	Перерахунок масиву координат та кольорів

Оскільки з OpenGL простіше працювати у локальній системі координат, а інформація про позицію об'єкту зберігається у глобальній системі(тобто широта, довгота та висота), потрібна точка на поверхні землі, яка буде відповідати початку координат у локальній системі OpenGL. Завдяки каналу ІСЛ стає можливим відображати координати точок відносно самого локатора. Тим самим знаючи глобальні координати локатора та глобальні координати точки, можна перевести координати точки з глобальної системи координат у локальну систему локатора.

3.5 Сімейство класів роботи з OpenGL

Для зручної роботи з OpenGL було створено базові класи для математичних обчислень, а також усі елементи що потрібно відобразити на

екрані були поділені на «слої», що дозволяє динамічно створювати нові елементи або змінювати старі. До базових класів математичних обчислень входять: GLPoint, GLColor та GLVector.

GLElem- клас інтерфейс, розроблений для визначення наступних методів які будуть реалізовані в класах нащадках. Дані та методи наведені у таблиці 3.5.

Таблиця 3.5- Опис полів та методів класу GLElem.

Поля та методи класу	Опис
<i>public:</i>	
GLElem(QObject*)	Конструктор
~GLElem()	Деструктор
virtual setSettings(QVariant)	Віртуальний метод, який реалізує задання налаштувань. Завдяки використанню QVariant, користувач має можливість реалізувати будь-яку структуру яка буде зберігати потрібні користувачеві данні.
virtual draw()=0	Чистий віртуальний метод, реалізуючий рендерінг слою. реалізація відрізняється в залежності від нащадку.
virtual setDependencies()	Необов'язковий віртуальний метод, відповідальний за налаштування OpenGL для коректного рендеру слою.

Класи слої це нащадки від базового класу GLElem, які реалізують віртуальну функцію draw(), що дозволяє використовувати масив вказівників на базовий клас, для зберігання всіх слоїв в одному масиві. Вигляд ієрархії наведено на рисунку 3.10. Кожен з нащадків має власну структуру *Settings яка описує налаштування конкретного слою, а також додаткові методи.

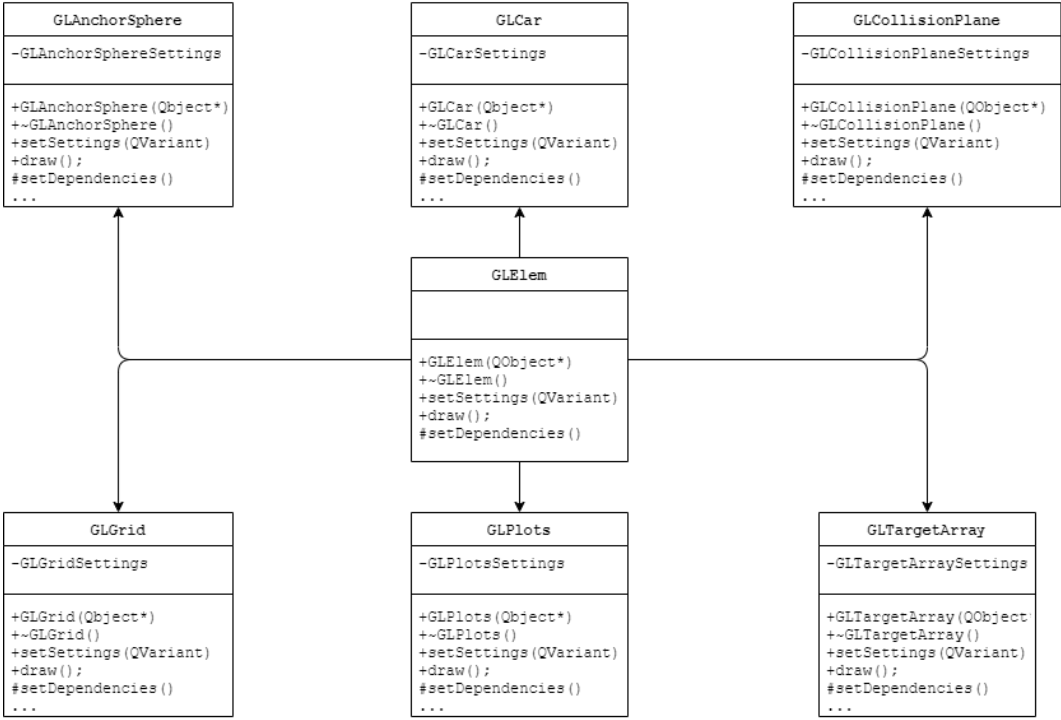


Рисунок 3.10 – Діаграма нащадків класу GLElem

Окрім класів рендеренгу слоїв, в програмі також є клас GLCamera та GLWidget.

GLCamera- клас камери, що відповідає за переміщення, поворот, нахил поля зору, та зміни типу відображення з перспективної на орфографічну і назад.

GLCamera має наступні методи та поля(табл 3.6).

Таблиця 3.6- Методи та поля класу GLCamera

Поля та методи класу	Опис
1	2
public:	
GLCamera()	Конструктор
setSettings(GLCamearSettings)	Встановлення налаштувань камери.
adjust(int, int)	Налаштування проєкції, враховуючи ширину та висоту поля для перегляду.
setPos(GLPoint, bool)	Встановлення положення камери у просторі
setPos(double, double, double, bool)	

Продовження таблиці 3.6

1	2
move(GLVector, bool)	Переміщення камери у просторі
move(eGLCamMoveDir, bool)	
setRot(GLVector, bool)	Встановлення напрямку зору камери
rotate(GLVector, bool)	Поворот камери навколо себе.
rotate(GLPoint, GLPoint, bool)	
rotate(eGLCamRotDir, bool)	
anchorRot(GLPoint, GLVector, bool)	Поворот камери навколо точки у просторі.
anchorRot(GLPoint, GLPoint, GLPoint, bool)	
anchorRot(GLPoint, eGLCamRotDir, bool)	
setMoveSpeed(double)	Встановлення швидкості камери
setRotSpeed(double)	Встановлення швидкості повороту камери
setBound(GLCamBound)	Встановлення границь відображення камери
setCamType(eGLCamType)	Встановлення типу камери(перспективна або ортогональна)
GLColor getScreenColor (QRect, GLPoint)	Знаходження кольору пікселя на координатах
GLColor ScreenToGLCoord(GLPoint, bool checkCollision)	Переведення координат монітору у координати OpenGL
GLColor GLCoordToScreen(GLPoint)	Переведення координат OpenGL у координати монітору
GLCamSettings getSettings()	Отримання налаштувань камери
signals:	
posChanged(GLPoint)	Сигнали, що викликаються при зміні положення камери у просторі
posChanged()	
private:	
GLPoint calculateAnchorPose(GLVector, GLPoint)	Підрахунок нового положення камери відносно точки у просторі, повернутої на вектор
applyRotation()	Застосувати поворот камери
normalizeRotation()	Нормалізувати поворот камери (від 0 до 360, від мінімального нахилу до максимального)
GLCamSettings m_settings	Налаштування камери

GLWidget- похідний клас від QOpenGLWidget, що відповідає за відображення сцени OpenGL, використовуючи стандартні засоби QtCreator. Цей віджет відповідає за зміну розміру вікна перегляду, оновлення екрану, передачу керування камерою у клас GLCamera та ініціалізацію сцени. Методи та поля класу наведені у таблиці 3.7.

Таблиця 3.7- Методи та поля класу GLWidget

Поля та методи класу	Опис
<i>public:</i>	
GLWidget()	Конструктор
<i>protected:</i>	
initializeGL()=0	Ініціалізація компонентів OpenGL
paintGL()=0	Рендерінг сцени OpenGL
resizeGL()=0	Зміна геометрії вікна відображення

Так як увесь функціонал буде виконуватись в класах, то все що необхідно зробити у основному циклі це оновити положення камери, та по черзі визвати кожен клас для відображення їх слоїв.

3.6 Алгоритми

Оскільки радіолокатори фіксують не тільки корисну інформацію про об'єкти, а також «завади» у вигляді хмар, птахів та інших об'єктів, потрібен механізм для вибірки корисної інформації.

Алгоритм вибірки по кількості точок дозволяє користувачеві виділити траєкторії, які мають мінімально допустиму кількість зареєстрованих точок згідно вимогам користувача.

Завдяки тому, що з самого початку було згруповано інформацію про об'єкти у відповідні траєкторії, для вибірки траєкторії по кількості точок, потрібно лише порівняти кількість точок кожної траєкторії з бажаним мінімальним значенням. Алгоритм наведено на рисунку 3.11.

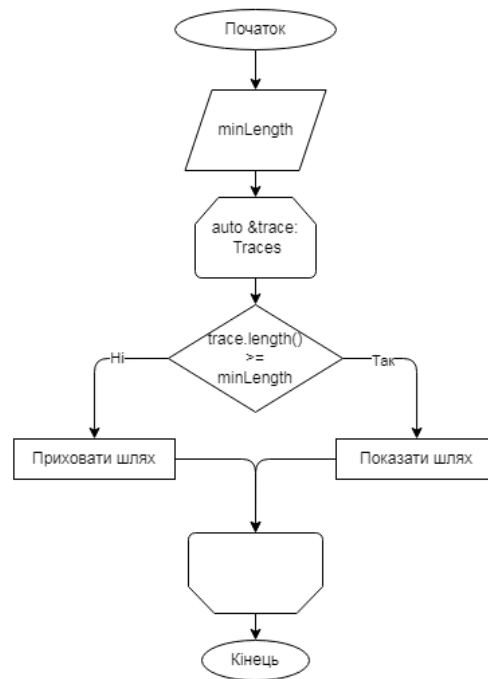


Рисунок 3.11 - Алгоритм вибірки траєкторій по кількості точок

Алгоритм вибірки по часу дозволяє користувачеві виділити певний проміжок часу використовуючи часовий повзунок, і програма відобразить тільки ті точки, які були зареєстровані у заданому користувачем проміжку. Алгоритм реалізовано наступним чином(рис 3.12).

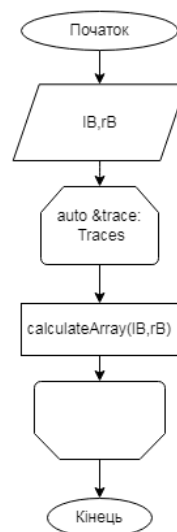


Рисунок 3.12- Алгоритм вибірки точок по часу

Алгоритм вибірки по типу точки дозволяє користувачу вибрати відобразити тільки ті точки, які мають той самий тип, який задав користувач.

Алгоритм реалізовано наступним чином(рис 3.13)

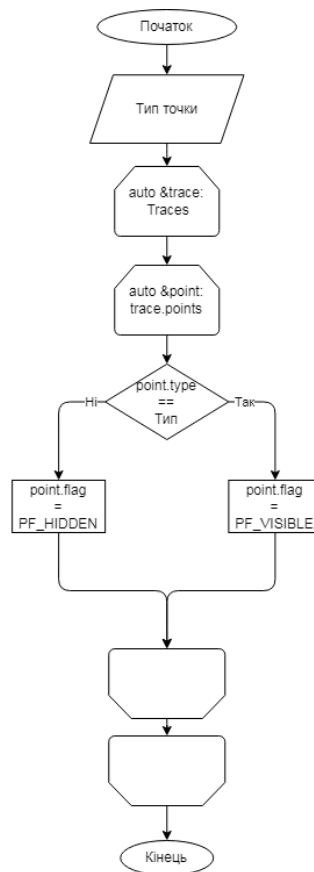


Рисунок 3.13- Алгоритм вибірки точок по типу

Для збереження інформації користувача, я вибрав формат .ini, оскільки він є доволі простим для розуміння, користувач у будь який момент зможе переглянути його, та змінити налаштування не запускаючи програму. А також у Qt є вбудована підтримка цього формату, що спрощує роботу з ним. Оскільки користувач може налаштовувати різні аспекти програми, я вирішив виділити їх у різні підгрупи.

Основним файлом налаштувань є файл «3DSettings.ini», який зберігає у собі усі налаштування користувача стосовно відображення двох та трьох вимірного простору. Структура файлу наведена на рисунку 3.14.

[illegible]

Рисунок 3.14- Структура файлу налаштувань двох та трьох вимірною вигляду

При запуску програми виконується метод «loadSettings» класу «PlaneViewerSettings», який задає параметри усім слоям які беруть участь у відображенні. Перед завершенням програми, виконується метод «saveSettings» який зберігає налаштування у файл. Алгоритм збереження та завантаження виглядає наступним чином (рис 3.15-3.16).

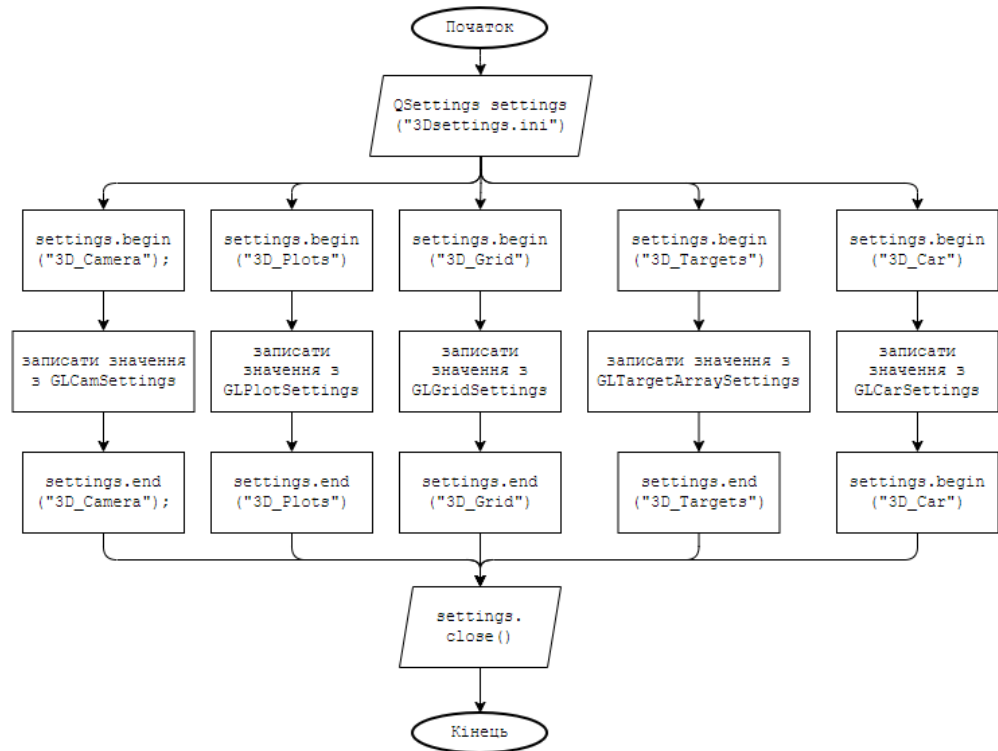


Рисунок 3.15- Запис налаштувань у файл 3DSettings.ini

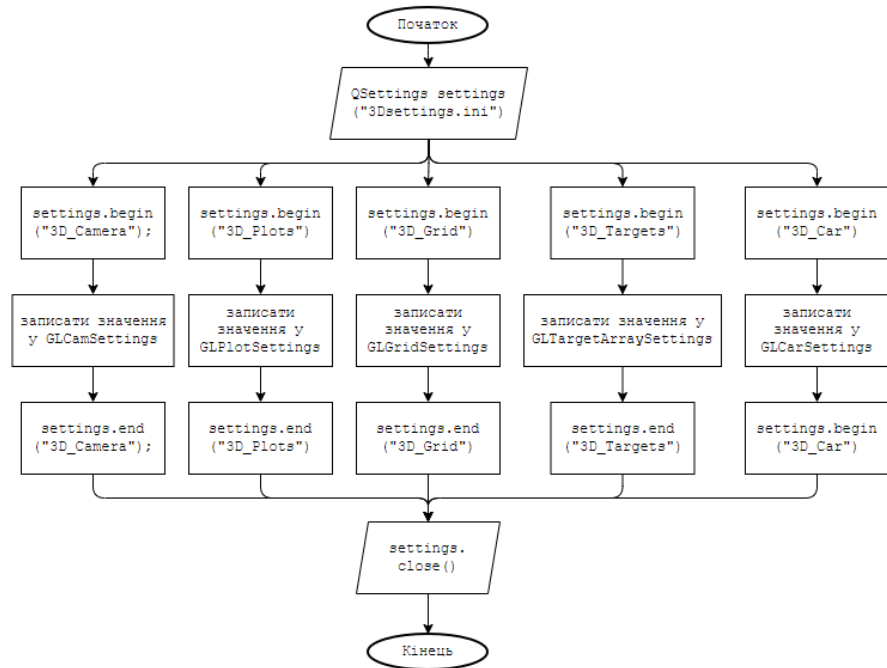


Рисунок 3.16- Завантаження налаштувань з файлу 3DSettings.ini

3.7 Висновок за розділом 3

У цьому розділі було описано реалізацію основних компонентів розроблюваного додатку. Було розглянуто реалізацію та організацію інтерфейсу користувача, структуру та організацію даних, основні класи для роботи з OpenGL та алгоритм завантаження та зберігання користувацьких налаштувань.

Реалізовано простий для користування та налаштування інтерфейс користувача, що складається з наступних сегментів : Таблиця траєкторій, Вікно відображення, Керування часом, Налаштування. Інтерфейс адаптується під різні розміри екрану, та має можливість налаштування розміру, або приховування деяких сегментів.

Додаток використовує OpenGL для рендеру зображення у двох та трьох вимірному просторах. Для реалізації різних компонентів сцени було розроблено наступні класи: GLElem, GLGrid, GLAnchorPoint, GLCollisionPlane, GLPlots, GLTargetArray. Для керування камерою було розроблено клас GLCamera. Щоб відображати сцену у вікні програми, було використано клас GLWidget. У купі ці класи реалізують гнучку та швидко взаємодію користувача з OpenGL.

Реалізовано функції збереження та завантаження користувацьких налаштувань вигляду вікна перегляду. Користувач може налаштувати вигляд програми під себе, та зберегти налаштування у файл settings.ini. При наступних запусках програма буде автоматично завантажувати останні налаштування користувача.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис застосунку

Розроблений застосунок призначений для зчитування файлів реєстрації роботи радіолокаційних систем, організації зчитаної інформації у зручний для навігації та пошуку вигляд та відображення її у двох та трьох вимірному просторі. Користувачам що використовують даний застосунок доступні наступні функції:

- вибір та завантаження файлів реєстрації;
- вибір типу відображення зчитаної інформації (двох та трьох вимірний);
- вибірка траєкторій об'єктів по кількості зареєстрованих точок;
- вибірка точок по часу;
- зміна загального вигляду програми;
- налаштування вигляду окремих траєкторій та окремих точок;
- програвання зчитаної інформації по часу.

4.2 Умови виконання програми

Мінімальні системні вимоги та програмне забезпечення, що необхідні для коректного функціонування застосунку:

- операційна система (ОС)- Unix системи чи Windows;
- відеокарта з підтримкою OpenGL 3.0 або вище;
- процесор Intel Core i3 (або аналогічний);
- 3 ГБ ОЗП або більше;
- місце на жорсткому диску 70 мб.

4.3 Інструкція по експлуатації

4.3.1 Характеристики програми

Застосунок для зчитування файлів реєстрації та подальшого відображення зчитаної інформації має наступні характеристики:

а) технології:

1) C++: низькорівнева мова програмування, має багато парадигм програмування, швидке виконання коду;

2) Qt Creator: сучасне IDE для створення крос-платформних застосунків з графічним інтерфейсом;

3) OpenGL: один із інтерфейсів роботи з графікою. Є крос-платформним та доволі зручним для використання та написання програм;

б) функціональність:

1) завантаження файлів реєстрації: користувач може завантажувати файли через відповідне меню у програмі, або просто перетягнувши файл у область вікна програми;

2) зміна відображення: користувач може змінити налаштування відображення програми за своїх потреб. Усі налаштування зберігаються, та будуть використані при наступному запуску програми;

3) організації інформації: вибірка потрібної траєкторії по кількості точок, відображення тільки точок по конкретному каналу та вибірка точок у часовому діапазоні;

4) відображення супутньої інформації: при виборі точки у таблиці траєкторій або у вікні відображення, супровідна інформація буде відображена у формулярі об'єкту;

в) інтерфейс користувача:

1) адаптивний дизайн: вікно програми складається з секторів, форму яких у разі потреби можна змінити, або приховати сектор повністю.

Застосунок для зчитування файлів реєстрації- легкий, швидкий та зручний інструмент для аналізу записаної інформації до файлів реєстрації.

4.3.2 Перелік файлів що потрібні для функціонування програми

Застосунок складається з наступних файлів та папок:

- tables: папка з таблицями для коректного перерахування координат об'єкту з глобальних до локальних координат локатору;
- styles: папка створена Qt Creator для коректного відображення стилів для елементів інтерфейсу;
- platforms: папка створена Qt Creator для коректної роботи застосунку на різних ОС;
- 3Dsettings.ini: користувацькі налаштування відображення ;
- settings.ini: остання папка, з якої було завантажено файл;
- RegReader.exe: основний виконуваний файл програми;
- *.dll: допоміжні бібліотеки для коректної роботи програми.

Цей перелік файлів та папок представляє собою основну структуру застосунку.

4.3.3 Звернення до програми

Оскільки розроблювальний застосунок є вузько-направленим, та його використання планується у вузько-направлених сферах, тому для його встановлення та запуску необхідно виконати наступні кроки:

- отримати архів з програмою: Отримайте архів .7z від розробника, уточнивши яку операційну систему ви плануєте використовувати для запуску додатку;
- розпакувати архів: Після отримання архіву від розробника, розпакувати його у зручне для вас місце;
- запустити програму: Зайдіть у папку з програмою і запустіть виконуючий файл RegReader*.

Додаток не вимагає додаткових налаштувань або конфігурацій.

4.3.4 Вихідні й вхідні дані

До вхідних даних належать:

- дані про налаштування: при запуску програми, вона зчитує збережені налаштування користувача з файлів 3Dsettings.ini та settings.ini;
- файл реєстрації: Програма зчитує файли реєстрації з розширенням .reg.

До вихідних даних належать:

- дані про налаштування: при кожному завершенню роботи застосунку, він записує свої налаштування до файлів 3Dsettings.ini та settings.ini.

4.4 Методика та результати тестування

Тестування застосунку проведемо у два етапи. На початку протестуємо функціональність застосунку. На другому етапі протестуємо максимальне навантаження яке зможе витримати застосунок.

Тестування функціоналу почнемо з завантаження файлу до програми двома способами: за допомогою вікна завантаження (рис 4.1-4.3) та перетягуванням.

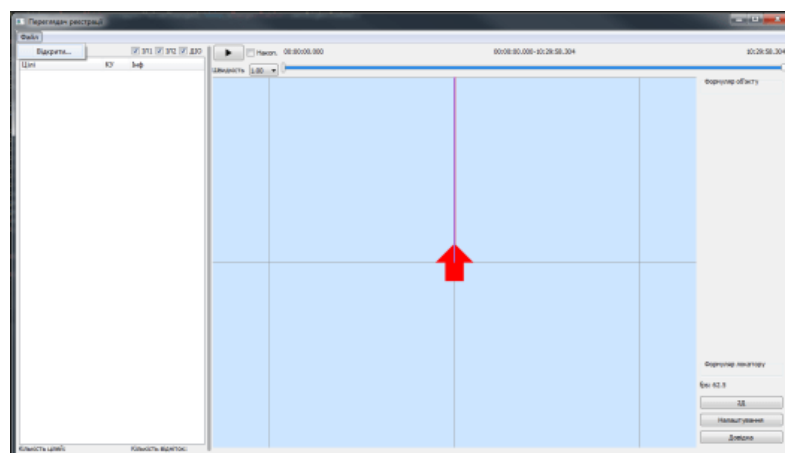


Рисунок 4.1- Виклик вікна відкриття файлу

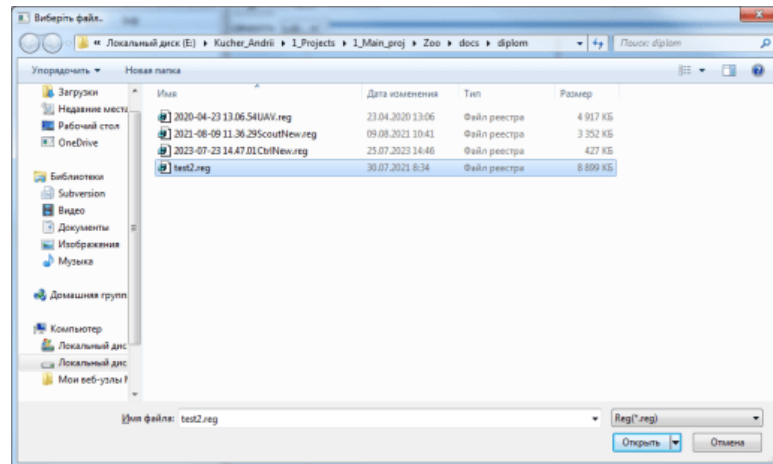


Рисунок 4.2- Вибір файлу реєстрації

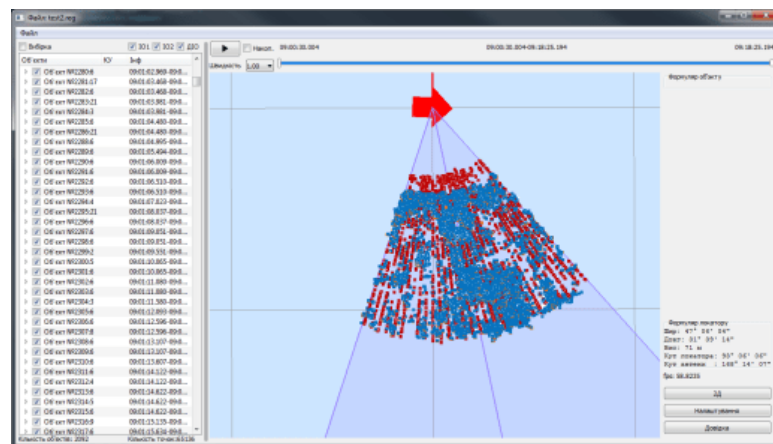


Рисунок 4.3- Відображення зчитаної інформації з файлу

Виконаємо вибірку по кількості точок на одну траєкторію, для цього над таблицею поставимо галочку «Вибірка» та встановимо мінімальне значення точок для траєкторії(рис 4.4). Після цього переконаємося що у таблиці та у вікні перегляду залишилися тільки ті траси, кількість точок яких задовольняє мінімальну(рис 4.5).

Рисунок 4.4- Встановлення мінімальної кількості точок на трасу

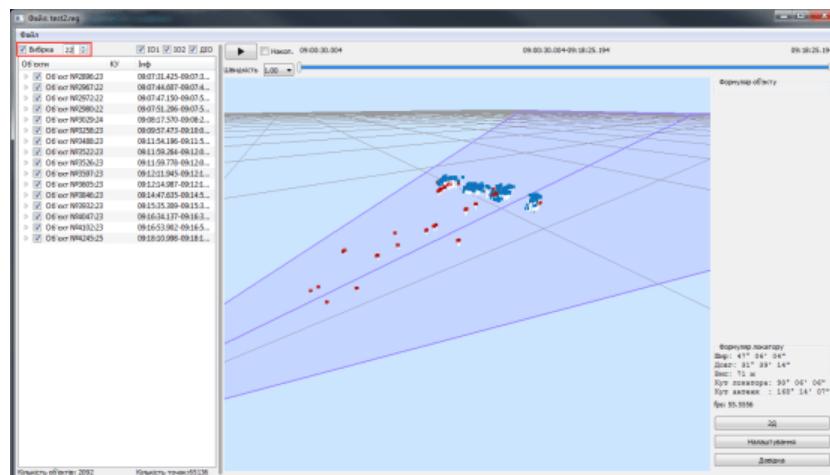


Рисунок 4.5- Результат вибірки траєкторій ко по кількості точок

Тепер виключемо відображення точок по каналу ІО, для цього над таблицею знімемо галочки «ІО1» та «ІО2», попередньо виключивши «Вибірку» щоб краще побачити результат(рис 4.6).

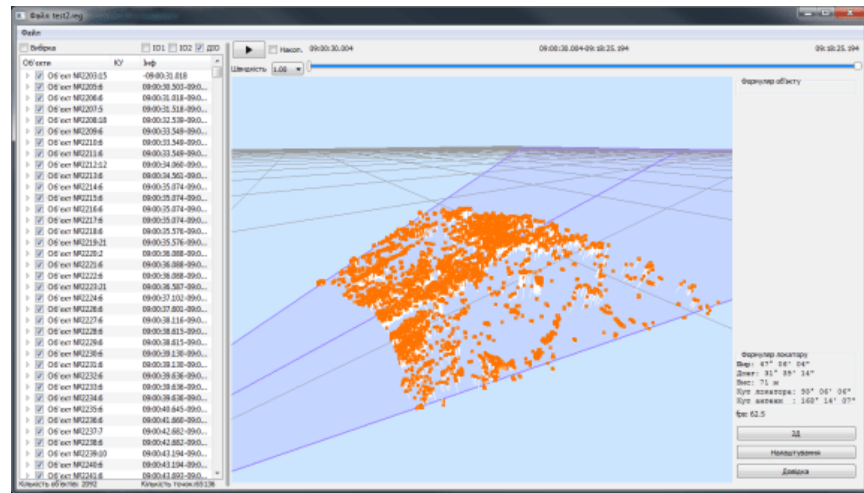


Рисунок 4.6- Відключення відображення точок по каналу «ІО»

Виділемо одну з точок та встановимо їй інший колір(рис 4.7-4.8)

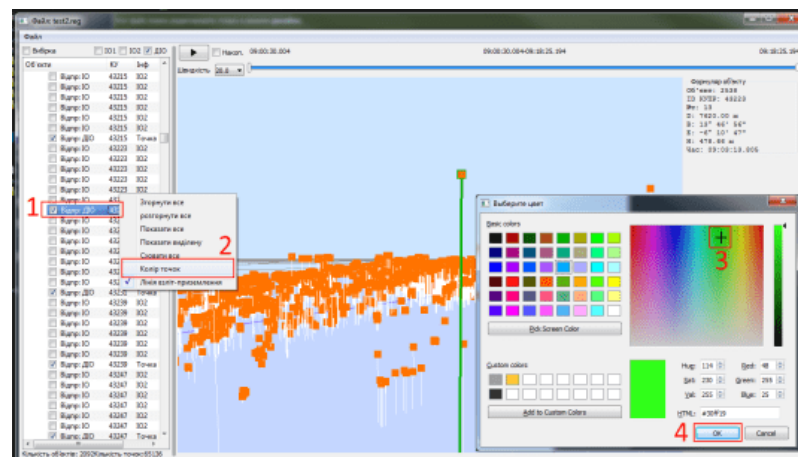


Рисунок 4.7- Встановлення нового кольору точки

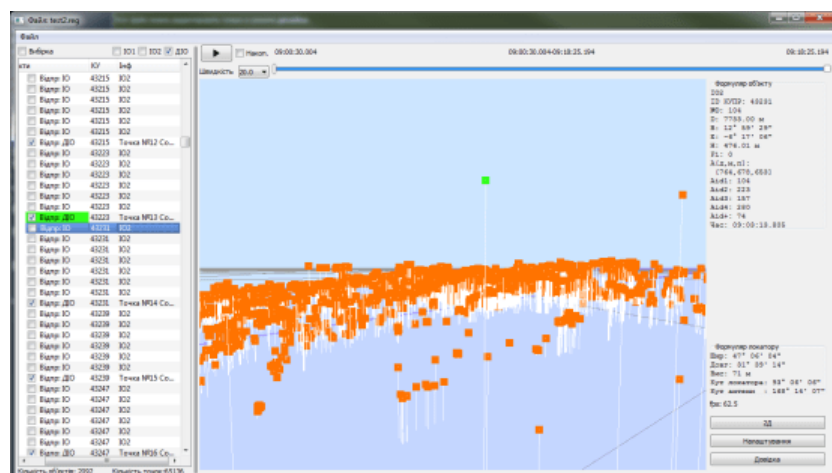


Рисунок 4.8- Результат встановлення кольору

Встановимо часовий проміжок(рис 4.9).

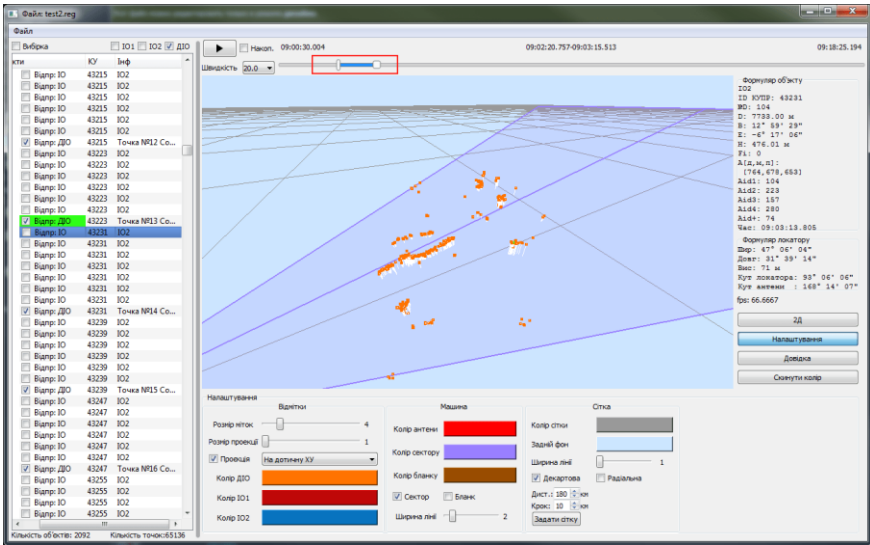


Рисунок 4.9- Встановлення часового проміжку

Для перевірки максимального навантаження, використаємо файл реєстрації який важить 18МБ. Цей файл реєстрував роботу радіолокаційного системи та має у собі інформацію про 278,354 точки. Як можна побачити з кількості кадрів, програма без проблем відкриває файли такого об'єму (рис 4.10).

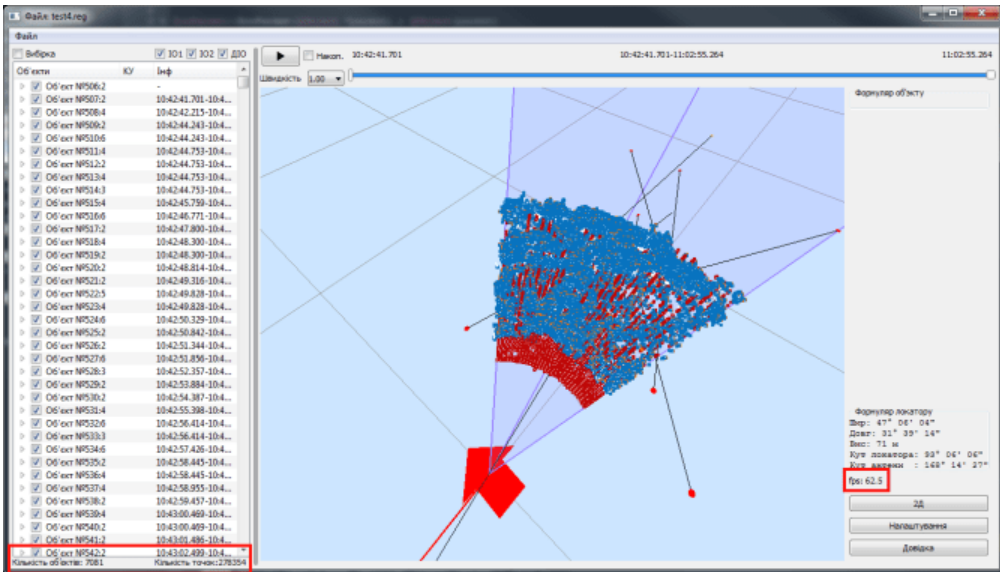


Рисунок 4.10- Продуктивність програми при відкриванні файлу 18МБ

Збільшимо об'єм файлу вдвоє, кількість точок зростає з 278,354 до 480,503. Не дивлячись на довге завантаження файлу, програма працює без нарікань(рис 4.11)

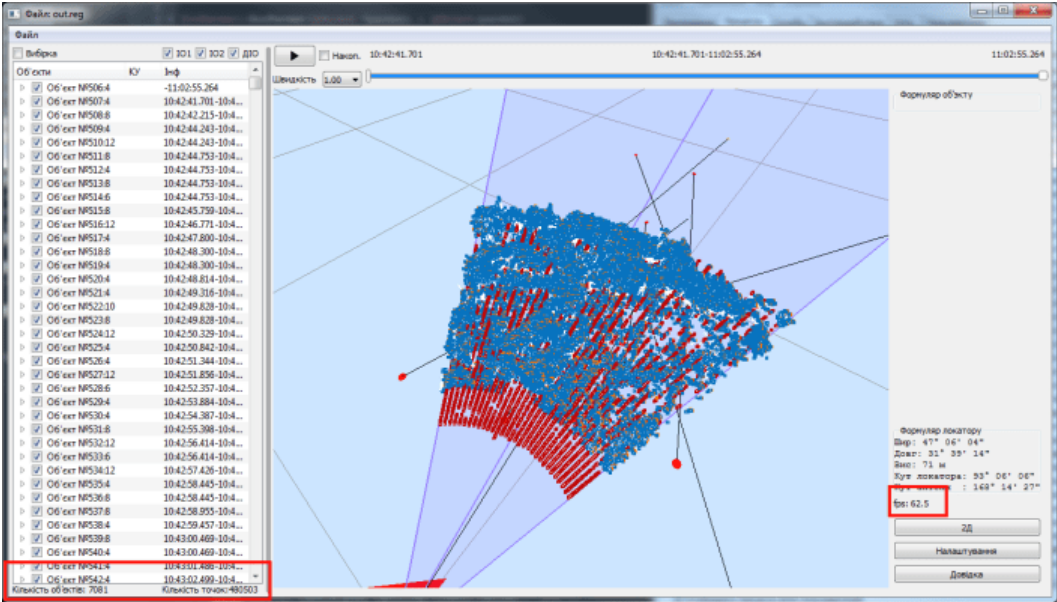


Рисунок 4.11- Продуктивність програми при відкриванні файлу 36МБ

При відкриванні файлу вагою 72МБ (884тис. точок), кількість кадрів впала у два рази(рис 4.12)

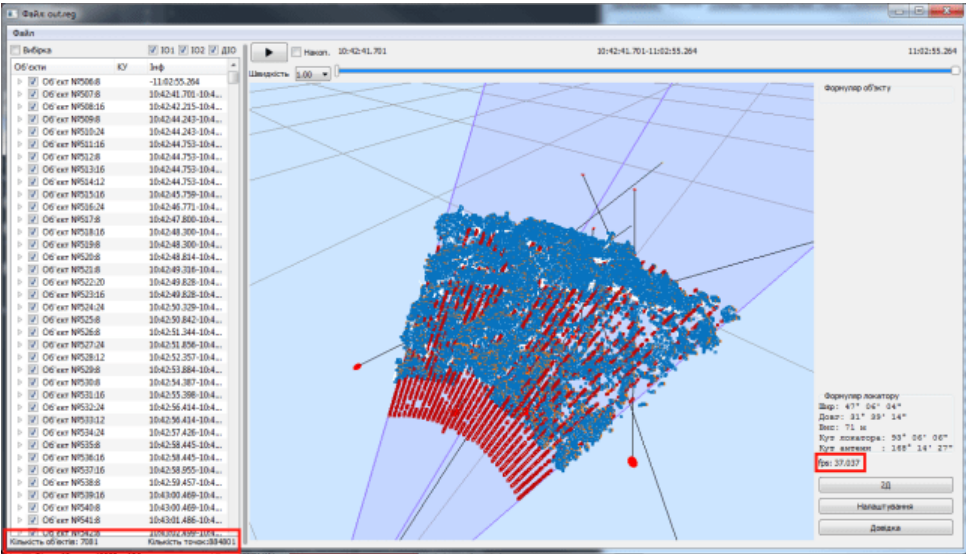


Рисунок 4.12- Продуктивність програми при відкриванні файлу 72МБ

В результаті виконання завдання, було отримано застосунок, який здатен відображати 278 тисяч точок одночасно, без втрати продуктивності.

ВИСНОВКИ

В результаті виконання дипломної роботи, мною було проаналізовано існуючі аналоги, виділено їх переваги та недоліки. Базуючись на результатах аналізу було вирішено розробити програмний засіб для читання файлів реєстрації радіолокаційних систем і подальшої організації зчитаної інформації у зручний для розуміння, навігації та аналізу формат.

Даний програмний засіб представляє собою застосунок з наступним функціоналом:

- завантаження файлів реєстрації радіолокаційних систем з розширенням «.reg»;
- відображення зареєстрованих координат повітряних об'єктів у 2Д та 3Д форматах;
- відображення результатів реєстрації на встановленому часовому інтервалу;
- налаштування зовнішнього вигляду окремих елементів відображення результатів реєстрації (колір, крок сітки і тд);
- відображення інформації по вибраному об'єкту;
- вибірка потрібної користувачеві інформації;
- зберігання налаштувань користувача.

Розроблений програмний засіб було протестовано відповідно технічному завданню.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Begin End OpenGL(для новачків) [Електронний ресурс]. – Режим доступу :
<https://www.youtube.com/watch?v=09GRZ0c0LlA&list=PLBOPkQsFLCR0XuXoCf6PMhVmD2ukHXIBK&pp=iAQB>.
2. Qt Style Sheets Reference [Electronic resource]. – Access mode:
<https://doc.qt.io/qt-5/stylesheets-reference.html>.
3. Rodríguez Jorge docs.GL [Electronic resource]. – Access mode:
<https://docs.gl/>.
4. TheCherno Game Engine [Electronic resource]. – Access mode:
https://www.youtube.com/watch?v=JxIZbV_XjAs&list=PLlrATfBNZ98dC-V-N3m0Go4deliWHPFwT&pp=iAQB.
5. TheCherno OpenGL Series [Electronic resource]. – Access mode:
https://www.youtube.com/watch?v=W3gAzLwfIP0&list=PLlrATfBNZ98foTJPJ_Ev03o2oq3-GGOS2&pp=iAQB.
6. MSDN OpenGL Tutorial [Electronic resource]. – Access mode:
<https://learn.microsoft.com/ru-ru/windows/win32/opengl/opengl>.
7. Qt vs Clion vs VisualStudio vs VSCode [Electronic resource]. – Access mode: <https://www.youtube.com/watch?v=dQw4w9WgXcQ>.
8. Read from file C++ [Electronic resource]. – Access mode:
<https://www.geeksforgeeks.org/how-to-read-from-a-file-in-cpp/>.
9. C++ Documentation CPLUSPLUS.com [Electronic resource]. – Access mode: <https://cplusplus.com/>.
10. GeekForGeeks [Electronic resource]. – Access mode:
<https://www.geeksforgeeks.org/how-to-read-from-a-file-in-cpp/>.
11. StackOverflow [Electronic resource]. – Access mode:
<https://stackoverflow.com/>.

ДОДАТОК А
Технічне завдання

Вступ

Метою роботи є розробка застосунку для читання файлів реєстрації радіолокаційних систем і подальшої організації зчитаної інформації у зручний для розуміння, навігації та аналізу формат. Застосунок повинен забезпечувати швидку роботу з файлами реєстрації, вибірку потрібної користувачеві інформації та відображення зчитаних даних у двох та трьох вимірному вигляді.

A.1 Підстава для розробки

Проаналізувавши існуючих конкурентів та виділивши їх переваги і недоліки, було вирішено що існуючі аналоги не мають функціонал, який задовольняв би потреби користувачів. На основі цього було вирішено реалізувати додаток з функціоналом який потребує користувач.

A.2 Призначення розробки

Завданням програмного засобу є спрощення аналізу вихідних даних радіолокаційних систем.

Користуватись програмою зможуть особи, які мають доступ до вищевказаних систем.

A.3 Основні вимоги до програми, що розроблюється

A.3.1 Вимоги до не функціональних характеристик

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування графічної частини застосунку:

- операційна система (ОС)- Unix системи чи Windows;
- процесор Intel Core i3 (або аналогічний);
- 3 ГБ ОЗП або більше;
- місце на жорсткому диску 70 мб.

A.3.2 Вимоги до функціональних вимог

Розроблений програмний продукт має забезпечувати користувача наступним функціоналом:

- програмний засіб повинен забезпечувати завантаження файлів реєстрації радіолокаційних систем з розширенням «.reg»;
- програмний засіб повинен забезпечити відображення зареєстрованих координат повітряних об'єктів у 2Д та 3Д форматах;
- програмний засіб повинен бути доступною, та не обмежувати свій функціонал (без монетизації);
- програмний засіб повинен працювати локально(без підключення до інтернету);
- програмний засіб повинен мати можливість відображення результатів реєстрації на встановленому часовому інтервалу;
- програмний засіб повинен забезпечувати можливість налаштування зовнішнього вигляду окремих елементів відображення результатів реєстрації (колір, крок сітки і тд);
- програмний засіб повинен забезпечувати відображення інформації по вибраному об'єкту;
- програмний засіб повинен забезпечити вибірку потрібної користувачеві інформації;
- програмний засіб повинен зберігати налаштування користувача;
- програмний засіб повинен забезпечувати дружній інтерфейс для перегляду та налаштування вигляду повітряної обстановки за вимогами

користувача.

A.3.3 Умови експлуатації

Для роботи застосунку потрібно:

- дискретна або вбудована відеокарта з підтримкою OpenGL 3.0 або вище.

A.4 Стадії та етапи розробки

Можна виділити такі стадії та етапи розробки системи:

- а) збирання і аналіз вимог. Проводиться аналіз існуючих рішень, виділяються їх недоліки і переваги;
- б) розробка технічного завдання (ТЗ). На основі результатів аналізу робиться висновок з детально розписаними вимогами до програмного забезпечення (ТЗ);
- в) вибір програмних засобів. Перш за все вибирається мова програмування, на якій буде розроблено додаток. Аналізуються декілька кандидатів, виділяються переваги та недоліки кожного. Порівнюються результати і вибирається найбільш-підходяща мова програмування. Так само порівнюються IDE та інтерфейси роботи з графікою;
- г) реалізація компонентів. Реалізується інтерфейс користувача та основний функціонал програми. Розробляються базові та похідні класи. Описуються основні алгоритми для роботи системи, організація та структура даних;
- д) тестування. Проводиться ряд тестів для знаходження недоліків у графічному інтерфейсі, «багів» у програмі;
- е) розміщення та технічна підтримка. Програма розміщується у локальній системі підприємства;
- ж) оформлення документації.

A.5 Порядок контролю та приймання системи

Приймання системи здійснюється відповідно до складу та змісту робіт, наведених у технічному завданні.

Звіт про проведену роботу доноситься у усній формі замовнику. Після звітування, демонструється результат проведеної роботи. При наявності «багів» або нових побажань від замовника, складається документ «Чернетка» зі списком побажань та правок.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Назва програмного застосунку- «Переглядач реєстрації повітряної обстановки».

Застосунок був розроблений на мові програмування C++ з використанням бібліотек та редактору від компанії Qt.

Б.2 Функціональне призначення

Програмний продукт має наступний функціонал:

- завантаження файлів реєстрації радіолокаційних систем з розширенням «.reg»;
- відображення зареєстрованих координат повітряних об'єктів у 2Д та 3Д форматах;
- відображення результатів реєстрації на встановленому часовому інтервалу;
- налаштування зовнішнього вигляду окремих елементів відображення результатів реєстрації (колір, крок сітки і тд);
- відображення інформації по вибраному об'єкту;
- вибірка потрібної користувачеві інформації;
- зберігання налаштувань користувача.

Б.2.1 Функція завантаження файлів реєстрації

Б.2.2 Функції відображення у 2Д та 3Д форматі

Функції відображення реалізують простий для розуміння та налаштування формат візуального відображення зчитаної інформації з файлу реєстрації.

Б.2.3 Відображення результатів на встановленому часовому проміжку

Функція дозволяє користувачеві задавати проміжок часу, тим самим відсортовувати відмітки, які входять до цього проміжку.

Б.2.4 Налаштування зовнішнього вигляду

Функціонал програми має гнучке налаштування вигляду вікна відображення, дозволяючи користувачеві налаштувати програму під себе.

Б.2.5 Відображення інформації по вибраному об'єкту

Функція дозволяє вибрати об'єкт у таблиці об'єктів або у вікні відображення, та отримати детальну інформацію про вибраний об'єкт у формулярі об'єкту.

Б.2.6 Вибірка інформації

Функціонал програми що передбачає вибірку об'єктів по заданим критеріям: кількість точок на один об'єкт, тип точок(ІО1, ІО2 та ДІО).

Б.2.7 Зберігання налаштувань користувача

Функціонал програми забезпечує зберігання останніх налаштувань користувача, та завантаження збережених налаштувань при наступному запуску програми.

Б.3 Опис логічної структури

Логічну структуру програми визначає її функціональне призначення й можливості обраних програмних і інструментальних засобів проектування. Логічну структуру системи можна представити у вигляді моделі, представленої на рисунку Б.1.

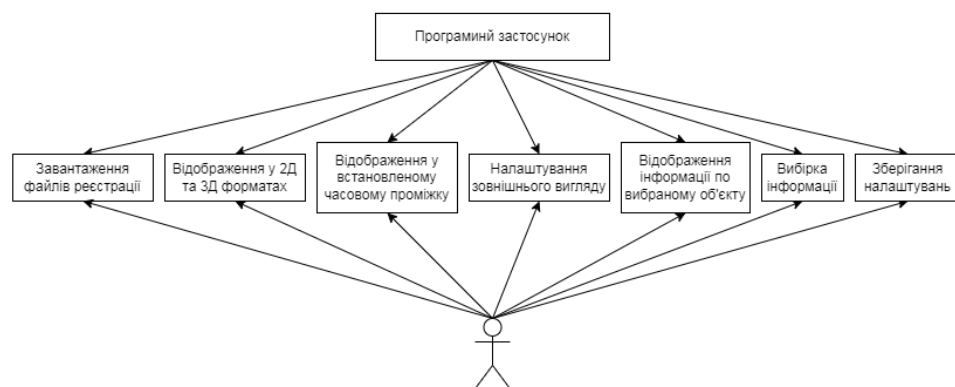


Рисунок Б.1-Логічна структура програми

Б.4 Використані технічні засоби

Для запуску програмного забезпечення необхідні такі програмно-технічні засоби:

- операційна система (ОС)- Unix системи чи Windows;
- відеокарта з підтримкою OpenGL 3.0 або вище;
- процесор Intel Core i3 (або аналогічний) ;
- 3 ГБ ОЗП або більше;
- місце на жорсткому диску 70 мб.

Б.5 Завантаження та запуск

Оскільки розроблювальний застосунок є вузько-направленим, та його

використання планується у вузько-направлених сферах, тому для його встановлення та запуску необхідно виконати наступні кроки:

- отримати архів з програмою: отримайте архів .7z від розробника, уточнивши яку операційну систему ви плануєте використовувати для запуску додатку;
- розпакувати архів: після отримання архіву від розробника, розпакувати його у зручне для вас місце;
- запустити програму: зайдіть у папку з програмою і запустіть виконуючий файл RegReader*.

Додаток не вимагає додаткових налаштувань або конфігурацій.

Б.6 Вхідні та вихідні дані

До вхідних даних належать:

- дані про налаштування: при запуску програми, вона зчитує збережені налаштування користувача з файлів 3Dsettings.ini та settings.ini;
- файл реєстрації: програма зчитує файли реєстрації з розширенням .reg.

До вихідних даних належать:

- дані про налаштування: при кожному завершенню роботи застосунку, він записує свої налаштування до файлів 3Dsettings.ini та settings.ini.

ДОДАТОК В
Текст програми

B.1 main.cpp

```
//#include "Windows/Main/mainwindow.h"
#include "windows/TargetTable/TargetTable.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    TargetTable w;
    w.setAcceptDrops(true);
    w.show();
    return a.exec();
}
```

B.2 mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QFileDialog>
#include <QDragEnterEvent>
#include <QDropEvent>
#include <QMimeData>
#include <QLabel>
#include <QTimer>
#include "Common/Codograms/CodogrMain.h"
#include "Parser/regParser.h"
```

```

#include "windows/Table/customTable.h"
#include "windows/TargetTable/TargetTable.h"
#include "Common/RegReaderAlgorithms.h"
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

signals:
    void newFile(QString);
private:
    void initConnections();
//    void parse(QString path);
private slots:
    void selectFile();
    void showParseState(bool);
    void showCU();
private:
    Ui::MainWindow *ui;
    RegParser m_parser;
    QLabel *state,*codCnt;
    TargetTable table;

```

```

CdgrVector* codograms;           //массив кодограм

// CdgrVectorPtr<UCdgrRegDIO>* Dio; //канал РПД-ВА18
// CdgrVectorPtr<UCdgrPs01Reg>* poi; //канал РПД-ПОИ
// CdgrAdmTimeMap* DioAdm;        //канал управления
// CdgrVectorPtr<UCdgrInside>* inside; //внутренний канал

// CdgrTargetMap targets;         //цели

QString lastPath=".";             //последний открытый каталог


// QWidget interface
protected:
    void dragEnterEvent(QDragEnterEvent *event) override;
    void dropEvent(QDropEvent *event) override;
};
#endif // MAINWINDOW_H

```

B.3 mainwindow.cpp

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    QSettings settings("settings.ini",QSettings::IniFormat);
    settings.beginGroup("lastPath");

```

```

lastPath=settings.value("Path").value<QString>();
settings.endGroup();

ui->setupUi(this);
initConnections();
state=new QLabel();
codCnt=new QLabel();
ui->statusbar->addWidget(codCnt);
ui->statusbar->addWidget(state);
setWindowTitle("Переглядач Зоо");
ui->showCU->setVisible(false);
// table.setTargets(&targets);
// table.show();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::initConnections()
{
    connect(ui->openFile,
&QAction::triggered,this,&MainWindow::selectFile);

//          connect(ui->showCU,          &QAction::triggered,
this,&MainWindow::showCU);

//      connect(&table,&TargetTable::newFileDropped,[=](QString    file){
parse(file);
//      setWindowTitle(file.split("\\").last()); setFocus();});

```

```
}
```

```
//void MainWindow::parse(QString path)
//{
////  codograms.clear();
////  Dio.clear();
////  targets.clear();
////  poi.clear();
////  DioAdm.clear();
////  inside.clear();

////  qDebug()<<"clear done";
////  m_parser.parse(codograms,path);
////  qDebug()<<"parce done";
////  organize(codograms,&dio,&io,&DioAdm,&inside,&targets);
////  qDebug()<<"org done";
////  if(inside.begin()->cdgr->header.type==cdgr::uiInsideTypeState)
////    table.setCarState(inside.begin()->cdgr->state);
////  qDebug()<<"car done";

////  ui->tableView->showCodograms(codograms);
////  qDebug()<<"tv done";

////  table.update();
////  qDebug()<<"3d done\n-----\n";
//}
```

```
void MainWindow::selectFile()
{
    QFileDialog dialog;
```

```

        QUrl fileName=dialog.getOpenFileUrl(this,"Виберіть
файл..",lastPath,"Reg(*.reg)");
        if(!fileName.isValid())
            return;
        setWindowTitle(fileName.fileName().split('/').last());
        lastPath=QFileInfo(fileName.toLocalFile()).absolutePath();
        // parse(fileName.toLocalFile());
        emit newFile(fileName.toLocalFile());
    }

void MainWindow::showParseState(bool parseState)
{
    QString result="Файл ",cods;
    result+= parseState? "зчитано":"не вдалось зчитати";
    cods= parseState? tr("КІОЬкість кодограм:%1").arg(codograms-
>length()):"";
    state->setText(result);
    codCnt->setText(cods);
    QTimer::singleShot(5000,[=]() { state->setText(""); });
}

void MainWindow::showCU()
{
    table.show();
}

void MainWindow::dragEnterEvent(QDragEnterEvent *event)
{
    if(event->mimeTypeData()->hasUrls() && event->mimeTypeData()-
>urls().last().fileName().split(".").last()=="reg")

```

```

        event->acceptProposedAction();
    }

void MainWindow::dropEvent(QDropEvent *event)
{
    for(auto url:event->mimeTypeData()->urls())
    {
        //    parse(url.toLocalFile());
        emit newFile(url.toLocalFile());
        setWindowTitle(url.fileName());
        table.setWindowTitle("Файл "+url.fileName());
    }
    event->acceptProposedAction();
    setFocus();
}

```

B.4 customTable.h

```

#ifndef CUSTOMTABLE_H
#define CUSTOMTABLE_H

#include <QMenu>
#include <QWidget>
#include <QLineEdit>
#include <QScrollBar>
#include <QTableView>
#include <QHeaderView>
#include <QStandardItem>
#include <QContextMenuEvent>
#include <QStandardItemModel>

```

```

#include "Common/RegReaderStruct.h"
#include "Parser/regParser.h"
class CustomTable : public QTableView
{
    Q_OBJECT
public:
    explicit CustomTable(QWidget *parent = nullptr);
    void showCodograms(QVector<RegCodogram>& codograms);
private:
    //Ініціалізація таблиці
    void initTable();
    void initModel();
    void adjustToView();

private:
    QStandardItemModel *m_table=nullptr;
    QMenu m_menu;
    QLineEdit *m_finder;

    CdgrVector *m_codograms = nullptr;
};

#endif // CUSTOMTABLE_H

```

B.5 customTable.cpp

```

#include "customTable.h"

CustomTable::CustomTable(QWidget *parent) : QTableView(parent)
{

```

```

initTable();
setHorizontalScrollMode(QAbstractItemView::ScrollPerPixel);
setVerticalScrollMode(QAbstractItemView::ScrollPerPixel);
setModel(m_table);
setWordWrap(false);
verticalHeader()->setDefaultSectionSize(23);
QPalette pl;
pl.setColor(QPalette::AlternateBase,qRgb(240, 240, 240));
setPalette(pl);
}

void CustomTable::showCodograms(QVector<RegCodogram>&
codograms)
{
    m_codograms=&codograms;
    QString data;
    // Mark buff;
    // uint32_t comID;
    // int overlapping;
    initModel();

    // QElapsedTimer timer;
    // timer.start();

    m_table->setRowCount(codograms.length());

    for(int i=0, sz = codograms.length();i<sz;i++)
    {
        //Заполняем поле HEX
        data=codograms[i].cdgr.toHex(' ').toUpper();
    }
}

```

```

for(int i=29;i<data.length();i+=30)
    data[i]='\n';
m_table->setItem(i,4,new QTableWidgetItem(data));

    //проверяем КС
    m_table->item(i,4)->setBackground(checkRegCodKs(codograms[i])?
QColor(57, 191, 84):QColor(191, 52, 52));

    //Заполняем поле Канал
    m_table->setItem(i,0,new
QStandardItem(cdgr::StrRegChannel[codograms[i].channel]));

    //Заполняем поле Тип
    switch (codograms[i].channel) {
        case RegCodogram::REG_DIO:
            {
                UCdgrRegDIO
                *cod=reinterpret_cast<UCdgrRegDIO*>(codograms[i].cdgr.data());
                m_table->setItem(i,2,new
QStandardItem(cdgr::StrRegCodTypes[codograms[i].channel][cod-
>header.type]));
            }
            break;

        case RegCodogram::REG_IO:
            {
                m_table->setItem(i,2,new
QStandardItem(cdgr::StrRegCodTypes[codograms[i].channel][reinterpret_cast<U
CdgrPs01Reg*>(codograms[i].cdgr.data())->header.type]));
            }
            break;
    }

```

```

        case RegCodogram::DIO_ADM:
        {
            m_table->setItem(i,2,new
QStandardItem(getRegAdmSubs(codograms[i])));
            //
SCdgrAdm
*cod=reinterpret_cast<SCdgrAdm*>(codograms[i].cdgr.data());
            //      qDebug()<<(int)cod->msg.ioHeadPart.admCdgrId;
            }
            break;
        case RegCodogram::Service_Inside:
        {
            m_table->setItem(i,2,new
QStandardItem(cdgr::StrRegCodTypes[codograms[i].channel][reinterpret_cast<U
CdgrInside*>(codograms[i].cdgr.data())->header.type]));
            }
            break;
        default:
            break;
    }

    //Заполняем инфу
    m_table->setItem(i,3,new QStandardItem(getRegData(codograms[i])));

    m_table->setVerticalHeaderItem(i,new
QStandardItem(codograms[i].time==(uint32_t)-1?
"--:--:--:---
":QTime::fromMSecsSinceStartOfDay(codograms[i].time).toString("hh:mm:ss.zzz
"))));

    if(codograms[i].admCod!=nullptr)
    {

```

```

        SCdgrAdm
        *admCod=reinterpret_cast<SCdgrAdm*>(codograms[i].admCod->cdgr.data());
        m_table->setItem(i,1,new    QStandardItem(QString::number(admCod-
>msg.ioHeadPart.admCdgrId)));
    }
}

// qDebug() << "Filling time:" << timer.elapsed();

setModel(m_table);
adjustToView();
}

void CustomTable::initTable()
{
    //Table init
    setStyleSheet("QTableView:!active{selection-background-
color:palette(highlight);selection-color:palette(highlighted-text);}");

    //QMenu init
    /*QAction* a1 = */m_menu.addAction("Налаштування таблиць");
    /*QAction* a2 = */m_menu.addAction("Налаштування кольору");
    /*QAction* a3 = */m_menu.addAction("Показати ЦВ");
    /*QAction* a4 = */m_menu.addAction("Показати 3D");
    QAction* a5 = m_menu.addAction("Знайти...");
    a5->setShortcut(Qt::CTRL | Qt::Key_F);
    connect(a5, &QAction::triggered, [=]() {m_finder->setVisible(!m_finder-
>isVisible());m_finder->setFocus();});

```

```

//Finder init
m_finder=new QLineEdit(this);
m_finder->setGeometry(5,30,125,25);
m_finder->setVisible(false);
m_finder->setPlaceholderText("Знайти...");

```

```

}

```

```

void CustomTable::initModel()

```

```

{
    if(m_table!=nullptr)
        delete m_table;
    m_table=new QStandardItemModel(0,5);
    m_table->setHorizontalHeaderItem(0,new QStandardItem("Канал"));
    m_table->setHorizontalHeaderItem(1,new QStandardItem("ID Упр"));
    m_table->setHorizontalHeaderItem(2,new QStandardItem("Тип"));
    m_table->setHorizontalHeaderItem(3,new QStandardItem("Данні"));
    m_table->setHorizontalHeaderItem(4,new QStandardItem("HEX"));
}

```

```

void CustomTable::adjustToView()

```

```

{
    resizeColumnToContents(0);
    resizeColumnToContents(1);
    resizeColumnToContents(2);
    resizeColumnToContents(4);
    setColumnWidth(3,width()-columnWidth(0)-columnWidth(1)-
columnWidth(2)-columnWidth(4)-50);
}

```

B.6 TargetTable.h

```

#ifndef TARGETTABLE_H
#define TARGETTABLE_H

#include <QLine>
#include <QWidget>
#include <QMainWindow>
#include <QLayout>
#include <QSpinBox>
#include <QTreeView>
#include <QSplitter>
#include "Parser/regParser.h"
#include "Common/ReaderAlgorithms.h"
#include "windows/TargetTable/TargetTreeWidget.h"
#include "windows/TargetViewer/Target3DWindow.h"
#include "windows/TargetTable/slider/TimeSlider.h"
QT_BEGIN_NAMESPACE
namespace Ui {class TargetTable;}
QT_END_NAMESPACE

class TargetTable : public QMainWindow
{
    Q_OBJECT

public:
    explicit TargetTable(QWidget *parent = nullptr);
    ~TargetTable();

```

```

void setCarState(SCdgrStateInside &state);
void update();
signals:
void newFileDropped(QString);

private:
void initConnections();
void initUI();

void selectFile();
void updateSlider();
void setLeftIndex();
void setRightIndex();

void parce(QString path);
void dropEvent(QDropEvent *event) override;
void dragEnterEvent(QDragEnterEvent *event) override;
void closeEvent(QCloseEvent *event) override;
private:
int prevL=0,prevR=0;
RangeSlider* m_slider;
Ui::TargetTable *ui;
QSplitter *splitter;
bool sliderActive=false;

regParser  m_parser;

CdgrVector codograms;           //массив кодограм

CdgrVectorPtr<UCdgrDIO> Dio; //канал РПД-BA18

```

```

CdgrVectorPtr<UCdgrPs01> io; //канал РПД-ПОИ
CdgrAdmTimeMap DioAdm;      //канал управления
CdgrVectorPtr<UCdgrInside> inside; //внутренний канал
CdgrTargetMap targets;      //цели

QString lastPath=".";        //последний открытый каталог

};

```

```

#endif // TARGETTABLE_H

```

B.7 TargetTable.cpp

```

#include "TargetTable.h"
#include "ui_TargetTable.h"
TargetTable::TargetTable(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::TargetTable)
{

    initUI();

    //подтягиваем последний открытый путь
    QSettings settings("settings.ini",QSettings::IniFormat);
    settings.beginGroup("lastPath");
    lastPath=settings.value("Path").value<QString>();

```

```
settings.endGroup();
```

```
initConnections();
```

```
}
```

```
TargetTable::~~TargetTable()
```

```
{
```

```
delete ui;
```

```
}
```

```
void TargetTable::setCarState(SCdgrStateInside &state)
```

```
{
```

```
SCdgrStateMsg *msg=&state.msg;
```

```
double lon=CAngle::fromMSecs(msg->longitude).toDegU(),
```

```
lat=CAngle::fromMSecs(msg->latitude).toDegU(),
```

```
carRot=CAngle::fromMSecs(msg->OrientationAz).toDegU(),
```

```
h=msg->altitude,
```

```
antRot=CAngle::fromMSecs(msg->antAzimuth).toDegU(),
```

```
antL=antRot-30,
```

```
antR=antRot+30;
```

```
ui->viewer->setCarPose(lat,lon,h,carRot,antL,antR);
```

```
}
```

```
void TargetTable::update()
```

```
{
```

```
ui->targetTree->update();
```

```
ui->viewer->clearSelection();
```

```
ui->targetCountL->setText("Кількість об'єктів: "+QString::number(ui->targetTree->update()));
```

```

int cnt=0;
for(auto trgt:targets)
    cnt+=trgt.Points.length();
ui->plotCountL->setText("Кількість точок:"+QString::number(cnt));
updateSlider();
}

void TargetTable::initConnections()
{
    connect(ui->vaChk,    &QCheckBox::clicked,    [=](bool    show){ui-
>targetTree->showType(ECdgrTypes::Dio,show);});
    connect(ui->obnChk,    &QCheckBox::clicked,    [=](bool    show){ui-
>targetTree->showType(ECdgrTypes::obn,show);});
    connect(ui->soprChk,    &QCheckBox::clicked,    [=](bool    show){ui-
>targetTree->showType(ECdgrTypes::sopr,show);});
    connect(ui->range,      &QCheckBox::clicked,[=](bool      apply){ui-
>targetTree->applyRange(apply,ui->pCount->value());});
    connect(ui->pCount,
QOverload<int>::of(&QSpinBox::valueChanged),[=](int      val){ui->targetTree-
>applyRange(true,val);});
    connect(ui->openFile,
&QAction::triggered,this,&TargetTable::selectFile);

    connect(ui->viewer,      &Target3DWindow::selectedIndex,ui-
>targetTree,&TargetTreeWidget::selectPoint);
    connect(ui->viewer,    &Target3DWindow::clearColor,    ui->targetTree,
&TargetTreeWidget::setDefaultColors);
    connect(ui->viewer,    &Target3DWindow::DioColor,    [=](GLColor
color){ui->targetTree->setDIODefaultColor(color);});
    connect(ui->viewer,    &Target3DWindow::OBNColor,    [=](GLColor

```

```

color){ui->targetTree->setIODefaultColor(cdgr::uiPs01TypeObn,color);});
    connect(ui->viewer, &Target3DWindow::SOPRColor, [=](GLColor
color){ui->targetTree->setIODefaultColor(cdgr::uiPs01TypeSopr,color);});

    connect(ui->targetTree,&TargetTreeWidget::selectedIndex,ui-
>viewer,&Target3DWindow::selectIndex);
    //          connect(ui->viewer,&Target3DWindow::startStopTimer,ui-
>timeSlider,&TimeSlider::startStopScroll);

connect(m_slider,&RangeSlider::lowerValueChanged,this,&TargetTable::setLeftI
ndex);

connect(m_slider,&RangeSlider::upperValueChanged,this,&TargetTable::setRight
Index);
}

void TargetTable::initUI()
{
    ui->setupUi(this);
    // ui->viewMenu->setVisible(false);
    m_slider=ui->timeSlider->slider;
    setWindowTitle("Переглядач реєстрації");

    ui->pCount->setVisible(false);
    // ui->rangeL->setVisible(false);

    splitter=new QSplitter(this);
    splitter->addWidget(ui->treeFrame);
    splitter->addWidget(ui->glFrame);
    splitter->setSizes({270,830});

```

```

splitter->setHandleWidth(6);
splitter->setContentsMargins(4,2,4,2);
splitter->setStyleSheet("QSplitter::handle {background-color:rgb(170,
171, 173); margin:3px; border-radius:3px;}");
setCentralWidget(splitter);
ui->targetTree->adjustColumns();
ui->targetTree->setTargets(&targets);
ui->viewer->setTargetArray(&targets);
updateSlider();
}

```

```

void TargetTable::selectFile()
{
    QFileDialog dialog;
    QString fileName=dialog.openFileName(this,"Виберіть
файл..",lastPath,"Reg(*.reg)");
    if(!fileName.isValid())
        return;
    lastPath=QFileInfo(fileName.toLocalFile()).absolutePath();
    parce(fileName.toLocalFile());
}

```

```

void TargetTable::updateSlider()
{
    if(targets.isEmpty())
        return;
    uint32_t time1=targets.begin()->Points.begin()->cod->time,
        time2=targets.begin()->Points.begin()->cod->time;
    sliderActive=false;
}

```

```

//устанавливаем границы слайдера...
for(auto &trgt:targets)
{
    for(auto &point:trgt.Points)
    {
        if(time1==(uint32_t)-1 || point.cod->time<time1)
            time1=point.cod->time;
        if(time2==(uint32_t)-1 || point.cod->time>time2)
            time2=point.cod->time;
    }
}
prevL=time1;
prevR=time2;
m_slider->SetRange(time1,time2);
m_slider->setMinimum(time1);
m_slider->setMaximum(time2);
sliderActive=true;

}

void TargetTable::setLeftIndex()
{
    if(!sliderActive)
        return;
    if(m_slider->GetMinimun()==m_slider->GetLowerValue())
    {
        for(auto &trgt:targets)
            trgt.startTimeIndex=0;
        prevL=m_slider->GetLowerValue();
        return;
    }
}

```

```

    }
    uint32_t value=m_slider->GetLowerValue();
    if(prevL<m_slider->GetLowerValue())
    {
        for(auto &trgt:targets)
            while(trgt.startTimeIndex<trgt.endTimeIndex      &&
(trgt.Points[trgt.startTimeIndex].cod->time==(uint32_t)-1    ||
trgt.Points[trgt.startTimeIndex].cod->time<=value))
                trgt.startTimeIndex++;
    }
    else
    {
        for(auto &trgt:targets)
            while(trgt.startTimeIndex>0      &&
(trgt.Points[trgt.startTimeIndex].cod->time==(uint32_t)-1    ||
trgt.Points[trgt.startTimeIndex].cod->time>=value))
                trgt.startTimeIndex--;
    }
    prevL=m_slider->GetLowerValue();
}

void TargetTable::setRightIndex()
{
    if(!sliderActive)
        return;
    if(m_slider->GetMaximun()==m_slider->GetUpperValue())
    {
        for(auto &trgt:targets)
            trgt.endTimeIndex=trgt.Points.length()-1;
        prevR=m_slider->GetUpperValue();
    }
}

```

```

        return;
    }
    uint32_t value=m_slider->GetUpperValue();
    if(prevR<m_slider->GetUpperValue())
    {
        for(auto &trgt:targets)
            while(trgt.endTimeIndex<trgt.Points.length()-1      &&
(trgt.Points[trgt.endTimeIndex].cod->time==(uint32_t)-1      ||
trgt.Points[trgt.endTimeIndex].cod->time<=value))
                trgt.endTimeIndex++;
    }
    else
    {
        for(auto &trgt:targets)
            while(trgt.endTimeIndex>trgt.startTimeIndex      &&
(trgt.Points[trgt.endTimeIndex].cod->time==(uint32_t)-1      ||
trgt.Points[trgt.endTimeIndex].cod->time>=value))
                trgt.endTimeIndex--;
    }
    prevR=m_slider->GetUpperValue();
}

void TargetTable::parce(QString path)
{
    ui->vaChk->setCheckState(Qt::Checked);
    ui->obnChk->setCheckState(Qt::Checked);
    ui->soprChk->setCheckState(Qt::Checked);
    setWindowTitle("Файл: "+path.split('/').last());
    codograms.clear();
    Dio.clear();
}

```

```

targets.clear();
io.clear();
DioAdm.clear();
inside.clear();

```

```

qDebug()<<"clear done";
m_parser.parse(codograms,path);
qDebug()<<"parce done";
GLTargetArraySettings settings=ui->viewer->getTargetSettings();
organize(codograms,
        &dio,&io,&dioAdm,&inside,
        &targets,

```

```

settings.defaultDioColor,settings.defaultOBNColor,settings.defaultSOPRColor);

```

```

qDebug()<<"org done";
if(inside.begin()->cdgr->header.type==cdgr::uiInsideTypeState)
    setCarState(inside.begin()->cdgr->state);
qDebug()<<"car done";

```

```

// ui->tableView->showCodograms(codograms);
// qDebug()<<"tv done";

```

```

update();
qDebug()<<"3d done\n-----\n";
}

```

```

void TargetTable::dropEvent(QDropEvent *event)
{
    for(auto url:event->mimeTypeData()->urls())
    {

```

```

        emit newFileDropped(url.toLocalFile());
        parce(url.toLocalFile());
    }
    event->acceptProposedAction();
}

void TargetTable::dragEnterEvent(QDragEnterEvent *event)
{
    if(event->mimeTypeData()->hasUrls() && event->mimeTypeData()-
>urls().last().fileName().split(".").last()=="reg")
        event->acceptProposedAction();
}

void TargetTable::closeEvent(QCloseEvent *event)
{
    QApplication::closeAllWindows();
    QSettings settings("settings.ini",QSettings::IniFormat);
    settings.beginGroup("lastPath");
    settings.setValue("Path",lastPath.isEmpty()? "":"/":lastPath);
    settings.endGroup();
    QMainWindow::closeEvent(event);
}

```

ДОДАТОК Г

Слайди презентації



Рисунок Г.1- Слайд №1



Рисунок Г.2- Слайд №2



Рисунок Г.3- Слайд №3

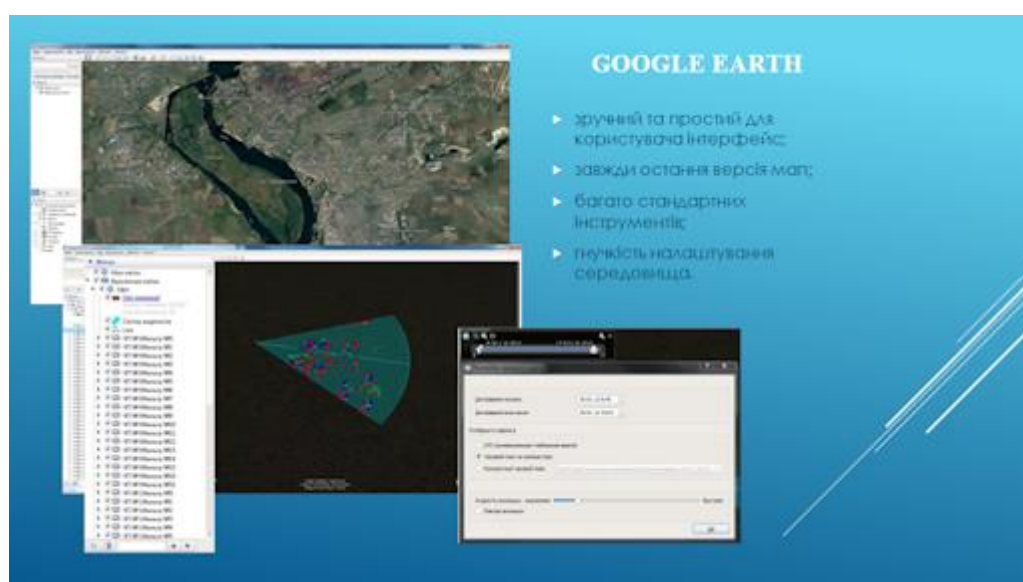


Рисунок Г.4- Слайд №4

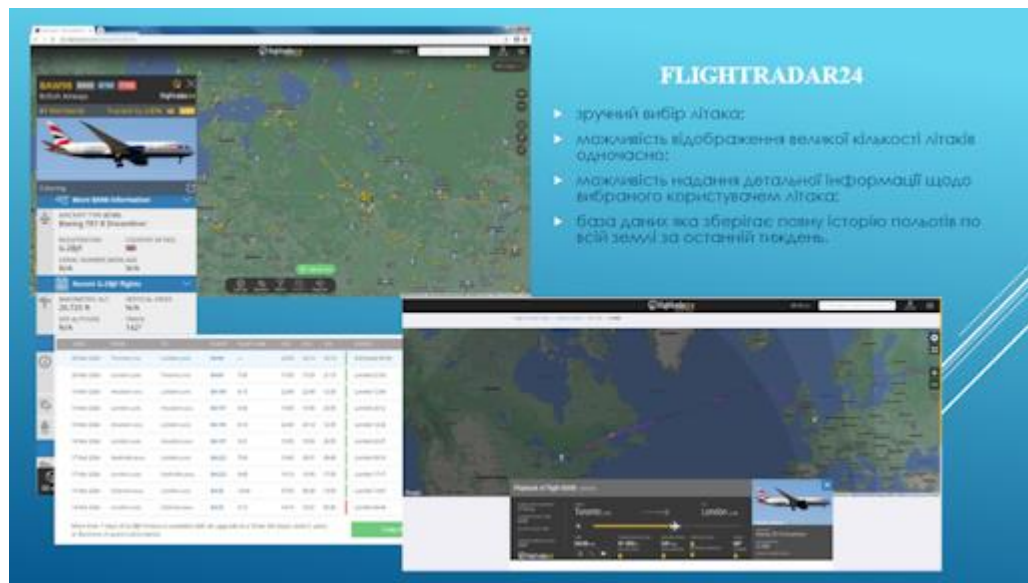


Рисунок Г.5- Слайд №5

ПОРІВНЯННЯ		
Критерії порівняння	Google Earth	Flight Radar 24
Доступність	Є повністю платною	Є платні послуги
Інформативність	Користувач сам вибирає інформацію для відображення	Середня
Зручність інтерфейсу	Дуже зручний	Середня
Додатковий інструментарій	Є	Немає
Підтримка формату реєстрації	Так, але з перетворенням у внутрішній формат	Не має можливості завантажувати файли
Оптимізація	Погана	Висока
Можливість роботи офлайн	Є	Немає

Рисунок Г.6- Слайд №6

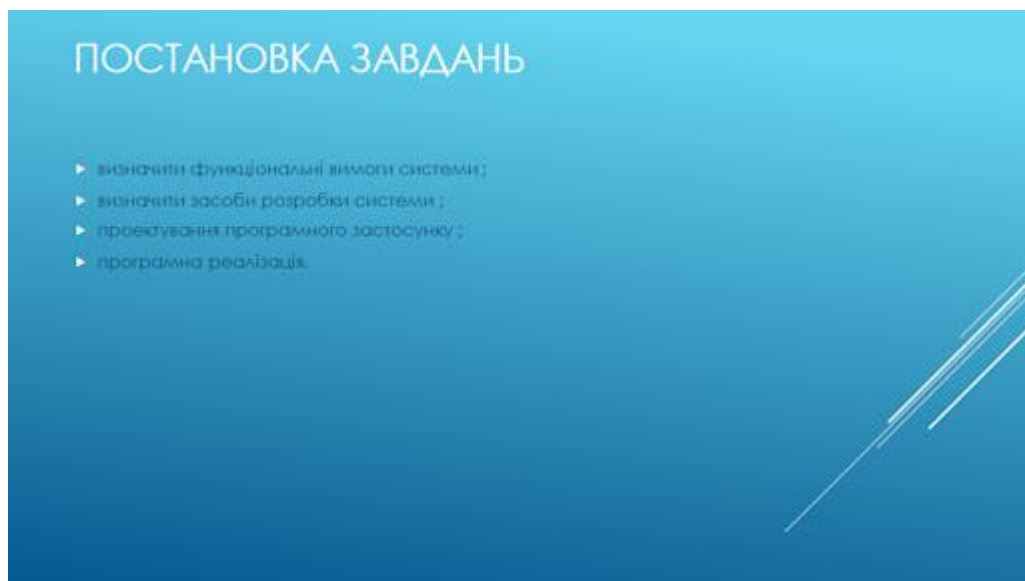


Рисунок Г.7- Слайд №7

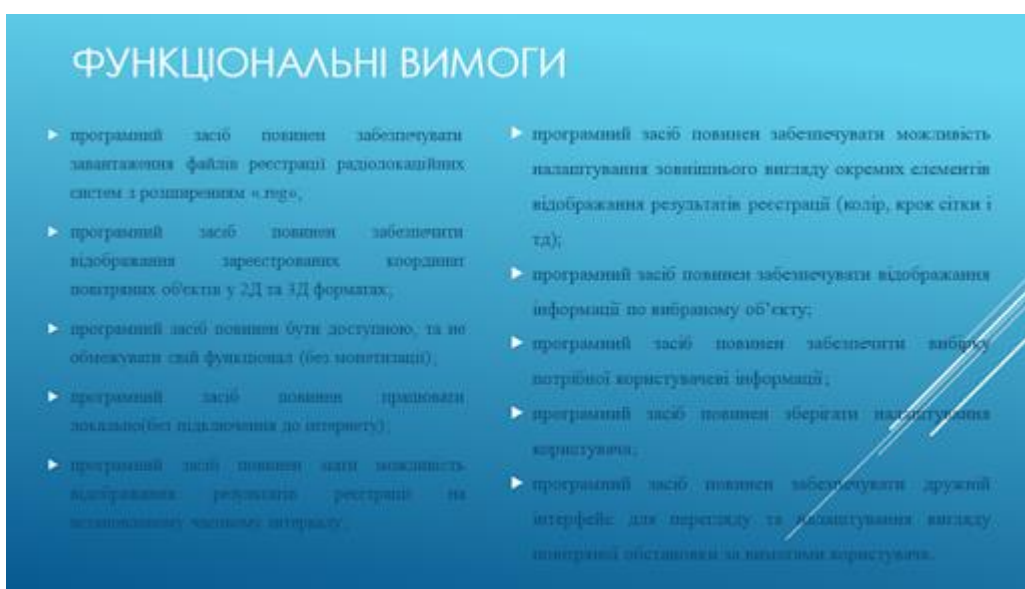


Рисунок Г.8- Слайд №8

ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерії порівняння	C++	Python	C#	Java
Швидкість	Висока	Низька	Висока	Середня
Наявність потрібних бібліотек	Так	Частково	Так	Так
Простота написання коду	Частково	Дуже просто	Частково	Складно
Зручна документація	На рівні IDE	Частково	Зручна	Не зручно
Крос-платформність	Так	Так	Так	Так

Рисунок Г.9- Слайд №9

ВИБІР СЕРЕДИ РОЗРОБКИ

	Qt Creator	Visual Studio	Visual Code	CLion
Рефакторинг	+	+	++	++
Пошук по проєкту	+	++	+	+
Служба підтримки	++	+	+	+
Українська мова	++	+	+	-
Підсвічування синтаксису	++	++	++	++

	Qt Creator	Visual Studio	Visual Code	CLion
Налагодження	++	+	+	+
Тестування програм	++	++	+	++
Мови, які підтримуються	C++, QML, Python	C++, C#	Будь-яка через доповнення	C++
Доступність	Безкоштовна з обмеженнями	Безкоштовна з обмеженнями	Безкоштовна	Повністю платна

Рисунок Г.10- Слайд №10

ВИБІР ФРЕЙМВОРКУ РОБОТИ З 3Д

Критерій	OpenGL	Vulkan	DirectX
Простота використання	++	-	+
Функціональність	+	++	+
Швидкодія	+	++	+
Спільнота	+	+	++
Документація	+	-	++
Кросплатформеність	+	-	-

Рисунок Г.11- Слайд №11

ДІАГРАМА КЛАСІВ РОБОТИ З OPENGL

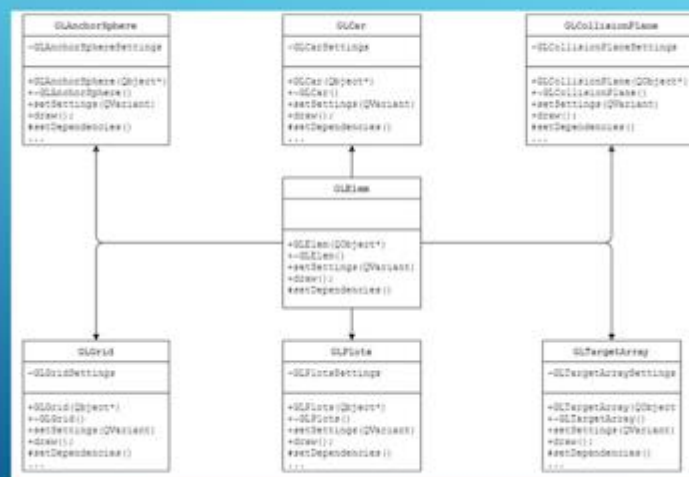


Рисунок Г.12- Слайд №12

