КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою "Інформатика"
спеціальності 122 Комп'ютерні науки на тему:

ДОПОВНЕНА РЕАЛЬНІСТЬ ДЛЯ БПЛА

Виконав студент 4-го курсу
Затилюк Андрій Олександрович _____
(підпис)

Науковий керівник:
Доктор фізико-математичних наук, професор
Терещенко Василь Миколайович _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи: 45 сторінок, 29 ілюстрацій, 0 таблиць, 34 використаних джерел.

Ключові слова: БПЛА, ВІДЕО, ДОПОВНЕНА РЕАЛЬНІСТЬ, ЗМІНИ В РЕАЛЬНОМУ ЧАСІ, МАВЛІНК, БАЛІСТИЧНИЙ КАЛЬКУЛЯТОР.

Об’єктом роботи є програма для моніторингу відео з БПЛА в реальному часі з накладанням польотної інформації та ландшафту на відео.

Метою кваліфікаційної роботи є створення програми для покращення та полегшення роботи оператора під час польоту БПЛА «Довбуш».

Інструментом створення є безкоштовний, вільно поширюваний редактор коду `Visual Studio 2022`, мова програмування `C++`. Використано бібліотеки `OpenGL`, `Mavlink`, `FFmpeg`, а також `Winapi`.

Результат роботи: розроблено та використовується операторами БПЛА «Довбуш», проте є власністю ТОВ «Рідне Небо».

ЗМІСТ

ВСТУП 4

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ 6

РОЗДІЛ 1. ОГЛЯД ДОПОВНЕНОЇ РЕАЛЬНОСТІ ДЛЯ БПЛА 7

1.1. Означення та властивості 7

1.2. Технологічні аспекти 8

1.3. Виклики та обмеження 10

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ СИСТЕМ КЕРУВАННЯ БПЛА 11

2.1. Mission Planner 11

2.2. QBase 3D 14

РОЗДІЛ 3. РЕАЛІЗАЦІЯ 17

3.1. Початковий план 17

3.2. Новий інтерфейс 21

3.3. Робота з відео 23

3.4. Робота з Mavlink 26

3.5. Інтеграція OpenGL 28

3.6. OSD пілота 29

3.7. Terrain 37

3.8. Реалізація точок інтересу 40

РОЗДІЛ 4. ВИСНОВКИ 43

РОЗДІЛ 5. ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 45

ВСТУП

В наш час використання дронів та БПЛА набирає оберти. Саме тому застосування доповненої реальності для них стає все більш важливою. Вже зараз існує велика кількість систем та протоколів, які надають віртуальну та доповнену реальність для різних застосувань, але використоання цих технологій для БПЛА лише починає набувати обертів.

Актуальність даної роботи полягає в розробці системи доповненої реальності для БПЛА, заснованої на OpenGL, FFmpeg, Winapi C++, Mavlink та використанні SRTM1, що забезпечить високий рівень візуалізації та інтерактивності для користувачів. Робота виконана на мові програмування C++.

Основна мета роботи полягає в аналізі потреб операторів БПЛА та створенні нового додатку, що здатний накладати ландшафт та польотні дані на відео.

Об’єктом дослідження є системи доповненої реальності та їх застосування в БПЛА. У процесі дослідження будуть використані теоретичні та практичні методи, включаючи аналіз наукової літератури, програмування, симуляції та реалізацію прототипів.

Можливі сфери застосування результатів роботи включають аерофотозйомку, сільське господарство, логістику, служби доставки та інші області, де використовуються БПЛА. Основна сфера використання - військова справа.

Дана робота базується на реалізації інноваційних технологій в області доповненої реальності, зокрема, використання OpenGL для рендерингу графіки, FFmpeg для обробки відео, Winapi C++ для інтерфейсу користувача, Mavlink для комунікації з БПЛА, та SRTM1 для візуалізації ландшафту. Основні принципи та ідеї цих технологій були адаптовані та інтегровані для створення цієї унікальної системи.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

БПЛА Безпілотний літальний аппарат;

SRTM1 Shuttle Radar Topography Mission 1 Arc-Second;

AR Доповнена реальність (Augmented reality);

ПЗ Програмне забезпечення;

GPS Global Positioning System;

RTSP Real Time Streaming Protocol;

RTMP Real Time Messaging Protocol;

OSD On-Screen Display

РОЗДІЛ 1. ОГЛЯД ДОПОВНЕНОЇ РЕАЛЬНОСТІ ДЛЯ БПЛА

1.1. Означення та властивості

Доповнена реальність для БПЛА включає в себе взаємодію між двома складовими: безпілотним літальним апаратом та системою доповненої реальності. Система доповненої реальності надає візуальний контент, який «доповнює» реальний світ, видимий через камеру БПЛА. Важливими характеристиками системи доповненої реальності для БПЛА є:

1. **Інтерактивність** (Interactivity): Система AR відповідає на дії користувача або зміни в оточуючому середовищі в реальному часі.
2. **Реалістичність** (Realism): Візуальний контент, який надає система AR, відображає реальний світ з достатньою деталізацією та точністю.
3. **Сумісність з реальністю** (Reality Alignment): Візуальний контент, який надає система AR, правильно вирівнюється та інтегрується з реальним світом.

1.2. Технологічні аспекти

В контексті розробки доповненої реальності для БПЛА, певні технології та бібліотеки будуть використовуватися для реалізації деяких фукнкціональних можливостей. Крім того, при правильному виборі цих елементів, додавання нових можливостей не буде проблемою.

1. **FFMpeg**: Це безкоштовна та відкрита бібліотека, яка включає в себе багато утиліт та бібліотек для обробки мультимедійних даних, включаючи аудіо, відео та потоки. В контексті БПЛА, FFmpeg може бути використаний для отримання, кодування, декодування та передачі відеопотоків в реальному часі. Це може включати збір відео з камери БПЛА, його передачу до контрольної станції на землі, а також інтеграцію доповненої реальності з відеопотоком. Крім того, вона дає змогу налаштувати затримку відео так, як потрібно користувачу.
2. **Windows API** (WinAPI): Набір інтерфейсів програмування застосунків (API), що надається Microsoft для використання в операційних системах Windows. WinAPI може бути використаний для створення графічного інтерфейсу користувача, обробки вводу, роботи з мережами, керування потоками даних та ін.
3. **MAVLink** (Micro Air Vehicle Link): Протокол для обміну інформацією між БПЛА та контрольною станцією. Це включає в себе команди для керування БПЛА, дані телеметрії, інформацію

про стан БПЛА та ін. За допомогою MAVLink, можна забезпечити двосторонню взаємодію між системою доповненої реальності та БПЛА.

4. **OpenGL** (Open Graphics Library): Специфікація для рендерингу 2D та 3D графіки. У контексті доповненої реальності для БПЛА, OpenGL може бути використаний для створення візуальних ефектів доповненої реальності, які накладаються на реальний відеопотік від БПЛА. Це може включати рендеринг 3D-моделей, текстур, освітлення, тіней та ін.
5. **STRM**: Це міжнародний проект, що був спільно реалізований Національним управлінням з аеронавтики і космонавтики США (NASA) і Національною геоспатіальною розвідувальною службою (NGA), з метою отримання високодеталізованих тривимірних карт земної поверхні. SRTM використовує радарні дані для створення точних висотних карт земної поверхні. В контексті доповненої реальності для БПЛА, ці дані можуть бути використані для відтворення реального ландшафту в 3D.

1.3. Виклики та обмеження

Використання доповненої реальності в додатках для керування БПЛА представляє певний виклик та обмеження, що вимагають особливої уваги при розробці та використанні.

1. **Обробка даних та обчислювальні ресурси**: Однією з основних проблем може бути потреба в великій обчислювальній потужності. FFMPEG та OpenGL, наприклад, вимагають певної обчислювальної потужності для обробки відео в реальному часі та рендерингу 3D графіки. Крім того, обробка та інтеграція SRTM теж може бути вимогливою до ресурсів.
2. **Інтерфейс користування**: Сворення інтуїтивного та ефективного інтерфейсу користувача з використанням WinAPI може бути викликом. Важливо, щоб користувач міг легко інтерпретувати інформацію доповненої реальності та вчасно реагувати на неї.
3. **Точність відстеження та моделювання**: Підтримка реалістичної доповненої реальності вимагає точного відстеження положення та орієнтації БПЛА, а також точного моделювання реального світу за допомогою даних SRTM. Будь-яка невідповідність може спричинити розрив між реальним і віртуальними світами, що може призвести до поганих результатів або навіть помилок у навігації.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ СИСТЕМ КЕРУВАННЯ БПЛА

2.1. Mission Planner

Mission Planner - це комплексне рішення Ground Control Station (GCS), яке використовується для керування безпілотниками, що використовують ArduPilot flight software. Це програмне забезпечення з відкритим кодом, що розробляється спільнотою ArduPilot, і має ряд ключових властивостей, які роблять його цінним для дипломної роботи, особливо якщо розглядати його можливості в контексті доповненої реальності для БПЛА (див. Рисунок 1).

Рисунок 1: Mission Planner

Однією з головних особливостей Mission Planner є його здатність планувати місії. Це включає можливість програмування точок контролю, налаштування автоматичних дій при досягненні кожної точки, та керування різними режимами політу. Це може бути корисно для реалізації різних аспектів доповненої реальності в

БПЛА, наприклад, для відображення маршрутів та точок контролю в AR-інтерфейсі (див. Рисунок 2).

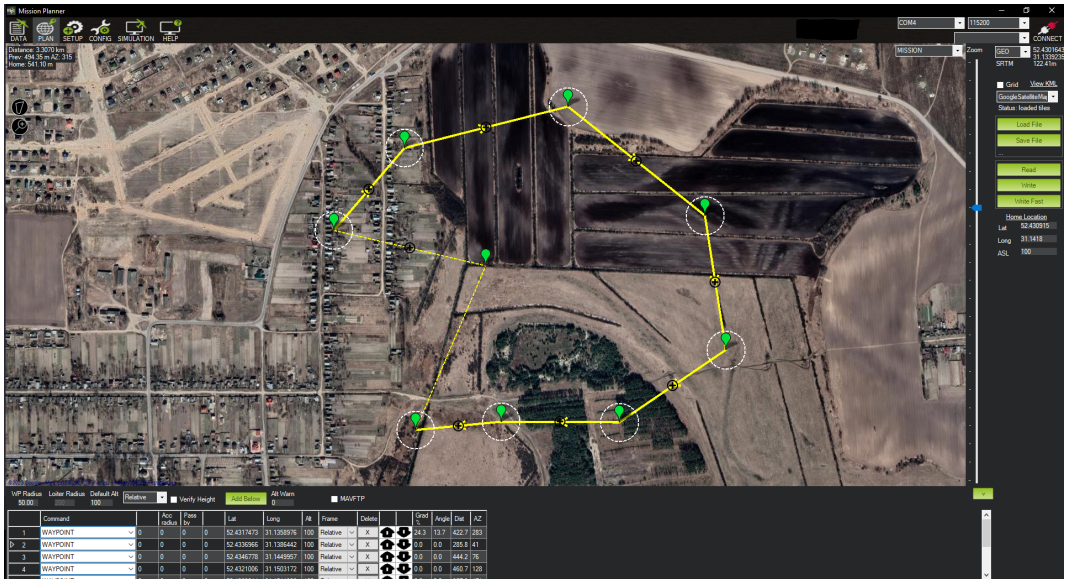

Рисунок 2: Побудова плану польоту

Крім того, Mission Planner надає розширені можливості телеметрії, дозволяючи користувачам відстежувати і відображати в реальному часі детальну телеметричну інформацію від БПЛА. Це може бути використано для розробки більш інформативних та реалістичних AR-досвідів, дозволяючи користувачам бачити таку інформацію, як висоту політу, швидкість або стан батареї, накладену на відео з БПЛА (див Рисунок 3).

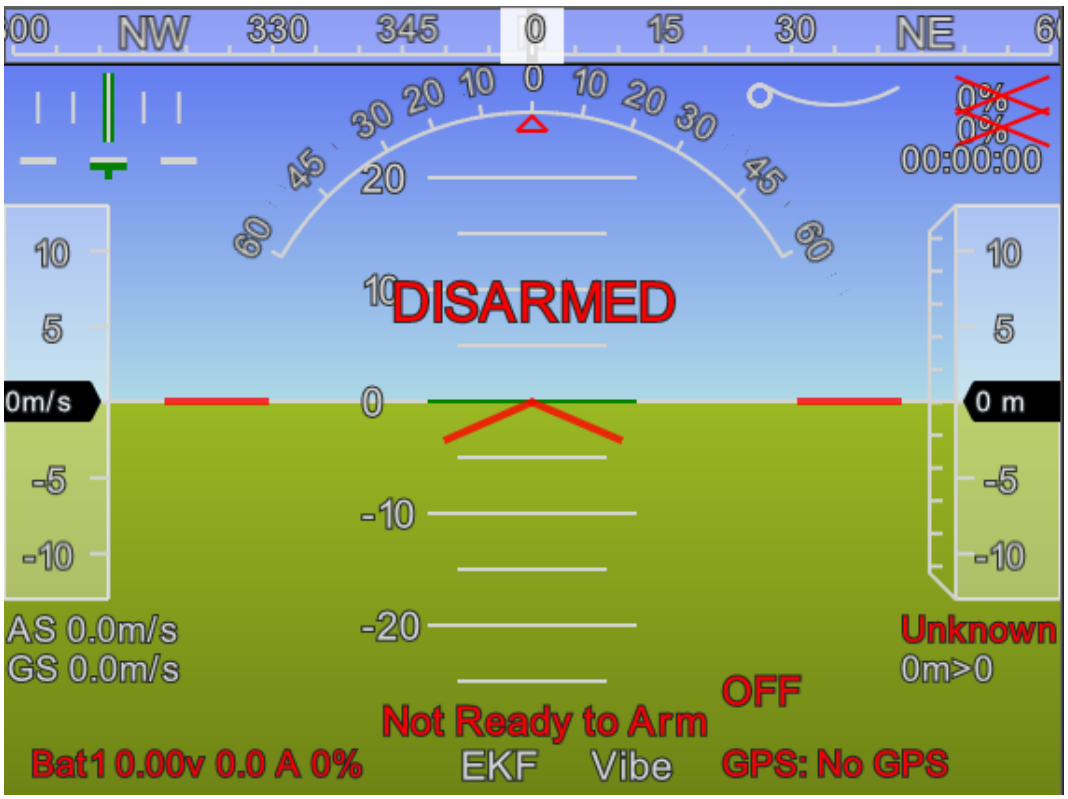

Рисунок 3: Польотний інтерфейс

Саме тому Mission Planner часто використовується в академічних дослідженнях і проектах, що включають БПЛА, включаючи роботи, що досліджують використання доповненої реальності для підвищення ефективності та безпеки політів БПЛА [1].

2.2. QBase 3D

QBase 3D - це ще одне важливе ПЗ для планування місій БПЛА, розроблене компанією Quantum-Systems. Це спеціалізований продукт, який призначений для роботи з тривимірними картами і планування маршрутів політу, враховуючи реальний ландшафт і перешкоди (див. Рисунок 4).

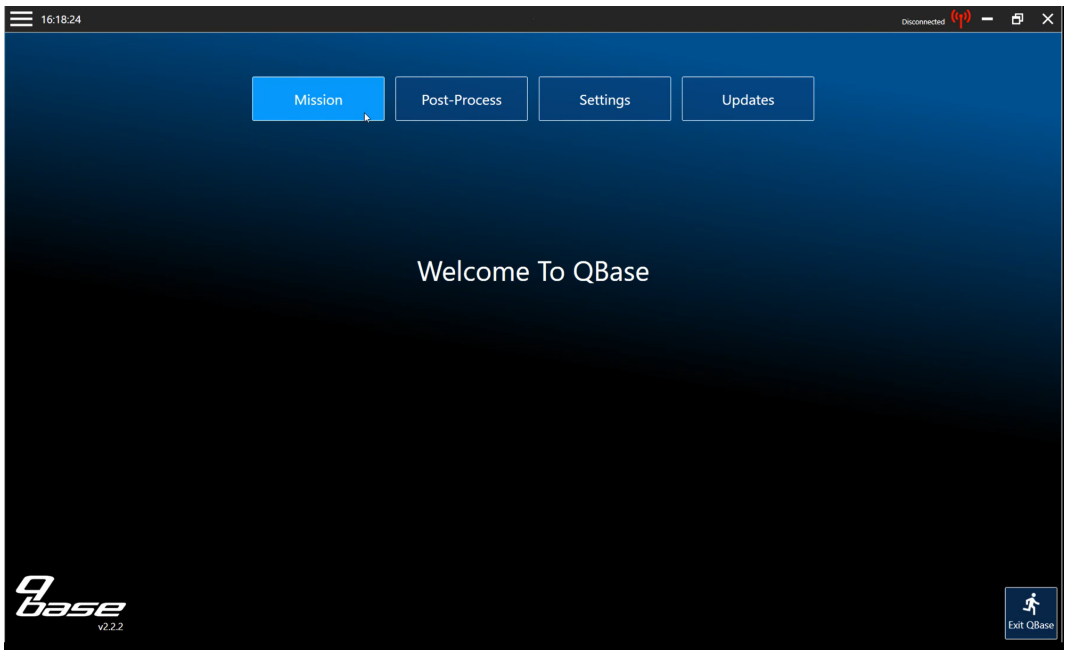

Рисунок 4: QBase 3D

Основною перевагою QBase 3D є його здатність використовувати детальні 3D-карти для планування місій. Це може бути особливо корисно для дипломного проекту, оскільки 3D-карти можуть бути використані для створення реалістичних AR-систем. Наприклад, маршрути та точки контролю можна відображати в тривимірному просторі, накладаючи їх на реальний світ (див. Рисунок 5).

Рисунок 5: Планування маршруту

QBase 3D також пропонує розширені можливості для планування політів для зйомки з висоти, включаючи налаштування, які враховують кут камери, перекриття знімків та резолюцію. Це може бути важливим для дослідження, оскільки ці налаштування впливають на якість та точність даних, які можуть бути використані для створення AR-візуалізацій (див. Рисунок 6).

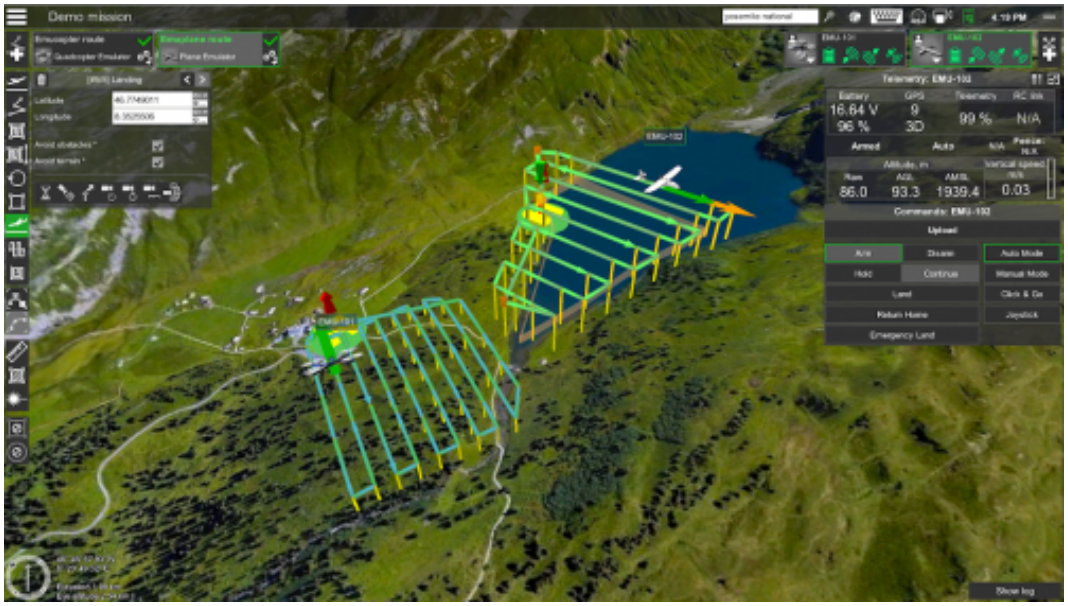

Рисунок 6: Планування польоту для зйомки

Нарешті, QBase 3D пропонує можливість автоматичного розрахунку часу польоту і витрат енергії, що може бути корисно для розробки AR-додатків, які допомагають користувачам краще розуміти та контролювати використання ресурсів БПЛА [2].

РОЗДІЛ 3. РЕАЛІЗАЦІЯ

3.1. Початковий план

Початковою задачею було створити ПЗ для військового призначення, а саме - програму для відтворення, запису та стрімінгу відео з БПЛА «Довбуш Т10», та можливстю підключень додаткових функцій, таких як накладання OSD пілота на відео, вибору типу підключення, можливість підключення до різних камер, моделювання ландшафту в реальному часі та накладання на нього точок інтересу - точки Home та місце прильоту снаряду.

Для цього було створено програму, яка має назву Interface. Реалізація представляла собою комбінацію WinApi, MavSDK, OpenCV та OpenGL на C++. Проте вона мала певні проблеми і непотрібні (як виявилося) функції. (див. Рисунок 7)

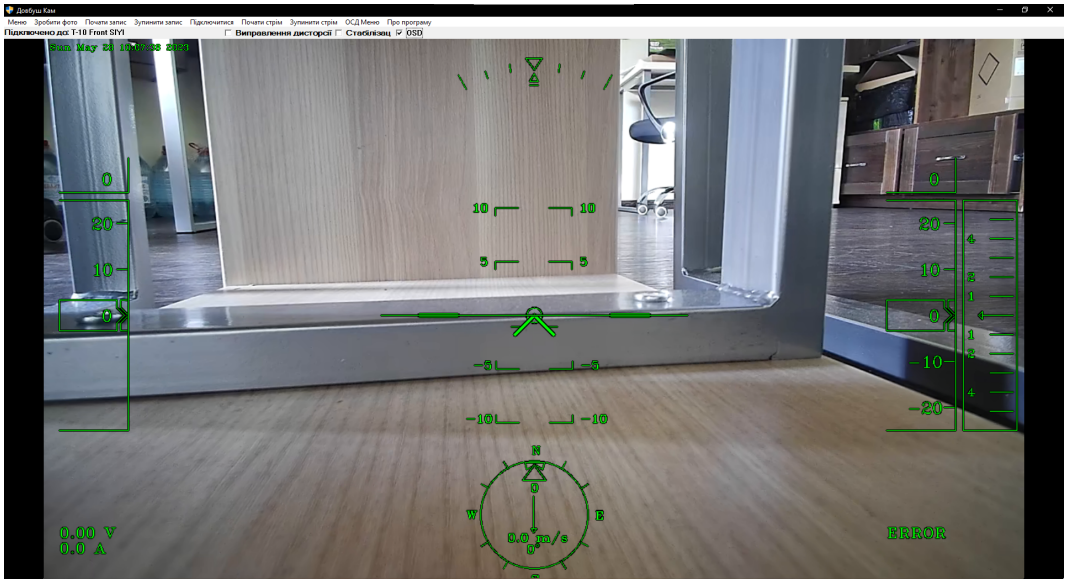

Рисунок 7: Довбуш Кам (Interface)

Перша проблема - складність додавання потрібних камер. В цій версії потрібно було зайти в налаштування, щоб побачити список всіх доступних камер (див. Рисунок 8). Відповідно через це вікно можна було додавати, змінювати, видаляти девайси та підключатися до них. Але, через велику кількість параметрів налаштування(див. Рисунок 9) користувачі плуталися. Тож було прийняте рішення замінити такий варіант на випадаюче меню з всіма доступними в наших БПЛА камерами.

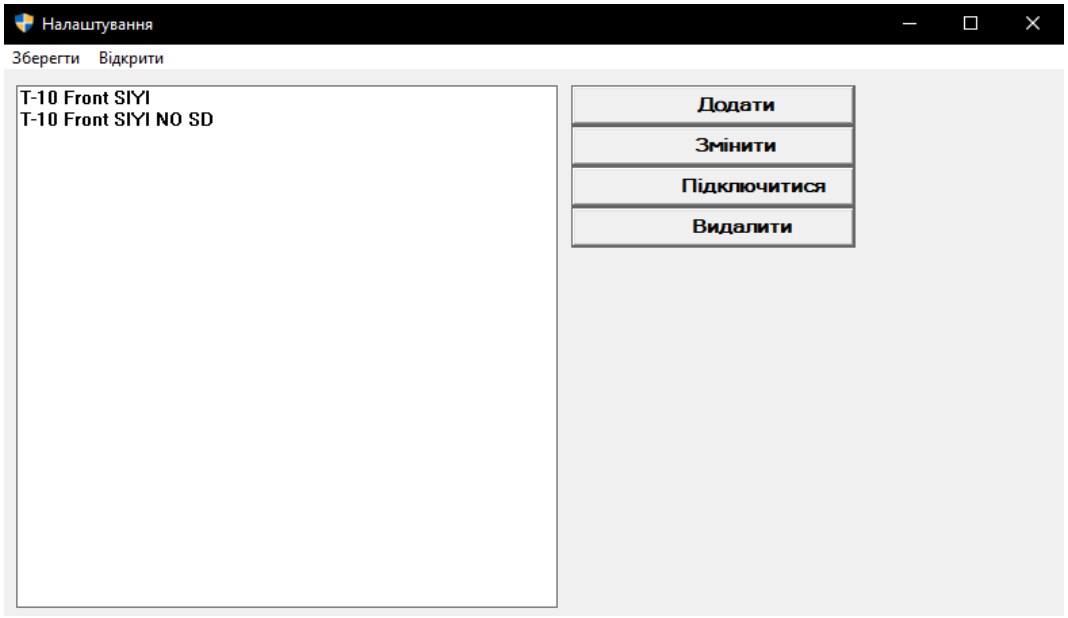

Рисунок 8: Список всіх камер

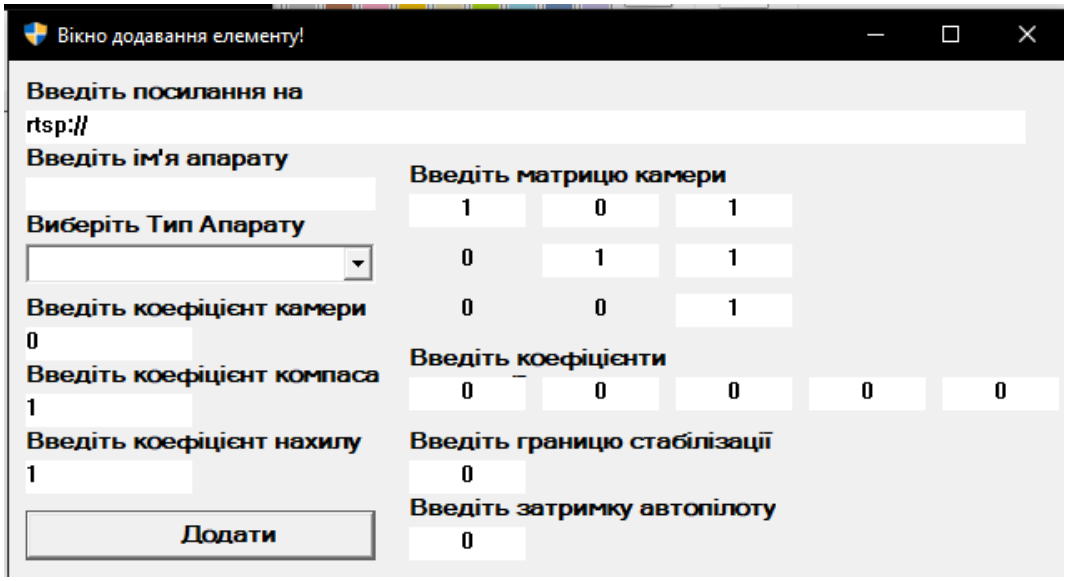

Рисунок 9: Вікно зміни/додавання камери

Наступна проблема - непотрібні функції. В початковому варіанті нашого БПЛА використовувалися камери з ефектом «риб'ячого ока» (див. Рисунок 10). Для виправлення використовувались можливості OpenCV (див. Рисунок 11), але виявилося, що для цього потрібні потужні комп'ютери, які не завжди доступні військовим в польових умовах. Тож було вирішено в майбутньому не використовувати такі камери. Крім цього, була функція програмної стабілізації відео, від якої відмовились на користь механічного варіанту.

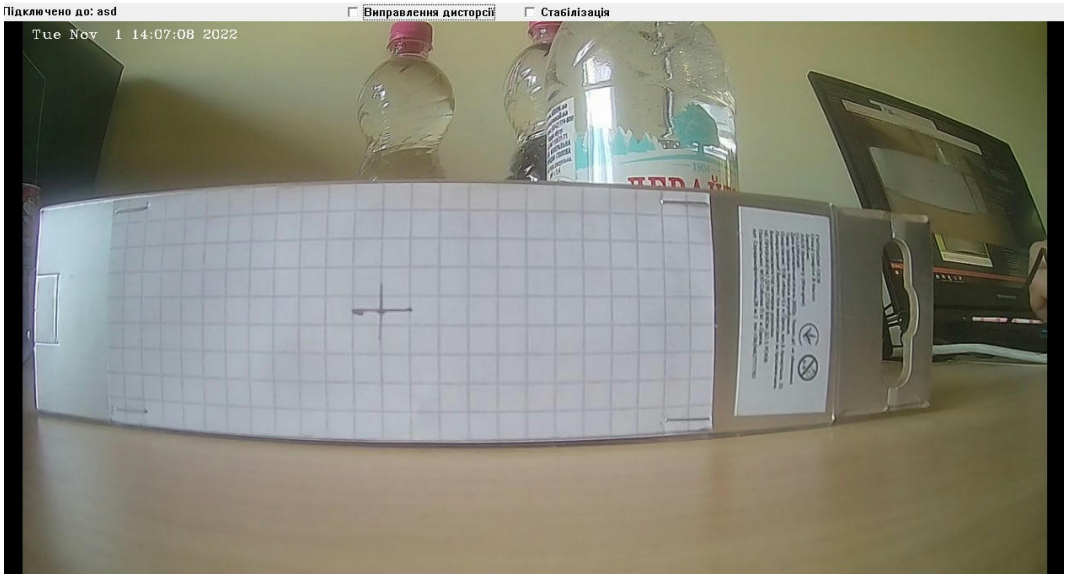

Рисунок 10: Відео з ефектом «риб’ячого ока»

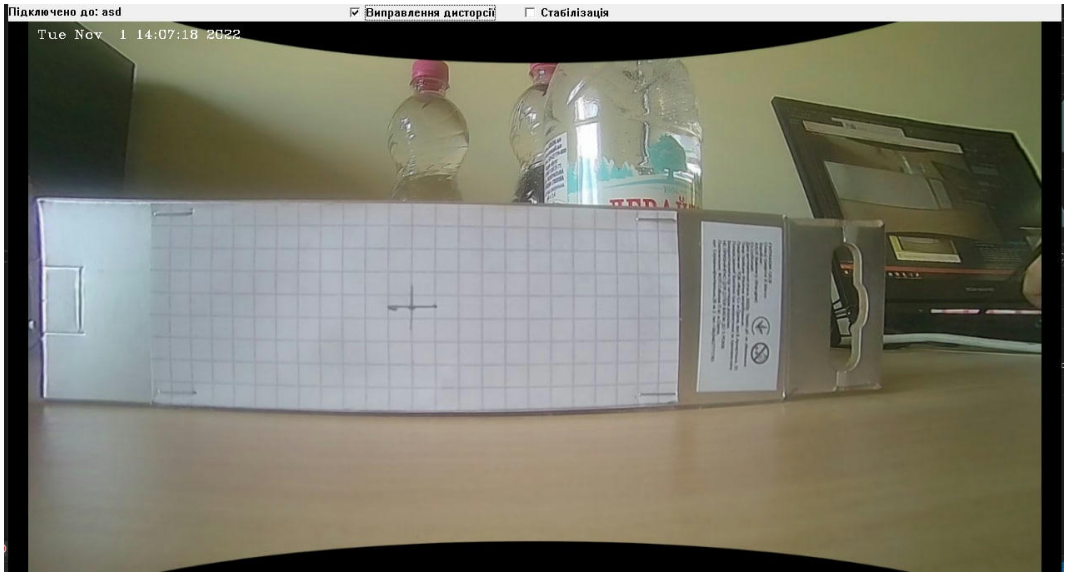

Рисунок 11: Виправлення «риб’ячого ока»

3.2. Новий інтерфейс

Новий інтерфейс було вирішино вистроїти максимально зрозуміло та зручно (див. Рисунок 12). Щоб користувачам було легше підключатися до камер, всі можливі варіанти було поміщено в випадаюче меню «Camera Select»(див. Рисунок 13). Там знаходятся кнопки підключення до всіх камер, які можливо встановити на «Довбуш». Також всі елементи взаємодії, такі як «Start Record» або «Make Picture» було винесено в великі кнопки.

Рисунок 12: Dovbush Cam

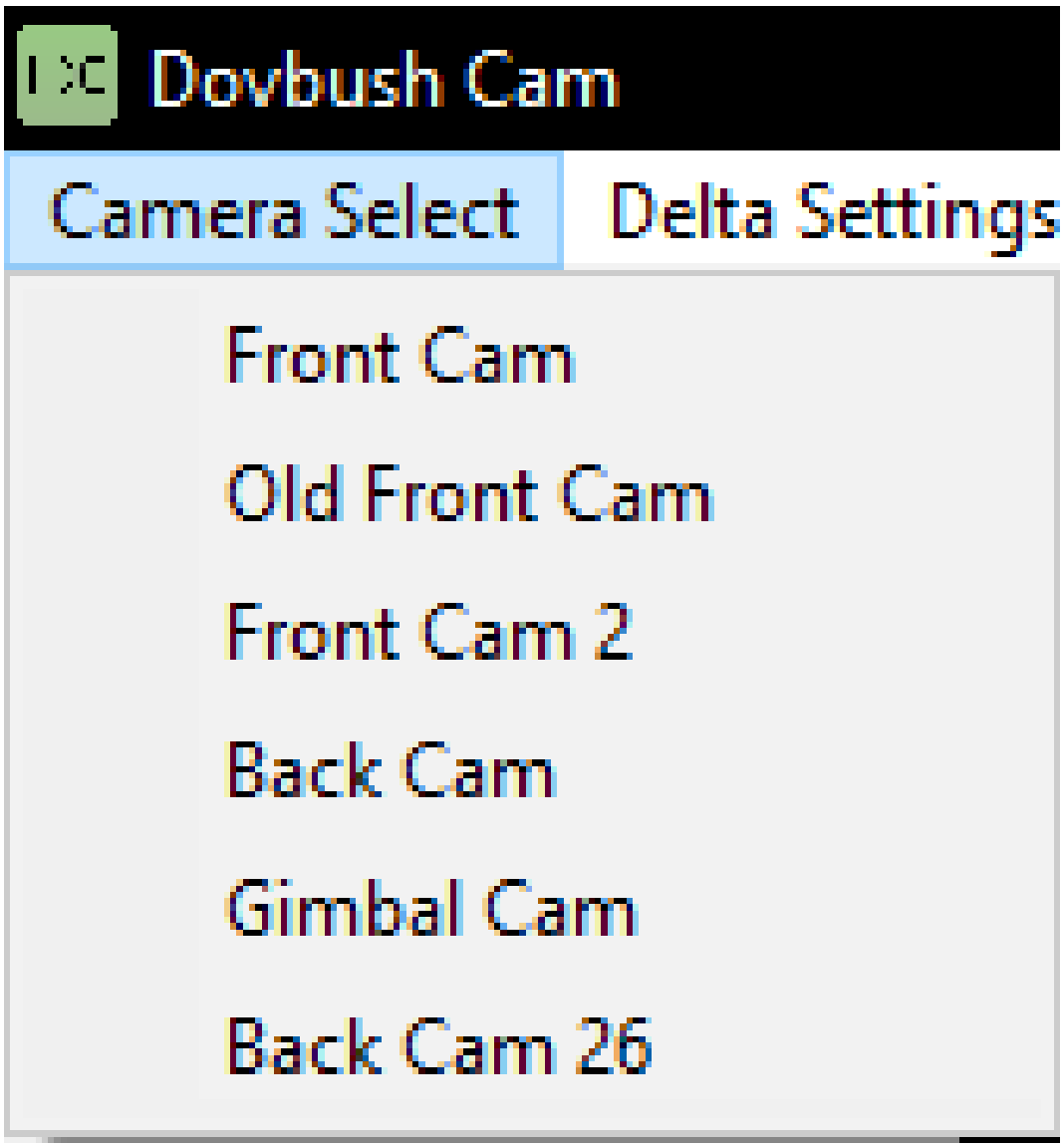

Рисунок 13: Меню вибору камери

Особливістю нової програми також є можливість переглядати попередні кадри при втраті зв’язку - в буфері зберігаються останні 200 кадрів, між якими без проблем можна переключатися. Цей функціонал виявився дуже потрібним, оскільки часто корегувати позицію літака потрібно саме по відео, бо GPS зв’язок відсутній.

3.3. Робота з відео

Отримувати якісне відео з мінімальною можливою затримкою було першочерговою задачею, адже пілоту дуже часто потрібно швидко зреагувати на потенційні загрози. В тому числі, під час посадки, оскільки затримка більше 100 мілісекунд може бути фатальною в такі моменти, та призвести до поломки літака. Тож було вирішено використовувати FFmpeg - безкоштовний кроссплатформений набір бібліотек та програм для отримання, запису та стрімінгу відео.

Для вирішення потрібних мені задач було створено два класи - RTSP_Stream(див. Рисунок 14) та Output_Video(див. Рисунок 15).

```
class RTSP_stream
{
public:
    bool open_stream(const char* src, int s_timeout, const char* connection_type, int rw_timeout);
    std::vector<uint8_t> get_frame();
    bool isInit, stopInit, isOn;
    std::string con_type;
    bool close_stream();
private:
    AVFormatContext* inputFormatContext;
    int videoStreamIndex;
    AVCodecParameters* videoCodecParameters;
    AVCodec* codec;
    AVCodecContext* videoCodecContext;
    AVDictionary* options;
    AVPacket* packet;
    AVFrame* frame;
    std::vector <uint8_t> Frame_to_RGB(AVFrame* tmp_frame);
};
```

Рисунок 14: RTSP_Stream Class

```
class Output_Video
{
public:
    bool init_output(const char* outputFilename, const char* type);
    void write_output(const std::vector<uint8_t>& rgbData);
    bool deinit_output();
    bool file_opened;
private:
    AVFrame* convertRGBToFrame(const std::vector<uint8_t>& rgbData);
    AVFrame* createSilentAudioFrame();
    bool init_video();
    bool init_audio();
    bool write_video(const std::vector<uint8_t>& rgbData);
    bool write_audio();
    const char* out_type;
    AVFormatContext* formatContext;
    AVCodecContext* videoCodecContext;
    AVCodecContext* audioCodecContext;
    AVStream* videoStream;
    AVStream* audioStream;
    int64_t frameCount;
};
```

Рисунок 15: Output_Video Class

Основною метою класу RTSP_Stream було отримання кадру з RTSP потоку використовуючи кодек H.264, але уже в RGB форматі. Тож було реалізовано 3 публічних метода open_stream, get_frame та close_stream.

В метод open_stream передаються важливі змінні - посилання на стрім, таймаут підключення, тип підключення та read/write таймаут.

В свою чергу, метод get_frame не приймає ніяких змінних, проте повертає так потрібний нам кадр в форматі rgb вектора.

Метод close_stream закриває стрім та очищає дані.

Також, важливим є і те, як обробляються дані в самій програмі. Потрібно було, щоб кадри приходили незалежно від того, що відбувається на екрані, а також, щоб перепідключення до відео було

максимально швидким. Тож було реалізовано окремий потік для перепідключення до відеопотоку, а метод отримання кадру та вивід його на екран відбувається 60 разів на секунду з використанням Таймеру в Winapi (див. Рисунок 16).

```
case WM_TIMER:
{
    if(wp == ID_TIMER_FRAME) write_frame(hWnd);
    break;
}
SetTimer(hWnd, ID_TIMER_FRAME, 1000/60, (TIMERPROC)NULL);
```

Рисунок 16: Winapi timer

Клас Ouput_Video, в свою чергу теж має декілька основних методів - init_output, write_output та deinit_output.

Метод init_output реалізований так, щоб вивід відео можна було робити як в файл, так і в RTMP. Саме черз це він приймає два аргументи. Перший - назва файлу або RTMP посилання. Другий - повідомлення, чи ми стрімимо в RTMP, чи в Файл.

write_output - такий метод, який приймає вектор RGB, і виводить його у потрібне місце - файл або стрім. Цікавим є те, що коли ми отримуємо кадри в класі RTSP_Stream, то відсутнє аудіо, а для корректної роботи RTMP стріму нам потрібно відправити аудіо та відео кадр. Тож було вирішено створити нульовий аудіокадр та разом з відео, яке задається RGB вектором, відправляти його.

Останній метод deinit_output - завершує файл/стрім та очищає дані.

3.4. Робота з Mavlink

Mavlink - це протокол для комунікації БПЛА з наземною станцією. Він був розроблений як header-only бібліотека для маршалінгу повідомлень. Я використовую кастомно згенеровану бібліотеку, оскільки в стандартних повідомленнях немає деяких речей, які використовуються в нашому літачку. Для цього я створив клас MavlinkClass, що використовує Winsock та Mavlink для підключення та отримання даних з літака (див. Рисунок 17).

```
class MavlinkClass
{
public:
    float pitch, roll, yaw; //rad
    int32_t latitude, longitude; //degE7
    int32_t altitude_MSL, relative_altitude; // mm
    uint16_t heading; //cdeg
    float airspeed, groundspeed, climb; //m/s
    uint16_t throttle; //%
    float aoa, ssa; //deg
    float wind_direction; //deg (from)
    float wind_speed; //m/s
    uint16_t battery_V; //mV
    uint16_t battery_A; //cA
    std::string mode; //autopilot mode
    float target_altitude; //m
    float target_airspeed; //m/s
    float target_bearing; //deg
    uint16_t RangeFinder; //cm
    int32_t home_altitude; //mm
    int32_t home_longitude, home_latitude; //degE7
    int32_t delivery_longitude, delivery_latitude; //degE7
    MavlinkClass();
    bool ConnectAutopilot();
    void GetData();
    ~MavlinkClass();
private:
    SOCKET sock;
    WSADATA wsaData;
    bool RecieveData(mavlink_attitude_t& attitude, mavlink_global_position_int_t& position
    bool RecieveAttitude(mavlink_attitude_t& attitude);
    bool RecievePosition(mavlink_global_position_int_t& position);
    bool RecieveFixedwing(mavlink_vfr_hud_t& fixedwing);
    bool RecieveAOA_SSA(mavlink_aoa_ssa_t& aoa_ssa);
    bool RecieveWind(mavlink_wind_t& wind);
    bool RecieveBattery(mavlink_battery_status_t& battery);
    bool RecieveHeartbeat(mavlink_heartbeat_t& heartbeat);
    bool RecieveTargetMessage(mavlink_nav_controller_output_t& target);
    bool RecieveRF(mavlink_distance_sensor_t& rf);
    bool RecieveHomePosition(mavlink_home_position_t& home);
    bool RecieveDelivery(mavlink_delivery_navigation_t& delivery);
    std::string DecodeMode(mavlink_heartbeat_t& heartbeat);
```

Рисунок 17: MavlinkClass

Хоч це і не є правильним, та я зберігаю всі отримані дані в цьому класі в публічних змінних, які можна без додаткових методів витягнути.

Основними ж методами є ConnectAutopilot та GetData. ConnectAutopilot - це модифіковане Winsocket підключення з таймаутом 1 секунда. GetData - це метод, який намагається отримати дані з літака, а якщо це неможливо - перепідключається до нього.

Оскільки часто можливо таке, що через слабкий зв’язок відео не приходить, але дані з літака є, було вирішено незалежно від відео отримувати дані з автопілоту, адже багатьом пілотам для керування їх досить. Більше про реалізацію в розділі OSD пілота.

3.5. Інтеграція OpenGL

Найскладнішим в інтеграції було підключення OpenGL до Winapi. Для цього я спочатку ініціалізую Render Context у функції init_Render_Context, після чого потрібно вже ініціалізувати сам GL та GLEW. Для полегшення, я також використовую бібліотеку glm. Після ініціалізації GL також потрібно було ініціалізувати Буфери та Шейдери, які потрібні мені. Обов’язковим також була реалізація функції resize_GL, яка змінює розмір вікна малювання.

Після ініціалізації стало питання виводу відео. Оскільки відео з літака завжди приходить в однаковому розмірі, а саме 1280 на 720 пікселів, а саме вікно програми може змінювати свій розмір, я прийшов до висновку, що потрібно використовувати Framebuffer. Ось чому я спочатку виводжу відео, після чого накладаю на відео сітку ландшафту, OSD та інші точки інтересу. Наступний крок - зберегти цей кадр уже з додатково намальованими речами - щоб в такому ж розмірі зберегти відео в файл або відправити в RTMP. Після чого я змінюю Framebuffer на стандартний, змінюю розмір на розмір екрану та виводжу відео.

3.6. OSD пілота

Один із найважливіших елементів програми - дані з літака та їх візуалізація. Перш за все - це положення літака в просторі - attitude. Це положення визначається трьома кутами - pitch, roll та yaw. На землі літак калібрується так, щоб горизонт був в положенні 0 по pitch та roll, інакше ці дані не будуть відображати реальне положення. Це - основа OSD. В нашому випадку, yaw - значення кута компаса літака. (див. Рисунок 18).

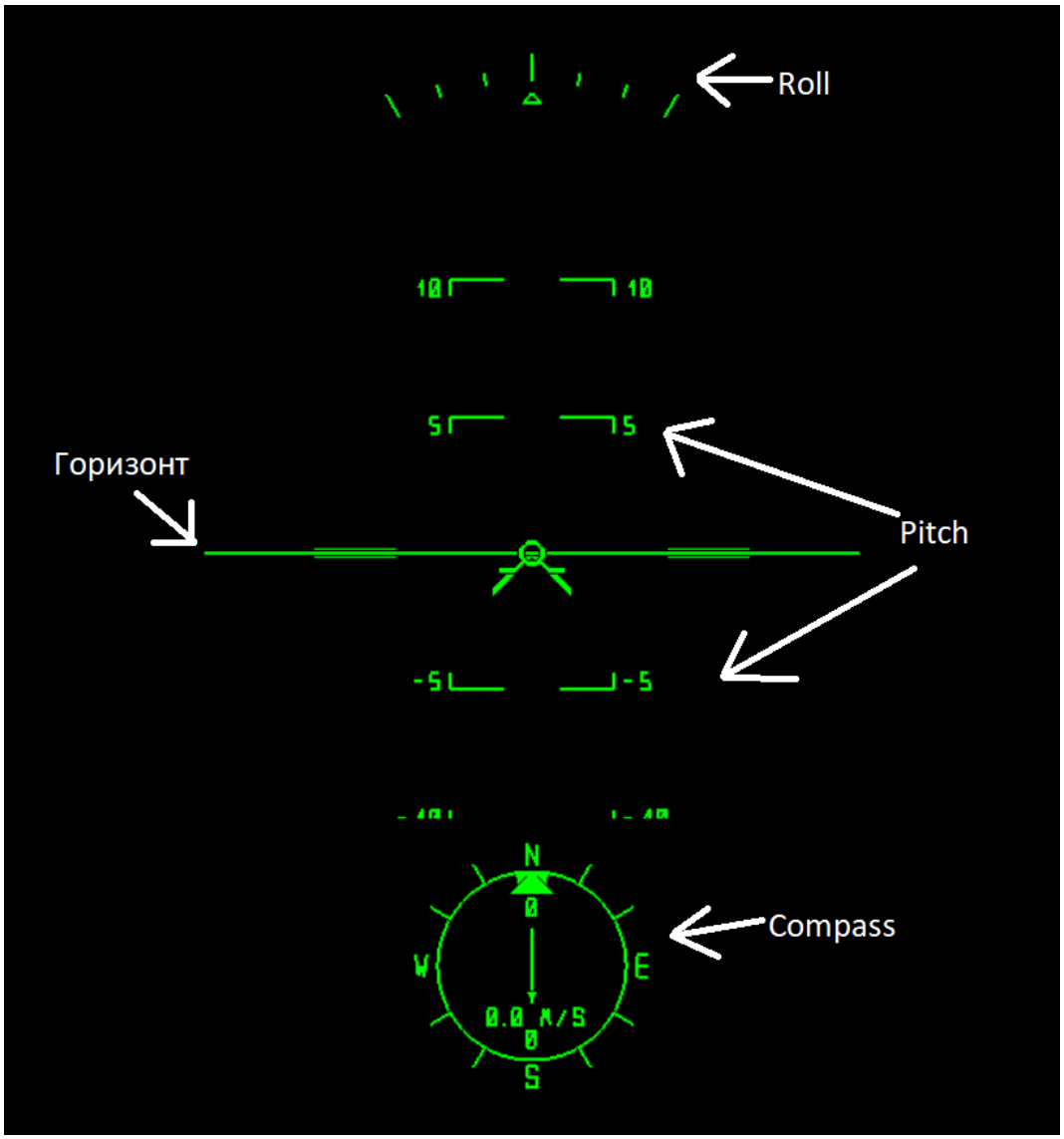

Рисунок 18: Attitude

Детальніше про кожен елемент:

1. Roll - ця панель показує крен літака. Кожна позначка означає 15 градусів повороту. Крім цього, крен також впливає на панель pitch.
2. Pitch - панель, на якій розташовано декілька елементів. По перше - лінія горизонту. Вона розташована на 0 значенні pitch. Ця лінія є основним орієнтиром для пілота під час польоту, адже якщо вона збігається з двома симетричними відносно центру невеликими відрізками, то літак летить рівно по горизонту. Відповідно при зміні куту pitch та/або roll ця лінія зміщується та/або повертається на відповідні кути. По друге - AOA_SSA. Це невелике коло з двомам симетричними галочками. Цей знак показує реальне зміщення літака в просторі. Відповідно, якщо він в центрі, то літак летить чітко вперед.
3. Compass - панель з компасом. Окрім стандартних речей, таких як проміжки в 30 градусів та позначки Півночі, Півдня, Заходу та Сходу, на лінії кола також є невелика позначка - це курс на наступну точку маршруту. Якщо Трикутник, що вказує на значення градуса, збігається з цією позначкою - значить ми летимо чітко на точку. Під трикутником маємо градусне значення нинішнього напряму літака, а ще нижче три значення - стрілочка та 2 числових значення. Стрілочка вказує на напрям вітру

відносно курсу літака, а числові значення - це швидкість вітру, та значення кута, ЗВІДКИ дує вітер.

Це основна панель польоту, вона реалізована досить просто - малюванням ліній та фігур з використанням OpenGL. Дані беруться з класу MavlinkClass.

Наступний елемент - панель швидкості(див Рисунок 19).

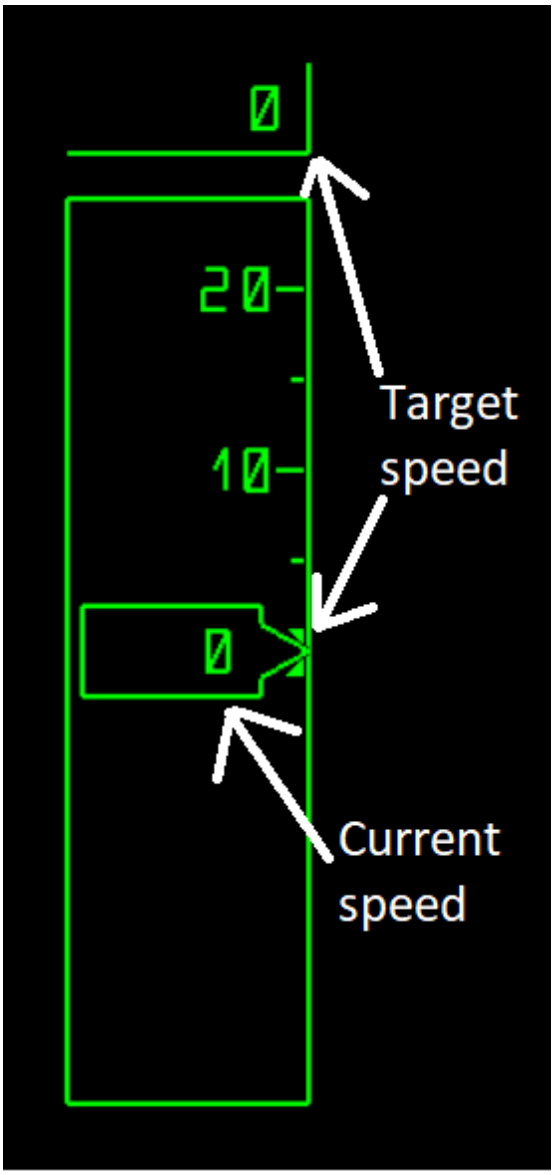

Рисунок 19: Speed panel

Важливе уточнення - виводиться саме повітряна швидкість літака, а не швидкість відносно землі, оскільки трубка Піто, що

встановлюється на літаку, дає нам дані саме про неї. Крок - 5 м/с, крім цього, target speed дублюється як числом, так і позначкою на панелі.

Перейдемо до Altitude panel(панелі висоти) (див Рисунок 20).

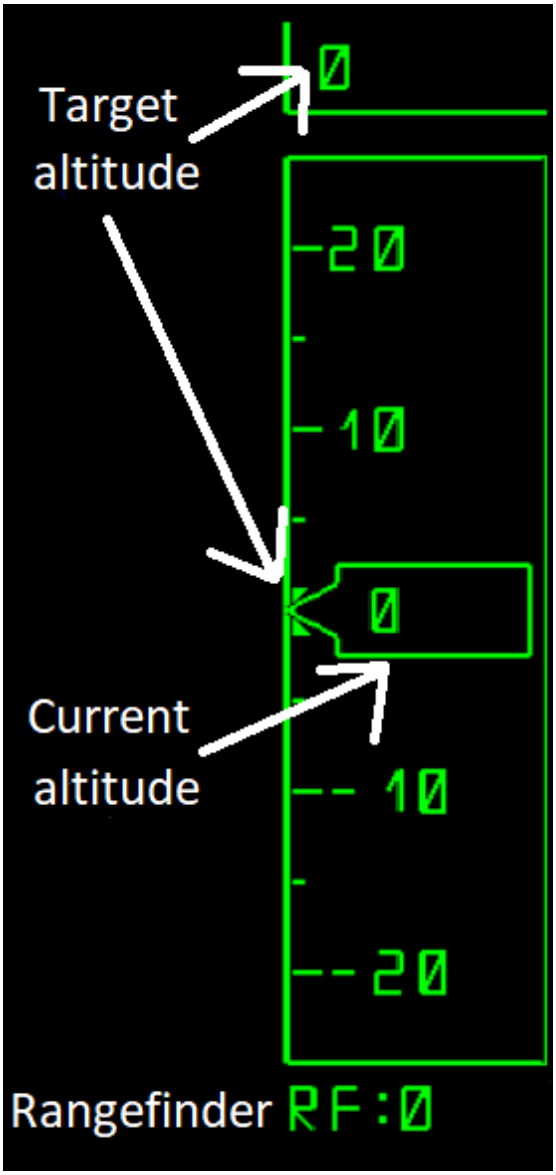

Рисунок 20: Altitude panel

Як і панель швидкості, панель висоти складається з поточної висоти, відміток з кроком в 5 метрів та target altitude - позначка на панелі плюс числове значення зверху. Крім цього, під нею є спеціальне значення - Rangefinder. Це лазерний висотомір, який

показує висоту з точністю до сантиметра, але працює лише на висоті до 12 метрів. Використовується в основному під час посадки літака.

Наступний елемент - панель швидкості набору або ж панель вертикальної швидкості (див. Рисунок 21).

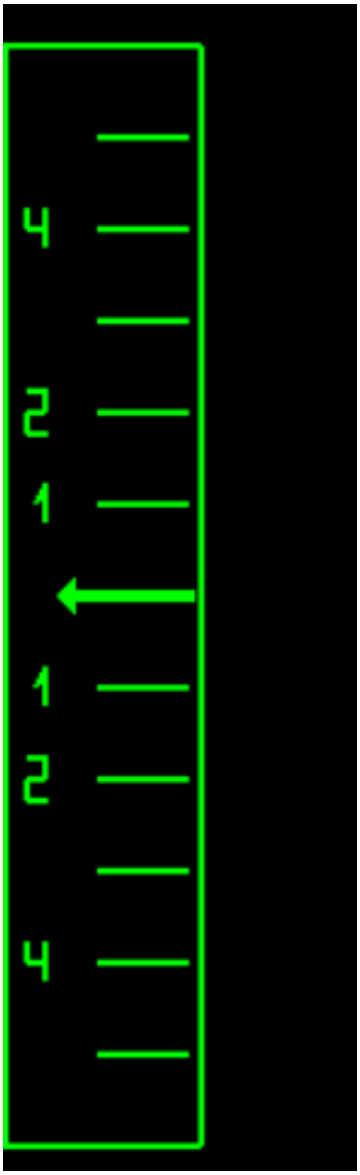

Рисунок 21: Vertical speed panel

Ця панель показує швидкість набору висоти літака в метрах за секунду. Різницею з попередніми двома є те, що в них показник завжди вказує на середину панелі і виводить відповідну швидкість, в той час як значення навколо змінюються. А в цьому варіанті він

рухається, поки значення залишаються на місці. Якщо показник вище центру - ми набираємо висоту, якщо нижче - втрачаємо.

Важливе уточнення по реалізації - для виключення варіанту накладання різних панелей одна на одну(основне - накладання attitude панелі на всі інші при поворотах) я використав StencilBuffer в OpenGL. Він дає змогу «вирізати» якусь ділянку екрану для того, щоб малювання об’єктів відбувалося лише на вирізаній частині або ж не відбувалося на ній.

Наступна панель - струм (див. Рисунок 22).

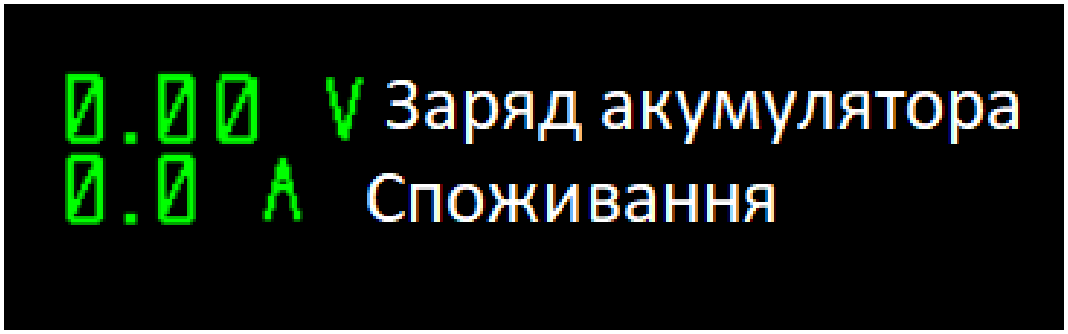

Рисунок 22: Панель струму

Основна частина БПЛА працює саме на акумуляторах, тож виводити струм та споживання потрібно обов’язково. Для моніторингу також додано зміну кольору на червоний. Це відбувається або якщо низький заряд батареї, або якщо низьке споживання при наборі висоти - це означає, що з ладу міг вийти регулятор двигуна.

Остання панель - режим польоту (див. Рисунок 23).

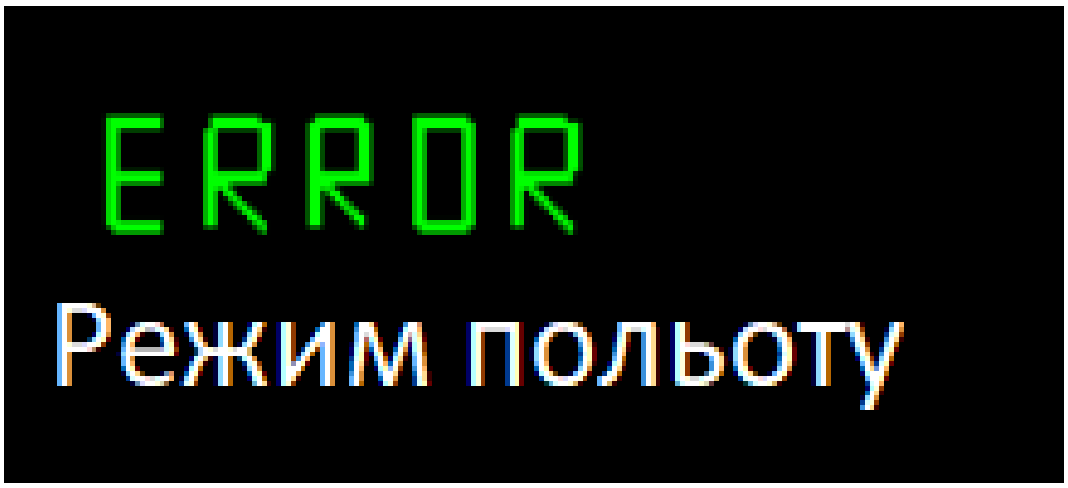

Рисунок 23: Панель режиму польоту

Ця панель створена для моніторингу режиму, в якому зараз знаходиться БПЛА. Якщо ж режим «ERROR», то зв’язок з літаком відсутній.

Також в програмі є декілька повідомлень, які попереджають пілота про зміни та/або небезпеку. Вони висвічуються і зникають декілька разів великими написами. Наприклад, про зміну режиму польоту (див. Рисунок 24).

Рисунок 24: Режим змінено на «ERROR» - зв’язок втрачено

Для простоти було вирішено створити свій «прямокутний» шрифт, який не має заокруглень. Також важливо те, що він має чорну обводку.

3.7. Terrain

Реалізація точної моделі поверхні землі навколо літака вимагала неабияких зусиль. Найперше питання, яке потрібно було вирішити - це набір даних. Для більшої точності можна було б використовувати спеціальні висотоміри, камери або якісь інші девайси, які точно могли б визначити в реальному часі відстань до певних об’єктів. Також, можна було б поставити на літак дві камери, і відносно них рахувати відстань. Але все це не підходило. Тож було вирішено взяти дані з місії SRTM.

SRTM - це міжнародний проект створенню цифроівої моделі висот землі за допомогою даних, отриманих спеціально модифікованою радарною системою , яка летіла на борту Space Shuttle Endeavour [3].

Оскільки нам потрібна була максимальна точність, то було взято варіант SRTM1, що включає знімки з кроком в 1 арк-секунду.

Дані поставляються в файлах .hgt з відповідними позначеннями, який квадрат покриває. Відповідно, ми зчитаємо цей файл і отримуємо матрицю 3600 на 3600 - значення висоти в кожній арк-секунді від нашого longitude та latitude до longitude + 1 та latitude + 1.

Для зчитування з файлів створимо union з назвою grid_io_block (див. Рисунок 25) і будемо записувати в нього блоками з дані.

```
union grid_io_block
{
    int16_t height[GRID_BLOCK_SIZE];
    char buffer[BLOCK_SIZE];
}
```

Рисунок 25: grid_io_block

Оскільки висоти в цих файлах рахуються відносно рівня моря, то і віртуальну камеру(в OpenGL) ми будемо виставляти на висоті літака відносно рівня моря. Перейдемо до реалізації самої сітки

Перш за все - виставляємо нашу віртуальну камеру на висоту, налаштовуємо проекцію відносно положення нашого літака в просторі. Не забуваємо про перспективу, яка залежить від характеристик камери літака. Вважаємо, що координати нашого літака (0, 0). Далі найцікавіше - «квадрат» 1 арк-секунда на 1 арк-секунду насправді не є квадратом - зазвичай це прямокутник, в залежності від того, в яких кординатах ми знаходимося. Тож, спершу потрібно вирахувати крок по x та y. Якщо по y це не проблема, оскільки на глобусі крок в 1 арк-секунду по latitude завжди однаковий, і дорівнює приблизно 30.8875 метрів, то по longitude він може змінюватися, в залежності від latitude. Формула розрахунку - $30.8875 * \cos(\text{current\_latitude})$.

Відповідно, маючи всі висоти, координати літака, його положення в просторі, кут камери та її характеристики, ми можемо

намалювати ландшафт. Використовуючи Z-buffer та модифіковані шейдери, отримуємо потрібний нам результат (див. Рисунок 26).

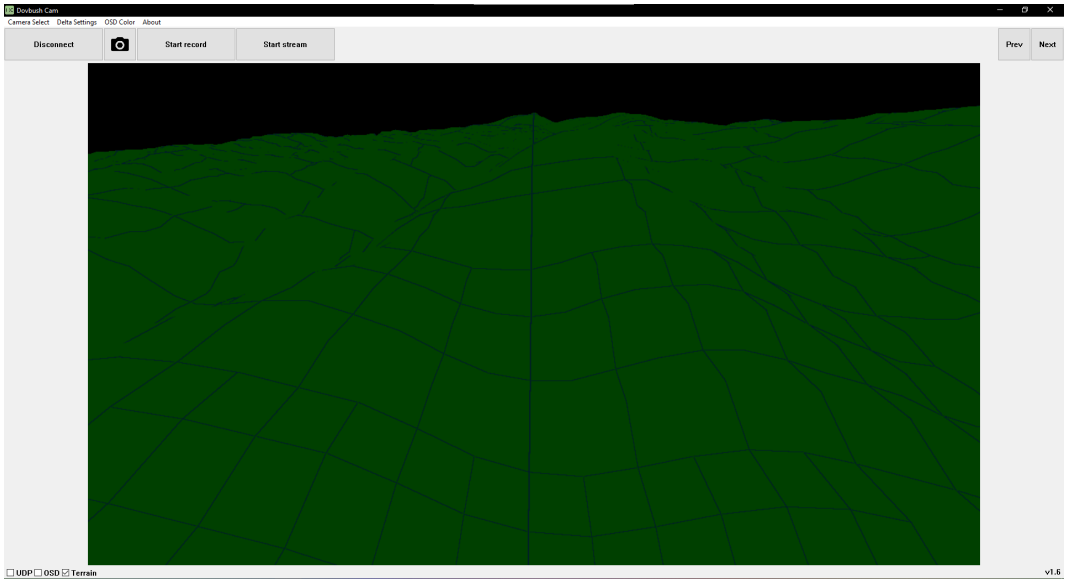

Рисунок 26: Terrain

Важливими моментами, реалізованими в шейдері, є напівпрозорість сітки та зміна розмірів кроку відносно висоти та віддаленості від літака.

3.8. Реалізація точок інтересу

На даному етапі було поставлено задачу реалізувати всього дві точки інтересу - приціл та точку Home.

Для першого на літаку за допомогою балістичного калькулятора розраховується точка, куди прилетить снаряд, та записуються у форматі (latitude, longitude). Тож все, що потрібно було додати - сам приціл та ймовірну зону ураження (див. Рисунок 27).

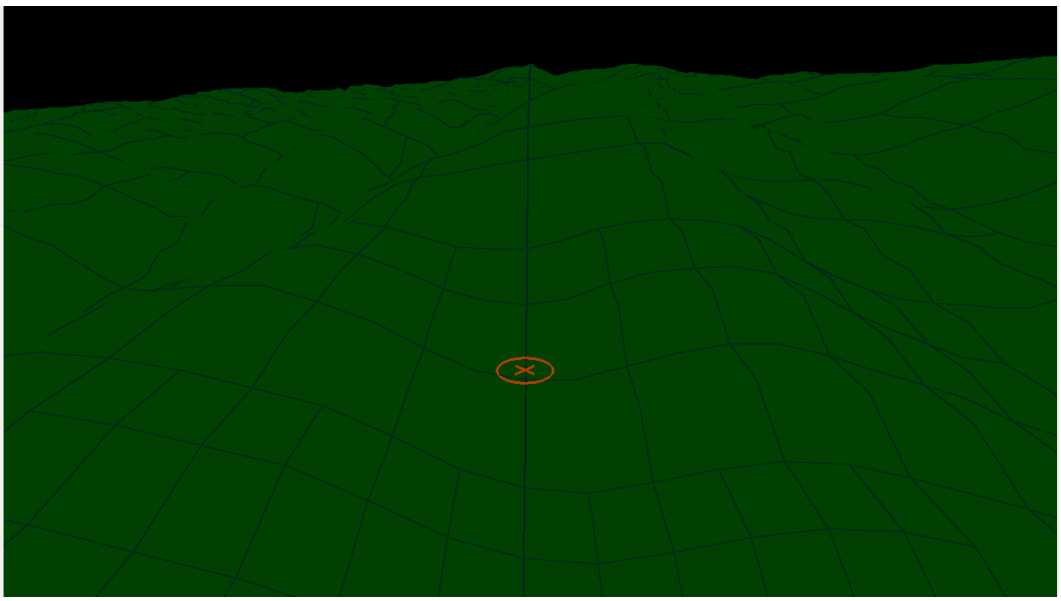

Рисунок 27: Приціл

Наразі, приціл не враховує весь ландшафт навколо місця прильоту та реальну зону ураження(див. Рисунок 28), проте робота над цим продовжується.

Рисунок 28: Вирахування реальної зони ураження [4]

Точка Home - це просто трикутник, який вказує на координати місця зльоту літака, також отримується в координатах (latitude, longitude). Важливо було реалізувати так, щоб її розмір змінювався відносно висоти польоту та був повернутий завжди до камери(див. Рисунок 29).

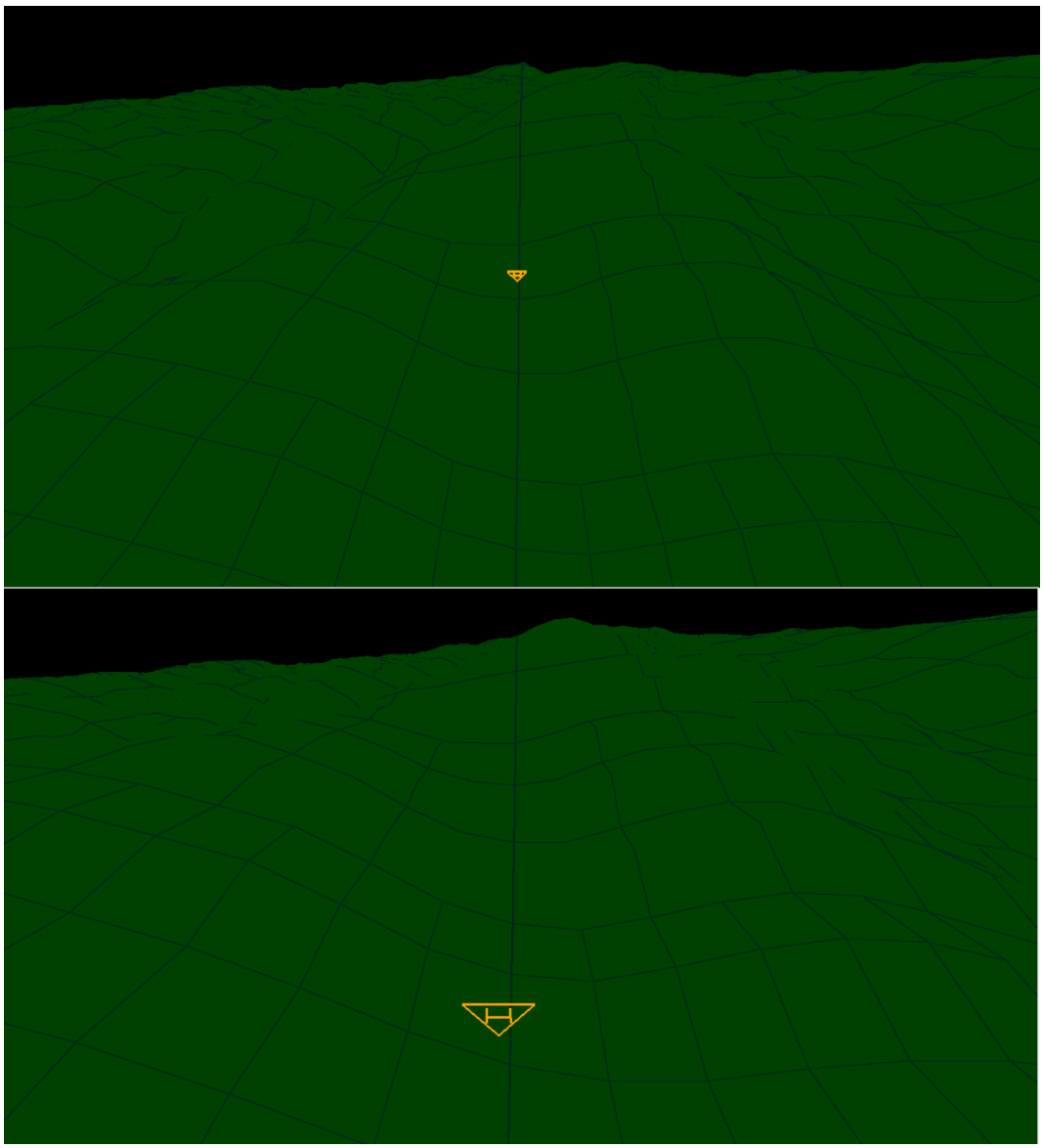

Рисунок 29: Точка Home

РОЗДІЛ 4. ВИСНОВКИ

У даній дипломній роботі було розглянуто застосування технології доповненої реальності для БПЛА. Це включає в себе розробку виводу відео, можливість запису в файл та стріму в RTMP, накладання на відео OSD пілота, моделювання та накладання ландшафту на відео, а також накладання на ландшафт точки Home та прицілу для снаряду.

Протягом виконання роботи було встановлено, що застосування доповненої реальності до безпілотних літальних апаратів значно покращує якість виконання задач та безпеку польотів. Зокрема, інтеграція технології доповненої реальності з виводом відео дозволяє операторам отримувати більше інформації про стан БПЛА і оточуюче середовище.

Реалізована можливість запису в файл і стріму в RTMP дозволяє виконувати онлайн-трансляції польотів безпілотників, що може бути використано для тренування пілотів, моніторингу стану апарата, а також для оцінки якості виконання задач.

Застосування технології накладання OSD пілота та моделювання ландшафту на відео забезпечує точніше і зручніше виконання завдань, що вимагають просторового орієнтування. Також полегшується процес корегування артилерії, реактивних

систем та іншого, завдяки використанню кроку сітки як одиниці виміру в метрах.

Приціл для снаряду, а також точка Home, що накладається на ландшафт, дозволяє пілоту швидко оцінити положення безпілотника відносно стартової точки та мети, що покращує точність навігації та ефективність виконання завдань.

Основним висновком дипломної роботи є те, що доповнена реальність може значно покращити ефективність і безпеку використання безпілотних літальних апаратів. Розроблена програма демонструє практичність та ефективність використання доповненої реальності в сфері БПЛА, включаючи покращення в прямому відео-виводі, веденні запису і онлайн-трансляції, а також у накладанні критичної інформації на відео для пілотів. Крім цього, відео використання підрозділами Міністерства оборони України доводить ефективність та зручність розробленої програми [5].

Далі, пропонується продовжувати дослідження в даному напрямку, зокрема, можна зосередитися на удосконаленні алгоритмів моделювання ландшафту і накладання графіки, а також на розробці більш точних методів визначення місця падіння снаряду. Важливо також продовжити дослідження щодо оптимізації роботи програми з метою зменшення вимог до обчислювальних ресурсів.

РОЗДІЛ 5. ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] ArduPilot Dev Team, “Mission planner.” https://ardupilot.org/planner/index.html

[2] QUANTUM-SYSTEMS GMBH, “Qbase 3d: Professional flight planning software. https://www.quantum-systems.com/qbase-3d/

[3] NASA, “U.S. Releases enhanced shuttle land elevation data.” https://www2.jpl.nasa.gov/srtm/

[4] BOYGA. *Accurate Hit on Target From STM's BOYGA, Autonomous Multi-Rotor UAV With Mortar Payload.* [Online Video]. Available: https://www.youtube.com/watch?v=aClS0VGxbQ4&ab_channel=DefensehereEnglish

[5] Державна прикордонна служба України. *Що У Тилу Окупантів? Там Горить! Працює Аеророзвідка Дпсу*. [Online Video]. Available: https://www.facebook.com/watch/?v=3373276239601481