МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)
ЗДОБУВАЧА ОСВІТНЬОГО СПУПЕНЯ **«БАКАЛАВР»**

Тема: «Технологія побудови віртуальних середовищ для дослідження проблем автономної навігації БПЛА»

Виконавець: Карина ІЛЬЧЕНКО

Керівник: к.т.н., доцент Костянтин ПРОКОПЕНКО

Нормоконтролер: к.т.н., доцент Вікторія СИДОРЕНКО

КИЇВ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *комп'ютерних інформаційних технологій*

Спеціальність *122 «Комп'ютерні науки»*

Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:

Завідувач кафедри КІТ

_____Аліна САВЧЕНКО

«____»_____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Ільченко Карини Олександрівни

(прізвище, ім'я, по батькові здобувача вищої освіти в родовому відмінку)

1. Тема кваліфікаційної роботи: «Технологія побудови віртуальних середовищ для дослідження проблем автономної навігації БПЛА», затверджена наказом ректора від «05» квітня 2024р. № 517/ст.

2. Термін виконання роботи: з 06 травня 2024 року по 16 червня 2024 року.

3. Вихідні дані до роботи: віртуальне середовище на ігровому рушії Unreal Engine з інтегрованим симулятором БПЛА Airsim та нейронною мережею YOLO, інтегроване середовище розробки Visual Studio 2022.

4. Зміст пояснювальної записки: 1) Аналітичний огляд і постановка задачі. 2) Огляд і вибір засобів та інструментів розробки. 3) Планування і реалізація роботи.

5. Перелік обов'язкового ілюстративного матеріалу: слайди презентації MS PowerPoint.

6. Календарний план-графік

№з/п	Завдання	Термін виконання	Підпис керівника
1	Аналіз літератури та джерел за темою предметної області. Проведення консультації з науковим керівником щодо написання розділів.	05.04.2024р. – 20.04.2024р.	
2	Аналітичний огляд і постановка задачі. Робота над першим розділом.	21.04.2024р. – 29.04.2024р.	
3	Огляд і вибір засобів та інструментів розробки. Робота над другим розділом.	30.04.2024р. – 12.05.2024р.	
4	Розробка і налаштування віртуального середовища. Робота над третім розділом.	13.05.2024р. – 26.05.2024р.	
5	Редагування і остаточне оформлення пояснювальної записки.	27.05.2024р. – 30.05.2024р.	
6	Проходження нормоконтролю та перевірки на антиплагіат. Підписання необхідних документів у встановленому порядку.	31.05.2024р. – 04.06.2024р.	
7	Підготовка до захисту роботи. Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	05.06.2024р. – 07.06.2024р.	

7. Дата видачі завдання: «05» квітня 2024 р.

Керівник кваліфікаційної роботи _____ Костянтин ПРОКОПЕНКО
(підпис керівника)

Завдання прийняла до виконання _____ Карина ІЛЬЧЕНКО
(підпис здобувача вищої освіти)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Технологія побудови віртуальних середовищ для дослідження проблем автономної навігації БПЛА» містить: 73 сторінки, 29 рисунків, 14 інформаційних джерел.

Об’єкт дослідження — процес локальної навігації БПЛА у симульованому середовищі.

Предмет дослідження — методи та засоби ефективної і швидкої побудови віртуального середовища для дослідження локальної автономної навігації БПЛА.

Мета роботи — дослідити локальну автономну навігацію БПЛА у віртуальному середовищі, розглянути методи підходу до побудови віртуальних середовищ, інтегрування симулятора БПЛА в них, а також проаналізувати переваги використання нейронних мереж.

Методи дослідження — логічний, алгоритмічний аналіз, порівняльний, аналіз інформаційних джерел, моделювання та симуляція.

Наукова новизна — вперше запропонований швидкий та ефективний метод побудови віртуальних середовищ для дослідження локальної навігації БПЛА за рахунок поєднання сучасних технологій, таких як ігровий рушій Unreal Engine, симулятор AirSim, мова програмування Python, нейронна мережа YOLO задля розв’язання різних прикладних задач.

Результат роботи — розроблена та готова до використання зручна технологія, яка представляє собою імітаційну модель віртуального середовища з інтегрованим симулятором БПЛА.

Практичне значення отриманих результатів. Отримана технологія побудови віртуальних середовищ для дослідження проблем автономної навігації БПЛА може бути використана як ефективний і зручний засіб для вирішення прикладних задач, наприклад, як виявлення різноманітних об’єктів камерою дрона.

Ключові слова: ВІРТУАЛЬНЕ СЕРЕДОВИЩЕ, ІГРОВИЙ РУШІЙ, UNREAL ENGINE, AIRSIM, СИМУЛЯЦІЯ, БПЛА, НАВІГАЦІЯ, YOLO.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

БПЛА	−	Безпілотні літальні апарати
3D	−	Тривимірне
UE	−	Ігровий рушій Unreal Engine 5
Віртуальне середовище	−	Цифрові імітації реальних середовищ або уявні середовища.
Blueprints Visual Scripting	−	Візуальна мова програмування в Unreal Engine.

ВСТУП

Питання навігації безпілотних літальних апаратів є важливою, цікавою і досить актуальною темою у сучасному світі на сьогоднішній день, яка потребує значної уваги до себе. Тому наявність автономної навігації БПЛА може принести надзвичайну користь і багато переваг, які розширять можливості та ефективність роботи літальних апаратів. Але оскільки розробка алгоритмів автономної навігації є доволі складною технологією, а також може потребувати використання багатьох ресурсів, тому щоб не витрачати марно літальні апарати та інші ресурси під час всіх спроб їхнього навчання, спочатку буде раціональніше розглянути інші методи розв'язання цього питання. Отже, замість навчання дронів в реальних умовах, можна обрати такий підхід, який полягає у створенні віртуального середовища, яке буде симулювати реальне зовнішнє середовище. Для цього необхідно побудувати віртуальне середовище, використовуючи для його розробки підходящу систему, щоб потім тестувати і навчати модель БПЛА у вже розробленому середовищі за допомогою симулятора дронів та проводити експерименти з різноманітними методами машинного зору.

Мета і завдання виконання кваліфікаційної роботи — дослідити локальну автономну навігацію БПЛА у віртуальному середовищі та її особливості, визначити, наскільки важливим і корисним є використання віртуальних середовищ для дослідження проблем навігації, розглянути методи підходу до побудови віртуальних середовищ, інтегрування симулятора БПЛА в них, а також проаналізувати переваги використання нейронних мереж для вивчення роботи машинного зору дрона.

Об'єктом дослідження є процес локальної навігації БПЛА у симульованому середовищі.

Предмет даного дослідження — методи та засоби ефективної і швидкої побудови віртуального середовища на базі ігрового рушія Unreal Engine з

інтегрованим симулятором БПЛА для дослідження локальної автономної навігації БПЛА.

Методи дослідження — логічний, алгоритмічний аналіз, порівняльний, аналіз інформаційних джерел, моделювання та симуляція.

Дана робота була створена з метою розроблення зручної, актуальної та новітньої технології для забезпечення сумісної роботи віртуального середовища, побудованого швидким і ефективним методом, симулятора БПЛА та нейронної мережі для вивчення та тестування систем локальної навігації безпілотних літальних апаратів задля полегшення користування БПЛА та процесу навчання керування ними. Це також допоможе дослідити і зробити користування літальними апаратами в реальних умовах більш безпечним після їхнього навчання у віртуальних середовищах, бо БПЛА зможуть автоматизовано працювати у важкодоступних та небезпечних територіях, де доступ людині може бути обмежений.

Кінцевою метою роботи є розроблена та готова до використання технологія, яка представляє собою імітаційну модель віртуального середовища з інтегрованим симулятором БПЛА. За допомогою цієї технології можна буде досліджувати і вдосконалювати локальну навігацію БПЛА та функціонування машинного зору дрона.

Актуальність обраної теми полягає в необхідності розвинення технологій та навчання спеціалістів у військовій галузі керування безпілотними літальними апаратами. Тому для проведення експериментів і навчання БПЛА автоматизованим алгоритмам навігації або іншим задачам, краще спочатку використовувати віртуальне середовище.

Наукова новизна — вперше запропонований швидкий та ефективний метод побудови віртуальних середовищ для дослідження локальної навігації БПЛА за рахунок поєднання сучасних технологій, таких як ігровий рушій Unreal Engine, симулятор AirSim, мова програмування Python, нейронна мережа YOLO задля розв’язання прикладних задач.

Практичне значення отриманих результатів. Отримана технологія побудови віртуальних середовищ для дослідження проблем автономної локальної навігації БПЛА може бути використана як ефективний і зручний засіб для вирішення прикладних задач, наприклад, як виявлення різноманітних об’єктів камерою дрона за допомогою методів машинного зору.

Звіт з практики складається зі вступу, основної частини, висновків, списку використаних літературних джерел та додатків. У основній частині було проведено аналітичний огляд навігації БПЛА, обґрунтовано вибір обраної теми, її актуальність і важливість у сучасному світі, визначено мету і завдання дослідження, наведено приклади застосування розробленої імітаційної моделі середовища. Також було визначено вимоги до роботи, проведено аналіз різних засобів та інструментів, наведено їх внутрішню структуру та зроблено вибір найбільш оптимальний та ефективних для роботи інструментів. Було детально опрацьовано всі етапи реалізації і налаштування віртуального середовища, створеного в Unreal Engine. У висновках були підсумовані результати всієї проведеної роботи, досягнуті цілі та виконані завдання, а також розглянуті перспективи використання і подальшого розвитку технології. У списку використаної літератури подається перелік допоміжних джерел, які були використані на різних етапах кваліфікаційної роботи.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД І ПОСТАНОВКА ЗАДАЧІ

Першочергово доцільно розглянути Безпілотні літальні апарати, що вони собою представляють, та аналізувати для чого варто використовувати віртуальні середовища для дослідження їхньої локальної навігації в них. Безпілотні літальні апарати — це літальні апарати, які керуються дистанційно і не мають пілота на борту, вони можуть мати різний рівень автономності, конструкцію, призначення та відрізнятися іншими параметрами [1]. Однією з їх основних функцій є виконання розвідувальних завдань.

Також важливо визначити, що таке віртуальне середовище і яку роль вони відіграють у даному дослідженні навігації моделі симуляції БПЛА. Віртуальні середовища — це створені на комп'ютері за допомогою ігрового рушія цифрові імітації реальних середовищ або уявні середовища, в яких об'єкти можуть взаємодіяти між собою та з користувачами [2]. Віртуальні середовища дозволяють виконувати різноманітні дослідження у будь-яких сферах за допомогою них та, наприклад, як в нашому випадку, вони можуть допомогти з проведенням навчання і тестування моделей віртуальних безпілотних літальних апаратів.

На жаль, не завжди є можливість оперувати безпілотні літальні апарати одразу в реальних умовах. Для того, щоб розв'язати цю проблему, спочатку для оцінки та навчання моделі дрона краще використовувати саме симуляцію реальності, а не реальні умови. Бо великою перевагою їх використання є те, що можливі збої в роботі моделі дрона ефективніше виявляти в симуляційному віртуальному середовищі, оскільки це економічно вигідніше, а також безпечніше для людей і довкілля.

Окрім того, за необхідністю в одному створеному віртуальному середовищі може експлуатуватися кілька безпілотників одночасно, що дозволяє проводити для них паралельні обчислення та тести. Також в віртуальному середовищі є можливість легко змінювати мапу, об’єкти, текстури й кольори будь-яких об’єктів, як тільки користувач забажає, за лічені хвилини.

Однак, використання симуляцій також має і деякі свої недоліки, основним з таких є можливість невідповідності між створеним симульованим середовищем і реальним світом. Або можливо зіткнутися з такими проблемами, як нереалістична обробка зіштовхнень моделі літального апарату з іншими об’єктами, нереалістична симуляція фізики та інші неточності симуляції, які можуть виникнути в процесі роботи. Ці невідповідності у поведінці дрона не завжди можуть бути критичними, але неточні розрахунки і виявлення зіткнень у симуляторі може надалі в майбутньому призвести до аварії при експлуатації БПЛА для виконання завдань у реальних умовах.

1.1. Огляд загальної і локальної навігації БПЛА

Навігація безпілотних літальних апаратів є насамперед ключовим аспектом їхнього функціонування, який є центром всього операційного процесу роботи безпілотника, визначаючи точність і безпеку маршруту польоту, а також успішне виконання поставлених завдань.

На борту БПЛА встановлено різноманітне обладнання, зазвичай таке обладнання складається з камер, сенсорів, датчиків, гіроскопів, щоб точно вимірювати кут нахилу безпілотника, та іншого додаткового обладнання [5]. Для безпілотних літальних апаратів, які не мають достатньої ступеня автоматизації, керування завжди залишається у руках пілота чи оператора дрона. На недорогих безпілотниках визначення положення та орієнтації здійснюється за допомогою візуального слідкування, що відбувається або з землі, враховуючи розташування користувача дроном, або з використанням бортових камер, які передають візуальні дані на монітор пілота дрона [4].

Перш за все, використання БПЛА у виконанні різноманітних місій у реальних умовах вимагає розробки додаткових захисних механізмів від радіоелектронних перешкод та негативного впливу погодних умов [3]. Також важливо забезпечити підвищену безпеку польотів безпілотних літальних апаратів у зонах, де відбувається виконання поставлених завдань.

Варто зазначити, що авжеж оператор бойового БПЛА не піддається такому ж самому ризику для свого здоров'я та життя, як в порівнянні з пілотом винищувача, однак БПЛА все одно має свої слабкі сторони — уразливість і чутливість системи дистанційного керування, що є надзвичайно критичним для дронів, які використовуються для військових завдань [1].

Обмеження використання [5] БПЛА включають декілька ключових факторів. По-перше, обчислювальні можливості на борту безпілотника не завжди дозволяють виконувати складні обчислення в режимі реального часу. По-друге, час роботи безпілотного літального апарату обмежується його джерелом живлення, що може сильно обмежувати роботу деяких високоресурсних алгоритмів. По-третє, також може виникнути потреба у якісному, але дорогому ремонті БПЛА через можливість серйозних пошкоджень високоточного обладнання внаслідок аварій, що, безперечно, є великими витратами. Всі ці обмеження підкреслюють важливість та необхідність проведення подальших досліджень з метою постійного вдосконалення характеристик безпілотних літальних апаратів.

Отже, підсумовуючи усе вище сказане, можна зазначити, що попереднє навчання БПЛА у віртуальних середовищах перед їхнім реальним практичним застосуванням може значно полегшити роботу операторів безпілотників і також зробити їхню роботу безпечнішою завдяки цьому. Це могло б призвести до покращення навігації літальних апаратів та зробити її більш автоматизованою або ж повністю автоматизованою.

1.2. Мета та завдання дослідження

Метою даного дослідження є отримання готової для використання віртуальної сцени середовища подальшого зручного вивчення та аналізу алгоритмів локальної навігації безпілотних літальних апаратів. Однак, щоб досягти поставленої мети дослідження, необхідно спочатку визначити і виконати головні завдання дослідження.

Перед початком моделювання тривимірних сцен, по-перше, слід розглянути методи підходу до побудови цих віртуальних середовищ, ознайомитися з технологіями й інструментами розробки, які будуть необхідні для виконання поставленого завдання. Дослідити їх вимоги та обов'язково ознайомитися з технічною документацією.

Коли будуть обрані потрібні системи побудови віртуального середовища, можна буде перейти до наступних задач. А саме — налаштування обраного середовища розробки, підготовка його для програмування алгоритмів локальної автономної навігації безпілотника. Далі слідує процес створення макетів для середовища, розробка композиції з урахуванням розміщення об'єктів, освітлення, текстур та інших важливих компонентів, щоб розробити якісне деталізоване середовище.

Після створення сцени мапи та розміщення моделей об'єктів у створеному віртуальному середовищі настає етап інтегрування симулятора БПЛА у розроблену модель середовища для симуляції поведінки безпілотника та забезпечення можливості взаємодії з ним задля тренування літального апарату алгоритмам локальної автономної навігації. Далі настає надзвичайно важливий етап перевірки розроблених та налаштованих компонентів для оптимізації продуктивності віртуального середовища.

По завершенню виконання всіх поставлених завдань, мета дослідження буде досягнута. Буде отримано повноцінну і готову до використання технологію, яка представляє собою реалістичну модель віртуального середовища і має в собі інтегрований симулятор безпілотних літальних апаратів. Технологія надасть можливість виконувати польоти БПЛА у віртуальному просторі, а також

тестувати різноманітні аспекти поведінки та функціональності безпілотних літальних апаратів.

1.3. Актуальність дослідження

При згадці про важливі технологічні розробки нашого століття, багатьом спочатку на думку спадають соціальні мережі та смартфони — щось, що стало незамінним в нашому повсякденному житті, тобто речі першої необхідності. Для інших це можуть бути «електромобілі», але ці технології вже досить відомі. Але насправді в українців є нестандартна, але важлива відповідь на це питання — безпілотні літальні апарати.

Дрони на сьогоднішній день здатні здійснювати високоякісні аерофото- та відеозйомки, які зараз широко застосовуються у різних галузях — від кіно та фотоіндустрії до військового спостереження. Коротко розглянемо їх використання у кіно та фотографії, а потім перейдемо до їхнього застосування у військовій сфері.

БПЛА зробили революцію в кінематографі та фотозйомці, надаючи нові унікальні повітряні перспективи, які раніше були недоступними або дорогими. Вони пропонують режисерам і фотографам універсальні та динамічні кадри, покращуючи оповідь і візуальну творчість завдяки своєму компактному розміру, маневреності та вдосконаленим можливостям камери.

У військовій сфері дрони служать як для спостереження, так і для наступальних операцій. Наприклад, FPV-дрони з видом від першої особи [6], які зазвичай використовуються для зйомок перегонів, різноманітних відео або фільмів, були модифіковані за допомогою саморобних вибухових пристроїв для нанесення точних ударів по нерухомих цілях.

На жаль, технологія безпілотних літальних апаратів також широко використовується російськими окупаційними військами. Але від них існують деякі засоби захисту. Одним з ефективних рішень є система протидії безпілотникам (Anti Drone System), яка забезпечує комплексний захист від атак безпілотників за допомогою багатошарової та багатосенсорної архітектури. Вона

включає такі модулі, як радіолокаційне виявлення безпілотників, відеоідентифікація і відстеження, радіолокаційне виявлення автономних безпілотників, об’єднання даних і командний центр, радіочастотне глушіння безпілотників, а також варіанти жорсткого знищення, такі як нейтралізація на основі кінетики і захоплення безпілотників на основі мережі для одиночних непідконтрольних безпілотних літальних апаратів [7].

Розуміння і посилення ролі безпілотних технологій в сфері оборони України має вирішальне значення, особливо з огляду на те, що вони набувають все більшого значення у протидії загрозам російського вторгнення.

Для покращення алгоритмів роботи БПЛА буде корисно реалізувати технологію, яка поєднує в собі такі інструменти й системи, як ігровий рушій для створення реалістичних віртуальних середовищ, симулятор БПЛА, а також буде раціонально пустити в дію навчену нейронну мережу, що теж принесе вклад у проведення експериментів з машинним зором дрона. На рис. 1.1 зображено мотиваційне поєднання декількох окремо по собі корисних систем та інструментів задля отримання однієї вдосконаленої технології для вивчення, покращення і тестування поведінки роботи БПЛА у віртуальних умовах, щоб в подальшому це допомогло у автоматизації навігації безпілотників в реальних умовах.

Рис. 1.1. Мета поєднання інструментів

Отже, актуальність даного дослідження очевидна, оскільки застосування новітніх технологій для навчання і вдосконалення безпілотників може значно посилити ефективність оборонних заходів та сприяти зміцненню обороноздатності України.

1.4. Переваги використання віртуальних середовищ та нейронних мереж для розв'язання проблеми

Перш за все, можна зазначити, що найважливішою перевагою використання віртуальних середовищ для вивчення в них проблем локальної автономної навігації БПЛА є те, що віртуальне середовище надає розробникам можливість легко симулювати поведінку безпілотного літального апарату, не маючи потрібного обладнання, яке може дорого коштувати. Тому замість того, щоб проводити дорогі та складні експерименти у реальних умовах, розробники можуть використовувати імітаційні моделі навколишнього середовища для швидкого та безпечного тестування своїх нових ідей та концепцій.

Крім того, використання віртуальних середовищ дозволяє здійснювати безліч варіантів тестування та моделювання різних сценаріїв, зменшуючи при цьому ризики та витрати. Це надасть можливість тестувати поведінку дрона на різних мапах в різноманітних умовах, включаючи випадки екстремальних умов або небезпеки, які можуть бути складно, небезпечно чи навіть неможливо відтворити в реальному житті.

Також використання нейронних мереж у поєднанні з віртуальними середовищами може бути корисним і значно покращити автономну навігацію БПЛА, дозволяючи безпілотникам вчитися та адаптуватися до різноманітних умов та ситуацій на основі накопичених даних та досвіду. Такий підхід сприяє розвитку більш надійних та ефективних систем управління безпілотними літальними апаратами.

Використання віртуальних середовищ разом з нейронними мережами допоможе вирішенню питання автономної навігації безпілотних літальних

апаратів. Їхнє поєднання відкриває нові можливості для розробників, тестувальників і дослідників у сфері розвитку та удосконалення систем БПЛА.

Розглянемо приклади застосування розробленої сцени симуляції середовища. Одним з таких прикладів користування розробленим віртуальним середовищем буде виконання БПЛА завдання пошуку, виявлення своєю камерою різних типів об’єктів, а також людських фігур. Це тільки один з прикладів використання, який в подальшому буде реалізований в роботі. Завдяки використанню нейронної мережи і методів машинного зору, можна реалізувати будь-які алгоритми локальної навігації БПЛА.

Наприклад, застосовуючи бібліотеку OpenCV, можна застосовувати простий, але ефективний метод виявлення людей під назвою HOG (Histograms of Oriented Gradients — гістограми орієнтованих градієнтів) [8]. Метод працює з камерою комп’ютера, на якому проводиться дослідження, але також можливо працювати з цим методом через інші підключені реальні чи навіть віртуальні камери, що і буде реалізовано в роботі. Цей метод спеціалізується на виявленні пішоходів, які переважно стоять і легко спостерігаються. Тому варто мати на увазі, що його ефективність може бути обмеженою в інших сценаріях, бо це доволі примітивна і спрощена модель. Але також існують інші методи виявлення людей та інші статичних і динамічних об’єктів.

Попередньо навчена нейронна мережа You Only Look Once (YOLO) може допомогти набагато легше впоратися з задачею виявлення та ідентифікації різноманітних об’єктів у реальному часі, а не тільки виявлення людей. Алгоритм YOLO спочатку отримує зображення на вході, а потім використовує просту глибоку згорткову нейронну мережу для виявлення об’єктів на зображенні [9]. Одна з простих моделей YOLO навчена розпізнавати близько 80 різних типів об’єктів. Приклад очікуваного принципу роботи зображено на рис. 1.2, де ми бачимо, що деякі об’єкти виділені жовтими рамками і також позначені відповідними назвами. Наведений приклад є приблизним очікуваним результатом, який буде отримано з камери віртуального дрона при тестуванні його у розробленому симульованому середовищі, яке буде імітувати реалістичні

об’єкти й умови. Тобто, коли у поле зору камери безпілотного літального апарату будуть потрапляти якісь об’єкти, які попередньо навчена модель нейронної мережі може розпізнавати, вони будуть у режимі реального часу виділятися на зображенні, переданого з камери дрона, рамкою та назвою, яка найбільш точно відповідає знайденому і виявленому об’єкту.

Рис.1.2. Очікуваний принцип роботи YOLO

Нейронну мережу YOLO можна легко інтегрувати в обране середовище розробки алгоритмів локальної навігації БПЛА, налаштувати усі потрібні для її роботи бібліотеки, а далі вже підготувати все для отримання зображення з камери симульованого дрона у створеному віртуальному середовищі. Це дозволить подальше тестування якості роботи функцій машинного зору БПЛА і вдосконалення поведінки дрона у створеному віртуальному середовищі.

1.5. Висновки до розділу 1

В результаті проведеного аналітичного огляду теми предметної області було розглянуто та досліджено важливі аспекти навігації безпілотних літальних апаратів, а саме локальної навігації БПЛА у віртуальному середовищі, досліджено і обґрунтовано актуальність обраної теми роботи. Також було

проаналізовано переваги і деякі недоліки використання віртуальних середовищ та нейронних мереж для вивчення і вдосконалення алгоритмів навігації БПЛА.

Підсумовуючи, можна відзначити, що наявність автономної навігації БПЛА має великий потенціал у багатьох сферах, особливу користь автономна навігація принесе у військовій сфері, проте розробка таких технологій буде вимагати значних зусиль.

Створення зручних віртуальних середовищ дозволить ефективно тестувати та покращувати алгоритми навігації без ризику втрат та великих витрат. Поєднання цих середовищ з нейронними мережами та іншими технологіями відкриває нові можливості для створення більш надійних та ефективних систем управління безпілотними літальними апаратами.

Метою даного розділу було розібратися в навігації безпілотних літальних апаратів, оцінити важливість і корисність використання віртуальних середовищ для вивчення проблем автономної навігації БПЛА, а також розглянути підходи до побудови таких імітаційних середовищ та інтеграцію нейронних мереж в них. Поставлена мета була досягнута шляхом проведення аналітичного огляду і дослідження безпілотних літальних апаратів, їх особливостей, компонентів та роботи навігації. Окрім цього, були запропоновані можливі ідеї використання готового розробленого віртуального середовища.

Робота має великий потенціал у подальшому розвитку та застосуванні, забезпечуючи зручність та безпеку користування безпілотними літальними апаратами під час навчання їх новим функціям і алгоритмам. Отже, використання віртуальних середовищ та нейронних мереж для розв’язання проблем навігації БПЛА відкриває нові перспективи для вдосконалення та розвитку цієї технології, що є важливим кроком у напрямку створення більш безпечних, надійних та ефективних систем управління безпілотними апаратами.

РОЗДІЛ 2
ВИБІР ЗАСОБІВ ТА ІНСТРУМЕНТІВ РОЗРОБКИ

Перед початком розробки віртуальних середовищ, для проведення там дослідження локальної навігації БПЛА, важливо спочатку детально розглянути головні вимоги до розробки майбутньої технології, щоб обрати потрібні для роботи інструменти, середовище розробки та системи для побудови симуляції реальності, які найкраще підходять для реалізації даної роботи. Також необхідно проаналізувати переваги та недоліки можливих методів та засобів, таких як технології для розробки віртуальних середовищ, мови програмування для створення алгоритмів навігації БПЛА, середовища для розробки та тестування вже розроблених компонентів технології. Зокрема, будуть обрані декілька варіантів інструментів, які є найоптимальнішими для виконання поставленого завдання, після цього буде розглянуто їхні основні характеристики, сфери застосування, тенденції в розвитку цих технологій та перспективи майбутнього застосування. Наступний етапом буде проведення порівняльного аналізу обраних варіантів та обґрунтовано вибір відповідних засобів та інструментів.

Отже, на початковому етапі аналізу необхідно врахувати різні аспекти майбутньої розробки даної технології. Наприклад, розглянути такі важливі деталі для розробки, як масштабованість, інтеграційні можливості, ефективність та безпека. Також важливим є розгляд апаратних вимог та можливостей, що можуть сильно вплинути на продуктивність роботи розробленого віртуального середовища. При виборі технологій слід обов’язково звернути увагу на їхню підтримку з боку спільноти розробників та наявність документації, що полегшить процес розробки та розв’язання проблем, які можуть виникнути в процесі розробки.

Окрім цього, особливу увагу потрібно приділити вибору мов програмування, бо саме вони будуть використовуватися для реалізації всіх алгоритмів навігації літальних апаратів. Обрані мови програмування мають ефективно обробляти дані у реальному часі та мати всі потрібні бібліотеки для роботи з віртуальними середовищами, машинним зором та нейронними мережами. Також важливо, щоб для обробки великих обсягів даних та складних обчислювальних завдань, мови програмування підтримували багатопоточність і паралельні обчислення. Інтегровані середовища для розробки повинні забезпечувати можливість зручно і легко моделювати різні механіки і сценарії польотів дрону. Це дозволить вдосконалити ефективність оцінювання роботи алгоритмів навігації БПЛА та вносити необхідні зміни в процесі розробки.

Після проведення аналізу всіх вимог та вибору найоптимальніших варіантів технологій, необхідно провести їхнє детальне дослідження, враховуючи і власний досвід попередніх розробок з використанням обраних інструментів. Також не буде зайвим розглянути наявні дослідження та приклади успішного використання тих чи інших засобів розробки, що допоможе визначити їхні сильні та слабкі сторони.

У підсумку, буде проведено загальний порівняльний аналіз всіх обраних інструментів за такими критеріями, як зручність використання, продуктивність, надійність та підтримка. На основі проведеного аналізу буде обґрунтовано вибір засобів та інструментів розробки, які найбільш ефективно допоможуть реалізувати поставлене завдання з моделювання віртуальних середовищ для дослідження автономної навігації БПЛА.

2.1. Аналіз вимог до технології

Технологія, перш за все, має допомогти у розробці автономної навігації для безпілотних літальних апаратів. Це вимагає створення ефективного, надійного і зручного інструменту, який забезпечить можливість тестування та вдосконалення алгоритмів навігації у віртуальних середовищах, наближених до реальних умов. Всі загальні вимоги до технології можна поділити на дві основні категорії:

функціональні та нефункціональні вимоги. Кожна з цих категорій охоплює важливі аспекти, необхідні для успішного виконання завдань локальної навігації БПЛА.

Дослідження автономної навігації для безпілотних літальних апаратів є досить складним і різнобічним завданням, що вимагає ретельного підходу до розробки та тестування технології. Віртуальні середовища відіграють вирішальну роль у цьому дослідженні, оскільки вони дозволяють моделювати різноманітні мапи для проведення експериментів сценаріїв польотів, тестувати автоматизовані алгоритми навігації та аналізувати їхню ефективність у безпечному та контрольованому створеному середовищі. Основна мета розроблюваної технології — забезпечити виконання поставлених завдань локальної автономної навігації БПЛА, надаючи всі необхідні інструменти для розробників.

Загальною вимогою до системи першочергово є забезпечення ефективної розробки та тестування алгоритмів автономної навігації для безпілотних літальних апаратів. Однією з важливих цілей технології є також створення реалістичного віртуального середовища, яке моделює умови реального світу, включаючи різні перешкоди та інші фактори, що можуть вплинути на політ БПЛА. Розроблена технологія має бути зручною для користувача, забезпечуючи інтуїтивно зрозуміле користування всіма її компонентами та легкість у використанні для осіб з різним рівнем технічної підготовки.

Окрім цього, технологія має бути гнучкою і масштабованою, що дозволить в подальшому продовжувати розвивати її функціонал у майбутньому відповідно до змін вимог та появи нових задач навігації. Надійність і стабільність роботи системи є надзвичайно важливими, оскільки від цього залежить успіх тестування алгоритмів локальної навігації БПЛА.

Пізніше у наступних розділах будуть більш детально розглянуті вимоги до технології за двома згаданими раніше характеристиками. Це дозволить скласти повну картину всіх необхідних характеристик та функцій технології, які забезпечать розв’язання поставлених задач і досягнення цілей роботи.

2.1.1. Функціональні вимоги

Функціональні вимоги (Functional Requirements) — це такі вимоги до системи, продукту або технології, які визначають її внутрішню роботу і описують її можливості, а також які основні функції вона має виконувати, щоб задовольнити всі потреби користувачів [10].

Опис функціональних вимог до структури технології передбачає детальне визначення необхідних функцій та можливостей, які технологія повинна забезпечувати для виконання завдань автономної локальної навігації безпілотних літальних апаратів. Ці вимоги формують основу для розробки та визначають основні компоненти та їх взаємодію.

Однією з головних функціональних вимог є можливість реалістичного моделювання симуляції зовнішнього середовища, обов'язкова наявність у обраній системі побудови віртуальних середовищ підтримки інтеграції технологій для симуляції поведінки БПЛА та технологій програмування для створення своїх алгоритмів локальної навігації. Модель віртуального середовища має включати різноманітні перешкоди, як дерева, дороги, будівлі, ліхтарі та інші об'єкти, а також динамічні об'єкти, на яких можна буде перевіряти і тренувати машинний зір БПЛА. Це дозволить перевіряти алгоритми локальної навігації польотів у більш реалістичних умовах для дрона, наближених до реальних, що допоможе покращенню точності та надійності алгоритмів. Також розроблена модель має забезпечувати налаштування сценаріїв, маршрутів та стартових точок польотів дрону.

Вказані вище функціональні вимоги створюють головну основу для розробки комплексної, ефективної та новітньої технології, яка допоможе у розробці, тестуванні та покращенні алгоритмів локальної автономної навігації безпілотних літальних апаратів, забезпечуючи при цьому зручність і надійність використання.

2.1.2. Нефункціональні вимоги

Нефункціональні вимоги (Non-Functional Requirements) — це набір вимог та обмежень до системи, які створюються задля покращення її ефективності [11]. На відміну від функціональних вимог, вони описують не які функції система має виконувати, а як саме вона їх має виконувати. Такі вимоги до системи відносяться до її зручності, стабільності та продуктивності роботи, масштабованості, сумісності з іншими системами, а також інших параметрів роботи.

Сумісність визначає, як саме технологія може співіснувати з різними операційними системами, програмним забезпеченням, додатками, браузерами та іншими системами в одному середовищі. Вимоги до даної технології передбачають сумісність з різними операційними системами, такими як Windows і Linux, сумісність з інструментами та бібліотеками для роботи з методами машинного зору та навігацією БПЛА.

Продуктивність характеризує те, наскільки технологія та всі її компоненти мають бути здатні продовжувати швидко працювати і як реагувати при певному навантаженні та при різних діях користувачів. Розроблена технологія повинна бути оптимізована для продуктивної швидкої обробки потенційно великих обсягів даних.

Масштабованість передбачає можливість розширення функціоналу будь-яких компонентів технології без значних змін у її базовій архітектурі, тобто легке додавання нових функцій без сильного впливу на її продуктивність.

Стабільність та надійність технології визначається стабільною і надійною роботою без частих збоїв.

Зручність використання полягає у тому, наскільки користувачам буде легко та інтуїтивно зрозуміло користуватися готовою технологією, чи легко буде навчитися користуватися нею. Тому користування розробленою технологією має бути просте, зрозуміле та зручне для користувачів з будь-яким рівнем технічної підготовки.

2.2. Системи побудови віртуальних середовищ

Існує багато різних широко і вузькоспрямованих систем для побудови віртуальних середовищ, але ми зупинимося на більш придатних для даного проєкту варіантах. Найбільш відомими й розвинутими ігровими рушіями для розробників є Unity та Unreal Engine. Тому розглянемо ці два рушії більш детально, проаналізувавши їх переваги та недоліки й порівнявши їх між собою. Спершу почнемо з загальних характеристик обох рушіїв, потім розглянемо більш детально, як влаштована внутрішня структура систем, порівняємо їх та виберемо найкращий варіант з цих двох.

2.2.1. Характеристика рушія Unity

Спочатку дослідимо характеристики Unity від компанії Unity Technologies, проаналізуємо його слабкі та сильні сторони та визначимо, наскільки ця система підходить для роботи. Перш за все, важливо зазначити, що цей ігровий рушій дуже відомий своєї зручністю і легкістю використання для тих, хто тільки починає ознайомлюватися з розробкою ігор і взагалі не мав жодного досвіду. Unity має достатньо навчальних матеріалів і документації, що допоможе швидко опанувати основи роботи з даним рушієм. І завдяки великій спільноті, користувачі Unity мають доступ до численних ресурсів, форумів та корисних відеоуроків. Це теж значно спрощує процес навчання і вирішення можливих проблем. Також рушій є дуже універсальним, має широкий набір вбудованих інструментів та підходить для розробки додатків та ігор будь-яких жанрів, форматів та для різних платформ. Розробники мають можливість створити мультиплатформений продукт для комп’ютерів, мобільних пристроїв, консолей, а також для віртуальної реальності.

Unity використовує мову програмування C# (C-Sharp), що дозволяє створювати складні ігрові механіки і логіку. C# є досить популярною мовою для вивчення програмування, що робить її доступною для багатьох розробників. Але в подальшому необхідно буде також дослідити, чи підходить ця мова для створення алгоритмів локальної навігації БПЛА.

Розробка в Unity не є ресурсозатратною і не вимагає багато коштів, щоб отримати на виході хороший продукт. Дуже корисним є те, що Unity має свій власний магазин Asset Store, де розробники можуть знайти, купити або навіть безкоштовно завантажити різноманітні ресурси для свого проєкту.

Підсумовуючи переваги, Unity є потужним і гнучким інструментом для розробки ігор та підходить для моделювання віртуальних середовищ, що робить його хорошим вибором для розробляємого проєкту. Однак, як і будь-який інший інструмент, рушій Unity все ж таки має свої обмеження і недоліки, тому обов'язково варто розглянути і їх. Хоча Unity випускають нові інструменти і оновлюють свою систему, ці оновлення часто виходять недопрацьованими і мають багато багів, деякі з яких можуть сильно заважати роботі. Також ці нові функції іноді є взагалі непотрібними і не несуть якоїсь користі, особливо якщо вони можуть погіршити роботу інший важливих функцій. Важливим недоліком є те, інтеграція сторонніх інструментів і плагінів може іноді викликати проблеми зі сумісністю або стабільністю роботи рушія, що ставить під питання можливість зручної інтеграції симулятора БПЛА та використання нейронної мережі у проєкті, якщо він буде створений в Unity. Але окрім цього, головним недоліком є те, що Unity є рушієм з закритим вихідним кодом, що може значно ускладнити розробку середовища. Бо закритий вихідний код сильно обмежує можливості, щоб змінювати або оптимізувати базові компоненти рушія під специфічні потреби свого проєкту, також закритий вихідний код може викликати питання щодо безпеки, оскільки буде важче самостійно перевірити, чи не містить рушій вразливостей.

2.2.2. Характеристика рушія Unreal Engine

Тепер для порівняння розглянемо характеристики Unreal Engine, розробниками якого є Epic Games. Unreal Engine є досить популярною програмою для створення ігор та інших проєктів. Важливо одразу зазначити, що, на відміну від Unity, Unreal Engine надає доступ до свого вихідного коду, що дає розробникам більше гнучкості та контролю над проєктом. Це спрощує

діагностику, виправлення помилок та оптимізацію продуктивності проєкту. Окрім цього, Unreal Engine відомий своєю високою якістю графіки, бо рушій підтримує детальні текстури й освітлення, фотореалістичну графіку і надає всі можливості для високої якості візуалізації, яку потребує робота.

Unreal Engine має свою дуже зручну візуальну мову програмування Blueprints Visual Scripting, що дозволяє створювати складні ігрові механіки, випробовувати нові ідеї та рішення взагалі без написання коду. Blueprints пропонують широкий набір різних інструментів для розробки, до яких є документація з користування ними, це робить даний рушій доступним для всіх користувачів, які не мають достатнього досвіду роботи з мовами програмування, і це надає велику перевагу вибору саме Unreal Engine. UE підтримує також використання мов програмування, основною мовою є C++, поєднання Blueprints з мовою програмування C++ можу бути надзвичайно зручним і корисним для розв’язання будь-яких задач проєкту. Також Unreal Engine містить численні інструменти для оптимізації продуктивності продукту, що дозволяє розробникам створювати високоякісні ігри навіть для платформ з обмеженими ресурсами. Рушій має кращу підтримку інтеграції та сумісність зі сторонніми технологіями. Unreal Engine, як і Unity, має велику спільноту розробників та навіть ще більше різних ресурсів для самонавчання, документації з користування, безліч відеоуроків та плагінів для вирішення багатьох проблем.

Unreal Engine теж має свій власний Marketplace, де розробники можуть купувати і продавати різні 3D-моделі та 2D-моделі, анімації, звукові ефекти, шаблони та плагіни, але авжеж у доступі є достатньо безкоштовних моделей. Це значно прискорює процес розробки, дозволяючи використовувати готові ресурси, що може стати корисним для розробки свого віртуального середовища. Unreal Engine є лідером з рендерингу в реальному часі, що робить його ідеальним для створення інтерактивних візуалізацій, віртуальних зйомок та інших проєктів, де потрібен швидкий відгук в реальному часі і високоякісна графіка.

Ці сильні сторони Unreal Engine роблять цей рушій привабливим вибором для розробників різних проєктів та особливо для тих, хто працює над більш

великими проєктами або потребує високоякісної графіки і широких можливостей кастомізації. Але також розглянемо, які є слабкі сторони Unreal Engine, і проаналізуємо їх. По-перше, скільки б переваг не мав би відкритий код, у нього все одно є свої недоліки — він може містити якісь баги або бути недопрацьованим. Хоча Unreal Engine пропонує багато потужних інструментів і використання зручних Blueprints, рушій може бути складнішим для освоєння, особливо для новачків, бо для деяких рішень і повноцінного використання всіх можливостей UE може знадобитися застосування мови програмування C++, а вивчення цієї мови та розуміння всіх можливостей рушія може вимагати значного часу та зусиль. UE потребує хорошого апаратного забезпечення для оптимальної роботи, особливо під час розробки великих і складних проєктів, але це не є критичним недоліком рушія. Якщо порівнювати з Unity, розмір проєктів створених в Unreal Engine є більшим і може швидко зростати в обсязі, тому це може в подальшому ускладнити процес внесення змін. Наведені недоліки варто враховувати при виборі рушія для роботи, щоб уникнути непередбачених проблем під час розробки. Але, незважаючи на свої недоліки, Unreal Engine все ще залишається потужним інструментом для створення високоякісних проєктів.

2.2.3. Внутрішня структура та архітектура обох систем

Окрім розглянутих основних головних характеристик рушіїв, має сенс розглянути та порівняти інтерфейс, а також роздивитися детальніше внутрішню структуру роботи обох систем. Unity і Unreal Engine мають різні підходи до архітектури і внутрішніх механізмів, що впливає на процес розробки і продуктивність. Unity використовує компонентно-орієнтовану архітектуру, де кожен об’єкт складається з набору компонентів, це дозволяє створювати гнучкі і модульні об’єкти. Unreal Engine ж використовує іншу архітектуру, а саме — об’єктно-орієнтовану архітектуру з підтримкою класів і наслідування, що дозволяє створювати складні ієрархії об’єктів. Як вже було зазначено, Unity використовує мову C# для написання своїх скриптів, що прикріплюються до об’єктів як компоненти, дозволяючи легко керувати їхньою поведінкою. Unreal

Engine підтримує як C++, так і візуальне програмування через Blueprints, що робить розробку більш доступною завдяки використанню обох методів. Unity організовує ігрові рівні та об’єкти за допомогою менеджера сцен, що спрощує управління ресурсами і оптимізацію продуктивності. Unreal Engine організовує віртуальні середовища за допомогою системи рівнів, які можуть бути завантажені і розвантажені під час гри, що ефективно керує ресурсами проєкту. Ці архітектурні відмінності систем впливають на процес розробки і продуктивність, визначаючи вибір рушія в залежності від потреб проєкту.

На рис. 2.1 зображено візуальне порівняння всіх основних елементів інтерфейсів Unity та Unreal Engine 5. Можна помітити, що деякі області в інтерфейсі кожного з рушіїв позначено однаковим кольором, це вказує на спільну функціональність цих областей. Кожен елемент інтерфейсу підписаний, щоб показати термінологію, яку використовують обидва рушії. Макет інтерфейсу редакторів повністю налаштовується, можна змінювати місцями вкладки та приховувати їх.

Рис. 2.1. Порівняння елементів інтерфейсу Unity та Unreal Engine

Внутрішня структура системи Unity технічно налаштована таким чином, як можна побачити на рис. 2.2, що означає, що кожен об’єкт у проєкті складається з набору компонентів, кожен з яких додає певну функціональність. Компонентно-орієнтована архітектура включає в себе GameObject, Components, Scripts та Scene. GameObject — це основний об’єкт у сцені, до якого додаються компоненти. Components — це такі модулі, які називаються компонентами і

додаються до GameObject для визначення його поведінки та властивостей, це можуть бути скрипти, фізичні компоненти, звукові ефекти. Scripts — це написані на мові C# компоненти, вони додаються до GameObject для керування логікою об’єкта. Scene — це контейнер, який містить всі додані GameObject з їх компонентами. На малюнку наведені лише деякі приклади з існуючих компонентів, ігровий об’єкт може мати в собі більше компонентів, які змінюють його різні параметри.

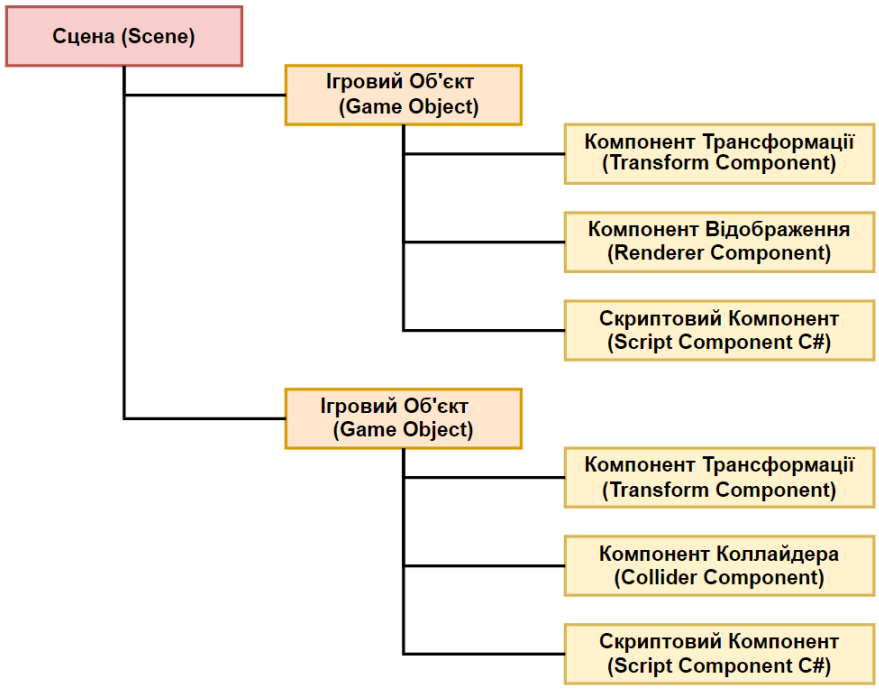

Рис. 2.2. Внутрішня структура системи Unity

Для порівняння на рис. 2.3 зображена технічна концепція внутрішньої структури Unreal Engine, щоб зрозуміти, як влаштовані обидві системи. UE використовує об’єктно-орієнтовану архітектуру, де кожен об’єкт є екземпляром класу, що дозволяє створювати складні ієрархії об’єктів. Actor — це базовий клас для всіх об’єктів у сцені, з якого можуть наслідувати інші об’єкти. Components — це модулі, які додаються до Actor для визначення, змінення його поведінки та властивостей. C++ Classes — основний спосіб написання логіки та управління об’єктами. Blueprints — візуальна система скриптування, яка дозволяє створювати логіку без написання коду. Level — контейнер, який містить Actor і їх компоненти. Як можна відмітити, внутрішня структура обох систем і процес

створення віртуальних середовищ в цих двох рушіях приблизно схожі, однак назви компонентів структури і вигляд інтерфейсів відрізняються.

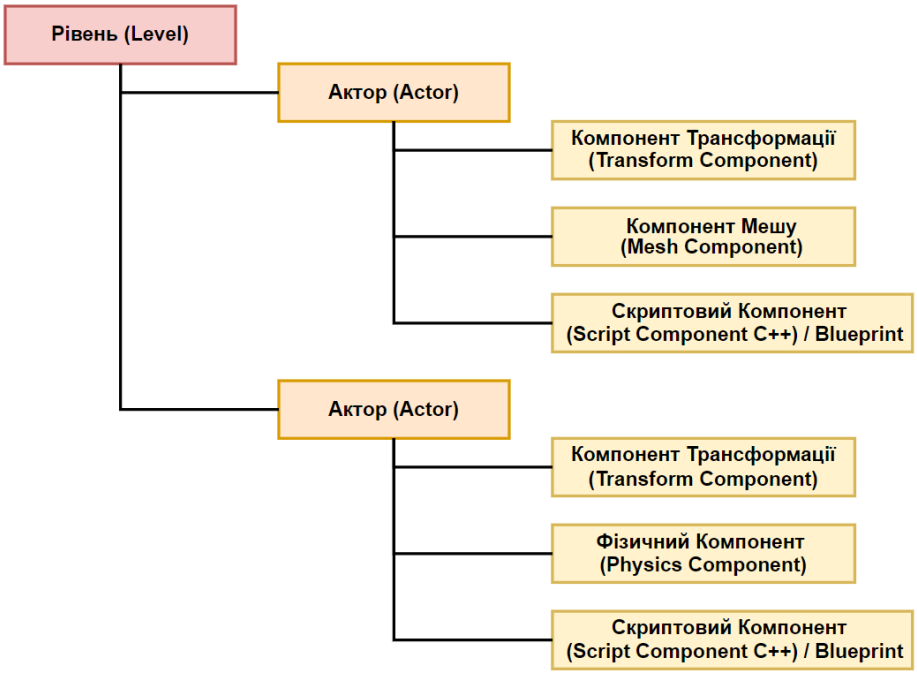

Рис. 2.3. Внутрішня структура системи Unreal Engine

В Unity, як і в Unreal Engine, існує набагато більше компонентів, які можна використовувати для різних цілей. Ці компоненти дозволяють розробникам створювати більш складні і багатофункціональні ігрові об’єкти, адаптуючи їх під всі потреби проєкту. У наведених вище схемах були показані лише деякі приклади ігрових об'єктів або акторів та їх компонентів.

Можна зробити висновки, що обидва рушія є гарними і зручними інструментами, добре підходять до розробки різних проєктів і можуть прекрасно виконувати майже будь-які поставлені задачі. Але відповідно до завдань цього проєкту, для створення віртуального середовища буде раціональніше обрати саме ігровий рушій Unreal Engine, бо він пропонує високу якість графіки, потужні інструменти для роботи з реалістичними 3D-моделями, підтримує візуальне програмування за допомогою Blueprints, має відкритий вихідний код, також має кращу підтримку інтеграції інших програм і систем.

2.3. Симулятор БПЛА AirSim

AirSim — це симулятор розроблений компанією Microsoft з відкритим вихідним кодом для дронів та автомобілів, він побудований на Unreal Engine, з експериментальною підтримкою Unity [12]. Він є кросплатформним і підтримує програмну симуляцію з популярними контролерами для управління польотів. Симулятор розроблений як плагін для Unreal Engine та Unity, який можна запросто додати в будь-яке створене віртуальне середовище. AirSim був створений як зручна симуляційна платформа для дослідження штучного інтелекту в роботі з автономними літальними апаратами та автомобілями, він надає можливість проводити експерименти з алгоритмами комп'ютерного зору та навчання з підкріпленням для автономних транспортних засобів.

Симулятор дає можливість керувати дроном вручну за допомогою пульта дистанційного керування, але також AirSim надає API для взаємодії з транспортним засобом у симуляції програмно, ці API можна використовувати для отримання різних зображень, визначення стану симуляційної моделі засобу або задля керуванням ним. API надаються через Remote Procedure Call (RPC) і доступні за допомогою різних мов, включаючи C++, Python, C# та Java. Remote Procedure Call — це технологія, яка дозволяє програмі виконувати код на віддаленій системі так, ніби він виконувався локально, у випадку користуванням AirSim RPC використовується для взаємодії між користувачем і сервером симулятора. Завдяки тому, що API є частиною окремої незалежної крос-платформної бібліотеки, є можливість розгорнути їх на екрані пульта управління дроном або бортовому комп'ютері автомобіля. Таким чином, можна спочатку розробляти та тестувати код на симуляторі, а потім запускати його на реальних транспортних засобах. Рис. 2.4 демонструє архітектуру AirSim та роботу всіх основних компонентів симулятора між собою. Схема показує, як симулятор взаємодіє з різними компонентами для забезпечення навігації. Бортовий комп'ютер надсилає задані бажані стани до API частини симулятора, потім він отримує оцінені стани та дані від датчиків і сприйняття. Контролер польоту надсилає дані до API та до моделі транспортного засобу і отримує дані від моделі

датчика. API забезпечує інтерфейс між двома зовнішніми системами та всіма внутрішніми компонентами симулятора AirSim, передаючи бажані та оцінені стани, дані датчиків та сприйняття. Модель датчика генерує дані датчиків на основі переданих їй кінематичних параметрів з фізичного рушія, також передає дані датчиків до API та контролера польоту. Модель транспортного засобу отримує сигнали приводу від контролера польоту, які далі використовується для моделювання поведінки транспортного засобу у середовищі. Рендеринг рушій відповідає за візуалізацію стану симуляції, генеруючи дані сприйняття на основі положення транспортного засобу. Фізичний рушій обчислює фізичні взаємодії, такі як різні фізичні сили і моменти, що діють на транспортний засіб, передає кінематичні дані до моделі датчика та моделі транспортного засобу. Модель середовища визначає параметри навколишнього середовища, такі як гравітація, тиск повітря та інші важливі параметри, що впливають на фізичний рушій. Ця структура дозволяє симулятору надавати реалістичні дані для навігації БПЛА та інших транспортних засобів, тестуючи і налагоджуючи алгоритми навігації та управління в безпечному і контрольованому віртуальному середовищі.

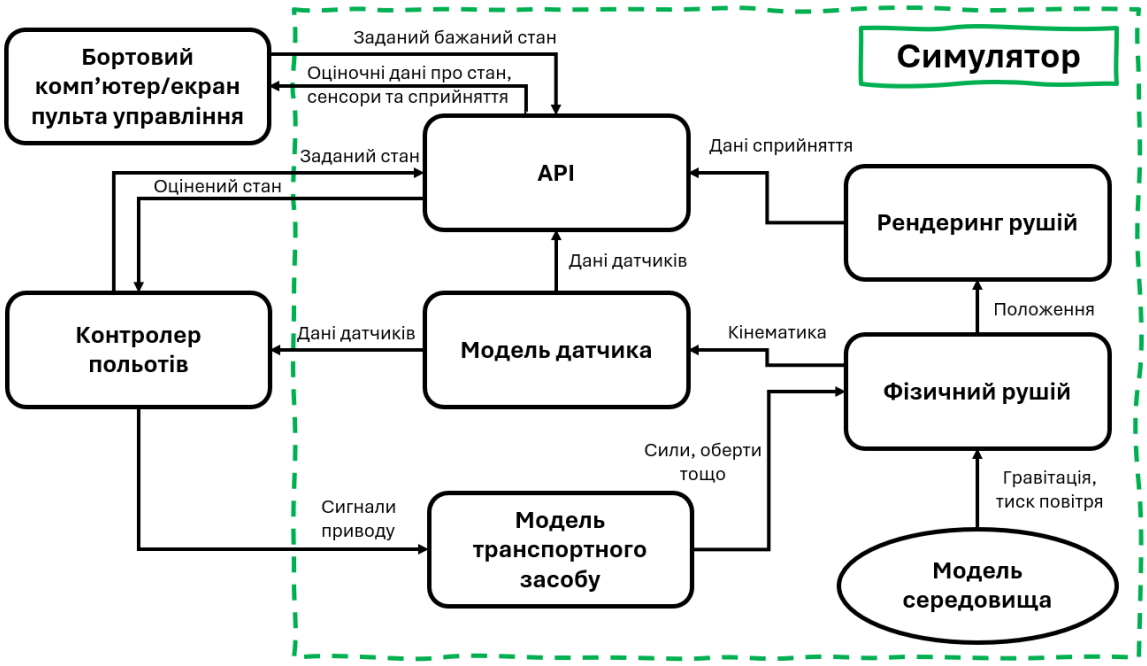

Рис. 2.4. Компоненти архітектури AirSim та їх взаємодія

Тепер розглянемо наглядний принцип роботи AirSim у середовищі. На рис. 2.5 зображено, як виглядає приклад симуляції БПЛА з використанням AirSim у віртуальному середовищі. Тут можна побачити в лівому верхньому кутку зеленим текстом використання симуляції Hardware-in-the-loop (HIL), при цьому методі реальні апаратні компоненти інтегруються з симуляційним середовищем для забезпечення реалістичного тестування та налагодження. Hardware-in-the-loop наглядно показує, що симуляція відбувається у реальному часі та дозволяє взаємодіяти з дроном вручну за допомогою пульта керування в умовах віртуальної симуляції. Вона показує, як літальний апарат реагує на всі дії в реальному часі та як апаратний контролер дрону сприймає віртуальні умови як реальне зовнішнє середовище.

Також на рисунку видно три зображення з камери безпілотного літального апарату. Ліва камера демонструє зображення з камери глибини в реальному часі, вона відображає інформацію про відстань до об’єктів у сцені, використовуючи різні відтінки сірого кольору. Камера посередині представляє зображення з сегментаційної камери, де кожен піксель на отриманому зображенні пофарбований у колір, який відповідає його класу об’єкта, наприклад будівля, дорога, дерева, люди та інші. Ця камера використовується для розпізнавання і класифікації об’єктів у середовищі. І остання, права камера демонструє звичайне зображення, яке показує сцену у віртуальному середовищі так, як її бачить звичайна камера. Вона використовується для візуального сприйняття та аналізу середовища дроном. Ці три зображення надають різні перспективи, корисні для завдань в дослідженні і розробці алгоритмів локальної навігації, дозволяючи БПЛА орієнтуватися у просторі, розпізнавати різні типи об’єктів та приймати рішення на основі отриманих даних.

Рис. 2.5. Роботи симулятора AirSim на прикладі дрона

Оригінальна версія AirSim була архівована і розробники перестали випускати подальші оновлення для симулятора і взагалі підтримувати минулі версії через те, що цей проєкт вже виконав свою мету, а також через зосередження всіх зусиль на розробці нового проєкту, який представляє собою нову симуляційну платформу, що буде розвиватися в аерокосмічному напрямку. Але велика спільнота користувачів все ще продовжує розвивати AirSim, розробляти нові ідеї та оновлювати симулятор для можливості інтегрувати його в новіші версії ігрових рушіїв та для різних операційних систем.

Однією з таких оновлених версій AirSim, яка зараз підтримується і розвивається спільнотою користувачів, є Colosseum. Colosseum — це новий репозиторій, що поділяє код і налаштування видимості оригінального репозиторію AirSim, який Microsoft вирішила закрити у 2022 році. Розробником цього оновленого репозиторію Colosseum є компанія Codex Laboratories LLC. Він зроблений для створення нової, кращої платформи для симуляції, експериментування і внесення цікавих змін у AirSim. Так само, як і оригінальна версія, Colosseum працює як плагін для двох популярних рушіїв Unreal Engine та Unity, але версія для Unity все ще експериментальна і недосконало працює. Це надає ще одну перевагу для використання саме Unreal Engine як більш стабільної

і надійної системи побудови віртуального середовища. Colosseum найкраще підтримує операційну систему Windows, але є також підтримка деяких версій Linux та MacOS, що робить його універсальним рішенням для різних операційних систем. Окрім цього, є інші корисні характеристики Colosseum та переваги використання цього симулятора для дослідження локальної навігації БПЛА. Colosseum розширює функціонал оригінального AirSim, надаючи більше можливостей для налаштування симуляційних середовищ, що робить його підходящим інструментом для дослідження і тестування алгоритмів навігації дронів у віртуальному середовищі. Спільнота користувачів активно підтримує і розвиває Colosseum, вдосконалюючи цю платформу під різноманітні проєкти, використовуючи всі можливості відкритого вихідного коду.

Підсумовуючи, Colosseum є потужною і гнучкою платформою для симуляції різних транспортних засобів, тому симулятор ідеально підходить і для дослідження локальної навігації БПЛА. Саме завдяки своїм розширеним можливостям і співпрацею зі спільнотою, Colosseum забезпечує ефективне середовище для експериментів і розробки нових алгоритмів. Також вибір ігрового рушія Unreal Engine як основної системи для моделювання віртуальних середовищ додатково підсилює переваги використання Colosseum, надаючи можливість використовувати реалістичні і масштабовані симуляції, необхідні для сучасних досліджень у сфері автономних систем безпілотних літальних апаратів.

2.4. Середовище розробки та технології програмування для алгоритмів локальної навігації БПЛА

Розробка алгоритмів локальної автономної навігації для безпілотних літальних апаратів, які будуть навчатися у віртуальному середовищі, є важливою задачею, що вимагає використання ефективних і надійних мов програмування та придатного середовища розробки, які допоможуть досягти бажаного результату. Середовище розробки і мова програмування мають ідеально інтегруватися в обраний ігровий рушій Unreal Engine та зручно використовуватися для написання

алгоритмів навігації з використанням симулятора Colosseum, розробленого на основі AirSim.

Як вже було зазначено, основною мовою програмування в Unreal Engine є мова C++, вона використовується для створення високопродуктивного і ефективного коду. C++ дозволяє розробникам глибоко інтегруватися з внутрішніми системами Unreal Engine. Але окрім цього, є також можливість використання мови програмування Python для створення інструментів і автоматизації процесів всередині. Unreal Engine добре підтримує Python для скриптування завдань, що не потребують високої продуктивності. Мова Python є доволі популярним вибором для розробки різноманітних алгоритмів завдяки своїй простоті, великій кількості корисних бібліотек і потужним можливостям для роботи з даними та машинного навчання, що є великою перевагою для його використання у проєкті. Python часто використовується для швидкої прототипізації алгоритмів, аналізу даних і симуляції.

Щодо середовищ розробки, Unreal Engine має вбудовану інтеграцію з середовищем Visual Studio. Воно є основним середовищем розробки для роботи з Unreal Engine, особливо для програмування на мові C++. Visual Studio має розширення для роботи з Unreal Engine, готові шаблони для проєктів, надає потужні інструменти для налагодження, а також має інтеграцію з системами контролю версій та розширює можливості розробки за допомогою плагінів. Окрім Visual Studio, існує інше популярне середовище JetBrains Rider для розробки з використанням Unreal Engine, яке пропонує інтелектуальні інструменти коду та підтримку C++ і Blueprints. Але його недоліком є те, що у нього є безкоштовна пробна версія тільки на один місяць, в той же час Visual Studio надає безкоштовних доступ до всіх своїх можливостей.

Для розробки алгоритмів локальної навігації БПЛА з використанням симулятора Colosseum в рушії Unreal Engine, найкраще використовувати Visual Studio як основне середовище розробки та використовувати мови програмування Python та C++, кожна з яких буде застосовуватися для розв’язання конкретних завдань дослідження, а мова Blueprints в Unreal Engine буде задіяна для швидкого

прототипування. Це забезпечить ефективний і гнучкий процес розробки, дозволяючи створювати складні алгоритми навігації та зручно тестувати їх у реалістичних віртуальних умовах середовища.

2.5. Висновки до розділу 2

Підіб'ємо підсумки щодо вибору засобів та інструментів розробки. Вибір кожної конкретної технології було обґрунтовано, були розглянуті всі важливі загальні характеристики систем в порівнянні з конкурентами, детально досліджена робота їх внутрішньої структури. Обрані системи відповідають поставленим вимогам і допоможуть з розробкою ефективної і корисної технології, що буде відповідати бажаному результату і виконувати розв'язувати різні задачі.

Ігровий рушій Unreal Engine обрано через його потужність у реалізації реалістичних ігрових середовищ з високоякісною графікою, що є важливим для проєкту через його прямий зв'язок з симуляцією, а також через вбудовану підтримку інтегрованого середовища розробки Visual Studio. Airsim став вибором для симуляції руху БПЛА через його широкі можливості для реалістичної симуляції повітряних динамічних систем, що дозволяє вивчати поведінку дрона та реакцію на різні умови. Visual Studio обрано як основне середовище розробки для зручності та продуктивності у програмуванні, оскільки середовище має широкий набір інструментів для розробки на різних мовах програмування, включно з мовами C++ та Python. А також Visual Studio підтримується Unreal Engine, що надає перевагу його використанню в роботі.

Отже, всі обрані інструменти та засоби відповідають поставленим вимогам проєкту з урахуванням його потреб у графічній реалістичності, симуляції безпілотних літальних апаратів, ефективності розробки та швидкості виконання. Кожен інструмент і мова програмування призначені для конкретних завдань, що дозволяє забезпечити оптимальну розробку проєкту.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ І НАЛАШТУВАННЯ ВІРТУАЛЬНОГО СЕРЕДОВИЩА

Створення моделі віртуального середовища насамперед починається з планування розробки всіх його компонентів. Тому важливим питанням є планування зручного розробляємого середовища для дослідження алгоритмів локальної навігації БПЛА. Спочатку варто визначити основні елементи віртуального середовища, їх вигляд, взаємозв'язки та принцип роботи. Для успішної реалізації та налаштування віртуального середовища необхідно продумати метод створення ефективної реалістичної моделі середовища, спланувати налаштування компонентів середовища та забезпечення взаємодії між різними компонентами системи. Далі буде більш детально розглянуто процес реалізації, налаштування віртуального середовища, інтеграції в нього обраних технологій для дослідження алгоритмів локальної навігації БПЛА. Метою є побудова ефективного і реалістичного симуляційного середовища, яке дозволить тестувати та вдосконалювати алгоритми навігації в контрольованих умовах. Також для демонстрації можливостей розробленої швидкої та ефективної технології буде розглянуто приклад її застосування для задачі виявлення об'єктів. Це дозволить наочно показати, як розроблена технологія може бути використана для вирішення практичних задач.

3.1. Планування віртуального середовища

Спершу необхідно почати з розробки концептуального дизайну середовища, що в подальшому допоможе чітко визначити структуру і компоненти середовища, зробити процес розробки цього середовища швидше, ефективно планувати наступні етапи роботи. На рис. 3.1 зображено приблизний вигляд простого, але ефективного майбутнього середовища.

Але, авжеж, під час розробки макет середовища буде вдосконалено та буде додано більше видів об’єктів для створення більш деталізованого реалістичного середовища, а також для проведення експериментів з машинним зором над різними типами об’єктів. Середовище містить такі елементи, як будівлі, дерева, дороги, людей, які будуть випадково переміщатися на заданій області дороги, початкові точки, тобто точки старту, на яких БПЛА буде починати свій маршрут, а також інші деталі середовища, розташовані по периметру створеної мапи, як декоративні елементи.

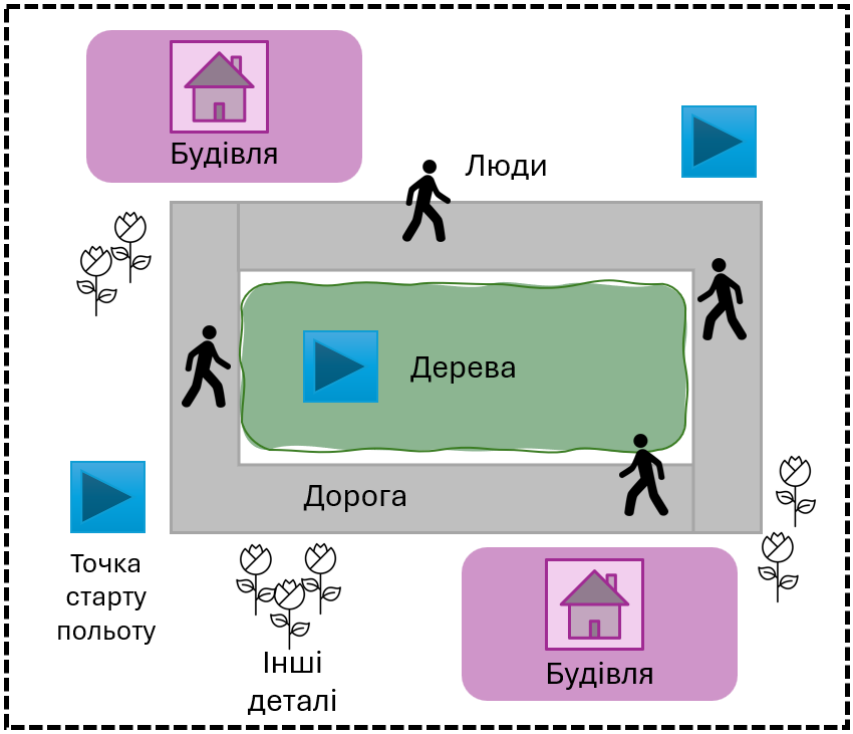

Рис. 3.1. Концептуальний дизайн віртуального середовища

Ця схема служить для попереднього планування структури віртуального середовища. Вона допомагає визначити ключові області віртуальної мапи, де будуть тестуватися різні алгоритми навігації і методи машинного зору, забезпечити збалансоване і правильне розташування об’єктів та перешкод для реалістичного моделювання різних сценаріїв автономної локальної навігації БПЛА. Це дозволяє ефективно планувати розробку, налаштування та тестування віртуального середовища, а також визначити необхідні компоненти та взаємозв’язки між ними.

3.2. Розробка тривимірної моделі середовища в Unreal Engine

Процес розробки віртуального середовища варто почати зі створення нового проєкту в Unreal Engine, але після того, як буде обрана потрібна версія ігрового рушія для роботи. Для проєкту була обрана версія 5.2.1, бо це поки що остання з версій UE 5, яка добре підтримується симулятором Colosseum. Можна обрати будь-яку заготовку проєкту, бо все одно є можливість повність переналаштувати середовище. На рис. 3.2 бачимо короткий перелік налаштувань за замовчуванням, які можна трохи змінити перед створенням проєкту. В нашому випадку обираємо заготовку гри від третього лиця з налаштуванням на Blueprint, потім буде можливість за потребою підключити C++ модуль до середовища. Цільова платформа визначає, для якої платформи буде оптимізовано проєкт, тому обираємо комп'ютер. Вибір якості Maximum означає, що проєкт буде використовувати найвищі налаштування якості.

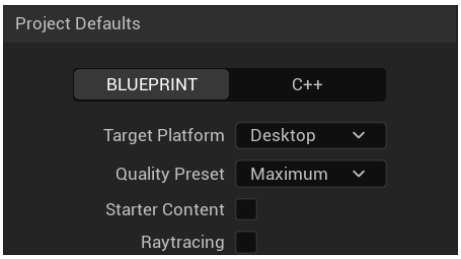

Рис. 3.2. Налаштування за замовчуванням для створення проєкту

Окрім цього, не забуваємо вказати місцезнаходження проєкту і змінити його ім'я. Далі, коли проєкт був створений і відкрився, переходимо до вкладки File та обираємо створення нового пустого рівня, який має тільки базове освітлення. На рис. 3.3 представлено вікно створення нового рівня, тобто мапи в Unreal Engine.

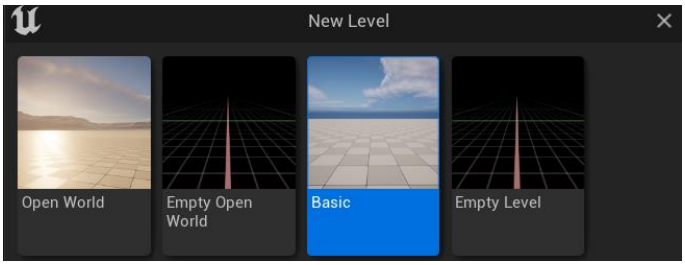

Рис. 3.3. Створення нового рівня

Наступний етапом розробки віртуального середовища є створення ландшафту в Landscape Mode, це незамінний компонент середовища, на якому і будуть розміщуватися всі в подальшому додані об’єкти рівня. На рис. 3.4 зображено вікно налаштування ландшафту, розміру його секцій, кількості цих секцій, матеріалу, тобто текстур, на інших параметрів. Є можливість самому створити новий ландшафт, а можна також імпортувати готовий з іншого файлу, що дуже зручно. Після того, як були встановлені всі параметри, натискаємо кнопку Create і в результаті буде створена основа майбутнього реалістичного середовища. Також після створення ландшафту, можна у тому ж вікні обрати інструменти Sculpt для створення більш цікавого рельєфу ландшафту, змінюючи його рівність, додаючи невеличкі пагорби та вигини у рельєфі.

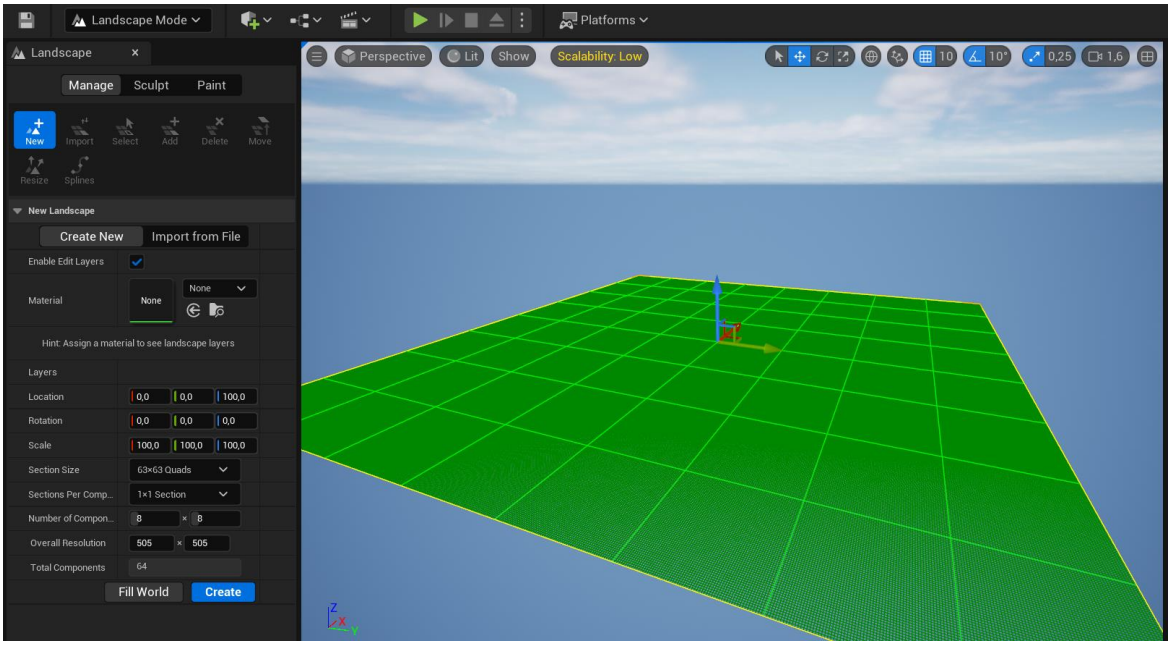

Рис. 3.4. Створення ландшафту

Необов’язково одразу обирати матеріал ландшафту у наведеному вище вікні, бо у будь-який момент можна змінити текстуру на іншу бажану, обравши в Selection Mode на головному екрані в одному з основних елементів інтерфейсу Outliner об’єкт створеного ландшафту та перейти в його деталі. Unreal Engine дозволяє зручно редагувати створені об’єкти будь-коли і як завгодно завдяки своєму ефективному і зрозумілому інтерфейсу. Деталі ландшафту і налаштування його матеріалу представлені на рис. 3.5.

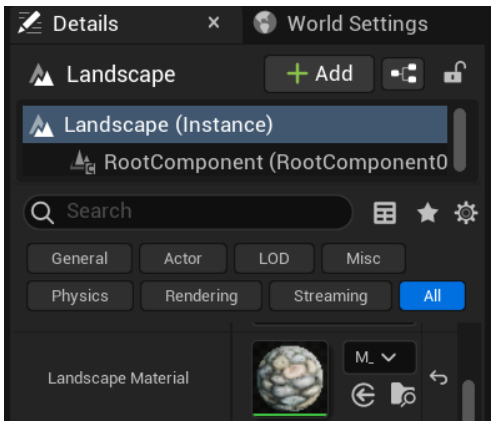

Рис. 3.5. Деталі ландшафту з налаштуванням його матеріалу

Далі переходимо в Foliage Mode, що представляє з себе набір різних інструментів для швидкого та зручного малювання або видалення таких об’єктів, як трави, квітів, дерев на створеному ландшафті. На рис. 3.6 зображений режим Foliage. Використовуючи цей режим, можна за короткий проміжок часу розфарбувати різними декоративними елементами великий простір ландшафту. Режим має різні налаштування пензлика для малювання, можна змінювати його інтенсивність та щільність малювання, відстань між об’єктами, можна окремо малювати тільки по одному об’єкту або одразу покрити весь ландшафт обраними об’єктами.

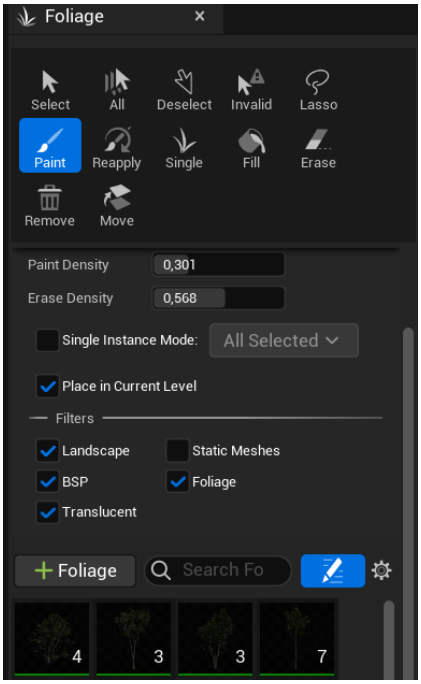

Рис. 3.6. Налаштування режиму Foliage

За допомогою Foliage Mode у роботу були додані трава, квіти та різні дерева. Всі використані 3D-моделі були взяті з безкоштовного вбудованому в Unreal Engine контент браузеру Quixel Bridge. Даний браузер надає багато об’єктів з Quixel Megascans, який є величезною бібліотекою високоякісних моделей, текстур та матеріалів, знятих з реального середовища.

Подальшим етапом є налаштування колізій всіх доданих об’єктів. Колізії визначають фізичні межі об’єкта, які використовуються для виявлення зіткнень з іншими об’єктами. Це є одним з найважливіших етапів, щоб потенційно БПЛА не пролітав скрізь об’єкти, а якось контактував з ними і фізично реагував на них у віртуальному середовищі. Для кожного об’єкта необхідно обов’язково створити колізійну оболонку, яка відповідає його формі, це може бути проста і примітивна форма або складна. На рис. 3.7 показано налаштування колізій для одного з доданих об’єктів. Вазон для квітів зображений з сіткою (mesh) і колізійною оболонкою (collision hull) на ньому. Справа показане меню налаштувань колізій. Для простих об’єктів можна використовувати комплексну детальну колізію як просту, але це не рекомендується робити для більш деталізованих і рельєфних об’єктів, бо це може значно вплинути на продуктивність. В налаштуваннях обрано «BlockAll», це означає, що об’єкт блокує всі інші об’єкти і взаємодіє з ними.

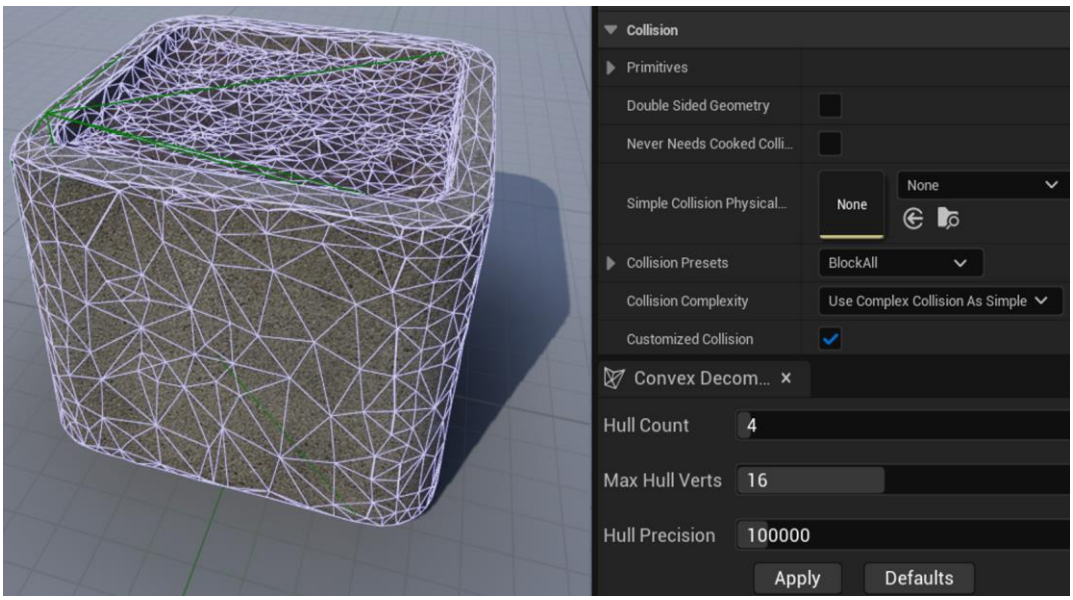

Рис. 3.7. Побудова колізій для об’єкта

Зображена на рис. 3.8 модель манекена була додана до рівня проєкту, щоб перевіряти на ній алгоритми машинного зору симульованого БПЛА завдяки нейронній мережі, коли віртуальне середовище буде повністю розроблено. Стандартна модель з деякими анімаціями руху з самого початку містилася у проєкті, так як для зручності проєкт був створений з шаблону для гри від третього лиця, але були внесені деякі зміни до моделі: вигляд манекена, його текстура, деталі руху, а також доданий алгоритм випадкового руху за допомогою Blueprints.

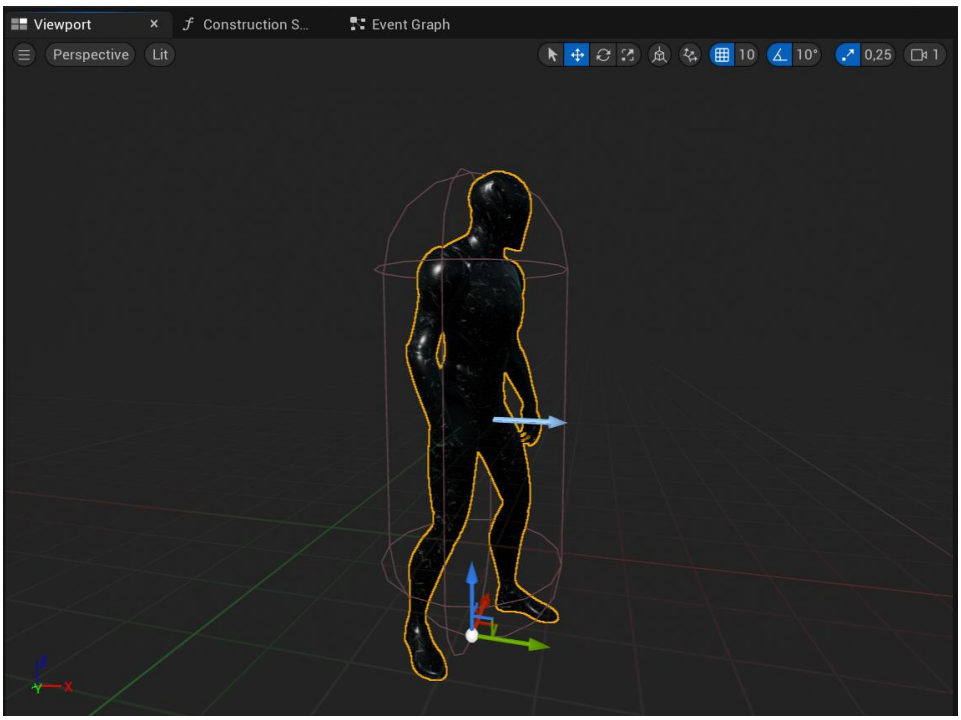

Рис. 3.8. Модель манекена

Наглядно рис. 3.9 демонструє налаштування деталей доданої моделі персонажа. Будь-які параметри можуть бути змінені, як, наприклад, довжина і висота кроку, швидкість ходьби, швидкість бігу, максимальне прискорення, мінімальна і максимальна швидкість руху, маса персонажа, швидкість його падіння та інші дуже користі параметри налаштування. Можливість зміни цих параметрів дозволяє налаштовувати поведінку персонажа під різні сценарії симуляції, забезпечуючи більш точний контроль над його рухом і взаємодією з віртуальним середовищем.

Рис. 3.9. Налаштування параметрів руху

Для реалізації алгоритму випадкового руху неігрових персонажів було використано спеціальний плагін AISupport з підтримкою штучного інтелекту (AI). Панель додання плагіну AISupport проілюстрована на рис. 3.10. Плагін AISupport в Unreal Engine забезпечує завантаження модулів AIModule і NavigationSystem під час виконання проєкту в реальному часі. Даний плагін спрощує взаємодію зі штучним інтелектом, він допомагає автоматично завантажити всі необхідні компоненти для функціонування AI.

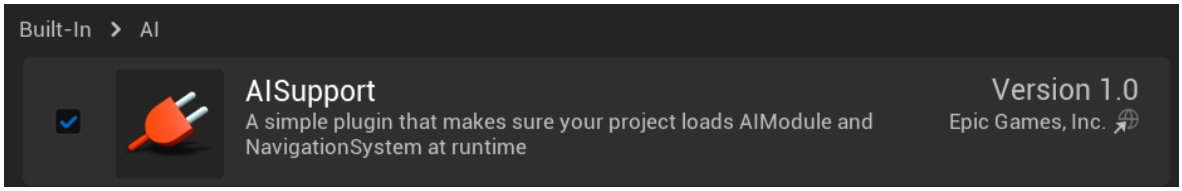

Рис. 3.10. Плагін AISupport

Алгоритм випадкового руху для Non-player character (неігрових персонажів), створений за допомогою візуальної мови програмування Blueprints, представлений на рис. 3.11. Розглянемо, як виглядає і працює алгоритм. Блок Event Tick викликається при кожному кадрі в секунду, він передає значення Delta Seconds (час між кадрами) у наступний блок. Random Float in Range генерує випадкове число з плаваючою точкою у визначеному діапазоні. Діапазон

встановлений від 2.0 до 6.0 секунд, це число визначає тривалість для наступного блоку. Блок затримки Delay після отримання випадкового значення часу з діапазону, чекає цю кількість секунд перед тим, як передати сигнал далі. Set Moving (true) після завершення часу затримки змінна Moving приймає значення «true». Get Actor Location отримує поточне місцезнаходження NPC. Get Random Reachable Point in Radius використовує поточне місцезнаходження персонажа як центр, а після генерує випадкову точку в радіусі 2500 одиниць, яка доступна для переміщення NPC туди. Блок AI MoveTo після отримання випадкової точки наказує NPC перемістити до неї. Set Moving (false) після завершення руху, незалежно від того, чи був він успішним або ні, змінна Moving приймає значення «false».

Таким чином, продемонстрований на рисунку алгоритм випадкового руху організований так, щоб NPC постійно рухався до якихось точок в межах заданого радіусу, зупиняючись і чекаючи випадковий проміжок часу між своїми переміщеннями від точки до точки.

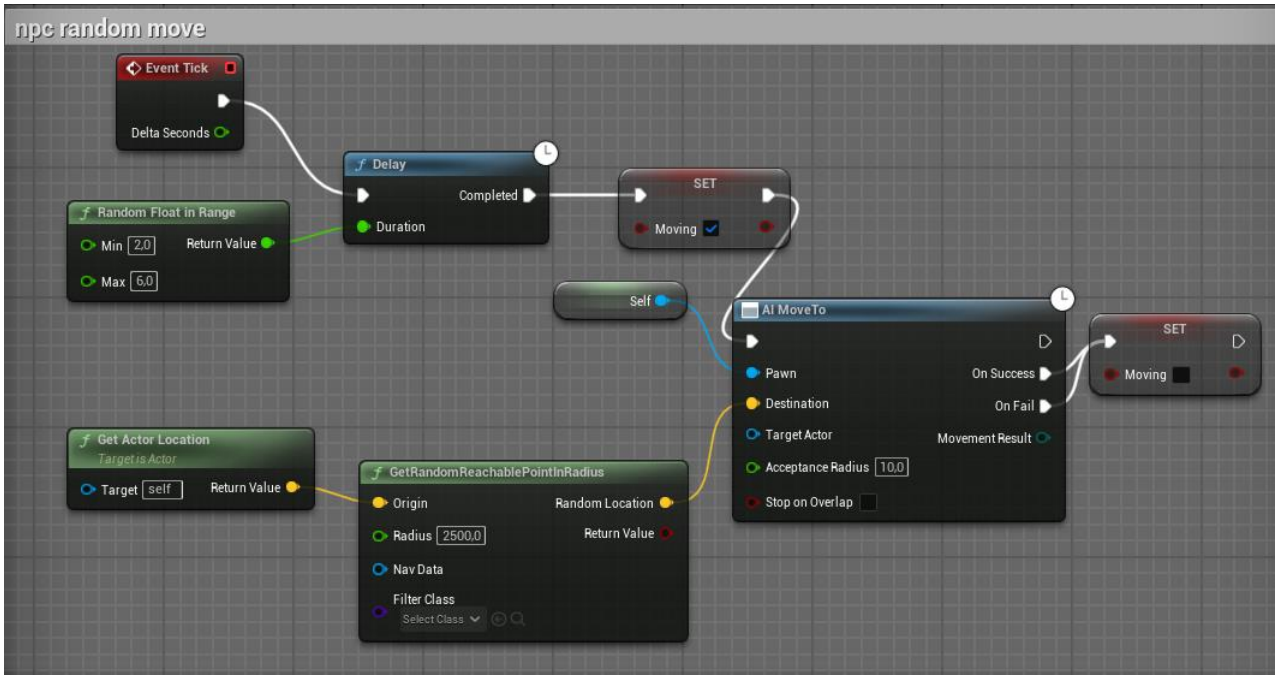

Рис. 3.11. Алгоритм випадкового руху NPC

Але одного запрограмованого алгоритму недостатньо, щоб змусити персонажа випадково рухатися у визначеному радіусі. Для цього на ландшафт

середовища треба накласти навігаційну сітку (Navigation Mesh) там, де персонаж буде вільно ходити. Якщо сітка правильно додана і налаштовала в середовищі, це буде помітно по виділеним зеленим областям на мапі, як це продемонстровано на рис. 3.12. Щоб алгоритм зміг знайти шлях між початковою позицією та наступним місцем призначення, навігаційна сітка генерується на основі геометрії колізій у середовищі. Ця спрощена полігональна сітка представляє навігаційний простір на створеному рівні середовища. За замовчуванням навігаційна сітка розбивається на плитки, що дозволяє перебудовувати її окремі частини. Алгоритм пошуку шляху буде намагатися знайти оптимальний шлях з найнижчою вартістю до пункту призначення.

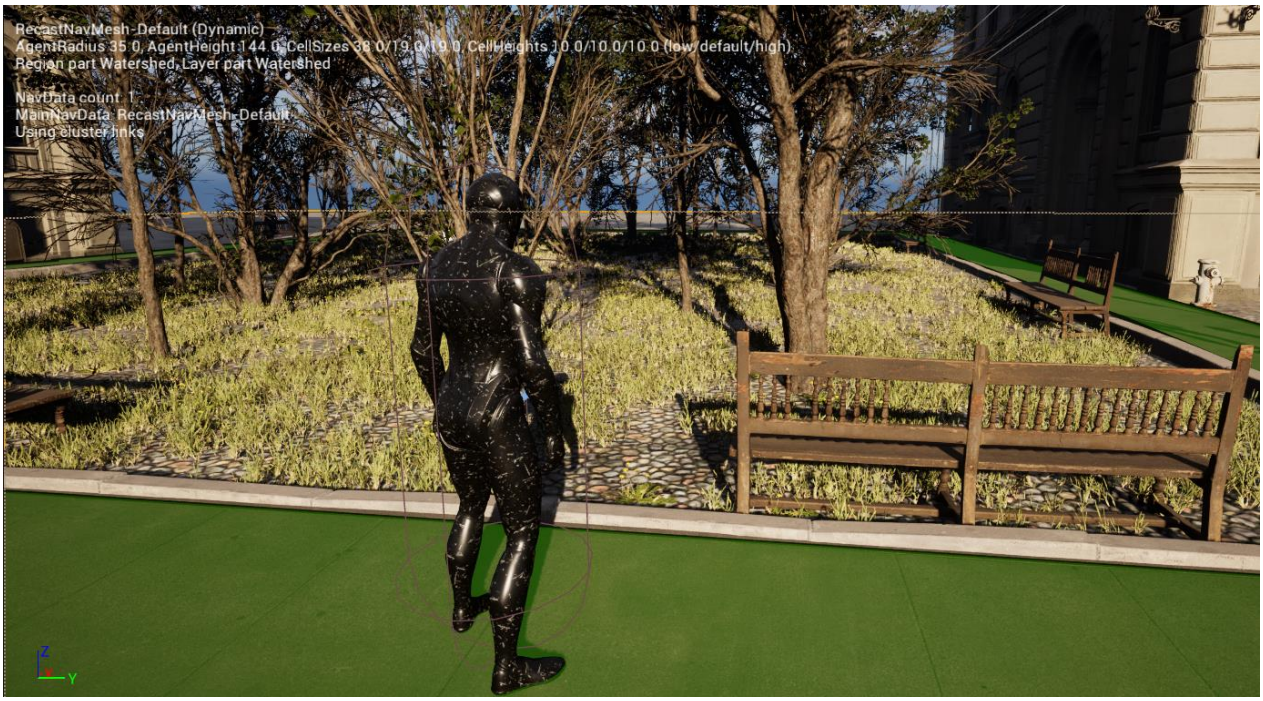

Рис. 3.12. Навігаційна сітка для NPC

На рис. 3.13 зображено повністю готову модель реалістичного і ефективного віртуального середовища, розроблену на ігровому рушії Unreal Engine. Кожен елемент середовища було створено з обґрунтованою користю для подальшої роботи і проведення експериментів.

Рис. 3.13. Готова модель віртуального середовища

Отже, в результаті проведеної роботи було швидко та ефективно реалізовано готову модель тривимірного середовища в Unreal Engine. Модель середовища готова до інтеграції з іншими обраними засобами та інструментами розробки.

3.3. Налаштування технологій програмування в середовищі Visual Studio

Коли віртуальне середовище готово, можна нарешті перейти до налаштування інтегрованого середовища розробки Visual Studio 2022. Переходимо до File menu, там обираємо створити новий клас C++. На рис. 3.14 наведено те, як додається новий пустий C++ клас до проєкту в Unreal Engine. Це необхідно зробити для того, щоб потім легко інтегрувати симулятор AirSim у розроблене симульоване середовище. Перед цим у Visual Studio має бути обов’язково встановлено Desktop Development з використанням C++, підтримка Windows 10 SDK та останній фреймворк .NET Framework SDK.

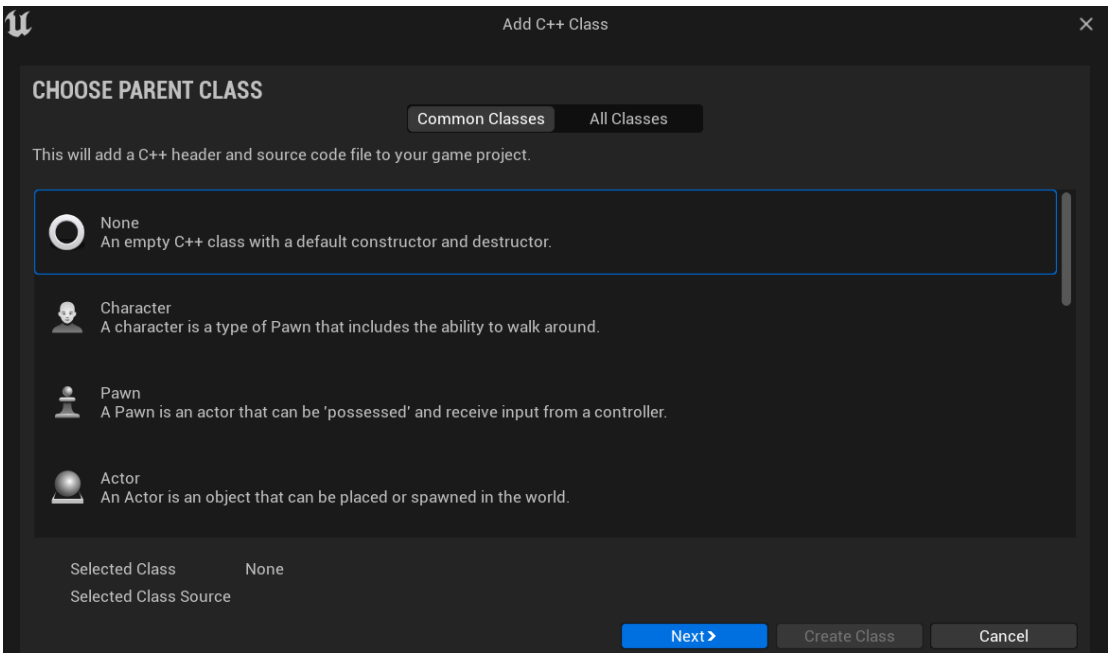

Рис. 3.14. Додавання C++ класу до проєкту в Unreal Engine

У налаштуваннях створення класу залишаємо тип класу за замовчуванням None. Натискаємо кнопку Next та також залишаємо за замовчуванням ім’я класу та його місце розташування, далі створюється заданий новий клас і Unreal Engine відкриє проєкт у Visual Studio для редагування коду. На рис. 3.15 зображений відкритий C++ клас у середовищі розробки. Можна побачити два створені файли: заголовковий файл MyClass.h та файл з реалізацією MyClass.cpp у відкритому проєкті у Visual Studio 2022. Отже, основні етапи налаштування середовища програмування для роботи були успішно виконані.

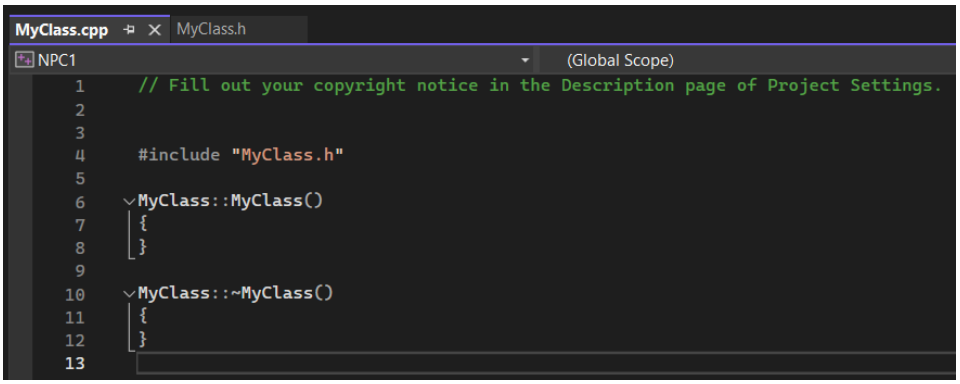

Рис. 3.15. Створений C++ клас

Виконані вище дії потрібні для наступного етапу роботи — інтеграції симулятора AirSim з відкритим вихідним кодом, а саме його оновленої версії Colosseum.

3.4. Інтеграція AirSim у віртуальне середовище

Настає етап інтеграції симулятора транспортних засобів AirSim у створене віртуальне середовище для симуляції поведінки БПЛА та забезпечення можливості взаємодії з безпілотником. Розберемо, як зібрати симулятор AirSim на системі Windows. Спершу запускаємо командний рядок розробника у Visual Studio 2022. Клонуємо репозиторій Colosseum з посилання Github у новий проєкт Visual Studio, переходимо у командному рядку до директорії AirSim за допомогою команди «cd AirSim.» Далі запускаємо build.cmd з командного рядка. В результаті цього створяться готові до використання біти плагінів у теці «Colosseum\Unreal\Plugins», які можна буде вставити у будь-який проєкт Unreal Engine.

Щоб додати AirSim до потрібного проєкту, необхідно теку «Plugins» перенести у теку свого створеного проєкту. Далі треба відредагувати головний файл проєкту типу Unreal Engine Project File у будь-якому текстовому редакторі. На додатку А наведений повний код відредагованого файлу NPC1.uproject. Внісши зміни у цей файл необхідно також відредагувати файл DefaultGame.ini у теці Config. У текстовому редакторі відкриваємо файл і додаємо останню строку коду, яка зображена на рис. 3.16. Це змушує Unreal Engine додати весь необхідний вміст AirSim у збірку проєкту.

```
[StartupActions]
bAddPacks=True
InsertPack=(PackSource="StarterContent.upack",PackName="StarterContent")
+MapsToCook=(FilePath="/AirSim/AirSimAssets")
```

Рис. 3.16. Редагування файлу DefaultGame.ini

Далі перевіряємо, щоб редактор Visual Studio та рушій Unreal Engine були закриті. Натискаємо на головний файл проєкту правою кнопкою миші і, як

продемонстровано на рис. 3.17, обираємо Generate Visual Studio project files. Ця дія виявить усі потрібні для роботи плагіни та вихідні файли у проєкті UE і автоматично згенерує .sln файл для Visual Studio.

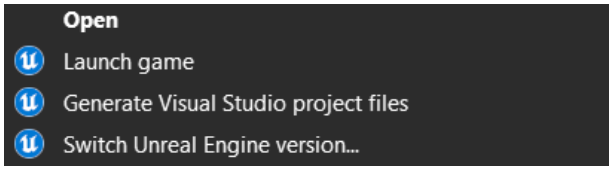

Рис. 3.17. Редагування файлу DefaultGame.ini

Після цього відкриваємо згенерований файл та у режимі DebugGame Editor запускаємо конфігурацію збірки. Це запустить роботи редактора Unreal Editor. Перше, що необхідно зробити у запущеному віртуальному середовищі, це створити об'єкт PlayerStart, тобто позначити місце, з якого буде починати свій маршрут дрон при кожному запуску середовища в реальному часі. PlayerStart це місце, де інстальований плагін AirSim, зображений на малюнку 3.18, створить і розмістить обраний транспортний засіб. Таких початкових точок на рівні можна створити безліч.

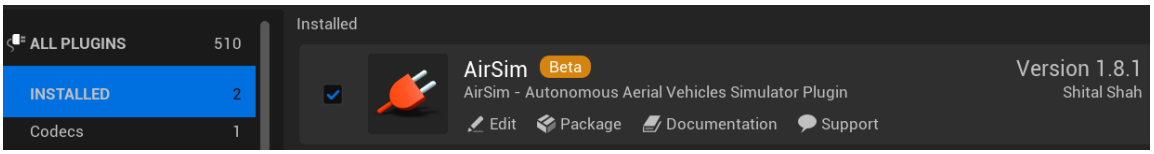

Рис. 3.18. Доданий плагін AirSim

Останній етапом налаштування роботи AirSim є зміна властивостей світу середовища. Щоб проєкт запускався у режимі симулятора AirSim, необхідно змінити налаштування GameMode Override на AirSimGameMode. Зміна ігрового режиму наведена на рис. 3.19. Зберігаємо всі внесені в проєкт зміни.

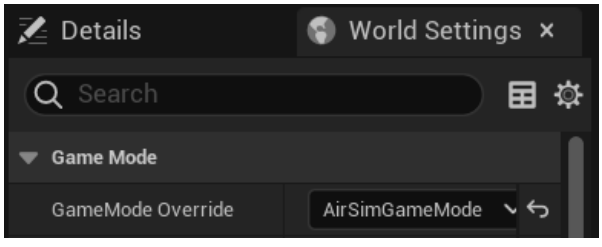

Рис. 3.19. Ігровий режим AirSimGameMode

Тепер при запуску проєкту ми бачимо вікно вибору бажаного транспортного засобу. Натискаємо «ні», щоб запустити симулятор безпілотного літального апарату.

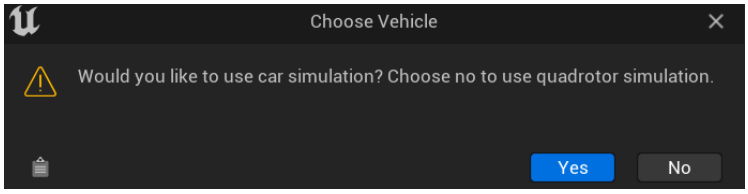

Рис. 3.20. Вибір транспортного засобу при запуску проєкту

На цьому етап інтеграції симулятора AirSim у розроблене віртуальне середовище завершений. При запуску проєкту можна відрити панель допомоги з управлінням транспортними засобами, зображену на рис. 3.21. Користувач може використовувати за замовченням встановлені клавіши швидкого доступу для управління транспортними засобами, зміни видів камери: на вид від першої особи (first person view), вид «fly with me», вид з землі, вид «chase with spring arm mode» та перемикання на ручний контроль камери; налагодження, запис польоту дрону та інших функцій симуляції.

Рис. 3.21. Робота проєкту в режимі симуляції дрона

Також у верхньому лівому куті під час роботи симуляції будуть надаватися такі важливі інформаційні повідомлення про роботу, як підтвердження готовності бази даних активів, інструкції з виклику допомоги, поточний режим камери, кількість зіткнень з іншими колізіями (Collision Count) та опис останнього зіткнення. Наведений вище рисунок свідчить про готовність середовища до подальшого тестування алгоритмів локальної навігації моделі симульованого дрона та тестування камери для вирішення прикладних задач з використанням для цього нейронної мережі YOLO.

3.5. Налаштування алгоритмів локальної навігації

AirSim має у собі безліч налаштувань алгоритмів його роботи. Є заготовки коду для виконання будь-яких завдань за допомогою симулятора. Окрім використання C++, є також алгоритми створені спеціально для використання мови програмування Python. Такі прості, але ефективні алгоритми роботи є для обох видів транспортних засобів. Готовий алгоритм можна дослідити та без проблем адаптувати і вдосконалити під поставлені до роботи вимоги.

У додатку Б наведено код робочого алгоритму навігації і дій дрона під час польоту. Розглянемо використані у коді важливі для роботи бібліотеки. AirSim — основна бібліотека для взаємодії з симулятором AirSim у віртуальному середовищі. Бібліотека дозволяє керувати дроном, отримувати дані сенсорів, робити знімки з його камер та виконувати інші дії, пов’язані з симуляцією. Numpy використовується для роботи з масивами даних, особливо для обробки зображень, отриманих з камер дрона. Os забезпечує функціональність для роботи з файловою системою, створення директорій та шляхи до файлів. Tempfile дозволяє створювати тимчасові файли та директорії, наприклад, тимчасові зображення з камер безпілотника. Бібліотека pprint використовується для форматованого виводу даних сенсорів та стану дрона, що полегшує читання складних структур даних. CV2 — це одна з найважливіших для роботи бібліотека OpenCV, яка використовується для обробки та збереження зображень, бібліотека дозволяє зберігати зображення, які були отримані з камери дрона, у форматі PNG.

Алгоритм починається з підключення до симулятора AirSim і підтвердження цього успішного підключення. Зчитуються та виводяться на екран дані сенсорів симульованого дрона, після чого дрон піднімається в повітря. Дрон послідовно переміщується до заданих координат (waypoints), зупиняючись на кожній з них для очікування від користувача команди продовження маршруту польоту. Після проходження всіх шляхових точок, дрон зависає в повітрі на останній з координат для того, щоб зробити знімки у різних режимах зі встановленій на ньому камери. Зображення зберігаються на комп'ютері користувача у тимчасовій директорії «Temp\airsim_drone». Після завершення роботи дрон повертається до початкового стану, вимикається керування.

Підсумовуючи вище сказане, цей код демонструє основні можливості керування дроном у симуляторі AirSim, зчитування даних сенсорів, виконання переміщення по заданим координатам, а також обробку та збереження зображень, зроблених БПЛА.

3.6. Інтегрування YOLO і застосування розробленої технології на прикладі задачі виявлення об'єктів

Нейронна мережа YOLO — це один з найсучасніших і ефективних методів для швидкого і ефективного виявлення об'єктів у зображеннях та відео. Мережа відрізняється своєю швидкістю і точністю роботи, YOLO є одним з найшвидших алгоритмів для виявлення об'єктів, оскільки він обробляє зображення в один проход (one pass), на відміну від інших методів, які можуть виконувати кілька проходів. Незважаючи на високу швидкість, модель також демонструє високу точність виявлення об'єктів, особливо добре це працює для великих і чітко видимих об'єктів. YOLO використовує одну нейронну мережу для аналізування меж об'єктів та їхніх класів, що робить його більш інтегрованим і менш складним у використанні в порівнянні з методами, які розділяють ці задачі на окремі етапи. І найголовніше, завдяки своїй швидкості та точності, YOLO є придатним для застосування у виконанні та розв'язанні задач в реальному часі, таких як

відеоспостереження, різноманітні алгоритми автономних транспортних засобів та дослідження логіки роботів-дронів.

Для інтегрування обраної моделі YOLO необхідно за допомогою бібліотеки Ultralytics встановити та налаштувати модель. Бібліотека Ultralytics забезпечує простий доступ до останніх версій моделей YOLO, таких як YOLOv5 та YOLOv8. Для роботи була використана модель YOLOv8n. Встановлення бібліотеки здійснюється через пакетний менеджер командою «pip install ultralytics». Після встановлення можна завантажити попередньо навчену модель або тренувати власну на нових даних.

Наведений в додатку В код підключає модель YOLO до віртуальної камери дрона у середовищі AirSim для виявлення об`єктів у режимі реального часу. Спочатку імпортуються необхідні бібліотеки та налаштовується TensorFlow для використання процесора замість GPU. Завантажується модель TensorFlow Lite та її метадані з файлу YAML. Далі створюється клієнт для підключення до віртуального дрона AirSim у віртуальному середовищі. Камера БПЛА налаштовується на отримання зображень в реальному часі. Основний цикл програми безперервно отримує зображення з камери дрона, обробляє їх за допомогою моделі YOLO для виявлення 80 різних типів об’єктів, список яких можна переглянути у файлі «metadata.yaml», та відображає результат з накладеними рамками на виявлені об’єкти з відповідними їм назвами. Виявлені на зображені з камери дрона об’єкти відображаються на вікні камери разом з показником FPS (кадрів в секунду). Користувач може зупинити виконання програми натисканням відповідних призначених для цього клавіш. Програма завершується натисканням клавіші «q», «x» або «Escape».

Для дослідження і розв’язання задачі виявлення об’єктів на рівні створеного реалістичного середовища, була використана попередньо навчена модель, яка має непоганий початковий набір об’єктів. Результат успішної роботи моделі у віртуальному середовищі представлений на рис. 3.22. Модель YOLO виявила через зображення, отримане з камери дрона, об’єкт неігрового

персонажа, який дуже схожий на людську фігуру, тому цей об’єкт і отримав над собою назву «person».

Рис. 3.22. Приклад виявлення об’єкту персонажа

Ця модель, завдяки своєму простому та ефективному початковому набору об’єктів, може розпізнавати різноманітні об’єкти, що дозволяє швидко і ефективно адаптувати її до конкретних вимог проєкту та різних умов симуляції. Такий підхід дозволяє прискорити процес дослідження і розвитку розробленої технології, забезпечуючи надійні результати навіть на ранніх етапах розробки.

3.7. Висновки до розділу 3

Розробка тривимірної моделі середовища в Unreal Engine дозволила швидко та ефективно створити реалістичну, високоякісну і деталізовану віртуальну мапу для дослідження локальної навігації БПЛА та роботи методів машинного зору при різних сценаріях взаємодії дрона з навколишнім середовищем. Налаштування технологій програмування в середовищі Visual Studio забезпечило ефективну інтеграцію та налагодження коду, що і подальшому допомогло розробці стабільних і продуктивних програмних рішень.

Інтеграція симулятора AirSim у створене віртуальне середовище надала можливість симулювати реалістичну поведінку безпілотного літального апарату в умовах, наближених до реальних, що значно полегшило розробку, тестування

та вдосконалення алгоритмів. Налаштування алгоритмів локальної навігації дало можливість дрону автономно орієнтуватися по заданим точкам, забезпечуючи безпеку та ефективність польотів.

Також використання YOLO в віртуальному середовищі для роботи з БПЛА значно розширило можливості розробленої технології, перетворюючи дрони на потужний інструмент для різноманітних застосувань у реальному світі. Нейронна мережа вдосконалила функціональність БПЛА, надавши йому здатність автоматично ідентифікувати об’єкти, коли вони потрапляють в його поле зору. Це відкриває широкі можливості для застосування подібних технологій у різних галузях, включаючи рятувальні операції, завдання розвідки та моніторингу. Використання передових алгоритмів виявлення об’єктів, таких як YOLO, робить роботу безпілотників більш автономною, ефективною та швидкою, що допоможе розвитку застосування БПЛА у багатьох сферах.

ВИСНОВКИ

У рамках дослідження процесу локальної навігації безпілотних літальних апаратів у розробленому симульованому середовищі було досягнуто значних результатів та виконано поставлені завдання. Головною метою роботи було дослідити локальну автономну навігацію БПЛА у віртуальному середовищі та визначити важливість і корисність використання таких середовищ для розв’язання подібних задач та проблем.

Було успішно розроблено тривимірне середовище за допомогою ігрового рушія Unreal Engine, що дозволяє моделювати реалістичні умови для польотів БПЛА. В середовищі Visual Studio були налаштовані необхідні інструменти та середовища для розробки програмного забезпечення, що взаємодіє з симулятором AirSim. Було інтегровано симулятор AirSim у віртуальне середовище, що дозволило ефективно моделювати поведінку БПЛА у реалістичних умовах. Також було реалізовано і налаштовано алгоритми локальної навігації, які забезпечують автоматизоване управління дроном. Була проведена інтеграція нейронної мережі YOLO для виявлення об’єктів у режимі реального часу, що значно підвищило функціональність і автономність БПЛА.

В результаті проведеної роботи було проаналізовано методи та засоби ефективної й швидкої побудови віртуальних середовищ для дослідження локальної автономної навігації БПЛА, визначено переваги використання віртуальних середовищ для моделювання та тестування навігаційних алгоритмів. Розроблено інтегровану технологію, яка поєднує в собі сучасні програмні засоби, інструменти та новітні алгоритми для створення ефективного симуляційного середовища.

Технологія має свої перспективи розвитку, наприклад, як подальше вдосконалення алгоритмів навігації та виявлення об’єктів з використанням передових методів машинного навчання. Розширення функціональності симуляційного середовища для підтримки нових сценаріїв та умов польотів. Впровадження додаткових інструментів та розширення можливостей для аналізу

даних в режимі реального часу. Технологія може бути ще більше вдосконалена та використана для навчання та тренування операторів БПЛА у безпечних умовах віртуального середовища. Віртуальне середовище може стати перспективною платформою для тестування нових навігаційних алгоритмів та обробки даних без ризику отримання пошкоджень реального обладнання БПЛА. Використання у наукових дослідженнях для вивчення поведінки БПЛА в різних умовах та ситуаціях.

Завдяки кваліфікаційній роботі було досягнуто опанування такими корисними технологіями, як Unreal Engine, AirSim та нейронними мережами для розв’язання прикладних задач. Отримані знання та навички можуть бути застосовані для подальших досліджень та розробок у сфері автономних систем та робототехніки. Загалом, проєкт продемонстрував ефективність поєднання сучасних технологій для створення потужного інструменту для дослідження та вдосконалення автономної навігації БПЛА у віртуальних середовищах.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Розвідка, перехоплення цілей і керування вогнем: які безпілотники задіяно у війні в Україні [Електронний ресурс]. Режим доступу: https://fakty.com.ua/ua/svit/20230227-osnovne-pryznachennya-rozvidka-najpopulyarnishi-modeli-bezpilotnykiv-v-ukrayini-ta-rosiyi/ (дата звернення 17.04.2024) — Безпілотники (БПЛА).

2. Kelly S. Hale, Kay M. Stanney. Handbook of Virtual Environments: Design, Implementation, and Applications, Second Edition. Florida: publisher CRC Press, 2018. 1272 p.

3. Кривоножко А. М., Романюк В. М., Дудко М. В., Руденко Д. В. Метод навігації безпілотного літального апарату при виконанні завдань за призначенням / А. М. Кривоножко, В. М. Романюк, М. В. Дудко, Д. В. Руденко // Збірник наукових праць Харківського національного університету Повітряних Сил. 2020. № 2(64). С. 61-68. Бібліогр.: 10 назв.

4. How Does a UAV Navigation System Work? [Електронний ресурс]. Режим доступу: https://transitiva.com/knowledgebase/how-does-a-uav-navigation-system-work/ (дата звернення 17.04.2024) — How Do Drones Navigate?

5. Стасенко Д. В., Яковина В. С. Аналіз наявних методів і засобів удосконалення навігації БПЛА з використанням штучного інтелекту / Д. В. Стасенко, В. С. Яковина // Науковий вісник НЛТУ України. 2023. № 33(4). С. 78-83. Бібліогр.: 34 назв.

6. FPV-дрон [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/FPV-%D0%B4%D1%80%D0%BE%D0%BD_(%D0%B7%D0%B1%D1%80%D0%BE%D1%8F) (дата звернення 19.04.2024) — FPV-дрон.

7. Anti Drone System (CUAS) [Електронний ресурс]. Режим доступу: https://www.zentechnologies.com/anti-drone-system-counter-drone-cuas.php (дата звернення 20.04.2024) — Anti Drone System.

8. Real-time Human Detection with OpenCV [Електронний ресурс]. Режим доступу: https://thedatafrog.com/en/articles/human-detection-video/ (дата звернення 26.04.2024) — People detection.

9. YOLO: Algorithm for Object Detection Explained [Електронний ресурс]. Режим доступу: https://www.v7labs.com/blog/yolo-object-detection (дата звернення 23.04.2024) — What is YOLO?

10. Що таке функціональні вимоги: приклади, визначення, повний посібник [Електронний ресурс]. Режим доступу: https://visuresolutions.com/uk/blog/functional-requirements/ (дата звернення 23.04.2024) — Що таке функціональні вимоги?

11. Нефункціональні вимоги: приклади, типи, підходи [Електронний ресурс]. Режим доступу: https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/ (дата звернення 23.04.2024) — Нефункціональні вимоги.

12. Colosseum, a successor of AirSim [Електронний ресурс]. Режим доступу: https://github.com/CodexLabsLLC/Colosseum (дата звернення 25.05.2024) — Introduction.

13. AirSim [Електронний ресурс]. Режим доступу: https://microsoft.github.io/AirSim/ (дата звернення 27.07.2024) — Programmatic control.

14. Положення про кваліфікаційні роботи (проєкти) здобувачів вищої освіти Національного Авіаційного Університету. СМЯ НАУ П 03.01(10) – 03 – 2024. Київ 2024, 62с.

ДОДАТКИ

Додаток А

Unreal Engine Project File


```
{
	"FileVersion": 3,
	"EngineAssociation": "5.2",
	"Category": "",
	"Description": "",
	"Modules": [
		{
			"Name": "NPC1",
			"Type": "Runtime",
			"LoadingPhase": "Default",
			"AdditionalDependencies": [
				"AirSim"
			]
		}
	],
	"Plugins": [
		{
			"Name": "ModelingToolsEditorMode",
			"Enabled": true,
			"TargetAllowList": [
				"Editor"
			]
		},
		{
			"Name": "PCG",
			"Enabled": true
```

```
        },
        {
            "Name": "Landmass",
            "Enabled": true
        },
        {
            "Name": "Water",
            "Enabled": true
        },
        {
            "Name": "AirSim",
            "Enabled": true
        },
        {
            "Name": "StaticMeshEditorModeling",
            "Enabled": true
        }
    ],
    "TargetPlatforms": [
        "MacNoEditor",
        "WindowsNoEditor",
        "Windows",
        "HoloLens"
    ]
}
```

Додаток Б

Алгоритм навігації БПЛА

```
import setup_path
import airsim
import numpy as np
import os
import tempfile
import pprint
import cv2

# connect to the AirSim simulator
client = airsim.MultirotorClient()
client.confirmConnection()
client.enableApiControl(True)

state = client.getMultirotorState()
s = pprint.pformat(state)
print("state: %s" % s)

imu_data = client.getImuData()
s = pprint.pformat(imu_data)
print("imu_data: %s" % s)

barometer_data = client.getBarometerData()
s = pprint.pformat(barometer_data)
print("barometer_data: %s" % s)

magnetometer_data = client.getMagnetometerData()
s = pprint.pformat(magnetometer_data)
print("magnetometer_data: %s" % s)
```

```
gps_data = client.getGpsData()
s = pprint.pformat(gps_data)
print("gps_data: %s" % s)

airsim.wait_key('Press any key to takeoff')
print("Taking off...")
client.armDisarm(True)
client.takeoffAsync().join()

state = client.getMultirotorState()
print("state: %s" % pprint.pformat(state))
airsim.wait_key('Press any key to move vehicle to waypoints')
waypoints = [
    (12, 12, -19, 8),
    (0, 0, -10, 10),
    (10, -10, -10, 5),
    (-10, 8, -5, 3),
]
for waypoint in waypoints:
    airsim.wait_key(f'Press any key to move vehicle to {waypoint}')
    client.moveToPositionAsync(*waypoint).join()

client.hoverAsync().join()
state = client.getMultirotorState()
print("state: %s" % pprint.pformat(state))

airsim.wait_key('Press any key to take images')
# get camera images from the car
responses = client.simGetImages([
```

```
    airsim.ImageRequest("0", airsim.ImageType.DepthVis),  #depth visualization
image
    airsim.ImageRequest("1", airsim.ImageType.DepthPerspective, True), #depth in
perspective projection
    airsim.ImageRequest("1", airsim.ImageType.Scene), #scene vision image in png
format
    airsim.ImageRequest("1", airsim.ImageType.Scene, False, False)])  #scene vision
image in uncompressed RGBA array
print('Retrieved images: %d' % len(responses))

tmp_dir = os.path.join(tempfile.gettempdir(), "airsim_drone")
print ("Saving images to %s" % tmp_dir)
try:
    os.makedirs(tmp_dir)
except OSError:
    if not os.path.isdir(tmp_dir):
        raise

for idx, response in enumerate(responses):
    filename = os.path.join(tmp_dir, str(idx))

    if response.pixels_as_float:
        print("Type %d, size %d" % (response.image_type,
len(response.image_data_float)))
        airsim.write_pfm(os.path.normpath(filename + '.pfm'),
airsim.get_pfm_array(response))
    elif response.compress: #png format
        print("Type %d, size %d" % (response.image_type,
len(response.image_data_uint8)))
```

```
        airsim.write_file(os.path.normpath(filename + '.png'),
response.image_data_uint8)
    else: #uncompressed array
        print("Type %d, size %d" % (response.image_type,
len(response.image_data_uint8)))
        img1d = np.fromstring(response.image_data_uint8, dtype=np.uint8) # get
numpy array
        img_rgb = img1d.reshape(response.height, response.width, 3) # reshape array to
4 channel image array H X W X 3
        cv2.imwrite(os.path.normpath(filename + '.png'), img_rgb) # write to png

airsim.wait_key('Press any key to reset to original state')
client.reset()
client.armDisarm(False)
client.enableApiControl(False)
```

Алгоритм підключення нейронної мережі до віртуальної камери БПЛА

```
import time
import cv2
import yaml
import tensorflow as tf
from detection_routines import detect_objects_on_frame,
draw_detections_on_frame, nms
import airsim

def run_cam(model_folder, model_filename):
    tf.config.set_visible_devices([], 'GPU')
    print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

    with open(model_folder+'/metadata.yaml', 'r') as file:
        metadata = yaml.safe_load(file)

    interpreter = tf.lite.Interpreter(model_path=model_folder+'/'+model_filename)
    interpreter.allocate_tensors()

    cameraType = "scene"
    cameraTypeMap = {
        "depth": airsim.ImageType.DepthVis,
        "segmentation": airsim.ImageType.Segmentation,
        "seg": airsim.ImageType.Segmentation,
        "scene": airsim.ImageType.Scene,
        "disparity": airsim.ImageType.DisparityNormalized,
        "normals": airsim.ImageType.SurfaceNormals
    }
```

```
    client = airsim.MultirotorClient()
    print("Connected: now while this script is running, you can open another")
    print("console and run a script that flies the drone and this script will")
    print("show the depth view while the drone is flying.")

    fontFace = cv2.FONT_HERSHEY_SIMPLEX
    fontScale = 0.5
    thickness = 2
    textSize, baseline = cv2.getTextSize("FPS", fontFace, fontScale, thickness)
    print(textSize)
    textOrg = (10, 10 + textSize[1])
    frameCount = 0
    startTime = time.time()
    fps = 0
    client.confirmConnection()
    client.enableApiControl(True)

    while True:
        # because this method returns std::vector<uint8>, msgpack decides to encode it
as a string unfortunately.
        rawImage = client.simGetImage("0", cameraTypeMap[cameraType])
        if (rawImage == None):
            print("Camera is not returning image, please check airsim for error
messages")
            return
        else:
            png = cv2.imdecode(airsim.string_to_uint8_array(rawImage),
cv2.IMREAD_COLOR)
```

```
        all_detections = detect_objects_on_frame(frame=png,
yolo_interpreter=interpreter, confidence_threshold=0.25)
        if len(all_detections):
            nms_index = nms(detections=all_detections, overlap_threshold=0.5)
            detections = [all_detections[i] for i in nms_index]
            draw_detections_on_frame(frame=png, detections=detections,
class_names=metadata['names'])

        cv2.putText(png, 'FPS ' + str(fps), textOrg, fontFace, fontScale, (255, 0, 255),
thickness)
        cv2.imshow("Drone camera", png)
        frameCount = frameCount + 1
        endTime = time.time()
        diff = endTime - startTime
        if (diff > 1):
            fps = frameCount
            frameCount = 0
            startTime = endTime

        key = cv2.waitKey(10) & 0xFF
        if (key == 27 or key == ord('q') or key == ord('x')):
            break
        if key == ord('t'):
            landed = client.getMultirotorState().landed_state
            if landed == airsim.LandedState.Landed:
                print("taking off...")
                client.takeoffAsync()
            else:
                print("landing...")
```

```
            client.landAsync()
            # client.hoverAsync().join()
        if key == ord('d'):
            client.moveByVelocityBodyFrameAsync(3, 0, 3, 5,
drivetrain=airsim.DrivetrainType.MaxDegreeOfFreedom,
yaw_mode=airsim.YawMode(False, 0))
        if key == ord('a'):
            client.moveByVelocityBodyFrameAsync(-3, 0, 0, 5,
drivetrain=airsim.DrivetrainType.MaxDegreeOfFreedom,
yaw_mode=airsim.YawMode(False, 0))

    cv2.destroyAllWindows()
if __name__ == '__main__':
    run_cam(model_folder='yolov8n_saved_model',
model_filename='yolov8n_float32.tflite')
```