

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА
ПРИКЛАДНОЇ МАТЕМАТИКИ**

**Кафедра системного програмування і
спеціалізованих комп'ютерних систем**

До захисту допущено:
Завідувач кафедри
Віталій РОМАНКЕВИЧ

«___»_____20__р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

**«Системне програмування та спеціалізовані комп'ютерні системи»
спеціальності 123 «Комп'ютерна інженерія»**

**на тему: «Засіб розробки систем групової навігації літальних апаратів із
застосуванням розпізнавання образів»**

Виконав (-ла):

студент IV курсу, групи KB-21 Бурма
Максим Сергійович

Керівник:

Асистент, PhD,
Сергієнко Павло Анатолійович

Консультант з нормоконтролю:

доцент кафедри СПІСКС, к.т.н., доцент,
Клятченко Ярослав Михайлович

Рецензент:

доцент кафедри ПЗКС, к.т.н., доцент,

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА
ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Віталій РОМАНКЕВИЧ

«__»_____20__р.

ЗАВДАННЯ

на дипломний проєкт студента

Бурми Максима Сергійовича
(прізвище, ім'я, по батькові)

1. Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів, керівник проєкту Сергієнко Павло Анатолійович, Асистент, PhD, затверджені наказом по університету від «28» травня 2026р. №1984-с
2. Термін подання студентом проєкту _____
3. Вихідні дані до проєкту див. Технічне завдання
4. Зміст пояснювальної записки
 - 1) Аналіз існуючих рішень та обґрунтування теми дипломного проєкту
 - 2) Опис використаних програмних засобів
 - 3) Алгоритмічна та програмна частина проєкту. Використані асети
 - 4) Особливості створеного симулятора. Тестування його роботи.

1. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- 1) Структурна схема взаємодії класів
- 2) Алгоритм роботи системи навігації
- 3) Алгоритм визначення напрямку ухилення
- 4) Алгоритм виявлення перешкод

2. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к. т. н., доцент каф. СП і СКС		

3. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	30.04.2026	
2.	Розроблення та узгодження технічного завдання	15.05.2026	
3.	Аналіз існуючих рішень	18.05.2026	
4.	Підготовка матеріалів першого розділу дипломного проєкту	20.05.2026	
5.	Підготовка матеріалів другого розділу дипломного проєкту	26.05.2026	
6.	Підготовка матеріалів третього розділу дипломного проєкту	28.05.2026	

7.	Підготовка матеріалів четвертого розділу дипломного проекту	30.05.2026	
8.	Підготовка графічної частини дипломного проекту	30.05.2026	
9.	Оформлення документації дипломного проекту	05.06.2026	
10.	Попередній огляд матеріалів диплому на кафедрі	08.06.2025	

Студент

(підпис)

Максим БУРМА
(ініціали, прізвище)

Керівник проекту

(підпис)

Павло СЕРГІЄНКО
(ініціали, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (- с., - рис. - табл., - додатки).

Об'єкт розробки – засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів, який дозволяє проводити тестування систем групової навігації frv-дронів.

Засіб розробки дозволяє: проводити тестування створених систем навігації у віртуальному середовищі. В якому передбачена, фізика польоту, отримання та аналіз кадрів відео з камери кожного дрону, власне саме середовище з розміщеними перешкодами, у вигляді дерев. В процесі розробки було використано рушій Unity, та систему розпізнавання образів Yolo. Як основну мову програмування було використано C#, для тренування Yolo було використано Python

В ході розробки:

- Проведено аналіз існуючих симуляторів польоту frv-дронів;
- Сформульовані вимоги до засобу розробки систем групової навігації літальних апаратів;
- Розроблений базовий симулятор польоту frv-дронів ;
- Натренована нейронна-мережа для розпізнавання перешкод з якими дрон вірогідно зіткнеться під час польоту до цілі;
- Інтегровано натреновану мережу у симулятор;
- Розроблена базова система навігації для тестування роботи створеного програмного забезпечення.

Використання створеного засобу у тестуванні систем навігації, значно пришвидшить їх розробку.

Ключові слова: засіб розробки системи групової навігації літальних апаратів, Yolo, Python, C#, розпізнавання образів.

ANNOTATION

The qualification work includes an explanatory note (- p., - fig. - table., - appendices).

The development object is a tool for developing group navigation systems for aircraft using pattern recognition, which allows testing group navigation systems for FPV drones.

The development tool allows: to test the created navigation systems in a virtual environment. Which provides for flight physics, receiving and analyzing video frames from the camera of each drone, the environment itself with obstacles placed that most often occur in life, namely: walls, trees, bushes, etc. The Unity engine and the Yolo pattern recognition system were used in the development process. C# was used as the main programming language, Python was used for training Yolo

During the development:

- An analysis of existing fpv drone flight simulators was conducted;
- Requirements for the development tool for group navigation systems of aircraft were formulated;
- A basic fpv drone flight simulator was developed;
- A neural network was trained to recognize obstacles that the drone is likely to encounter during the flight to the target;
- The trained network was integrated into the simulator;
- A basic navigation system was developed for testing the operation of the created software.

Using the created tool in testing navigation systems will significantly speed up their development, since during the analysis of existing simulators.

Keywords:

TOOL FOR DEVELOPING GROUP NAVIGATION SYSTEMS FOR AIRCRAFT, YOLO, PYTHON, C#, IMAGE RECOGNITION.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	A4	ІАЛЦ. 045490.002 ТЗ	Засіб розробки систем	4		
			групової навігації літальних			
			апаратів із застосуванням			
			розпізнавання образів			
			Технічне завдання			
	A4	ІАЛЦ.045490.003 ТП	Засіб розробки систем	2		
			групової навігації літальних			
			апаратів із застосуванням			
			розпізнавання образів			
			Відомість технічного			
			проекту			
	A4	ІАЛЦ.045490.004 ПЗ	Засіб розробки систем	55		
			групової навігації літальних			
			апаратів із застосуванням			
			розпізнавання образів			
			Пояснювальна записка			
	A4	ІАЛЦ.045490.005 Д1	Засіб розробки систем	1		
			групової навігації літальних			
			апаратів із застосуванням			
			розпізнавання образів			
			Структурна схема взаємодії			
			модулів програми			
Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 045490.001 ОА Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів Опис альбому Літ. Аркуш Аркушів 1 2 КПІ ім. Ігоря Сікорського, ФПСМ KB-21	
Розробив	Бурма М. С.					
Перевірів	Сергієнко П.А.					
Консульт.						
Н. контроль	Клятченко Я.М.					
Зав. каф.	Романкевич В.О.					

[illegible]

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ	1
2. ПІДСТАВА ДЛЯ РОЗРОБКИ	1
3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ	1
4. ДЖЕРЕЛА РОЗРОБКИ	1
5. ТЕХНІЧНІ ВИМОГИ	1
5.1. Вимоги до програмного продукту, що розробляється	1
5.2. Вимоги до апаратного забезпечення	3
5.3. Вимоги до програмного та апаратного забезпечення користувача	3
6. ЕТАПИ РОЗРОБКИ	3

					ІАЛЦ. 045490.002 ТЗ			
Змін.	Арк.	№ докум.	Підпис	Дата	Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів Технічне завдання			
Розробив	Бурма М. С.							
Перевірів	Сергієнко П.А.							
Консульт.								
Н. контроль	Клятченко Я.М.							
Зав. каф.	Романкевич В.О.							
					Літ.	Аркуш	Аркушів	
							1	3
					КПІ ім. Ігоря Сікорського, ФПСІМ КВ-21			

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів».

Галузь застосування: тестування навігаційних систем для FPV-Дронів та інших БПЛА подібного класу.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення засобу тестування групової навігації літальних апаратів із застосуванням розпізнавання образів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з операційною системою Windows
- можливість налаштовувати технічні характеристики дронів;
- можливість додавання нових дронів;
- можливість редагувати вже додані дрони;

					ІАЛЦ. 045490.002 ТЗ	Лист
Зм	Лист	№ докум.	Підп.	Дата		2

- можливість окреме виведення зображення з камери кожного дрону;
- наявність зручної системи підсумку тестування: вивід кількістю долетівших дронів, середньої швидкості на протязі польоту, тощо;
- наявність можливості ручного керування;

5.2. Вимоги до апаратного забезпечення

- Процесор: intel 7 114000f, Ryzen 5 7500F;
- Оперативна пам'ять: 16 Гб;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows 10;

ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	30.04.2026
2.	Розроблення та узгодження технічного завдання	18.05.2026
3.	Аналіз існуючих рішень	19.05.2026
4.	Підготовка матеріалів першого розділу дипломного проекту	20.05.2026
5.	Підготовка матеріалів другого розділу дипломного проекту	26.05.2026
6.	Підготовка графічної частини дипломного проекту	30.05.2026
7.	Оформлення документації дипломного проекту	5.06.2026
8.	Попередній огляд матеріалів диплому на кафедрі	14.06.2026

[illegible]

[illegible]

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ.....	2
ВСТУП.....	3
1.АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ	5
1.1 Визначення	5
1.2 Розгляд існуючих рішень.....	5
1.3 Способи покращення існуючих рішень	11
1.3 Висновки за першим розділом	12
2.ОПИС ВИКОРИСТАНИХ ПРОГРАМНИХ ЗАСОБІВ	13
2.1 Особливості FPV – дрону	13
2.2 Обраний рушій	19
2.3 Використані мови програмування.....	21
2.4 Висновки за 2 розділом.....	23
3.АЛГОРИТМІЧНА ТА ПРОГРАМНА ЧАСТИНА ПРОЄКТУ. ВИКОРИСТАНІ АССЕТИ.....	24
3.1 Вибір, підготовка, навчання та інтеграція нейромережі.....	24
3.2 Опис загальної взаємодії класів	27
3.3 Детальний опис алгоритму роботи кожного класу.	29
3.4 Ассети використані при створенні зовнішньої оболонки.....	40
3.5 Висновки за 3 розділом.....	41
4.ОСОБЛИВОСТІ СТВОРЕНОГО СИМУЛЯТОРУ.ТЕСТУВАННЯ ЙОГО РОБОТИ.....	42
4.1 Особливості створеної віртуальної моделі дрону	42
4.2 Загальні особливості симулятора.....	44
4.3 Тестування навігації.....	46
4.4 Завершення польоту.....	47
4.5 Тестування роботи симулятора	48
4.6 Порівняння створеного ПЗ з попередньо розглянутими програмами.....	49

					<i>ІАЛЦ. 045490.004 ПЗ</i>																
Змін.	Арк.	№ докум.	Підпис	Дата	Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів Пояснювальна записка																
Розробив	Бурма М. С.														Літ.			Аркуш		Аркушів	
Перевірів	Сергієнко П.А.																	1		55	
Консульт.															КПІ ім. Ігоря Сікорського, ФПСІМ КВ-21						
Н. контроль	Клятченко Я.М.																				
Зав. каф.	Романкевич В.О.																				

4.7	Аналіз можливих покращень	50
4.8	Висновки за 4 розділом	50
ВИСНОВКИ.....		52
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ		54
ДОДАТОК 1.....		56
ДОДАТОК 2.....		58
ДОДАТОК 3.....		60
ДОДАТОК 4.....		62

Зм	Лист	№ докум.	Підп.	Дата

ІАЛЦ. 045490.002 ПЗ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – програмне забезпечення.

FPV дрон – first person vision дрон, тобто дрон камера якого надає зображення від першого лиця.[1]

БПЛА – безпілотний літальний апарат[1]

API – це посередник, який дозволяє різним програмам «спілкуватися» між собою та обмінюватися даними

GPS – сукупність радіоелектронних засобів, що дозволяє визначати положення та швидкість руху об'єкта на поверхні Землі або в атмосфері.

РЕБ – Радіоелектронна боротьба

ESC – Electronic Speed Controller

FC – Flight Controller

LiPo – lithium-ion polymer тип акумулятора

					ІАЛЦ. 045490.002 ПЗ	Лист
						2
Зм	Лист	№ докум.	Підп.	Дата		

ВСТУП

В останній час розповсюдженість FPV дронів значно зросла; на сьогоднішній день їх використання присутнє як у цивільній, так і військовій сфері. Особливої актуальності цій темі надає їх масове використання у війні, в яку, нажаль втягнута Україна. У військовій сфері безпілотні літальні апарати виконують безліч різних завдань, а саме:

- Виконання глибокої і при цьому відносно безпечної розвідки.
- Нанесення шкоди ворогу шляхом використання прікріпленої до нього зброї.
- Постачання припасів на передову.
- Виконання ретрансляції сигналу, для цього використовуються спеціальні FPV-дрони з встановленим на них необхідним обладнанням.

Але виконання цих задач для пілотованих дрон не є таким простим, як здавалося. Причиною цього є те, що якщо вести дрон по прямій до цілі дуже високий шанс, що вашу позицію викриють ще до того, як ви виконаєте своє завдання. Саме тому при прокладанні маршруту до цілі зазвичай замість прямої дороги вибирається велика дуга. А через це значно зростає час та дистанція польоту, що значно виснажує пілота та в результаті може призвести до його помилок при керуванні. Проте, якщо довгу частину польоту до цілі на себе візьме автоматична система, це дозволить зменшити навантаження на пілота. Саме тому я вважаю доцільним створення програмного забезпечення для тестування таких систем.

Але, нажаль задача тестування навігаційних систем бпла не може мати універсального рішення через те, що кожен літальний апарат має свої особливості, обмеження та задачі. Тому було вирішено в якості основної особливості створеного програмного забезпечення надати можливість тестувати навігаційні системи з використанням комп'ютерного зору для FPV-дронів. Також, оскільки для тестування роботи навігаційної системи потрібен

комп'ютерний зір. Він буде створений у симуляторі за допомогою попередньо натренованої нейронної мережі.

					ІАЛЦ. 045490.002 ПЗ	Лист
						4
Зм	Лист	№ докум.	Підп.	Дата		

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Визначення

Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів – це спеціалізоване програмне забезпечення, призначене для проєктування, моделювання, тестування та впровадження алгоритмів колективного керування групою безпілотних літальних апаратів (БПЛА). У моїй роботі пріоритет віддано саме тестуванню, хоча й інші частини циклу розробки будуть реалізовані на мінімальному рівні. Головною метою цього засобу є забезпечення більш зручного способу тестування навігаційних систем, ніж існуючі на даний момент. Головна ідея навігації із застосуванням розпізнавання образів зменшити залежність навігації від GPS та подібних технологій, додаючи функцію розпізнавання перешкод для подальшого ухиляння від них, а в ідеалі й можливість розпізнавати орієнтири, що дозволить системі навігації працювати на якісному рівні незалежно від сигналу GPS. З технічної точки зору розпізнавання образів відбувається завдяки тому, що натренована під відповідне завдання нейронна мережа отримує зображення з камери дрона та аналізує його на наявність об'єктів, які вона може класифікувати. У моєму випадку це будуть будь-які об'єкти, які можуть завадити польоту, під спільною класифікацією «Перешкода». Я вважаю це правильним рішенням, так як задача полягає саме в навігації.

1.2 Розгляд існуючих рішень

Оскільки під час розгляду існуючих рішень не було знайдено таких, що б повністю покривали тематику дипломного проєкту, будуть розглянуті існуючі симулятори FPV-дронів. А саме з відомих можна виділити такі: LiftOff [11], Uncrashed Drone Simulator [12] та Ukrainian Fight Drone Simulator (UFDS) [13]. Кожен із них має як переваги, так і недоліки, що і будуть проаналізовані далі.

Розглянемо симулятор польоту дрона LiftOff.

					ІАЛЦ. 045490.002 ПЗ	Лист
						5
Зм	Лист	№ докум.	Підп.	Дата		

LiftOff — це програма, що дозволяє як отримати базові навички керування дроном, так і розвивати наявні; зображення головної сторінки цієї програми наведено на рисунку 1.1. І хоча вона не надає можливості тестувати навігаційну систему напрямку, при використанні стороннього ПЗ, яке б передавало інформацію між нашою системою та LiftOff, це можливо.

До переваг цього симулятора можна віднести:

- максимально наближена до реальності фізика польотів та зіткнень;
- можливість створити користувацьку модель дрона та налаштувати її вагу (редактор зображено на рисунку 1.2);
- можливість налаштування особливостей керування;
- наявність симуляції розрядки акумулятора;
- можливість модифікувати користувацький інтерфейс (OSD) [21], що виводиться під час польоту.

Недоліки:

- відсутність врахування затримки радіосигналу, його часткової або повної втрати;
- наявність незначних проблем з оптимізацією.

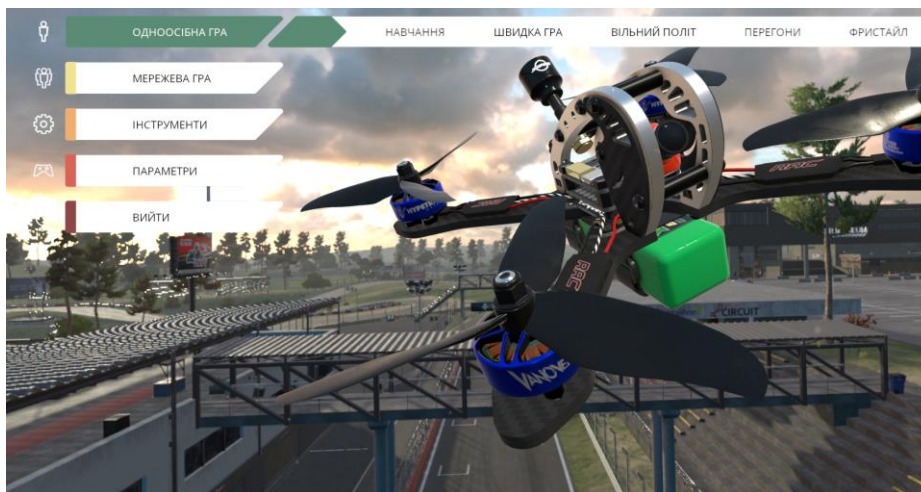


Рисунок 1.1 – Головна сторінка симулятора політу дрону LiftOff.

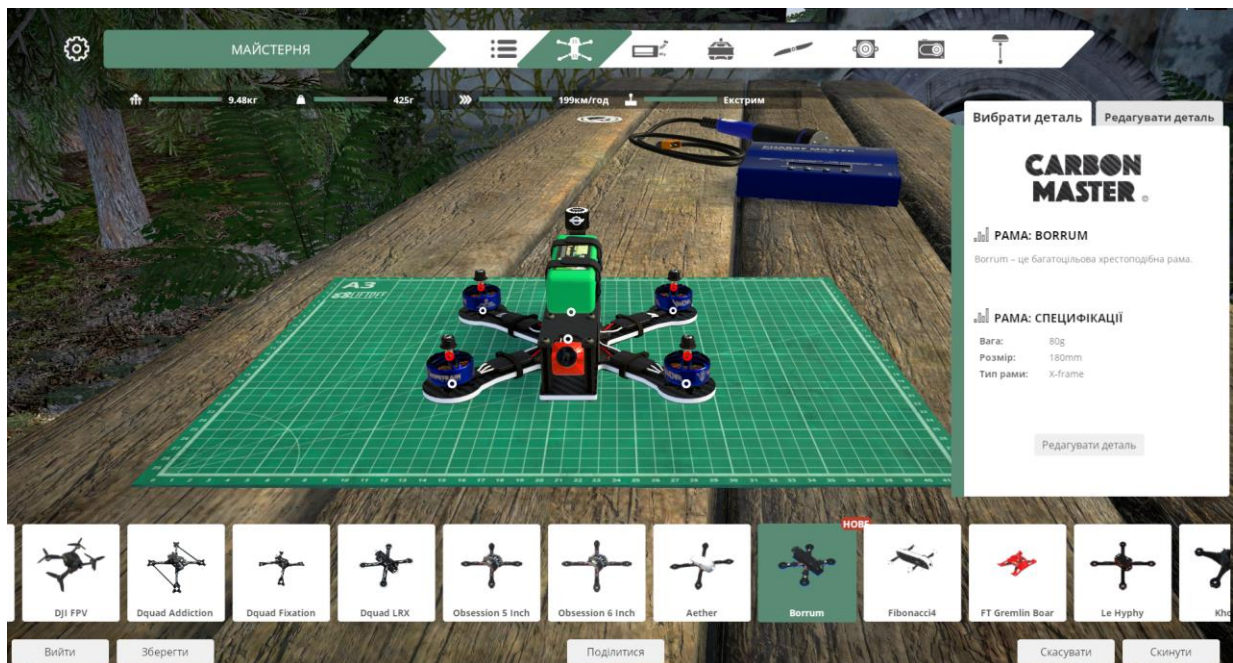


Рисунок 1.2 – Редактор дронів симулятору польоту дрону LiftOff

Розглянемо симулятор польоту дрону Uncrashed Drone Simulator.

Uncrashed – це сучасне програмне забезпечення, орієнтоване на надання максимально реалістичного візуального та фізичного досвіду пілотування FPV-дронів. Програма дозволяє як тренувати базові навички, так і відпрацьовувати складні кінематографічні або гоночні маневри, зображення головної сторінки цієї програми наведено на Рисунку 1.3. Як і в попередньому випадку, симулятор не має вбудованого API для прямого тестування автономних навігаційних систем, проте інтеграція можлива за рахунок стороннього ПЗ.

До переваг цього симулятору можна віднести:

- високодеталізовану графіку та великі, реалістичні локації.
- наявність динамічних об'єктів на картах (наприклад, автомобілі, що рухаються).
- максимально наближена до реальності фізика польотів та зіткнень
- можливість гнучкого налаштування параметрів дрона, а також зміни його фізичних характеристик (вага, потужність моторів, крок пропелерів), меню налаштувань зображено на рисунку 1.4.

- можливість кастомізації екранного меню (OSD), що імітує реальні інтерфейси відеозв'язку.

Недоліки:

- високі вимоги до апаратного забезпечення комп'ютера (зокрема до відеокарти) через використання надсучасного графічного рушія.
- фізика зіткнень із зовнішніми об'єктами іноді є спрощеною (присутній ефект "відскакування" замість руйнування чи застрягання).

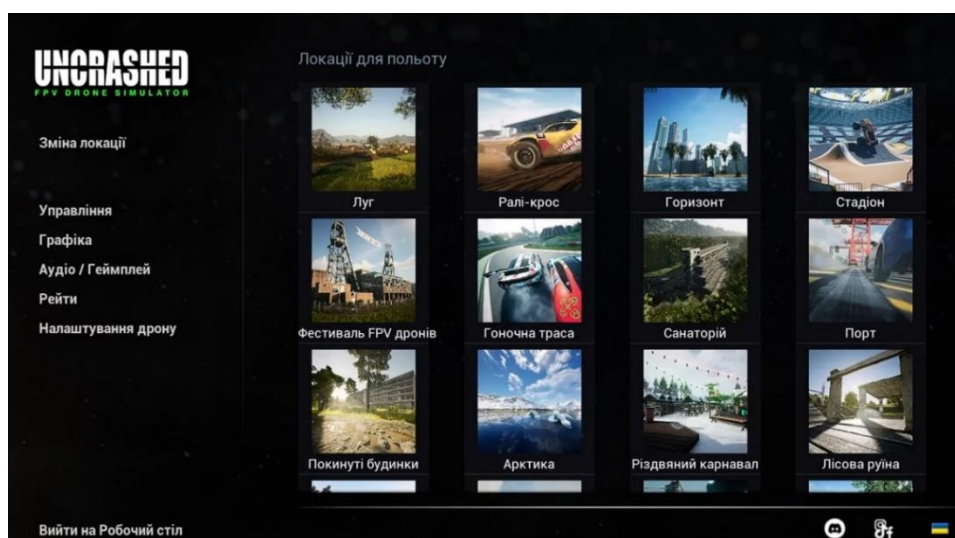


Рисунок 1.3 – Головна сторінка симулятору політу дрону UnCrashed Drone Simulator



Рисунок 1.4 – Редактор дронів симулятору політу дрону UnCrashed Drone Simulator

Розглянемо симулятор польоту дрону UFDS.

UFDS – це симулятор, що надає можливість тренуватися керувати дроном в умовах, наближених до бойових, тобто симулюється дія РЕБ, наявність бойового навантаження на дроні, присутня повна та часткова втрата сигналу тощо. З мінусів для завдання розробки навігаційних систем знову можна виділити неможливість їх прямої інтеграції. Головне меню цієї програми зображено на рисунку 1.5

У загальному до переваг цього симулятора можна віднести:

- найбільш наближену до реальної графіку серед розглянутих.
- наявність кастомізації дрона (хоча вибір деталей і значно менше в порівнянні з попередніми), меню налаштувань зображено на рисунку 1.6.
- наявність імітації роботи РЕБ [22] чим він і виділяється з поміж інших.
- наявність можливості бойового застосування дрону.
- наявність динамічних об'єктів на картах.
- максимально наближена до реальності фізика польотів та зіткнень.

До недоліків же можна віднести:

- неінтуїтивний користувацький інтерфейс, на який є досить багато скарг

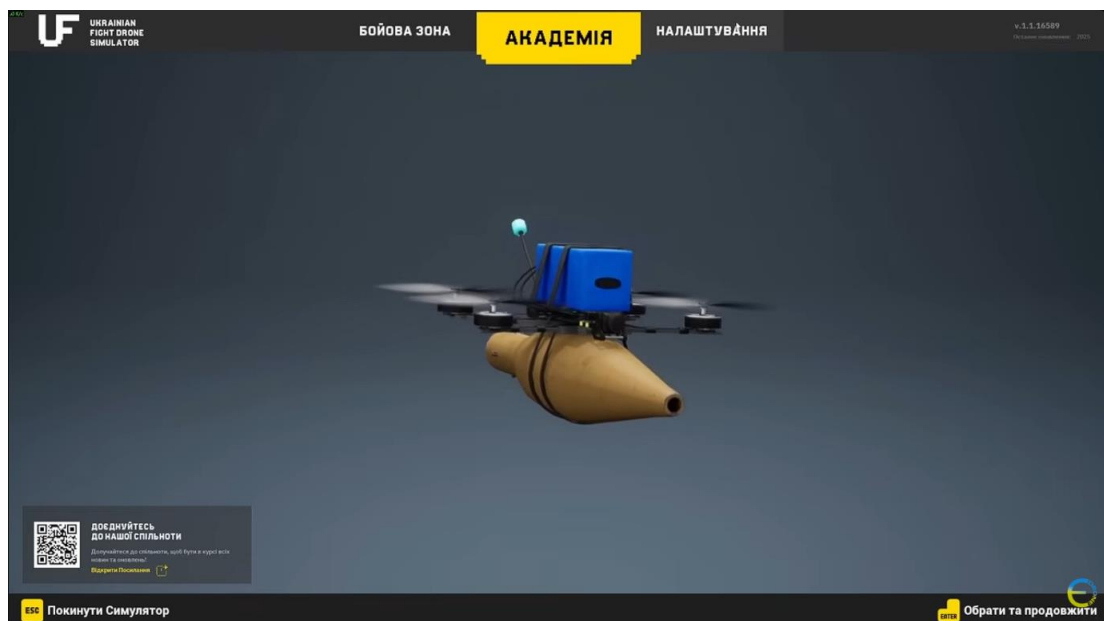


Рисунок 1.5 – Головна сторінка симулятору політу дрону Ukrainian Fight Drone Simulator

					ІАЛЦ. 045490.002 ПЗ	Лист
Зм	Лист	№ докум.	Підп.	Дата		9

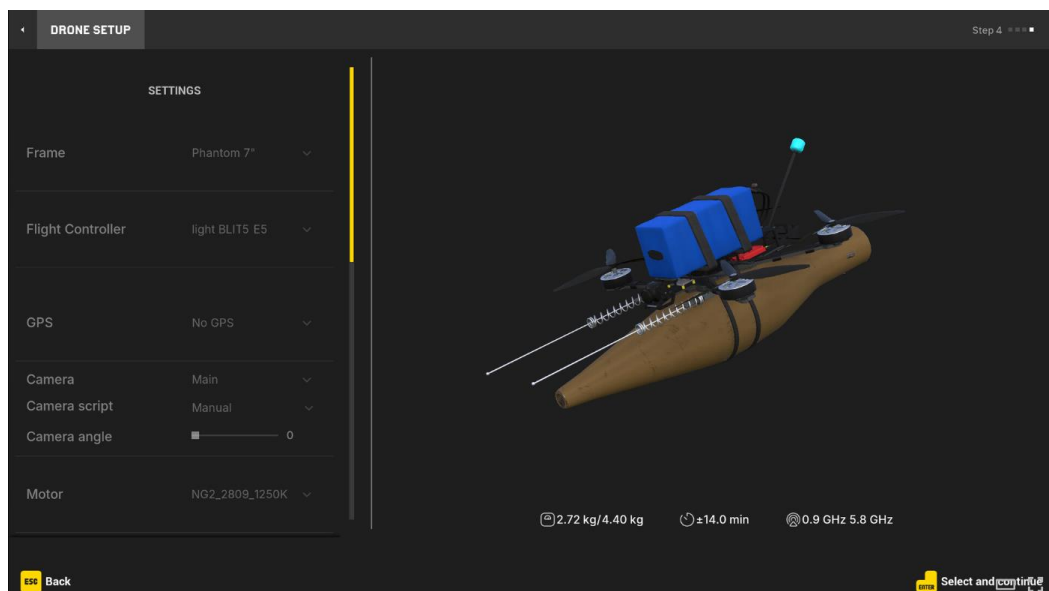


Рисунок 1.6 – Редактор дронів симулятору політу дрону Ukrainian Fight Drone Simulator

Загалом його можна вважати свого роду інноваційним, так як найперший реалістичний симулятор FPV дронів, який пропонує політ у умовах наближених до бойових.

Порівняння симуляторів один з одним за можливостями наведено в таблиці 1.1

Таблиця 1.1 – Порівняльна характеристика симуляторів

Назва симулятора	Фізика	Графіка	Імітація роботи радіозв'язку	Вбудована можливість тестування автопілота
LiftOff	Реалістична, але є проблеми з зіткненнями	середня	-	-
Uncrashed	реалістична	Фото-реалізм	Базовий рівень	-
UFDS	реалістична	реалістична	Детальна, з імітацією РЕБ	-

Загалом у всіх симуляторах фізика реалістична в усіх аспектах, тобто присутня інерція, є можливість літати з урахуванням вітру, як очевидно на дрон впливає сила гравітації сила якої в свою чергу залежить від ваги дрона. Також в кожному симуляторі обробляється фізика зіткнень, а особливо той момент, коли дрон вривається обертаючимся гвинтом, це також реалізовано в усіх симуляторах, проте на мою думку у Liftoff з цим маються певні проблеми. Але, з іншого боку, у Liftoff присутній тепловий вплив на дрон, тобто якщо підлетіти до надто гарячого об'єкта, наприклад магми, дрон почне плавитися. Щодо графіки, лідером безперечно буде Uncrashed Drone Simulator з його фотореалістичною графікою, але в нашому випадку така якість не потрібна, і перевагу краще віддати графіці, подібній до тієї, що використана в Liftoff. А от у галузі імітації радіозв'язку лідером вже буде український симулятор Ukrainian Fight Drone Simulator, оскільки там не лише розраховується дистанція втрати сигналу, а й враховуються різні рухомі/нерухомі перешкоди, а також робота засобів радіоелектронної боротьби.

1.3 Способи покращення існуючих рішень

В данному випадку можемо бачити, що існуючі рішення не мають інтегрованого рішення для тестування систем групової навігації дронів, тому більша частина часу буде витрачена саме на реалізацію цієї задачі. Як додаткове покращення під задачу тестування саме навігаційних систем буде доцільно замість того щоб використовувати графіку найвищої якості, використати просту і невимогливу, графіку загалом та трохи більш детальну для перешкод, які має розпізнавати дрон, через те що машині фактично всеодно на якість графіки, а тренування нейронної мережі для виконання задачі розпізнавання образів, від програми не потребується, це дозволить сфокусуватися саме на фізичних особливостях польоту, при цьому зберігаючи припустимі вимоги до апаратного забезпечення.

1.3 Висновки за першим розділом

Враховуючи те, що для задачі саме тестування систем навігації для FPV дронів ще не було створено ні єдиного рішення, було вирішено проаналізувати можливості симуляторів керування FPV дроном, без цієї можливості. Для аналізу були обрані такі симулятори:

- LiftOff – досить відомий симулятор із гарною фізикою та непоганою графікою.
- Uncrashed Drone Simulator – відомий симулятор із фотореалістичною графікою.
- Ukrainian Fight Drone Simulator – швидко набираючий популярність український симулятор, який пріоритизує симуляцію бойового застосування дрона та має для цього відповідний додатковий функціонал.

В результаті цього аналізу було вирішено що при розробці потрібно віддати пріоритет саме створенню можливості тестувати автопілот, адже ця функція ще ніде не реалізована. Серед іншого важливого функціоналу для такого роду симуляторів було вибрано реалізацію реалістичного польоту дрона, що включає в себе що найменше реалістичне керування, наявність сили тяжіння та інерції, а також вплив вітру на FPV. Також було вирішено додати можливість модифікації дрону хоча б на рівні змінних у редакторі. Щодо графіки, її було вирішено спростити, адже нейронній мережі фактично всеодно, що розпізнавати фотореалістичну графіку чи просту. Але при цьому використання простої графіки значно знизить навантаження на комп'ютер.

2. ОПИС ВИКОРИСТАНИХ ПРОГРАМНИХ ЗАСОБІВ

2.1 Особливості FPV – дрону

FPV–дрон – це надзвичайно мобільний безпілотний літальний апарат, що оснащений камерою, яка у режимі реального часу передає зображення “від першого лиця” пілоту [1]. Керується в переважній більшості випадків за допомогою спеціального пульта керування, в залежності від радіопередавача на якому змінюєть дистанції зв'язку. Вбудовані передавачі найчастіше надають радіус роботи в 1-2 км, в той час як ті, що встановлюються додатково, можуть розширити радіус роботи до 40 км.

Компоненти базового дрону [1], [2], [4]:

- Рама
- Польотний контроллер
- Регулятор швидкості
- Двигуни
- Камера
- Акумулятор
- Радіоприймач (RX)
- Відеопередавач (VTX)

Рама є основною частиною дрона, на яку встановлюються усі інші елементи. Від її розміру та форми залежить, які комплектуючі буде можливо встановити, її приклад зображено на рисунку 2.1. Але, окрім того що рама є платформою для комплектуючих вона, також має захищати їх від ударів. Тому в якості матеріалу для неї найчастіше використовують карбон, але у військовій галузі іноді використовують пластик, через те що він значно дешевший, а від дрону не потребується довговічність. Конструктивно ж рами відрізняються за розміром та геометрією променів, наприклад True-X, Deadcat або Stretch-X, що впливає на максимальне навантаження та аеродинаміку.



Рисунок 2.1 – Зовнішній вигляд рами дрона

Польотний контроллер(FC) є найважливішою частиною дрону, відповідаючи за власне керування польотом та усі дані з ним пов'язвні, приклад такого контроллера зображений на рисунку 2.2. Для цього він містить процесор для виконання необхідних обчислень, набір різних датчиків (барометр, термометр, тощо), гіроскоп та порти для підключення додаткових пристроїв. В якості цього додаткового пристрою може бути GPS, динамік, тощо. Для керування польотом він використовує спеціальне ПЗ (Betaflight [16] або щось подібне), за допомогою нього процесор обробляє отримані команди та відправляє відповідні команди двигунам або іншим приладам.

Зм	Лист	№ докум.	Підп.	Дата

ІАЛЦ. 045490.002 ПЗ

Лист
14



Рисунок 2.2 – Польотний контролер

Регулятор швидкості (ESC – Electronic Speed Controller) відповідає за зв’язок між контролером та двигунами, керуючи їх обертами [6], приклад такого контролеру зображений на рисунку 2.3.



Рисунок 2.3 – Регулятор швидкості

Загалом їх можна поділити на 2 типи, ті які ставляться окремо для кожного мотора, та 4в1, які керують одразу всіма. Також він перетворює постійний струм акумулятора на змінний, необхідний для роботи моторів. В якості важливих характеристик цього модулю можна виділити максимальний підтримуваний струм, напругу та прошивку. Прошивка цього модуля загалом відповідає за якість та швидкість реакції моторів на команди.

Двигуни відповідають за підйомну силу, а також за те, які гвинти можна буде ефективно використовувати, його зображення наведено на рисунку 2.4.



Рисунок 2.4 – Двигун дрону на 900KV

Всього їх на дроні 4, і усі направлені догори, через що горизонтальна швидкість набирається шляхом нахилу дрону. На сьогоднішній день, найчастіше використовують безколекторні електродвигуни, які мають достатньо високу надійність та ККД. Найважливішою характеристикою для цієї частини дрону є показник KV – відношення кількості обертів на хвилину до кількості вольтів, поданих на двигун.

Зм	Лист	№ докум.	Підп.	Дата

ІАЛЦ. 045490.002 ПЗ

Лист
16

Основну камери встановлюють спереду дрону, щоб надати пілоту можливість бачити очима дрона. Приклад зовнішнього вигляду типової камери наведено на рисунку 2.5.



Рисунок 2.5 – Камера дрона

Як можна побачити на зображенні, камери для дронів мають кругле кріплення збоку, що при встановленні дозволяє вільно вибирати кут огляду.

Також на військових дронах іноді встановлюються додаткові камери, що направлені в інші місця, наприклад для точного скиду боєприпасу вниз. Камери між собою можна поділити на аналогові та цифрові. Перші надають стабільну картинку з мінімальною затримкою за невисоку ціну, проте жертвують якістю зображення. Другі, у свою чергу, надають значно кращу картинку, жертвуючи всім іншим.

Акумулятор є джерелом живлення для всієї електроніки та силової установки FPV-дрону. Найчастіше використовують саме літій-полімерні акумулятори, які маркуються міткою LiPo, через їх високу віддачу струму. Максимальна напруга акумулятора визначається кількістю послідовно з'єднаних елементів. Зазвичай кількість елементів позначається у назві: 2Р, 3Р кількість

паралельних ланцюгів, 4S, 6S кількість послідовних елементів. Приклад літій-полімерного акумулятора на 3 паралельно з'єднаних ланцюга по 6 елементів наведено на рисунку 2.6



Рисунок 2.6 – Акумулятор на 3 паралельно з'єднаних ланцюга по 6 елементів

Радіоприймач отримує команди від контролера пілота та передає їх на польотний контролер для обробки. Найчастіше для керування використовуються частоти від 865 до 915 МГц. В якості протоколів зв'язку використовуються найчастіше EXPRESSlrs або TBS Crossfire. Також окрім прийому команд, зазвичай, додається можливість надсилання назад даних телеметрії [7].

Відеопередавач передавач відповідає, як зрозуміло з назви, за передачу відеопотоку з камери. Він необхідний у вигляді окремого елемента, так як для швидкої передачі відео потрібна досить висока пропускна здатність зв'язку. Саме через це такі передавачі у більшості випадків використовують частоту 5.8 ГГц, що і дозволяє транслювати відео в реальному часі. Проте разом з підвищенням частоти роботи зростає і тепловиділення передавача, через що він потребує постійного охолодження. Саме тому на дронах, досить часто, можна

бачити винесений на певну відстань від рами модуль, який і є цим передавачем[7].

Ознайомившись з тим, що таке FPV дрон та особливостями його роботи, зрозуміло, що для того, щоб тестувати їх групу, потрібно буде створити 3D середовище, симулювати фізику та роботу камери кожного з них, що є непростим завданням, а отже, буде доцільно використовувати певний рушій, замість написання програми з нуля. Це дозволить пришвидшити розробку проєкту.

2.2 Обраний рушій

Серед існуючих рушіїв, я розглядав декілька варіантів: Unity, Unreal Engine та Godot.

Розглянемо Unity [17] – це кросплатформенний рушій для створення програм з 2D та 3D графікою, задача якого надати можливість створити повноцінну програму в одному редакторі, його редактор зображено на рисунку 2.7.

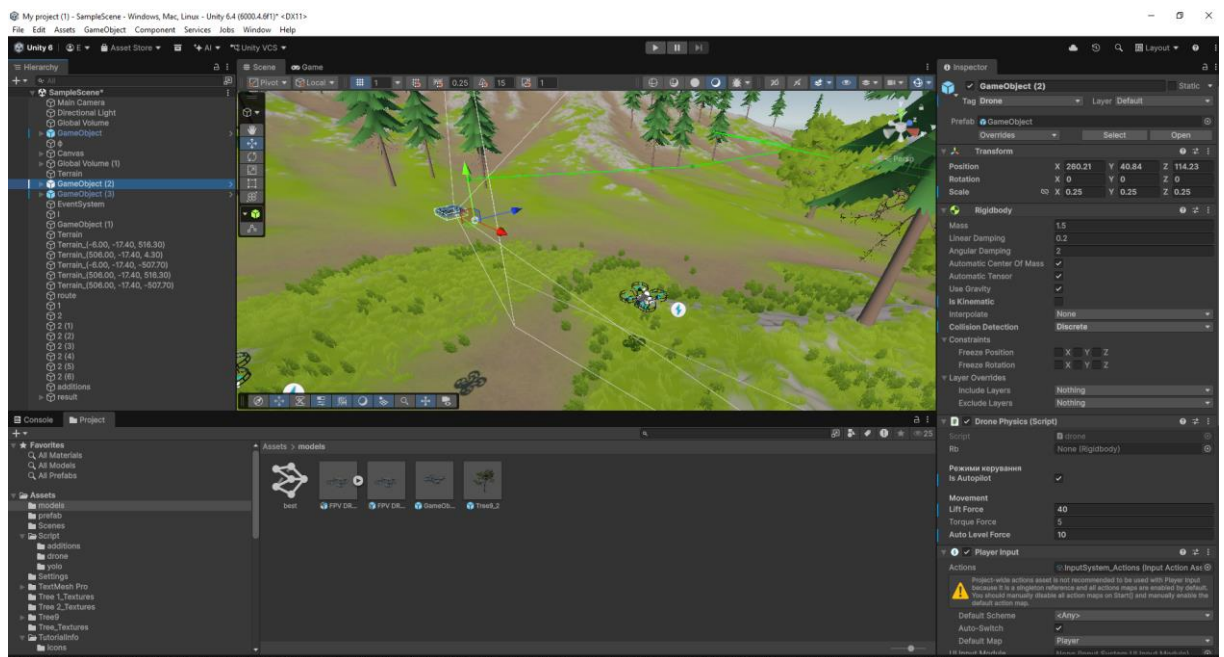


Рисунок 2.7 – Редактор рушія Unity

Цей рушій характеризується помірними системними вимогами та можливістю створювати непогану графіку, але без фотореалізму. В якості основної мови програмування він використовує C#. А завдяки тому, що він був створений досить давно, для нього існує велика кількість документації, додаткових інструментів, бібліотек тощо.

Тепер розглянемо рушій Unreal Engine [18].

Unreal Engine – це набір інструментів для 3D розробки та створення відео, який також дозволяє вести кросплатформенну розробку, зображення редактора цього рушія наведено на рисунку 2.8. Цей рушій, на відміну від попереднього, характеризується високими системними вимогами, та фотореалістичною графікою з використанням технологій Lumen та Nanite, в якості мови програмування ж цей рушій використовує C++. Аналогічно з Unity, завдяки тому, що він з'явився досить давно, кількість довідкових матеріалів для нього досить велика.

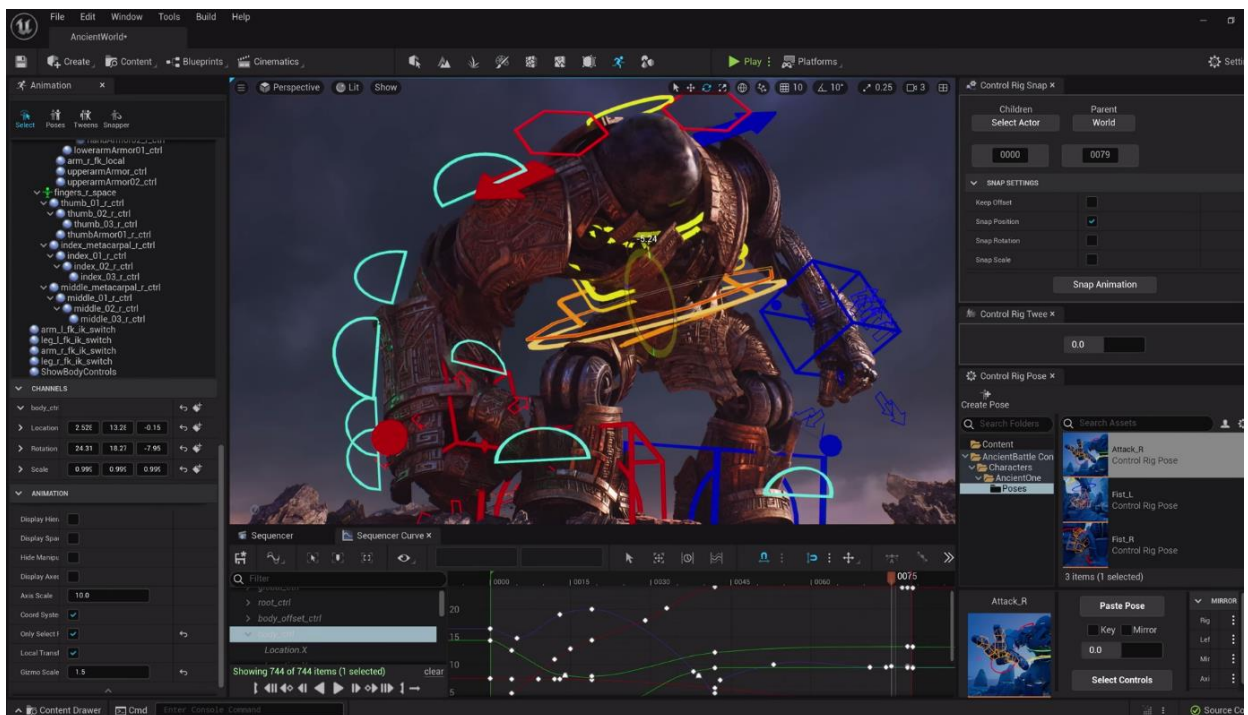


Рисунок 2.8 – Редактор рушія Unreal Engine

Наступним було розглянуто рушій Godot [19]

Godot – це рушій орієнтований на 2D та 3D розробку з можливістю кросплатформенної розробки, зображення редактора цього рушія наведено на рисунку 2.9.

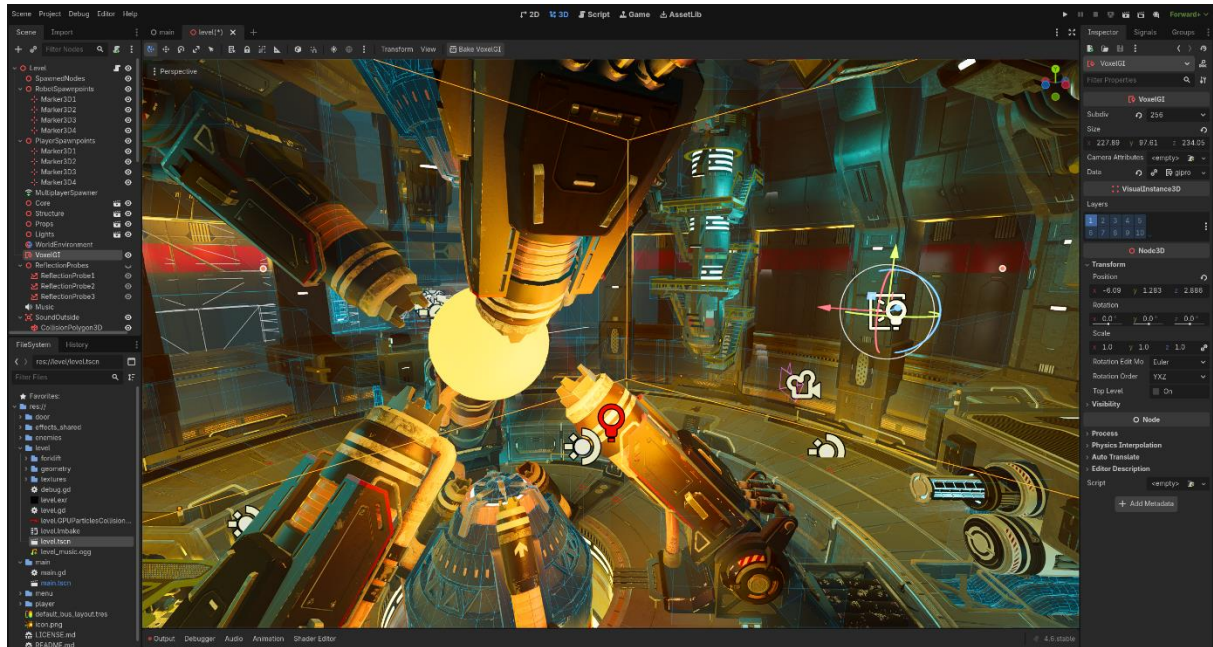


Рисунок 2.9 – Редактор рушія Godot

Його характеризують низькі системні вимоги, гарна 2D та непогана 3D графіка, а в якості мови програмування він підтримує як C++, так і C#. Але на жаль, через те, що він вийшов лише декілька років тому, кількість документації та додаткових бібліотек, інструментів для нього значно менша в порівнянні з іншими.

Враховуючи розглянуті особливості, було обрано Unity, через те що він має значно нижчі системні вимоги, ніж Unreal Engine, а також має значно більший вибір довідкових матеріалів в інтернеті, ніж Godot.

2.3 Використані мови програмування

Враховуючи те, що в якості рушія був вибраний Unity, очевидно, що в якості основної мови програмування буде використано C# [9], оскільки це мова,

скрипти якою рушій підтримує за замовчуванням. На цій мові були розроблені такі елементи програми як:

- фізика польоту та керування дрону.
- симуляція вітру.
- взаємодія нейронної мережі з симуляцією та вивід отриманого результату на екран.
- модуль, відповідальний за створення маршруту для системи навігації
- тестова система навігації.
- скрипт, відповідальний за вивід результатів тесту.

При їх розробці були використані такі бібліотеки та пакети:

- Unity.Sentis – це пакет від Unity, що відповідає за використання нейронних мереж, у моїй програмі він відповідає за обробку зображення з дрона на графічному процесорі, згідно створено молі нейронної мережі[20].
- UnityEngine – це бібліотека, в якій знаходиться базовий простір імен та класів для Unity.
- UnityEngine.UI – це бібліотека, що використовується для роботи з користувацьким інтерфейсом.
- System.Collection.Generic бібліотека C# для роботи з колекціями.

Але оскільки для створення комп'ютерного зору потрібно натренувати нейронну мережу, яка відіграє ключову роль, у його роботі. Додатково була використана мова програмування Python [10] для виконання цього процесу. На ній було написано три невеликі програми які виконують такі функції:

- тренування нейронної мережі за попередньо створеним датасетом.
- перевірка натренованої моделі на обраному файлі.
- конвертація моделі у формат, який підтримується рушієм Unity

В результаті виконання цих програм було отримано натреновану нейронну мережу, готову до виконання поставленої задачі.

2.4 Висновки за 2 розділом

В результаті дослідження задачі створення симулятора керування безпілотним літальним апаратом з можливістю тестування групових навігаційних систем, було вирішено використовувати ігровий рушій Unity. Також була визначена основна та додаткова мови програмування. Для створення самого симулятора та допоміжних скриптів у рушії було вирішено використовувати C#, через те що, це мова яку Unity використовує за замовчуванням. Також, було вирішено використовувати Python як допоміжну мову. Такий вибір був зроблений з огляду на те, що для виконання задачі створення комп'ютерного зору, потрібно натренувати нейронну мережу. А python має дуже гарну кількість бібліотек, створених спеціально для цієї задачі, включаючи бібліотеку Ultralytics, яка створено спеціально для моделі нейронної мережі YOLOv8 nano, яка і буде використана для виконання цієї задачі. А для створення датасету необхідного для навчання нейронної мережі було вирішено використати програму Label Studio.

3. АЛГОРИТМІЧНА ТА ПРОГРАМНА ЧАСТИНА ПРОЄКТУ.

ВИКОРИСТАНІ АССЕТИ

3.1 Вибір, підготовка, навчання та інтеграція нейромережі

В якості нейромережі для комп'ютерного зору було обрано YOLOv8. Ця мережа створена компанією Ultralytics в 2023 році та спеціалізується саме на задачі створення комп'ютерного зору [5]. Основною її перевагою перед попередніми версіями є те, що вона замість вгадування рамки навколо об'єкту, намагається знайти його центр і відстань від нього до рамок, така модифікація дозволила підвищити як точність роботи моделі, так і спростити її навчання.

Для тренування нейронної моделі було вирішено створити власний датасет із зображень зроблених у симуляції камерою дрона. Загалом було створено 20 зображень різних дерев з різної дистанції та ракурсів. Щоб підготувати їх для навчання мережі, необхідно розмітити місця знаходження дерев на них власноруч, а також бісля відформатувати данні під стандарт, який приймає YOLO. Для виконання цієї задачі було використано програму Label Studio, редактора розмітки зображень на рисунку 3.1.

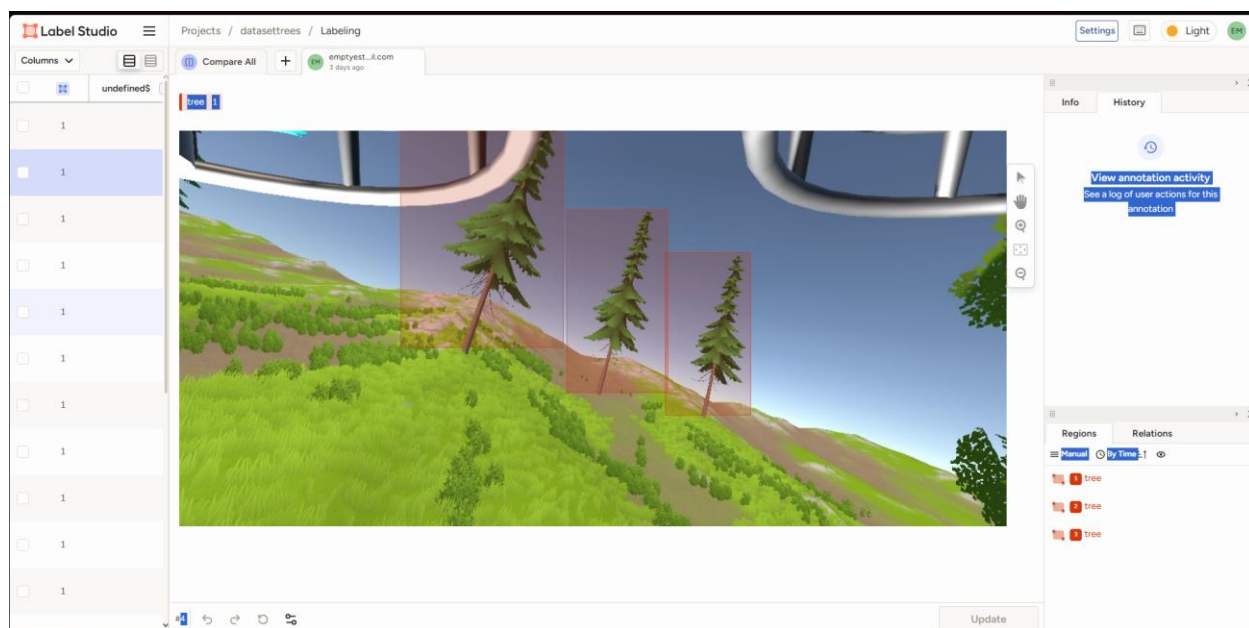


Рисунок 3.1 – Редактор програми Label Studio

Label studio це безкоштовний інструмент для створення датасетів, що встановлюється подібно іншим бібліотекам на python та відкривається у браузері. В результаті роботи з ним, був створений датасет з такою файловою структурою, яка зображена на Рисунку 3.2

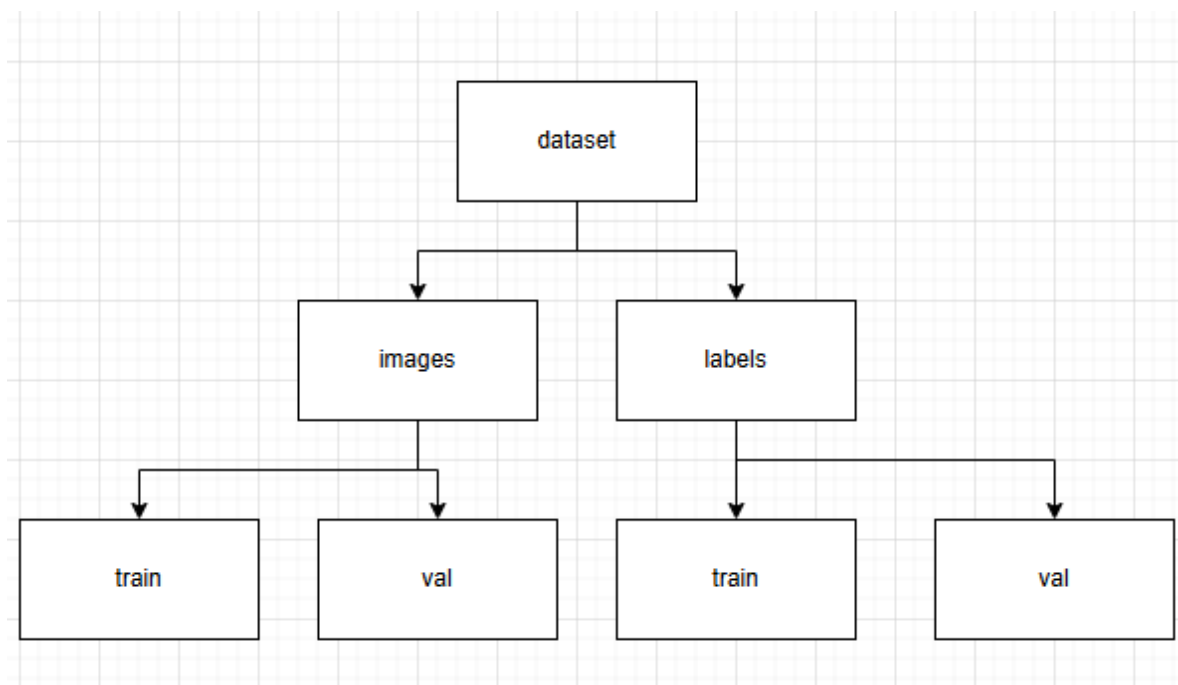


Рисунок 3.2 – Схема структури файлів датасету

У папці images знаходяться зображення, а у папці labels, в свою чергу, інформація про знаходження дерев на цих зображеннях. Інформація в обох папках ділиться на дві інші, з навчальними та перевірочними даними, train та val відповідно. Цей поділ відбувається у співвідношенні 80% до 20%, це дозволяє зберегти достатню кількість даних як для навчання, так і для впевненої перевірки результату тренування.

Навчалася нейронна мережа не з нуля, була використана базова модель yolov8n.pt, яка була попередньо натренована її розробниками розпізнавати базові геометричні фігури, текстури, тощо. Такий метод називають трансферним навчанням, завдяки тому, що при його використанні береться модель яка вже натренована для виконання базових задач, бажаного результату при навчанні вона досягає значно швидше. В моєму випадку для цього знадобилося 50 епох, у

кожній із яких аналізувалося по 16 зображень, процес навчання зображено на рисунку 3.3. Одне коло такого навчання займає 126 секунд, хоча це і досить довго, але достатній результат був отриманий вже після першого кола, проте для надійності навчання було виконано ще тричі.

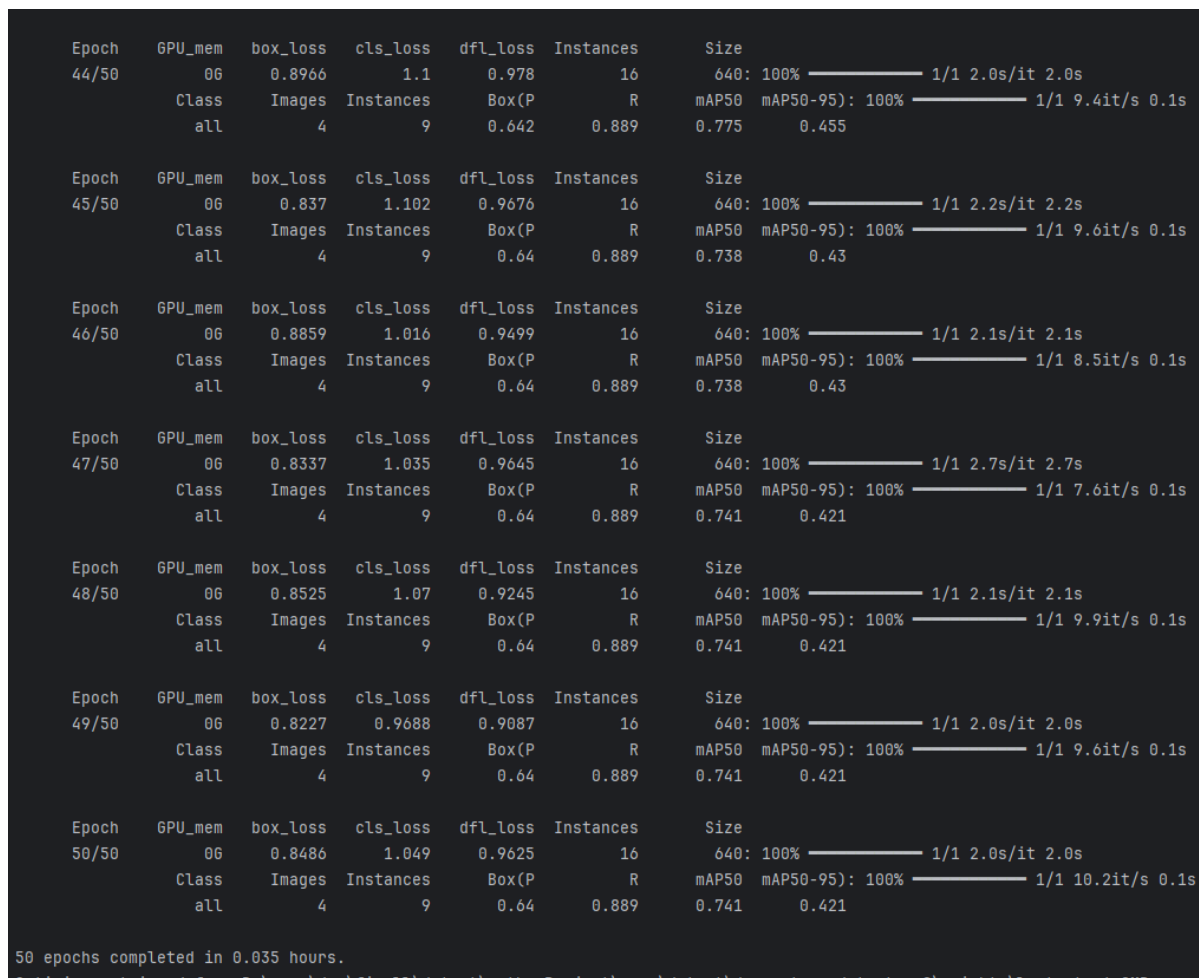


Рисунок 3.3 – Процес тренування нейронної мережі

Після того як нейронну мережу було натреновано, була використана наступна програма на мові програмування Python, для того щоб перевірити роботу нейронної мережі. Ця програма завантажує попередньо створену нейронну мережу та використовує її, для того щоб вгадати місце знаходження дерева на обраному зображенні та вивести поверх нього прямокутник, з відсотком впевненості нейронної мережі у своєму результаті. Результат роботи цієї програми зображено на рисунку 3.4.



Рисунок 3.4 – Перевірка роботи натренованої неронної мережі

Після цього нейромережу було інтегровано до Unity. Для цього було використано Unity.Sentis, що дозволило запускати нейронну мережу напряду в Unity, без використання зовнішньої програми на Python.

3.2 Опис загальної взаємодії класів

Усього при розробці програми було створено 7 класів які взаємодіють між собою, а саме:

- Result – це клас, що відповідає за вивід результату роботи програми.
- windManager – це клас, що відповідає за створення та різноманітні налаштування вітру.
- PathManager – клас, відповідальний за прокладення маршруту для навігації.
- DroneNavigator – клас, що у свою чергу відповідає за саму навігацію
- YoloUpdate – це клас, що відповідає за виведення зображення з камери дрона з результатом роботи нейронної мережі на екран.

- DronePhysics – це клас, відповідальний за основну фізику та керування дрона.
- YoloDetector – це клас відповідальний за аналіз нейронною мережею зображень та занесення результату в колекцію.

В результаті запуску програми насамперед, для кожного дрону запускається по два скрипти, у яких знаходяться класи DronePhysics та DroneNavigator. DronePhysics передбачає можливість керування користувачем, проте якщо увімкнути режим автопілота, або ж поставити відповідний прапорець перед запуском програми, керування буде прийматися від класу DroneNavigator.

Також при ініціалізації камери кожного дрону створюються скрипти, YoloDetector та YoloUpdate. Першим працює YoloDetector, отримуючі дані з камери у розширенні 640x640 та знаходячи на зображенні дерева, після ці дані використовуються класами YoloUpdate та DroneNavigator. YoloUpdate, у свою чергу отримавши результат роботи класу YoloDetector, виводить у лівий верхній куток екрану зображення з камери першого дрону з нанесеними на неї мітками над розпізнаними деревами.

Скрипти з класами Result, windManager та PathManager запускаються в одиничному екземплярі. Клас windManager використовується для попереднього налаштування швидкості та напрямку вітру, а також для його імітації його мінливості. Отримані зміни в силі вітру виводяться в публічну змінну, до якої звертається функція FixedUpdate у класі DronePhysics для кожного окремого дрону.

Клас PathManager необхідний для зберігання інформації про точки маршруту та прокладання маршруту по цим точкам, цей маршрут можна бачити на екрані до запуску симуляції. За інформацією про кількість точок та їх координатами до нього звертається функція FixedUpdate із класу DroneNavigator.

Клас Result зберігає інформацію про кількість вилетівших та долетівших дронів. Цю інформацію їй надає кожен дрон за допомогою класу DroneNavigator при старті та фініші. Інформація про старт надається при ініціалізації класу за допомогою функції Start(), а інформація про фініш в свою чергу надається у

функції FixedUpdate(), у той момент, коли змінна isFinished отримує значення true.

Схема взаємодії усіх класів програми знаходиться у графічному додатку 1.

3.3 Детальний опис алгоритму роботи кожного класу.

Клас DronePhysics, як було написано в попередньому підрозділі, це клас, який описує фізику дрона, його керування, а також саме через нього на його політ впливають інші класи. Для реалізації сили тяжіння та інерції був використаний Rigidbody це компонент Unity, який дозволяє надати об'єкту фізичні властивості, а також взаємодіяти з ними через відповідні функції. Напрямую через нього дрону було надана маса від розміру якої залежить, як сильно сила тяжіння тягне дрон до землі, а також були налаштовані показники швидкості гасіння інерція. Зображення меню налаштувань Rigidbody зображено на рисунку 3.5.

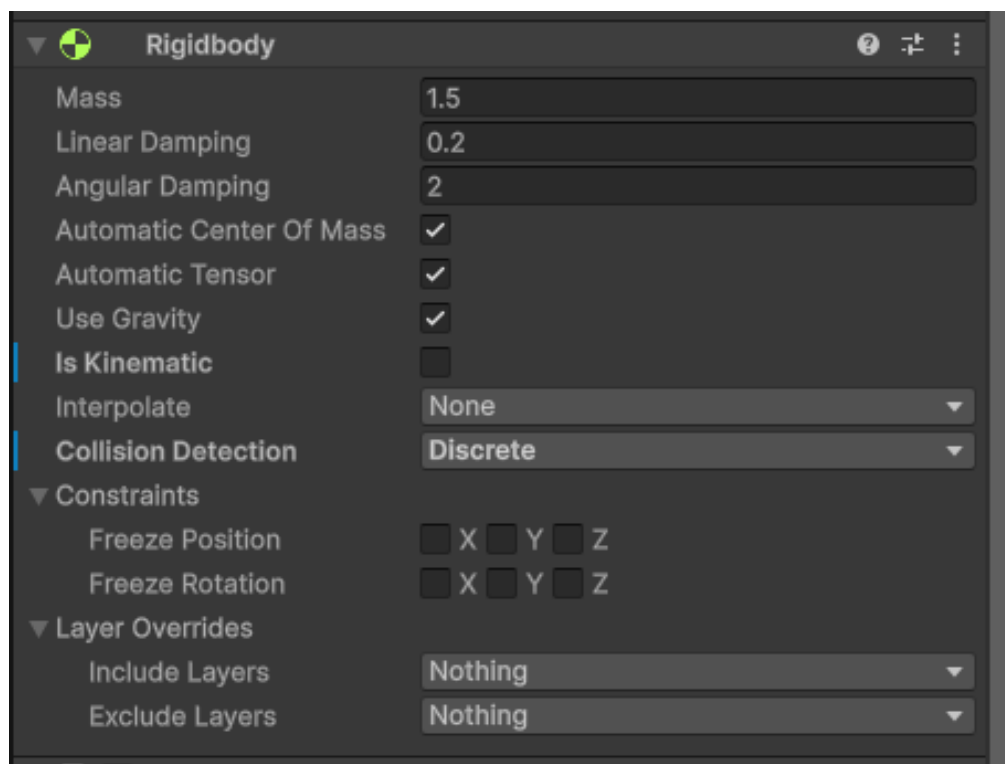


Рисунок 3.5 – Налаштування Rigidbody

Далі завдяки набору функцій які надає Rigidbody [8] у функції FixedUpdate було реалізовано ручне керування дроном, блок схема цієї функції зображена на рисунку 3.6.

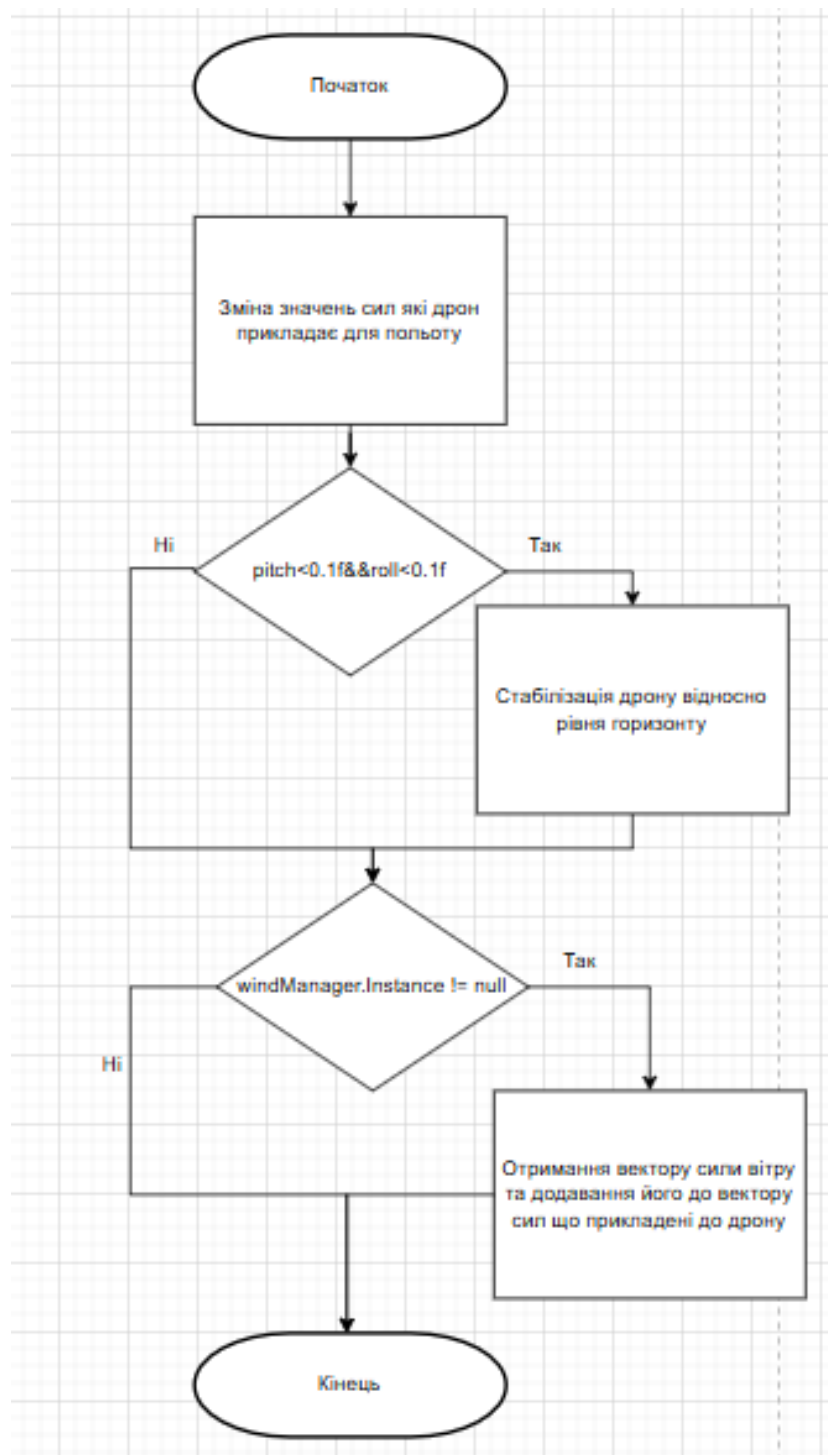


Рисунок 3.6 – Блок схема функції FixedUpdate

Наступним цій функції була реалізована автоматична стабілізація, тобто за умови будь-якого іншого вводу дрон автоматично вирівнюється до рівня

горизонту. Для налаштування сили, з якою відбувається стабілізація була створена змінна `autolevelforce` в залежності від вказаного числа вирішується, наскільки високу тягу буде використовувати дрон для стабілізації і, відповідно, як швидко буде відбуватися цей процес. На додавання такої функції мене надихнув DJI Mavic [3].

Останнім в цю функцію було додано зчитування з класу `WindManager` вектору вітру та додавання впливу його сили до вектору руху дрону.

Тепер розглянемо функцію `Update` у ній до змінних, пов'язаних з керуванням дроном, були прив'язані клавіші а саме:

- `Throttle` – це тяга, для керування нею були використані клавіші `'space'` та `'ctrl'`.
- `Pitch` – це тангаж, для керування ним була обрана горизонтальна вісь, на яку потім в налаштуваннях unity було призначено клавіші `'w'` та `'s'`.
- `Roll` – це крен, для керування ним було обрано горизонтальну вісь, на яку потім в налаштуваннях unity було призначено клавіші `'a'` та `'d'`.
- `Yaw` – це рискання, для керування ним було призначено клавіші `'q'` та `'e'`.

Окрім цього, у цій функції було призначено клавішу для вмикання/вимикання автопілота, це клавіша `'M'`. На рисунку 3.7 зображено лістинг цього фрагменту.

```
26 void Update()
27 {
28     // Перемикання режиму клавішею 'M'
29     if (Input.GetKeyDown(KeyCode.M))
30     {
31         isAutopilot = !isAutopilot;
32         Debug.Log("Автопілот: " + isAutopilot);
33     }
34     if (!isAutopilot)
35     {
36         throttle = 0;
37         if (Input.GetKey(KeyCode.Space)) throttle = 1;
38         if (Input.GetKey(KeyCode.LeftControl)) throttle = -0.5f;
39
40         pitch = Input.GetAxis("Vertical");
41         roll = Input.GetAxis("Horizontal");
42
43         yaw = 0;
44         if (Input.GetKey(KeyCode.Q)) yaw = -1;
45         if (Input.GetKey(KeyCode.E)) yaw = 1;
46     }
47 }
```

Рисунок 3.7 – Лістинг коду з призначенням клавіш керування

А для налаштування потужності моторів дрону у цей клас було додано публічні змінні `torqueForce` та `liftForce`, значення в цих змінних відповідають за те, з якою силою дрон може розвертатися та набирати вертикальну швидкість, їх значення можна вказати як в коді напряду, так і через меню редактора unity для кожного дрону окремо.

Тепер розглянемо клас `DroneNavigator`, він був створений для демонстрації роботи навігаційної системи у створеному симуляторі. Через те, що створена мною система створювалась лише для демонстрації можливостей симулятора, вона має певні недоліки, які були виявлені під час тестування проте з функцією демонстрації можливостей вона виконує на належному рівні.

Цей клас складається з 4 функцій одна основна що відповідає за політ, дві допоміжні що відповідають за визначення необхідності та напрямів ухилення від дерев та інших дронів, а також функція, що відповідає за ініціалізацію класу. Розглянемо детально роботу допоміжної функції `GetYoloAvoidance`, вона відповідає за визначення необхідності ухилення від побаченого дерева та напряду ухилення від нього. Спочатку ця функція, перевіряє чи перешкоди взагалі наявні на екрані, після вона перевіряє кожну з перешкод, записаних у колекцію у класі `YoloDetector`. Для визначення необхідності ухилення від дерева перевіряється, наскільки близько воно до центру зображення, такий спосіб обраний, оскільки камера направлена по курсу польоту дрона, а отже, об'єкт на зображенні з цієї камери знаходиться по центру або близько до нього, він знаходить ся по курсу польоту дрона. Для визначення курсу ухилення, перевіряється на якій половині зображення знаходиться більша частина перешкоди, якщо справа потрібно летіти наліво, якщо ні летимо на право. Для того щоб дрон не витрачав час на об'єкти далеко від центру екрану, було створено сліпу зону, її можна окрема налаштувати по необхідності, як приклад за умови використання більшої моделі дрона, її потрібно буде зменшити оскільки дрон не буде достатньо сильно повертати та при прольоті повз

перешкоду не буде набирати необхідну відстань. Блок схема цієї функції зображена у додатку.2

Тепер розглянемо основну функцію FixedUpdate, її задача полягає в тому, щоб керувати дроном за умови ввімкненого автопілоту. Спочатку при кожному запуску функції перевіряється, чи дрон завершив маршрут, якщо так, автопілот починає утримувати дрон на одній висоті. В протилежному випадку перевіряється, чи увімкнений автопілот, а число точок в маршруті не рівно нулю. Далі отримується точка маршруту, до якої зараз прямує дрон, та перевіряється, чи він не досягнув її. Якщо дрон знаходиться на необхідній точці маршруту, перевіряється, чи не є вона останньою, якщо так, виконується функція з класу Result для збільшення числа фінішуваних дронів на один, а дрону присвоюється значення фінішувального. В протилежному випадку до індексу цільової точки маршруту додається один. Далі виконується функція Vector3 GetSwarmSeparation(), яка надає значення зміни вектору руху необхідне для ухилення від дрону, якщо він наблизився занадто близько, чим ближче інший дрон, тим більше тяги буде прикладено для ухилення. Далі відбувається доворот дрону до напрямку бажаного вектору та прикладення тяги, для зміни вектору руху на бажаний. Наступною дією відбувається виконання функції GetYoloAvoidance для перевірки, чи необхідно ухилятися від дерева, якщо перешкода є відбувається поворот у потрібну сторону за допомогою осей рискання та крену. Останньою перевіркою в функції перевіряється, чи нахил дрона не перевищує допустимий рівень, якщо так, у пов'язані з креном та рисканням змінні встановлюється значення 0 для запобігання перевертання дрону, в залежності від мобільності тестованого дрону значення максимального нахилу можна змінювати, це дозволяє перевірити, з яким нахилом можна довірити автопілоту керувати дрон без ризику його перекидання. Останньою командою у функції відбувається передача даних керування у клас DronePhysics за допомогою функції SetAutopilotInputs. Блок схема з цією функцією знаходиться в додатку 3.

Розглянемо клас windManager, як було розглянуто в попередньому розділі, він відповідає за створення вітру. В цьому класі всього дві функції, перша відповідає за його ініціалізацію, а друга – GetWindForce за симуляцію вітра та отримання його вектору силу класом DronePhysics. Розглянемо цю функцію детальніше. Спочатку генеруються зміни вітру по 2 осям X та Y. Для цього використовується функція, яка повертає число із ряду чисел шуму Перліна на основі поточного кадру симуляції. Це дозволяє отримувати число, яке плавно змінюється від -1 до 1. Після отримане число множиться на силу змінного вітру. Це значення задається в публічній змінній, якщо необхідно вимкнути мінливий вітер, цьому значенню потрібно присвоїти значення 0. Після цього попередньо заданий напрям вітру множиться на силу вітру та після до отриманого значення додається вектор мінливого вітру. Напрямок вітру та сила вітру також є публічними змінними, які можна задати з меню об'єкта additions у підменю Wind Manager (Script) перед запуском програми. Після функція повертає це значення. Блок схема цієї функції зображена на рисунку 3.8.



Рисунок 3.8 – Блок схема функції GetWindForce

Розглянемо клас Path Manager, який був створений для прокладання та візуальної демонстрації маршруту для системи навігації. В ньому знаходиться всього одна функція, а також дві змінні: список з усіма точками маршруту та дистанцію при прольоті на якій точка вважається пройденою. Функція у цьому класі відповідає за те, щоб до запуску симуляції можна було побачити лінію, яка з'єднує точки маршрута між собою. Розглянемо більш детально алгоритм її роботи, його блок схема зображена на рисунку 3.9.

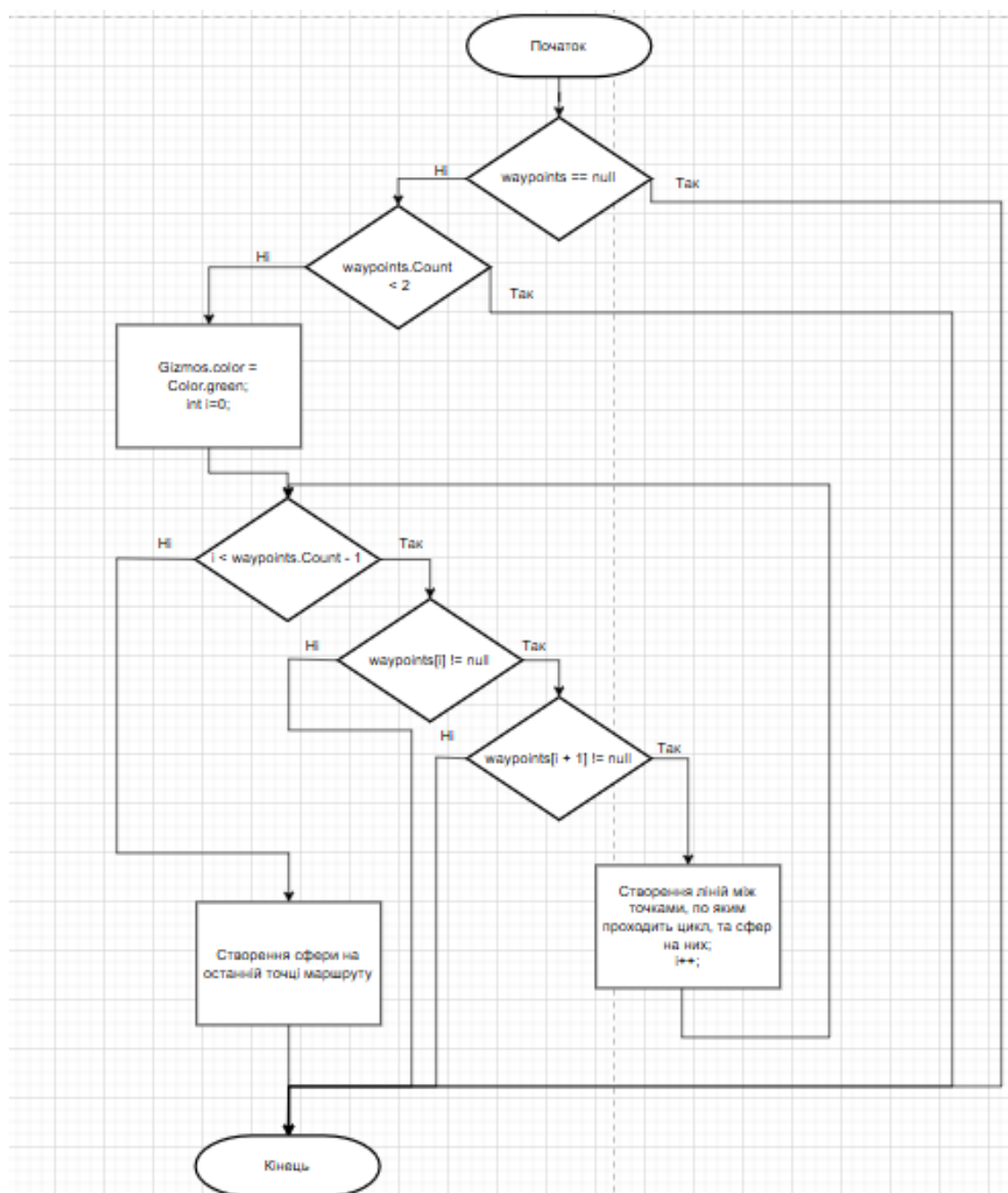


Рисунок 3.9 – Блок схема функції OnDrawGizmos

Спочатку перевіряється, чи кількість точок не нижче двох або не дорівнює null. Якщо перевірка пройдена вибирається колір мальованої лінії, а після цього починається цикл який відбувається доки не пройде по всім точкам крім останньої. В тілі циклу між кожними двома точками за допомогою Gizmos[8] малюється лінія, а також навколо кожної точки створюється сфера. Після завершення циклу на останній точці, теж створюється сфера.

Наступним класом розглянемо YoloDetection, це клас що напряму взаємодіє з камерою дрона та створеною нейронною мережею. Цей клас має 3 функції:

- start – відповідає за ініціалізацію класу.
- Update – відповідає за функціонал розпізнавання перешкод.
- OnDestroy – спрацьовує при видаленні класу, очищає відеопам'ять зупиняючи процес роботи нейронної мережі.

Окрім цього, у класі створюється публічна колекція, в якій зберігаються дані про виявлені перешкоди. Розглянемо алгоритм роботи функції Update. Спочатку створюється пуста математична матриця, з розширенням стандартним для YOLO, а саме 640 на 640 пікселів. Далі за допомогою спеціальної команди кожен піксель зображення замінюється на число від 0.0 до 1.0 та переміщається у нашу математичну матрицю. Це потрібно, щоб нейронна мережа могла аналізувати зображення. Після цього створюємо завдання для відеокарти, згідно наданої нейронної мережі обробити цю матрицю. Після цього отримуємо адресу місця, куди був записаний результат, а далі завантажуюмо його у масив outputdata. Наступним кроком очищається колекція currentTrees, а після починається цикл, що проходить по усім елементам отриманого масива, всього їх 8400. У тілі циклу відбуваються такі дії. Спочатку з масиву даних отримується значення впевненості нейронної мережі у своєму передбаченні номер і. Якщо показник впевненості більший ніж нами заданий, з масиву дістаються дані про координати цієї перешкоди та потім додаються до колекції з інформацією про

виявлені перешкоди. Після завершення роботи циклу робота функції завершується. Блок схема цієї функції зображена в додатку 4.

Тепер розглянемо клас YoloUpdate – цей клас відповідає за вивід на екран зображення камери одного з дронів, а також результату роботи класа YoloDetection поверх цього зображення. Розглянемо основну функцію цього класу OnGUI, її блок схем знаходиться на рисунку 3.10.

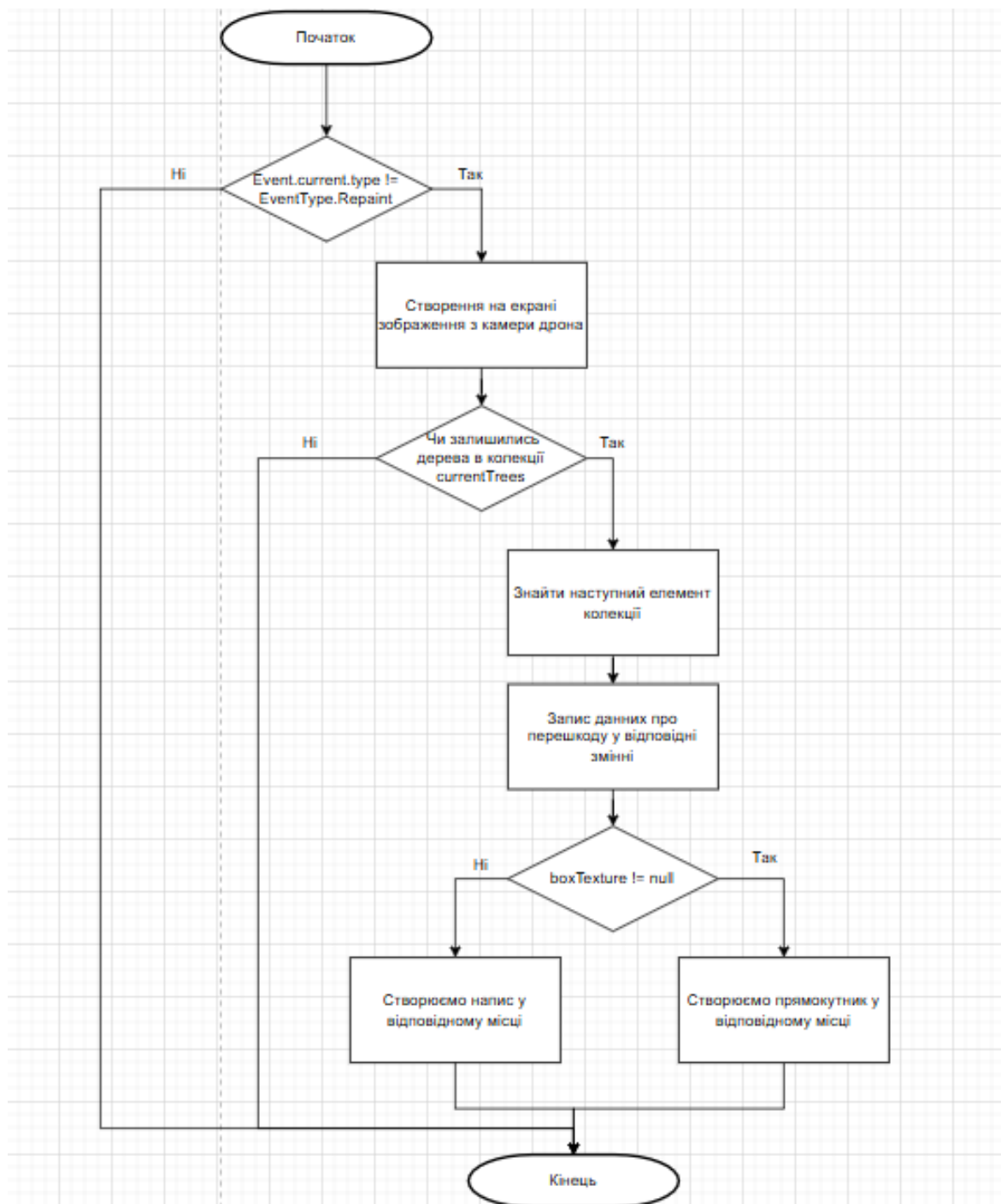


Рисунок 3.10 – Блок схема функції OnGUI

OnGUI – відповідає за виконання основної задачі класу. Спочатку перевіряється чи почався в конкретний момент рендер зображення у Unity. Якщо так, спочатку на екран виводиться поточний кадр з камери дрон, а після у цикл `for` проходить по масиву з перешкодами. Всередині циклу перевіряється, чи надана текстура для створення прямокутника на зоні з перешкодою. Якщо так, поверх перешкоди виводиться прямокутник, це зроблено для того, щоб наочно побачити, як дрон бачить перешкоду. Якщо ж текстура відсутня, над перешкодою просто виводиться напис про її присутність, такий режим дозволяє сфокусуватися саме на польоті дрону.

Також у класі присутня публічна змінна `Texture2D boxTexture`, у яку за допомогою вбудованого інструментарію Unity завантажується зображення з камери відповідного дрону та функція ініціалізатор, яка виконується при початку роботи класу.

Далі розглянемо клас `Result` – він відповідає за виведення результату тестування, а саме числа вилетівших та долетівших дронів. В цьому класі є 4 функції:

- `Start` – функція, що виконується при ініціалізації програми.
- `RegisterDrone` – функція, що викликається у класі `DroneNavigator` в функції ініціалізації, при створенні дрона з цим класом.
- `DroneFinished` – функція, що виконується в класі `DroneNavigator` в основній функції при досяганні дроном фінальної точки.
- `Update` – основна функція класу, яка за умови натиснутої клавіші `TAB` виводить результати на екран.

Розглянемо алгоритм роботи функції `Update` більш детально. Найперше перевіряється чи була натиснута клавіша `tab` протягом поточного кадру. Якщо так, наступним перевіряється, чи було надано класу вікно для виводу інформації. Після цього виконується серія команд, що змінює поточний стан екрану на протилежний, зображення їх лістингу наведено на рисунку 3.11.

44				<code>bool isActive = resultWindow.activeSelf;</code>
45				<code>resultWindow.SetActive(!isActive);</code>

Рисунок 3.11 – команди для зміни поточного стану вікна

Після цього перевіряється, чи надано текстове вікно для написання тексту, а також чи в результаті попередньої дії вікно стало активним. Якщо отримано відповідь True, у текстове вікно виводяться результати симуляції, з отриманими на даний момент числами вилетівших дронів та долетівших. Блок схема описаної функції зображена на рисунку 3.12.

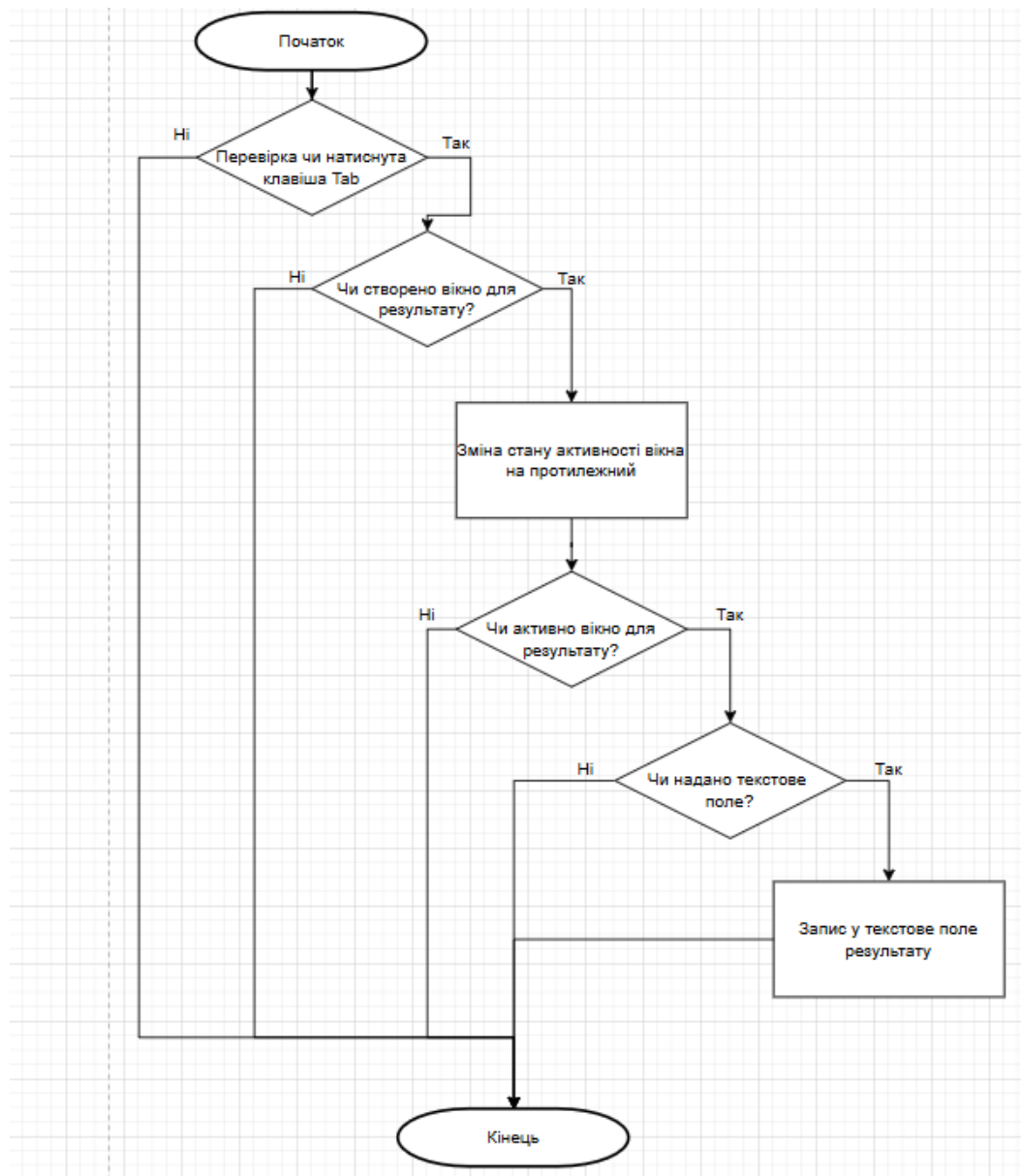


Рисунок 3.12 – Блок схема функції result

3.4 Ассети використані при створенні зовнішньої оболонки.

Для пошуку ассетів було використано офіційні сайти Unity Asset Store та cgtrader на яких можна як продавати, так і купувати різноманітні ассети, головним принципом при виборі ассетів була наявність вільного доступу до нього та підтримка цього ассету рендерером який використовується у проєкті.

Найпершим в проєкт було додано ассет з моделлю FPV дрона [14]. Нажаль серед безкоштовних варіантів був знайдений лише файл у форматі програми 3D моделювання Blender, тому перед додаванням у проєкт він був конвертований у розширення, яке підтримує Unity, зображення отриманої моделі у русії знаходиться на рисунку 3.13.

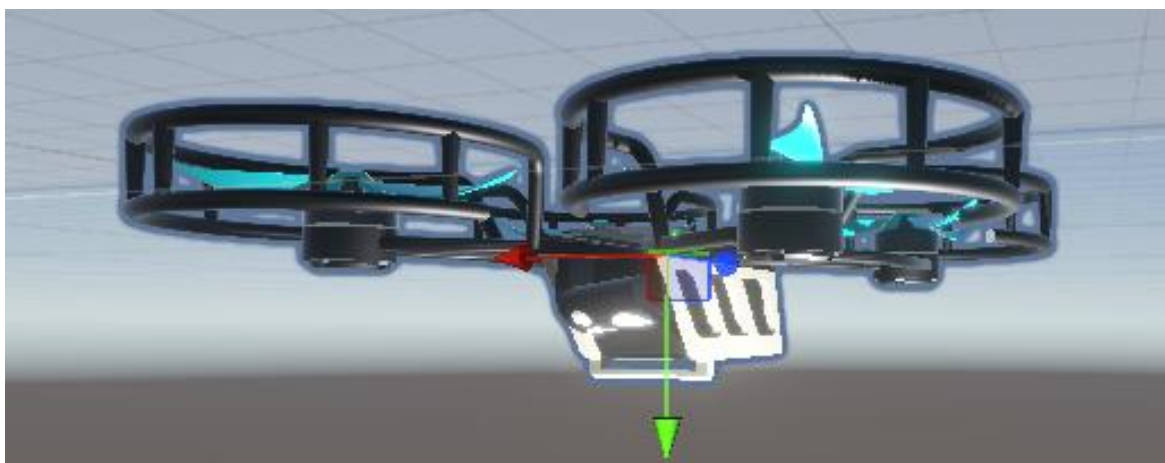


Рисунок 3.13 – Завантажений у Unity ассет дрону.

Далі було завантажено інструмент для створення довкілля, за допомогою якого, спочатку було створено два види дерев, а після заповнено ними попередньо редактовану за допомогою вбудованих інструментів Unity площину, по якій будуть літати дрони. Для створення дерев було вирішено обрати саме інструмент замість готових ассетів, через те що в подальшому він дозволить створити безліч унікальних дерев, на відміну від ассетів які не передбачають їх швидкої зміни. Цей інструмент називається Vegetation Spawner[15].

Зм	Лист	№ докум.	Підп.	Дата

3.5 Висновки за 3 розділом.

Під час розробки цього програмного забезпечення було використано дві мови Python та C#. На C# було створено 7 скриптів, у кожному з яких був створений свій клас. Основні функції кожного класу виконуються паралельно та кожного кадру. Скрипти пов'язані насамперед з роботою дрона, запускаються для кожного дрону окремо. Тому при додаванні кожного дрону кількість паралельно працюючих скриптів зростає на 3. На мові Python в свою чергу було створено три прості програми які відповідають за тренування, перевірку та конвертацію нейронної мережі. Щодо зовнішньої оболонки, для неї були використані ассети завантажені з Unity asset store, та вбудований інструментарій рушія Unity.

4. ОСОБЛИВОСТІ СТВОРЕНОГО СИМУЛЯТОРУ.ТЕСТУВАННЯ ЙОГО РОБОТИ

4.1 Особливості створеної віртуальної моделі дрону

З розглянутих особливостей будови дрона зрозуміло, що модель дрону у віртуальному середовищі насамперед має імітувати специфіку управління такого літального апарату. Цього можна досягти декількома способами:

Перший – імітувати роботу кожного двигуна окремо. Цей спосіб дозволив би імітувати пошкодження та повну втрату двигунів, проте якщо симулювати декілька дронів, він значно збільшує навантаження на систему. Другий – одразу імітувати результат роботи певних груп двигунів, тобто поворот дрону у різних осях. Що дозволить обраховувати меншу кількість сил, впливаючих на рух дрону.

З огляду на завдання був обраний другий спосіб через те, що пошкодження двигуна через врізання у об'єкт, це вже провал для системи навігації, а без необхідності це обраховувати в першому способі немає сенсу через те що він підвищує системні вимоги програми. Після надання можливості дрону літати, також була додана стабілізація, яка за умови відсутності інших команд, намагається привести дрон у горизонтальне положення, користуючись тим самим керуванням, що використовує користувач.

Наступний по важливості елемент моделі дрону це камера, вона знаходиться спереду моделі та виводить те, що бачить на екран, а також надає зображення нейронній мережі YOLOv8 для пошуку перешкод, яка у свою, чергу виводить написи над ідентифікованими перешкодами на екран, демонстрацію чого можна бачити на рисунку 4.1.

Окрім цього, було додано альтернативний спосіб виводу знайдених перешкод для того, щоб можна було наочно побачити, як комп'ютерний зір бачить зображення, приклад цього наведено на рисунку 4.2.

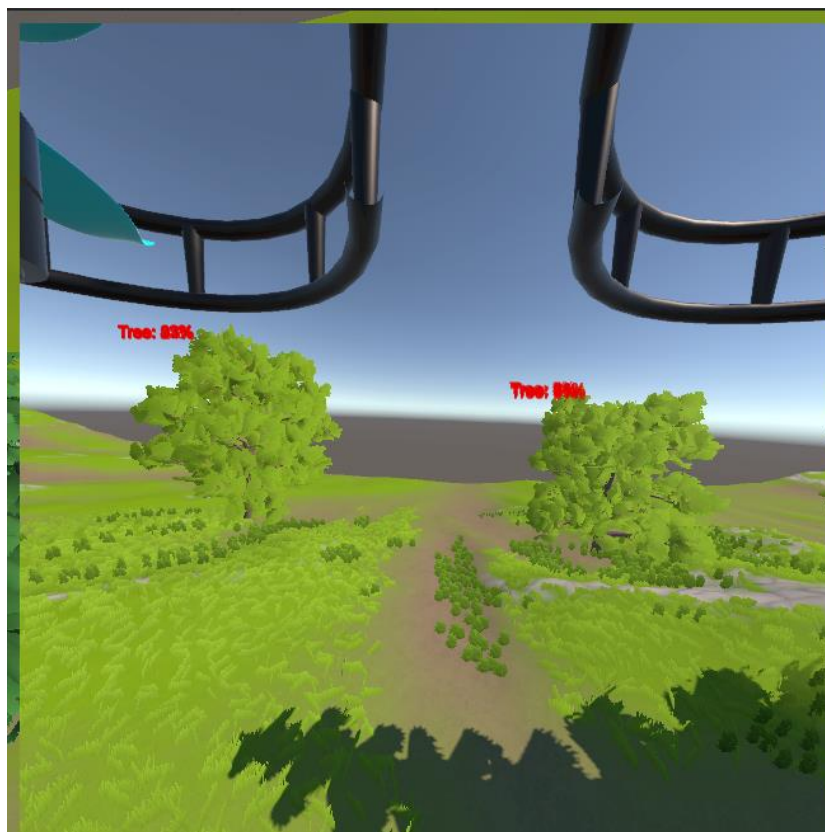


Рисунок 4.1 – Вид з камери моделі дрона з виводом з нейромережі

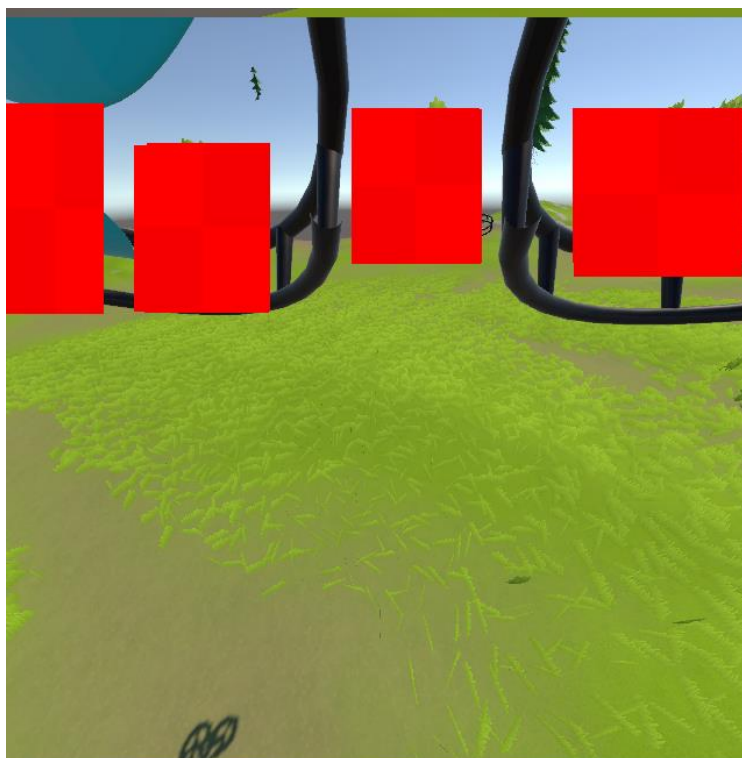


Рисунок 4.2 – Альтернативний вивід перешкод на зображення

Окрім цього, до моделі було додано базову фізику, тобто на дрон діє гравітація, яка, в незалежності від його положення, діє до низу. Також була додана фізика зіткнень, яка задля оптимізації обробляє дрон як паралелепіпед, сітка, за якою обробляються зіткнення з дроном, зображена з різних ракурсів на рисунках 4.3–4.4.

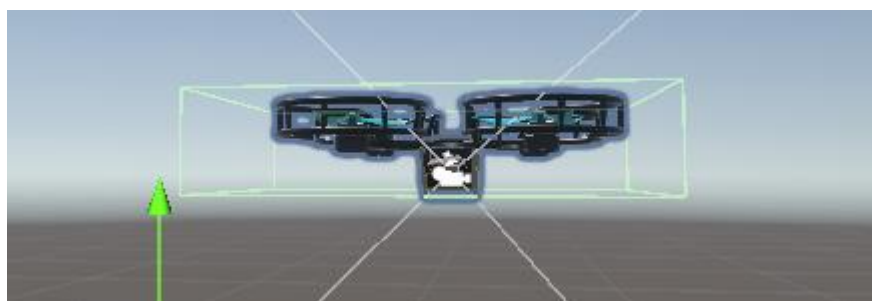


Рисунок 4.3 – Сітка колізій дрону, вид спереду

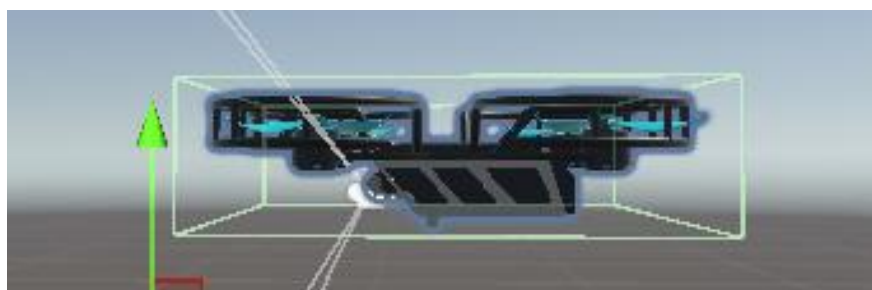


Рисунок 4.4 – Сітка колізії дрону, вид збоку

4.2 Загальні особливості симулятора

Для початку була створена проста площина з невеликими перепадами висот, в якості перешкод були вибрані дерева. Такий вибір зумовлений тим що найчастіше на великі відстані дрони літають саме рівнинами з мінімальною кількістю перешкод, щоб зайвий раз не маневрувати. Дерев ж були вибрані через те, що це досить велика перешкода, яку при цьому не завжди можна передбачити за картою. А вибір такої простої графіки, зумовлений тим що тестуватися буде саме навігаційна система, якій, на відміну від людини, не

Зм	Лист	№ докум.	Підп.	Дата

ІАЛЦ. 045490.002 ПЗ

Лист
44

важлива якість зображення. Саме тому для додаткової оптимізації була використана досить проста графіка.

Далі був доданий вивід на екран загального зображення рівнини зверху під кутом, для того щоб можна було зі сторони бачити, як відбувається політ, демонстрація цього зображена на рисунку 4.5.



Рисунок 4.5 – Зображення площини для тестових польотів зверху

А щоб підняти рівень реалізму, була симульована дія вітру, яка з обраною силою та напрямом постійно діє на дрон під час польоту, що дозволяє проводити тести в умовах, більш наближених до реальних. Для того щоб вітер був більш схожий на справжній, йому була додана можливість трохи змінюватися у будь-яку сторону на обране значення помножене число отримане з функції шуму Перліна. В свою чергу шум Перліна це фактично безкінечний набір плавно змінюваних значень від 0 до 1. Приклад цього ряду у вигляді графіка зображено на рисунку 4.6. Для знаходження необхідного нам числа ми використовуємо номер кадру від початку симуляції, що і дозволяє симулювати плавні зміни вітру наближені до реальних.

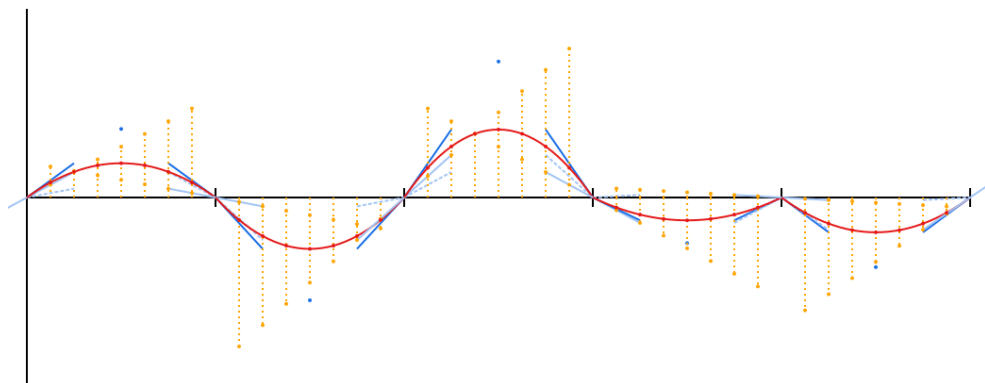


Рисунок 4.6 – Графік шуму Перліна

Щоб впевнитись, в тому що вітер працює як задумано, досить просто у симуляції на ручному керуванні підлетіти трохи вгору, і дрон одразу почне зносити у вказану сторону.

4.3 Тестування навігації

Навігація в моєму проєкті фактично створена з двох елементів маршруту та системи, яка намагається за ним слідувати. За для зручності була створена можливість включення та вимкнення системи навігації на клавішу 'm'.

В якості системи навігації використовується скрипт на мові програмування C#. В цей скрипт записана моя базова система навігації, яка створена для тестування, але його можна замінити на будь-яку інший, і хоча це не дозволить розробити повноцінну навігаційну систему з нуля, це дозволить протестувати на практиці різні алгоритми та ідеї для системи навігації і, у разі успіху, створювати систему за вже перевіреним алгоритмом, що знизить шанс виникнення проблем на етапі тестування вже розробленої системи. Щодо групової навігації, у моєму прототипі системи навігації, при наближенні до іншого дрону, дрон намагається летіти в іншу сторону що хоч і не є ідеальним рішенням, але демонструє факт можливості тестувати не тільки політ окремого дрону, а й цілої групи. Розмір тестованої групи, у свою чергу, фактично обмежений лише можливостями комп'ютера.

Маршрут для цього скрипта, у свою чергу, створений за допомогою системи невидимих точок, за якими по черзі слідує система, їх зображення наведено на рисунку 4.7.



Рисунок 4.7 – Шлях заяким намагається летіти навігатор

І хоча швидкого способу додавати ці точки створено не було, їх все ще можна створювати нескінченну кількість. Для цього потрібно створити пустий об'єкт та додати його у меню Path manager до об'єкта route.

4.4 Завершення польоту

Для простого підбиття підсумків було створено текстове меню із статистикою польоту, яке можна викликати у будь-який момент тестування або після його завершення на клавішу “ТАВ”. Зовнішній вигляд цього меню зображено на рисунку 4.8. У цьому меню буде виведено кількість дронів, що

вилетіли, а також кілськість тих що вже долетіли. Це дозволить прибрати необхідність спостерігати за кожним дроном, щоб бути впевненим у результаті тесту .

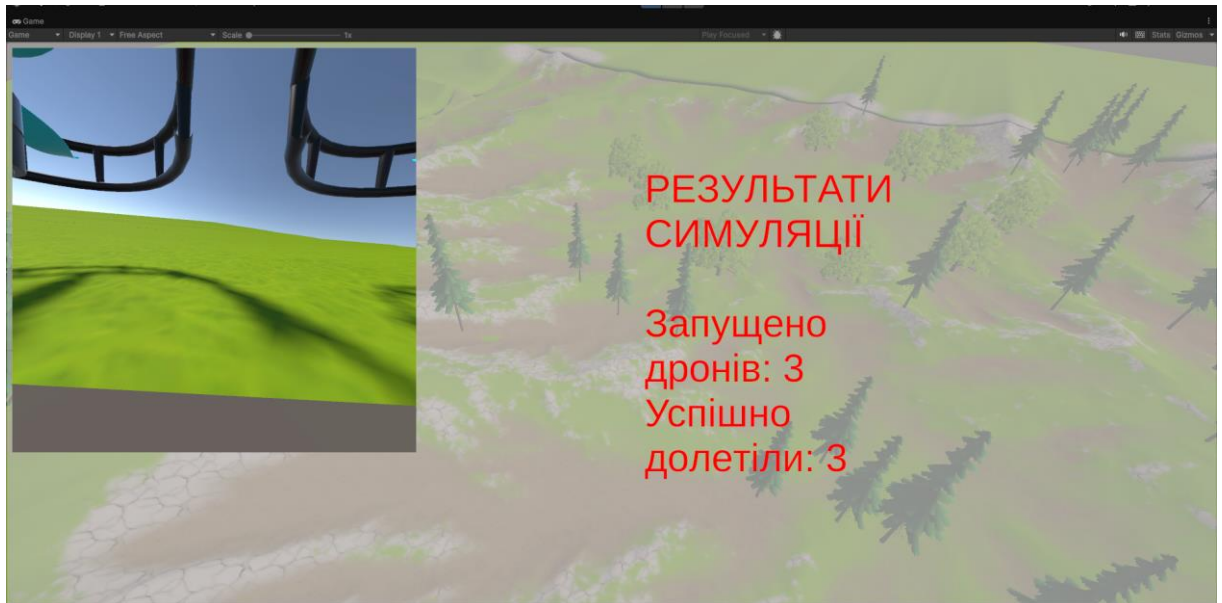


Рисунок 4.8 – Вікно з результатами симуляції

4.5 Тестування роботи симулятора

Для початку було вирішено протестувати роботу ручного керування та системи виводу зображення з камери дрону з нанесеними на нього результатами роботи нейронної мережі. Під час тестування ручного керування проблем виявлено не було, а нейронна мережа стабільно та безпомилково позначала усі дерева на зображенні з камери. Вітер та колізії з іншими об'єктами також відпрацювали так, як задумано.

Далі було протестовано прототип навігаційної системи. В результаті тестів з самим симулятором проблем виявлено не було. А от з навігаційною системою була виявлено одну проблему. Що фактично і було задачею створеного ПЗ для тестування систем навігації. Розглянемо цю проблему більш детально, перша з них полягає, у тому що дрони не запам'ятовують позиції дерев та при розвороті на 90-180 градусів можуть врізатися в них через силу інерції, через те що вони його перестають бачити.

4.6 Порівняння створеного ПЗ з попередньо розглянутими програмами

Порівняння створеного ПЗ з програмами розглянутими у першому розділі наведено на таблиці 4.1.

Таблиця 4.1 – Порівняння створеного ПЗ з існуючими симуляторами

Назва симулятора	Фізика	Графіка	Імітація роботи радіозв'язу	Вбудована можливість тестування автопілота
LiftOff	Реалістична, але є проблеми з зіткненнями	середня	-	-
Uncrashed	реалістична	Фото-реалізм	Базовий рівень	-
UFDS	реалістична	реалістична	Детальна, з імітацією РЕБ	-
Створене ПЗ	Реалістична, але без детальної обробки зіткнень	Низько-середня	-	Наявна, з можливістю тестування групової навігації

У розробленому симуляторі, була створена фізика наближена до реальної так само, як і в інших симуляторах, але без детального прорахунку зіткнень, тобто при ударі не враховується якою частиною та об яку поверхню вдарився дрон. Графіка була обрана низько-середня оскільки це дозволить значно розвантажити відеокарту, що потрібно, оскільки через роботу нейронної мережі на неї поступить значне додаткове навантаження. Імітація роботи радіозв'язу нажаль розроблена не була. В якості основної відмінності, в програму була додана вбудована можливість тестування групових систем навігації.

4.7 Аналіз можливих покращень

Найочевиднішим способом покращити створене програмне забезпечення буде збільшення кількості різних перешкод, а також додавання вбудованого редактору полігону. Також буде доброю ідеєю створити скрипт, який буде аналізувати ефективність прийнятих дій навігатора кожного дрону та виводити в кінці не лише кількість долетівших дронів, а й те наскільки сильно дрон відхилився від оптимальної траєкторії. Наступним можна створити, за прикладом існуючих симуляторів, можливість вибирати із різних дронів, а не лише налаштовувати існуючих за допомогою редагування змінних. А після можна додати можливість збирати дрон із обраних запчастин. Наступною ідеєю, можна реалізувати можливість заміняти навігаційну систему разом з моделлю нейронної мережі прямо в процесі виконання програми. Також непоганим покращенням, буде вілагодження створеного прототипу навігаційної системи, для того, щоб він слугував нижньою планкою якості для систем, що будуть тестуватися у подальшому. Ще одним непоганим покращенням буде створити можливість у середині програми додавати моделі своїх перешкод, це значно підвищить рівень універсальності створеного ПЗ.

4.8 Висновки за 4 розділом

У результаті було створено непоганий симулятор, який надає можливість літати на дроні, в якому реалізована базова фізика польоту тобто у дрону є інерція, на нього впливає сила тяжіння, тощо. Також була створена симуляція мінливого вітру, мінливість якого заснована на шумі Перліна. Для виконання поставленої задачі тестування навігаційних систем та симуляції їх роботи як у реальності, була додана нейромережа, яка аналізує зображення з камери та надає інформацію цій системі. В якості заглушки для тестування була створена базова навігаційна система, яка для запобігання зіткнень просто перевіряє, з якої сторони екрану перешкода та летить у протилежну, а при наближенні до іншого дрону починає рухатися назад, що загалом не є гарним рішенням так як вже при

трьох дронах з такої системою зіткнення хоч і рідко, але стаються. Але з іншого боку це гарно демонструє, що створене програмне забезпечення не попогано справляється з поставленою задачею тестування алгоритмів на такі недоліки. Що і вирізняє на фоні усіх інших симуляторів, які впершу чергу створювалися для тренування пілотів.

В результаті аналізу можливих покращень було розглянуто декілька ідей які дозволили б розвинути симулятор в майбутньому, найпершою з яких було створення вбудованого редактору полігону та розширення вибору доступних для використання моделей перешкод.

ВИСНОВКИ

У ході виконання дипломного проєкту, для початку було проаналізована програми найбільш близькі по функціоналу до створюваного програмного забезпечення. В результаті аналізу було виявлено, що для задачі тестування навігаційних систем ще не створено жодної програми. Тому було вирішено сфокусуватися саме на цьому аспекті програми. Серед інших важливих деталей що були виявлені в результаті аналізу варто виділити те що в такому роді програм найважливішим аспектом є емуляція фізики, тому її створювання отримало таку ж саму пріоритетність як і можливість тестування навігаційних систем.

При розгляді особливостей роботи та будови, самого FPV дрона було виявлено що летіти він може лише вверх, а отже для набору горизонтальної швидкості та поворотів потрібно нахилитися відносно горизонту, що і було реалізовано як у ручному керуванні так і керування автопілотом.

Наступним кроком було розглянуто існуючі рушії, які дозволяють створювати програми з тривимірною графікою. В результаті чого було вирішено обрати рушій Unity. Він був вибраний за те що надає необхідне співвідношення графіки та оптимізації, а також за наявність великої кількості різноманітної документації та додаткових бібліотек, інструментів створених за час його існування.

Далі в результаті розробки було створено проєкт у рушії Unity що дозволяє без зайвих проблем тестувати навігаційні системи. Для його роботи було створено 7 скриптів на мові C#, включаючи скрипт з прототипом навігаційної системи, яку в подальшому можна замінити а базову.

Після цього було протестовано роботу ПЗ, в результаті чого не було виявлено проблем з роботою створеного симулятора. З іншого боку була виявлена проблем з прототипом навігаційної системи що і було задачею створеного програмного забезпечення. А враховуючи те що програма виконала своє завдання її працездатність можна вважати підтвердженою. Програмне

					ІАЛЦ. 045490.002 ПЗ	Лист
						52
Зм	Лист	№ докум.	Підп.	Дата		

забезпечення продемонструвало стабільну роботу в незалежності від кількості одночасно запущених дронів, а також дозволило успішно виявити недолік у створеному прототипі систему навігації.

Щодо досягнених показників створеної програми:

- Створену базову графіку
- Завдяки зниженій якості зображення, в порівнянні з іншими продуктами було отримано нижчі системні вимоги ніж в інших програмах.
- Додано можливість тестування навігаційних систем що є основним досягненням.
- Створено реалістичну фізику яка в усіх аспектах, окрім зіткнень наближена до реальної.

Щодо системних вимог, програма при роботі вимагає щонайменше 4 ГБ відео пам'яті, 3500 МБ оперативної пам'яті за умови симуляції роботи лише одного дрону, та витрачає додатково по 50 МБ на кожного додаткового, а навантаження на процесор AMD Ryzen 5 7500F склало 20% та кожним додатковим дроном зростало приблизно на 0,5%. А об'єм пам'яті необхідний для зберігання цієї програми склав 85 МБ, без урахування рушія необхідного для запуску.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Є. Є. Волканін СХЕМОТЕХНІКА БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ [Електронний ресурс] URL: <https://dspace.univd.edu.ua/server/api/core/bitstreams/7249579f-d734-4f30-a5fb-10983e1e0306/content>
2. Як зібрати FPV дрон: Компоненти, інструменти, нюанси збирання [Електронний ресурс] URL: <https://drones.com.ua/blog/yak-zibraty-fpv-dron-komponenty-instrumenty-nyuansy-zbyrannya>
3. Стаття з вікіпедії про дрон DJI Mavic [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/DJI_Mavic
4. Основні компоненти FPV-дронів: призначення та функціонал [Електронний ресурс] URL: <https://eagle.com.ua/osnovni-komponenty-fpv-droniv-pryznachennia-ta-funktsional/>
5. Ultralytics YOLOv8 [Електронний ресурс] URL: <https://docs.ultralytics.com/models/yolov8#overview>
6. Підключення ESC до FPV-дрона: схема, пайка, налаштування [Електронний ресурс] URL: <https://drones.com.ua/blog/pidklyuchennya-esc-do-fpv-drona>
7. Радіозв'язок для FPV [Електронний ресурс] URL: <https://ekran.store/radiozviazok-dlia-fpv/?srsltid=AfmBOoqUZKF9hT4CUaV1zzbIwxNmARXb9A4plxFwKi07Bc0fLqZ9ld7y>
8. Unity documentation [Електронний ресурс] URL: <https://docs.unity3d.com/Manual/index.html>
9. Мова програмування C# [Електронний ресурс] URL: <https://dotnet.microsoft.com/en-us/languages/csharp>
10. Мова програмування Python [Електронний ресурс] URL: <https://www.python.org>
11. Симулятор керування дроном Liftoff [Електронний ресурс] URL: https://store.steampowered.com/app/410340/Liftoff_FPV_Drone_Racing/?l=ukrainian
12. Симулятор керування дроном Uncrashed : FPV Drone Simulator [Електронний ресурс] URL: https://store.steampowered.com/app/1682970/Uncrashed_FPV_Drone_Simulator/
13. Симулятор керування дроном Ukrainian Fight Drone Simulator (UFDS) [Електронний ресурс] URL: https://store.steampowered.com/app/2862860/Ukrainian_Fight_Drone_Simulator_UFDS_FPV_drone_simulator/
14. Використана модель дрону [Електронний ресурс] URL: <https://www.cgtrader.com/items/4453673/download-page>
15. Vegetation Spawner | FREE - Automatic Tree/Grass Placement [Електронний ресурс] URL: <https://assetstore.unity.com/packages/tools/terrain/vegetation-spawner-free-automatic-tree-grass-placement-177192>
16. Betaflight [Електронний ресурс] URL: <https://betaflight.com/docs/wiki>
17. Рушій Unity [Електронний ресурс] URL: <https://unity.com>

18. Рушій Unreal engine [Електронний ресурс] URL: <https://www.unrealengine.com>
19. Рушій Godot [Електронний ресурс] URL: <https://godotengine.org>
20. Unity Sentis [Електронний ресурс] URL: <https://docs.unity3d.com/Packages/com.unity.sentis@2.1/manual/index.html>
21. OSD Електронний ресурс] URL: <https://www.eurofest.org.ua/osd-shcho-tse-ekranne-meniu-dlia-monitoriv-i-ne-tilky/>
22. Радіoeлектронна боротьба [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Радіoeлектронна_боротьба

ДОДАТОК 1

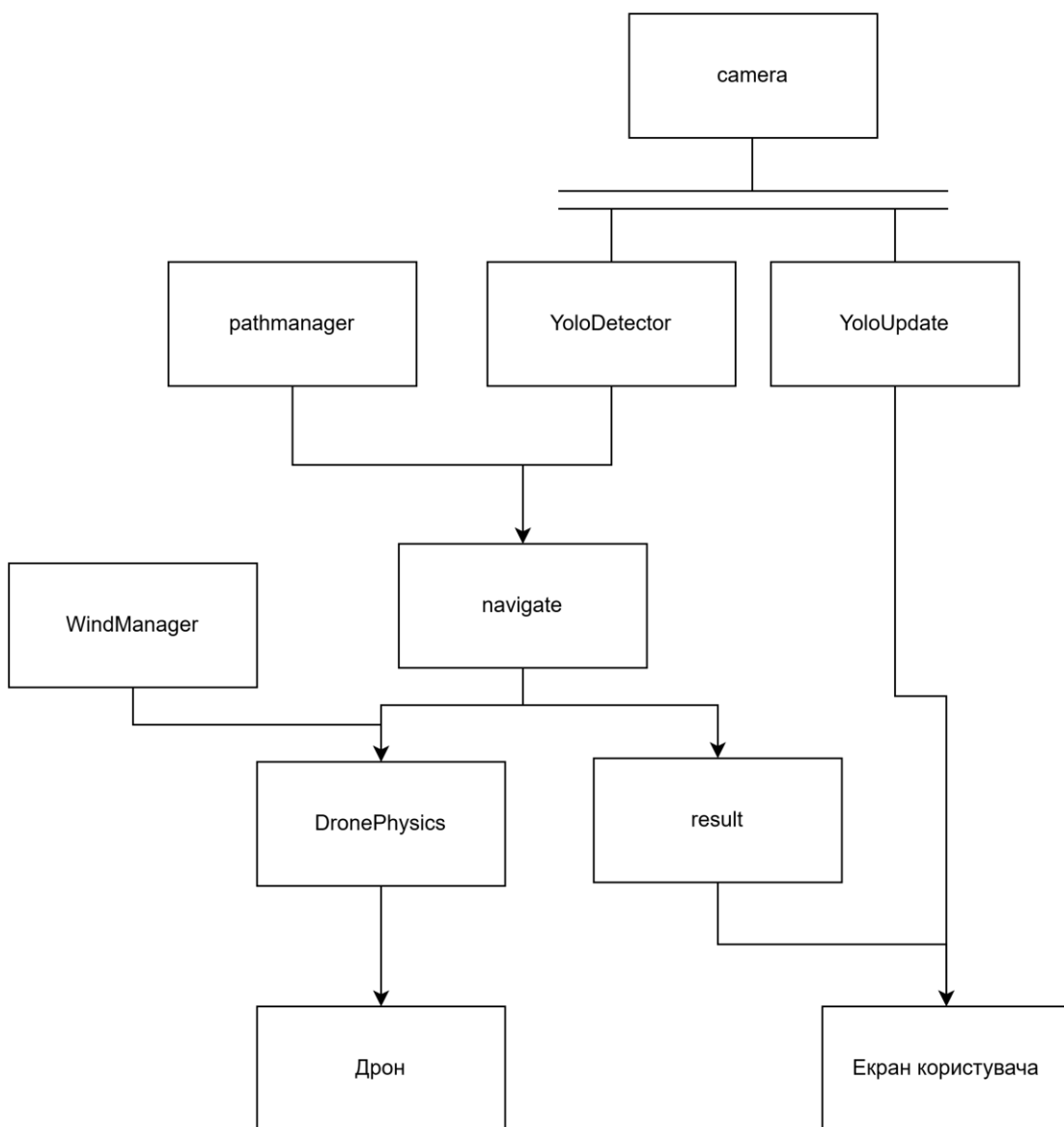
Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів

Структурна схема взаємодії класів
ІАЛЦ.045490.005 Д1

Аркушів 1

Київ - 2026

					ІАЛЦ. 045490.002 ПЗ	Лист
						56
Зм	Лист	№ докум.	Підп.	Дата		



					ІАЛЦ. 045490.005 Д1			
Змін.	Арк.	№ докум.	Підпис	Дата				
Розробив	Бурма М. С.				Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів Структурна схема взаємодії модулів програми	Літ.	Аркуш	Аркушів
Перевірів	Сергієнко П.А.						1	1
Консульт.						КПІ ім. Ігоря Сікорського, ФПСІМ КВ-21		
Н. контроль	Клятченко Я.М.							
Зав. каф.	Романкевич В.О.							

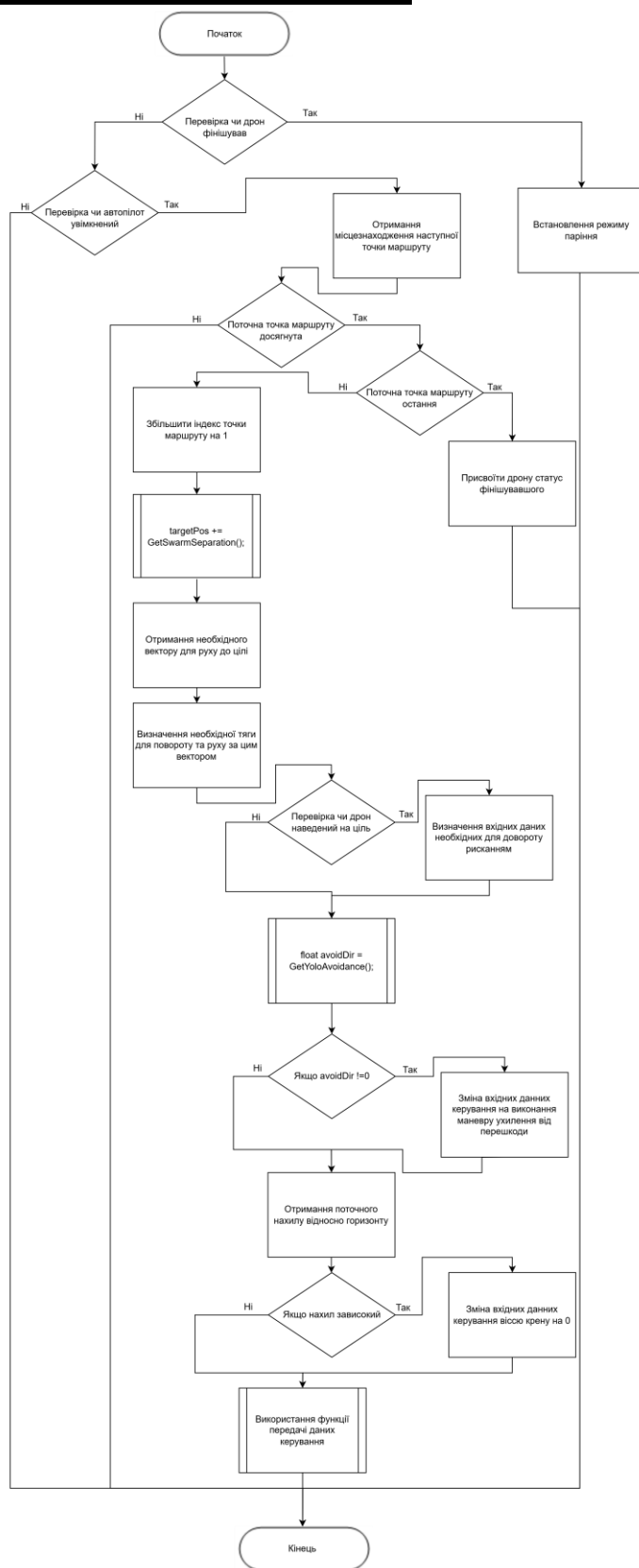
ДОДАТОК 2

Засіб розробки систем групової навігації літальних апаратів із застосуванням
розпізнавання образів

Алгоритм роботи системи навігації
ІАЛЩ.045490.006 Д2

Аркушів 1

Київ - 2026



Змін.	Арк.	№ докум.	Підпис	Дата
Розробив	Бурма М. С.			
Перевірів	Сергієнко П.А.			
Консуьлт.				
Н. контроль	Клятченко Я.М.			
Зав. каф.	Романкевич В.О.			

ІАЛЦ. 045490.006 Д2

Засіб розробки систем групової навігації
літальних апаратів із застосуванням
розпізнавання образів
Алгоритм роботи системи навігації

Літ.	Аркуш	Аркушів
	1	1
КПІ ім. Ігоря Сікорського, ФПСІМ KB-21		

ДОДАТОК 3

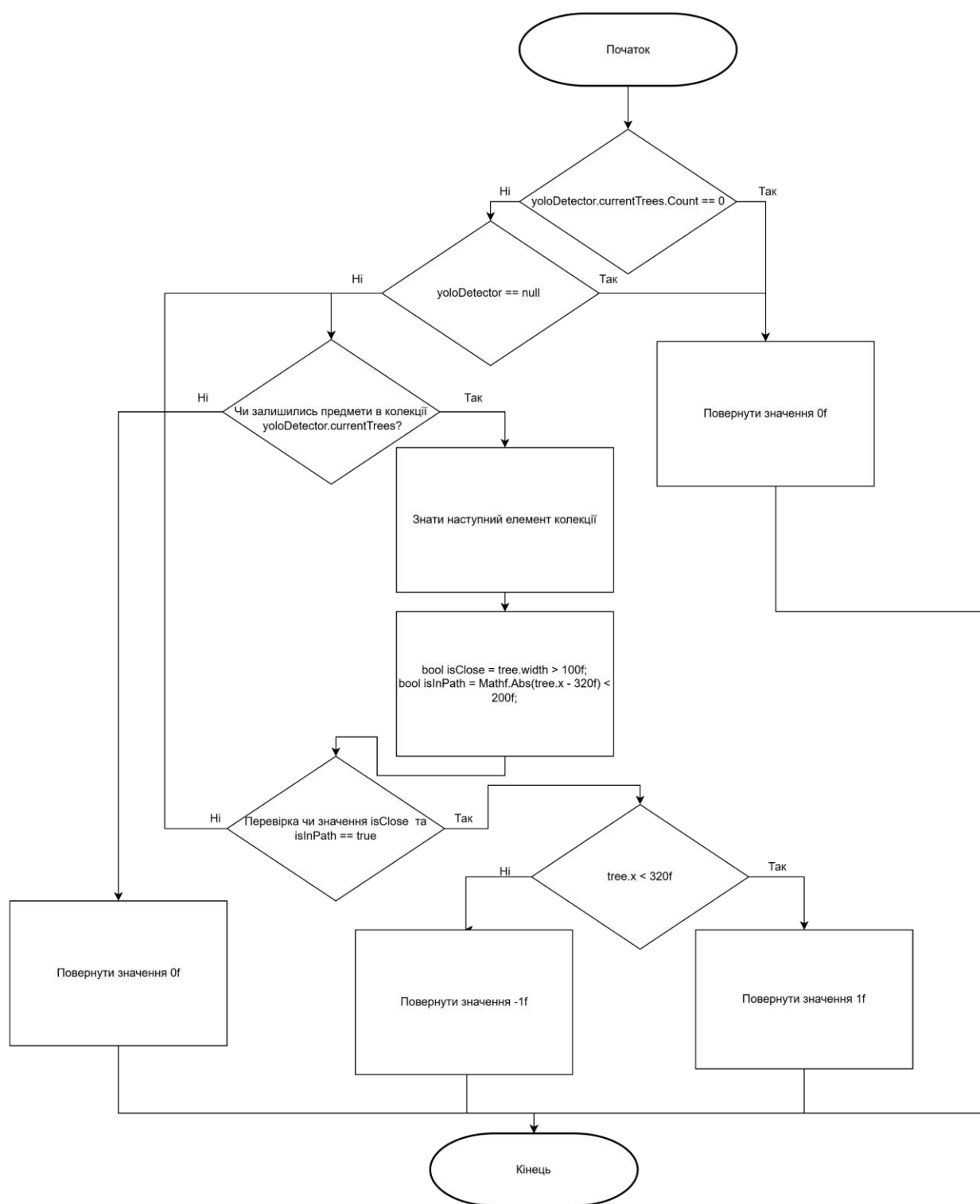
**Засіб розробки систем групової навігації літальних апаратів із застосуванням
розпізнавання образів**

Алгоритм визначення напрямку ухилення від перешкоди

ІАЛЦ.045490.007 ДЗ

Аркушів 1

Київ - 2026



					ІАЛЦ. 045490.007 ДЗ		
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив	Бурма М. С.				Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів Алгоритм визначення напрямку ухилення від перешкоди	Літ.	Аркуш
Перевірів	Сергієнко П.А.						
Консульт.						1	1
Н. контроль	Клятенко Я.М.					КПІ ім. Ігоря Сікорського, ФПСІМ КВ-21	
Зав. каф.	Романкевич В.О.						

ДОДАТОК 4

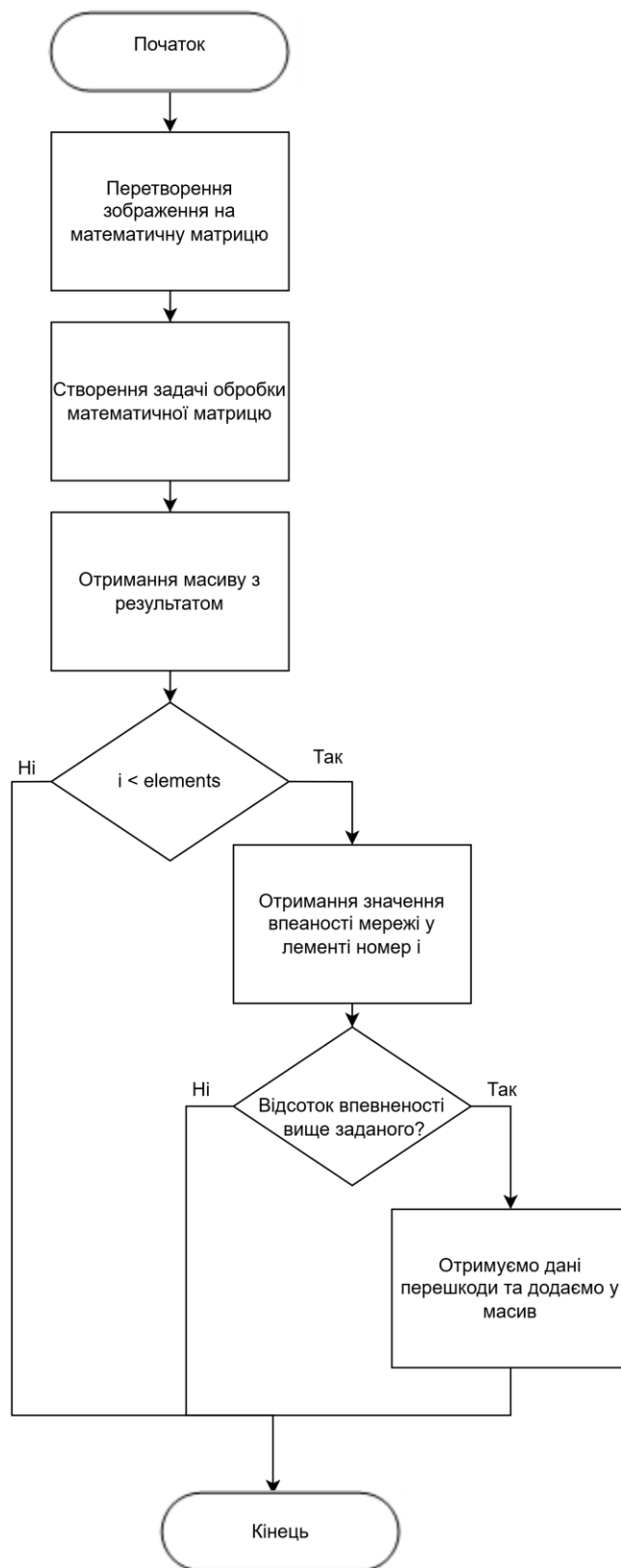
Засіб розробки систем групової навігації літальних апаратів із застосуванням
розпізнавання образів

Алгоритм виявлення перешкод

ІАЛЦ.045490.008 Д4

Аркушів 1

Київ - 2026



					ІАЛЦ. 045490.008 Д4		
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив	Бурма М. С.				Засіб розробки систем групової навігації літальних апаратів із застосуванням розпізнавання образів Алгоритм виявлення перешкод		
Перевірів	Сергієнко П.А.						
Консульт.							
Н. контроль	Клятченко Я.М.						
Зав. каф.	Романкевич В.О.						
					Літ.	Аркуш	Аркушів
						1	1
					КПІ ім. Ігоря Сікорського, ФПСІМ КВ-21		

ДОДАТОК 5

Засіб розробки систем групової навігації літальних апаратів із застосуванням
розпізнавання образів

Лістинг скрипту фізики та керування дрона

ІАЛЦ.045490.009 Д5

Аркушів 1

Київ - 2026

drone.cs

```
using UnityEngine;

public class DronePhysics : MonoBehaviour
{
    public Rigidbody rb;

    [Header("Автопілот")]
    public bool isAutopilot = false;

    [Header("Пух")]
    public float liftForce = 20f;
    public float torqueForce = 5f;
    public float autoLevelForce = 10f;

    [HideInInspector] public float throttle;
    [HideInInspector] public float pitch;
    [HideInInspector] public float roll;
    [HideInInspector] public float yaw;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void Update()
    {
        // Перемикання режиму клавішею 'M'
        if (Input.GetKeyDown(KeyCode.M))
        {
            isAutopilot = !isAutopilot;
            Debug.Log("Автопілот: " + isAutopilot);
        }
        if (!isAutopilot)
        {
            throttle = 0;
            if (Input.GetKey(KeyCode.Space)) throttle = 1;
            if (Input.GetKey(KeyCode.LeftControl)) throttle = -0.5f;

            pitch = Input.GetAxis("Vertical");
            roll = Input.GetAxis("Horizontal");

            yaw = 0;
            if (Input.GetKey(KeyCode.Q)) yaw = -1;
            if (Input.GetKey(KeyCode.E)) yaw = 1;
        }
    }

    public void SetAutopilotInputs(float t, float p, float r, float y)
    {
        if (isAutopilot)
        {
            throttle = t;
            pitch = p;
            roll = r;
            yaw = y;
        }
    }

    void FixedUpdate()
    {

```

```
rb.AddForce(transform.up * liftForce * throttle);
rb.AddTorque(transform.right * pitch * torqueForce);
rb.AddTorque(-transform.forward * roll * torqueForce);
rb.AddTorque(transform.up * yaw * torqueForce);
```

```
if (Mathf.Abs(pitch) < 0.1f && Mathf.Abs(roll) < 0.1f)
{
    Vector3 currentUp = transform.up;
    Vector3 targetUp = Vector3.up;
    Vector3 levelingTorque = Vector3.Cross(currentUp, targetUp) * autoLevelForce;
    rb.AddTorque(levelingTorque);
}
```

```
if (windManager.Instance != null)
{
    rb.AddForce(windManager.Instance.GetWindForce());
}
```

```
}
```

ДОДАТОК 6

Засіб розробки систем групової навігації літальних апаратів із застосуванням
розпізнавання образів

Презентація

ІАЛЦ.045490.010 Д6

Аркушів 1

Київ - 2026

ЗАСІБ РОЗРОБКИ СИСТЕМ ГРУПОВОЇ НАВІГАЦІЇ ЛІТАЛЬНИХ АПАРАТІВ ІЗ ЗАСТОСУВАННЯМ РОЗПІЗНАВАННЯ ОБРАЗІВ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Факультет програмних систем та прикладної математики

Виконав: Бурма Максим Сергійович КВ-21

Керівник: Сергієнко П.А.

- ▶ Швидкий розвиток БПЛА
- ▶ Відсутність повноцінних рішень проблеми

На сьогоднішній день сфера використання БПЛА переживає швидкий розвиток. За останні час їх почали значно частіше використовувати як у цивільних так і військових сферах.

АКТУАЛЬНІСТЬ ТЕМИ



- ▶ Метою даного проекту є створення засобу тестування групової навігації літальних апаратів із застосуванням розпізнавання образів
- ▶ Завдання – щоб досягти цієї мети потрібно спочатку розробити базовий симулятор керування FPV дроном, потім створити комп'ютерний зір для нього та після цього надати створеній програмі можливість працювати з комп'ютерними системами.

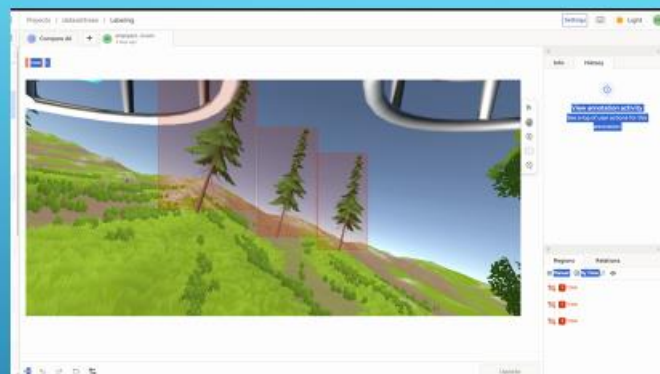
МЕТА СТВОРЕНОГО ПРОЄКТУ

- ▶ Liftoff
- ▶ Uncrashed
- ▶ UFDS



АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

- Для навчання нейронної мережі YOLO, було створено датасет у програмі Label Studio



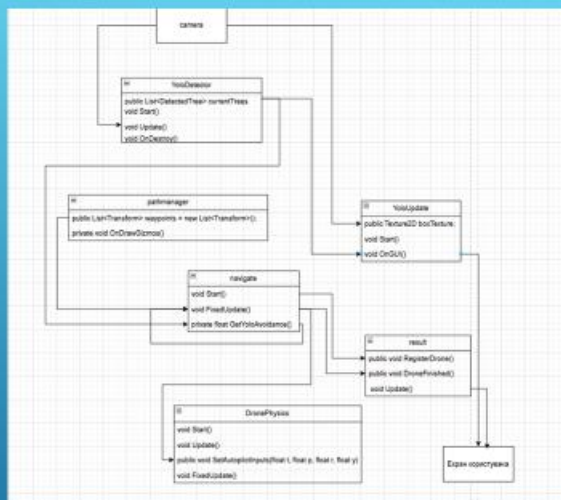
СТВОРЕННЯ ДАТАСЕТУ ДЛЯ НЕЙРОННОЇ МЕРЕЖІ

- Для навчання була обрана модель YOLOv8 Nano.
- Метод навчання трансферний.
- Кількість епох в одному колі навчання – 50.
- Заголом було виконано 3 кола навчання

НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

- Для використання моделі безпосередньо у unity, була використана бібліотека Unity Sentis, а модель мережі конвертована у формат onnx що підтримується Unity.

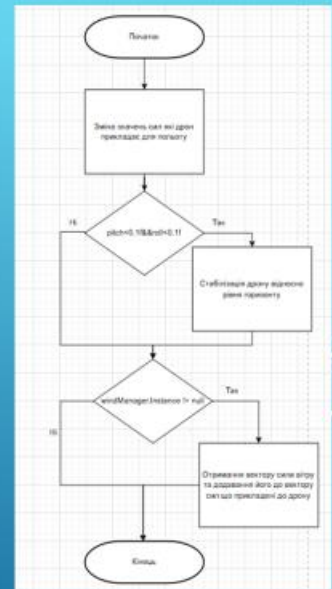
ІНТЕГРАЦІЯ НЕЙРОННОЇ МЕРЕЖІ ДО UNITY



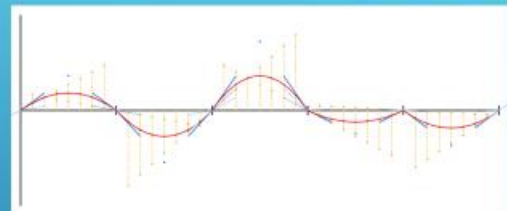
АРХІТЕКТУРА СИМУЛЯТОРА

- ▶ Для надання реалізму створеному симулятору були симульовані такі аспекти вітру.
- ▶ Інерція – створена за допомогою rigid body, та використання поступового впливу на вектор руху в якості керування
- ▶ Сила тяжіння – релізована за допомогою rigid body
- ▶ Вплив вітру – був реалізований за допомогою стороннього впливу на вектор руху дрона
- ▶ Базова обробка фізики зіткнень – була реалізована за допомогою надання дрону сітки, за якої обраховуються колізії

ФІЗИКА ДРОНА



- ▶ Для симуляції мінливого вітру був використаний шум перліна
- ▶ Отримане з ряду перліна число множиться на задане кожного кадру, а отримане значення додається до загальної сили вітру, в результаті чого і отримується загальне значення сили вітру.
- ▶ Потім воно за допомогою векторної суми додається до всіх сил що діють на дрон



РЕАЛІЗАЦІЯ ВІТРУ

- ▶ Для задання маршруту використовується система невидимих точок за якими намагається пролетіти дрон
- ▶ Сама навігаційна система знаходиться в окремому скриптовому файлі, де її можна вільно замінити.

НАВІГАЦІЯ

- ▶ ПЗ забезпечення при тестування трьох дронів займає близько 3500 мб оперативної пам'яті, витрачаючи приблизно по 50-100 мб додатково на кожного доданого дрону зверху.
- ▶ При тестування запуску 10 дронів одночасно програма працювала стабільно.
- ▶ Навантаження на відеокарту склало 3600-3900 мб

НАВАНТАЖЕННЯ НА СИСТЕМУ

Назва симулятора	Фізика	Графіка	Імітація роботи радіозв'язу	Вбудована можливість тестування автопілота
Liftoff	Реалістична, але є проблеми з зіткненнями	середня	-	-
Uncrashed	реалістична	Фото-реалізм	Базовий рівень	-
UFDS	реалістична	реалістична	Детальна, з імітацією РЕБ	-
Створене ПЗ	Реалістична, але без детальної обробки зіткнень	Низько-середня	-	Наявна, з можливістю тестування групової навігації

ПОРІВНЯННЯ РЕЗУЛЬТАТІВ

Було створено ПЗ для тестування навігаційних систем з вбудованою в нього системою комп'ютерного зору на базі нейронної мережі YOLOv8.

У ньому було реалізовано:

- ▶ Наближену до реальної, фізику
- ▶ Базову графіку
- ▶ Як можливість ручного керування так і навігаційною системою
- ▶ Також було створено систему комп'ютерного зору
- ▶ Для комп'ютерного зору було створено два способи відображення
- ▶ Можливість тестування цілої групи дронів, одночасно

ВИСНОВКИ

ДЯКУЮ ЗА УВАГУ