УДК 004.418:623.746-519
DOI https://doi.org/10.32782/2663-5941/2026.1.2/20

Костюк І.В.
https://orcid.org/0009-0001-5150-7464
Запорізький національний університет

Кривохата А.Г.
https://orcid.org/0000-0001-9664-0659
Запорізький національний університет

РОЗРОБКА КРОСПЛАТФОРМНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ СИМУЛЯЦІЇ КЕРУВАННЯ ВІДДАЛЕНИМИ ПРИСТРОЯМИ

Статтю присвячено розробці доступного високоточного симулятора для підготовки операторів БПЛА у режимі «від першої особи» (FPV). Дослідження спрямоване на подолання бар'єрів входу в галузь, як-от: висока вартість обладнання, ризики пошкодження техніки під час навчання, надмірні вимоги наявного програмного забезпечення до обчислювальних ресурсів.

Метою роботи є проєктування та програмна реалізація кросплатформної системи на базі рушія Unity, яка забезпечує реалістичне моделювання фізики польоту в «акро-режимі» та характеризується високою мірою інваріантності щодо апаратних платформ, включаючи Windows, macOS, Linux та Android. Дослідження базується на побудові багаторівневої ієрархічної архітектури, що дозволяє відокремити збір та абстракцію вхідних даних від високорівневої логіки польоту та низькорівневої фізичної взаємодії. Використання спеціалізованого інтерфейсу DroneInputs забезпечує стандартизацію сигналів від різних типів контролерів, що гарантує універсальність застосунку.

Наукова новизна роботи полягає в оптимізації алгоритмів диференціального змішування тяги двигунів у межах компонента Drone_Controller. Запропонована модель здійснює точну трансляцію векторів керування (Pitch, Roll, Yaw, Throttle) у фізичні сили та крутні моменти. Поєднання можливостей фізичного рушія NVIDIA PhysX із кастомними скриптами обробки інерції та опору повітря дозволило досягти високого ступеня кореляції між поведінкою віртуального об'єкта та динамікою реальних гоночних квадрокоптерів.

Окрему увагу приділено технічним аспектам кросплатформної оптимізації. Обґрунтовано впровадження графічного конвеєра Universal Render Pipeline для забезпечення стабільної продуктивності (60 FPS на ПК та 30 FPS на Android) на пристроях з обмеженими ресурсами. Описано застосування бекенда IL2CPP та методу стиснення текстур ASTC, що є критично важливим для мобільної розробки. Практична цінність дослідження полягає у створенні відкритої модульної платформи для тренування пілотів та проведення досліджень у сфері автономної навігації. У висновках окреслено перспективи розвитку системи, зокрема впровадження динамічної телеметрії та інтеграції з VR-технологіями для підвищення зануреності у навчання.

Ключові слова: *комп'ютерні інформаційні технології, Unity, кросплатформність, програмування, інформаційна система, архітектура програмного забезпечення, операційна система.*

Постановка проблеми. Динамічна інтеграція безпілотних літальних апаратів (БПЛА) у цивільний та спеціалізований сектори зумовлює зростання вимог до кваліфікації операторів, зокрема пілотів систем із видом від першої особи (First Person View, FPV). Процес підготовки фахівців у реальних умовах супроводжується значними фінансовими витратами та високими ризиками пошкодження обладнання, що робить використання програмних симуляторів критично необхідним етапом навчання.

Проте аналіз сучасного ринку FPV-симуляторів виявив низку чинників, що обмежують їхнє масове впровадження. Більшість професійних рішень характеризуються високою ресурсоємністю, вимагають спеціалізованого апаратного забезпечення та мають високу вартість. Це створює суттєві бар'єри для початківців та освітніх

© Костюк І.В., Кривохата А.Г., 2026
Стаття поширюється на умовах ліцензії відкритого доступу CC BY 4.0

установ, які не володіють потужними графічними станціями.

Таким чином, виникає актуальне науково-практичне завдання – розробка кросплатформної інформаційної системи симуляції, яка б поєднувала автентичність фізичної моделі польоту з низькими вимогами до обчислювальної потужності пристроїв, зокрема мобільних платформ на базі ОС Android, забезпечуючи доступність навчання пілотуванню для широкого кола користувачів.

Аналіз останніх досліджень і публікацій. Ефективність використання симуляційних платформ як фундаментального інструменту підготовки операторів БПЛА доведена в численних дослідженнях [1, 2]. Проте аналіз існуючих рішень у сегменті FPV-симуляції свідчить про наявність суттєвих технологічних та економічних бар'єрів, що обмежують їхнє масове впровадження. Аналіз наявного на ринку програмного забезпечення дозволив виділити наступні проблеми:

1. Висока ресурсна місткість та апаратна залежність. Популярні симулятори, такі як Liftoff та Velocidrone, орієнтовані на досягнення максимальної фізичної точності та візуальної деталізації [3-6]. Це призводить до високої обчислювальної складності алгоритмів, що вимагає використання спеціалізованих ігрових станцій. Експлуатація таких систем на стандартних ноутбуках або бюджетних комп'ютерах супроводжується низькою частотою кадрів, що нівелює тренувальний ефект через затримку введення. Аналогічні проблеми характерні для DRL Simulator та FPV SkyDive, де акцент на фотореалістичності графіки створює надмірне навантаження на графічний процесор, роблячи їх непридатними для розгортання в межах типових навчальних лабораторій [7-10].

2. Екосистемна замкненість. Окрему категорію представляють рішення від виробників обладнання, зокрема DJI Virtual Flight (SRIZFLY). Попри відносну доступність, такі системи мають штучні обмеження сумісності, підтримуючи лише пропрієтарні контролери виробника [11, 12]. Це унеможливлює використання стандартних радіопередавачів, зокрема Radiomaster, FrSky, які є галузевим стандартом, і змушує користувачів нести додаткові витрати на специфічне апаратне забезпечення.

3. Невідповідність функціональних можливостей. Менш відомі або нішеві симулятори, наприклад, Rotor Rush, часто демонструють або надмірну спеціалізацію на гоночних аспектах, або мають суттєві вади в моделюванні аеродинамічних процесів [13, 14]. Як наслідок, на ринку спостерігається дихотомія: високоточні, але ресурсоємні та дорогі системи протиставляються доступним, але функціонально обмеженим рішенням [15, 16].

Для реалізації симулятора було обрано платформу Unity. Ключовим фактором стала інтеграція з фізичним рушієм NVIDIA PhysX, що забезпечує детерміновану обробку динаміки твердих тіл (тяга, опір середовища, гіроскопічні моменти) за помірних витрат ресурсів центрального процесора. Використання мови C# надає можливість здійснювати прецизійне маніпулювання векторами сил та крутних моментів у реальному часі, що є критичним для ітеративного вдосконалення польотної моделі [17, 18].

У межах дослідження було розглянуто деякі альтернативні рішення:

1. Unreal Engine (Chaos Physics): попри високу візуальну точність, рушій характеризується надмірною ресурсомісткістю та складністю оптимізації для мобільних платформ і бюджетних комп'ютерів [19, 20].

2. Godot Engine: демонструє високу продуктивність на слабких системах, проте на етапі розробки виявлено недостатню зрілість інструментарію для побудови складних 3D-аеродинамічних моделей порівняно з екосистемою Unity [23, 24].

3. Розробка власного рушія (C++/OpenGL): визнана недоцільною через критичне зростання часових витрат на реалізацію низькорівневих систем (рендеринг, обробка вводу), що зміщує фокус із розробки фізики польоту на системне програмування [25, 26].

Для забезпечення стабільної частоти кадрів на пристроях із низькими технічними характеристиками використано конвеєр Universal Render Pipeline. Це надало можливість впровадити гнучку систему керування графічними ресурсами: вибіркове відключення ресурсомістких ефектів (пост-процесинг, складні тіні тощо) без втрати функціональності симуляції [21, 22]. Такий підхід забезпечує уніфікацію коду для розгортання в операційних системах Windows, Linux, macOS та Android, що є ключовою вимогою до проєктованої системи.

Результати критичного аналізу наявних рішень та інструментарію розробки дозволяють сформулювати низку ключових тез, що визначають вектор подальшого дослідження:

1. Виявлено суттєвий технологічний розрив між функціональними можливостями провідних FPV-симуляторів (Liftoff, Velocidrone тощо) та їхньою доступністю для широкого кола користу-

вачів. Висока обчислювальна складність алгоритмів і жорсткі вимоги до апаратного забезпечення створюють критичні бар'єри для впровадження таких систем у межах освітніх програм та на мобільних платформах.

2. Обґрунтовано необхідність розробки альтернативного рішення, що базується на компромісі між автентичністю фізичної моделі польоту та ресурсоефективністю. Пріоритетними критеріями проєктування визначено:

– обчислювальну ефективність: гнучку оптимізацію під неспеціалізовані ПК та пристрої на базі ОС Android;

– апаратну універсальність: підтримку стандартних протоколів підключення широкого спектра контролерів;

– фізичну адекватність: забезпечення реалізму, достатнього для формування базових навігаційних навичок без надмірного ускладнення математичних розрахунків.

3. Визначено та обґрунтовано оптимальний технологічний стек, що базується на мультикритеріальному аналізі середовищ розробки. Обраний рушій Unity у поєднанні з фізичною бібліотекою NVIDIA PhysX та конвеєром рендерингу URP забезпечує детермінованість фізичних процесів за умов обмежених обчислювальних ресурсів. Таке поєднання дозволяє реалізувати кросплатформну архітектуру з високим ступенем контролю над графічною продуктивністю.

Таким чином, розробка доступної інформаційної системи симуляції є актуальним науково-практичним завданням. Реалізація запропонованого підходу дозволить суттєво мінімізувати технічний та фінансовий поріг входження у процес навчання пілотуванню БПЛА, роблячи його доступним для користувача.

Постановка завдання. Метою статті є розробка та реалізація доступного кросплатформного симулятора FPV-польотів, що поєднує низькі системні вимоги з адекватним відтворенням фізичної моделі польоту для забезпечення масової базової підготовки пілотів у неспеціалізованих навчальних середовищах.

Виклад основного матеріалу. Основним етапом реалізації є проєктування архітектури інформаційної системи, що дозволяє формалізувати концептуальні вимоги у вигляді технічних специфікацій.

На основі аналізу завдань підготовки пілотів FPV-дронів сформовано перелік ключових функціональних вимог, які згруповано за основними модулями системи:

1. Фізико-математична модель польоту.

– Реалізація динаміки твердого тіла: обчислення траєкторії польоту з урахуванням векторів тяги, сили тяжіння, інерції та аеродинамічного опору.

– Підтримка режиму «Acro»: забезпечення прямого керування кутовими швидкостями (тангаж, крен, рискання) без автоматичної стабілізації горизонту, що є критичним для професійної підготовки.

– Динамічна обробка зіткнень: імітація фізичної взаємодії з об'єктами середовища та поверхнею землі.

2. Система керування та інтерфейсу.

– Універсальність введення: підтримка стандартних ігрових контролерів (протоколи XInput/DirectInput), клавіатури, а також віртуальних джойстиків для сенсорних екранів.

– Адаптивна модель тяги: впровадження накопичувального методу введення газу, що відтворює механіку роботи реальних радіопередавачів.

– FPV-візуалізація: рендеринг зображення з камери, розташованої на віртуальному дроні, для відтворення специфіки польоту від першої особи.

3. Навчально-тренувальний функціонал.

– Модуль маршрутизації: система послідовних контрольних точок для відпрацювання точності маневрування з візуальною індикацією активного етапу.

– Інструментарій відновлення: функція миттєвого скидання стану для мінімізації пауз у навчальному процесі після падіння.

– Тривимірне середовище: наявність спеціалізованої локації, оптимізованої для тренувальних польотів.

4. Системні вимоги та кросплатформність. Забезпечення повної ідентичності фізичної моделі та функціоналу для ОС Windows, macOS, Linux та Android.

Нефункціональні вимоги до проєктованої системи визначають параметри якості, що забезпечують стабільність симуляції та ефективність навчального процесу. Ключові вимоги класифіковано за наступними напрямами:

1. Продуктивність та обчислювальна ефективність.

– Частота кадрів (FPS): забезпечення стабільного показника не менше 60 FPS для персональних комп'ютерів та 30 FPS для мобільних пристроїв середнього сегмента, що є критичним для запобігання ефекту «закачування».

– Мінімізація латентності: зведення затримки введення до мінімальних значень для забезпе-

чення миттєвого відгуку віртуальної моделі на дії пілота.

– Швидкість ініціалізації: час завантаження ігрових сцен та фізичних об'єктів не повинен перевищувати 10 секунд.

2. Надійність та стабільність.

– Детермінованість фізичних розрахунків: забезпечення стабільної та передбачуваної поведінки об'єктів за будь-яких експлуатаційних сценаріїв у межах інтегрованого фізичного рушія.

– Відмовостійкість: відсутність критичних помилок під час штатної роботи та коректна обробка виняткових ситуацій.

3. Масштабованість та сумісність.

– Архітектурна гнучкість: модульна структура коду для легкої інтеграції нових моделей БПЛА, локацій та тренувальних маршрутів без зміни ядра системи.

– Апаратна сумісність: підтримка архітектур ARMv7 та ARM64 для ОС Android, а також забезпечення коректної роботи на різних типах ЦП та ГП у межах кросплатформного розгортання.

4. Ергономічність та візуальна адекватність.

– Адаптивність інтерфейсу: масштабованість елементів керування для різних роздільних здатностей екранів та мінімізація перекриття зони огляду на мобільних пристроях.

– Якість візуалізації: чітке поле зору камери та високий контраст навігаційних об'єктів (контрольних точок) відносно оточення для забезпечення просторової орієнтації.

Для формалізації функціональної структури та внутрішньої архітектури симулятора використано методологію UML-моделювання. Діаграма варіантів використання (рис. 1) відображає архітектурну логіку взаємодії користувача з інформаційною системою. Основним актором є пілот, чия діяльність структурована за трьома функціональними векторами:

1. Модуль пілотування: реалізує пряме маніпулювання фізичною моделлю через канали тяги, тангажу, крену та рискання. Зв'язки типу «extend» вказують на необов'язковість одночасного використання всіх осей керування, що дозволяє гнучко адаптувати введення залежно від типу маневру.

2. Навчально-тренувальна логіка: варіант використання «Пролетіти маршрут» обов'язково містить процес послідовної активації контрольних точок. Успішне проходження дистанції розширюється подією завершення маршруту та відповідною індикацією в інтерфейсі.

3. Моніторинг та відновлення стану: передбачає можливість реініціалізації просторового положення дрона («Перевернути дрон») у разі критичного зіткнення, що забезпечує безперервність тренувального процесу.

Програмна реалізація симулятора базується на компонентно-орієнтованому підході з використанням принципів об'єктно-орієнтованого проєктування. Внутрішню структуру системи відображено на діаграмі класів (рис. 2), де виділено три ключові функціональні модулі: ядро фізичної симуляції, систему обробки вхідних даних та модуль логіки тренувального процесу.

1. Модуль фізичної симуляції та керування польотом.

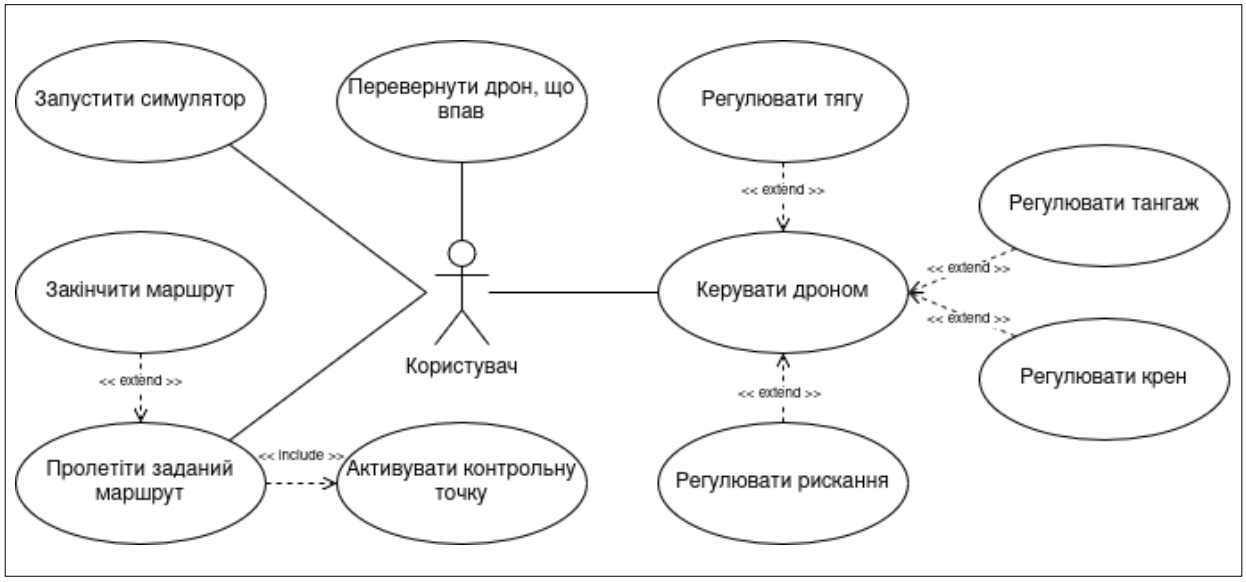

Рис. 1. Діаграма варіантів використання

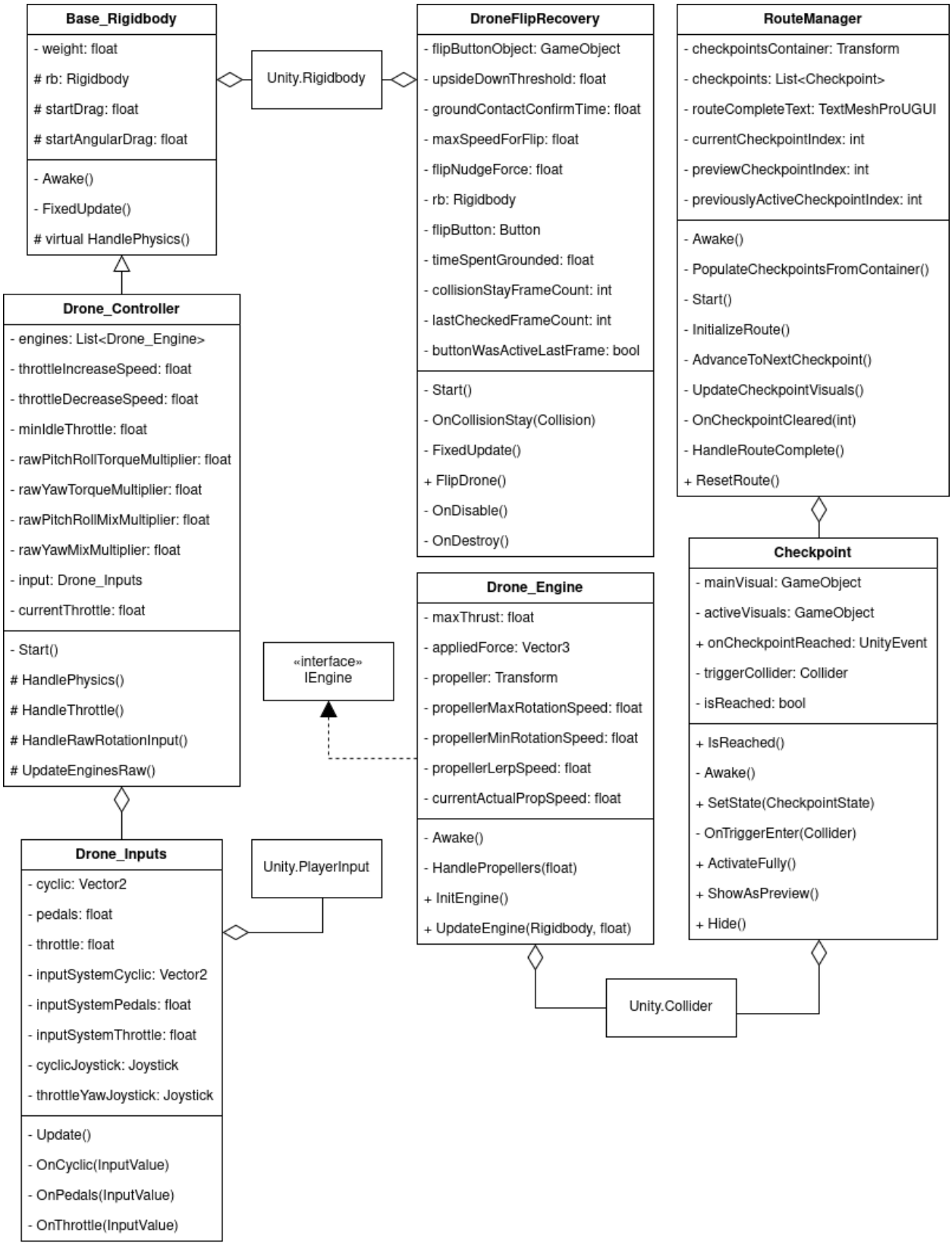

Рис. 2. Діаграма класів

Основою архітектури є ієрархія класів, що відповідають за динаміку твердого тіла:

– Base_Rigidbody: абстрактний базовий клас, що інкапсулює посилання на фізичний компонент Unity та визначає віртуальний метод HandlePhysics(). Це забезпечує детермінованість оновлення фізичного стану в циклі FixedUpdate().

– Drone_Controller: центральний керуючий компонент, що наслідує Base_Rigidbody. Він виконує роль оркестратора: обчислює сумарний вектор тяги та крутні моменти на основі даних від модуля вводу та розподіляє їх між двигунами.

– IEngine та Drone_Engine: реалізація інтерфейсного підходу дозволяє уніфікувати взаємодію з силовими установками. Клас Drone_Engine виконує безпосереднє прикладання сили до об'єкта за допомогою методу AddForceAtPosition() та візуалізує динаміку обертання пропелерів.

2. Модуль абстракції вхідних даних.

Для забезпечення кросплатформності реалізовано клас Drone_Inputs, який виконує роль прошарку між фізичними пристроями та контролером дрона. Він агрегує дані з двох джерел:

– Системи подій Unity PlayerInput для апаратних контролерів та клавіатури.

– Віртуальних екранних джойстиків Joystick для мобільних платформ. Така архітектура надає можливість відокремити фізику польоту від конкретної реалізації апаратного вводу, забезпечуючи високу гнучкість системи.

3. Логіка навчального процесу та станів. Навчальний функціонал реалізовано через систему взаємодії менеджерів та тригерів:

– RouteManager та Checkpoint: реалізують патерн «Спостерігач». RouteManager керує послідовністю проходження маршруту, тоді як об'єкти Checkpoint стежать за проходженням дрона через активну зону та через систему подій UnityEvent сповіщають менеджер про зміну стану.

– DroneFlipRecovery: автономний модуль моніторингу просторової орієнтації. Він аналізує вектори нормалі об'єкта відносно поверхні та ініціює алгоритм відновлення положення («Flip») у разі критичного перекидання.

Ключовими аспектами такого архітектурного рішення є:

1. Модульність. Чітке розділення відповідальності між класами полегшує тестування та подальше розширення системи (наприклад, додавання нових моделей БПЛА).

2. Ефективність. Використання інтерфейсів та наслідування надає можливість оптимізації викликів у фізичному циклі Unity, що критично для забезпечення стабільного FPS на мобільних пристроях.

3. Масштабованість. Архітектура дозволяє інтегрувати нові типи маршрутів та складніші аеродинамічні моделі без рефакторингу ядра системи.

В основі розробленого симулятора лежить концептуальна модель динаміки твердого тіла, інтегрована в середовище Unity (фізичний рушій NVIDIA PhysX). Модель орієнтована на відтворення режиму польоту «Acro», що вимагає детермінованого розрахунку сил та моментів у реальному часі. Технічні аспекти впровадження фізичної моделі та алгоритмів керування в середовищі Unity:

1. Компоненти фізичних сил та моментів. Рух віртуального об'єкта визначається суперпозицією наступних векторів: гравітаційна складова, сила тяги, аеродинамічний опір (реалізований через апроксимацію лінійного (drag) та кутового (angularDrag) опору).

2. Модель керування в режимі «Acro». На відміну від систем з автостабілізацією, реалізована модель передбачає пряме керування кутовими швидкостями. Вхідні дані користувача інтерпретуються як цільова швидкість обертання навколо локальних осей (тангаж, крен, рискання). Для досягнення стабільного відгуку система розраховує коригувальний крутний момент на основі різниці між поточною та цільовою кутовою швидкістю, застосовуючи його через метод AddRelativeTorque().

3. Алгоритм мікшування двигунів. Для трансформації команд керування у конкретні значення потужності окремих двигунів використано систему арифметичного мікшування.

4. Реалізація обчислювального циклу. З метою забезпечення високої точності та передбачуваності фізичної поведінки, всі розрахунки динаміки та детектування зіткнень інкапсульовано в циклі FixedUpdate(). Це дозволяє нівелювати вплив варіативності частоти кадрів (FPS) на точність інтегрування рівнянь руху. Спрощена аеродинамічна модель (без урахування ефектів екранної подушки та вихрових кілець) дозволяє мінімізувати обчислювальні витрати, зберігаючи при цьому автентичність досвіду пілотування на мобільних пристроях.

Запропонована система керування безпілотним літальним апаратом базується на принципах багаторівневої ієрархічної архітектури, що забезпечує декомпозицію функціональних завдань на рівні збору даних, їхньої аналітичної обробки, високо-

рівневого моделювання польоту та низькорівневої взаємодії з виконавчими механізмами.

Структурна організація системи включає такі ключові компоненти:

1. Модуль абстракції вхідних даних. Використання підсистеми *Unity Input System* дозволяє реалізувати гнучкий механізм отримання вхідних сигналів незалежно від апаратного забезпечення. Спеціалізований інтерфейс DroneInputs здійснює нормалізацію та стандартування отриманих значень, трансформуючи сирі дані у вектори керування для подальшої передачі в основний контролер.

2. Високорівневий контролер польоту (Drone_Controller). Даний модуль є ядром системи, де реалізовано математичну модель динаміки польоту. Він відповідає за обчислення параметрів загальної тяги, крутних моментів для стабілізації кутового положення та реалізацію алгоритмів диференціального змішування потужності двигунів.

3. Виконавчий рівень (Drone_Engine). Низькорівнева реалізація представлена окремими об'єктами двигунів, які здійснюють безпосереднє прикладання результуючих векторів сил до фізичної моделі об'єкта. Крім фізичної взаємодії, цей рівень забезпечує візуальну анімацію роторів відповідно до поточної кутової швидкості обертання.

Модульний підхід до проєктування системи забезпечує високу точність керування в нелінійному режимі пілотування («акро-режим»). Така конфігурація дозволяє досягти інваріантності системи до методів введення даних та створює умови для прецизійної ітераційної оптимізації аеродинамічних характеристик моделі без втручання в архітектуру збору даних.

Процес підготовки програмного забезпечення для цільових платформ Windows, macOS, Linux, Android базується на уніфікованому конвеєрі збірки середовища Unity з урахуванням специфічних апаратних архітектур та вимог до продуктивності, як-от:

1. Десктопні платформи (Windows, macOS, Linux).

Для настільних систем пріоритетом є забезпечення високої графічної точності при стабільному фреймрейті.

– Графічні інтерфейси. Впроваджено автоматичне перемикання між API (DirectX 11/12 для Windows, Metal для macOS та Vulkan/OpenGL Core для Linux).

– Архітектурні особливості. Використано 64-бітну архітектуру (x86_64), а для систем macOS реалізовано підтримку Universal Binaries (Intel та Apple Silicon).

– Скриптовий бекенд. Застосовано комбінований підхід: Mono для етапів тестування та IL2CPP для фінальних релізних збірок з метою підвищення продуктивності коду.

– Безпека. Для macOS та Windows передбачено процедури цифрового підпису та нотаріального засвідчення, що мінімізує конфлікти з підсистемами безпеки ОС.

2. Мобільна платформа Android.

Через високу фрагментацію апаратного забезпечення реалізація для Android вимагає прецизійного налаштування:

– Обчислювальна архітектура. Обов'язковим є використання IL2CPP із підтримкою паралельних архітектур ARMv7 та ARM64 (відповідно до вимог Google Play).

– Графічна оптимізація. Пріоритет надається API Vulkan з регресією до OpenGLES3. Використано формат стиснення текстур ASTC, що забезпечує оптимальний баланс між візуальною якістю та обсягом відеопам'яті.

– Керування ресурсами. Впроваджено формат Android App Bundle (AAB), що дозволяє динамічно генерувати оптимізовані APK-пакети під конкретні конфігурації пристроїв.

– Інтерфейс та ввід. Адаптація UI реалізована через динамічне масштабування, а обробка вхідних сигналів здійснюється через віртуальні органи керування, інтегровані в систему DroneInputs.

3. Верифікація та дистрибуція.

Завершальним етапом є профілювання на цільових пристроях для запобігання термальному тротлінгу (особливо для мобільних систем) та коректності відгуку фізичної моделі. Результат успішної імплементації інтерфейсу в середовищі Linux представлено на рисунку 3.

Висновки. У результаті проведеного дослідження було розроблено та реалізовано кросплатформну інформаційну систему симуляції керування FPV-дронами, що вирішує науково-практичну задачу зниження бар'єрів доступності професійного навчання пілотів. На основі отриманих результатів зроблено такі висновки:

1. Обґрунтовано архітектурний підхід: доведено ефективність використання модульної ієрархічної структури на базі рушія Unity та фізичного ядра NVIDIA PhysX. Застосування графічного конвеєра Universal Render Pipeline (URP) надало можливість досягти цільових показників продуктивності (60 FPS на десктопних та 30 FPS на мобільних платформах) без втрати реалістичності фізичної моделі.

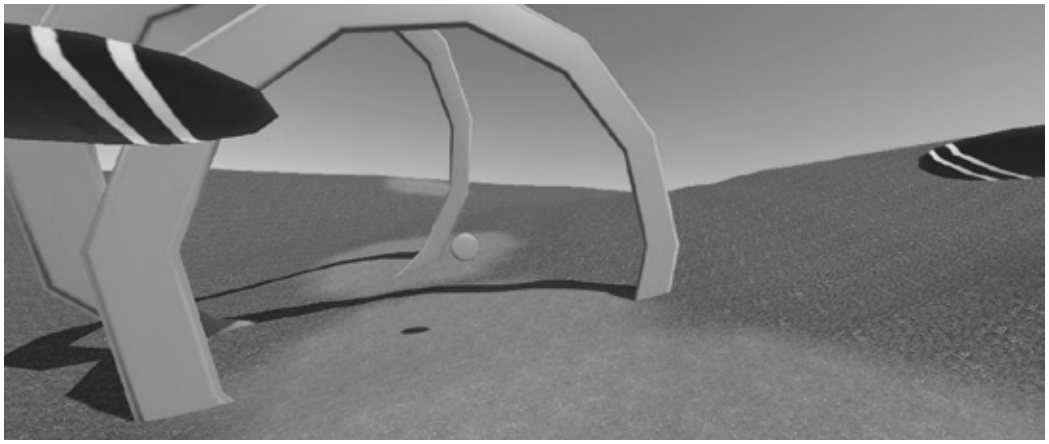

Рис. 3. Інтерфейс функціонуючого програмного засобу

2. Реалізовано фізичну модель «акро-режиму»: розроблено алгоритми диференціального змішування тяги двигунів, що надають можливість здійснювати точне керування кутовими швидкостями. Це забезпечує досвід пілотування, ідентичний роботі з реальними польотними контролерами (Betaflight тощо).

3. Завдяки впровадженню абстрактного рівня обробки вхідних даних (Drone_Inputs), система демонструє стабільну роботу з широким спектром пристроїв введення (геймпади, клавіатури, віртуальні джойстики) на операційних системах Windows, macOS, Linux та Android.

4. Розроблене програмне забезпечення є альтернативою комерційним симуляторам, що дозволяє використовувати його в освітніх цілях та тренувань у мобільних умовах.

Подальший розвиток системи передбачає інтеграцію складних аеродинамічних ефектів і режимів стабілізації, розширення аналітичного інструментарію через модулі телеметрії та оцінки помилок пілотування. Технологічна модернізація охоплюватиме оптимізацію взаємодії з професійними радіоапаратурами та VR-гарнітурами, а також візуальне вдосконалення локацій із можливістю точного налаштування фізичних конфігурацій моделей дронів.

Список літератури:

1. Somerville A., Lynar T., Joiner K., Wild G. Use of Simulation for Pre-Training of Drone Pilots. *Drones*. Vol. 8(11), 640. P. 1–14. URL: https://doi.org/10.3390/drones8110640.

2. Savaş T., Özdemir U., Esen Y. E. The Role of Flight Simulators in Pilot Training. *Journal of Aviation*. 2025. 9(1). P. 137–145. URL: https://doi.org/10.30518/jav.1588557

3. Top FPV Drone Simulators for Mastering Your Skills. *SRIZFLY*. URL: https://srizfly.com/top-fpv-drone-simulators-for-mastering-your-skills/.

4. Liftoff Simulator Review (Is This the Best FPV Simulator?). *FPVCraft*. URL : https://fpvcraft.com/liftoff-simulator-review/.

5. Features. *VelociDrone*. URL: https://www.velocidrone.com/features

6. VelociDrone FPV Racing Simulator. *Team Blacksheep*. URL: https://www.team-blacksheep.com/simulator

7. The Drone Racing League Simulator. *EpicGames Store*. URL: https://store.epicgames.com/en-US/p/the-drone-racing-league-simulator

8. PyFlyt – UAV Simulation Environments for Reinforcement Learning Research / J.J. Tai and etc. *Journal of Latex Class Files*. 2021. Vol. 14, no. 8. P. 1–10. URL: https://doi.org/10.48550/arXiv.2304.01305

9. FPV SkyDive : FPV Drone Simulator. *Steam*. URL: https://store.steampowered.com/app/1278060/FPV_SkyDive__FPV_Drone_Simulator/

10. Nikolaiev M., Novotarskyi M. Comparative Review of Drone Simulators. *Information, Computing and Intelligent Systems*. 2024. No. 4. P. 79–99. URL: https://doi.org/10.20535/2786-8729.4.2024.300614

11. DJI Flight Simulator Compatible Controllers Guide. *SRIZFLY*. URL: https://srizfly.com/dji-flight-simulator-compatible-controllers-guide/

12. Won J., Hong G. Research on Smart Construction Education Training Contents Using a Drone Simulator. *Journal of Multimedia Information System.* 2022. 9(4). P. 345–354. URL: https://doi.org/10.33851/JMIS.2022.9.4.345

13. Mairaj A., Baba A. I., Javaid A. Y. Application Specific Drone Simulators: Recent Advances and Challenges. *Simulation Modelling Practice and Theory.* 2019. P. 100–117. URL: https://doi.org/10.1016/j.simpat.2019.01.004

14. Survey of Simulators for Aerial Robots: An Overview and In-Depth Systematic Comparisons / C. A. Dimmig and etc. *IEEE Robotics & Automation Magazine.* 2024. P. 2–15. URL: https://doi.org/10.1109/MRA.2024.3433171

15. A Survey on Open-Source Simulation Platforms for Multi-Copter UAV Swarms / Z. Chen and etc. *Robotics*. 2023. Vol. 12(2), 53. P. 1–24. URL: https://doi.org/10.3390/robotics12020053

16. Carretta T. R., Dunlap R. D. Transfer of Training Effectiveness in Flight Simulation: 1986 to 1997. *United States Air Force Research Laboratory*. 1998. URL: https://doi.org/10.21236/ada362818

17. Physics solutions for game development. *Unity*. URL: https://unity.com/solutions/programming-physics

18. Enhanced physics performance for smooth gameplay. *Unity*. URL: https://unity.com/how-to/enhanced-physics-performance-smooth-gameplay

19. Physics in Unreal Engine. *Epic Developer Community*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/chaos-physics-overview

20. Physics introduction. *Godot Docs*. URL: https://docs.godotengine.org/en/stable/tutorials/physics/physics_introduction.html

21. Unity: Understanding URP, HDRP, and Built-In Render Pipeline. *Wayline*. URL: https://www.wayline.io/blog/unity-understanding-urp-hdrp-built-in

22. Introduction to URP for advanced creators (Unity 6 edition). *Unity*. URL: https://unity.com/resources/introduction-to-urp-advanced-creators-unity-6

23. Unity проти Unreal Engine – який рушій обрати для гри та чому. Зважуємо всі за та проти з розробниками. *Gamedev.DOU.UA*. URL: https://gamedev.dou.ua/articles/unity-or-unreal-engine/

24. Unreal vs Unity vs Godot. *Ruah Tech Solutions*. URL: https://ruahtech.com.au/unreal-vs-unity-vs-godot/

25. Воробець Д. С. Ігровий рушій для розробки інді ігор на основі OpenGL : кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник В. І. Заяц. Запоріжжя : ЗНУ, 2023. 93 с. URL: https://dspace.znu.edu.ua/jspui/handle/12345/12502

26. Gamedev.DOU.UA. Власний Game Engine: як і навіщо створювати ігровий рушій. *Gamedev.DOU.UA*. URL: https://gamedev.dou.ua/blogs/how-and-why-to-create-a-game-engine/

Kostiuk I.V., Kryvokhata A.G. DEVELOPMENT OF A CROSS-PLATFORM INFORMATION SYSTEM FOR REMOTE DEVICE CONTROL SIMULATION

The article is devoted to the development of an accessible, high-precision simulator for training UAV operators in First Person View (FPV) mode. This research is specifically oriented toward overcoming significant industry entry barriers such as high equipment costs, the risk of hardware damage during the learning process, and the excessive computational demands placed by existing simulation software on available hardware resources.

The aim of the work is the design and software implementation of a cross-platform system based on the Unity engine, which provides realistic flight physics modeling in «Acro-mode» and is characterized by a high degree of hardware platform invariance, including Windows, macOS, Linux, and Android. The research is based on constructing a multi-level hierarchical architecture that allows for the separation of input data collection and abstraction from high-level flight logic and low-level physical interaction. The use of a specialized DroneInputs interface ensures the standardization of signals from various types of controllers, guaranteeing the application's versatility.

The scientific novelty of the work lies in the optimization of motor mixing algorithms within the Drone_Controller component. The proposed model performs precise translation of control vectors (Pitch, Roll, Yaw, Throttle) into physical forces and torques. Combining the capabilities of the NVIDIA PhysX engine with custom scripts for processing inertia and air resistance achieved a high degree of correlation between the behavior of the virtual object and the dynamics of real racing quadcopters.

Particular attention is paid to the technical aspects of cross-platform optimization. The implementation of the Universal Render Pipeline is justified to ensure stable performance (60 FPS on PC and 30 FPS on Android) on resource-constrained devices. The application of the IL2CPP backend and the ASTC texture compression method is described, both of which are critical for mobile development. The practical value of the study lies in the creation of an open modular platform for pilot training and research in the field of autonomous navigation. The conclusions outline the prospects for system development, including the implementation of dynamic telemetry and integration with VR technologies to enhance training immersion.

Keywords: *computer information technologies, Unity, cross-platform, programming, information system, software architecture, operating system.*

Дата першого надходження статті до видання: 13.01.2026
Дата прийняття статті до друку після рецензування: 19.02.2026
Дата публікації (оприлюднення) статті: 08.04.2026