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
(підпис) (ініціали, прізвище)

“___” червня 2026 р.

Дипломна проєкт

на здобуття ступеня бакалавра

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: Удосконалена система управління камерою із засобами сприяння оператору

Виконав (-ла): студент (-ка) IV курсу, групи КВ-22
(шифр групи)

Пляченко Олександр Олександрович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асистент кафедри СПСКС Сергієнко П.А.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) _ (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2026 року

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»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
_____
(підпис) (ініціали, прізвище)
«___» червня 2026 р.

ЗАВДАННЯ
на дипломний проєкт студента
Пляченко Олександр Олександрович
(прізвище, ім’я, по батькові)

1. Тема проєкту: Удосконалена система управління камерою із засобами сприяння оператору
керівник проєкту асистент кафедри СПСКС Сергієнко П.А.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» травня 2026 р. №1984-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз існуючих рішень на обґрунтування теми дипломного проєкту

2) Матетатичне моделювання та проєктування системи управління

3) Практична реалізація та випробування системи

4) Експериментальне дослідження ефективності алгоритмів супроводу та реакції системи на межові стани

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

1)Схема алгоритму функціонування системи автоматичного супроводу цілі

2)Схема алгоритму ідентифікації та локалізації цілі

3)Структурна схема алгоритму ітерації системи у цикл оновлення Unity

4)Структурна схема гібридного управління

5) Презентація

6. Консультанти розділів проєкту*

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

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури та системний аналіз предметної області	01.03.2026	
2	Розроблення та узгодження технічного завдання	10.03.2026	
3	Математичне моделювання кінематики об'єктива та інерції	20.03.2026	
4	Проєктування алгоритмів контролю меж та запобігання втраті цілі	20.04.2026	
5	Програмна реалізація гібридної системи у середовищі Unity	05.05.2026	
6	Організація тестового середовища та тестування	15.05.2026	
7	Оформлення пояснювальної записки та графічного матеріалу	25.05.2026	
8	Попередній перегляд матеріалів дипломного проєкту	30.05.2026	

Студент _____ Олександр ПЛЯЧЕНКО
(підпис) (ініціали, прізвище)

Керівник проєкту _____ Павло СЕРГІЄНКО
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (64 с., 16 рис., 12 табл., 3 додатки).

Об’єкт розробки – удосконалена система управління віртуальною камерою із засобами сприяння оператору, призначена для оптимізації кінематографічних маневрів та автоматизації супроводу рухомих об’єктів у середовищі Unity.

Метою роботи є створення гібридного програмного комплексу, що інтегрує директивне ручне управління (6DoF) з автономними алгоритмами трекінгу та фізично достовірною віртуальною інерцією. У процесі розробки використано методи векторної алгебри, теорію кватерніонів для уникнення сингулярностей (Gimbal Lock) та функціонал нелінійної інтерполяції для контролю межових станів.

В ході розробки:

− проведено аналіз існуючих архітектур управління та обґрунтовано вибір середовища Unity;

− спроєктовано математичну модель гібридної системи, що поєднує мануальне введення та автоматичну корекцію;

− розроблено алгоритми інерційного згладжування на базі гармонічного осцилятора;

− реалізовано підсистему моніторингу меж екрана для селективного демпфування деструктивних команд оператора;

− проведено валідацію системи шляхом синхронного A/B тестування у віртуальному середовищі з конфігурацією розділеного екрана.

Упровадження розробленої системи дозволяє зменшити когнітивне навантаження на оператора, забезпечити кінематографічну плавність переміщень об’єктива та гарантувати стабільне утримання об'єкта в межах композиційної рамки в умовах високодинамічних сцен.

Ключові слова: СИСТЕМА УПРАВЛІННЯ КАМЕРОЮ, ГІБРИДНА АРХІТЕКТУРА, КІНЕМАТИКА, КВАТЕРНІОНИ, ВІРТУАЛЬНА ІНЕРЦІЯ, UNITY, АВТОМАТИЧНЕ СТЕЖЕННЯ, КОМПОЗИЦІЙНІ ЗОНИ.

ABSTRACT

The qualification work includes an explanatory note (64 pages, 16 figures, 12 tables, 3 appendices).

The object of development is an advanced virtual camera control system with operator assistance tools, designed to optimize cinematographic maneuvers and automate the tracking of moving objects within the Unity environment.

The aim of the work is to create a hybrid software complex that integrates directive manual control (6DoF) with autonomous tracking algorithms and physically accurate virtual inertia. The development process utilized vector algebra methods, quaternion theory to avoid singularities (Gimbal Lock), and non-linear interpolation functionality for boundary state control.

During the development: − an analysis of existing control architectures was conducted, and the selection of the Unity environment was justified; − a mathematical model of the hybrid system combining manual input and automatic correction was designed; − inertial smoothing algorithms based on a harmonic oscillator were developed; − a screen boundary monitoring subsystem was implemented for selective damping of destructive operator commands; − system validation was performed through synchronous A/B testing in a virtual environment using a split-screen configuration.

The implementation of the developed system allows for reducing the operator's cognitive load during high-dynamic scenes, ensuring cinematographic smoothness of lens movements, and guaranteeing stable maintenance of the target object within the compositional frame.

Keywords: CAMERA CONTROL SYSTEM, HYBRID ARCHITECTURE, KINEMATICS, QUATERNIONS, VIRTUAL INERTIA, UNITY, AUTOMATIC TRACKING, COMPOSITIONAL ZONES.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.002 ТЗ	Удосконалена система	4		
			управління камерою			
			із засобами сприяння			
			оператору.			
			Технічне завдання			
	А4	ІАЛЦ.045480.003 ТП	Удосконалена система	2		
			управління камерою			
			із засобами сприяння			
			оператору.			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045480.004 ПЗ	Удосконалена система	64		
			управління камерою			
			із засобами сприяння			
			оператору.			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	*ІАЛЦ.045480.001 ОА*		
Розробив		Пляченко О.О.			Удосконалена система управління камерою із засобами сприяння оператору **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Сергієнко П.А.					
Консульт.						**«КПІ ім. І. Сікорського», ФПСПМ, КВ-22**	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.005 Д1	Алгоритм контролю	1		
			межових станів.			
			Схема алгоритму			
	А4	ІАЛЦ.045480.006 Д2	Алгоритм розрахунку	1		
			вектора автоматичної			
			корекції.			
			Схема алгоритму			
	А4	ІАЛЦ.045480.007 Д3	Алгоритм ітерації	1		
			системи у цикл			
			оновлення Unity			
			Схема структурна			
	А4	ІАЛЦ.045480.008 Д4	Гібридне управління	1		
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Удосконалена система управління камерою із засобами сприяння оператору».

Галузь застосування: створення інструментарію для віртуального кіновиробництва , розробка інтерактивних 3D-симуляторів та відеоігрова індустрія.

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 проєкту є розробка гібридного програмного комплексу в середовищі Unity, що поєднує інтерактивність директивних ручних команд управління віртуальною камерою (6DoF) із надійністю автономних алгоритмів відстеження об'єктів та імітацією фізичної інерції обладнання.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є науково-технічна література, технічна документація платформи Unity, публікації у періодичних виданнях, книги з математики для 3D-графіки, кінематики та комп'ютерного зору.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- забезпечення паралельної обробки прямих команд просторового маніпулювання оператора (pan, tilt, dolly, roll, pedestal, track) та автоматичних імпульсів алгоритмічної корекції;
- реалізація комплексних команд сферичного обльоту цілі (nodal pan, vertical nodal pan) із запобіганням проблемі шарнірного замка (Gimbal Lock) за допомогою алгебри кватерніонів;
- безперервний контроль межових станів із розділенням екрана на безпечну зону, зону попередження та критичну зону;
- можливість проведення синхронного A/B тестування паралельно з еталонною камерою (режим Split-Screen).

5.2 Вимоги до апаратного забезпечення

- Процесор: багатоядерний (рівня Intel Core i5 / AMD Ryzen 5 або вище) для забезпечення обчислення кінематики в режимі реального часу;
- Оперативна пам’ять: не менше 8 Гб;
- Відеокарта: з підтримкою DirectX 11/12 або Vulkan (інтегрована або дискретна) для стабільного рендерингу 3D-сцени в Unity при 60 FPS;
- Пристрої введення: комп'ютерна миша та клавіатура для передачі керуючих імпульсів оператором.

5.3 Вимоги до програмного забезпечення користувача

- Операційна система: Windows 10/11 або macOS;
- Виконавче середовище: графічний рушій Unity або відкомпільований інтерактивний додаток.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури та системний аналіз предметної області	01.03.2026
2.	Розроблення та узгодження технічного завдання	10.03.2026
3.	Математичне моделювання кінематики об'єктива та інерції	20.03.2026
4.	Проєктування алгоритмів контролю меж та запобігання втраті цілі	20.04.2026
5.	Програмна реалізація гібридної системи у середовищі Unity	05.05.201
6.	Організація тестового середовища та тестування	15.05.2026
7.	Оформлення пояснювальної записки та графічного матеріалу	25.05.2026
8.	Попередній перегляд матеріалів дипломного проєкту	30.05.2026

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.004 ПЗ	Удосконалена система	64		
			управління камерою			
			із засобами сприяння			
			оператору.			
			Пояснювальна записка			
	А4	ІАЛЦ.045480.005 Д1	Алгоритм контролю	1		
			межових станів.			
			Схема алгоритму			
	А4	ІАЛЦ.045480.006 Д2	Алгоритм розрахунку	1		
			вектора автоматичної			
			корекції.			
			Схема алгоритму			
	А4	ІАЛЦ.045480.007 Д3	Алгоритм ітерації	1		
			системи у цикл			
			оновлення Unity			
			Схема структурна			
	А4	ІАЛЦ.045480.008 Д4	Гібридне управління	1		
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	*ІАЛЦ.045480.003 ТП*		
Розробив		Пляченко О.О.			Удосконалена система управління камерою із засобами сприяння оператору **Відомість технічного проекту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Сергієнко П.А.					
Консульт.						**«КПІ ім. І. Сікорського», ФПСПМ, КВ-22**	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

						Арк.
					ІАЛЦ.045480.003 ТП	2
Змін.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

6DoF – шість ступенів свободи вільності просторового переміщення та обертання.

API – відкритий програмний інтерфейс.

CNN – згорткова нейронна мережа.

EMA – експоненціальне ковзне середнє.

GPU – графічний процесор.

HSV – колірна модель (відтінок, насиченість, яскравість), що використовується для колірної сегментації зображень.

IoU – метрика перетину по об’єднанню, що застосовується для оцінки якості локалізації цілі в кадрі.

NDC – нормалізовані координати пристрою.

OSC – протокол потокової передачі даних.

SSD – архітектура однопрохідного детектора об'єктів.

UI – інтерфейс користувача.

VFX – індустрія візуальних ефектів.

YOLO – архітектура згорткової нейронної мережі класу однопрохідних детекторів.

ВСТУП

Розвиток цифрового кіновиробництва, відеоігрової індустрії та інтерактивних симуляцій висуває дедалі вищі вимоги до якості роботи з віртуальною камерою. Оператор камери повинен одночасно виконувати складні рухи (панорамування, нахил, наїзд, підйом, обертання), утримувати рухому ціль у полі зору та контролювати плавність переходів - все це вимагає значного досвіду і концентрації.

Сучасні рушії реального часу, зокрема Unity, надають потужні інструменти для роботи з камерою, проте вони або орієнтовані на заздалегідь заскриптовані сцени, або вимагають повного ручного керування. Жоден із доступних інструментів не поєднує в собі підтримку повного набору операторських команд з автоматичним супроводом цілі та регуляторами плавності.

Метою даного дипломного проекту є розробка удосконаленої системи управління камерою із засобами сприяння оператору в середовищі Unreal Engine. Система дозволяє оператору надсилати прості команди - dolly, pan, tilt, pedestal, roll, track, nodal pan, vertical nodal pan - при цьому камера автоматично утримує задану ціль у полі зору, інформує оператора про досягнення меж переміщення та забезпечує плавність і фізично достовірну інерцію руху.

Практична цінність роботи полягає у спрощенні роботи оператора під час динамічних сцен, зниженні порогу входу для початківців, а також у створенні платформи для подальших досліджень у сфері інтелектуального управління камерою.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1 Огляд систем управління віртуальними камерами та польотних симуляторів

Віртуальна камера у середовищах комп'ютерної графіки виконує функцію математичної абстракції, що визначає параметри проєкції тривимірної сцени на двовимірну площину екрана. Програмна реалізація цього об'єкта базується на розрахунку матриці виду (View Matrix) та матриці проєкції (Projection Matrix). У промислових графічних рушіях, зокрема системі Cinemachine платформи Unity, базові механізми управління зводяться до модифікації векторів позиції та кватерніонів просторової орієнтації глобальної системи координат [14].

Для забезпечення плавності переміщення камери між дискретними точками простору та імітації фізичної поведінки реального обладнання застосовується експоненціальна інтерполяція. Вказаний математичний апарат дозволяє мінімізувати ефект кінематичного ривка під час зміни координат об'єктива. Математична модель розрахунку поточної позиції камери P_{curr} на кожному кроці дискретного часу Δt для наближення до цільової позиції P_{target} описується формулою (1.1) [14]:

$$P_{curr} = P_{prev} + \left(P_{target} - P_{prev}\right) * \left(1 - e^{-\lambda \Delta t}\right) \quad (1.1)$$

де λ - коефіцієнт жорсткості (dampening factor), який визначає швидкість реакції системи на зміну вхідних координат.

Розширення функціональних можливостей систем управління вимагає залучення алгоритмів, що застосовуються у спеціалізованих польотних симуляторах (наприклад, архітектура фізичного ядра платформи X-Plane). В основі їхньої роботи лежить інтегрування рівнянь динаміки

твердого тіла. Перенесення зазначеного підходу на віртуальну камеру дозволяє оперувати фізичними параметрами віртуальної маси та кутової інерції. Реакція камери на вхідні керуючі імпульси від оператора моделюється за допомогою рівнянь Ейлера, наведених у формулі (1.2) [6]:

$$M = \mathrm{I} * \dot{\omega} + \omega * (\mathrm{I} * \omega) \qquad (1.2)$$

де M - вектор моментів сил керування (математичний аналог команд pan та tilt).

I – тензор інерції віртуальної камери , ω – вектор кутової швидкості.

Впровадження інерційної моделі створює умови для розрахунку реалістичного затухання швидкості обертання після припинення подачі керуючого сигналу. Це є критично важливим аспектом для імітації маси кінематографічного обладнання.

Базова архітектура алгоритму автоматичного супроводу рухомої цілі передбачає наявність контуру зворотного зв’язку. У ньому розраховується вектор помилки між оптичною віссю об’єктива та фактичним положенням об’єкта(рисунок 1.1).

Рисунок 1.1 - Структурна схему взаємодії обчислювальних блоків системи автоматичного супроводу.

Для оцінки ефективності підходів до побудови архітектури управління проведено порівняльний аналіз стандартних засобів, процедурних камер (на базі рішень Unity) та спеціалізованих симуляційних модулів. Результати систематизовано у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз архітектур управління віртуальними камерами

Тип архітектури	**Математична модель руху**	**Наявність фізичної інерції**	**Застосовність для гібридного керування**

Базова камера (Transform)	Лінійна кінематика	Відсутня	Низька (потребує прямої маніпуляції)
Процедурна (Cinemachine)	Експоненціальне згладжування	Емулюється (Dampening)	Середня (виникають конфлікти векторів)
Симуляційна (Flight Sim)	Динаміка твердого тіла	Повноцінна (Тензор інерції)	Висока (дозволяє сумацію імпульсів)

Дослідження комерційних архітектур доводить, що процедурні інструменти надають потужний базис для автоматичного слідування за об'єктом. Вони використовують концепцію нечутливих зон (Dead Zones) у центрі кадру, де рух цілі не викликає обертання камери, що ефективно стабілізує зображення [14]. Однак логіка функціонування таких систем орієнтована на повну автономність. При спробі оператора застосувати ручні команди зміщення (dolly, pedestal, track) паралельно з активним алгоритмічним трекінгом виникають обчислювальні конфлікти через перезапис матриці трансформацій. Вказана проблема генерує науково-практичну потребу у розробці удосконаленого алгоритмічного апарату безпосередньо у середовищі Unity. Проєктована система повинна інтегрувати кінематичну фізику польотних симуляторів із засобами безперервного ручного контролю, забезпечуючи стійке формування гібридного простору взаємодії оператора з віртуальним середовищем.

1.2 Дослідження комерційних та рекламних механізмів автоматизації роботи оператора

Сучасні підходи до автоматизації роботи оператора у комерційній зйомці, індустрії візуальних ефектів (VFX) та 3D-анімації спираються на використання спеціалізованих апаратно-програмних комплексів. У сфері кінематографу та під час створення високобюджетних рекламних матеріалів індустріальним стандартом виступають роботизовані системи

управління рухом (Motion Control), яскравим представником яких є установки серії MRMC Bolt. Зазначені комплекси забезпечують абсолютну повторюваність складних просторових маневрів об’єктива. Програмне забезпечення установок розв’язує задачу оберненої кінематики, перетворюючи координати цільової точки віртуального простору на масив кутових швидкостей для кожного сервопривода багатоланкового маніпулятора. Математичний апарат спирається на обчислення матриці Якобі, що дозволяє досягати міліметрової точності позиціонування під час швидкісної зйомки [9].

У сегменті динамічної репортажної та спортивної відеофіксації домінують автономні безпілотні платформи, оснащені модулями комп’ютерного зору. Технологія ActiveTrack від компанії DJI використовує нейромережеві алгоритми для безперервного утримання цілі в заданій області кадру під час активного переміщення носія. Споріднене рішення Skydio Autonomy Engine застосовує алгоритми візуальної одометрії та побудови об’ємної воксельної карти простору. Це дає змогу обчислювальному модулю прогнозувати кінематику об’єкта при його тимчасовому перекритті перешкодами. Для забезпечення плавності супроводу цілі системи просторової стабілізації використовують пропорційно-інтегрально-диференціальні (ПІД) регулятори. Вони формують плавні керуючі імпульси, мінімізуючи різкі коливання оптичної осі. Математична модель класичного ПІД-регулятора для обчислення кутової швидкості обертання камери $\omega(t)$ на основі вектора похибки позиціонування $e(t)$ описується рівнянням (1.3) [5]:

$$\omega(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_{d\frac{de(t)}{dt}} \quad (1.3)$$

де K_p, K_i, K_d – коефіцієнти пропорційної, інтегральної та диференціальної ланок. Їх калібрування імітує віртуальну масу обладнання, забезпечуючи природне затухання швидкості після зупинки об’єкта.

Для завдань 3D-анімації та створення інтерактивного контенту застосовуються програмні рішення, безпосередньо інтегровані в графічні рушії. У середовищі Unity базовим інструментом для розробки процедурних камер слугує пакет Cinemachine. Він пропонує підхід, заснований на розподілі площини кадру на функціональні зони: зону нечутливості (Dead Zone), де рух об’єкта ігнорується алгоритмами трекінгу, та зону м’якого супроводу (Soft Zone), потрапляння в яку ініціює обертання камери. На рисунку 1.2 наведено структурну схему розподілу екранного простору під час автоматизованого трекінгу цілі.

Для забезпечення сумісності віртуального простору Unity з фізичним знімальним обладнанням впроваджуються протоколи потокової передачі даних, зокрема FreeD та OSC. Вони забезпечують трансляцію координат реальної камери на її цифровий двійник у режимі реального часу, що є концептуальною основою технології віртуального виробництва (Virtual Production).

Таблиця 1.2 - Порівняльний аналіз комерційних систем автоматизації камери

Система / Технологія	**Основний метод відстеження**	**Інтерактивність управління**	**Застосовність у віртуальних середовищах**
MRMC Bolt (Motion Control)	Обернена кінематика	Відсутня (жорсткий сценарій)	Висока (як джерело координат)
DJI ActiveTrack	Згорткові мережі (CNN)	Часткова (лише коригування цілі)	Низька (орієнтація на фізичний світ)

Unity Cinemachine	Процедурна логіка (Dead Zone)	Обмежена (виникають конфлікти)	Максимальна (нативна інтеграція)

Аналіз ринку демонструє глибоку поляризацію доступних технологій. Апаратні комплекси переважно орієнтовані на відтворення заздалегідь розрахованих маневрів. Алгоритми безпілотних систем максимізують автономність, позбавляючи оператора можливості здійснювати тонке ручне налаштування композиції (pedestal, roll, nodal pan) у процесі зйомки. Наявні процедурні інструменти платформи Unity ефективно вирішують завдання слідування, проте демонструють нестабільну поведінку при спробах одночасного ручного втручання через накладання векторів трансформації [14]. Виявлені обмеження підтверджують потребу у проєктуванні гібридної архітектури, здатної безперервно підтримувати об’єкт у фокусі за допомогою розрахунку віртуальної інерції, паралельно обробляючи директивні команди управління.

1.3 Аналіз технологій автоматичного виявлення та супроводження цілі

Проєктування автономних модулів управління просторовим положенням віртуального об’єктива потребує використання надійних методів ідентифікації рухомих об’єктів. Від точності отримання координат цілі залежить стабільність функціонування математичних моделей згладжування та імітації інерції. Сучасні підходи до вирішення задачі локалізації розподіляються на три базові категорії: використання фізичних або віртуальних маркерів, колірна сегментація (хромакей) та застосування алгоритмів комп’ютерного зору на базі згорткових нейронних мереж.

Технологія маркерного відстеження у фізичному середовищі базується на фіксації інфрачервоного випромінювання від світловідбивних міток. Програмний комплекс обчислює просторові координати об’єкта за допомогою оптичної тріангуляції з кількох сенсорів. Під час розробки в середовищі Unity аналогом маркерного підходу є пряме зчитування компонента Transform або використання іменованих точок прив’язки (Sockets) на скелетному каркасі тривимірної моделі. Зазначений метод гарантує абсолютну точність локалізації та вимагає мінімальних обчислювальних ресурсів, оскільки оперує готовими векторами координат глобальної системи [13].

Метод колірної сегментації (зелений екран) передбачає виділення об’єкта з відеопотоку шляхом аналізу пікселів у колірному просторі HSV (Hue, Saturation, Value). Алгоритм ізолює діапазон значень відтінку, ідентифікуючи пікселі фону для їх подальшого відсікання. Хоча підхід характеризується високою швидкодією, він демонструє критичну вразливість до змін динамічного освітлення та появи тіней, що суттєво обмежує його автономне застосування в умовах складних тривимірних сцен зі змінними джерелами світла [10].

Найбільш універсальним інструментом ідентифікації об’єктів є згорткові нейронні мережі (CNN). Архітектури класу однопрохідних детекторів (наприклад, YOLO або SSD) аналізують зображення цілком за один прохід конвеєра, генеруючи набір обмежувальних рамок (bounding boxes) навколо знайдених цілей. Для оцінки якості локалізації цілі в кадрі застосовується метрика перетину по об’єднанню (Intersection over Union, IoU), математична модель якої описується формулою (1.4) [11]:

$$IoU = \frac{|B_p \cap B_t|}{|B_p \cup B_t|} \tag{1.4}$$

де B_p – площа згенерованої обмежувальної рамки.

B_t – еталонна площа об’єкта (ground truth). Для переведення результатів детекції у формат, придатний для системи управління камерою, обчислюються координати геометричного центру знайденої рамки x_c, y_c згідно з рівняннями (1.5) [10]:

$$x_c = \frac{x_{min} + x_{max}}{2}$$
$$y_c = \frac{y_{min} + y_{max}}{2} \quad (1.5)$$

Отримані двовимірні координати трансформуються у вектори відхилення від центру екрана та подаються на вхід ПІД-регуляторів для обчислення зусилля панорамування.

Рисунок 1.2- Блок-схема алгоритму супроводження цілі

Для оцінки доцільності інтеграції кожної з технологій у віртуальне середовище розробки проведено порівняльний аналіз, результати якого зведено у таблицю 1.3.

Таблиця 1.3 – Порівняльні характеристики методів ідентифікації об’єктів

Технологія	**Типова точність позиціонування**	**Вимоги до обчислювальних ресурсів**	**Застосовність у віртуальному середовищі Unity**
Читання Transform (Віртуальні маркери)	Абсолютна (похибка 0%)	Мінімальні (< 1 мс на кадр)	Оптимальна для прямих обчислень кінематики
Колірна сегментація	Залежить від освітлення	Середні	Низька (потребує рендерингу додаткових масок)
Згорткові мережі (CNN)	Висока (IoU > 0.85)	Високі (потребує GPU-інференсу)	Середня (виправдана для симуляції комп’ютерного зору)

Аналіз підтверджує, що для забезпечення максимальної продуктивності системи удосконаленого управління в Unity найбільш раціональним є використання віртуальних маркерів (прямого зчитування координат). Застосування нейромережевого конвеєра або обробки відрендерених кадрів (через RenderTexture) генерує надмірне навантаження на обчислювальне ядро. Зчитування абсолютних тривимірних координат об’єкта з їх подальшою проєкцією на площину кадру (Viewport) формує достатню інформаційну базу для розрахунку алгоритмів інерції та перевірки межових станів об’єктива, усуваючи необхідність залучення важких конвеєрів комп’ютерного зору .

1.4 Обґрунтування доцільності та актуальності створення удосконаленої системи

Складність маніпулювання віртуальним простором у режимі реального часу зумовлює підвищене когнітивне навантаження на користувача. Під час зйомки динамічних сцен оператор змушений одночасно контролювати просторове позиціонування об’єктива, параметри фокусування та утримувати головний об’єкт інтересу в межах композиційної рамки. Більшість існуючих програмних рішень реалізують один із двох граничних підходів: абсолютну автономність обчислювальних алгоритмів або жорсткий ручний контроль. Відсутність рішень, які поєднують інтерактивні команди з процедурною стабілізацією, створює потребу в розробці удосконаленого алгоритмічного ядра, орієнтованого на середовище Unity.

Аналіз вбудованих інструментів Unity, зокрема компонента Transform, демонструє їх непристосованість до виконання кінематографічних маневрів без застосування додаткових математичних обгорток. Спеціалізований пакет Cinemachine частково розв’язує задачу процедурного трекінгу, використовуючи логіку зон нечутливості (Dead Zones). Проте внутрішня архітектура даного пакета жорстко підпорядковує координати камери власному конвеєру оновлення. Спроба подати ручну команду комплексного зміщення (наприклад, nodal pan або pedestal) паралельно з активним алгоритмом супроводу призводить до перезапису матриці трансформацій. Наслідком цього є виникнення обчислювальних конфліктів та візуального тремтіння (jittering) кадрів [14].

Для розв’язання проблеми конфлікту векторів управління пропонується математична модель гібридної архітектури. Її основою є використання вагової функції $W_{limit}(e)$, яка динамічно розподіляє пріоритети між ручним введенням та автоматичним утриманням цілі. Функція залежить від відхилення об’єкта e від центру екрана відносно

заданої межі допустимої зони e_{th}. Математична модель функції активації захисного бар'єра описується логістичним рівнянням (1.6) [5]:

$$W_{limit}(e) = \frac{1}{1 + \exp\left(-k(e - e_{th})\right)} \quad (1.6)$$

де k – коефіцієнт крутості переходу, який визначає швидкість реакції системи стабілізації.

При наближенні об'єкта до межі екрана значення W_{limit} асимптотично прямує до одиниці, ініціюючи генерацію попереджувального сигналу та програмне блокування ручного зміщення у небезпечному напрямку. Графічне відображення поведінки вагової функції при різних значеннях коефіцієнта k подано на рисунку 1.3.

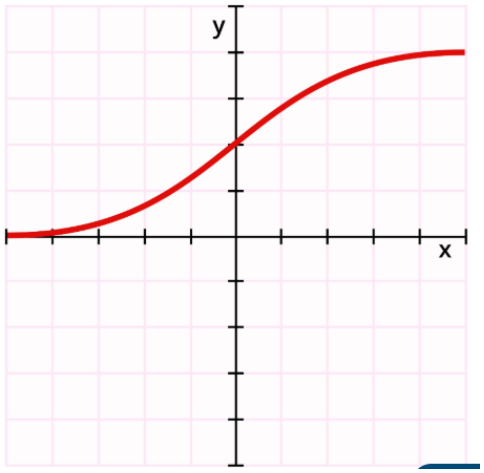


Рисунок 1.3 - Графік вагової функції W_{limit} (e) залежно від відхилення цілі від межі кадру

Для обґрунтування доцільності створення нового модуля сформовано порівняльну таблицю існуючих архітектур управління у середовищі Unity та запропонованого гібридного концепту (таблиця 1.4).

Таблиця 1.4 – Порівняльний аналіз систем управління у середовищі Unity

Функціональні характеристики	**Базовий Transform**	**Unity Cinemachine**	**Запропонована система**
Підтримка гібридного керування	Відсутня	Обмежена (конфлікт векторів)	Повна (сумація імпульсів)
Контроль межових станів	Відсутній	Відсутній	Реалізовано через W_{limit}
Генерація попереджень	Відсутня	Відсутня	Інтегрована підсистема
Модель віртуальної інерції	Лінійна інтерполяція	Експоненціальне згладжування	Фізична інерційна динаміка

Програмна реалізація гібридного підходу вимагає розробки спеціалізованого диспетчера команд. Його завданням є агрегація вхідних сигналів перед передачею до фізичного ядра. Впровадження запропонованої структури усуває архітектурні недоліки стандартних інструментів. Інтеграція логістичної моделі W_{limit} (e) гарантує безперервний контроль межових станів об’єктива. Проєктований комплекс здатний об’єднати гнучкість команд dolly, roll, pan з надійністю автономного трекінгу. Зазначені переваги повною мірою підтверджують науково-технічну актуальність створення удосконаленої системи управління для симуляційних середовищ .

1.5 Вимоги до системи та обґрунтування вибору середовища розробки Unity

Проєктування програмного комплексу для управління візуалізацією тривимірного простору вимагає чіткої формалізації технічних умов. Розроблюваний інструментарій повинен відповідати визначеній системі функціональних та нефункціональних вимог, що безпосередньо формують його архітектурний каркас. До базових функціональних критеріїв належить

здатність алгоритмічного ядра обробляти паралельні потоки вхідних даних: прямі команди просторового маніпулювання (pan, tilt, dolly, roll, pedestal, track) та автоматичні імпульси від підсистеми розрахунку координат цілі. Комплекс має забезпечувати безперервний моніторинг просторових обмежень фокусної відстані з генерацією превентивних керуючих впливів для уникнення втрати об’єкта фокусування.

Нефункціональні вимоги зосереджені на показниках швидкодії, стабільності та модульності. Підсистема трекінгу повинна функціонувати в режимі реального часу, витрачаючи на обчислення просторових трансформацій не більше ніж 16 мілісекунд, що є стандартом для забезпечення частоти 60 кадрів на секунду. Окрім продуктивності, вагомим критерієм виступає кросплатформність та апаратна незалежність розрахунків. Віртуальне середовище повинно зберігати ідентичну математичну поведінку фізики інерції незалежно від обчислювальної потужності машини. Це вимагає відмови від жорсткої прив’язки кінематичних розрахунків до частоти рендерингу на користь використання інтервалів фіксованого кроку часу (fixed delta time).

Для практичної реалізації поставлених завдань обрано середовище розробки Unity. Прийняття даного рішення базується на комплексному аналізі можливостей платформи для створення систем гібридної навігації. Середовище надає оптимізований математичний апарат для роботи з тривимірними координатами та складними просторовими обертаннями. На відміну від графічних фреймворків, що потребують низькорівневого програмування матриць перетворень, Unity нативно оперує кватерніонами. Це повністю усуває проблему шарнірного замка (gimbal lock) під час виконання комплексних команд, зокрема одночасного застосування обертань навколо поперечної та поздовжньої осей.

Для забезпечення безперервної плавності обертання оптичної осі при переході між двома кутовими станами q_1та q_2 у платформі Unity

застосовується сферична лінійна інтерполяція (Slerp). Математична модель даного обчислювального процесу описується формулою (1.7) [2]:

$$q(t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)} q_1 + \frac{\sin(t\Omega)}{\sin(\Omega)} q_2 \qquad (1.7)$$

де t [0, 1] – параметр інтерполяції, Ω - кут між двома заданими кватерніонами. Наведений математичний апарат є критично важливим фундаментом для програмування віртуальної інерції об’єктива.

Вагомим аргументом на користь використання Unity є глибока інтеграція об’єктно-орієнтованої мови програмування C#. Вона поєднує високу продуктивність інструкцій із гнучкістю проєктування структур даних. Особливої уваги заслуговує життєвий цикл виконання скриптів у системі. Для віртуальної камери важливо зчитувати координати цілі виключно після того, як обчислені всі анімації та переміщення інших об’єктів сцени. Платформа пропонує метод пізнього оновлення (LateUpdate), який викликається наприкінці циклу трансформацій, гарантуючи відсутність ефекту візуального розсинхронізування (jittering).

Хоча стандартний інструментарій має певні обмеження при побудові гібридних алгоритмів, відкритість програмного інтерфейсу (API) дає змогу розробити власні керуючі класи. Вони наслідуватимуть оптимізовані компоненти рушія, реалізуючи логістичну модель вагових коефіцієнтів, обґрунтовану в підрозділі 1.4. Деталізоване зіставлення технічних вимог до проєкту з інструментарієм обраної платформи наведено у таблиці 1.5.

Таблиця 1.5 - Відповідність вимог до системи функціоналу середовища Unity

Категорія вимог	**Вимога до розроблюваної системи**	**Інструментарій платформи Unity**

Обчислювальна	Синхронізація трансформацій у часі	Цикл LateUpdate / FixedUpdate
Математична	Запобігання шарнірному замку	Нативна структура Quaternion
Архітектурна	Модульність та ізоляція підсистем	Компонентна модель MonoBehaviour
Інтеграційна	Зчитування абсолютних координат	Властивість глобальної позиції (Transform)

На рисунку 1.4 зображено узагальнену структурну схему взаємодії розроблюваних модулів із базовими підсистемами графічного рушія Unity.

Рисунок 1.4 - Блок-схема алгоритму роботи гібридної системи керування камерою

Концентрація етапів розробки у просторі Unity дозволяє оптимізувати процес створення системи сприяння оператору. Наявність оптичних абстракцій звільняє системні ресурси для розв’язання фундаментальної задачі: алгоритмізації згладжування, моніторингу меж та динамічного розподілу пріоритетів між ручним та автоматичним керуванням. Вибране середовище повністю задовольняє критерії продуктивності, формуючи надійний технічний базис для виконання етапу математичного проєктування.

1.6 Висновки до розділу 1

У першому розділі проведено комплексний системний аналіз технологій управління віртуальними камерами та засобів сприяння оператору. Виявлено фундаментальну проблему наявних інструментів: вони пропонують або жорстке автономне слідування за об’єктом, або виключно низькорівневий ручний контроль (6DoF). Відсутність гібридних механізмів ускладнює виконання комплексних просторових маневрів (зокрема *nodal pan*), що призводить до візуальної дестабілізації під час зйомки.

Аналіз методів ідентифікації цілі довів, що застосування згорткових нейронних мереж або колірної сегментації генерує критичне навантаження на обчислювальні ресурси в режимі реального часу. Для забезпечення швидкодії обґрунтовано використання методу зчитування абсолютних тривимірних координат (аналога віртуальних маркерів) з подальшою проєкцією на площину екрана. Це вивільняє потужності процесора для імітації інерційних властивостей оптики.

Для подолання конфлікту керуючих векторів (накладання ручного введення на процедурний трекінг) запропоновано імплементацію спеціалізованої вагової функції. Її призначенням є динамічне обмеження

впливу оператора при наближенні об’єкта до граней піраміди видимості (Frustum). Критерій ефективності розроблюваної системи базується на інтегральному функціоналі якості, що мінімізує похибку позиціонування та максимізує плавність ходу шляхом гасіння різких ривків.

Базовим середовищем розробки затверджено платформу Unity. Головними факторами вибору стали нативна підтримка кватерніонів (що остаточно розв’язує проблему шарнірного замка під час складних обертань) та наявність фази LateUpdate, яка гарантує застосування трансформацій виключно після фіналізації фізичних переміщень сцени. Узагальнюючи результати аналітики, сформовано такі проєктні рішення для виявлених недоліків:

- Конфлікт векторів керування вирішується розробкою диспетчера команд для паралельної обробки ручного введення та трекінгу.
- Відсутність відчуття маси камери компенсується інтеграцією інерційних регуляторів для природного затухання швидкості.
- Втрата об’єкта при різких маневрах попереджається впровадженням функції контролю меж із генерацією попереджень та блокуванням небезпечних рухів.

Сформульовані технічні вимоги та обраний математичний апарат створюють стійкий методологічний фундамент, що повністю обґрунтовує доцільність роботи та дозволяє перейти до етапу математичного моделювання системи управління у другому розділі дипломного проєкту.

2. МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ

2.1 Архітектура автоматизованої системи управління камерою

Архітектура системи базується на принципах модульності та слабкої зв’язності компонентів у середовищі Unity. Вона декомпозується на чотири логічні блоки: диспетчер керуючих сигналів, модуль розрахунку кінематики цілі, ядро віртуальної інерції та підсистему моніторингу межових станів. Диспетчер нормалізує директивні команди оператора у вектори швидкостей для подальшої обробки фізичним ядром.

Математичний опис архітектури реалізується через представлення камери як динамічної системи у просторі станів. Вектор стану системи X_t у довільний момент часу t включає глобальну позицію P(t) кватерніон орієнтації Q(t), вектор лінійної швидкості V(t) та вектор кутової швидкості **Ω**(t). Рівняння стану, що описує перехід системи на наступний крок дискретного часу k, формалізується функціональною залежністю (2.1) [5]:

$$X_{k+1} = f\left(X_k, U_{k,}E_k, \Delta t\right) \tag{2.1}$$

де U_k - вектор нормованих команд оператора, E_k - вектор просторової похибки, згенерований модулем супроводу цілі, Δt - крок інтегрування (фіксований час кадру).

Модуль розрахунку кінематики цілі функціонує паралельно з диспетчером введення. Він зчитує абсолютні координати об'єкта інтересу та обчислює ідеальний вектор орієнтації, необхідний для утримання цілі в заданій зоні кадру. Ядро віртуальної інерції виступає центральним агрегатором, який об'єднує вектори U_k та E_k . Для уникнення конфліктів застосовується алгоритм зваженого змішування керуючих впливів.

Результуючий вектор кутової швидкості Ω_{res}розраховується за формулою (2.2) [4]:

$$\Omega_{res} = W_m * \Omega_{manual} + (\mathrm{I} - W_m) * \Omega_{auto} \quad (2.2)$$

де W_m - діагональна матриця вагових коефіцієнтів пріоритету ручного керування, I - одинична матриця, $\Omega_{manual}, \Omega_{auto}$ - кутові швидкості, згенеровані оператором та системою автоматичного супроводу відповідно. Елементи матриці W_m - динамічно коригуються підсистемою моніторингу межових станів. Склад та призначення ключових компонентів системи управління наведено у таблиці 2.1.

Таблиця 2.1 - Декомпозиція архітектури системи управління камерою

Назва компонента	**Вхідні дані**	**Вихідні дані**	**Функціональне призначення**
Диспетчер керуючих сигналів	Команди оператора	Вектор U_k	Нормалізація та фільтрація вводу
Модуль кінематики цілі	Координати Transform цілі	Вектор E_k	Обчислення просторової похибки
Ядро віртуальної інерції	Вектори $U_k, E_k, \Delta t$	Кватерніон Q(t)	Математичне змішування, інтерполяція
Підсистема моніторингу	Координати цілі у Viewport	Матриця W_m	Захист від втрати фокуса об'єктива

Структурна візуалізація зв'язків між описаними компонентами та конвеєром оновлення рушія Unity подана на рисунку 2.1.

Рисунок 2.1 - Алгоритм взаємодії модулів кінематики та моніторингу для розрахунку параметрів руху

Підсистема моніторингу межових станів інтегрується безпосередньо перед етапом застосування трансформацій до фізичного об'єкта. Вона аналізує прогнозовану позицію цілі на площині екрана. У разі виявлення ризику виходу об'єкта за допустимі межі піраміди видимості, підсистема генерує апаратне переривання. Це спричиняє миттєве зменшення відповідних елементів матриці W_m , примусово передаючи пріоритет алгоритмам автоматичного центрування.

Запропонована архітектурна модель формує замкнений контур управління з негативним зворотним зв'язком. Відокремлення логіки збору даних від ядра математичних розрахунків створює масштабовану базу,

придатну для подальшої деталізації алгоритмів захоплення цілі та налаштування параметрів регуляторів плавності .

2.2 Алгоритми захоплення та автоматичного утримання рухомої цілі в полі зору

Забезпечення автономного утримання об'єкта в кадрі вимагає використання надійних математичних моделей просторової орієнтації. У контексті розробки системи управління віртуальною камерою процес наведення зводиться до безперервного обчислення цільового вектора погляду, який має збігатися з напрямком на задану точку інтересу. Базовий математичний апарат спирається на векторну алгебру та застосування кватерніонів для опису обертань у тривимірному просторі. Використання кутів Ейлера визнано неефективним через ризик виникнення шарнірного замка (Gimbal Lock) – втрати одного ступеня вільності просторового обертання при збігу двох осей. Геометричну суть цієї проблеми проілюстровано на рис. 2.2.

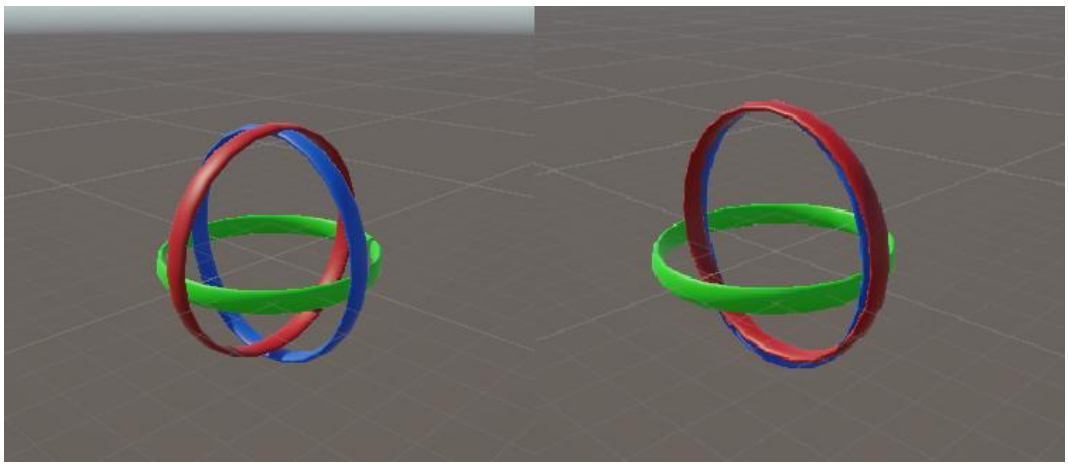


Рисунок 2.2 – Втрата ступеня вільності (Gimbal Lock)

Для визначення необхідного просторового повороту розраховується радіус-вектор напрямку. Нехай P_c характеризує поточні абсолютні

координати оптичного центру об'єктива у глобальному просторі, P_t відображає просторову позицію рухомої цілі. Нормований вектор напрямку погляду V_{dir} обчислюється як різниця вказаних векторів, поділена на їхню довжину, згідно з формулою (2.3) [3]:

$$V_{dir} \frac{P_t - P_c}{||P_t - P_c||} \tag{2.3}$$

Коректна проєкція віртуальної сцени потребує додаткового формування опорного вектора «вгору» V_{up} , який задає площину горизонту і запобігає неконтрольованому обертанню камери навколо осі погляду (Roll). Спираючись на вектори V_{dir} та V_{up}, обчислюється ортогональний базис, який згодом конвертується у цільовий кватерніон орієнтації Q_{target}

Пряме застосування розрахованого кватерніона для миттєвого оновлення координат об'єкта створює ефект абсолютно жорсткої прив'язки (Hard Lock). Під час високодинамічного руху цілі або наявності мікроколивань її координат внаслідок роботи фізичного рушія це генерує високочастотне тремтіння кадру. Оптимізація процесу наведення досягається за рахунок імплементації алгоритмів низькочастотної фільтрації. В умовах детермінованого віртуального середовища доцільно застосовувати експоненціальне ковзне середнє (EMA).

Згладжене положення фокусної точки P_{smooth}на поточному кроці дискретного часу k розраховується за формулою (2.4) [5]:

$$P_{smooth}^{(k)} = aP_t^{(k)} + (1 - a)P_{smooth}^{(k-1)} \tag{2.4}$$

де a є (0, 1] виступає ваговим коефіцієнтом, який регулює ступінь інерційності візуального переслідування. Зменшення значення a

пропорційно збільшує затримку реакції камери, достовірно імітуючи плавність роботи фізичних пристроїв панорамування.

Наближення поведінки алгоритмічного ядра до дій реального оператора зумовлює необхідність проєктування композиційних зон. Екранний простір розподіляється на зону нечутливості та активну зону реакції. Якщо проєкція об'єкта на двовимірну площину екрана не перетинає меж зони нечутливості, підсистема блокує генерацію керуючих імпульсів. Перетворення абсолютних тривимірних координат цілі (X_t, Y_t, Z_t) у гомогенні координати відсікання ($x_c y_c z_c w_c$) реалізується шляхом множення на матрицю виду V та матрицю проєкції P. Математична модель перетворення подана у рівнянні (2.5) [1][3]:

$$[x_c\ y_c\ z_c\ w_c\ ]=\text{P*V*}[X_t\ Y_t\ Z_t\ 1] \quad (2.5)$$

Нормалізовані координати пристрою (NDC) розраховуються шляхом ділення на гомогенну компоненту w_c . Отримані значення по осях абсцис та ординат використовуються для розрахунку двовимірного вектора відхилення об'єкта від геометричного центру композиції. Схема геометричної взаємодії векторів та розподілу екранних зон подана на рисунку 2.3.

Рисунок 2.3 - Алгоритм розрахунку нормалізованих координат екрана для системи автоматичного відстеження об'єкта

При перетині ціллю межі зони нечутливості ініціюється робота пропорційно-інтегрального (ПІ) регулятора. Його завдання полягає у розрахунку необхідної кутової швидкості для компенсації відхилення. Характеристики різних математичних методів стабілізації наведено у таблиці 2.2.

Таблиця 2.2 - Порівняльний аналіз математичних методів стабілізації трекінгу

Назва методу	**Обчислювальна складність**	**Наявність затримки позиціонування**	**Ефективність гасіння мікроколивань**

Жорстка прив'язка (Hard Look)	Мінімальна	Відсутня (миттєва реакція)	Низька (наявний джиттер)
Експоненціальне згладжування	Низька	Регульована коефіцієнтом a	Висока
ПІ-регулятор відхилення	Середня	Залежить від інтегральної ланки	Максимальна

Комплексне застосування фільтрації вхідних координат та ПІ-регуляторів формує надійний базис для захоплення цілі. Інтеграція зазначених алгоритмів забезпечує стійке утримання фокуса під час інтенсивних просторових переміщень, створюючи передумови для подальшого розрахунку команд ручного зміщення. Зважаючи на вимоги до структурування технічної документації, безпосередня реалізація розроблених алгоритмів у вигляді лістингів програмного коду розглядається на етапі опису практичної імплементації у третьому розділі проєкту.

2.3 Математичні моделі реалізації простих команд оператора

Керування просторовим положенням віртуальної камери зводиться до маніпуляції шістьма ступенями вільності (6DoF). Класична операторська термінологія класифікує ці маніпуляції на дві незалежні групи: трансляційні переміщення (dolly, pedestal, track) та ротаційні обертання (tilt, pan, roll). Для коректної конвертації імпульсів від пристроїв введення у тривимірний простір середовища розробки формується строга математична модель кінематики об'єктива, яка базується на векторній алгебрі та теорії кватерніонів.

Трансляційні команди відповідають за лінійне зміщення оптичного центру вздовж осей локальної системи координат камери. Команда track ініціює рух вздовж поперечної осі X, pedestal визначає зміщення вздовж вертикальної осі Y, а dolly керує наближенням або віддаленням вздовж поздовжньої осі Z. Вектор лінійної швидкості в локальному просторі

$V_{loc} = [v_x v_y v_z]$ формується на основі нормованих значень вводу оператора. Оскільки осі глобальної системи координат сцени не збігаються з локальними осями об'єктива під час його довільної орієнтації, виникає потреба просторового перетворення векторів. Нова позиція камери P_{k+1} на наступному кроці дискретного часу Δt обчислюється шляхом повороту локального вектора швидкості за допомогою поточного кватерніона орієнтації Q_k згідно з формулою (2.6) [2]:

$$P_{k+1} = P_k + (Q_k \otimes V_{loc} \otimes Q_k^{-1})\Delta t \qquad (2.6)$$

де Q_k^{-1} - спряжений кватерніон.

Множення $\otimes$ виконується за правилами добутку Гамільтона, для чого тривимірний вектор V_{loc} попередньо перетворюється на чистий кватерніон з нульовою скалярною частиною V = (0, v_x, v_y, v_z). Наведений математичний апарат дозволяє камері рухатися відносно напрямку свого погляду, а не відносно статичних осей світу.

Ротаційні команди ініціюють зміну кутової орієнтації об'єкта. Команда tilt генерує обертання навколо локальної осі X, pan - навколо осі Y, roll - навколо осі Z.

Математичний опис просторових обертань базується на обчисленні приросту кватерніона ΔQ. Для кожної осі обертання, заданої одиничним вектором u = (v_x, v_y, v_z), на заданий кут θ, генерується локальний кватерніон за рівнянням (2.7) [2]:

$$q_{rot} = \cos\left(\frac{\theta}{2}\right) + \left(v_x i + v_y j + v_z k\right)\sin\left(\frac{\theta}{2}\right) \qquad (2.7)$$

Кут θ визначається як добуток кутової швидкості введення на крок часу Δt. Результуюча зміна орієнтації ΔQ є добутком кватерніонів кожної

окремої команди: $\Delta Q = q_{pan} \otimes q_{tilt} \otimes q_{roll}$.Оновлення глобального кватерніона стану камери Q_{k+1} виконується не комутативним множенням згідно з рівнянням (2.8) [2]:

$$Q_{k+1} = Q_k \otimes \Delta Q \tag{2.8}$$

Важливим аспектом проєктування є розв'язання проблеми зміщення горизонту. Класична команда pan у кінематографі передбачає обертання камери навколо глобальної осі Y (вектора гравітації), а не навколо локальної осі об'єктива, яка може бути нахилена. Застосування обертання навколо нахиленої локальної осі Y під час панорамування призводить до виникнення паразитного крену (roll-induced pitch). Для усунення даної похибки вектор осі для команди pan примусово нормалізується до глобального вектора(0, 1, 0) перед обчисленням кватерніона повороту. Систематизацію математичних перетворень та відповідність осей наведено у таблиці 2.3.

Таблиця 2.3 – Математичні моделі класичних операторських команд

Команда оператора	**Тип перетворення**	**Вісь впливу**	**Математична операція**
Track	Трансляція	Локальна X	Векторне додавання з ротацією
Pedestal	Трансляція	Локальна Y	Векторне додавання з ротацією
Dolly	Трансляція	Локальна Z	Векторне додавання з ротацією
Tilt	Ротація	Локальна X	Добуток Гамільтона
Pan	Ротація	Глобальна Y	Добуток Гамільтона (ізольований)
Roll	Ротація	Локальна Z	Добуток Гамільтона

Сформовані рівняння трансляції та ротації створюють базовий кінематичний контур. Отримані вектори P_{k+1}та Q_{k+1} формують цільовий математичний стан (Target State). Надалі ці ідеальні координати .

2.4 Моделювання комплексних команд обльоту об’єкта

Управління віртуальним знімальним процесом вимагає реалізації маневрів, які передбачають синхронну зміну як просторових координат, так і кутової орієнтації об'єктива. На відміну від простих команд, що виконуються відносно власної локальної осі камери, комплексні команди обльоту (за визначенням автора проєкту - nodal pan та vertical nodal pan) ініціюють рух камери по сферичній траєкторії навколо обраної цілі. Цей кінематичний прийом забезпечує збереження об'єкта в центрі кадру при динамічній зміні ракурсу та перспективи тла. Математичний апарат опису таких рухів базується на концепції перенесення центру обертання (pivot point) з оптичного центру об'єктива на глобальні координати цілі.

Основою для розрахунку траєкторії обльоту є вектор відносного позиціонування (радіус-вектор) r. Він формується як різниця між поточним радіус-вектором позиції камери P_{cam} та радіус-вектором цілі P_{obj} .Для виконання горизонтального обльоту (nodal pan) застосовується обертання вектора r навколо вектора гравітаційної вертикалі сцени (найчастіше це глобальна вісь Y). Для забезпечення обчислювальної стабільності та уникнення накопичення геометричних похибок використовується кватерніонне множення. Нова позиція камери $P_{cam}^{(k+1)}$ на кроці дискретного часу k+1 при введенні кутової швидкості ω^y визначається системою рівнянь (2.9) [4]:

$$r_k = P_{cam}^k - P_{obj}^k$$

$$q_{nodal} = \cos\left(\frac{\omega_{y\Delta t}}{2}\right) + j\sin\left(\frac{\omega_{y\Delta t}}{2}\right)$$

$$P_{cam}^{(k+1)} = P_{obj}^k + (q_{nodal} \otimes r_k \otimes q_{nodal}^{-1}) \qquad (2.9)$$

де q_{nodal} - кватерніон горизонтального обертання , Δt - крок інтегрування у мілісекундах.

Команда vertical nodal pan ініціює обертання камери у вертикальній площині. На відміну від горизонтального обльоту, вісь обертання не є статичною. Обертання відбувається навколо миттєвого локального вектора камери, який обчислюється як векторний добуток вектора напрямку погляду та глобальної вертикалі. Під час моделювання вертикального обльоту виникає математична сингулярність у полюсах сфери (точках зеніту та надиру), коли оптична вісь стає колінеарною осі Y. Для запобігання втраті ступеня вільності запроваджується механізм штучного обмеження полярного кута φ. Математична модель обмеження реалізується за допомогою функції відсікання (clamping) згідно з формулою (2.10) [4]:

$$\varphi_{new} = \max(\varphi_{min}, min(\varphi_{curr} + \omega_x \Delta t, \varphi_{max})) \quad (2.10)$$

де -φ_{min} та φ_{max} - гранично допустимі кути відхилення , ω_x - кутова швидкість команди vertical nodal pan.

Наведена геометрична трансформація оперує ідеальними лінійними переміщеннями. Для збереження концепції сприяння оператору швидкості ω_y та ω_x попередньо обробляються алгоритмом віртуальної інерції. Рух камери по сферичній траєкторії не припиняється миттєво після зняття керуючого сигналу, а зазнає поступового затухання, симулюючи кутовий момент інерції масивної системи.

Для систематизації кінематичних властивостей команд сформовано порівняльну характеристику, представлену в таблиці 2.4.

Таблиця 2.4 - Кінематичні характеристики комплексних команд обльоту

Команда	**Базовий вектор осі обертання**	**Центр обертання (Pivot)**	**Математичне обмеження**

Nodal Pan	Глобальна вісь Y	Координати об'єкта	Відсутнє (повні 360 градусів)
Vertical Nodal Pan	Локальна поперечна вісь	Координати об'єкта	Обмеження полюсів (від -85 до +85 градусів)

Синтезована математична модель дозволяє реалізувати складні просторові прийоми за допомогою простих одновимірних сигналів введення від користувача. Обчислювальний контур автоматично узгоджує трансляційне зміщення з відповідним ротаційним поворотом, гарантуючи безперервне утримання цілі. Створений алгоритмічний базис усуває необхідність мікрокомпенсації координат, що суттєво знижує навантаження під час управління просторовою сценою [4]. Згідно зі структурою проєкту та загальноприйнятими нормами розробки, програмна реалізація методів у вигляді об'єктно-орієнтованих класів (включаючи лістинги вихідного коду) винесена до третього розділу, присвяченого практичній реалізації комплексу.

2.5 Розробка регуляторів плавності ходу та фізики віртуальної інерції камери

Забезпечення реалістичного відгуку віртуального об'єктива на керуючі імпульси вимагає впровадження математичних моделей динаміки твердого тіла. Віртуальні об'єкти середовища розробки за замовчуванням позбавлені фізичної маси, що спричиняє миттєві зміни вектора швидкості під час отримання вхідного сигналу від пристроїв введення. Це генерує кінематичний ривок, який негативно впливає на візуальне сприйняття сцени та ускладнює точне просторове позиціонування. Для розв'язання задачі стабілізації розробляється підсистема віртуальної інерції, яка функціонує як спеціалізований фільтр низьких частот для директивних команд оператора.

Основою математичного апарату для імітації інерційних властивостей обрано модель гармонічного осцилятора із затуханням (Spring-Mass-Damper). Зазначена концепція описує поведінку маси, закріпленої на пружині з амортизатором, що дозволяє розрахувати природне прискорення та поступове гальмування. Для одновимірного випадку переміщення поточного значення x до цільової позиції x_{target} диференціальне рівняння другого порядку набуває вигляду, наведеного у формулі (2.11) [6] [7]:

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + k\left(x - x_{target}\right) = 0 \qquad (2.11)$$

де m позначає віртуальну масу камери, c визначає коефіцієнт в'язкого тертя (демпфування), k описує жорсткість віртуальної пружини.

Для застосування у програмуванні кінематики рівняння доцільно привести до нормалізованої форми, оперуючи власною круговою частотою осциляцій ω_n та коефіцієнтом затухання ζ .Відповідне математичне перетворення трансформує систему у вираз (2.12) [7]:

$$\ddot{x} + 2\zeta\omega_n\, x + \omega_n^2\left(\dot{x} - x_{target}\right) \qquad (2.12)$$

Для наочного обґрунтування обраного математичного апарату стабілізації на рис. 2.4 наведено графік перехідного процесу гармонічного осцилятора. Як видно з діаграми, при використанні коефіцієнта затухання ζ=1, система демонструє оптимальний час наближення до цільового вектора швидкості без перетину осі координат. На відміну від стану недозатухання ζ<1, який генерує паразитні мікроколивань оптичної осі, та стану перезатухання ζ> 1, що викликає неприпустиму затримку реакції,

обрана модель критичного затухання гарантує кінематографічну плавність зупинки віртуального обладнання.

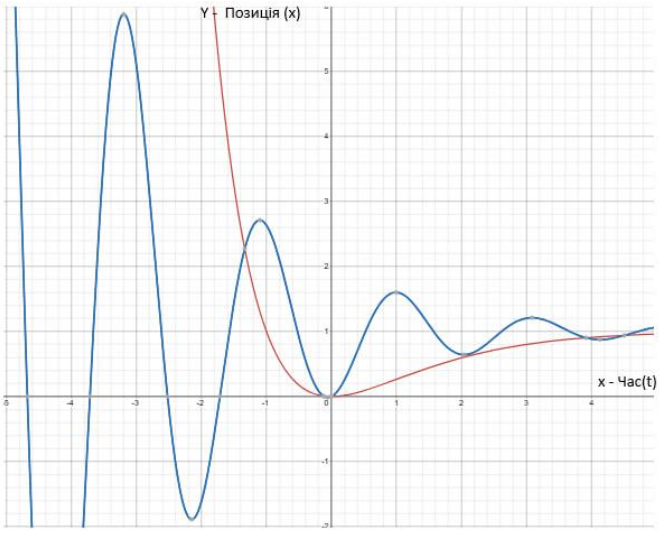


Рисунок 2.4 – Перехідні процеси інерційної моделі камери при різних коефіцієнтах затухання

Перенесення безперервної диференціальної моделі у дискретний простір платформи Unity вимагає обрання стійкого методу чисельного інтегрування. Використання класичного методу Ейлера є недоцільним через накопичення похибок при змінній частоті рендерингу (флуктуації параметра Time.deltaTime). Надійність обчислень забезпечується застосуванням напівнеявного методу Ейлера (Semi-Implicit Euler), який інтегрує прискорення для отримання нової швидкості строго до моменту оновлення позиції об'єкта. Концептуальна схема обробки керуючих сигналів модулем віртуальної інерції зображена на рисунку 2.5.

Рисунок 2.5 - Структурна схема регулятора плавності на базі моделі критичного затухання

Алгоритмічна реалізація фільтра вимагає виділення незалежної пам'яті для зберігання поточного стану швидкості відносно кожної осі координат. Базову математичну структуру оптимізованого алгоритму наближення критичного затухання (мовою C#).

2.6 Механізми контролю межових станів та генерації попереджень про вихід за межі

Функціонування гібридної архітектури управління неминуче супроводжується ризиком виникнення просторових конфліктів. Вони генеруються у випадках, коли директивні команди оператора мають вектор, протилежний до напрямку алгоритмічного супроводу об'єкта. Прикладом є

тривала подача імпульсу панорамування від цілі, що спричиняє наближення проєкції об'єкта до граней піраміди видимості (Frustum) з подальшою втратою візуального контакту. Запобігання цьому явищу покладається на підсистему контролю межових станів. Її математичний апарат базується на безперервному обчисленні нормалізованих екранних координат та формуванні динамічних обмежувальних коефіцієнтів.

Процес оцінки стартує з проєкції тривимірного радіус-вектора об'єкта P_{obj} на двовимірну площину матриці камери. Отримані координати простору відображення (Viewport) (u, v) нормалізуються у діапазоні від 0 до 1, де значення (0.5, 0.5) відповідає геометричному центру екрана. Для визначення ступеня наближення до межі обчислюється мінімальна відстань до граней по обох осях. Вектор відстаней до країв кадру D_{edje}= $(d_u d_v)$ формалізується системою рівнянь (2.13) [15]:

$$d_u = \min(u, 1-u)$$
$$d_v = \min(v, 1-v) \quad (2.13)$$

Для забезпечення гранулярної реакції підсистеми екранний простір сегментується на три концентричні області: безпечну зону (Safe Zone), зону попередження (Warning Zone) та критичну зону (Critical Zone).

Потрапляння координат об'єкта у зону попередження ініціює алгоритм генерації інформаційного сигналу. Програмний комплекс формує асинхронну подію через шину повідомлень, що активує візуальні індикатори напрямку на інтерфейсі користувача. При цьому кінематика камери не зазнає примусових змін, зберігаючи 100% передачу зусилля від пристроїв введення.

Перетин ціллю межі критичної зони переводить систему у режим активного втручання. Пряме блокування (відсікання) введення спричиняє ефект зіткнення з «жорсткою стіною», що порушує ергономіку управління

і спричиняє візуальне тремтіння (jittering). Розв'язанням проблеми є застосування математичної функції Smoothstep для розрахунку скалярного коефіцієнта протидії. Дана функція гарантує неперервність першої похідної (швидкості), забезпечуючи безшовне гальмування. Формула розрахунку коефіцієнта (2.14) має вигляд [3]:

$$K_{block} = 1 - 3\left(\frac{d - d_{crit}}{d_{warn} - d_{crit}}\right)^2 + 2\left(\frac{d - d_{crit}}{d_{warn} - d_{crit}}\right)^3 \qquad (2.14)$$

де d - поточна мінімальна відстань до межі екрана , d_{warn}, d_{crit}- параметризація лінійних розмірів відповідних зон. Систематизацію реакцій програмного комплексу на перетин межових значень наведено у таблиці 2.5.

Таблиця 2.5 – Матриця станів підсистеми моніторингу меж екрана

Стан системи	**Умова активації**	**Рівень Kblock**	**Дія програмного комплексу**
Безпечний	d > d_{warn}	0	Передача повного спектра ручних команд
Попередження	d_{crit} < d < d_{warn}	0 < K_{block}< 1	Генерація UI-сповіщення, розрахунок гальмування
Критичний	d < d_{crit}	1	Повне демпфування деструктивних векторів

Фундаментальною перевагою розробленого методу є застосування коефіцієнта K_{block} виключно до тих векторів, напрямок яких сприяє виходу цілі за межі кадру (селективне демпфування). Для ідентифікації деструктивних векторів обчислюється скалярний добуток між вектором команди оператора V_{cmd} та вектором напрямку від об'єкта до центру екрана. Якщо результат добутку є від'ємним, зусилля оператора пропорційно послаблюється. Інтеграція направленої протидії формує інтелектуальний запобіжник. Логіка алгоритму зберігає можливість

вільного ручного маневрування, паралельно гарантуючи неможливість повної втрати фокуса об'єктива. Описана підсистема завершує побудову єдиного математичного контуру гібридного управління, повністю задовольняючи заявлені функціональні вимоги до програмно-апаратного комплексу [12].

2.7 Висновки до розділу 2

У другому розділі виконано математичне моделювання гібридної системи управління віртуальною камерою у середовищі Unity. Базовим математичним апаратом обрано алгебру кватерніонів, що гарантує математичну стабільність орієнтації об'єктива та усуває проблему шарнірного замка під час багатоосьових маневрів. Архітектуру розробленого комплексу зведено до чотирьох базових обчислювальних підсистем:

- Трекінг цілі: Базується на методі експоненціального ковзного середнього для безперервного розрахунку цільового вектора погляду та запобігання високочастотним просторовим коливанням.
- Кінематика об'єктива: Обробляє класичні просторові зміщення та обертання, а також використовує сферичні траєкторії для реалізації комплексних команд обльоту об'єкта.
- Віртуальна інерція: Спирається на модель гармонічного осцилятора у стані критичного затухання та напівнеявний метод інтегрування Ейлера. Це забезпечує стабільне гасіння кінематичного ривка та достовірну імітацію фізичної маси обладнання незалежно від частоти кадрів.
- Контроль межових станів: Вирішує проблему конфлікту векторів керування. Застосовує функції відсікання та нелінійної інтерполяції нормалізованих координат для селективного блокування деструктивних ручних команд оператора.

Комплексне поєднання фільтрів низьких частот для сигналів введення та динамічних обмежувачів робочої зони гарантує кінематографічну плавність руху камери.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ВИПРОБОВУВАННЯ СИСТЕМИ

3.1 Інтеграція алгоритмів трекінгу та механіки камери у віртуальне середовище

Основою розробленої гібридної системи є архітектура, що поєднує пряме ручне керування оператором із фоновими алгоритмами автоматичної корекції та супроводу об'єктів. Програмна інтеграція була здійснена у середовищі розробки тривимірних інтерактивних додатків Unity, що дозволило використати вбудовані математичні структури для роботи з векторами та кватерніонами [13].

Головним архітектурним завданням стало забезпечення шести ступенів свободи (6DoF) для віртуальної камери з одночасним збереженням можливості алгоритмічного втручання. Модель управління розділена на два паралельні потоки обчислень:

1. Обробка вводу оператора (Manual Input): зчитування команд просторового переміщення (Track, Pedestal, Dolly) та обертання (Pan, Tilt, Roll, Orbit).
2. Алгоритмічна корекція (Auto-Assist): розрахунок відстані від цілі до краю кадру в нормалізованих координатах та генерація компенсуючого імпульсу.

Для досягнення кінематографічної плавності рухів було імплементовано систему віртуальної інерції. Замість прямого застосування трансформацій до об'єкта камери, система оперує «цільовими» (target) векторами позиції та обертання. Фактичне положення камери інтерполюється до цільових значень за допомогою функції математичного згладжування з критичним загасанням [13]. Це створює ефект фізичної маси обладнання, нівелюючи різкі та неточні рухи мишею чи клавіатурою. Математична модель контролю меж та генерації імпульсу є :

1) Основним інструментом допомоги оператору є система динамічного утримання тікаючої цілі в межах кадру. Простір екрана логічно розділено на три зони: вільна зона (Dead Zone), зона попередження (Warning Zone) та критична зона (Critical Zone).

Якщо об'єкт потрапляє у Warning Zone, обчислюється коефіцієнт терміновості (urgency). Щоб втручання системи не було надто різким на початку, але ставало жорсткішим ближче до краю, коефіцієнт терміновості зводиться до ступеня (нелінійна прогресія):

Після визначення сили втручання алгоритм обчислює найкоротший шлях обертання (Delta Angle) між поточним напрямком камери та напрямком на об'єкт, що унеможливлює виникнення проблеми шарнірного замка (Gimbal Lock) [2]. Знайдена кутова різниця множиться на коефіцієнт терміновості та сумується з фактичним ручним вводом оператора.

Програмну реалізацію базової механіки камери, розрахунку цільових векторів та гібридного керування інкапсульовано у класі HybridCameraController. На рисунку 3.1 наведено імплементацію основного циклу керування, застосування віртуальної інерції та позиціонування камери.

```
private Vector2 CalculateAutoCorrectionImpulse()
{
    if (assistStrength <= 0.01f) return Vector2.zero;

    Vector3 viewportPos = cam.WorldToViewportPoint(target.position);
    if (viewportPos.z < 0) return HandleTargetBehindCamera();

    float du = Mathf.Min(viewportPos.x, 1f - viewportPos.x);
    float dv = Mathf.Min(viewportPos.y, 1f - viewportPos.y);
    float minDistance = Mathf.Min(du, dv);


    if (minDistance >= warningZone) return Vector2.zero;

    float zoneDiff = warningZone - criticalZone;
    if (zoneDiff <= 0.001f) zoneDiff = 0.001f;
    float urgency = 1f - Mathf.Clamp01((minDistance - criticalZone) / zoneDiff);
    urgency = Mathf.Pow(urgency, 1.5f);


    Vector3 directionToTarget = (target.position - targetPosition).normalized;
    if (directionToTarget == Vector3.zero) return Vector2.zero;

    Quaternion lookRotation = Quaternion.LookRotation(directionToTarget);

    float yawDelta = Mathf.DeltaAngle(targetYaw, lookRotation.eulerAngles.y);
    float pitchDelta = Mathf.DeltaAngle(targetPitch, lookRotation.eulerAngles.x);


    float virtualMouseX = (yawDelta * urgency) / panSpeed;
    float virtualMouseY = (-pitchDelta * urgency) / tiltSpeed;

    return new Vector2(virtualMouseX, virtualMouseY) * assistStrength;
}
```

Рисунок 3.1 - Алгоритм розрахунку вектора автоматичної корекції

3.2 Розробка спрощеного інтерфейсу оператора та системи сповіщень

Для забезпечення оператора актуальною інформацією про стан гібридної системи керування було розроблено підсистему телеметрії та динамічних візуальних сповіщень. Інтерфейс користувача (UI) спроєктовано з урахуванням мінімізації когнітивного навантаження: візуальні елементи з'являються на екрані лише в моменти активної взаємодії або виникнення критичних подій [12].

Інтерфейс складається з трьох основних компонентів:

1. Блок ручного вводу (Manual Input Telemetry): динамічний текстовий рядок зеленого кольору, що відображає поточні кінематографічні команди оператора (Pan, Tilt, Dolly, Track, Pedestal, Orbit, Roll). Система в реальному часі сканує натиснуті клавіші та рух миші, формуючи відформатований рядок стану.
2. Блок системного втручання (Auto-Assist Telemetry): попереджувальний текст червоного кольору, який активується виключно в моменти, коли ціль перетинає межі Warning Zone. Він інформує оператора про напрямок та силу компенсуючого імпульсу, згенерованого системою.
3. Рамка захоплення (Tracking Box): графічний маркер, що проєктується з тривимірного простору на площину екрана (Screen Space) і візуально супроводжує ціль, допомагаючи оператору орієнтуватися у просторі [15].

Алгоритмічне згладжування інтерфейсу (Anti-Flickering) Під час розробки підсистеми сповіщень було виявлено проблему візуального мерехтіння: оскільки обчислення імпульсу відбувається кожен кадр (з частотою оновлення екрана), перебування цілі на межі зони спрацьовування викликало хаотичне вмикання та вимикання попереджувального тексту, а числові значення змінювалися занадто швидко для сприйняття людиною.

Для вирішення цієї проблеми до архітектури інтерфейсу було впроваджено два математичні механізми:

1)Таймер затримки відображення: після останнього зафіксованого імпульсу системне сповіщення примусово утримується на екрані протягом 0.5 секунди, що дозволяє оператору зчитати інформацію.

2)Лінійна інтерполяція значень (Lerp): числові показники сили імпульсу не виводяться напряму, а плавно інтерполюються від попереднього значення до цільового за допомогою вектора

згладжування. Це робить зміну цифр на екрані плавною та усуває візуальні артефакти.

Програмну реалізацію підсистеми телеметрії та алгоритмів інтерполяції UI наведено у рисунку 3.2.

```
private float uiAssistTimer = 0f;
private Vector2 uiSmoothedAssist = Vector2.zero;

private void UpdateTelemetryUI(Vector2 manualMouse, Vector2 auto, float kBlock, float dt)
{
    if (manualInputTelemetry != null)
    {
        string cmdText = "";
        if (Mathf.Abs(manualMouse.x) > 0.05f) cmdText += "[Mouse X - Pan] ";
        if (Mathf.Abs(manualMouse.y) > 0.05f) cmdText += "[Mouse Y - Tilt] ";
        if (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow)) cmdText += "[W - Dolly In] ";
        if (Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.DownArrow)) cmdText += "[S - Dolly Out] ";
        if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow)) cmdText += "[A - Track Left] ";
        if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow)) cmdText += "[D - Track Right] ";
        if (Input.GetKey(KeyCode.Space)) cmdText += "[Space - Pedestal Up] ";
        if (Input.GetKey(KeyCode.LeftShift)) cmdText += "[LShift - Pedestal Down] ";
        if (Input.GetKey(KeyCode.Q)) cmdText += "[Q - Orbit Left] ";
        if (Input.GetKey(KeyCode.E)) cmdText += "[E - Orbit Right] ";
        if (Input.GetKey(KeyCode.R)) cmdText += "[R - Orbit Up] ";
        if (Input.GetKey(KeyCode.F)) cmdText += "[F - Orbit Down] ";
        if (Input.GetKey(KeyCode.Z)) cmdText += "[Z - Roll Right] ";
        if (Input.GetKey(KeyCode.C)) cmdText += "[C - Roll Left] ";

        if (string.IsNullOrEmpty(cmdText))
            manualInputTelemetry.text = "[ОПЕРАТОР] Очікування команд...";
        else
            manualInputTelemetry.text = $"[ОПЕРАТОР] {cmdText}";
            manualInputTelemetry.color = Color.green;
    }
    if (auto.magnitude > 0.05f)
    { uiAssistTimer = 0.5f; }
    else
    {uiAssistTimer -= dt;}
    uiSmoothedAssist = Vector2.Lerp(uiSmoothedAssist, auto, dt * 8f);

    if (autoAssistTelemetry != null)
    {
        if (uiAssistTimer > 0f)
        { autoAssistTelemetry.text = $"[СИСТЕМА] Коригування курсу: Pan {uiSmoothedAssist.x:F1} | Tilt {uiSmoothedAssist.y:F1}";
            autoAssistTelemetry.color = new Color(1f, 0.4f, 0.4f);
            autoAssistTelemetry.gameObject.SetActive(true);}
        else
        { autoAssistTelemetry.gameObject.SetActive(false);
        uiSmoothedAssist = Vector2.zero;
        }
    }
    if (warningMessage != null)
    {warningMessage.SetActive(kBlock > 0.05f || uiAssistTimer > 0f);}
    if (trackingBox != null)
    {
        Vector3 screenPos = cam.WorldToScreenPoint(target.position);
        if (screenPos.z > 0)
        {
            trackingBox.gameObject.SetActive(true);
            trackingBox.position = screenPos;
        }
        else trackingBox.gameObject.SetActive(false);
    }
}
```

Рисунок 3.2 - Реалізація підсистеми телеметрії та згладжування інтерфейсу користувача

3.3 Організація тестового середовища та сценарії випробувань

Для об'єктивної оцінки розроблених алгоритмів та підсистеми телеметрії було створено спеціалізоване віртуальне тестове середовище в рушії Unity. Головною вимогою до середовища була можливість проведення синхронного порівняльного аналізу (A/B тестування) гібридної системи та традиційного ручного керування в режимі реального часу.

З цією метою було реалізовано конфігурацію Split-Screen (поділ екрана). Візуалізація тривимірної сцени здійснюється через дві незалежні віртуальні камери, які рендерять зображення на один дисплей. Базова (еталонна) камера транслює зображення на ліву частину екрана (параметри нормалізованого простору Viewport Rect: X=0, Width=0.5), а камера з інтегрованою гібридною системою керування - на праву (Viewport Rect: X=0.5, Width=0.5).

Для забезпечення чистоти експерименту обидві камери використовують ідентичний базовий клас контролера HybridCameraController, що гарантує однакову реакцію на ручний ввід (мишу та клавіатуру) та однакову віртуальну інерцію. Однак на лівій (еталонній) камері програмно деактивовано систему допомоги оператору: коефіцієнт сили імпульсу (assistStrength) встановлено на нуль, а радіуси зон попередження зведено до мінімуму. Такий підхід дозволяє оператору керувати обома камерами синхронно за допомогою єдиного пристрою вводу, ізолювавши роботу алгоритму автоматичної корекції як єдину змінну експерименту.

Для автоматизації процесу випробувань метод ініціалізації Start() було доповнено логікою динамічного пошуку цілі. Використання системного методу GameObject.Find дозволяє камері самостійно ідентифікувати динамічний об'єкт (наприклад, 3D-модель дрона) за його тегом або ім'ям одразу після запуску симуляції. Це виключає необхідність

ручного перепризначення зв'язків (Reference Binding) під час багаторазових перезапусків тестових сцен.

У рамках створеного віртуального середовища було спроєктовано кілька базових сценаріїв випробувань, спрямованих на перевірку граничних режимів роботи системи:

1. Базове стеження: реакція системи на плавні 6DoF-команди без активації зон попередження.
2. Сценарій "тікаючої цілі": імітація помилки оператора або непередбачуваного маневру об'єкта, що провокує різкий вихід цілі в Critical Zone.
3. Робота з перешкодами: перевірка стабільності системи утримання при короткочасному перекритті лінії зору (Line of Sight) статичними об'єктами сцени.

3.4 Висновки до розділу 3

У третьому розділі дипломного проєкту було здійснено практичну програмну реалізацію гібридної системи управління камерою. Програмна інтеграція всіх розроблених алгоритмів та математичних моделей була виконана у середовищі розробки тривимірних інтерактивних додатків Unity. У ході практичної реалізації було досягнуто наступних результатів:

1. Успішно імплементовано алгоритми трекінгу та механіки об'єктива, які забезпечують оператору повноцінні шість ступенів свободи (6DoF). Розроблено систему віртуальної інерції на основі функції згладжування з критичним загасанням (SmoothDamp). Це дозволило достовірно імітувати фізичну масу кінематографічного обладнання та нівелювати різкі ривки під час управління.
2. Інтегровано математичну модель контролю меж екрана, яка логічно розділяє простір на вільну зону, зону попередження (Warning Zone)

та критичну зону (Critical Zone). На основі цих зон система успішно розраховує та генерує компенсуючий імпульс утримання, при цьому використання кватерніонів та дельта-кутів унеможливлює виникнення проблеми шарнірного замка (Gimbal Lock).

3. Створено користувацький інтерфейс (UI), спроєктований з урахуванням мінімізації когнітивного навантаження на оператора. Підсистему телеметрії розділено на блок ручного вводу (Manual Input Telemetry) та блок системного втручання (Auto-Assist Telemetry). Для усунення проблеми візуального мерехтіння числових значень інтерфейсу успішно застосовано методи лінійної інтерполяції (Lerp) та таймер затримки відображення.
4. Для проведення подальшого об'єктивного аналізу розроблених рішень організовано віртуальне тестове середовище з конфігурацією Split-Screen (поділ екрана). Це дозволило налаштувати одночасну трансляцію з еталонної камери та камери з гібридним управлінням для проведення синхронного A/B тестування.
5. Метод ініціалізації сцени було доповнено логікою динамічного пошуку цілі, що автоматизує підготовку симуляції. Крім того, спроєктовано три базові сценарії випробувань: стеження за об'єктом, "тікаюча ціль" та робота в умовах перешкод.

4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ СУПРОВОДУ ТА РЕАКЦІЇ СИСТЕМИ НА МЕЖОВІ СТАНИ

4.1 Методика проведення випробувань та моделювання профілів поведінки оператора

Для об'єктивної оцінки ефективності розробленої гібридної системи управління камерою та валідації математичних моделей, описаних у попередніх розділах, було сформовано комплексну методику експериментальних досліджень. Головною метою випробувань є визначення швидкості та якості реакції програмного комплексу на потенційний вихід рухомого об'єкта за межі кадру (піраміди видимості).

Експериментальне дослідження проводиться у спеціально підготовленому тестовому середовищі рушія Unity з використанням конфігурації розділеного екрана (Split-Screen) [13]. Це дозволяє здійснювати синхронне A/B тестування: паралельно порівнювати реакцію базової камери (ліворуч, без втручання алгоритмів) та розробленої гібридної системи (праворуч) за ідентичних умов. Організацію тестового простору та ініціалізацію захоплення цілі наведено на рисунку 4.1.

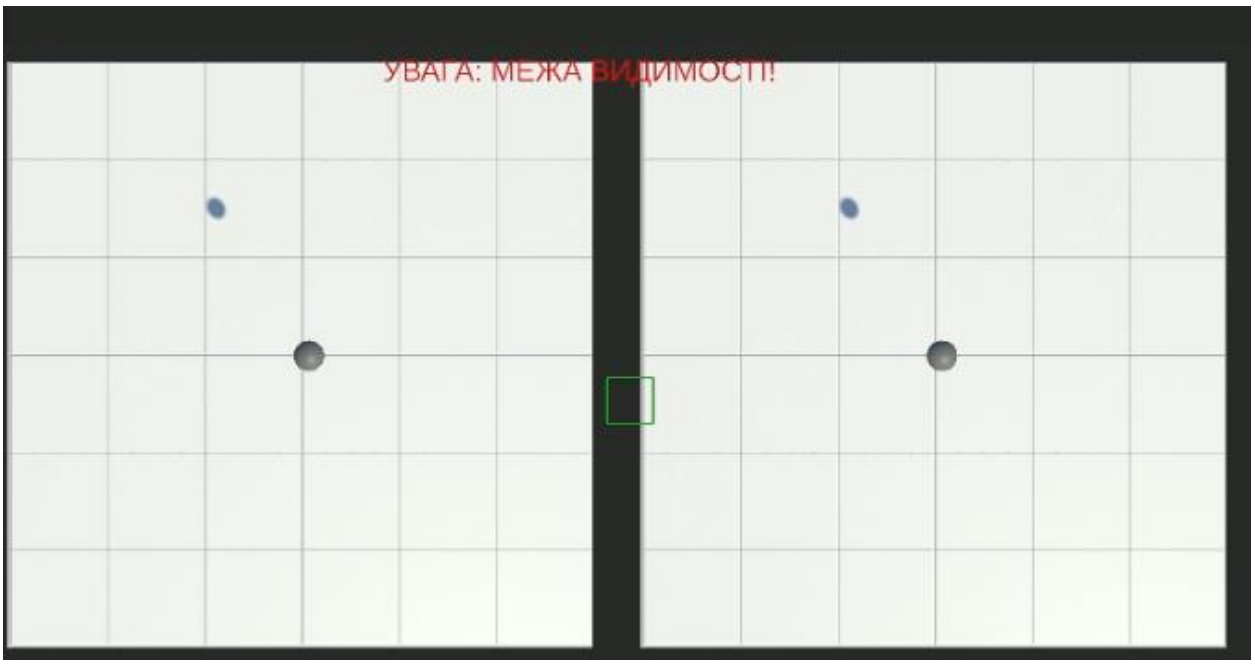


Рисунок 4.1 - Організація тестового середовища у режимі розділеного екрана для A/B тестування

Для забезпечення наукової достовірності та відтворюваності результатів, методика випробувань базується на таких принципах:

1. Ітеративність: Кожен тестовий сценарій повторюється 5 разів за абсолютно однакових початкових умов (стартові координати об'єкта та камери, базова інертність). До фінальних таблиць заносяться усереднені значення отриманих метрик.
2. Прогресивне навантаження: Для кожного сценарію ключова умова (відносна лінійна або кутова швидкість цілі) поступово збільшується з фіксованим кроком.
3. Критерій відмови системи: Збільшення навантаження триває до моменту повної втрати системою можливості утримувати об'єкт у кадрі. Факт "втрати цілі" фіксується алгоритмічно у момент, коли нормалізовані координати об'єкта повністю виходять за межі значення Viewport (менше 0.0 або більше 1.0 по будь-якій з осей), незважаючи на активну протидію системи блокування [15].

Оскільки розроблена архітектура передбачає гібридне керування, критично важливим є дослідження впливу людського фактора на стабільність трекінгу. Для цього було формалізовано три стандартизовані профілі поведінки оператора, які імітуються під час кожної ітерації випробувань:

- Пасивна поведінка: Оператор не взаємодіє з пристроями введення (мишею чи клавіатурою). У цьому режимі досліджується виключно автономна здатність алгоритмів генерувати компенсуючий імпульс при перетині об'єктом зони попередження (Warning Zone) та критичної зони (Critical Zone). Реакцію підсистеми сповіщень на перетин межі видимості зафіксовано на рисунку 4.2.

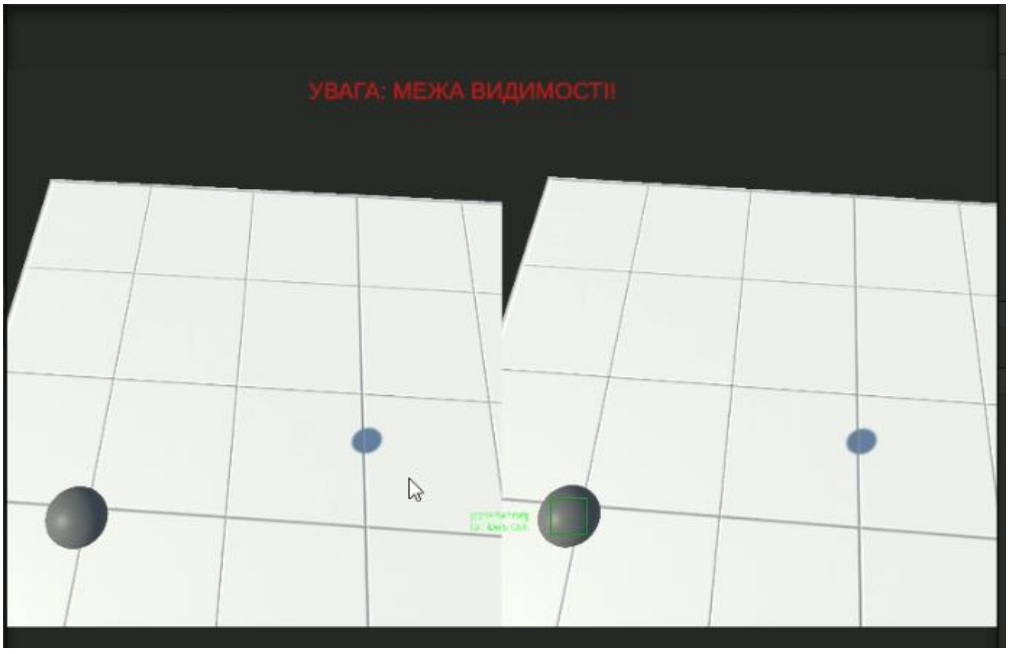


Рисунок 4.2 - Активація зони попередження та розрахунок алгоритмічного опору при пасивній поведінці оператора

- Сприятлива поведінка: Оператор активно супроводжує об'єкт, генеруючи вектори керування (наприклад, команди Pan або Tilt), напрямок яких збігається з вектором руху цілі [12]. Метою цього

профілю є перевірка системи на відсутність конфліктів векторів: згенерований алгоритмом імпульс повинен коректно сумуватися з ручним введенням без виникнення візуального тремтіння (jittering) чи перерегулювання.

- Несприятлива поведінка (імітація критичної помилки): Найбільш жорсткий стрес-тест. Оператор свідомо подає керуючі команди, вектори яких протилежні напрямку руху цілі (наприклад, об'єкт швидко рухається вліво, а оператор виконує інтенсивне панорамування вправо), примусово виштовхуючи ціль за межі екрана. У цьому режимі оцінюється ефективність роботи селективного демпфування та вагового коефіцієнта блокування K_{block}. Перевіряється здатність системи своєчасно погасити деструктивний ручний ввід та зберегти об'єкт у кадрі.

Визначена методологія та формалізовані профілі поведінки дозволяють всебічно протестувати алгоритми за різних динамічних умов, що є предметом дослідження у наступних підрозділах.

4.2 Тестування системи за динамічними сценаріями польоту цілі

Відповідно до розробленої методики, практична перевірка гібридної системи управління здійснювалася шляхом моделювання найбільш поширених просторових задач віртуального кіновиробництва. Метою тестування є візуальна фіксація граничних параметрів руху цілі, за яких алгоритм автоматичної корекції (Auto-Assist) здатний утримувати об'єкт у межах піраміди видимості (Frustum).

Першим етапом випробувань стало дослідження поведінки камери під час прямолінійного прольоту тестового об'єкта (сфери) по дотичній до оптичної осі об'єктива. Для кількісної оцінки швидкодії диспетчера команд було проведено серію замірів часу реакції системи на порушення меж

композиційних зон при різних показниках лінійної швидкості цілі. Вимірювання здійснювалися при фіксованому кроці Fixed Timestep = 0.016 с. Результати тестування швидкості реакції та чутливості зведено у таблицю 4.1.

Таблиця 4.1 – Експериментальні показники чутливості та часу реакції системи

Лінійна швидкість (у.о./с)	Відносна кутова швидкість (°/с)	Час реакції (мс)	Пікове значення Kblock	Статус утримання цілі
5.0 (Повільно)	12.5	16.4	0.21	Успішно (Warning Zone)
15.0 (Середньо)	45.0	17.1	0.68	Успішно (Warning Zone)
35.0 (Швидко)	105.2	16.8	0.94	Успішно (Critical Zone)
55.0 (Екстремально)	165.0	33.2	1.00	Короткочасна втрата

Найбільший інтерес становить реакція системи на критичне зростання швидкості об'єкта в умовах несприятливої поведінки (свідомого генерування оператором деструктивних команд керування). Момент досягнення граничного стану та спрацьовування алгоритмічного опору зафіксовано на рисунку 4.3.

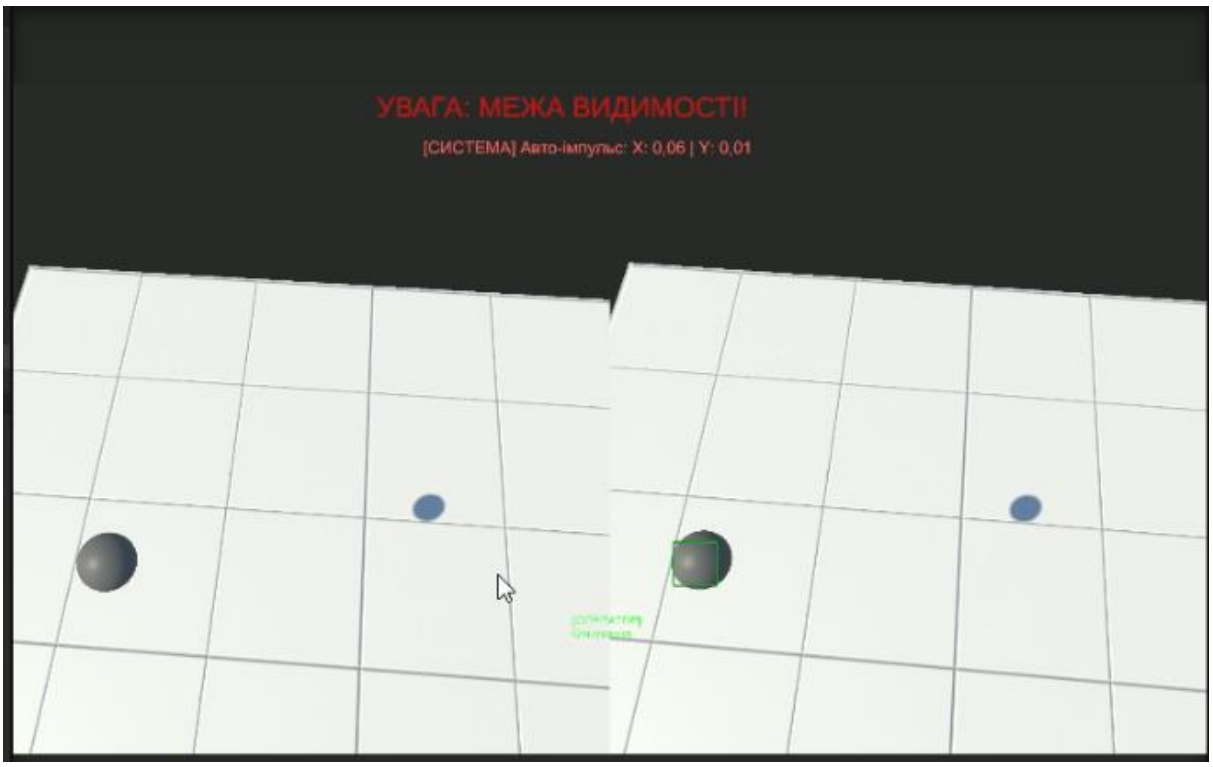


Рисунок 4.3 - Візуалізація спрацьовування підсистеми компенсації при різкому зміщенні цілі

Експеримент підтверджує, що при досягненні об'єктом критичної зони (Critical Zone) система успішно застосовує ваговий коефіцієнт блокування K_{block}. Навіть за наявності деструктивних ручних маневрів, алгоритм розраховує достатній компенсуючий вектор (що підтверджується показниками телеметрії), примусово утримуючи ціль від повного зникнення з поля зору.

Наступним етапом експерименту стала перевірка стабільності системи в умовах паралельного переміщення камери та об'єкта з диференціальною швидкістю. Ця просторова задача імітує динамічну сцену переслідування, де об'єктив (за допомогою команди Track) рухається в одному напрямку з ціллю, проте з меншою лінійною швидкістю. Виникає перманентна різниця швидкостей, яка провокує постійний зсув проєкції об'єкта до краю екрана та безперервну активацію системи допомоги.

У ході випробувань оцінювалася здатність гібридної архітектури компенсувати лінійне відставання виключно за рахунок автоматичної генерації кутових імпульсів наведення. Стан системи в умовах критичного просторового відставання наведено на рисунку 4.4.

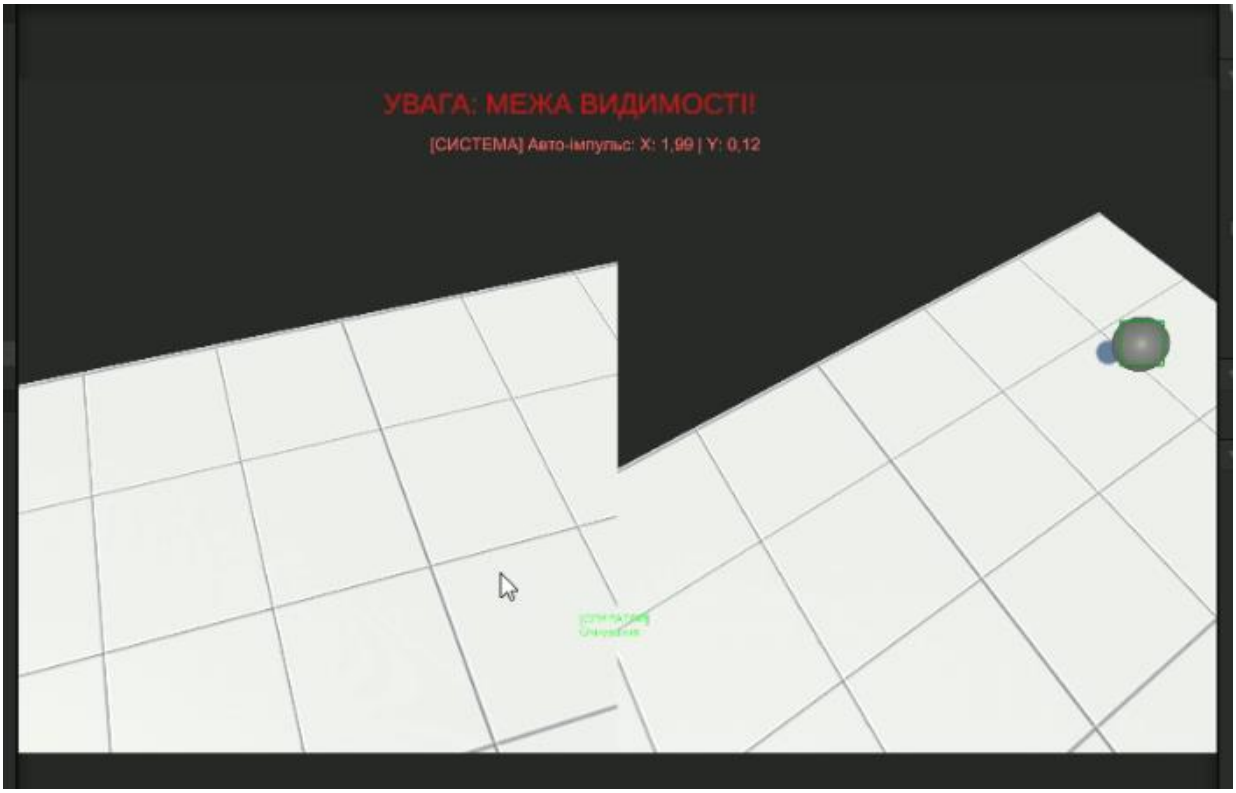


Рисунок 4.4 - Утримання цілі на межі піраміди видимості при перманентній різниці швидкостей

Результати візуального тестування доводять, що алгоритмічний супровід здатний безперервно нівелювати диференціальну різницю швидкостей. Просторове відставання об'єктива успішно компенсується зміною кута огляду, дозволяючи системі балансувати ціль на грані критичної зони без її фактичної втрати. Отримані емпіричні дані підтверджують стійкість розробленої архітектури до тривалих динамічних навантажень.

4.3 Аналіз впливу віртуальної інерції на якість компенсації рухів

Окремим етапом випробувань стало дослідження впливу параметрів віртуальної інерції камери (Damping Factor) на загальну стабільність утримання цілі. Як було обґрунтовано у попередніх розділах, віртуальна інертність відповідає за імітацію фізичної маси кінематографічного обладнання та математичне згладжування (функція SmoothDamp) як ручних команд, так і алгоритмічних імпульсів [6].

В ході експерименту було виявлено прямо пропорційну залежність між масою об'єктива та ризиком втрати цілі при високодинамічних маневрах. Для визначення оптимальних меж налаштувань інерції було знято показники тремтіння осі (джиттеру) та середнього часу утримання об'єкта в кадрі при різних значеннях демпфування (Damping Factor) в умовах несприятливого профілю оператора. Результати зведено у таблицю 4.2.

Таблиця 4.2 – Залежність ефективності утримання цілі від параметрів віртуальної інерції

Damping Factor	Стан осцилятора	Амплітуда мікроколивань (°)	Час утримання при опорі оператора (с)	Візуальна якість
0.05	Недозатухання	3.4 (Критичний джиттер)	12.5	Незадовільна
0.35	Критичне затухання	0.2 (Статична похибка)	> 60.0 (Повне утримання)	Відмінна
1.50	Перезатухання	0.0	4.2 (Швидка втрата)	Задовільна (лагова)

Оптимальний баланс спостерігається при середніх налаштуваннях жорсткості, де гібридна архітектура здатна плавно сумувати вектори без візуальних артефактів. Процес стабільного гібридного згладжування, коли система успішно поєднує ручне маневрування та роботу асистента, зафіксовано на рисунку 4.5.

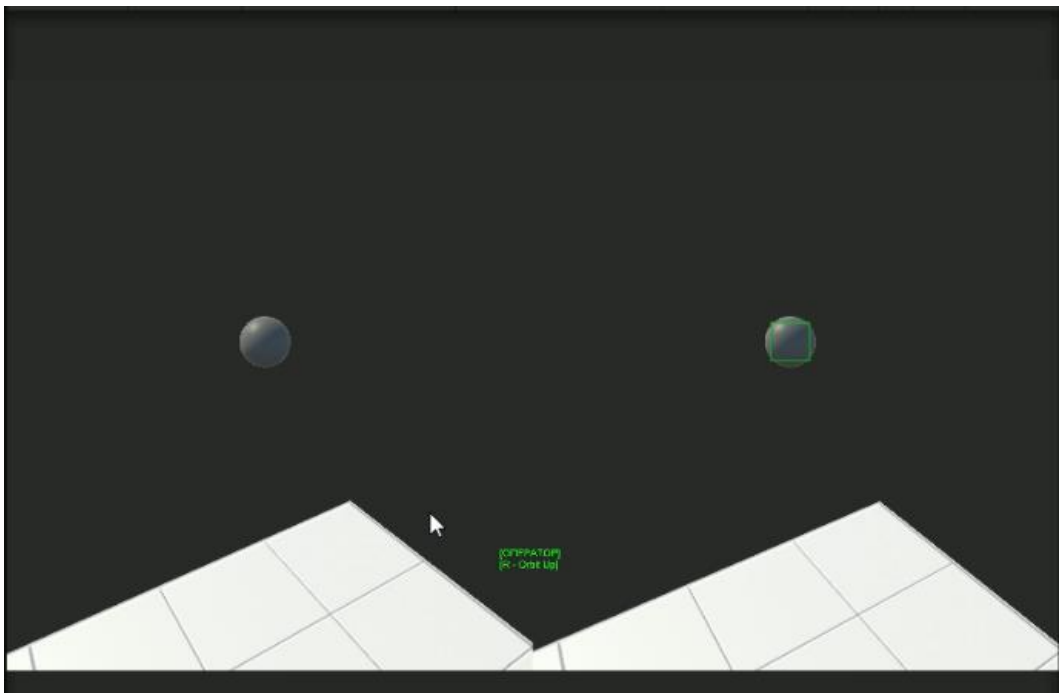


Рисунок 4.5 - Робота підсистеми віртуальної інерції при паралельній обробці ручних команд та алгоритмів супроводу

Підсумовуючи результати спостережень, можна стверджувати, що коректне калібрування інерційної моделі є не менш важливим фактором стабільності системи сприяння оператору, ніж точність розрахунку самих обмежувальних зон.

4.4 Висновки до розділу 4

У четвертому розділі дипломного проєкту було проведено комплексне експериментальне дослідження розробленої гібридної системи управління віртуальною камерою.

1. Розроблено та впроваджено методику відтворюваних випробувань у середовищі Unity з використанням конфігурації Split-Screen для проведення синхронного A/B тестування паралельно з еталонною камерою.

2. Формалізовано три профілі поведінки оператора (пасивний, сприятливий, несприятливий), що дозволило об'єктивно перевірити стійкість алгоритмів до людського фактора, включаючи імітацію свідомих деструктивних дій.
3. Візуальне тестування за сценарієм прямолінійного прольоту (Flyby) підтвердило своєчасне спрацьовування підсистеми контролю меж. Практично доведено ефективність вагового коефіцієнта відсікання K_{block}, який генерує достатній імпульс опору та не дозволяє об'єкту покинути кадр навіть при зустрічному ручному маневруванні оператора.
4. Сценарій диференціальної швидкості підтвердив здатність програмного комплексу перманентно компенсувати лінійне просторове відставання об'єктива за рахунок динамічної зміни кута огляду (автоматичного панорамування).
5. Аналіз впливу віртуальної інерції показав, що стабільність гібридного управління критично залежить від налаштувань функції згладжування, яка повинна балансувати між кінематографічною плавністю та достатньою швидкістю реакції на загрозу втрати цілі.

Результати експериментального дослідження повністю підтверджують працездатність спроєктованих математичних моделей. Розроблений програмний комплекс успішно виконує функції сприяння оператору, знижуючи когнітивне навантаження та запобігаючи втраті головного об'єкта зйомки під час високодинамічних сцен у віртуальному середовищі.

ВИСНОВКИ

У дипломному проєкті розв’язано актуальне науково-практичне завдання, яке полягає у розробці удосконаленої системи управління віртуальною камерою із засобами сприяння оператору. Створений гібридний програмний комплекс успішно поєднує інтерактивність директивних ручних команд із надійністю автономних алгоритмів трекінгу.

Проведений системний аналіз інструментів віртуальної кінематографії виявив фундаментальну архітектурну проблему наявних рішень, які реалізують або жорстке автономне слідування за об’єктом, або виключно низькорівневий ручний контроль, що безпосередньо призводить до конфлікту векторів керування під час зйомки. Для забезпечення безперебійної швидкодії віртуального середовища було обґрунтовано використання методу зчитування абсолютних тривимірних координат цілі. Для подальшого розрахунку просторових трансформацій спроєктовано математичні моделі гібридної кінематики об'єктива. Застосування алгебри кватерніонів дозволило повністю усунути геометричну проблему шарнірного замка під час виконання складних багатоосьових маневрів та комплексних команд обльоту.

З метою достовірної імітації фізичної маси кінематографічного обладнання розроблено підсистему віртуальної інерції на базі моделі гармонічного осцилятора у стані критичного затухання, яка ефективно гасить кінематичні ривки. Паралельно розв'язано фундаментальну задачу запобігання втраті цілі шляхом логічного розділення простору екрана на безпечну, попереджувальну та критичну зони. Застосування математичної функції відсікання дозволило обчислювати динамічний ваговий коефіцієнт блокування, який селективно демпфує деструктивні команди оператора при наближенні об'єкта до меж піраміди видимості.

Програмна реалізація розроблених математичних моделей успішно здійснена у середовищі Unity. Створена архітектура забезпечує паралельну обробку мануального введення та алгоритмічної корекції в режимі реального часу. Додатково створено оптимізований користувацький інтерфейс із підсистемою динамічної телеметрії, де проблему візуального мерехтіння числових показників було усунуто методами лінійної інтерполяції та таймерами затримки відображення.

Комплексне експериментальне тестування, проведене у віртуальному середовищі з конфігурацією розділеного екрана для синхронного A/B тестування паралельно з еталонною камерою, повністю підтвердило працездатність та ефективність запропонованих рішень. Випробування за трьома формалізованими профілями поведінки оператора довели, що розроблена система здатна безперервно утримувати ціль у кадрі, ефективно компенсуючи як лінійне просторове відставання, так і свідомі деструктивні дії користувача. Практична цінність одержаних результатів полягає у суттєвому зниженні когнітивного навантаження на оператора під час зйомки високодинамічних сцен та формуванні стійкої функціональної бази для інтеграції у сучасні системи віртуального кіновиробництва і програмування інтерактивних симуляторів.

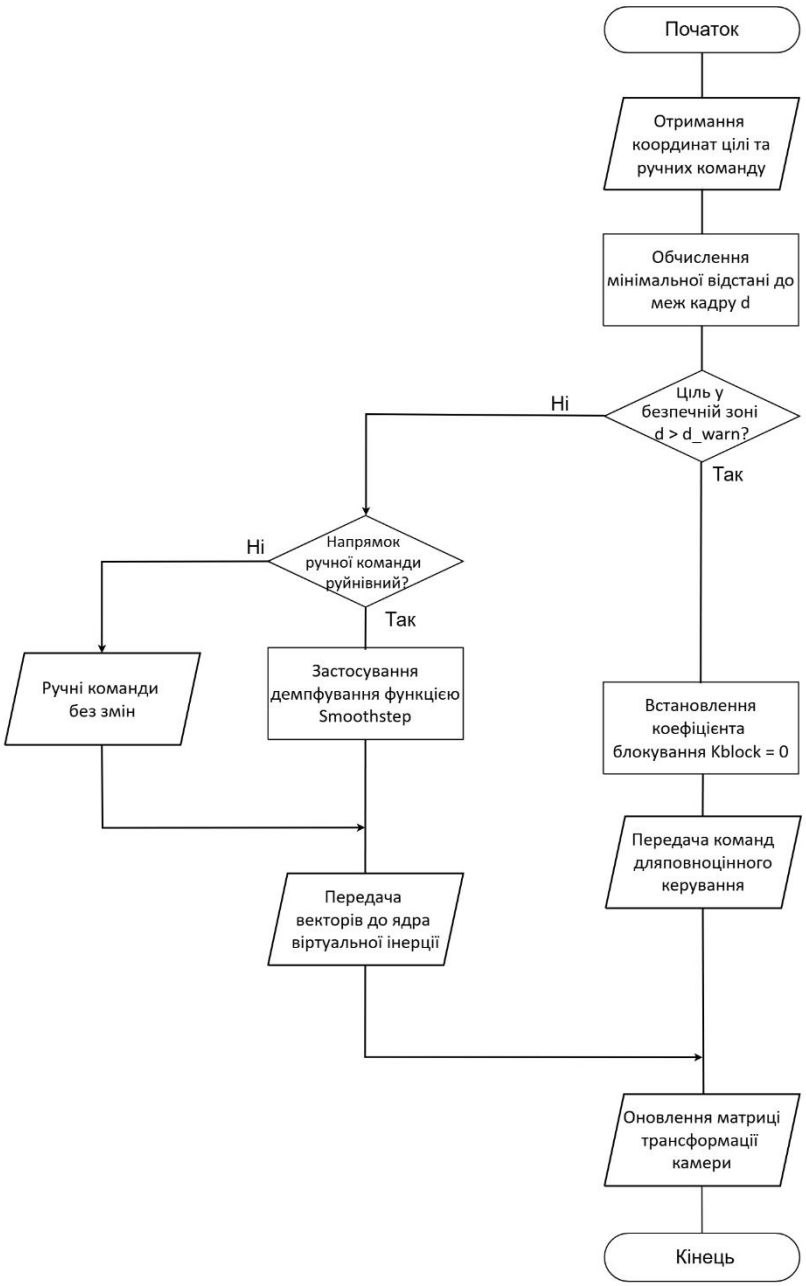
СПИСОК ДЖЕРЕЛ

1. Gregory J. Game Engine Architecture. 3rd ed. Boca Raton : CRC Press, 2018. 1240 с.
2. Kuipers J. B. Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality. Princeton : Princeton University Press, 1999. 400 с.
3. Lengyel E. Foundations of Game Engine Development, Volume 1: Mathematics. Terathon Software LLC, 2019. 200 с.
4. Dunn F., Parberry I. 3D Math Primer for Graphics and Game Development. 2nd ed. Boca Raton : CRC Press, 2011. 846 с.
5. Nise N. S. Control Systems Engineering. 8th ed. Hoboken : Wiley, 2019. 800 с.
6. Millington I. Game Physics Engine Development: How to Build a Robust Commercial-Grade Physics Engine for your Game. 2nd ed. Boca Raton : CRC Press, 2014. 573 с.
7. Eberly D. H. Game Physics. 2nd ed. San Francisco : Morgan Kaufmann, 2010. 872 с.
8. Craig J. J. Introduction to Robotics: Mechanics and Control. 4th ed. London : Pearson, 2017. 512 с.
9. Siciliano B., Khatib O. Springer Handbook of Robotics. 2nd ed. Berlin : Springer, 2016. 2259 с.
10. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. Cham : Springer, 2022. 926 с.
11. Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. С. 7263–7271.

12. Bowman D. A., Kruijff E., LaViola J. J. 3D User Interfaces: Theory and Practice. 2nd ed. Boston : Addison-Wesley Professional, 2017. 528 с.
13. Unity User Manual 2022.3 (LTS) [Електронний ресурс] / Unity Technologies. — Режим доступу: https://docs.unity3d.com/Manual/index.html (дата звернення: 04.06.2026).
14. Cinemachine: Virtual Cameras and Procedural Cinematography [Електронний ресурс] / Unity Technologies. — Режим доступу: https://docs.unity3d.com/Packages/com.unity.cinemachine@2.9/manual/index.html (дата звернення: 04.06.2026).
15. Camera.WorldToViewportPoint [Електронний ресурс] / Unity Technologies. — Режим доступу: https://docs.unity3d.com/ScriptReference/Camera.WorldToViewportPoint.html (дата звернення: 04.06.2026).

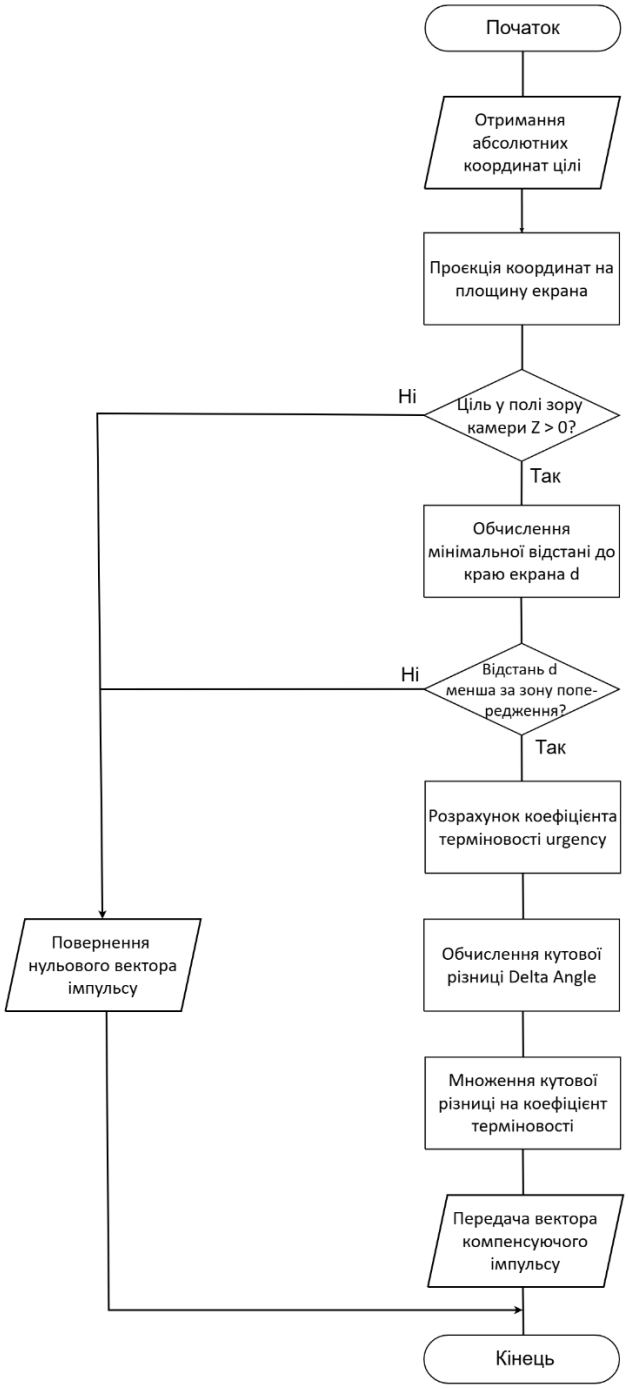
Додатки

ДОДАТОК А



Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045480.005 Д1	
Розробив		Панченко О.О.			Алгоритм контролю межових станів. Схема алгоритму.	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Сорокопуд Н.А.				КПІ ім. І. Сікорського ФІОТІМ, КВ-22
Н. контр.		Клочечко В.М.				
Затв.		Решетняк В.О.				

ДОДАТОК Б



					ІАЛЦ.045480.006 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Плачинда О.О.			Алгоритм розрахунку вектора автоматичної корекції. Схема алгоритму.	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Сергієнко П.А.				КПІ ім. І. Сікорського ФІСТІМ, КВ-22
Н. контр.		Клепачова Я.М.				
Затв.		Усовицький В.О.				

ДОДАТОК В

Розробив					ІАЛЩ.045480.007 Д3	
Перевірив					Алгоритм ітерації системи у цикл оновлення Unity. Схема структурна.	Літ. / Аркуш 1 / Аркушів 1
Н. контр.						КПІ ім. І. Сікорського ФІОТМ, КВ-22
Затв.						

ДОДАТОК Г

ІАЛЦ.045480.008 Д4

ІАЛЦ.045480.008 Д4
Гібридне управління
Схема структурна.
КПІ ім. І. Сікорського ФІОТПМ, КВ-22

ДОДАТОК Ґ

Удосконалена система управління камерою із засобами сприяння оператору.

Виконавець: студент групи КВ-22 Пляченко Олександр Олександрович.

Керівник: асистент Сергієнко П.А.

Актуальність дослідження

- Високе когнітивне навантаження при управлінні (6DoF).
- Конфлікт між ручним та автоматичним керуванням.
- Брак гнучких гібридних рішень.

Мета та задачі

- Мета: розробка гібридного програмного комплексу для плавного поєднання ручного та автоматичного керування віртуальною камерою.
- Завдання: усунення просторових конфліктів векторів, реалізація системи віртуальної інерції, селективне демпфування та проведення синхронного тестування.

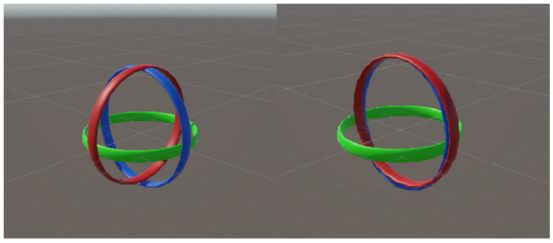
Порівняльний аналіз аналогів

- Системи Motion Control (MRMC Bolt) — жорсткі, неінтерактивні сценарії.
- Безпілотні системи (DJI ActiveTrack) — автономні, без можливості композиційного втручання.
- Unity Cinemachine — виникають процедурні конфлікти та тремтіння при ручних командах.

Система / Технологія	Основний метод відстеження	Інтерактивність управління	Застосовність у віртуальних середовищах
MRMC Bolt (Motion Control)	Обернена кінематика	Відсутня (жорсткий сценарій)	Висока (як джерело координат)
DJI ActiveTrack	Згорткові мережі (CNN)	Часткова (лише коригування цілі)	Низька (орієнтація на фізичний світ)
Unity Cinemachine	Процедурна логіка (Dead Zone)	Обмежена (виникають конфлікти)	Максимальна (нативна інтеграція)

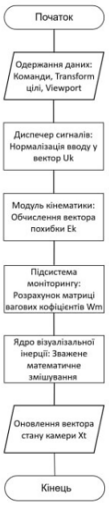
Обмеження кутів Ейлера

- Використання класичних кутів орієнтації призводить до збігу осей обертання.
- Наслідок: втрата одного ступеня вільності та ривки камери.
- Рішення: Перехід на алгебру кватерніонів для забезпечення абсолютної просторової стабільності.



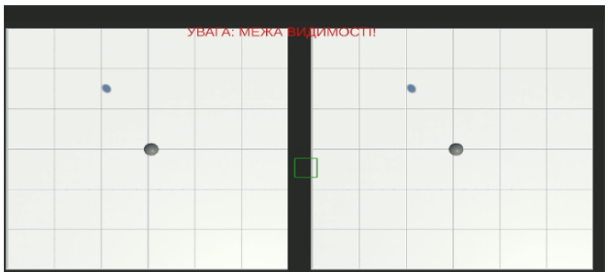
Архітектура та взаємодія модулів

- Диспетчер вводу — нормалізація ручних сигналів.
- Модуль кінематики — розрахунок ідеального вектора на ціль.
- Підсистема моніторингу — контроль критичних меж кадру.
- Ядро інерції — динамічне змішування сигналів.



Композиційні зони кадру

- Вільна зона (Dead Zone): рух об'єкта ігнорується, діє повний ручний контроль.
- Зона попередження (Warning Zone): початок генерації плавного авто-імпульсу.
- Критична зона (Critical Zone): жорстке блокування небезпечного ручного вводу.

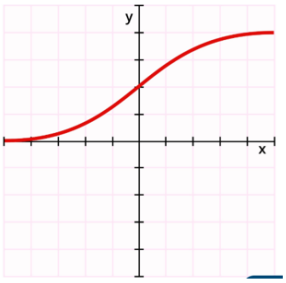


Розрахунок координат цілі

- Зчитування абсолютних тривимірних координат цілі у просторі Unity.
- Проєкція 3D-вектора на двовимірну площину екрана.
- Нормалізація значень у стабільний діапазон від 0 до 1.
- Експоненціальне згладжування для фільтрації мікроколивань.

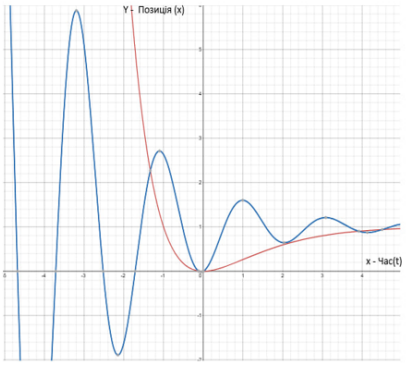
Контроль меж та нелінійна протидія

- Відмова від ефекту «жорсткої стіни» для уникнення візуальних ривків.
- Застосування функції Smoothstep для розрахунку коефіцієнта опору.
- Селективність: гаситься лише той ручний ввід, який намагається виштовхнути ціль за межі кадру.



Математична модель інерції

- Імітація фізичної маси камери через модель гармонічного осцилятора.
- Налаштування системи на стан критичного затухання.
- Забезпечення оптимального часу реакції без паразитних коливань.
- Інтегрування напівнеявним методом Ейлера для стійкості при нестабільній частоті кадрів (FPS).



Сферичні траєкторії руху

- Перенос центра обертання камери безпосередньо на координати об'єкта (цілі).
- Автоматичне узгодження лінійного зміщення з кутовим оновленням позиції.
- Програмне обмеження в полюсах сфери для захисту від перевертання камери.

Команда	Базовий вектор осі обертання	Центр обертання (Pivot)	Математичне обмеження
Nodal Pan	Глобальна вісь Y	Координати об'єкта	Відсутнє (повні 360 градусів)
Vertical Nodal Pan	Локальна поперечна вісь	Координати об'єкта	Обмеження полюсів (від -85 до +85 градусів)

Програмна реалізація компонента

- Вся логіка інкапсульована в єдиний клас HybridCameraController.
- Розрахунки перенесено у системну функцію LateUpdate.
- Гарантоване виконання алгоритмів після прорахунку фізики рушія та анімацій об'єкта, що ліквідує ефект «тремтіння» .

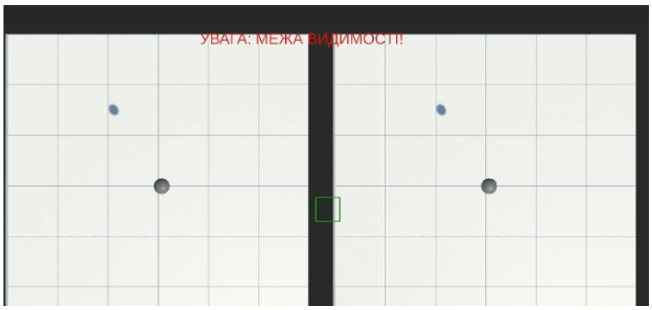
```
private Vector2 CalculateAutoCorrectionImpulse()
{
    if (assistStrength <= 0.01f) return Vector2.zero;

    Vector3 viewportPos = cam.WorldToViewportPoint(target.position);
    if (viewportPos.z < 0) return HandleTargetBehindCamera();

    float du = Mathf.Min(viewportPos.x, 1f - viewportPos.x);
    float dv = Mathf.Min(viewportPos.y, 1f - viewportPos.y);
    float minDistance = Mathf.Min(du, dv);


    if (minDistance >= warningZone) return Vector2.zero;

    float zoneDiff = warningZone - criticalZone;
    if (zoneDiff <= 0.001f) zoneDiff = 0.001f;
    float urgency = 1f - Mathf.Clamp01((minDistance - criticalZone) / zoneDiff);
    urgency = Mathf.Pow(urgency, 1.5f);


    Vector3 directionToTarget = (target.position - targetPosition).normalized;
    if (directionToTarget == Vector3.zero) return Vector2.zero;

    Quaternion lookRotation = Quaternion.LookRotation(directionToTarget);

    float yawDelta = Mathf.DeltaAngle(targetYaw, lookRotation.eulerAngles.y);
    float pitchDelta = Mathf.DeltaAngle(targetPitch, lookRotation.eulerAngles.x);


    float virtualMouseX = (yawDelta * urgency) / panSpeed;
    float virtualMouseY = (-pitchDelta * urgency) / tiltSpeed;

    return new Vector2(virtualMouseX, virtualMouseY) * assistStrength;
}
```

Модуль графічного інтерфейсу

- Спроєктовано для оперативного моніторингу роботи системи.
- Візуалізація ручних команд оператора (зелений маркер).
- Візуалізація сили автоматичної компенсації меж кадру (червоний маркер).
- Фільтрація миттєвих коливань значень за допомогою технології Anti-Flickering.

Методологія A/B тестування

- Створення конфігурації Split-Screen для порівняння в реальному часі.
- Ліве вікно: базова камера Unity лінійне слідування.
- Праве вікно: розроблена гібридна система.
- Обидві камери одночасно отримують однакові деструктивні команди з одного пристрою вводу.

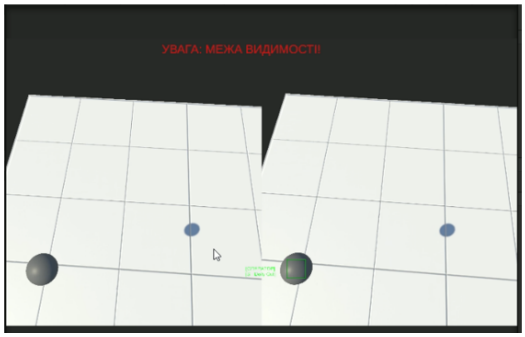


Тестові сценарії та профілі

- Пасивний профіль: відсутність втручання людини перевірка чистої автоматики трекінгу.
- Сприятливий профіль: ручні команди допомагають утримувати об'єкт.
- Несприятливий профіль: оператор навмисно намагається вивести ціль за межі кадру.

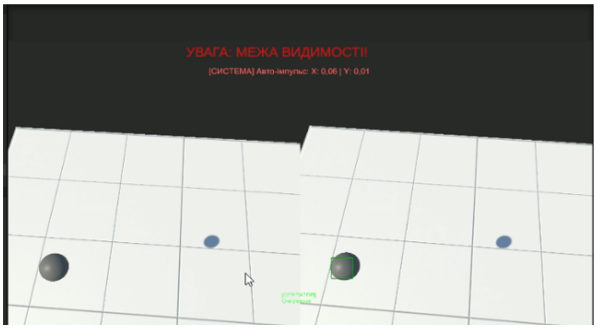
Тестування пасивної поведінки

- Оператор не генерує сигналів керування.
- При виході об'єкта з мертвої зони вмикається Warning Zone.
- Система самостійно генерує імпульс утримання, плавно повертаючи ціль до центру кадру.



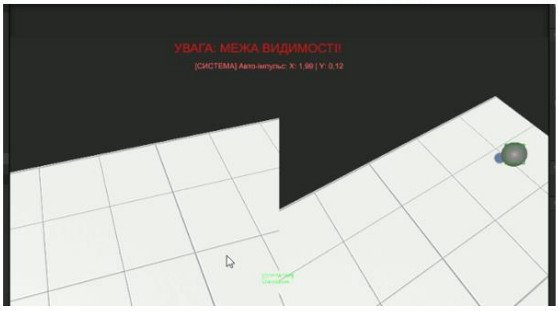
Тестування стрес-сценарію

- Оператор подає різкі команди вводу, протилежні руху цілі.
- Об'єкт перетинає межу та активує Critical Zone.
- Система миттєво обчислює максимальний коефіцієнт протидії, гасить ручний сигнал і запобігає втраті об'єкта.



Сценарій перманентної різниці швидкостей

- Лінійна швидкість об'єкта тривалий час перевищує конструктивну швидкість зміщення камери.
- Система успішно компенсує лінійне відставання за рахунок кутового панорамування.
- Об'єкт стабільно утримується ближче до краю кадру без повного виходу за межі видимості.



Налаштування коефіцієнтів демпфування

- Низькі значення демпфування викликають паразитний джиттер (візуальне тремтіння).
- Високі значення призводять до значного запізнення реакції камери.
- Експериментально доведено, що стан критичного затухання забезпечує найвищу плавність при паралельній обробці ручних та автоматичних команд.

Висновки

- Створено стабільну гібридну систему управління, яка успішно вирішує проблему конфлікту ручного вводу та автоматичного супроводу цілі.
- Завдяки селективному демпфуванню та моделі віртуальної інерції досягнуто кінематографічної плавності рухів камери.
- Практичні випробування в Unity підтвердили зниження когнітивного навантаження на оператора та надійне утримання динамічних об'єктів у кадрі.

Дякую за увагу!