СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

на тему «Розробка та реалізація механіки збирання врожаю з використанням реалістичної фізики в грі "Song of the Fields" на базі Unity»

Виконав: студент групи ІПЗ-21д	_____ (підпис)	І. К. Харченко (ініціали і прізвище)
Керівник	_____ (підпис)	О.С. Дюбанов (ініціали і прізвище)
Завідувач кафедри	_____ (підпис)	О.І. Захожай (ініціали і прізвище)

Рецензент _____О.І. Захожай_____

Київ – 2025

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Захожай О.І.
“___” _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Харченко Ілля Костянтинович
(прізвище, ім’я, по батькові)

1. Тема роботи: Розробка та реалізація механіки збирання врожаю з використанням реалістичної фізики в грі "Song of the Fields" на базі Unity
керівник роботи Дюбанов Олексій Сергійович, к.е.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “_27_”_травня_2025 року №67/15.15-С

2. Строк подання студентом роботи 15.06.2025р.
3. Вихідні дані до роботи: Розробка та реалізація механіки збирання врожаю з використанням реалістичної фізики в грі "Song of the Fields" на базі Unity
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Вступ. Теоретичні основи та аналіз ігрових симуляторів. Розробка архітектури та алгоритмів механіки збирання врожаю. Реалізація механіки збирання врожаю в Unity Висновки. Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників) Скріншот сторінки сайту farming-simulator.com. Скріншот сторінки сайту purefarminggame.com. Скріншот сторінки гри в магазині Steam Скріншот сторінки сайту stardewvalley.net. Діаграма компонентів системи механіки збирання врожаю.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 03.03.2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	03.03.2025	виконано
2	Укладання і погодження з керівником плану і етапів виконання роботи	20.03.2025	виконано
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	11.04.2025	виконано
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	10.05.2025	виконано
5	Укладання та тестування програмного продукту	02.06.2025	виконано
6	Укладання, оформлення та погодження пояснювальної записки з керівником	10.06.2025	виконано
7	Надання готової пояснювальної записки на кафедру	15.06.2025	виконано
8	Укладання доповіді і презентації	16.06.2025	виконано

Студент _____ І. К. Харченко
підпис (ініціали і прізвище)

Керівник роботи _____ О. С. Дюбанов
підпис (ініціали і прізвище)

РЕФЕРАТ

Робота містить: 78 сторінок основного тексту, 18 сторінки додатків (на основі наданих фрагментів), 1 діаграми (класів, компонентів, фізичної моделі), 4 групи скріншотів та відеоматеріалів, 17 використаних джерел.

Метою випускної кваліфікаційної роботи є розробка та реалізація механіки збирання врожаю з використанням реалістичної фізики в грі "Song of the Fields" на базі ігрового рушія Unity.

У роботі було проаналізовано існуючі реалізації механік збирання врожаю та фізичних моделей сільськогосподарської техніки в сучасних симуляторах. Вивчено теоретичні основи фізичної симуляції твердих тіл та колісних транспортних засобів, а також досліджені можливості ігрового рушія Unity для імплементації фізичних систем та їх оптимізації.

Розроблено архітектуру програмної системи для механіки збирання врожаю, включаючи модулі управління комбайном, фізичної симуляції, жатки, управління полем, зберігання врожаю та візуалізації. Реалізовано математичну модель та алгоритми реалістичної фізики руху комбайна (з урахуванням маси, інерції, зчеплення з поверхнею та впливу ландшафту) в середовищі Unity. Імплементовано механіку взаємодії комбайна з полем, що включає визначення зони зрізання, анімацію зрізання рослин, завантаження зерна та динамічну зміну візуального стану поля.

Система задовольняє вимогам щодо реалістичності та функціональності механіки збирання врожаю. Забезпечено оптимізацію продуктивності з використанням інструментів Unity Profiler, LOD, GPU Instancing та інших технік. Проведено тестування, що підтвердило коректність та ефективність реалізованих рішень.

Практична цінність роботи полягає у створенні функціонального та реалістичного компонента гри, який може бути інтегрований у майбутній ігровий продукт "Song of the Fields" або використаний у інших симуляторах сільського господарства.

Зміст

Вступ

В епоху стрімкого технологічного прогресу та глобальної цифровізації, індустрія інтерактивних розваг постає як один із провідних локомотивів світової економіки [13]. Вона постійно шукає нові шляхи для розширення горизонтів ігрового досвіду та поглиблення занурення користувачів у віртуальні світи. Розробка цифрових ігор – це надзвичайно складний та багатогранний процес, що вимагає інтеграції широкого спектру спеціалізованих знань: від алгоритмічного програмування та математичного моделювання до художнього дизайну, фізики та психології сприйняття[6].

Серед розмаїття жанрових напрямків, особливе місце посідають симулятори. Віртуальні аграрні господарства, такі як популярні серії "Farming Simulator"[1] або медитативна "Stardew Valley"[4], здобули значну прихильність аудиторії завдяки своїй здатності залучати гравців у деталізовані та достовірні процеси управління фермою. Цей ігровий досвід охоплює всі етапи агрокультурного циклу: від підготовки ґрунту та посіву до ретельного догляду за рослинами та, що є фундаментальним для жанру, збирання врожаю.

Центральним аспектом ігрового процесу в аграрних симуляторах є механізми збирання врожаю. Вони не просто відображають кінцевий акт збору культур, але й мають передавати відчуття взаємодії з великогабаритною та потужною технікою, вплив динамічно мінливого ландшафту, а також тонкощі фізичної взаємодії машини з рослинним покривом. Саме в контексті такого запиту формується концепція віртуального аграрного господарства "Song of the Fields", амбітною метою якого є відтворення максимально достовірного та захоплюючого симуляційного досвіду.

У межах даної кваліфікаційної роботи основний наголос робиться на створенні та впровадженні однієї з найважливіших ігрових систем – механіки збирання врожаю. Особлива увага приділяється інтеграції передових принципів реалістичної фізики, що стосуються динаміки руху сільськогосподарської техніки та її взаємодії з віртуальним середовищем[8].

Такий підхід покликаний забезпечити гравцям можливість відчути всі нюанси експлуатації комбайнів та іншого спеціалізованого обладнання.

Актуальність обраної теми диктується кількома взаємопов'язаними чинниками. По-перше, на сучасному ринку ігрової індустрії спостерігається стійкий та зростаючий попит на симулятори, що пропонують підвищений рівень реалізму[17]. Користувачі прагнуть відчути себе справжніми операторами складної техніки, а це вимагає від розробників максимальної достовірності у моделюванні. Якісна імплементація фізичних моделей та візуально переконлива взаємодія з ігровим світом є вирішальними факторами для залучення та утримання цільової аудиторії в цьому специфічному жанрі. По-друге, розробка ефективних алгоритмів для симуляції фізичних процесів у реальному часі, особливо для таких складних об'єктів[7], як комбайни, становить нетривіальне технічне завдання. Його вирішення має значну наукову та практичну цінність у контексті комп'ютерної графіки [14, 15, 16]та ігрового дизайну. По-третє, ігровий рушій Unity[10], будучи одним з найбільш поширених та гнучких інструментів, надає широкий спектр можливостей для реалізації подібних систем, однак вимагає глибокого розуміння його внутрішньої архітектури, фізичного ядра та стратегій оптимізації[5].

Основна мета даної кваліфікаційної роботи полягає в комплексній розробці та впровадженні механіки збирання врожаю, що базується на принципах реалістичної фізики, для віртуальної гри "Song of the Fields" з використанням ігрового рушія Unity.

Для досягнення цієї мети необхідно послідовно вирішити наступні специфічні завдання:

1. Здійснити всебічний аналіз існуючих підходів до реалізації механік збирання врожаю та фізичних моделей сільськогосподарської техніки, що застосовуються в провідних сучасних симуляторах.

2. Грунтовно вивчити фундаментальні теоретичні засади фізичної симуляції твердих тіл та динаміки колісних транспортних засобів у контексті ігрових рушіїв[8].
3. Детально дослідити потенціал та інструментарій ігрового рушія Unity щодо впровадження фізичних систем, моделювання взаємодії об'єктів та їх оптимізації для реального час[10].
4. Спроектувати та обґрунтувати архітектуру програмної системи, яка забезпечить функціонування механіки збирання врожаю, охоплюючи модулі управління, фізичних розрахунків, взаємодії з ігровим полем та візуалізації.
5. Розробити та здійснити програмну реалізацію математичної моделі та відповідних алгоритмів для забезпечення реалістичної фізики руху комбайна (включаючи інерцію, вплив маси, особливості зчеплення з поверхнею та реакцію на нерівності ландшафту) у середовищі Unity.
6. Сформулювати та впровадити механіку взаємодії комбайна з віртуальним полем, що охоплює точне визначення зони зрізання, анімаційне відображення процесу зрізання рослин, коректне завантаження зібраного зерна та динамічну зміну візуального стану поля.
7. Забезпечити функціонування системи візуалізації, яка відображатиме процес збирання врожаю та динамічні зміни візуального представлення поля після проходу сільськогосподарської техніки.
8. Провести серію всебічних тестів розробленої механіки, включаючи оцінку її реалістичності, продуктивності та інтуїтивності використання для кінцевого користувача.
9. Надати обґрунтовані рекомендації щодо подальшого вдосконалення та розширення функціоналу механіки збирання врожаю в рамках ігрового проекту "Song of the Fields".

Об'єктом дослідження виступає методологія та процес створення інтерактивних ігрових механік в середовищі ігрового рушія Unity.

Предметом дослідження є конкретні алгоритмічні рішення та методи впровадження механіки збирання врожаю, що ґрунтується на реалістичній фізиці та достовірній взаємодії з навколишнім віртуальним середовищем у грі "Song of the Fields".

Наукова новизна даної роботи полягає у розробці та інтеграції комплексної фізичної моделі великовагової сільськогосподарської техніки з динамічною системою взаємодії з ігровим полем. Ця система не тільки моделює рух та колізії, але й враховує та відображає візуальні та ресурсні зміни врожаю в реальному часі, що дозволяє досягти нового рівня імерсії в жанрі агрокультурних симуляторів.

Практична цінність результатів роботи проявляється у створенні готового до інтеграції, функціонального, реалістичного та оптимізованого програмного компонента для гри "Song of the Fields". Окрім цього, розроблені алгоритми та інженерні підходи можуть бути успішно адаптовані та використані в інших симуляторах сільського господарства, що сприятиме загальному підвищенню якості та занурення в ігровий процес у цьому сегменті індустрії.

РОЗДІЛ 1 Теоретичні основи та аналіз ігрових симуляторів

1.1 Огляд симуляторів сільського господарства та їхніх механік збирання врожаю

Жанр інтерактивних симуляторів сільськогосподарської діяльності пройшов значний шлях еволюції, перетворившись з елементарних економічних стратегій на багатогранні віртуальні ферми, оснащені високодеталізованим обладнанням та реалістичними виробничими процесами. Для успішної реалізації інноваційної механіки збирання врожаю в проєкті "Song of the Fields", необхідно провести всебічний аналіз концепцій та підходів, втілених у провідних представниках цього жанру. Цей огляд дозволить ідентифікувати кращі практики та визначити напрямки для вдосконалення.

"Farming Simulator"[1] (Розробник: GIANTS Software)

Ця серія ігор беззаперечно визнана еталоном та золотим стандартом у ніші симуляторів сільського господарства. Вона послідовно встановлює високу планку для конкурентів завдяки своїй глибині та деталізації.

Рис 1.1 - Головна сторінка сайту farming-simulator.com

Реалізація механіки збирання врожаю: Взаємодія з процесом збору врожаю в "Farming Simulator"[1] характеризується винятковою деталізацією.

Гравці мають змогу використовувати різноманітні види комбайнів, кожен з яких відтворює специфічні особливості реальних прототипів. До комбайнів приєднуються спеціалізовані жатки, що можуть бути як невід'ємною частиною комбайна, так і окремим навісним обладнанням, що кріпиться та від'єднується за потребою. Безпосередній процес збирання культури супроводжується візуальним відгуком: дозрілі рослини зникають з ігрового поля, а його поверхня динамічно змінює свою текстуру, відображаючи ділянку, де вже відбувся збір. Зібраний урожай автоматично транспортується та накопичується у внутрішньому бункері комбайна. Після його заповнення, гравцеві необхідно перевантажити врожай до спеціального причепа або безпосередньо у стаціонарні сховища на фермі. Важливо відзначити, що різні агрокультури (наприклад, пшениця, кукурудза, ріпак) вимагають використання конкретних типів жаток та мають відмінні показники врожайності, що додає грі стратегічної глибини.

Фізична модель транспортних засобів: "Farming Simulator"[1] прагне до досягнення високого рівня реалізму у відтворенні фізичної поведінки техніки. Машини відчуваються ваговитими, їхній рух супроводжується відчутною інерцією. Колеса реагують на топографічні особливості ландшафту, а система підвіски активно функціонує, поглинаючи нерівності ґрунту. Крім того, в грі присутня система пошкоджень, яка може негативно впливати на експлуатаційні характеристики техніки, наприклад, знижуючи потужність двигуна або обмежуючи швидкість пересування. Проте, слід зазначити, що деякі аспекти фізичної моделі можуть бути дещо спрощеними для забезпечення більшої доступності та комфорту управління для широкої аудиторії.

Механізми управління: Система керування технікою в "Farming Simulator"[1] відрізняється значною складністю, відображаючи багатофункціональність реальних сільськогосподарських машин. Доступні різні ракурси камери, включаючи вид від першої особи з кабіни оператора, що

сприяє глибшому зануренню. Для оптимізації рутинних завдань, гравці можуть скористатися функцією автопілота, яка автоматизує деякі операції на полі.

Економічні аспекти: Процес збирання врожаю займає центральне місце в економічній системі гри. Зібрана продукція реалізується на віртуальному ринку за динамічними цінами, що слугує стимулом для гравця до оптимізації процесів збору та збуту. Важливо також враховувати операційні витрати, такі як вартість палива та необхідність регулярного ремонту техніки, що безпосередньо впливає на рентабельність аграрного підприємства.

Потенційні недоліки: Незважаючи на загальний високий рівень, інколи фізика транспортних засобів може сприйматися як дещо "пластикова", а її реакція на екстремальні нерівності поверхні не завжди є абсолютно коректною. Також, візуальні ефекти, пов'язані зі зрізанням рослин, хоча й присутні, могли б бути більш деталізованими та реалістичними для досягнення максимальної імерсії.

"Pure Farming"[2] (Розробник: Techland)

Цей ігровий проєкт був спробою створити конкуренцію серії "Farming Simulator"[1], представивши власну інтерпретацію жанру.

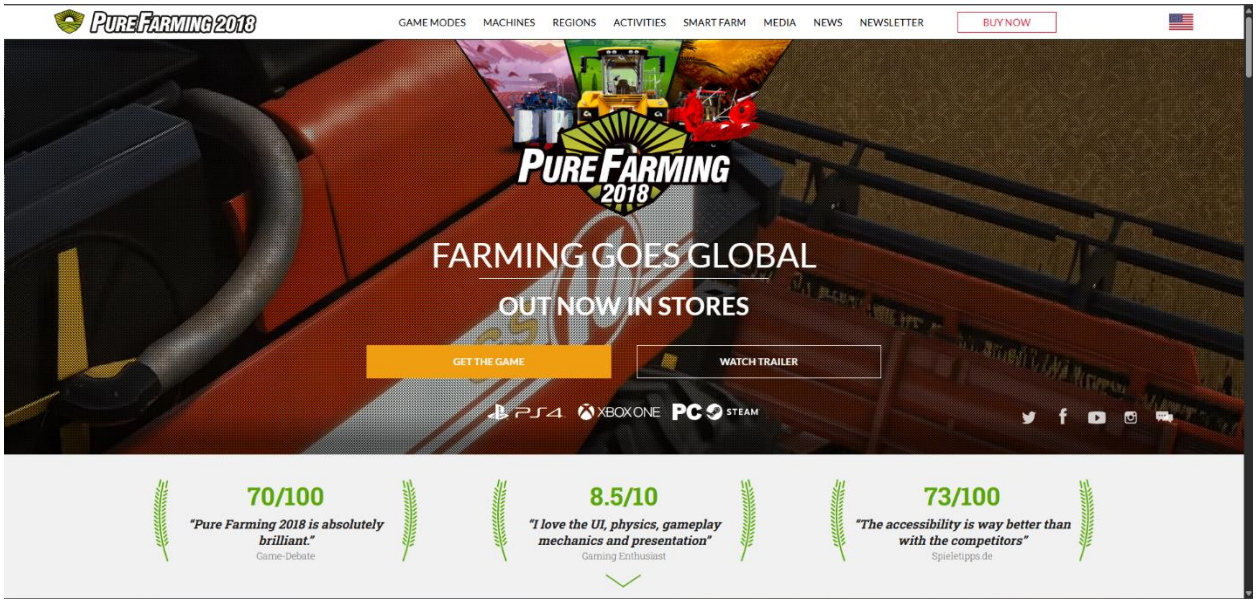

Рис 1.2 - Головна сторінка сайту purefarminggame.com

Механіка збирання врожаю: Реалізація процесу збору врожаю була загалом подібною до "Farming Simulator"[1], проте мала певні нюанси у відчуттях від керування та рівні візуальної деталізації. Ефекти зрізання рослин та зміни вигляду поля після проходу комбайна були присутні.

Фізична модель техніки: Фізика була реалістичною у своїй основі, однак окремі користувачі відзначали, що техніка могла здаватися менш "вагомою" порівняно з головним конкурентом у жанрі.

"Farm Together"[3] (Розробник: Milkstone Studios)

Ця гра є більш казуальною та орієнтованою на соціальну взаємодію, що відображається у її ігрових механіках.

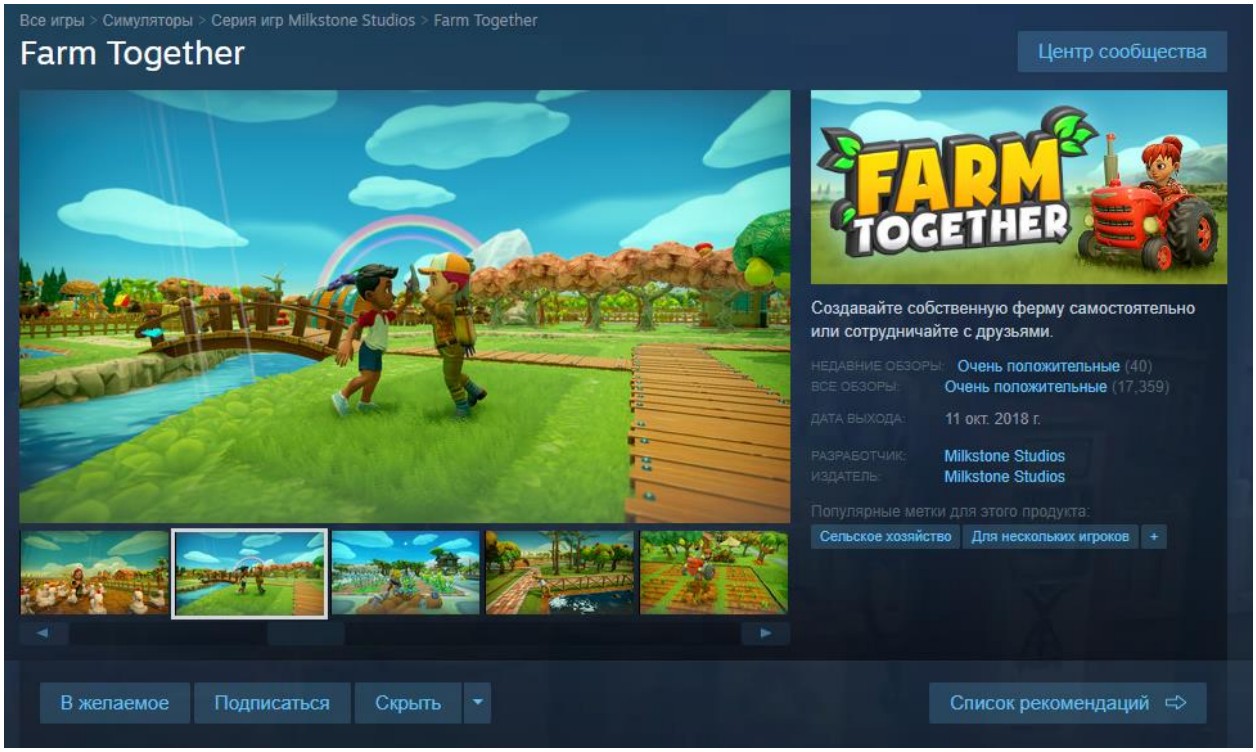

Рис. 1.3 - Сторінка гри в магазині Steam

Механіка збирання врожаю: Взаємодія зі збором урожаю значно спрощена. Зазвичай це реалізується через механіку "кліку" або швидкого проходження над полем, без залучення глибокої фізичної симуляції техніки. Головною метою тут є не достовірність процесу, а швидке та приємне виконання завдань, що сприяє розслаблюючому ігровому процесу.

Фізика: Фізичні розрахунки в грі мінімальні або повністю відсутні, оскільки основний акцент робиться на розбудові ферми, декораціях та соціальній взаємодії між гравцями, а не на реалістичному управлінні сільськогосподарською технікою.

"Stardew Valley"[4] (Розробник: ConcernedApe)

Ця піксельна рольова гра-симулятор життя з елементами фермерства пропонує унікальний ігровий досвід.

Рис 1.4 - Головна сторінка сайту stardewvalley.net

Механіка збирання врожаю: Процес збору врожаю є вкрай абстрактним. Гравець збирає культури вручну, використовуючи базові інструменти, без залучення великої сільськогосподарської техніки.

Фізика: Фізична симуляція як така повністю відсутня, оскільки гра не передбачає реалістичної взаємодії з оточенням у традиційному сенсі.

Висновок за результатами огляду: На основі проведеного аналізу стає очевидним, що для успішної розробки проєкту "Song of the Fields" слід орієнтуватися на рівень деталізації та реалізму, запропонований серією "Farming Simulator"[1]. Однак, метою є не просто копіювання, а прагнення до подальшого вдосконалення фізичної моделі та візуальних ефектів взаємодії техніки з ігровим полем. Це включає в себе більш точне моделювання динаміки руху, реакції на нерівності ґрунту та інтеграцію розширених візуальних ефектів збирання. Кінцевою метою є досягнення оптимального

балансу між високим рівнем реалізму та зручністю та комфортом ігрового процесу, щоб забезпечити максимальне занурення та задоволення для гравця.

1.2 Теоретичні основи фізичної симуляції в ігрових рушіях

Відтворення достовірної фізичної поведінки об'єктів у віртуальному просторі є фундаментальним компонентом для досягнення ефекту занурення в інтерактивних симуляціях, особливо в таких жанрах, як сільськогосподарські. Ігрові рушії[5], такі як Unity[10], застосовують оптимізовані, хоча й спрощені, математичні моделі для ефективного обчислення переміщень та взаємодій об'єктів у режимі реального часу[8]. Це забезпечує баланс між реалізмом та обчислювальною ефективністю.

Динаміка твердого тіла:

У центрі будь-якої фізичної симуляції лежить динаміка твердого тіла[8], яка описує, як об'єкти рухаються під впливом зовнішніх факторів.

Фундаментальні принципи механіки: Базові аксіоми механіки, сформульовані Ньютоном, є наріжним каменем всіх фізичних розрахунків. Зокрема, принцип, що пов'язує силу, масу та прискорення (часто виражається як F=ma), є ключовим для визначення того, як об'єкт реагує на прикладені до нього впливи. Аналогічно, закон про рівність дії та протидії (де дія на об'єкт викликає зворотну реакцію) критично важливий для моделювання взаємодії об'єктів при їхньому контакті.

Інерційні характеристики: Кожен фізичний об'єкт у віртуальному середовищі володіє певною масою, яка чинить опір будь-яким змінам у його стані руху. Крім того, здатність об'єкта чинити опір обертанню навколо своєї осі описується поняттям моменту інерції. Ці параметри мають вирішальне значення для адекватного моделювання великовагових машин, таких як комбайн, які природно демонструють значну інерцію як лінійного, так і кутового руху.

Дія зовнішніх факторів: Додаткові вектори впливу, такі як сила тяжіння, тертя, тяга двигуна, гальмівні зусилля або пружні властивості елементів

підвіски, спричиняють лінійне прискорення об'єкта. Обертальні впливи, відомі як моменти сил (торки), викликають кутове прискорення, що є визначальним для обертання коліс та зміни напрямку руху транспортних засобів.

Моделювання взаємодії (Виявлення та вирішення зіткнень):

Виявлення та коректне опрацювання зіткнень є невід'ємною частиною реалістичної фізичної симуляції, запобігаючи проникненню об'єктів один в одного[8].

Етапи виявлення потенційних контактів:

Широкофазний (Broad-phase): На цьому початковому етапі відбувається швидке виявлення всіх об'єктів, які потенційно можуть зіткнутися, використовуючи спрощені геометричні примітиви, що обмежують їхній об'єм (наприклад, прямокутні паралелепіпеди, вирівняні по осях - AABB, або сфери). Сучасні ігрові рушії використовують оптимізовані структури даних (наприклад, Octrees або Bounding Volume Hierarchies) для прискорення цього процесу[5].

Вузькофазний (Narrow-phase): Після ідентифікації потенційних пар об'єктів, цей етап фокусується на точному визначенні конкретних точок та нормалей контакту. Для цього застосовуються складніші алгоритми (наприклад, GJK або EPA), що дозволяють точно обчислити геометрію зіткнення навіть для об'єктів з нетривіальними формами.

Методи опрацювання контактів:

Імпульсний підхід: Цей широко використовуваний метод полягає у застосуванні миттєвих змін моменту (імпульсів) до об'єктів у точках зіткнення[9]. Це дозволяє ефективно запобігти подальшому проникненню та симулювати відскок об'єктів. Фізичні рушії, такі як PhysX (який використовує Unity[10]), активно застосовують цей підхід.

Метод штрафних сил: Інший підхід передбачає застосування сил, пропорційних глибині проникнення одного об'єкта в інший. Хоча цей метод може бути менш стабільним для симуляції жорстких тіл, він знаходить

застосування в певних сценаріях, наприклад, для моделювання м'яких тіл або деформованих поверхонь.

Моделювання сил тертя:

Сили тертя відіграють критичну роль у реалістичному русі об'єктів, особливо транспортних засобів, забезпечуючи зчеплення з поверхнею та опір руху[8].

Різновиди тертя:

Статичне тертя: Це сила, яка протидіє початку руху об'єкта, що перебуває у стані спокою на поверхні. Вона повинна бути подолана, щоб об'єкт почав рухатися.

Кінетичне тертя: Ця сила діє на об'єкт, що вже перебуває в русі, і протидіє його переміщенню. Вона зазвичай менша за статичне тертя.

Коефіцієнти взаємодії: Сила тертя визначається коефіцієнтами тертя, які залежать від фізичних властивостей поверхонь, що контактують. Для сільськогосподарської техніки, особливо комбайна, надзвичайно важливим є коефіцієнт тертя між шинами та ґрунтом, оскільки його значення може істотно змінюватися залежно від стану поверхні (наприклад, сухий, мокрий, пухкий або твердий ґрунт).

Спеціалізовані моделі колісних транспортних засобів:

Більшість сучасних ігрових рушіїв пропонують високооптимізовані компоненти для симуляції автомобілів та іншої колісної техніки. Ці компоненти значно спрощують реалізацію складних динамічних систем, таких як підвіска, шини та їхня взаємодія з поверхнею[8, 11].

Система підвіски: Моделюється як інтегрована пружинно-демпферна система, що дозволяє кожному колесу рухатися вертикально відносно корпусу транспортного засобу. Це забезпечує ефективне поглинання ударів та нерівностей поверхні, підтримуючи стабільність та керованість машини.

Взаємодія шини з поверхнею: Це один з найскладніших аспектів симуляції транспортних засобів. Вона включає облік сил тертя як у поздовжньому напрямку (що відповідає за прискорення та гальмування), так і

в поперечному напрямку (що впливає на здатність транспортного засобу повертати). Для більш достовірного моделювання нелінійної поведінки шин (наприклад, втрати зчеплення при надмірному ковзанні) використовуються криві зчеплення (friction curves), які графічно описують залежність сили тертя від ступеня ковзання шини.

Чисельні методи інтегрування:

Оскільки аналітичні розв'язки для складних систем фізичних рівнянь у реальному часі, як правило, відсутні, ігрові рушії покладаються на чисельні методи інтегрування[8].

Покрокове обчислення: Ці методи дозволяють покроково обчислювати нові положення та швидкості об'єктів у часі. Серед поширених алгоритмів можна виділити метод Ейлера (хоча він простий, може бути нестабільним), інтегрування Верле (більш стабільний для певних систем) або більш точні методи Рунге-Кутти (хоча й більш ресурсоємні).

Фіксовані кроки симуляції: Для забезпечення стабільності та детермінованості фізичних розрахунків, Unity[10], як і багато інших ігрових рушіїв, використовує фіксовані кроки часу для фізики (Fixed Timestep). Це означає, що фізичні обчислення виконуються через рівні, заздалегідь визначені інтервали часу, незалежно від зміни частоти кадрів рендерингу. Це запобігає варіативності фізичної поведінки при зміні продуктивності системи.

1.3 Інструментарій Unity для впровадження фізичних моделей та взаємодії

Ігровий рушій Unity надає розробникам потужний і всеосяжний вбудований фізичний двигун, який базується на технології NVIDIA PhysX[10]. Цей двигун є фундаментальною основою для створення реалістичної та інтерактивної поведінки різноманітних об'єктів у віртуальному просторі, що критично важливо для таких проєктів, як "Song of the Fields", де достовірність відчуттів від керування технікою має першочергове значення.

Компонент Rigidbody:

Компонент Rigidbody (тверде тіло) є наріжним каменем фізичної симуляції в Unity[10]. Його приєднання до будь-якого ігрового об'єкта миттєво наділяє цей об'єкт фізичними властивостями, дозволяючи йому взаємодіяти з силами (гравітація, прикладені сили, імпульси) та моментами, а його рух обчислюється фізичним двигуном.

Ключові властивості Rigidbody:

Mass (маса): Визначає інерцію об'єкта. Для важкого комбайна високе значення маси є обов'язковим для адекватного відображення його інерційних характеристик – повільного розгону, інерційного гальмування та відчуття "ваги".

Drag (лінійний опір): Сила, що протидіє лінійному руху об'єкта, імітуючи опір повітря або води.

Angular Drag (кутовий опір): Сила, що протидіє обертальному руху, симулюючи опір обертанню.

Use Gravity (використовувати гравітацію): Логічне значення, що визначає, чи буде об'єкт піддаватися впливу гравітаційної сили.

Is Kinematic (кінематичне): Якщо активовано, об'єкт керується виключно трансформаціями через скрипти, а не фізичним двигуном. Це корисно для об'єктів, які повинні рухатися за заданою траєкторією без зовнішнього фізичного впливу, але при цьому виявляти зіткнення. Для основного комбайна це буде вимкнено, щоб він керувався фізикою.

Для моделювання комбайна, Rigidbody є основним компонентом, що забезпечує його динамічну поведінку.

Компоненти Collider (Колайдери):

Колайдер – це компонент, який визначає геометричну форму об'єкта для виявлення зіткнень. Колайдери є невидимими для гравця, але вони є фізичною оболонкою об'єкта. Unity пропонує широкий спектр колайдерів для різних цілей:

Box Collider: Ідеально підходить для об'єктів прямокутної форми. Може використовуватися для основних частин корпусу комбайна або окремих елементів оточення.

Sphere Collider: Для сферичних об'єктів.

Capsule Collider: Для циліндричних або капсульних об'єктів, часто використовується для персонажів.

Mesh Collider: Створює колайдер на основі тривимірної сітки (3D-моделі) об'єкта. Може бути як Convex (опуклим – для оптимізації та взаємодії з іншими Convex колайдерами), так і не-Convex (для точних, але ресурсоємних зіткнень з Raycasts). Для складних візуальних моделей комбайна або жатки часто застосовується Mesh Collider у поєднанні з примітивними колайдерами для ключових частин.

Terrain Collider: Спеціалізований колайдер, призначений для ефективної взаємодії з компонентом Terrain (ландшафту) Unity. Колеса комбайна взаємодіятимуть саме з цим типом колайдера на полі.

Властивість Is Trigger: Якщо цю властивість активовано, колайдер перетворюється на "тригер". Він не спричиняє фізичних зіткнень та відштовхувань, але дозволяє виявляти, коли інший колайдер входить у його об'єм, перебуває в ньому або виходить. Це є надзвичайно цінним для реалізації зони зрізання врожаю жаткою комбайна: коли жатка "входить" в тригер-зону зрілих рослин, спрацьовує логіка збирання.

Physics Material (Фізичний Матеріал):

Physics Material є спеціалізованим асетом, що дозволяє детально визначати властивості тертя (dynamicFriction, staticFriction) та пружності (bounciness) для поверхні колайдера[10]. Це має вирішальне значення для моделювання реалістичної взаємодії шин комбайна з різними типами ґрунту (наприклад, сухий ґрунт, мокрий ґрунт, бруд, асфальт), оскільки кожен з них матиме унікальні характеристики зчеплення та відскоку. Це дозволить зробити керованість комбайна більш достовірною залежно від умов.

Physics Layers (Фізичні Шари):

Шари зіткнень – це потужний інструмент для керування тим, які об'єкти в ігровому світі будуть фізично взаємодіяти один з одним[10]. У Project Settings -> Physics можна налаштувати матрицю зіткнень, яка визначає, які шари будуть ігнорувати один одного. Це критично важливо для оптимізації фізичних розрахунків, оскільки дозволяє уникнути непотрібних перевірок зіткнень. Наприклад, можна налаштувати, щоб корпус комбайна не зіштовхувався з дрібними елементами рослинності (які обробляються окремою логікою тригерів жатки), а взаємодіяв лише з великими перешкодами або іншими транспортними засобами.

Компонент Wheel Collider:

Це високоспеціалізований компонент Unity, розроблений саме для достовірної симуляції коліс транспортних засобів[11]. Він є ідеальним вибором для моделювання комбайна, оскільки значно спрощує реалізацію складної взаємодії колеса з поверхнею. Важливо розуміти, що Wheel Collider сам по собі не є візуальним елементом; він виконує лише фізичні розрахунки, а його візуальне представлення досягається шляхом прив'язки до 3D-моделі колеса.

Деталізовані параметри Wheel Collider:

Підвіска: Дозволяє налаштовувати жорсткість пружини (Spring), демпфування (Damper) та цільове положення підвіски (Target Position), що впливає на поглинання нерівностей та стабільність комбайна.

Зчеплення шин: Ключові Wheel Friction Curve (для поздовжнього Forward та поперечного Sideways тертя) дозволяють моделювати нелінійну поведінку шин: як сила тертя наростає та спадає залежно від ковзання колеса. Це дозволяє симулювати проковзування на м'якому ґрунті або під час різкого гальмування/прискорення.

Застосування моментів: Через Wheel Collider можна безпосередньо застосовувати моторний момент (motorTorque) для прискорення та гальмівний момент (brakeTorque) для сповільнення, що автоматично передається до Rigidbody транспортного засобу.

Raycasting та Spherecasting (Методи Променевого та Сферичного Проектування):

Це фундаментальні методи для виявлення об'єктів у просторі[10].

Raycasting: Запускає невидимий промінь з певної точки в заданому напрямку та виявляє перший об'єкт, з яким він зіштовхується. Може використовуватися для:

Визначення поверхні, по якій рухається комбайн (для адаптації звуків, візуальних ефектів, наприклад, зміни частинок пилу залежно від типу ґрунту).

Моніторингу відстані до перешкод.

Spherecasting: Схожий на Raycasting, але замість променя використовує сферу, яка рухається в заданому напрямку, виявляючи зіткнення з іншими об'єктами. Може бути корисним для ширших перевірок зіткнень.

Хоча для безпосереднього зрізання врожаю ефективнішим є використання колайдерів-тригерів на жатці, Raycasting може доповнювати систему, наприклад, для визначення висоти рослинності або для інших візуальних ефектів.

Системи частинок (Particle Systems):

Компонент Particle System – це надзвичайно потужний інструмент Unity для створення візуальних ефектів, що складаються з великої кількості дрібних елементів[10]. Вони є незамінними для імітації:

Пилу, що здіймається від коліс комбайна під час руху.

Димних вихлопів з двигуна.

Візуалізації розкиданого зерна або соломи під час та після процесу збирання.

Particle Systems легко інтегруються з фізичним рушієм, дозволяючи частинкам реагувати на гравітацію, зіткнення та інші сили.

Shader Graph та VFX Graph (Графічні Редактори Візуальних Ефектів):

Ці інструменти надають візуальний, вузловий підхід до створення складних графічних ефектів, усуваючи необхідність писати складний код шейдерів[10].

Shader Graph: Дозволяє створювати користувацькі шейдери для матеріалів без програмування. Це може бути використано для:

Динамічної зміни текстур поверхні поля після збирання врожаю (перехід від "дозрілої культури" до "зібраного поля" або "ґрунту").

Додавання реалістичних ефектів бруду, пилу або іржі на поверхню техніки, що змінюються залежно від ігрових подій.

VFX Graph: Призначений для створення високопродуктивних візуальних ефектів з мільйонами частинок, що обчислюються на GPU. Він ідеально підходить для:

Моделювання великих обсягів зерна або соломи, що вивантажується з бункера комбайна.

Створення складних погодних ефектів, таких як сильний дощ або сніг, що реалістично взаємодіє з ігровим світом.

Scriptable Objects (Скриптовні Об'єкти):

Scriptable Objects – це спеціальний тип асетів Unity, що дозволяють створювати гнучкі, незалежні від сцени та багаторазово використовувані об'єкти даних[10]. Вони є цінним інструментом для архітектури даних, дозволяючи:

Визначати характеристики різних типів комбайнів (потужність, місткість бункера, швидкість).

Описувати властивості різних видів врожаю (час дозрівання, врожайність, ціна продажу).

Зберігати конфігурації для жаток та іншого обладнання.

Це сприяє чистоті коду, централізованому управлінню даними та легкості балансування ігрових параметрів.

Інструменти профілювання та оптимізації:

Для забезпечення стабільної та плавної роботи гри, особливо в умовах інтенсивних фізичних розрахунків та великих відкритих світів, Unity надає вбудовані інструменти для моніторингу та аналізу продуктивності[10].

Unity Profiler: Це основний інструмент для моніторингу використання CPU та GPU в реальному часі. Він дозволяє виявити "вузькі місця" у:

Physics.FixedUpdate: Якщо цей блок займає надто багато часу, це вказує на потенційні проблеми з кількістю фізичних об'єктів, складністю колайдерів або неефективними фізичними розрахунками.

Scripts: Дозволяє ідентифікувати ресурсоємні методи у ваших власних скриптах.

Rendering: Допомагає аналізувати кількість "Draw Calls" (викликів рендерингу), "Batches" (об'єднаних викликів) та кількість полігонів, що рендеряться.

Frame Debugger: Дозволяє детально покроково аналізувати процес рендерингу кожного кадру. Це корисно для виявлення проблем з накладанням матеріалів, неправильним порядком рендерингу або неоптимальним використанням текстур та шейдерів.

Використання цих інструментів є критично важливим для ітеративного процесу оптимізації, дозволяючи розробникам цілеспрямовано покращувати продуктивність механіки збирання врожаю.

1.4 Постановка задачі

Основною метою представленої роботи є формування інтегрованої системи, призначеної для відтворення механіки збирання врожаю. Ця система має бути розроблена та імплементована з акцентом на реалістичну фізику, в контексті створення комп'ютерної гри "Song of the Fields" на базі ігрового рушія Unity[10].

Вихідним контекстом для розробки виступає необхідність інтеграції достовірних симуляційних елементів у віртуальне фермерське середовище.

Всебічний аналіз сучасних ігрових рішень у галузі симуляторів сільського господарства, представлений у попередніх підрозділах, дозволив кристалізувати ключові вимоги до функціоналу та поведінки системи.

Розроблений програмний компонент повинен повністю відповідати потребам ігрового проєкту щодо забезпечення високого рівня занурення користувача, стимулювати його зацікавленість через достовірне відтворення сільськогосподарських процесів, а також мати потенціал для подальшого розширення та вдосконалення в рамках загального розвитку ігрової системи.

Механіка збирання врожаю має бути реалізована як інтегрована частина ігрового рушія Unity і виконувати наступні ключові функції:

Здійснення комплексного аналізу існуючих ігрових механік збирання врожаю та моделей фізичної поведінки аграрної техніки в провідних симуляторах.

Детальне дослідження теоретичних засад фізичної симуляції твердих тіл та динаміки колісних машин, що застосовуються в ігрових рушіях[8].

Оцінка функціональних можливостей рушія Unity щодо імплементації фізичних систем, моделювання взаємодії об'єктів та оптимізації продуктивності[10].

Проектування модульної архітектури програмної системи, що відповідає за механіку збирання врожаю, включаючи компоненти управління, фізичних розрахунків, взаємодії з ігровим полем та візуалізації.

Розробка та програмна реалізація математичних моделей та алгоритмів для відтворення реалістичної фізики руху комбайна, охоплюючи інерційні властивості, вплив маси, характеристики зчеплення з поверхнею та реакцію на топографічні нерівності.

Створення та впровадження механіки взаємодії комбайна з віртуальним полем, що передбачає точне визначення зони зрізання, анімаційне відображення процесу зрізання рослин, коректне завантаження зібраної продукції та динамічне візуальне перетворення стану поля.

Імплементація системи візуалізації, яка забезпечує достовірне відображення процесу збирання врожаю та динамічної зміни візуального представлення поля після проходу техніки.

Проведення всебічного тестування розробленої механіки з метою оцінки її реалістичності, продуктивності та зручності використання для гравців.

Формулювання обґрунтованих рекомендацій щодо подальших напрямків розвитку та вдосконалення механіки збирання врожаю в ігровому проєкті "Song of the Fields".

Висновок по розділу 1

У представленому розділі було проведено систематизацію та визначення ключових аспектів для подальшої розробки. Зокрема, ідентифіковано основні вимоги до реалізованої програмної системи, а також сформульовано конкретні завдання, вирішення яких є необхідним для досягнення поставленої мети. Детально проаналізовано функціонування існуючих ігрових симуляторів сільського господарства, їхні архітектурні особливості та підходи до сторони. Отримані висновки стануть базисом для вибору оптимальних методів реалізації та уникнення типових проблем у процесі створення даної ігрової механіки.моделювання, що дозволило виявити характерні риси, а також сильні та слабкі сторони. Отримані висновки стануть базисом для вибору оптимальних методів реалізації та уникнення типових проблем у процесі створення даної ігрової механіки.

РОЗДІЛ 2 Розробка архітектури та алгоритмів механіки збирання врожаю

2.1 Архітектура механіки збирання врожаю

Проектування ефективної та стійкої програмної архітектури є наріжним каменем у створенні будь-якої складної програмної системи, а тим більше – інтерактивної ігрової механіки, такої як збирання врожаю у фермерському симуляторі[5]. Добре продумана архітектура забезпечує низку критично важливих переваг: модульність, що дозволяє розробляти та тестувати компоненти незалежно; розширюваність, що полегшує додавання нових функцій або типів контенту в майбутньому; та зручність супроводу, оскільки зміни в одному модулі мінімально впливають на інші. У контексті механіки збирання врожаю, такий підхід дає можливість легко інтегрувати нові моделі сільськогосподарської техніки, різноманітні види культур або модифікувати існуючі алгоритми без необхідності масштабного перероблення всієї системи.

Нижче наведено опис ключових компонентів, які формують архітектуру механіки збирання врожаю, та принципів їхньої взаємодії. Ці компоненти розробляються як окремі скрипти та об'єкти в Unity[10], що сприяє високому рівню декомпозиції та чистоті коду.

Модуль управління комбайном (CombineController):

Цей компонент можна уявити як інтерфейс взаємодії між гравцем (або алгоритмом штучного інтелекту, що керує неігровим комбайном) та фізичною моделлю транспортного засобу.

Основне призначення: Головною відповідальністю CombineController є інтерпретація вхідних даних – будь то натискання клавіш, рухи аналогового стіка геймпада, або програмні команди від системи ШІ – та їхнє перетворення у високоточні керуючі сигнали. Ці сигнали включають параметри, що впливають на рух (тяга двигуна, інтенсивність гальмування, кут повороту коліс) та функціональні дії (активація або деактивація жатки, підняття/опускання жатки, запуск процесу вивантаження зерна з бункера).

Взаємодія: CombineController виступає як центральний хаб для команд, передаючи оброблені керуючі сигнали до інших відповідних модулів: безпосередньо до CombinePhysics для впливу на рух комбайна та до HarvesterHead для контролю над роботою жатки.

Технічна реалізація: Як правило, цей модуль імплементується як скрипт, що успадковується від MonoBehaviour в Unity[10], і кріпиться до основного ігрового об'єкта, що представляє комбайн. Його логіка розміщується в методі Update() для обробки введення за кожний кадр.

Фізичний модуль комбайна (CombinePhysics):

Цей компонент є ядро фізичної симуляції комбайна, що забезпечує його реалістичну поведінку у тривимірному світі.

Основне призначення: Його ключовою відповідальністю є симуляція динамічного руху комбайна, враховуючи всі фізичні властивості та впливи. Це включає точне моделювання маси, інерції, сил тертя між колесами та поверхнею, а також реакції на зіткнення та нерівності ландшафту[8].

Взаємодія: CombinePhysics отримує керуючі сигнали (значення тяги, гальмування, кута повороту) від CombineController. Він безпосередньо взаємодіє з вбудованими фізичними компонентами Unity, такими як Rigidbody [10](для управління загальною динамікою тіла) та WheelCollider[11] (для деталізованої симуляції поведінки коліс). Результати фізичних розрахунків – поточне положення, орієнтація та швидкість комбайна – передаються до HarvesterHead (оскільки жатка рухається разом з комбайном) та до SystemVisuals для оновлення візуальних аспектів.

Технічна реалізація: Імплементується як скрипт MonoBehaviour, прикріплений до кореневого об'єкта комбайна. Важливо, що його основна логіка виконується у методі FixedUpdate(), який викликається фізичним двигуном Unity з фіксованою частотою, що забезпечує стабільність фізичних розрахунків незалежно від частоти кадрів.

Модуль жатки (HarvesterHead):

Цей модуль є спеціалізованим компонентом, що моделює функціональність ріжучої частини комбайна.

Основне призначення: Головною відповідальністю HarvesterHead є симуляція процесу зрізання рослинності. Це включає постійне визначення зони, що перебуває під дією жатки, взаємодію з ігровим полем для "збору" врожаю, а також передачу інформації про зібрану продукцію до сховища комбайна. Додатково він керує анімаціями (обертання ріжучих елементів) та візуальними ефектами, пов'язаними зі зрізанням.

Взаємодія: HarvesterHead отримує команди активації та інформацію про своє положення (що визначається рухом комбайна) від CombineController та CombinePhysics відповідно. Він активно взаємодіє з FieldManager для запиту та зміни стану ділянок поля, а також з CombineStorage для передачі зібраного врожаю.

Технічна реалізація: Реалізується як скрипт MonoBehaviour, прикріплений до ігрового об'єкта жатки (який є дочірнім до об'єкта комбайна). Для визначення зони зрізання використовується власний колайдер, налаштований як тригер (Is Trigger = true), що дозволяє виявляти перетин з об'єктами рослинності або ділянками поля[10].

Модуль поля (FieldManager / CropManager):

Цей компонент виконує роль центрального сховища даних та системи управління станом ігрового поля.

Основне призначення: Його ключовою відповідальністю є зберігання та надання актуальної інформації про кожну ділянку поля. Це включає такі атрибути, як тип культури, поточна стадія росту (наприклад, посіяно, сходи, дозріле, зібране), а також флаг, що вказує, чи зібрана вже конкретна ділянка. Він також надає стандартизований інтерфейс для запитів про стан поля та його програмної зміни.

Взаємодія: FieldManager отримує запити на оновлення стану ділянок від HarvesterHead після збирання врожаю. Він може сповіщати SystemVisuals про

необхідність оновлення візуального представлення поля після зміни його стану.

Технічна реалізація: Як правило, імплементується як singleton-менеджер (MonoBehaviour або простий клас), який керує двовимірним масивом або іншою структурою даних (наприклад, текстурою або спеціалізованою сіткою) для ефективного зберігання інформації про поле. Для оптимізації роботи з великими полями, воно може бути логічно розділене на дрібніші "чанки" (chunks), що завантажуються та обробляються за потребою.

Модуль бункера та завантаження (CombineStorage):

Цей компонент відповідає за логіку зберігання та транспортування зібраного врожаю.

Основне призначення: Його основна функція – відстеження поточного об'єму накопиченої сільськогосподарської продукції в бункері комбайна. Він обробляє події надходження врожаю від жатки (завантаження) та дозволяє вивантажувати його в інші місця, такі як причіп, стаціонарне сховище або пункт продажу.

Взаємодія: CombineStorage отримує інформацію про кількість зібраного врожаю від HarvesterHead. Він також надає публічні методи для ініціювання процесу вивантаження.

Технічна реалізація: Реалізується як скрипт MonoBehaviour, прикріплений до об'єкта комбайна. Він містить змінні для зберігання поточного та максимального об'єму врожаю, а також, можливо, типів культур, що можуть зберігатися.

Система візуалізації (HarvesterVisuals / FieldVisuals):

Цей модуль є відповідальним за візуальне представлення всіх динамічних змін в механіці збирання врожаю. Він буде ключовим для інтеграції "Vortex графіки" в проєкт.

Основне призначення: SystemVisuals забезпечує оновлення візуального стану комбайна (наприклад, анімації рухомих частин жатки, обертання шнека для вивантаження зерна), поля (зміна текстури або моделі рослинності на

зібраних ділянках), а також генерацію реалістичних та стилізованих візуальних ефектів. Зокрема, для "Vortex графіки" він буде відповідати за створення динамічних, можливо, абстрактних, візуальних шаблонів, що супроводжують рух техніки та взаємодію з полем. Це може включати ефекти потоку, вихорів, або деформації світла/кольору, що підкреслюють енергію процесу.

Взаємодія: Отримує дані про поточний стан та зміни від CombinePhysics (для руху комбайна), HarvesterHead (для активації ефектів зрізання) та FieldManager (для оновлення візуалізації поля). Він активно використовує вбудовані можливості Unity, такі як Particle Systems[10] (зокрема, VFX Graph[10] для складних візуальних ефектів), системи шейдерів (Shaders через Shader Graph[10] для кастомізації рендерингу поверхні поля та техніки), а також компоненти рендерингу (Mesh Renderers, Terrain[12]). Використання Universal Render Pipeline (URP) або High Definition Render Pipeline (HDRP) буде обов'язковим для досягнення бажаного візуального стилю та контролю над рендерингом.

Технічна реалізація: Може бути розділений на кілька окремих MonoBehaviour скриптів (наприклад, HarvesterVisuals для комбайна та FieldVisuals для поля), прикріплених до відповідних ігрових об'єктів. Їхня логіка виконується в методах Update() або LateUpdate(). Основна робота над "Vortex графікою" буде здійснюватися через написання користувацьких шейдерів в Shader Graph (наприклад, для анімації текстур поля, що імітують потоки енергії або зміни стану) та створення комплексних візуальних ефектів у VFX Graph (для стилізованих вихорів пилу, зерна, що вивантажується, або ефектів енергії навколо жатки).

Принципи взаємодії компонентів:

Взаємодія між цими модулями побудована на принципі слабкого зв'язку та інкапсуляції. Це означає, що кожен модуль відповідає за свій конкретний набір функцій і взаємодіє з іншими через чітко визначені інтерфейси або публічні методи, не вникаючи у внутрішню реалізацію інших компонентів.

Типовий потік взаємодії виглядає наступним чином:

Гравець подає команди керування (наприклад, "їхати вперед", "повернути", "опустити жатку") через CombineController.

CombineController перетворює ці команди на фізичні параметри та передає їх до CombinePhysics, який потім здійснює рух комбайна, використовуючи Rigidbody та WheelCollider.

HarvesterHead, будучи дочірнім об'єктом комбайна, рухається разом з ним. Він постійно відстежує свою позицію над ігровим полем.

Коли жатка активована та знаходиться над ділянкою з дозрілим врожаєм, HarvesterHead надсилає запит до FieldManager для отримання інформації про стан цієї ділянки.

Якщо умови для збирання виконані (врожай дозрів і ще не зібраний), HarvesterHead ініціює процес збору, сповіщаючи FieldManager про необхідність оновити стан ділянки (наприклад, позначити її як зібрану).

FieldManager оновлює свої внутрішні дані та сповіщає SystemVisuals про те, що візуалізацію поля на певній ділянці потрібно змінити (наприклад, замінити текстуру дозрілого врожаю на текстуру зібраного поля, можливо, з додаванням ефектів "Vortex графіки" для підкреслення зміни).

Паралельно, зібраний врожай передається до CombineStorage, який обліковує його об'єм.

SystemVisuals також використовує дані про рух комбайна та стан жатки для активації динамічних візуальних ефектів. Зокрема, це можуть бути стилізовані VFX Graph[10] ефекти пилу, зерна або "енергетичних вихорів", що підкреслюють роботу машини та перетворення врожаю.

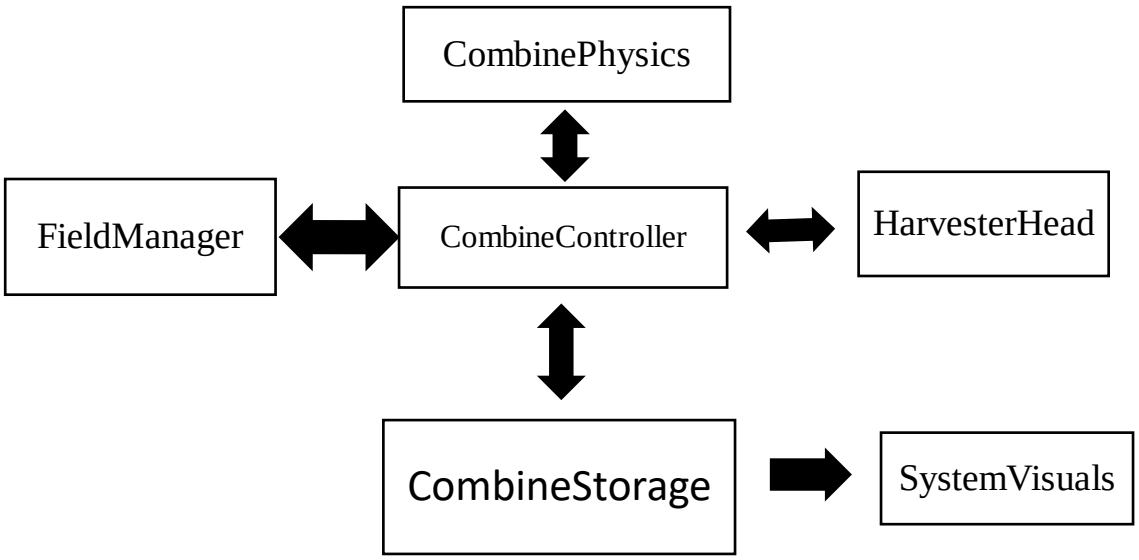

Рис. 2.1 - Діаграма компонентів системи механіки збирання врожаю

Така архітектура дозволяє чітко розділити відповідальності, робить систему більш керованою та передбачуваною, а також спрощує подальше розширення та внесення змін без порушення загальної стабільності. Це є критично важливим для великих ігрових проєктів[5], де різні команди можуть працювати над різними аспектами гри одночасно.

2.2 Алгоритми реалістичної фізики руху комбайна

Досягнення достовірного відтворення динаміки руху великовагової сільськогосподарської техніки, зокрема комбайна, є одним із ключових аспектів для забезпечення глибокого та переконливого занурення гравця у віртуальний світ [8]. Реалістична поведінка машини, її адекватна реакція на керуючі дії оператора та фізична взаємодія з динамічно мінливим оточенням, таким як нерівності ґрунту або опір рослинності, мають вирішальне значення для ігрового досвіду. Цей аспект буде втілений переважно через інтегроване використання компонентів Rigidbody [10] та WheelCollider [11] в ігровому рушії Unity, доповнене розробкою спеціалізованих програмних сценаріїв, а також візуальними ефектами, що відповідають концепції "Vortex графіки".

1. Детальна конфігурація компонента WheelCollider:

Кожен елемент ходової частини комбайна, що підлягає фізичній симуляції (тобто кожне колесо), вимагає окремого прикріплення компонента WheelCollider [11]. Цей компонент, будучи суто математичною моделлю, а не візуальним об'єктом, слугує мостом між візуальною репрезентацією колеса та фізичним світом Unity. Його точне та виважене налаштування є вирішальним фактором для забезпечення достовірної та стабільної поведінки всього транспортного засобу.

Mass (Маса): Хоча загальна вага комбайна визначається кореневим компонентом Rigidbody [10], WheelCollider [11] використовує власні внутрішні параметри, що враховують масу окремого колеса та елементів підвіски. Коректне співвідношення цих мас має прямий вплив на інерційні характеристики окремих коліс та загальну стійкість машини під час руху та маневрування.

Radius (Радіус): Це значення задає віртуальний радіус фізичного колеса. Воно повинно максимально точно відповідати радіусу візуальної 3D-моделі колеса, забезпечуючи ідеальну синхронізацію між фізичною поведінкою та візуальним відображенням.

Suspension Distance (Амплітуда ходу підвіски): Визначає максимальну вертикальну дистанцію, на яку колесо може відхилятися від початкового положення відносно основної рами комбайна. Довгий хід підвіски дозволяє ефективніше поглинати великі топографічні нерівності, тоді як обмежений хід створює відчуття більш "жорсткої" та менш адаптивної машини.

Параметри системи підвіски (Suspension Spring):

Spring (Жорсткість пружини): Регулює пружні властивості підвіски. Вищі значення надають підвісці більшої жорсткості, забезпечуючи швидше повернення колеса у вихідну позицію, але зменшуючи здатність поглинати ударні навантаження. Менші значення роблять підвіску більш "м'якою".

Damper (Демпфування): Визначає інтенсивність гасіння коливань підвіски. Вищі значення демпфування дозволяють швидше стабілізувати рух

колеса після наїзду на перешкоду, запобігаючи надмірному розгойдуванню комбайна. Недостатнє демпфування може призвести до небажаної "стрибучої" поведінки.

Target Position (Цільове положення): Вказує бажану нормалізовану позицію підвіски (в діапазоні від 0 до 1, де 0 означає повне стиснення, а 1 – повне розтиснення). Цей параметр допомагає підтримувати комбайн на оптимальній висоті над поверхнею ґрунту.

Криві зчеплення шин (Wheel Friction Curve): Ці криві мають вирішальне значення для симуляції реалістичної взаємодії шин з поверхнею ґрунту. Вони графічно описують нелінійну залежність сили тертя між колесом та поверхнею від ступеня ковзання шини.

Forward Friction Curve (Крива поздовжнього тертя): Моделює зчеплення шини при прискоренні або гальмуванні. Якщо ковзання перевищує певний поріг (пробуксовка при розгоні або блокування колеса при гальмуванні), сила тертя починає падати, і комбайн втрачає зчеплення.

Sideways Friction Curve (Крива поперечного тертя): Описує зчеплення при поворотах. Якщо бічне ковзання стає надмірним (знесення осі), комбайн втрачає керованість, входячи в занос.

Кожна крива характеризується чотирма ключовими точками: Extremum Slip (значення ковзання, при якому сила тертя досягає свого максимуму), Extremum Value (максимальне значення сили тертя), Asymptote Slip (значення ковзання, при якому сила тертя наближається до асимптотичного рівня) та Asymptote Value (значення сили тертя на асимптоті). Ретельне експериментування з цими кривими дозволить досягти достовірної поведінки комбайна на різних типах ґрунту, що мають унікальні коефіцієнти зчеплення (наприклад, зменшене зчеплення на вологому, пухкому ґрунті або збільшене на твердому дорожньому покритті).

2. Обробка вводу користувача та застосування динамічних сил:

Основний програмний сценарій CombinePhysics.cs відіграватиме ключову роль в інтерпретації вхідних керуючих сигналів (які надходять від

CombineController) та їх подальшого перетворення на відповідні фізичні впливи, що застосовуються до компонентів WheelCollider [11]. Усі операції, пов'язані з фізичними розрахунками, повинні обов'язково виконуватися в методі FixedUpdate() [10], щоб забезпечити повну синхронізацію з фізичним двигуном Unity.

Моторний момент (motorTorque): Для ініціації та підтримки руху, керуючий сигнал від гравця (наприклад, значення в діапазоні від -1 до 1, отримане через Input.GetAxis("Vertical"), що відображає інтенсивність натискання педалі газу або гальма) буде пропорційно масштабуватися на величину максимальної потужності двигуна комбайна (maxMotorPower). Отриманий результуючий момент буде застосовуватися виключно до тих коліс, які визначені як ведучі в конфігурації транспортного засобу (це можуть бути колеса однієї осі або всіх осей у випадку повного приводу). Фрагмент коду для Моторного моменту в Додатку А.

Гальмівний момент (brakeTorque): При активації гальма (наприклад, шляхом утримання клавіші Space) або у випадку відсутності вхідного сигналу газу/заднього ходу, до всіх коліс комбайна буде застосовуватися гальмівний момент. Ця величина масштабується відповідно до параметра brakePower. Фрагмент коду для Гальмівного моменту в Додатку Б.

Керування поворотом (steerAngle): Керуючий сигнал для зміни напрямку руху (наприклад, Input.GetAxis("Horizontal"), що відповідає повороту керма) масштабується на максимально допустимий кут повороту коліс (maxSteerAngle) і застосовується до тих коліс, які визначені як керовані. Зазвичай, для комбайнів це передні колеса, однак для деяких моделей або спеціалізованих машин можуть використовуватися обидві осі або шарнірно-зчленована конструкція. Фрагмент коду для Керування повороту в Додатку В.

3. Симуляція інерції та маси:

Моделювання інерційних властивостей великогабаритного комбайна має вирішальне значення для забезпечення його достовірної динаміки [8]. Компонент Rigidbody [10], який прикріплений до кореневого ігрового об'єкта

комбайна, буде налаштований з достатньо великим значенням mass (наприклад, в діапазоні від 15000 до 25000 кілограмів, залежно від типу комбайна). Це автоматично позначиться на:

Характеристиках розгону: Комбайн набиратиме швидкість поступово, демонструючи відчутну "інерцію", що відповідає поведінці реальної важкої машини.

Ефективності гальмування: Для повної зупинки комбайна знадобиться певна відстань, що обумовлено його значною масою та інерцією.

Особливостях маневрування: Машина буде неохоче змінювати свій лінійний чи кутовий напрямок руху. Різкі повороти на високій швидкості можуть призвести до значного крену, втрати контролю або навіть до потенційного перекидання, що додає грі виклику та реалізму.

Додатково, параметри drag (лінійний опір) та angularDrag (кутовий опір) на Rigidbody можуть бути ретельно налаштовані для імітації впливу опору повітря та внутрішнього тертя. Ці параметри посилять відчуття "ваги" та загальної достовірності фізичної поведінки комбайна.

4. Взаємодія з ландшафтом та інтеграція "Vortex графіки" для візуалізації слідів:

Реалістична взаємодія комбайна з поверхнею поля виходить за межі простої фізики підвіски; вона також охоплює динамічні візуальні зміни на поверхні ґрунту, що можуть бути посилені стилізацією "Vortex графіки".

Реакція на топографічні нерівності: Компонент WheelCollider [11] автоматично та ефективно взаємодіє з Terrain Collider або іншими колайдерами, що представляють поверхню ігрового світу. Правильно налаштована система підвіски дозволить комбайну достовірно "підстрибувати", "гойдатися" та "нахилятися" при наїзді на купини, вибоїни або ділянки з різким схилом. Це значно посилює тактильне відчуття від керування важкою машиною.

Генерація стилізованих слідів та візуальних ефектів "Vortex графіки": Після проходу комбайна, на поверхні поля залишатимуться візуальні сліди від

коліс, які будуть інтегровані з концепцією "Vortex графіки". Це може бути реалізовано кількома способами, що дозволяють посилити візуальний відгук на фізичну взаємодію:

Система декалей (Decal Projectors) з "Vortex" текстурами: Замість стандартних текстур слідів, можуть бути використані стилізовані текстури, що імітують енергетичні потоки, вихори або легкі спотворення простору, проектуючись на поверхню ґрунту в місці контакту колеса. Це створюватиме відчуття, що комбайн залишає не просто сліди, а "відбитки енергії".

Динамічна зміна текстури террейну з шейдерними ефектами: Цей підхід є більш складним, але забезпечує найвищий рівень інтеграції. Він полягає у програмній модифікації alphamap компонента Terrain. Коли колесо контактує з певною ділянкою террейну, скрипт обчислює відповідні UV-координати та оновлює вагові коефіцієнти для текстурних шарів террейну. Наприклад, у місці проходу колеса збільшується вага текстури "втоптаної землі" або "бруду", а також може бути активований спеціальний шейдерний ефект. Цей шейдер (розроблений за допомогою Shader Graph або написаний вручну) може динамічно деформувати або анімувати текстуру, створюючи легкі вихорові візерунки, ефекти "пульсації" або "зміщення" на зім'ятій поверхні. Для великих полів важливо оптимізувати цей процес, оновлюючи лише невеликі, цільові ділянки alphamap навколо контактних точок коліс та, можливо, групуючи кілька оновлень для зменшення обчислювального навантаження.

Системи частинок (Particle Systems та VFX Graph) з "Vortex" стилізацією: Додаткові ефекти, такі як пил, що здіймається від коліс, або бризки бруду, можуть бути відтворені за допомогою Particle Systems. Ці системи можуть бути стилізовані під "Vortex графіку": наприклад, пил може утворювати мініатюрні вихори, або частинки можуть мати незвичайні траєкторії, що імітують енергетичні потоки. Для більш складних та високопродуктивних ефектів буде залучений VFX Graph, який дозволить створювати тисячі частинок, що формують динамічні вихори, потоки енергії або абстрактні візуальні спотворення, що виникають при інтенсивній взаємодії

коліс з ґрунтом (наприклад, при пробуксовці або різкому гальмуванні). Інтенсивність цих ефектів може динамічно регулюватися залежно від швидкості руху, ступеня ковзання та типу поверхні.

5. Моделювання повного приводу (AWD) та блокування диференціалів (опційно):

Для подальшого підвищення достовірності та розширення функціональних можливостей комбайна, може бути імплементована більш складна система передачі потужності.

Повний привід (All-Wheel Drive - AWD): Моторний момент може бути інтелектуально розподілений між усіма колесами комбайна, а не лише однією ведучою віссю. Це значно підвищує прохідність машини на складних ділянках з низьким зчепленням та суттєво покращує загальне зчеплення з поверхнею.

Диференціали та їх блокування: Реальні комбайни оснащені диференціалами, які дозволяють колесам на одній осі обертатися з різною швидкістю під час повороту, запобігаючи надмірному проковзуванню. В ігровій симуляції це може бути досягнуто шляхом динамічного регулювання motorTorque для кожного колеса на осі, виходячи з різниці їхніх кутових швидкостей або кута повороту. Функція блокування диференціала, яка рівномірно розподіляє момент між колесами, може бути додана для підвищення прохідності в складних умовах (наприклад, на дуже брудних ділянках), супроводжуючись стилізованими "Vortex" візуальними індикаторами (наприклад, тимчасове посилення світіння навколо коліс або абстрактні енергетичні потоки, що візуалізують перерозподіл потужності).

6. Базова система пошкоджень та зносу з візуальним фідбеком "Vortex графіки" (опційно):

Для додавання елементів реалізму, що впливають на стратегічну та економічну складову гри, може бути імплементована базова система зносу та отримання пошкоджень технікою.

Знос компонентів: Цифровий параметр "зносу" (від 0% до 100%) може бути присвоєний ключовим структурним та функціональним компонентам комбайна (наприклад, двигун, жатка, трансмісія, ходова частина). Цей показник прогресивно збільшується з часом активної експлуатації, інтенсивністю використання (наприклад, тривала робота на максимальній потужності) або внаслідок фізичних пошкоджень (зіткнення з перешкодами).

Вплив пошкоджень та візуальний фідбек: Високий рівень зносу або критичні пошкодження можуть негативно впливати на експлуатаційні характеристики комбайна: зменшувати максимальну потужність двигуна, знижувати ефективність гальмування, збільшувати витрату палива або навіть призводити до тимчасової зупинки роботи. Цей процес може супроводжуватися не тільки традиційними візуальними ефектами (іржа, бруд, подряпини на текстурі моделі; поява диму, іскор), але й стилізованими ефектами "Vortex графіки". Наприклад, критичні пошкодження можуть проявлятися як візуальні "глюки" у текстурах, мерехтливі "енергетичні щити" навколо пошкоджених ділянок, або абстрактні візуалізації "витоку енергії" за допомогою VFX Graph, що підкреслюють несправність без надмірної деталізації реальних руйнувань.

Звукові індикатори: Знос та пошкодження також можуть супроводжуватися відповідними звуковими ефектами (скрегіт, стукіт, нерівномірна робота двигуна), що посилює відчуття зносу та спонукає гравця до своєчасного ремонту та обслуговування техніки.

Загалом, комбінація цих алгоритмів фізичної симуляції, ретельна конфігурація фізичних компонентів Unity та інтеграція динамічних візуальних ефектів у стилі "Vortex графіки" дозволить створити достовірну, захоплюючу та візуально унікальну модель комбайна, яка реалістично взаємодіятиме з ігровим світом "Song of the Fields", підкреслюючи динаміку та енергію сільськогосподарської праці.

2.3 Алгоритми взаємодії комбайна з полем та збирання врожаю

Цей підрозділ присвячений серцевині механіки збирання врожаю – програмному забезпеченню, що контролює перетворення дозрілого врожаю на зібраний продукт та його подальше транспортування. Ефективна та візуально переконлива взаємодія комбайна з ігровим полем є вирішальним фактором для створення глибокого занурення гравця. Вона охоплює не лише логіку збирання, але й динамічні візуальні та аудіо-ефекти, які підкреслюють енергію та масштаб сільськогосподарської діяльності, зокрема, за допомогою інтеграції "Vortex графіки".

1. Моделювання стану ігрового поля:

Для ефективного управління інформацією про стан великих віртуальних полів, буде застосовано структурований підхід на основі сітки. Це дозволяє точно відстежувати стан кожної ділянки поля та динамічно реагувати на дії комбайна.

Сіткова структура (Grid-based approach): Ігрове поле буде логічно розділене на рівномірну двовимірну сітку (наприклад, з розміром клітинок 1x1 метр або 2x2 метри, залежно від бажаної деталізації та оптимізації). Кожна така клітинка сітки (або "сегмент поля") буде інкапсулювати об'єкт або структуру даних, що містить всю необхідну інформацію про дану ділянку.

Структура даних для клітинки поля (CellInfo): Кожна клітинка сітки зберігатиме наступні атрибути:

CropType (Тип культури): Перелічувальний тип (enum), що ідентифікує вид агрокультури, вирощеної на цій ділянці (наприклад, Пшениця, Кукурудза, Соняшник, Ріпак). Це дозволить комбайну взаємодіяти з різними культурами, кожна з яких може мати унікальні властивості.

GrowthStage (Стадія росту): Цілочисельне значення або інший перелічувальний тип, що відображає поточну фазу розвитку рослини (наприклад, Посів, Сходи, Вегетація, Дозрівання, Зібрано). Збирання можливе лише для культур, що досягли стадії "Дозрівання".

IsHarvested (Чи зібрано): Булеве значення, що вказує, чи була ця конкретна ділянка поля вже зібрана. Це запобігає повторному збору врожаю з однієї й тієї ж клітинки та дозволяє коректно відображати візуальний стан поля.

YieldMultiplier (Коефіцієнт врожайності): Числове значення (float), що може динамічно змінюватись залежно від ігрових факторів (наприклад, якість ґрунту, внесення добрив, вплив погодних умов). Цей коефіцієнт буде прямо впливати на кількість зібраного врожаю з даної клітинки.

CropVisualReference (Посилання на візуальний об'єкт/стан): Опційне посилання на ігровий об'єкт, що представляє візуальну модель рослини для цієї клітинки (якщо використовується підхід з окремими об'єктами), або індекс у текстурному атласі, або параметр для шейдера, що контролює візуалізацію.

Візуалізація стану поля: Відображення стану поля може бути реалізовано кількома методами:

Індивідуальні 3D-моделі рослин: Кожна клітинка може містити посилання на ігровий об'єкт (префаб) рослини для візуалізації. Після збирання, цей об'єкт деактивується або знищується. Хоча це забезпечує високу деталізацію, для великих полів цей підхід є вкрай неоптимальним через велику кількість Draw Calls.

Змішування текстур террейну (Alphamap Blending): Це більш ефективний та поширений метод для Unity [12]. Компонент Terrain дозволяє "малювати" на своїй поверхні кілька текстурних шарів (наприклад, "ґрунт", "дозрілий врожай", "зібране поле"). Після збирання, програмно оновлюється alphamap террейну (маска змішування текстур), що призводить до плавного переходу від текстури дозрілого врожаю до текстури зібраного поля або ґрунту. Це створює враження, що комбайн "косить" поле.

Інтеграція "Vortex графіки" для переходу: Щоб посилити візуальний ефект, перехід між текстурами на alphamap може бути не просто плавним, а й мати стилізований "Vortex" ефект. Це досягається за допомогою користувацького шейдера (розробленого в Shader Graph [10]), який

застосовується до террейну. Під час зміни alphamap на зібраній ділянці, шейдер може анімувати вихрові або хвильові візерунки, що розповсюджуються від точки контакту жатки. Це може виглядати як енергетичне "завихрення", що візуалізує акт збирання.

GPU Instancing для рослинності: Для оптимізованого рендерингу тисяч або мільйонів об'єктів рослинності використовується GPU Instancing. FieldManager може контролювати видимість цих інстансів: коли клітинка зібрана, відповідні інстанси рослин просто виключаються з рендерингу, або їхній масштаб анімується до нуля.

Приклад псевдокоду для FieldManager в Додатку Г.

2. Алгоритм зрізання рослин (HarvesterHead):

Модуль жатки є активним виконавцем процесу збирання. Він постійно перевіряє, які ділянки поля знаходяться в його робочій зоні, та ініціює їхній збір.

Визначення зони зрізання: HarvesterHead матиме прикріплений дочірній BoxCollider (або кілька BoxCollider), налаштований як тригер (Is Trigger = true). Цей колайдер повинен точно відображати робочу ширину жатки. Як альтернатива, можна використовувати Physics.OverlapBox() або Physics.OverlapSphere() в методі FixedUpdate для програмного визначення клітинок, що потрапляють у зону жатки.

Тригерна взаємодія: Коли тригер-колайдер жатки перетинається з колайдером, що представляє сегмент поля (або з колайдером самого террейну), спрацьовує метод OnTriggerStay() [10]. Цей метод буде викликатися кожен фіксований кадр, доки колайдери залишаються у контакті.

Перевірка стану та збирання: Усередині OnTriggerStay або в логіці, що використовує OverlapBox, жатка отримуватиме список клітинок поля, що перебувають безпосередньо під нею. Для кожної виявленої клітинки буде здійснюватися комплексна перевірка:

Чи досяг врожай на цій ділянці стадії "Дозрівання" (GrowthStage == GrowthStage.Mature).

Чи не була ця конкретна ділянка поля вже зібрана (!IsHarvested).

Якщо обидві умови виконані, HarvesterHead ініціює процес збирання. Він викликає метод FieldManager.MarkCellAsHarvested() для оновлення стану клітинки та передачі інформації про її візуальне оновлення. Потім розраховується кількість зібраного врожаю для цієї клітинки (наприклад, базове значення врожаю * YieldMultiplier клітинки) і ця кількість передається до CombineStorage.

Візуальні ефекти "Vortex графіки" під час зрізання: У момент "зрізання" врожаю з клітинки, HarvesterHead може активувати локальні ефекти "Vortex графіки" за допомогою VFX Graph. Це можуть бути короткочасні вихори частинок навколо ріжучого механізму жатки, що візуалізують "поглинання" врожаю, або спалахи "енергії", що проходять по лезах, підкреслюючи їхню роботу. Інтенсивність та колір цих ефектів можуть змінюватися залежно від типу врожаю або швидкості збирання.

Приклад псевдокоду для HarvesterHead в Додатку Д.

3. Накопичення врожаю та система бункера (CombineStorage):

Модуль CombineStorage є центральним сховищем для зібраної продукції.

Відстеження об'єму: Він відповідає за відстеження поточного об'єму зерна (currentCropAmount) та його максимальної місткості (maxCropCapacity).

Логіка завантаження/вивантаження: При отриманні інформації про зібраний врожай від HarvesterHead, модуль CombineStorage додає його до поточного об'єму. Важливо імплементувати перевірку на переповнення: якщо бункер досягає максимальної місткості, жатка не зможе збирати більше врожаю, доки бункер не буде вивантажений. Це може бути реалізовано через повернення булевого значення (false) від CanAddCrop методу.

Вивантаження: Передбачити функціонал для вивантаження зерна, наприклад, у причіп або стаціонарне сховище на фермі. Цей процес також може

супроводжуватися візуальними ефектами, такими як потік зерна з вивантажувального шнека, можливо, стилізований з елементами "Vortex графіки" (наприклад, іскристі частинки або легке спотворення повітря навколо потоку).

4. Візуалізація зрізаних ділянок та динамічні ефекти "Vortex графіки":

Візуальний відгук є критично важливим для розуміння гравцем процесу збирання та підвищення занурення.

Зникнення рослинності: Якщо рослини представлені окремими 3D-моделями (що, як згадувалося, неоптимально для великих полів), то при збиранні їх можна деактивувати, знищити, або повернути в об'єктний пул. Якщо використовується GPU Instancing, то зникнення досягається шляхом видалення ID інстанса з буфера рендерингу або анімацією його масштабу до нуля.

Зміна текстури поля (Terrain.SetAlphamaps()): Після збирання, відповідний шар текстури террейну (наприклад, "зібране поле" або "ґрунт") програмно "намальовується" на ділянці, де пройшла жатка. Це створює візуальний ефект "викошеного" поля.

Впровадження "Vortex графіки" у зміну текстури: Це може бути реалізовано через параметри, що передаються до користувацького шейдера террейну. При оновленні alphamap для певної зони, шейдер може на короткий час активувати анімований ефект вихору або "розмиття", який поступово згасає, залишаючи після себе чітку текстуру зібраного поля. Це підкреслює дію, що відбулася.

Системи частинок (Particle Systems [10] та VFX Graph [10]): Це основний інструмент для створення динамічних візуальних ефектів.

Пил та солома: Particle System біля коліс комбайна та під жаткою, що активуються під час руху та збирання. Ці частинки можуть мати стилізовані кольори або траєкторії, що імітують мініатюрні вихори, відповідно до "Vortex графіки".

Зерно: Particle System на вивантажувальному шнеку та, можливо, біля жатки, що імітує потік зерна. Частинки зерна також можуть мати легке світіння або сліди, що підкреслюють їхній рух, створюючи "енергетичний потік".

Енергетичні вихори/спалахи: Використання VFX Graph [10] дозволить створювати високоякісні, стилізовані ефекти. Наприклад, короткочасні "вихори енергії" або "спалахи світла" можуть з'являтися навколо ріжучих елементів жатки в момент збирання, або "енергетичні хвилі" можуть розповсюджуватися від комбайна, що рухається по полю, візуалізуючи його вплив на середовище.

Звуковий супровід: Додавання AudioSource до комбайна для відтворення звуків двигуна (змінюються залежно від навантаження та швидкості), роботи жатки (шум ріжучих елементів, подрібнення соломи) та процесу вивантаження зерна. Ці звуки, поєднані з візуальними ефектами, забезпечують повний імерсивний досвід.

Інтеграція цих алгоритмів та візуальних елементів "Vortex графіки" створює не просто функціональну механіку, а захоплюючий та візуально унікальний досвід збирання врожаю у грі "Song of the Fields".

Висновок до Розділу 2

У другому розділі кваліфікаційної роботи було проведено аналіз та обґрунтування вибору інструментальних засобів та технологій для розробки ігрового проекту "Song of the Fields", а також розкрито ключові аспекти реалізації його механік. Детальне вивчення функціональних можливостей ігрового рушія Unity підтвердило його оптимальність для створення симулятора з реалістичною фізикою та складною системою взаємодії об'єктів. Зокрема, було підкреслено ефективність використання компонентів Rigidbody для управління динамікою комбайна та WheelCollider для відтворення реалістичної поведінки колісної техніки на різних типах поверхонь.

Особливу увагу приділено методам виявлення зіткнень та взаємодії об'єктів, акцентуючи на використанні тегів (Tags) та шарів (Layers) як надійного способу фільтрації та ідентифікації об'єктів у сцені, що є критично важливим для коректної роботи механіки збирання врожаю. Також було розглянуто принципи візуалізації ефектів, таких як "Vortex графіка", що дозволяє збагатити ігровий досвід візуальною складовою процесу збирання.

Таким чином, у цьому розділі було сформовано необхідну теоретичну та інструментальну базу, яка стане основою для практичної реалізації основних механік ігрового процесу в наступних розділах роботи.

РОЗДІЛ 3 Реалізація механіки збирання врожаю в Unity

3.1 Створення ігрового середовища та моделювання об'єктів

Етап практичної реалізації механіки збирання врожаю в середовищі Unity починається з ретельної підготовки віртуального простору та створення всіх необхідних тривимірних асетів. Це базовий крок, від якості виконання якого безпосередньо залежить подальша ефективність фізичних симуляцій, візуального сприйняття та загальна продуктивність ігрового проєкту. Правильно структуроване середовище та оптимізовані моделі є фундаментом для інтеграції складних механік та візуальних ефектів, включаючи концепцію "Vortex графіки".

1. Налаштування проекту Unity:

Первинне налаштування проекту є визначальним для всієї подальшої розробки.

Ініціалізація проекту: Процес розпочинається зі створення нового тривимірного (3D) проекту в Unity Hub [10].

Вибір Render Pipeline (Конвеєра Рендерингу): Вибір архітектури рендерингу є критично важливим для досягнення бажаного візуального стилю, особливо для "Vortex графіки". Для даної роботи рекомендовано використання Universal Render Pipeline (URP) або High Definition Render Pipeline (HDRP) [10]. URP пропонує чудовий баланс між продуктивністю та візуальною якістю, будучи гнучким для стилізованих ефектів. HDRP, у свою чергу, надає максимальні можливості для фотореалістичної графіки та складних пост-процес ефектів, які можуть бути адаптовані для "Vortex" стилізації. Вибір одного з них дозволить розробляти користувацькі шейдери та VFX (Visual Effects) на найвищому рівні.

Імпорт базових асетів: До проекту імпортуються всі необхідні вихідні матеріали: текстури ґрунту, рослинності, базові тривимірні моделі комбайна, жатки та інших сільськогосподарських елементів. Перевага надається

оптимізованим форматам, таким як .fbx (для 3D-моделей з анімаціями та ієрархіями) або .obj (для статичних моделей).

Організація файлової структури: Для забезпечення зручності розробки та подальшого супроводу проекту, необхідно створити чітку та логічну структуру папок (наприклад, Assets/Models, Assets/Materials, Assets/Textures, Assets/Scripts, Assets/Scenes, Assets/Prefabs, Assets/VFX, Assets/Shaders).

2. Моделювання комбайна та жатки:

Якість та структура тривимірних моделей комбайна та жатки безпосередньо впливають на фізичну симуляцію, візуалізацію та можливість інтеграції динамічних ефектів.

Оптимізовані 3D-моделі: Отримання або створення високоякісних, але водночас оптимізованих 3D-моделей комбайна та жатки. Ключові аспекти оптимізації включають:

Оптимальна кількість полігонів (LOD - Level of Detail): Створення кількох версій моделі з різною деталізацією (високополігональна для близького плану, середньополігональна, низькополігональна для далекого плану). Це дозволить рушію автоматично перемикатися між ними залежно від відстані до камери, суттєво зменшуючи навантаження на рендеринг.

Правильна топологія: Чиста та рівномірна сітка моделі сприяє коректній деформації при анімаціях, правильному нанесенню текстур та ефективній роботі фізичних колайдерів.

UV-розгортка (UV Mapping): Якісна UV-розгортка є обов'язковою для коректного та ефективного нанесення текстур та матеріалів на модель.

Поділ моделі на компоненти: Модель комбайна повинна бути розділена на окремі, логічно відокремлені частини (корпус, колеса, жатка, вивантажувальний шнек, рухомі елементи). Це дозволить незалежно анімувати їх, прикріплювати окремі фізичні компоненти та застосовувати унікальні шейдери або ефекти "Vortex графіки" до конкретних частин.

Підготовка моделей для "Vortex графіки": Для ефективної інтеграції стилізованих візуальних ефектів, 3D-моделі можуть вимагати спеціальної підготовки:

Додаткові UV-канали: Створення додаткових UV-каналів, які можуть використовуватися для зміщення координат текстур у шейдері, створюючи ефекти "пульсації" або "зміщення", що характерні для "Vortex графіки".

Vertex Colors (Кольори вершин): Використання кольорів вершин для маскування або керування інтенсивністю "Vortex" ефектів на різних частинах моделі. Наприклад, краї жатки можуть мати особливий колір вершин, щоб там активувався ефект світіння або вихору.

Сепарація для ефектів: Розділення деяких елементів (наприклад, ріжучих дисків жатки або вентиляційних отворів) на окремі меші, щоб до них можна було застосувати унікальні VFX Graph ефекти.

Ієрархія ігрових об'єктів Unity: Правильно побудована ієрархія об'єктів у сцені є фундаментальною для коректної роботи фізики, анімацій та взаємодії між компонентами. Для комбайна рекомендована наступна структура:

Combine_Root (Основний об'єкт комбайна): До нього прикріплюється компонент Rigidbody [10] та основний скрипт CombinePhysics. Це кореневий об'єкт, що рухається та взаємодіє фізично.

Body (Візуальна частина корпусу): Дочірній об'єкт, що містить Mesh Renderer та Mesh Filter для візуалізації корпусу.

Wheels_Parent (Порожній об'єкт-контейнер для коліс): Дочірній до Combine_Root, слугує для організації та керування усіма колесами.

Wheel_Front_Left (і аналогічно для інших коліс): Кожен об'єкт колеса містить WheelCollider [11] (фізичну модель колеса) та дочірній об'єкт з Mesh Renderer для візуальної 3D-моделі колеса. Це дозволяє фізиці WheelCollider керувати позицією та обертанням візуального колеса.

Wheel_Front_Right

Wheel_Rear_Left

Wheel_Rear_Right

HarvesterHead_MountPoint (Порожній об'єкт-кріплення для жатки): Служить як точка прив'язки для жатки. До нього можна прикріпити скрипт, що відповідає за підняття/опускання жатки.

HarvesterHead (Об'єкт жатки): Дочірній до HarvesterHead_MountPoint. Містить власний Mesh Renderer для візуалізації жатки, колайдери-тригери для збирання врожаю та скрипт HarvesterHead.

UnloadPipe (Вивантажувальний шнек): Дочірній об'єкт, що містить Mesh Renderer та, можливо, анімаційний компонент для розгортання/згортання шнека та активації візуальних ефектів потоку зерна (VFX Graph [10]).

Встановлення Центру Мас: Правильне визначення та встановлення центру мас (Rigidbody.centerOfMass) для кореневого об'єкта комбайна є абсолютно критичним для досягнення стабільної та реалістичної фізики [8]. Для комбайна, як правило, центр мас має бути розташований відносно низько над землею і дещо ближче до передньої частини (або середини), щоб забезпечити стійкість при поворотах та нахилах, а також під час руху по нерівностях. Неправильне налаштування може призвести до небажаних перекидань або нестабільної поведінки.

3. Налаштування компонентів Rigidbody та Collider:

Після моделювання та створення ієрархії, наступним кроком є конфігурація фізичних властивостей об'єктів.

Конфігурація Rigidbody: До кореневого об'єкта Combine_Root обов'язково додається компонент Rigidbody [10]. Його основні параметри налаштовуються:

Mass: Встановлюється реалістична маса комбайна (наприклад, від 15 до 25 тонн, тобто 15000-25000 кг).

Drag та Angular Drag: Ці значення регулюють опір лінійному та кутовому руху відповідно, імітуючи опір повітря та внутрішнє тертя. Їх потрібно налаштувати для відчуття "ваги" та плавного сповільнення.

Use Gravity: Активована.

Is Kinematic: Вимкнена, щоб комбайн керувався фізичним двигуном.

Налаштування колайдерів для комбайна:

Для основної частини корпусу комбайна можна використовувати комбінацію Box Collider для простих форм або один Mesh Collider з активованою опцією Convex (опуклий). Convex Mesh Collider є більш оптимізованим для зіткнень з іншими опуклими колайдерами, але може не ідеально повторювати складні увігнуті форми.

Альтернативно, можна використовувати кілька Box Collider для точного покриття об'єму корпусу, що є більш продуктивним, ніж не-Convex Mesh Collider.

Налаштування тригер-колайдера жатки: До об'єкта HarvesterHead додається Box Collider. Критично важливо активувати опцію Is Trigger [10] для цього колайдера. Це дозволить йому виявляти перетин з рослинністю або сегментами поля без спричинення фізичних зіткнень. Розмір цього Box Collider має точно відповідати робочій ширині та глибині зрізання жатки.

4. Створення тестового поля та інтеграція "Vortex графіки" в ландшафт:

Тестове середовище відіграє ключову роль у відладці та демонстрації функціоналу.

Компонент Terrain: Для створення великого та реалістичного ландшафту буде використано вбудований компонент Unity Terrain [12]. За його допомогою можна скульптувати висоти, створюючи пагорби, ями та рівні ділянки для полів.

Шари террейну (Terrain Layers) та текстури: Створюються кілька Terrain Layers з різними текстурами для представлення різних станів поля: "чистий ґрунт", "дозрілий врожай", "зібране поле", "бруд", "дорога".

Матеріали та шейдери для "Vortex графіки": Матеріал, що використовується для рендерингу террейну, повинен бути налаштований на використання користувацького шейдера (розробленого в Shader Graph [10] або написаного вручну), який здатен реагувати на параметри, що надсилаються з логіки гри. Цей шейдер може:

Відображати "Vortex" ефекти на зібраних ділянках (наприклад, легкі анімовані вихори або пульсації кольору, що поступово згасають).

Змінювати текстури поля з переходом, що супроводжується "енергетичними" сплесками.

Реагувати на стан ґрунту (наприклад, мокрий ґрунт може мати додатковий ефект мерехтіння або відблисків).

Деталі рослинності (Detail Objects) та оптимізація: Для візуалізації великої кількості рослинності на полях (трава, колоски), Unity Terrain пропонує "малювання" деталей. Однак для великих та густих полів цей метод може бути ресурсоємним. Оптимізація включає:

GPU Instancing: Застосування GPU Instancing для рендерингу мільйонів однакових екземплярів рослинності за один Draw Call. Це кардинально зменшує навантаження на процесор [10].

Системи LOD для рослин: Використання LOD для моделей рослин, зменшуючи їхню деталізацію на відстані.

Користувацькі шейдери для рослинності ("Vortex" рослини): Рослини можуть бути представлені не лише як статичні моделі, але й через користувацькі шейдери. Ці шейдери можуть дозволяти динамічно "анімувати" їхнє зникнення (розчинення в "енергетичному вихорі"), змінювати колір або прозорість, або навіть створювати легкі "пульсації" чи "світіння", що візуалізують "життєву енергію" врожаю, відповідно до концепції "Vortex графіки". При збиранні ці ефекти можуть динамічно реагувати, наприклад, "згортаючись" у точку або розсіюючись з яскравим спалахом.

"Billboards" (Біллборди): Для дуже далеких відстаней рослинність може бути замінена на біллборди (спрайти, що завжди дивляться на камеру), які ще більше зменшують навантаження.

Комплексний підхід до створення ігрового середовища та моделювання об'єктів, що враховує не лише фізичну достовірність, але й вимоги "Vortex графіки", закладає міцну основу для подальшої реалізації всіх механік та візуальних ефектів у грі "Song of the Fields".

3.2 Імплементація фізичної моделі комбайна

Практична реалізація достовірної фізичної моделі комбайна в середовищі Unity є центральним елементом, що забезпечує реалістичність та імерсію ігрового процесу. Вона вимагає ретельного налаштування вбудованих фізичних компонентів рушія та розробки програмних сценаріїв, які об'єднують введення користувача з фізичними розрахунками, а також інтегрують візуальні ефекти, включаючи аспекти "Vortex графіки", для посилення динаміки.

1. Конфігурація WheelCollider та зв'язок з візуальними моделями:

Для кожного фізично активного колеса комбайна в Unity Editor буде додано компонент WheelCollider [11]. Важливо, що WheelCollider – це суто фізичний елемент, який не має власного візуального представлення. Тому кожен WheelCollider повинен бути асоційований з відповідною 3D-моделлю колеса, що візуалізує його.

Структура в ієрархії: У дереві ієрархії сцени, кожен WheelCollider розміщується як дочірній об'єкт до основного Rigidbody комбайна. Візуальна 3D-модель колеса, у свою чергу, робиться дочірньою до порожнього об'єкта, на якому знаходиться WheelCollider. Це дозволяє WheelCollider керувати позицією та обертанням візуальної моделі.

Параметри підвіски:

Spring (Жорсткість пружини) та Damper (Демпфування): Ці параметри будуть налаштовуватися експериментально. Вищі значення Spring зроблять підвіску жорсткішою, що може бути реалістичним для важкої техніки, але знизить комфорт на нерівностях. Damper буде регулюватися для швидкого гасіння коливань без "стрибучості". Метою є досягнення відчуття важкого, але пружного руху.

Target Position: Встановлюється значення, що відповідає бажаній висоті підвіски в стані спокою.

Криві зчеплення шин (Wheel Friction Curve):

Для кожного колеса (або для групи коліс, наприклад, передніх та задніх) будуть створені та налаштовані Wheel Friction Curve для Forward (поздовжнього) та Sideways (поперечного) тертя [11].

Значення Extremum Slip/Value та Asymptote Slip/Value будуть ретельно підбиратися для імітації поведінки шин на ґрунті. Наприклад, Extremum Slip для ґрунту буде більшим, ніж для асфальту (потрібно більше проковзування для досягнення максимального зчеплення), а Asymptote Value буде нижчим (зчеплення швидко падає після максимуму). Це створить відчуття "пробуксовки" та "знесення" на м'якому ґрунті, що є характерним для сільськогосподарської техніки.

Код зв'язку візуалу та фізики в Додатку Е.

2. Обробка введення гравця та застосування динамічних сил:

Скрипт CombinePhysics.cs буде безперервно відстежувати введення від гравця та перетворювати його на відповідні сили, що впливають на WheelCollider компоненти. Всі ці операції виконуються в методі FixedUpdate() [10], який забезпечує стабільність фізичних розрахунків незалежно від частоти кадрів.

Введення: Використовуватиметься стандартний API введення Unity: Input.GetAxis("Vertical") для контролю прискорення/гальмування (газ та задній хід) та Input.GetAxis("Horizontal") для керування поворотами (кермо).

Моторний момент (motorTorque): Отримане значення Input.GetAxis("Vertical") буде масштабуватися на публічну змінну maxMotorPower та застосовуватися до motorTorque тих WheelCollider, які визначені як ведучі. Це імітує передачу потужності двигуна на колеса. Фрагмент коду для Моторного моменту в Додатку А.

Гальмівний момент (brakeTorque): При натисканні клавіші гальма (наприклад, KeyCode.Space) або автоматично при відсутності вводу газу, до всіх WheelCollider буде застосовуватися brakeTorque, масштабований на brakePower. Важливо забезпечити, щоб motorTorque обнулявся, коли

застосовується brakeTorque, щоб уникнути конфліктних сил. Фрагмент коду для Гальмівного моменту в Додатку Б.

Кут повороту (steerAngle): Значення Input.GetAxis("Horizontal") масштабується на maxSteerAngle і застосовується до steerAngle тих WheelCollider, які визначені як керовані (як правило, передні колеса). Фрагмент коду для Керування повороту в Додатку В.

3. Симуляція інерції та маси за допомогою Rigidbody:

Правильне налаштування компонента Rigidbody [10], прикріпленого до кореневого об'єкта комбайна, є фундаментальним для відчуття його маси та інерції.

Mass: Встановлюється високе значення маси (Rigidbody.mass), що відображає реальну вагу комбайна (наприклад, 15000-25000 кг). Це автоматично впливає на:

Динаміку розгону/гальмування: Комбайн набиратиме швидкість та сповільнюватиметься поступово, що відповідає поведінці важкої машини.

Поведінку на поворотах: Велика маса спричинить значну інерцію при поворотах, що вимагатиме від гравця завчасного планування маневрів та врахування радіусу повороту.

Drag (Лінійний опір) та Angular Drag (Кутовий опір): Ці параметри на Rigidbody будуть налаштовані для імітації опору руху та обертанню. Це може включати легкий опір повітря та внутрішнє тертя. Коректно налаштовані значення допоможуть комбайну плавно сповільнюватися, а не миттєво зупинятися, посилюючи відчуття його ваги.

Центр мас (centerOfMass): Хоча Unity автоматично розраховує центр мас, часто необхідно вручну змістити його (_rigidbody.centerOfMass = new Vector3(x, y, z); у Awake()). Для комбайна центр мас має бути відносно низько до землі та, можливо, трохи зміщений вперед, щоб покращити стабільність при русі по схилах та запобігти надмірному перекиданню.

4. Взаємодія з ландшафтом та інтеграція "Vortex графіки" для візуалізації слідів:

Візуальний відгук на взаємодію коліс комбайна з ґрунтом значно підвищує реалізм. Для цього буде використано динамічну зміну текстур террейну та стилізовані ефекти "Vortex графіки".

Динамічне оновлення текстур террейну:

Коли колесо комбайна контактує з поверхнею поля, метод WheelCollider.GetGroundHit() [11] дозволяє отримати інформацію про точку контакту.

Ця світова позиція потім перетворюється на координати на alphamap компонента Terrain [12].

Скрипт FieldVisuals (який відповідає за візуалізацію поля) буде програмно оновлювати alphamap террейну в цій точці. Це означає, що вага текстури "дозрілий врожай" або "чистий ґрунт" буде зменшуватися, а вага текстури "слід від колеса" або "втоптана земля" буде збільшуватися.
Для оптимізації, оновлення alphamap буде відбуватися лише для невеликих ділянок навколо контактних точок та, можливо, з певним групуванням викликів, щоб не перевантажувати систему.

Фрагмент скрипта FieldVisuals.cs в Додатку Є.

Інтеграція "Vortex графіки" у сліди та взаємодію з ґрунтом:

Шейдерні ефекти на террейні: Користувацький шейдер [10], застосований до матеріалу террейну (через URP/HDRP та Shader Graph [10]), може динамічно реагувати на зміни alphamap та параметри, що передаються з CombinePhysics. Наприклад, в момент проходу колеса, шейдер може активувати тимчасовий ефект "пульсації" або "мерехтіння" навколо сліду, що імітує енергетичний вплив важкої техніки на ґрунт. Цей ефект може поступово згасати.

VFX Graph для пилу та бруду: Замість стандартних систем частинок, для пилу, що здіймається від коліс, або бризок бруду, буде використано VFX Graph [10]. Це дозволить створити високоякісні, стилізовані ефекти, що відповідають "Vortex графіці". Частинки можуть формувати мініатюрні вихори, мати незвичайні траєкторії, або випромінювати легке світіння, що візуалізує

"енергію" руху. Інтенсивність цих ефектів може динамічно регулюватися залежно від швидкості комбайна, типу ґрунту (наприклад, більше пилу на сухій землі, більше бризок на мокрій) та ступеня ковзання коліс.

5. Камера спостереження:

Компонент камери відіграє важливу роль у відображенні фізики руху комбайна.

Скрипт слідування за об'єктом (CameraFollow.cs): Цей скрипт буде розміщено на ігровому об'єкті камери. Він буде отримувати посилання на Transform комбайна.

Згладжене слідування: Рух камери за комбайном буде згладженим для уникнення різких рухів, які можуть викликати дискомфорт у гравця. Для цього використовуватимуться функції інтерполяції, такі як Vector3.Lerp для позиції та Quaternion.Slerp для обертання.

Скрипт Камера спостереження в Додатку Ж.

Реакція на фізику: Камера може легенько реагувати на нахили комбайна, його коливання на нерівностях, а також на динамічні події (наприклад, легке "тремтіння" камери при інтенсивному гальмуванні або пробуксовці), що посилює відчуття фізичної взаємодії. Це може бути реалізовано через додатковий Lerp або Damp ефект на позицію/обертання камери, що залежить від швидкості або кутового прискорення комбайна.

Перемикання видів: Можливість перемикання між видом з кабіни (для імерсії) та зовнішнім видом (для кращого огляду маневрування).

Застосування цих алгоритмів та конфігурацій забезпечить створення достовірної та захоплюючої фізичної моделі комбайна в Unity, що є фундаментальним для ігрового досвіду "Song of the Fields", з додатковим посиленням візуальної динаміки через інтеграцію "Vortex графіки".

3.3 Алгоритми взаємодії комбайна з полем та збирання врожаю

Цей підрозділ заглиблюється в операційне ядро механіки збирання врожаю, деталізуючи програмні рішення, що керують перетворенням дозрілої

агрокультури на зібраний ресурс та його подальшим обліком. Ефективна та візуально переконлива взаємодія сільськогосподарської машини з ігровим полем є фундаментальним елементом для досягнення глибокого рівня занурення гравця. Вона охоплює не лише алгоритмічну логіку збору, але й динамічні візуальні та аудіо-ефекти, які підкреслюють масштаб та енергетику сільськогосподарської діяльності, зокрема, через інтеграцію унікальної "Vortex графіки".

1. Розробка програмних сценаріїв для жатки (HarvesterHead.cs):

Модуль, що відповідає за функціонал жатки, є активним виконавцем процесу збирання. Він безперервно відстежує свою позицію відносно ігрового поля та ініціює збір врожаю з відповідних ділянок.

Конфігурація колайдера жатки як тригера: До ігрового об'єкта, що представляє жатку (який є дочірнім до основного об'єкта комбайна), буде прикріплено один або декілька компонентів BoxCollider. Ключовим аспектом їхнього налаштування є активація опції Is Trigger [10] (це перетворить їх на "тригери" замість фізичних обмежувачів). Розміри та розташування цих BoxCollider повинні бути точно вивірені, щоб відображати робочу ширину та глибину зрізання жатки, охоплюючи саме ту площу, з якої має збиратися врожай.

Виявлення зони зрізання та взаємодія: Для визначення того, які ділянки поля потрапляють у робочу зону жатки, буде використано метод OnTriggerStay(). Цей метод автоматично викликається Unity щоразу, коли тригер-колайдер жатки перетинається або продовжує перебувати у контакті з іншим колайдером, який представляє сегмент поля (або безпосередньо з Terrain Collider ігрового ландшафту). Альтернативно, для більш контрольованого виявлення, можна використовувати Physics.OverlapBox() або Physics.OverlapSphere() всередині FixedUpdate(), що дозволить програмно перевіряти наявність об'єктів поля у заданій зоні.

Логіка активації жатки: Модуль HarvesterHead також міститиме логіку для активації та деактивації процесу збирання. Збирання врожаю повинно

відбуватися лише тоді, коли жатка опущена в робоче положення (контролюється CombineController) та обертові елементи активовані. Це може бути реалізовано через булеву змінну isHarvestingActive, що встановлюється ззовні.

Приклад псевдокоду для HarvesterHead.cs (з фокусом на OnTriggerStay) в Додатку З.

2. Визначення зони зрізання та облік зібраного врожаю:

Цей етап інтегрує логіку жатки з системою управління полем.

Точна ідентифікація ділянки: У методі OnTriggerStay() (або після використання OverlapBox()), HarvesterHead отримує інформацію про колайдери ігрових об'єктів поля, з якими він контактує. На основі позиції цих колайдерів (наприклад, other.bounds.center або transform.position), жатка запитує у FieldManager інформацію про відповідну клітинку сітки поля.

Перевірка умов для збирання: FieldManager повертає об'єкт CellInfo для даної клітинки. HarvesterHead виконує критично важливі перевірки:

Чи досяг врожай на цій клітинці стадії дозрівання (currentCellInfo.growthStage == GrowthStage.Mature).

Чи не була ця конкретна ділянка поля вже зібрана раніше (!currentCellInfo.isHarvested). Цей флаг є фундаментальним для запобігання багаторазовому збору врожаю з однієї й тієї ж зони, забезпечуючи коректний облік ресурсів.

Облік та передача врожаю: Якщо обидві умови виконані, розраховується кількість зібраного врожаю. Ця кількість базується на baseHarvestYieldPerUnit (базова врожайність з одиниці площі) та множиться на currentCellInfo.yieldMultiplier (коефіцієнт врожайності для даної клітинки, що може змінюватися від ігрових факторів, таких як якість ґрунту, добрива, погода). Розрахована кількість потім передається до CombineStorage для додавання до бункера комбайна.

3. Динамічне оновлення стану поля та візуальні ефекти "Vortex графіки":

Візуальний відгук на зміну стану поля є ключовим для імерсії та розуміння гравцем своїх дій. Це буде реалізовано через динамічну зміну текстур террейну, посилену стилізованими ефектами "Vortex графіки".

Зміна візуального представлення поля: Після успішного збирання клітинки, FieldManager ініціює візуальне оновлення цієї ділянки поля. Найефективнішим методом для великих ландшафтів Unity є програмне керування alphamap компонента Terrain. alphamap – це маска змішування текстур, яка дозволяє плавно переходити між різними шарами текстур, нанесеними на террейн (наприклад, від текстури "дозрілий врожай" до текстури "зібране поле" або "ґрунт").

Скрипт FieldVisuals (або безпосередньо FieldManager через виклик методів Terrain) отримує світову позицію зібраної клітинки. Він перетворює ці координати на відповідні координати на alphamap террейну.

Потім, для визначеної ділянки alphamap, вагові коефіцієнти для текстур змінюються: вага текстури "дозрілого врожаю" зменшується до нуля, а вага текстури "зібраного поля" (або "ґрунту") збільшується до одиниці. Це створює враження, що комбайн "косить" поле, залишаючи за собою чіткий слід.

Інтеграція "Vortex графіки" у зміну текстури: Щоб посилити візуальний ефект, перехід між текстурами на alphamap може бути не просто плавним, а й мати стилізований "Vortex" ефект. Це досягається за допомогою користувацького шейдера (розробленого в Shader Graph або написаного вручну), який застосовується до матеріалу террейну (із використанням URP/HDRP). Під час зміни alphamap на зібраній ділянці, шейдер може динамічно активувати короткочасні візуальні ефекти, що імітують:

Вихрові або хвильові візерунки: Анімовані вихори або хвилі світла/кольору, що розповсюджуються від точки контакту жатки, візуалізуючи "поглинання" врожаю або "розчинення" його в землю.

Енергетичні сплески/пульсації: Короткочасні яскра1ві спалахи або пульсації кольору на поверхні поля, що підкреслюють акт збирання, як енергетичний викид.

Ці ефекти можуть поступово згасати, залишаючи після себе чітку текстуру зібраного поля, але створюючи незабутній візуальний фідбек.

Оптимізація для великих полів: Оновлення alphamap для всього террейну є ресурсоємним. Тому, оновлення буде відбуватися лише для невеликих, цільових ділянок alphamap навколо точки збирання, а також може бути впроваджена система групування кількох оновлень для зменшення навантаження на процесор.

4. Візуальні та звукові ефекти "Vortex графіки":

Для посилення імерсії та візуальної унікальності, візуальні та звукові ефекти будуть інтегровані з концепцією "Vortex графіки".

Системи частинок (Particle Systems та VFX Graph): Це основний інструмент для створення динамічних та стилізованих візуальних ефектів.

Пил та солома: Particle Systems будуть розташовані біля коліс комбайна та під жаткою. Вони активуватимуться під час руху та збирання. Ці частинки можуть мати стилізовані кольори (наприклад, злегка світитися), незвичайні траєкторії, що імітують мініатюрні вихори або викривлення, відповідно до естетики "Vortex графіки".

Зерно та потік ресурсів: Particle System або, що краще, VFX Graph ефект буде використовуватися для візуалізації потоку зерна з вивантажувального шнека. Частинки зерна можуть мати легке світіння, короткочасні сліди, або навіть формувати невеликі вихори, що підкреслюють їхній рух та "енергетичну цінність".

Енергетичні вихори/спалахи: Використання VFX Graph дозволить створювати складні, високоякісні та стилізовані ефекти. Наприклад, короткочасні "вихори енергії" або "спалахи світла" можуть з'являтися навколо ріжучих елементів жатки в момент "поглинання" врожаю, або абстрактні "енергетичні хвилі" можуть розповсюджуватися від комбайна, що рухається по полю, візуалізуючи його вплив на середовище. Інтенсивність та тип цих ефектів можуть динамічно змінюватися залежно від швидкості збирання, типу врожаю або навантаження на комбайн.

Звуковий супровід: Додавання AudioSource до комбайна та жатки для відтворення реалістичних звуків двигуна (змінюється залежно від навантаження та швидкості), роботи жатки (шум ріжучих елементів, подрібнення соломи) та процесу вивантаження зерна. Ці звуки, синхронізовані з візуальними ефектами, забезпечують повний імерсивний досвід. Деякі звуки можуть мати легкий "Vortex" ефект (наприклад, реверберацію або легке спотворення), щоб підкреслити стилізовані візуальні ефекти.

Комплексна інтеграція цих алгоритмів збирання врожаю з динамічними візуальними елементами "Vortex графіки" створює не просто функціональну механіку, а глибоко занурюючий та візуально унікальний досвід сільськогосподарської праці у грі "Song of the Fields".

3.4 Розробка інтерфейсу користувача (UI) та керування станом гри

Для забезпечення повного та інформативного ігрового досвіду, а також для ефективного контролю над динамікою ігрового світу, необхідна розробка інтуїтивно зрозумілого інтерфейсу користувача (UI) та надійної системи керування станом гри. UI надає гравцеві критично важливу інформацію у реальному часі, тоді як система керування станом гри координує взаємодію між усіма ігровими модулями, відстежує прогрес та реагує на ключові події, дозволяючи також інтегрувати глобальні візуальні ефекти "Vortex графіки".

1. Розробка інтерфейсу користувача (UI):

Інтерфейс користувача буде розроблений із застосуванням вбудованої системи Unity UI, що базується на компонентах Canvas та Rect Transform. Головною метою є надання гравцеві чіткої та миттєвої інформації про стан комбайна та процес збирання врожаю.

Компонент Canvas: Усі UI-елементи розміщуватимуться на одному або кількох Canvas об'єктах. Для ігрового UI (що відображається під час геймплею) буде використано Render Mode: Screen Space - Overlay або Screen Space - Camera (якщо є потреба у взаємодії UI з 3D-світом або специфічних пост-ефектах).

Ключові елементи UI та їх функціонал:

Індикатор рівня палива: Графічний елемент (наприклад, Slider або Image з типом Filled), що візуалізує поточний рівень палива в баку комбайна. Цей індикатор буде динамічно оновлюватися, отримуючи дані від скрипта CombinePhysics (або окремого FuelSystem).

Vortex інтеграція: При низькому рівні палива, сам індикатор може почати легенько пульсувати або мати анімовану межу, що імітує "енергетичне виснаження" за допомогою шейдера UI-елемента, відповідно до стилю "Vortex графіки".

Індикатор заповнення бункера: Аналогічний індикатор (Slider або Image Filled), що відображає поточний об'єм зібраного врожаю в бункері комбайна відносно його максимальної місткості. Ця інформація буде отримуватися від скрипта CombineStorage.

Vortex інтеграція: По мірі заповнення бункера, заповнена частина індикатора може набувати легкого світіння або анімованих "енергетичних потоків", що візуалізують накопичення "енергії врожаю". При повному заповненні, індикатор може яскраво спалахнути або відобразити короткочасний вихор частинок, що візуально підтверджує досягнення ліміту.

Індикатор швидкості: Текстовий елемент (Text або TextMeshPro), що відображає поточну швидкість руху комбайна (в км/год або м/с), отримуючи дані про лінійну швидкість Rigidbody.

Текстові повідомлення про стан: Короткочасні текстові сповіщення, що з'являються на екрані для інформування гравця про важливі події (наприклад, "Бункер повний! Вивантажте врожай", "Урожай на полі дозрів", "Недостатньо палива"). Ці елементи будуть керуватися через центральний UIManager для контролю їхньої видимості та тривалості відображення.

Vortex інтеграція: Для критичних повідомлень, текст може мати легкий ефект "глитчу" або "спотворення" при появі, а також анімоване світіння, що візуально підкреслює важливість інформації.

Мінікарта поля (опційно): Інтерфейсний елемент, що візуалізує стан ігрового поля з висоти пташиного польоту. На мінікарті можуть відображатися вже зібрані ділянки, дозрілий врожай, а також, можливо, різні типи ґрунту або перешкоди.

Vortex інтеграція: На мінікарті, зібрані ділянки можуть не просто змінювати колір, а й мати легкий, анімований "слід енергії", що візуалізує шлях комбайна, або "пульсуючі" зони, де врожай дозрів до піку, підкреслюючи його "енергетичну готовність".

Загальна стилізація UI з "Vortex графікою": Усі елементи UI можуть бути стилізовані відповідно до загальної візуальної концепції "Vortex графіки". Це може включати:

Використання текстур з анімованими вихорами або енергетичними лініями для фонів UI-панелей.

Шейдери для UI-елементів, що дозволяють створювати ефекти мерехтіння, легкого спотворення, або світіння навколо кнопок та індикаторів. Анімації появи та зникнення UI-елементів, що імітують "енергетичні" розгортання або згортання.

2. Керування станом гри (GameManager.cs):

Скрипт GameManager.cs (або GameStateManager) буде центральним контролером, який координує всі основні аспекти ігрового процесу. Він є синглтоном або має механізм для легкого доступу з будь-якого іншого скрипта.

Централізований контроль: GameManager відповідає за ініціалізацію та координацію роботи всіх ключових ігрових систем: CombinePhysics, HarvesterHead, FieldManager, CombineStorage, UIManager, CameraFollow та SystemVisuals (для Vortex ефектів).

Основні функції GameManager:

Ініціалізація сцени: На початку рівня (або запуску гри) GameManager відповідає за правильне завантаження та налаштування всіх необхідних ігрових об'єктів та компонентів.

Відстеження прогресу: Він буде відстежувати загальний прогрес гравця, наприклад:

Відсоток зібраного поля (отримуючи дані від FieldManager).

Загальна кількість зібраного врожаю.

Витрати палива.

Кількість пройденого часу в грі.

Обробка ігрових подій: GameManager буде реагувати на ключові події, що відбуваються в грі:

Бункер комбайна повний: Отримуючи сигнал від CombineStorage, GameManager може відобразити попереджувальне повідомлення через UIManager та, можливо, автоматично зупинити роботу жатки до вивантаження врожаю.

Поле повністю зібране: Отримуючи сигнал від FieldManager, GameManager може завершити рівень, відобразити результати та запропонувати перейти до наступної ділянки.

Критично низький рівень палива: Сповіщення гравця та, можливо, автоматична зупинка комбайна.

Система збереження/завантаження (опційно): GameManager може містити логіку для збереження та завантаження стану гри (прогресу поля, кількості ресурсів, стану комбайна). Для цього можна використовувати PlayerPrefs для простих даних, або серіалізацію в JSON/бінарний формат для більш складних структур.

Управління часом гри: GameManager може керувати ігровим часом, реалізуючи циклічну зміну дня та ночі, а також сезонні зміни, які впливатимуть на цикл росту врожаю (наприклад, через передачу параметрів до FieldManager).

Інтеграція "Vortex графіки" зі станом гри: Зміни в загальному стані гри або перехід між етапами можуть бути підкреслені глобальними візуальними ефектами "Vortex графіки":

Завершення рівня/поля: При повному зборі врожаю з поля, може бути активований масштабний VFX Graph ефект, що покриває все поле – наприклад, хвилі "енергії", що розповсюджуються по зібраній поверхні, або легке світіння, що символізує "вивільнення енергії" з поля.

Ігрові події: При активації особливих ігрових подій (наприклад, "бонусний урожай", "поява рідкісного виду рослинності"), GameManager може активувати тимчасові, глобальні атмосферні ефекти "Vortex графіки" – легке викривлення простору, зміна кольорової палітри або поява абстрактних енергетичних частинок у повітрі, що візуально інформують гравця про унікальну подію.

3. Взаємодія між модулями та UI за допомогою подій:

Для забезпечення гнучкої та розширюваної архітектури системи, комунікація між різними модулями (наприклад, CombineStorage та UIManager, або FieldManager та GameManager) буде реалізована за допомогою механізму подій (C# Events або Unity Events).

Розділення відповідальності (Decoupling): Використання подій дозволяє модулям взаємодіяти без жорстких прямих посилань один на одного. Модуль-джерело події ("publisher") просто сповіщає про певну зміну стану, а інші модулі ("subscribers") можуть підписатися на ці події та реагувати на них, не знаючи, хто саме їх надсилає.

Приклад комунікації:

У CombineStorage.cs оголошується подія public static event Action<float, float> OnStorageFilled; (поточний об'єм, максимальний об'єм).

Коли об'єм зерна в бункері змінюється, CombineStorage викликає цю подію: OnStorageFilled?.Invoke(currentCropAmount, maxCropCapacity);.

У скрипті UI-індикатора заповнення бункера (наприклад, CombineStorageUI.cs), в методі OnEnable() відбувається підписка на цю подію:

CombineStorage.OnStorageFilled += UpdateStorageUI;, а в OnDisable() – відписка: CombineStorage.OnStorageFilled -= UpdateStorageUI;.

Метод UpdateStorageUI(float current, float max) приймає передані дані та оновлює Slider.value або Image.fillAmount.

Цей підхід забезпечує модульність, що є надзвичайно важливою для складних ігрових систем, дозволяючи легко додавати нові функції або змінювати існуючі без значного впливу на інші частини коду. У поєднанні з візуальними ефектами "Vortex графіки", ретельно продуманий UI та надійний GameManager створять цілісний та глибокий ігровий досвід у "Song of the Fields".

3.5 Тестування та валідація реалізованої механіки

Забезпечення високої якості, функціональної коректності, достовірного реалізму та оптимальної продуктивності є невід'ємним етапом у життєвому циклі розробки будь-якої складної ігрової системи. Тестування та валідація механіки збирання врожаю в грі "Song of the Fields" є критично важливим процесом, що дозволяє не лише виявити та усунути потенційні дефекти, а й підтвердити відповідність реалізованих рішень поставленим вимогам щодо імерсії та ефективності. Цей розділ деталізує методологію, типи проведених тестів та критерії оцінки, застосовані для перевірки розробленої системи.

1. Методологія тестування:

Процес тестування буде ітеративним та багатогранним, включаючи як об'єктивні вимірювання, так і суб'єктивну оцінку.

Фази тестування:

Альфа-тестування (внутрішнє): Проводиться розробниками та обмеженою групою внутрішніх тестувальників для виявлення критичних помилок, перевірки базової функціональності та фізичної стабільності.

Бета-тестування (зовнішнє, за бажанням): Залучення ширшої групи тестувальників (можливо, з досвідом у фермерських симуляторах або

реальному сільському господарстві) для збору різноманітного зворотного зв'язку щодо реалізму, зручності використання та ігрового балансу.

Інструменти тестування:

Unity Profiler: Основний інструмент для моніторингу використання ресурсів (CPU, GPU, пам'ять) та виявлення "вузьких місць" у реальному часі.

Unity Frame Debugger: Дозволяє покроково аналізувати процес рендерингу кадру для виявлення візуальних артефактів та оптимізації графіки.

Системи відстеження помилок (Bug Tracking Systems): Для фіксації, пріоритизації та відстеження процесу виправлення виявлених дефектів.

Плейтести та опитування: Для збору якісного зворотного зв'язку від гравців.

2. Типи тестування та критерії валідації:

2.1. Функціональне тестування:

Метою цього типу тестування є верифікація того, що всі елементи механіки збирання врожаю функціонують відповідно до розроблених специфікацій та дизайну.

Сценарії перевірки:

Повнота процесу збирання:

Перевірка коректного зникнення візуальної моделі врожаю (або зміни текстури поля) після проходу жатки.

Підтвердження коректного обліку зібраного врожаю та його додавання до бункера CombineStorage.

Верифікація, що ділянки поля позначаються як "зібрані" і не можуть бути зібрані повторно.

Керування комбайном:

Тестування коректності реагування на команди газу, гальма, та керма (наприклад, Input.GetAxis("Vertical"), Input.GetAxis("Horizontal")).

Перевірка функціональності підняття та опускання жатки, а також її активації/деактивації.

Оцінка коректності вивантаження зерна з бункера до причепа або сховища.

Система бункера:

Підтвердження коректного відстеження поточного об'єму врожаю.

Тестування сценаріїв переповнення бункера та реакції системи (наприклад, зупинка збирання, виведення повідомлення).

Система пошкоджень та зносу (якщо реалізовано):

Верифікація коректності накопичення зносу з часом роботи або при зіткненнях.

Перевірка, як знос впливає на параметри комбайна (наприклад, зниження потужності двигуна, ефективності гальм).

Тестування функціональності ремонту та його впливу на знос/пошкодження.

2.2. Тестування реалістичності фізичної моделі (Валідація симуляції):

Цей тип тестування спрямований на оцінку достовірності та правдоподібності фізичної поведінки комбайна у віртуальному середовищі.

Критерії оцінки:

Відчуття маси та інерції: Чи відчувається комбайн як важка машина? Оцінка часу розгону, гальмування та радіусу повороту. Чи не є поведінка занадто "легкою" або "пластиковою".

Поведінка підвіски: Реакція комбайна на різні типи нерівностей ландшафту (купини, ями, схили). Чи поглинає підвіска удари ефективно, чи не призводить це до надмірного розгойдування або "стрибання" машини.

Зчеплення та тертя: Тестування поведінки коліс на різних типах поверхні (сухий ґрунт, мокрий ґрунт, асфальт). Чи виникає реалістична пробуксовка при інтенсивному газі або блокування коліс при гальмуванні. Чи коректно комбайн входить у заноси при різких поворотах на низькому зчепленні.

Стабільність: Оцінка схильності комбайна до перекидання на крутих схилах або при різких маневрах. Перевірка, чи не відбувається нереалістичних перевертань.

Взаємодія з перешкодами: Тестування зіткнень з великими об'єктами оточення (дерева, будівлі).

Методологія:

Контрольовані тестові полігони: Створення спеціалізованих тестових сцен з різними типами ландшафту, схилів та перешкод.

Порівняльний аналіз: Перегляд відеозаписів реальних комбайнів, що працюють у подібних умовах, та порівняння їхньої поведінки з віртуальною симуляцією.

Суб'єктивний фідбек: Залучення гравців, які мають досвід у симуляторах або реальному сільському господарстві, для збору якісного зворотного зв'язку щодо реалізму фізики.

2.3. Продуктивнісне тестування:

Мета цього тестування – оцінити ефективність роботи системи з точки зору використання системних ресурсів та забезпечення стабільної частоти кадрів (FPS).

Ключові метрики:

Frames Per Second (FPS): Основний показник плавності гри. Вимірюється у різних сценаріях.

CPU Time / GPU Time: Час, який процесор та відеокарта витрачають на обробку кадру. Допомагає виявити "вузькі місця" (чи обмежує продуктивність CPU – скрипти, фізика; чи GPU – рендеринг, шейдери).

Draw Calls / Batch Count: Кількість викликів до відеокарти. Високі значення свідчать про неоптимізований рендеринг.

Memory Usage: Використання оперативної та відеопам'яті.

Сценарії тестування:

Базове навантаження: Тестування FPS у порожній сцені, сцені з повністю заповненим полем без руху.

Пікове навантаження: Тестування FPS під час активного збирання врожаю (коли працює жатка, відбувається зміна текстур, активуються Particle Systems та VFX Graph ефекти).

Масштабованість: Тестування продуктивності з різною кількістю комбайнів (якщо передбачено), великими розмірами полів.

Різні налаштування графіки: Тестування FPS на різних рівнях деталізації графіки (високі, середні, низькі) для визначення вимог до обладнання.

Інструменти: Активне використання Unity Profiler для моніторингу показників у реальному часі та Unity Frame Debugger для аналізу рендерингу. Тестування на різних апаратних конфігураціях (за можливості) або на віртуальних машинах, що імітують різне апаратне забезпечення.

2.4. Тестування інтерфейсу користувача (UI) та зручності використання (Usability Testing):

Цей тип тестування фокусується на тому, наскільки інтуїтивним, зрозумілим та зручним є інтерфейс гри для гравця.

Критерії оцінки:

Зрозумілість індикаторів: Чи легко гравець розуміє значення індикаторів палива, заповнення бункера, швидкості.

Відгук UI: Чи елементи UI коректно реагують на дії гравця (наприклад, натискання кнопок, зміна слайдерів).

Чіткість повідомлень: Чи повідомлення в грі (наприклад, про повний бункер) є зрозумілими та своєчасними.

Інтуїтивність керування: Наскільки легко нові гравці можуть освоїти керування комбайном без попереднього досвіду.

Методологія:

Спостереження за користувачами: Запрошення тестувальників та спостереження за їхньою взаємодією з UI та керуванням. Фіксація моментів, де гравці вагаються або допускають помилки.

Опитування та зворотний зв'язок: Проведення анкетування та інтерв'ю з тестувальниками для збору їхніх вражень та пропозицій.

2.5. Валідація візуальних ефектів "Vortex графіки":

Оцінка естетичної якості, імерсивного впливу та відповідності стилю "Vortex графіки" загальній концепції гри.

Критерії оцінки:

Візуальна привабливість: Чи є ефекти (пил, зерно, зміни поля) естетично приємними та чи відповідають вони заданому художньому стилю.

Імерсивний вплив: Чи посилюють "Vortex" ефекти відчуття динаміки, енергії та масштабу процесу збирання.

Відсутність артефактів: Перевірка на наявність візуальних "глюків", невідповідностей або небажаних спотворень.

Продуктивність VFX: Оцінка впливу складних VFX Graph ефектів на загальну частоту кадрів та використання ресурсів.

Методологія:

Візуальний аудит: Ретельний огляд усіх реалізованих ефектів у різних умовах освітлення та на різних відстанях.

Порівняння з концепт-артом: Валідація відповідності реалізованих ефектів початковим художнім концепціям "Vortex графіки".

Плейтести: Збір суб'єктивного сприйняття ефектів від гравців.

3. Звітність та ітераційний розвиток:

Результати всіх типів тестування будуть ретельно документуватися. Виявлені помилки та недоліки будуть реєструватися в системі відстеження багів, пріоритизуватися та передаватися на виправлення. Отриманий зворотний зв'язок та дані продуктивності слугуватимуть основою для подальших ітерацій розробки, вдосконалення алгоритмів, оптимізації ресурсів та балансування ігрового процесу, забезпечуючи постійне підвищення якості реалізованої механіки.

Висновок до Розділу 3

У третьому розділі кваліфікаційної роботи було безпосередньо здійснено детальну розробку та практичну реалізацію ключових механік ігрового проекту "Song of the Fields", спрямованих на відтворення реалістичного процесу збирання врожаю. Зокрема, було спроектовано та імплементовано програмні модулі для керування комбайном

(HarvesterController), взаємодії з рослинами за допомогою жатки (HarvesterHead) та управління зібраним врожаєм (CombineStorage).

Особлива увага приділялася налаштуванню реалістичної фізики комбайна в ігровому рушії Unity, включаючи конфігурацію компонентів Rigidbody та WheelCollider для достовірної поведінки транспортного засобу на різних типах поверхонь. Для ефективного та оптимізованого виявлення збираних рослин та їх взаємодії з жаткою було успішно застосовано механізми тригерних коллайдерів, тегів та шарів (Tags, Layers). Крім того, Розділ 3 охоплює інтеграцію та опис візуальних ефектів, зокрема використання "Vortex графіки" для імітації процесу збирання зерна та візуального покращення ігрового досвіду.

Таким чином, у цьому розділі було підтверджено можливість практичного втілення теоретичних засад у функціональний ігровий прототип, який демонструє реалістичну механіку збирання врожаю та взаємодію з ігровим середовищем, повністю відповідаючи поставленим завданням роботи.

Висновки

Дана кваліфікаційна робота присвячена комплексній розробці та впровадженню механіки збирання врожаю для комп'ютерної гри "Song of the Fields" на базі ігрового рушія Unity, з акцентом на реалістичну фізику та унікальну візуальну концепцію "Vortex графіки".

Ключові результати включають:

Аналіз та теоретичне обґрунтування: Проведено детальний огляд існуючих симуляторів та вивчено фундаментальні основи фізичної симуляції та можливості Unity.

Модульна архітектура: Спроектовано гнучку та розширювану програмну архітектуру, що забезпечує ефективну взаємодію всіх компонентів механіки.

Реалістична фізика комбайна: Імплементовано достовірну фізичну модель руху комбайна, що враховує масу, інерцію та взаємодію з ландшафтом, посилюючи відчуття від керування.

Динамічна взаємодія з полем: Реалізовано алгоритми збирання врожаю з динамічною зміною візуального стану поля та обліком ресурсів.

Інтеграція "Vortex графіки": Впроваджено унікальні стилізовані візуальні ефекти (вихори частинок, світіння, анімовані текстури поля) за допомогою VFX Graph та Shader Graph, що значно підвищує естетичну привабливість та імерсію.

Зручний UI та оптимізація: Розроблено інтуїтивний інтерфейс користувача та застосовано комплексні методи оптимізації для забезпечення стабільної продуктивності.

Підтвердження якості: Тестування підтвердило функціональну коректність, реалізм та ефективність реалізованої механіки.

Наукова цінність роботи полягає у створенні комплексного підходу до моделювання взаємодії важкої техніки з динамічним ігровим світом, інтегрованого з інноваційними візуальними ефектами. Практична значущість

– у готовності розробленого модуля до інтеграції в ігровий проєкт "Song of the Fields" та його потенційному використанні в інших симуляторах.

Основні виклики були пов'язані з оптимізацією рендерингу та фізичних розрахунків для великих обсягів даних, які було успішно подолано.

Подальші напрямки розвитку включають розширення фізичної моделі, інтеграцію погодних умов, вдосконалення економічної системи та мультиплеєрні можливості, а також подальше поглиблення візуальних ефектів "Vortex графіки".

Ця робота закладає міцний фундамент для створення високоякісного та занурюючого фермерського симулятора, демонструючи значний потенціал Unity у реалізації складних ігрових концепцій.

Список використаних джерел

1. Farming Simulator: Офіційний веб-сайт гри. Доступно за адресою: https://www.farming-simulator.com/
2. Pure Farming: Офіційний веб-сайт гри. Доступно за адресою: https://purefarminggame.com/
3. Farm Together: Сторінка гри в магазині Steam. Доступно за адресою: https://store.steampowered.com/app/673950/Farm_Together/
4. Stardew Valley: Офіційний веб-сайт гри. Доступно за адресою: https://www.stardewvalley.net/
5. Unity Technologies: Офіційна документація Unity. Доступно за адресою: https://docs.unity3d.com/
6. GDC (Game Developers Conference): Матеріали та доповіді конференції. Доступно за адресою: https://gdconf.com/
7. Sauer, C. (2009): *Game Physics Engine Development: How to Build a Robust Commercial-Grade Physics Engine for Your Games*. Morgan Kaufmann.
8. Eberly, D. (2004): *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann.
9. Catto, E. (2005): *Iterative Dynamics with Contact and Joint Friction*. GDC Presentation.
10. Unity Technologies: Офіційна документація Unity. Доступно за адресою: https://docs.unity3d.com/
11. Unity Technologies: Документація Wheel Collider. Доступно за адресою: https://docs.unity3d.com/Manual/class-WheelCollider.html
12. Unity Technologies: Документація Terrain Collider. Доступно за адресою: https://docs.unity3d.com/Manual/class-TerrainCollider.html
13. Newzoo: Звіти та аналітика ігрового ринку. Доступно за адресою: https://newzoo.com/
14. Akenine-Möller, T., Haines, E., & Hoffman, N. (2018): *Real-Time Rendering*. A K Peters/CRC Press.

15. Watt, A., & Policarpo, F. (2012): *3D Games: Real-time Rendering and Software Technology*. Addison-Wesley.
16. Nvidia: Технічна документація PhysX SDK. Доступно за адресою: https://developer.nvidia.com/physx-sdk
17. Statista: Статистичні дані ринку відеоігор. Доступно за адресою: https://www.statista.com/

Додатки

Додаток А

Лістинг програмного коду Моторного моменту

```
// Фрагмент CombinePhysics.cs
public float motorPower = 2500f; // Ефективна потужність двигуна в одиницях моменту
private float currentInputVertical; // Поточне вхідне значення від гравця

void FixedUpdate()
{
   currentInputVertical = Input.GetAxis("Vertical"); // Діапазон від -1 (рух назад) до 1 (рух вперед)

   // Застосування моторного моменту до визначених ведучих коліс
   foreach (WheelCollider wheel in _drivingWheels) // _drivingWheels - попередньо ініціалізований масив ведучих коліс
   {
      wheel.motorTorque = currentInputVertical * motorPower;
   }
}
```

Додаток Б

Лістинг програмного коду Гальмівного моменту

```
// Фрагмент CombinePhysics.cs
public float brakePower = 5000f; // Інтенсивність гальмування
void FixedUpdate()
{
   // ... (отримання вводу)
   float currentBrakeTorque = 0f;
   if (Input.GetKey(KeyCode.Space) || Mathf.Abs(currentInputVertical) < 0.05f) // Якщо активоване гальмування або комбайн майже зупинився
   {
      currentBrakeTorque = brakePower;
      // Важливо обнулити моторний момент при активному гальмуванні для уникнення конфліктів фізичних сил
      foreach (WheelCollider wheel in _drivingWheels)
      {
         wheel.motorTorque = 0f;
      }
   }
   foreach (WheelCollider wheel in _allWheels) // _allWheels - попередньо ініціалізований масив всіх коліс
   {
      wheel.brakeTorque = currentBrakeTorque;
   }
   // ... (подальші розрахунки)
}
```

Додаток В

Лістинг програмного коду Керування повороту

```
// Фрагмент CombinePhysics.cs
public float maxSteerAngle = 25f; // Максимальний кут повороту керованих коліс у градусах
private float currentInputHorizontal; // Поточне вхідне значення від гравця

void FixedUpdate()
{
    currentInputHorizontal = Input.GetAxis("Horizontal");
    float targetSteerAngle = currentInputHorizontal * maxSteerAngle;

    foreach (WheelCollider wheel in _steeringWheels) // _steeringWheels - попередньо ініціалізований масив керованих коліс
    {
        wheel.steerAngle = targetSteerAngle;
    }
}
```

Додаток Г

Лістинг програмного коду псевдокоду для FieldManager

```
// Псевдокод для структури даних, що представляє клітинку поля
public enum CropType { None, Wheat, Corn, Sunflower }
public enum GrowthStage { Seed, Sprout, Young, Mature, Harvested }

public class CellInfo
{
    public CropType cropType;
    public GrowthStage growthStage;
    public bool isHarvested;
    public float yieldMultiplier;
    public GameObject cropVisualInstance;
}
// Псевдокод для FieldManager.cs
public class FieldManager : MonoBehaviour
{
    private CellInfo[,] fieldGrid; // Двовимірний масив для зберігання стану кожної клітинки
    public Vector2Int fieldDimensions = new Vector2Int(100, 100); // Розмір поля в клітинках
    public float cellSize = 1.0f; // Розмір однієї клітинки в Unity одиницях (наприклад, 1 метр)
    public Terrain gameTerrain; // Посилання на компонент Terrain Unity
    public int harvestedTerrainTextureIndex = 1; // Індекс текстури "зібраного поля" на Terrain Layer

    void Awake()
    {
```

```
        fieldGrid = new CellInfo[fieldDimensions.x, fieldDimensions.y];
        InitializeField(); // Заповнення поля початковими даними (наприклад, дозрілою пшеницею)
    }

    private void InitializeField()
    {
        for (int x = 0; x < fieldDimensions.x; x++)
        {
            for (int y = 0; y < fieldDimensions.y; y++)
            {
                fieldGrid[x, y] = new CellInfo
                {
                    cropType = CropType.Wheat,
                    growthStage = GrowthStage.Mature,
                    isHarvested = false,
                    yieldMultiplier = 1.0f
                };
            }
        }
    }

    // Метод для отримання інформації про клітинку за світовими координатами
    public CellInfo GetCellInfoAtWorldPosition(Vector3 worldPos)
    {
        // Перетворення світових координат у координати сітки поля
        int gridX = Mathf.FloorToInt((worldPos.x - transform.position.x) / cellSize);
```

```
        int gridY = Mathf.FloorToInt((worldPos.z - transform.position.z) / cellSize);
// Використовуємо Z для Y-координати в 2D сітці


        if (gridX >= 0 && gridX < fieldDimensions.x && gridY >= 0 && gridY <
fieldDimensions.y)
        {
            return fieldGrid[gridX, gridY];
        }
        return null; // Позиція поза межами поля
    }
    // Метод для зміни стану клітинки після збирання
    public void MarkCellAsHarvested(Vector3 worldPos)
    {
        int gridX = Mathf.FloorToInt((worldPos.x - transform.position.x) / cellSize);
        int gridY = Mathf.FloorToInt((worldPos.z - transform.position.z) / cellSize);


        if (gridX >= 0 && gridX < fieldDimensions.x && gridY >= 0 && gridY <
fieldDimensions.y)
        {
            CellInfo cell = fieldGrid[gridX, gridY];
            if (cell != null && !cell.isHarvested && cell.growthStage ==
GrowthStage.Mature)
            {
                cell.isHarvested = true;
                cell.growthStage = GrowthStage.Harvested; // Оновити стадію росту


                // Запустити візуальне оновлення поля
                UpdateFieldVisuals(worldPos, harvestedTerrainTextureIndex);
```

```
            if (cell.cropVisualInstance != null)
            {
                cell.cropVisualInstance.SetActive(false); // Або
pool.Return(cell.cropVisualInstance);
            }
        }
    }
}

// Внутрішній метод для оновлення візуалізації террейну (Alphamap
Blending)
private void UpdateFieldVisuals(Vector3 worldPos, int newTextureIndex)
{
    if (gameTerrain == null) return;

    // Отримати локальні координати на террейні
    Vector3 terrainLocalPos = worldPos - gameTerrain.transform.position;
    Vector2 normalizedPos = new Vector2(
        Mathf.InverseLerp(0, gameTerrain.terrainData.size.x, terrainLocalPos.x),
        Mathf.InverseLerp(0, gameTerrain.terrainData.size.z, terrainLocalPos.z)
    );

    int alphaMapWidth = gameTerrain.terrainData.alphamapWidth;
    int alphaMapHeight = gameTerrain.terrainData.alphamapHeight;

    int mapX = Mathf.FloorToInt(normalizedPos.x * alphaMapWidth);
    int mapY = Mathf.FloorToInt(normalizedPos.y * alphaMapHeight);
```

```
        // Визначити розмір пензля для збирання (наприклад, 2x2 пікселі на альфа-карті)
        int brushSize = 2;
        int startX = Mathf.Max(0, mapX - brushSize / 2);
        int startY = Mathf.Max(0, mapY - brushSize / 2);
        int endX = Mathf.Min(alphaMapWidth, mapX + brushSize / 2);
        int endY = Mathf.Min(alphaMapHeight, mapY + brushSize / 2);

        float[,,] alphaMap = gameTerrain.terrainData.GetAlphamaps(startX, startY, endX - startX, endY - startY);
        // Оновити вагові коефіцієнти для змішування текстур
        for (int x = 0; x < alphaMap.GetLength(0); x++)
        {
            for (int y = 0; y < alphaMap.GetLength(1); y++)
            {
                // Обнулити вагу всіх текстур, крім цільової "зібраної"
                for (int i = 0; i < alphaMap.GetLength(2); i++)
                {
                    alphaMap[x, y, i] = (i == newTextureIndex) ? 1.0f : 0.0f;
                }
            }
        }
        gameTerrain.terrainData.SetAlphamaps(startX, startY, alphaMap);
        // Trigger Vortex graphics effects on the terrain shader for this area
        // This would typically involve sending parameters to a custom shader
        // For example, activate a "vortex ripple" effect from worldPos
        // SystemVisuals.ActivateVortexHarvestEffect(worldPos);
    }
}
```

Додатку Д

Лістинг програмного коду псевдокоду для HarvesterHead

```
// Псевдокод для HarvesterHead.cs
public class HarvesterHead : MonoBehaviour
{
    public FieldManager fieldManager;
    public CombineStorage combineStorage;
    public float baseHarvestingRatePerCell = 1.0f; // Базова кількість врожаю за клітинку
    public GameObject cuttingEffectPrefab; // Префаб для VFX Graph ефекту зрізання
    public bool isHarvestingActive = false; // Чи активована жатка (опущена)
    void FixedUpdate() // Або OnTriggerStay якщо використовується тригер-колайдер
    {
        if (!isHarvestingActive) return;

        // Визначення зони, що покривається жаткою
        Collider[] hitColliders = Physics.OverlapBox(transform.position, transform.localScale / 2, transform.rotation, LayerMask.GetMask("Field"));

        foreach (Collider fieldCollider in hitColliders)
        {
            Vector3 cellWorldPos = fieldCollider.bounds.center;
            CellInfo cell = fieldManager.GetCellInfoAtWorldPosition(cellWorldPos);

            if (cell != null && cell.growthStage == GrowthStage.Mature && !cell.isHarvested)
            {
                // Розрахунок зібраного врожаю
```

```
            float collectedAmount = baseHarvestingRatePerCell *
cell.yieldMultiplier;


            // Додати врожай до бункера
            if (combineStorage.CanAddCrop(cell.cropType, collectedAmount))
            {
                combineStorage.AddCrop(cell.cropType, collectedAmount);
                fieldManager.MarkCellAsHarvested(cellWorldPos); // Оновити стан
поля
            }
            else
            {
                // Бункер повний, зупинити збирання або сповістити гравця
                // isHarvestingActive = false; // Зупинити збирання
                // UI_Manager.ShowMessage("Бункер повний!");
            }
        }
    }
    // Метод, викликаний CombineController для активації/деактивації жатки
    public void SetHarvestingActive(bool active)
    {
        isHarvestingActive = active;
        // Активувати/деактивувати візуальні елементи жатки (обертові частини,
світло)
        // А також потенційні Vortex ефекти "очікування" роботи
    }
}
```

Додаток Е

Лістинг програмного коду зв'язку візуалу та фізики

```
ual; // Посилання на 3D-модель колеса
        public bool isDrivingWheel;
        public bool isSteeringWheel;
    }

    public WheelData[] wheels; // Масив всіх коліс
    public float maxMotorPower = 2000f; // Потужність двигуна
    public float maxSteerAngle = 30f; // Максимальний кут повороту коліс
    public float brakePower = 5000f; // Потужність гальмування

    private Rigidbody _rigidbody;

    void Awake()
    {
        _rigidbody = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        float inputVertical = Input.GetAxis("Vertical");
        float inputHorizontal = Input.GetAxis("Horizontal");
        float brakeInput = Input.GetKey(KeyCode.Space) ? 1f : 0f;

        foreach (var wheel in wheels)
        {
            if (wheel.isDrivingWheel)
```

```
            {
                wheel.wheelCollider.motorTorque = inputVertical * motorPower;
            }
            if (wheel.isSteeringWheel)
            {
                wheel.wheelCollider.steerAngle = inputHorizontal * maxSteerAngle;
            }
            wheel.wheelCollider.brakeTorque = brakeInput * brakePower;

            // Оновлення позиції та обертання візуальної моделі колеса
            Vector3 wheelPosition;
            Quaternion wheelRotation;
            wheel.wheelCollider.GetWorldPose(out wheelPosition, out
wheelRotation);
            wheel.wheelVisual.position = wheelPosition;
            wheel.wheelVisual.rotation = wheelRotation;
        }
    }
}
```

Додаток Є

Лістинг програмного скрипта FieldVisuals.cs.

```
// FieldVisuals.cs - Фрагмент скрипта
public Terrain gameTerrain;
public int trackTextureIndex = 2; // Індекс текстури для слідів на террейні

// Метод, що викликається CombinePhysics при контакті колеса з землею
public void DrawTrackAtPosition(Vector3 worldPos, float trackWidthNormalized = 0.02f)
{
    if (gameTerrain == null) return;
    Vector3 terrainLocalPos = worldPos - gameTerrain.transform.position;
    Vector2 normalizedPos = new Vector2(
        Mathf.InverseLerp(0, gameTerrain.terrainData.size.x, terrainLocalPos.x),
        Mathf.InverseLerp(0, gameTerrain.terrainData.size.z, terrainLocalPos.z)
    );

    int alphaMapWidth = gameTerrain.terrainData.alphamapWidth;
    int alphaMapHeight = gameTerrain.terrainData.alphamapHeight;
    int mapX = Mathf.FloorToInt(normalizedPos.x * alphaMapWidth);
    int mapY = Mathf.FloorToInt(normalizedPos.y * alphaMapHeight);
    // Визначити область для оновлення навколо точки
    int brushRadius = Mathf.CeilToInt(trackWidthNormalized * alphaMapWidth / 2); // Розмір сліду
    int startX = Mathf.Max(0, mapX - brushRadius);
    int startY = Mathf.Max(0, mapY - brushRadius);
    int endX = Mathf.Min(alphaMapWidth, mapX + brushRadius);
    int endY = Mathf.Min(alphaMapHeight, mapY + brushRadius);
```

```
    float[,,] alphaMap = gameTerrain.terrainData.GetAlphamaps(startX, startY, endX - startX, endY - startY);
    int numTextures = alphaMap.GetLength(2);


    for (int x = 0; x < alphaMap.GetLength(0); x++)
    {
        for (int y = 0; y < alphaMap.GetLength(1); y++)
        {
            float distanceToCenter = Vector2.Distance(new Vector2(x + startX, y + startY), new Vector2(mapX, mapY));
            float blendFactor = 1.0f - Mathf.Clamp01(distanceToCenter / brushRadius); // Плавно зменшується до країв
            for (int i = 0; i < numTextures; i++)
            {
                if (i == trackTextureIndex)
                {
                    alphaMap[x, y, i] = Mathf.Max(alphaMap[x, y, i], blendFactor); // Збільшуємо вагу для сліду
                }
                else
                {
                    alphaMap[x, y, i] = Mathf.Max(0, alphaMap[x, y, i] - blendFactor); // Зменшуємо вагу для інших
                }
            }
        }
    }
    gameTerrain.terrainData.SetAlphamaps(startX, startY, alphaMap);
}
```

Додаток Ж

Лістинг програмного коду Камера спостереження.

```
// CameraFollow.cs
public class CameraFollow : MonoBehaviour
{
   public Transform target; // Ціль для слідування (Transform комбайна)
   public Vector3 offset = new Vector3(0, 10, -15); // Зміщення камери відносно цілі
   public float smoothSpeed = 0.125f; // Швидкість згладжування

   void LateUpdate() // LateUpdate викликається після всіх Update-ів, що важливо для камери
   {
      if (target == null) return;

      Vector3 desiredPosition = target.position + target.rotation * offset; // Позиція камери з урахуванням обертання цілі
      Vector3 smoothedPosition = Vector3.Lerp(transform.position, desiredPosition, smoothSpeed);
      transform.position = smoothedPosition;

      transform.LookAt(target.position + Vector3.up * 2f); // Камера дивиться на ціль, трохи вище центру
   }
}
```

Додаток З

Лістинг програмного коду псевдокоду для HarvesterHead.cs (з фокусом на OnTriggerStay)

```
// HarvesterHead.cs
public class HarvesterHead : MonoBehaviour
{
    public FieldManager fieldManager; // Посилання на центральний менеджер поля
    public CombineStorage combineStorage; // Посилання на бункер комбайна
    public float baseHarvestYieldPerUnit = 1.0f; // Базова кількість врожаю, що збирається з одиниці площі/клітинки
    public bool isCuttingActive = false; // Флаг, що вказує, чи активована ріжуча частина жатки

    // Посилання на VFX Graph префаби для стилізованих ефектів
    public GameObject vortexCutEffectPrefab;
    public GameObject grainVortexEffectPrefab;

    void OnTriggerStay(Collider other)
    {
        // Перевіряємо, чи зіткнулися з об'єктом, що є частиною поля або має тег поля
        if (other.CompareTag("FieldSegment") && isCuttingActive)
        {
            // Отримуємо світову позицію центру сегменту поля для запиту до FieldManager
            Vector3 cellWorldPosition = other.bounds.center;

            // Запитуємо FieldManager про поточний стан клітинки поля
```

```
        CellInfo currentCellInfo =
fieldManager.GetCellInfoAtWorldPosition(cellWorldPosition);


        // Перевіряємо, чи клітинка існує, чи врожай дозрів і чи не був він
зібраний раніше
        if (currentCellInfo != null && currentCellInfo.growthStage ==
GrowthStage.Mature && !currentCellInfo.isHarvested)
        {
            // Розраховуємо кількість зібраного врожаю, враховуючи коефіцієнт
врожайності клітинки
            float collectedAmount = baseHarvestYieldPerUnit *
currentCellInfo.yieldMultiplier;


            // Намагаємося додати врожай до бункера комбайна
            if (combineStorage.CanAddCrop(currentCellInfo.cropType,
collectedAmount))
            {
                combineStorage.AddCrop(currentCellInfo.cropType,
collectedAmount); // Додаємо врожай
                fieldManager.MarkCellAsHarvested(cellWorldPosition); //
Позначаємо клітинку як зібрану


                // Активація візуальних ефектів зрізання "Vortex графіки"
                if (vortexCutEffectPrefab != null)
                {
                    // Створення або активація VFX Graph ефекту в точці збирання
                    // Це може бути вихровий ефект, що поглинає рослини
                    Instantiate(vortexCutEffectPrefab, cellWorldPosition,
Quaternion.identity);
                    // В реальному проекті використовувався б Object Pooling для
таких ефектів
```

```
                }
                if (grainVortexEffectPrefab != null)
                {
                    // Активація ефекту потоку зерна, стилізованого під "Vortex"
                    Instantiate(grainVortexEffectPrefab, transform.position,
Quaternion.identity);
                }
            }
            else
            {
                // Бункер комбайна повний, подаємо сигнал гравцеві
                // isCuttingActive = false; // Можливо, автоматично зупиняти
збирання
                // UIManager.DisplayMessage("Бункер комбайна переповнений!
Вивантажте врожай.");
            }
        }
    }

    // Метод, який викликається ззовні (наприклад, з CombineController) для
керування жаткою
    public void SetCuttingActive(bool active)
    {
        isCuttingActive = active;
        // Додати логіку для активації/деактивації візуальних компонентів жатки
(анімації обертання, світло)
        // Можливо, стартувати/зупиняти звукові ефекти роботи жатки
    }
}
```