Київський столичний університет імені Бориса Грінченка

Факультет інформаційних технологій та математики

Кафедра комп'ютерних наук

Допущено до захисту

Завідувач кафедри комп'ютерних наук,

кандидат технічних наук, доцент

(науковий ступінь, вчене звання)

_____ Ірина МАШКІНА

(підпис)

« ___ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»

Спеціальність 122 Комп'ютерні науки

Освітня програма 122.00.01 Інформатика

Тема: РОЗРОБКА ВІДЕОГРИ FACTORY AUTOBATTLER ЗА ДОПОМОГОЮ СЕРЕДОВИЩА UNITY

Виконав:

студент IV курсу денної форми навчання

групи ІНб-2-21-4.0д

Крохмаль Ігор Ігорович _____

(прізвище, ім'я, по батькові) *(підпис)*

Науковий керівник:

кандидат педагогічних наук

(науковий ступінь, вчене звання)

Карпенко Анастасія Сергіївна _____

(прізвище, ім'я, по батькові) *(підпис)*

Київ – 2025

РЕФЕРАТ бакалаврської роботи

Кваліфікаційна робота: 36 с., 11 рис., 1 табл., 38 посилань.

Актуальність:

Об’єктом дослідження є процес розробки відеогри у жанрі autobattler за допомогою середовища Unity

Предмет дослідження є методи та інструменти які використовуються для створення механік, архітектури та інтерфейсу гри

Метою роботи є створення відеогри "Factory autobattler" з унікальними механіками та захоплюючим геймплеєм, який буде відповідати очікуванням сучасних гравців.

Завдання:

1. Провести аналіз ринку відеоігор у жанрі autobattler та вивчити існуючі підходи до побудови ігрових механік.
2. Вивчити особливості використання середовища Unity для розробки відеоігор та обґрунтувати його вибір для реалізації проекту.
3. Розробити концепцію гри "Factory autobattler", визначити цільову аудиторію та унікальні механіки.
4. Створити концептуальний геймплан гри, включаючи структуру рівнів, взаємодію юнітів та систем
5. Реалізувати та протестувати відеогру "Factory autobattler" за допомогою середовища Unity.

ОСНОВНІ РЕЗУЛЬТАТИ:

У процесі роботи було розроблено аналіз ринку конкурентів, виписана їхня характеристика. Вибір рушія. Була визначена цільова аудиторія, та створений концепт гри. Розроблена та протестована демо-версія гри.

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач
кафедри комп'ютерних наук,
кандидат технічних наук, доцент
(науковий ступінь, вчене звання)
_____ Ірина МАШКІНА
(підпис)
« ___ » _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Розробка відеогри Factory autobattler за допомогою середовища Unity»
Виконавець – студент групи ІНб-2-21-4.0д,
спеціальності 122 Комп'ютерні наук,
освітньої програми 122.00.01 Інформатика
Крохмаль Ігор Ігорович

1. Вихідні дані: Законодавство та нормативно-правові акти України в навчальній сфері, наукові роботи та публікації за напрямом дослідження, зокрема з інформатики.
2. Основні завдання: Здійснити огляд публікацій з розробки відеоігор за допомогою середовища Unity. Обґрунтувати мету, об'єкт, предмет та наукові основи дослідження. Розробити завдання, структуру та зміст бакалаврської роботи. Обрати та обґрунтувати методи і засоби для розробки відеоігор за допомогою середовища Unity. Підготувати матеріали до розділів роботи відповідно до індивідуального завдання. Проаналізувати науково-методичні джерела з розробки відеоігор за допомогою середовища Unity. Визначити складові модулі для розробки відеоігор за допомогою середовища Unity, скласти висновки та оформити бакалаврську роботу відповідно до затверджених на кафедрі вимог. Підготувати презентацію за темою роботи.
3. Пояснювальна записка: Обсяг – 36 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.
4. Графічні матеріали: презентація.
5. Строк подання роботи на кафедру: «__» _____ 2025р.
6. Строк подання роботи на кафедру: «___» _____2025 р.

Науковий керівник
кандидат педагогічних наук
Карпенко Анастасія Сергіївна
_____
Дата:

Виконавець:
Крохмаль Ігор Ігорович
_____
Дата:

Календарний план

№	Назва етапу дипломної роботи	Строк виконання етапу роботи	Примітка
1	Формування теми дипломної роботи бакалавра	до 17 жовтня	Виконано
2	Ознайомлення з методичними рекомендаціями до дипломної роботи	до 21 листопада	Виконано
3	Складання змісту та календарного плану роботи	до 20 грудня	Виконано
4	Підбір літератури, складання завдання	до 30 грудня	Виконано
5	Написання реферату та вступу	до 01 лютого	Виконано
6	Написання першого розділу	до 1 лютого	Виконано
7	Написання другого розділу	до 1 березня	Виконано
8	Написання третього розділу	до 31 березня	Виконано
9	Написання висновків	до 20 квітня	Виконано
10	Виправлення зауважень	до 5 травня	Виконано
11	Створення презентації	до 20 травня	Виконано
12	Захист матеріалів дипломної роботи на засіданні кафедри	за 1 місяць до захисту	Виконано
13	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	згідно графіку захистів	

Студент_____

Керівник роботи _____

ЗМІСТ

ВСТУП

Теперішній світ ігор розвивається стрімкими темпами, пропонуючи гравцям нові жанри, механіки та ігрові досвіди. Серед величезної кількості існуючих жанрів особливе місце займають autobattler-ігри, які поєднують у собі елементи стратегічного планування, тактичного мислення та автоматизованих боїв. Цей жанр швидко набув популярності завдяки своїй простоті та захоплюючому геймплеї, що робить його доступним для широкого кола користувачів. Проте, незважаючи на велику кількість ігор цього типу, завжди є простір для нових ідей та оригінальних механік, які можуть зробити досвід гри ще більш захоплюючим.

Одним із найпопулярніших інструментів для розробки відеоігор є середовище Unity, яке надає потужні можливості для створення як простих, так і складних ігрових проектів. Unity відрізняється гнучкістю, масштабованістю та підтримкою великої кількості платформ, що робить його ідеальним вибором для реалізації ідей у жанрі autobattler. Саме тому розробка відеогри "Factory autobattler" у цьому середовищі є актуальним та перспективним напрямком.

Актуальність теми дипломної роботи зумовлена необхідністю створення сучасної гри в жанрі autobattler

Метою роботи є створення відеогри "Factory autobattler" з унікальними механіками та захоплюючим геймплеєм, який буде відповідати очікуванням сучасних гравців.

Для досягнення мети необхідно виконати наступні завдання:

1. Провести аналіз ринку відеоігор у жанрі autobattler та вивчити існуючі підходи до побудови ігрових механік.
2. Вивчити особливості використання середовища Unity для розробки відеоігор та обґрунтувати його вибір для реалізації проекту.
3. Розробити концепцію гри "Factory autobattler", визначити цільову аудиторію та унікальні механіки.

4. Створити концептуальний геймплан гри, включаючи структуру рівнів, взаємодію юнітів та систем.

5. Реалізувати та протестувати відеогру "Factory autobattler" за допомогою середовища Unity.

Об'єктом дослідження є процес розробки відеоігор у жанрі autobattler за допомогою середовища Unity.

Предметом дослідження є методи та інструменти, які використовуються для створення механік, архітектури та інтерфейсу гри.

Для розв’язання поставлених завдань застосовуються методи аналізу та порівняння сучасних рішень, проєктування.

Результати дослідження та розробки мають практичну цінність, оскільки дозволяють глибше зрозуміти процес створення ігор у жанрі autobattler, а також запропонувати новий підхід до ігрових механік, який може бути використаний у подальших проектах.

РОЗДІЛ І. ПЕРЕДУМОВИ РОЗРОБКИ ВІДЕОГРИ FACTORYAUTOBATTLER ЗА ДОПОМОГОЮ СЕРЕДОВИЩА UNITY

1.1 Огляд ринку відеоігор у жанрі autobattler

В IT-секторі ігровий ринок характеризується величезною кількістю жанрів, кожен з яких задовольняє певні потреби гравців. Від стратегій до шутерів - кожен жанр має свої особливості та фанатів.

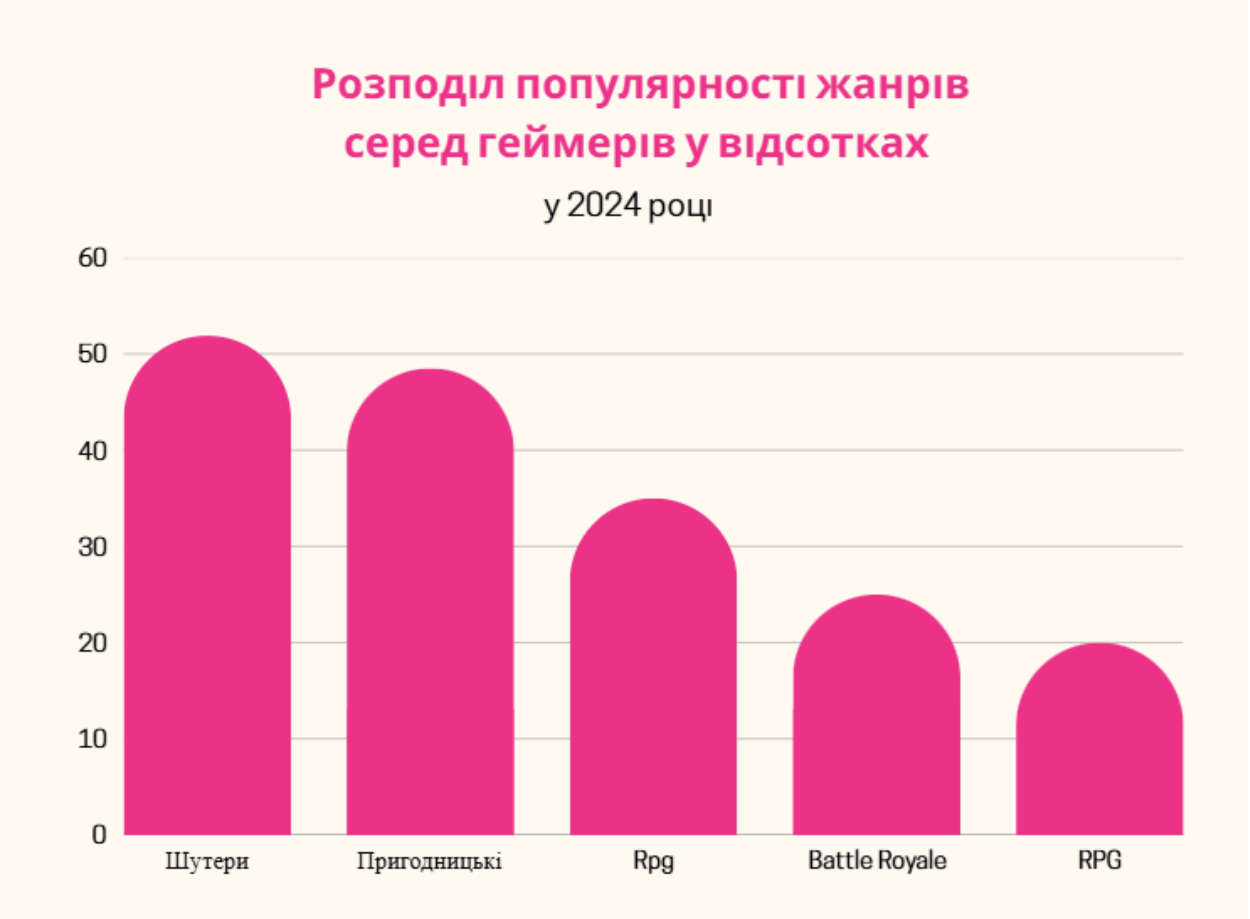

Рис. 1.2 Таблиця Популярності жанрів

Для себе можу виділити такі проєкти що досягли успіху:

1. Dota Auto Chess
2. Teamfight Tactics
3. Hearthstone Battlegrounds

Dota Auto Chess – була випущена 4 січня 2019р як додатковий режим в грі Dota 2. Режим був зроблений китайським розробником Drodo Studio(4 людини).

Коли режим випустили, він був безкоштовним, проте монетизувався віртуальною валютою. Найбільше користувався попитом серед гравців Заходу та Сходу. Його завантажили більше семи мільйонів гравців у Steam, а максимальна кількість одночасних гравців перевищила 300 000 у середині березня. ‘If the mod were a standalone game, this would have made it the #4 most-played title on Steam during this period, after Dota 2, PUBG, and CS:GO.’ – Tianyi Gu.

Режим захопив гравців через стратегічні синергії, автоматичний бій (що є ключовим критерієм автобатлерів), динамічними сесіями, прогресії, що базувалася на навичках гравця.

Рис. 1.3 Скриншот Dota Auto Chess

Teamfigt Tactics – була випущена студією Riot Games, на базі клієнта League of Legends. Вона теж безкоштовна, проте був монетизований Бойовим Пропуском(Battle Pass). Рейтингова система відрізнялася від Dota Auto Chess. Постійна зміна сетів бойових одиниць.

Рис. 1.4 TeamFight Tactics

Hearthstone Battlegrounds – це також режим в оригінальній грі Hearthstone, який мало чим відрізнявся від попередніх представників, проте він був виконаний в світі/стилі warcraft, та всі, хто грав в оригінальну колекційну карткову гру, могли спробувати старих юнітів в новій обгортці.

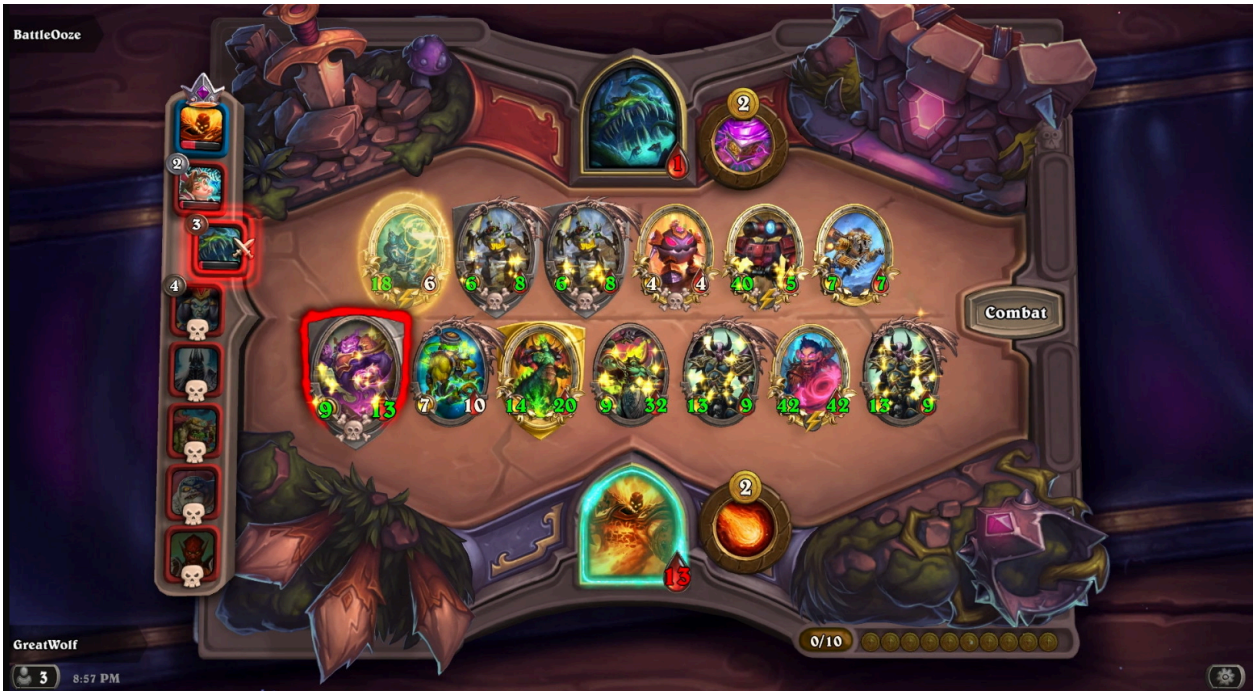

Рис. 1.5 Hearthstone Battlegrounds

Щодо зараз – багато інді студій по розробках ігор слідують тенденціям та випускають ігри в режиму autobattler, в якого вже є прихильники та велика аудиторія.

Tianyi Gu (1 квітня, 2019). Auto Chess Mod Boosts Dota 2's Core PC Player Base by 23% in Two Months and Spawns New Genre. newzoo. https://newzoo.com/resources/blog/auto-chess-mod-boosts-dota-2s-core-pc-player-base-by-23-in-two-months-and-spawns-new-genre

SteamBase
https://steambase.io/games

Results and Trends of the Gaming Market in 2024

https://games.logrusit.com/en/news/game-industry-trends/

Top Game Genres That Are Dominating the Industry in 2024

https://exeengineering.co.uk/top-game-genres-that-are-dominating-the-industry-in-2024/

Most Popular Video Game Genres in 2024: Revenue, Statistics

https://mobilehms.co.in/most-popular-video-game-genres-in-2024-revenue-statistics/

Most Popular Video Game Genres in 2025: Revenue, Statistics

https://rocketbrush.com/blog/most-popular-video-game-genres-in-2024-revenue-statistics-genres-overview

1.2 Аналіз архітектури відеоігор у жанрі autobattler: існуючі підходи до побудови ігрових механік

Жанр автобатлерів, захопливе поєднання стратегії та автоматизації, виник як помітний піджанр в ігровому світі приблизно у 2019 році. Це інноваційне поєднання стратегії та автоматизації проклало шлях до популярності жанру, що призвело до створення окремих ігор. З моменту зародження жанру ігрова механіка зазнала значної еволюції. Ранні ігри спиралися на простий вибір та розміщення юнітів, але з розвитком жанру з'явилися різні інновації, такі як синергії, системи предметів.

Архітектура гри автобатлеру схожа на інші ігри в аспекті використання рушія, проте має певні специфічні особливості через унікальні механіки. Основні компоненти архітектури включають:

Ігровий рушій:

Графічний рушій: Забезпечує відображення 2D чи 3D графіки, а також анімації юнітів.

Аудіо рушій: Відповідає за звукові ефекти та музику, які доповнюють атмосферу.

Фізичний рушій: Використовується для обробки рухів та взаємодій юнітів.

Серверна архітектура (для багатокористувацьких ігор): Забезпечує обробку даних і взаємодію між гравцями.

Модуль ігрової логіки:

Обробляє механіки бою, включаючи зміни станів юнітів (рух, атака).

Управляє "авто" аспектом боїв, коли дії персонажів контролюються алгоритмами.

Інтерфейс користувача: Забезпечує доступ до ключових функцій, таких як управління ресурсами, налаштування та побудова стратегій.

Модуль штучного інтелекту:

Генерація поведінки юнітів.

Прогресія складності ворогів у реальному часі.

Модуль управління ресурсами: Контролює внутрішньоігрові економічні механіки.

Існуючі підходи до побудови механік

Модульність:

Створення систем, де кожен елемент (юніт, клас, здатність) має окремі модулі, які легко комбінуються. Це дозволяє гнучко адаптувати гру до нових механік чи патчів.

Штучний інтелект:

Використовується для створення автономних юнітів, які реагують на зміну ситуації на полі бою.

Рандомізація:

Випадкові ефекти, наприклад, при виборі юнітів або здобичі ресурсів, додають стратегічної глибини, як це робиться в Teamfight Tactics.

Гнучкість стратегій:

Динамічне балансування, де гравець може адаптуватися до обставин на основі поточних подій, наприклад, з'являються бонуси залежно від комбінації юнітів або обраних навичок.

Прогресія складності:

Використання логарифмічної чи арифметичної прогресії для налаштування викликів у грі.

Початковий флоучарт Factory Autobattler гри (рис.1.1)

Рис. 1.1 Флоучарт послідовності розробки продукту

1.3 Огляд та обрання середовища для створення відеогри Factory autobattler

Сучасний ринок ігрових розробок пропонує величезну кількість середовищ (game engines), які допомагають розробникам ефективно створювати відеоігри. Для реалізації проекту "Factory autobattler" було проведено аналіз основних ігрових движків, таких як Unity, Unreal Engine,

Godot, CryEngine та інші. Кожен з них має свої переваги та недоліки, що впливає на вибір оптимального інструменту для конкретного проекту.

1. Аналіз популярних ігрових двигунів:

Unity:

Переваги: Висока гнучкість, підтримка великої кількості платформ (PC, мобільні пристрої, консолі), велике співтовариство та доступність документації, широкий набір інструментів для 2D/3D розробки.

Недоліки: Платна ліцензія для комерційних проектів (при доходах понад $100K на рік), можливі проблеми з продуктивністю при створенні масштабних проектів.

Unreal Engine :

Переваги: Висока якість графіки, потужний візуальний скриптинг (Blueprints), безкоштовна ліцензія (комісія 5% від доходів).

Недоліки: Складний у вивченні для новачків, менша орієнтованість на 2D-проекти, важка інтеграція з деякими сторонніми інструментами.

Godot:

Переваги: Безкоштовний, легкий у використанні, добре підходить для навчання та невеликих проєктів.

Недоліки: Менше співтовариство та обмежена кількість готових ассетів, порівняно з Unity та Unreal Engine.

CryEngine:

Переваги: Висока якість графіки, підтримка фотореалістичних ефектів.

Недоліки: Менш популярний, складний у вивченні, менша кількість документації.

2. Обґрунтування вибору Unity:

Після аналізу всіх доступних інструментів було вирішено використовувати середовище Unity для розробки відеогри "Factory autobattler". Основними факторами такого рішення стали:

Гнучкість та масштабованість : Unity дозволяє створювати як прості 2D-ігри, так і складні 3D-проекти, що надає можливість розширити функціональність гри в майбутньому.

Підтримка великої кількості платформ : Гра може бути легко адаптована для PC, мобільних пристроїв та консолей, що розширює її цільову аудиторію.

Велике співтовариство та документація : Unity має активне співтовариство, велику кількість туторіалів, форумів та готових рішень, що значно полегшує процес розробки.

Доступність інструментів : Unity надає широкий набір інструментів для створення механік, анімацій, інтерфейсів та інших компонентів гри.

Інтеграція з сторонніми сервісами : Unity легко інтегрується з такими сервісами, як Firebase (для аналітики та баз даних), AdMob (для реклами) та іншими, що спрощує монетизацію гри.

Таким чином, Unity є найоптимальнішим вибором для створення відеогри "Factory autobattler", оскільки воно поєднує в собі простоту використання, гнучкість та потужність, необхідні для успішної реалізації проекту.

1.4 Ознайомлення з особливостями використання середовища Unity для розробки відеогри Factory autobattler

Unity є одним із найпопулярніших ігрових движків, який широко використовується для створення 2D та 3D ігор. Для успішної розробки відеогри "Factory autobattler" важливо детально зрозуміти особливості його використання, інструменти та можливості, які він надає.

1. Основні компоненти Unity:

Scene (Сцена): Це основне середовище, де розробник створює ігровий світ. У "Factory autobattler" сцени будуть використовуватися для створення ігрового поля, меню та інших елементів.

GameObjects (Ігрові об'єкти): Усі елементи гри, такі як одиниці, будівлі, кнопки тощо, представлені як GameObjects. Кожен об'єкт може мати

компоненти, наприклад, Sprite Renderer (для зображень), Rigidbody (для фізики) або Collider (для взаємодії).

Components (Компоненти) : Компоненти додають функціональність до GameObjects. Наприклад, для реалізації механік бою в "Factory autobattler" будуть використовуватися скрипти (Scripts) та анімації (Animations).

Assets (Асети): Це ресурси, які використовуються в грі, такі як текстури, моделі, звуки та скрипти. Unity має власний Asset Store, де можна знайти готові ресурси для швидкого старту.

2. Особливості розробки механік гри:

Скриптинг: Unity використовує мову програмування C# для створення механік гри. Для "Factory autobattler" будуть написані скрипти для реалізації автоматичного бою, синергій між юнітами, управління ресурсами та інших механік.

UI (User Interface): Unity надає інструменти для створення інтерфейсу гри. Для "Factory autobattler" буде створено меню, панель управління юнітами, таблицю лідерів та інші елементи.

Animation System: Система анімацій Unity дозволяє створювати плавні переходи між станами об'єктів. Наприклад, анімації будуть використовуватися для демонстрації руху юнітів та їх атак.

3. Інтеграція з сторонніми сервісами:

Firebase: Unity може інтегруватися з Firebase для зберігання даних користувачів, аналітики та рейтингів.

AdMob: Для монетизації гри через рекламу можна використовувати AdMob.

Photon: Можна використовувати Photon для мережевої взаємодії.

4. Особливості тестування та оптимізації:

Play Mode: Unity має режим тестування (Play Mode), який дозволяє перевіряти механіки гри без необхідності повного запуску.

Profiler : Інструмент Profiler допомагає аналізувати продуктивність гри та знаходити "вузькі місця".

Build Settings : Unity надає можливість легко створювати версії гри для різних платформ, що є важливим для публікації.

5. Переваги Unity для "Factory autobattler":

- Швидке прототипування завдяки готовим асетам та інструментам.
- Підтримка 2D-графіки, що є достатньою для реалізації механік autobattler.
- Можливість легко інтегрувати сторонні сервіси для аналітики та монетизації.
- Активне співтовариство, яке допомагає вирішувати проблеми.

Таким чином, Unity надає всі необхідні інструменти для створення відеогри "Factory autobattler", забезпечуючи ефективність розробки, гнучкість та можливість подальшого розширення функціоналу.

РОЗДІЛ 2. МЕТОДИКА РОЗРОБКИ ВІДЕОГРИ FACTORY AUTOBATTLER ЗА ДОПОМОГОЮ СЕРЕДОВИЩА UNITY

2.1 Створення концепції відеогри Factory autobattler: сценарій, цільова аудиторія, унікальні особливості

Проаналізувавши продукти, та інструменти, дослідивши рушій, та надихнувшись схожими проектами, я почав створювати геймплан, дизайн, механіки.

Жанр: 2.5D стратегія - автобатлер

Сеттінг: стімпанк

Core mechanics: менеджмент фабрик

Мета рівня: знищити ворожу базу

Обрав жанр стратегії, так як, мені, як розробнику цікаво реалізовувати різні механіки пов’язані з стратегічним планування геймплею.

Пітч:

Ви опиняєтесь на 2д стімпанк рівні, де у вашому розпорядженні є ваша база, внутрішньо-ігрова валюта, фабрики, та ваш заклятий ворог, ваша мета знищити базу ворога.

Використовуючи надані ресурси та особливості фабрик, що випускають бойові машини, потрібно наростити більше сили ніж ворог та бути на ‘крок попереду’, на бій будуть впливати особливості фабрик, роботів, карта, та погодні умови

Детальніший опис:

В цій singleplayer грі вам потрібно будувати різні фабрики з руки які продукують бойових роботів для того щоб перемогти ворога.

Фабрики можна буде прокачувати даючи їм різні корисні особливості(можливо і негативні, щоб значно підвищити сильний ефект, але нівелювати слабшим негативним)

Цей автобатлер буде з арифметичною прогресією ШІ ворога.

Тим часом, поки фабрики по cooldown спавнять юнітів, самі юніти взаємодіють на полі бою. В кожного юніта будуть свої цікаві особливості.

Після визначення з сценарієм, та загальними відомостями про гру, створимо портрет потенційного клієнта та порівняємо з конкурентами:

Цільова аудиторія:

- Гравці, які люблять стратегічне планування та тактичне мислення.
- Користувачі, знайомі з жанром autobattler (наприклад, Dota Auto Chess, Teamfight Tactics).
- Новачки, які шукають просту, але захоплюючу гру.
- Мобільні гравці, які віддають перевагу швидким сесіям.
- Гравці, яким подобаються візуальні ефекти та деталізовані графічні елементи
- Користувачі, які хочуть з новим подихом переглянути жанр autobattler

Порівняння з існуючими іграми

Сильні сторони конкурентів:

Dota Auto Chess: Стратегічні синергії, велика кількість юнітів.

Teamfight Tactics: Регулярні оновлення механік (сети), бойовий пропуск.

Hearthstone Battlegrounds: Юніти з унікальними механіками, стиль Warcraft.

Переваги над існуючими іграми

- Присутність механік, пов’язаних з управлінням ресурсами (наприклад, фабриками чи виробництвом).
- Унікальні особливості юнітів, які роблять гру ще більш захоплюючою.
- Steampunk всесвіт – всесвіт парових механізмів, який дає змогу обдумувати та додавати більше унікальних комбінацій

2.2 Добір інструментів середовища Unity для реалізації механік при розробці відеогри Factory autobattler

Розробка відеогри "Factory autobattler" у середовищі Unity вимагає детального аналізу та обрання інструментів, які будуть використовуватися для реалізації ключових механік гри. Unity надає широкий набір інструментів, які можна адаптувати під специфіку проекту. Розглянемо основні механіки гри та інструменти, які будуть використані для їх реалізації.

1. Реалізація механіки автоматичного бою

Автоматичний бій є однією з ключових механік жанру autobattler. Для його реалізації потрібні інструменти, які дозволяють керувати поведінкою юнітів, обробляти взаємодії між ними та демонструвати результати бою.

Скриптинг (C#):

- Створення скриптів для управління логікою бою:
- Визначення цілей для атаки.
- Обчислення пошкоджень на основі характеристик юнітів.
- Реалізація механізму смерті юнітів.
- Обчислення покращень до характеристик юніта під час бою.
- Стан закінчення автоматичного бою.
- Спрацьовування унікальних ефектів юнітів.

Animation System:

Додавання анімацій для атаки, отримання пошкоджень, смерті юнітів, графічного інтерфейсу, фабрик, елементів карти. Використання Animator Controller для створення переходів між станами.

Physics Engine:

Використання компонентів Collider та Rigidbody для фізичної взаємодії між юнітами для реалізації механіки зіткнень між ігровими об’єктами.

2. Управління юнітами та їх синергіями

Механіка синергій є важливою частиною гри, оскільки вона дозволяє гравцям створювати стратегічні комбінації між юнітами.

Scriptable Objects:

Використання Scriptable Objects для зберігання даних про юніти, їх характеристики та синергії. Приклад: Об'єкт UnitData, який містить інформацію про здоров'я, силу атаки, швидкість атаки, дальність атаки, швидкість руху, тип юніта та його унікальні механіки.

UI Elements:

Створення інтерфейсу для відображення доступних фабрик та їх синергій.

Використання компонентів, таких як Canvas, Button, Image, Text, для взаємодії з гравцем.

Event System:

Використання Unity Events для відстеження активних синергій та їх ефектів на полі бою.

3. Збереження прогресу та рейтингової системи

Для забезпечення комфортного досвіду гравців важливо реалізувати механіки збереження прогресу та рейтингової системи.

PlayerPrefs:

Використання PlayerPrefs для збереження базових даних користувача (наприклад, рівень гравця, отримані досягнення).

Обмеження: Простота, але не підходить для масштабних даних.

Firebase Integration:

Використання Firebase Realtime Database або Firestore для зберігання рейтингів гравців та їх прогресу.

Переваги: Можливість синхронізації даних між пристроями та онлайн-відстеження результатів.

Leaderboard System:

Використання UI для відображення таблиці лідерів.

Інтеграція з Firebase для автоматичного оновлення рейтингів.

4. Інтерфейс користувача (UI)

Інтерфейс гри є ключовим елементом взаємодії з гравцем. Він повинен бути інтуїтивним та зручним.

Canvas:

Використання Canvas для створення меню, панелей управління та інших елементів інтерфейсу.

Приклад: Панель для вибору юнітів, індикатори здоров'я та ресурсів.

UI Toolkit:

Використання Unity UI Toolkit для створення складних інтерфейсів (якщо потрібна велика кастомізація).

Audio Source:

Додавання звукових ефектів для кнопок та інших інтерактивних елементів.

5. Тестування та оптимізація механік

Для забезпечення стабільності та продуктивності гри важливо використовувати інструменти Unity для тестування та оптимізації.

Play Mode:

Використання Play Mode для швидкого тестування механік без необхідності повного запуску гри.

6. Додаткові інструменти для покращення геймплею

Particle System:

Додавання частинкових ефектів для вибухів, пошкоджень та інших візуальних елементів.

Post-Processing:

Використання Post-Processing Stack для покращення графіки (ефекти затемнення, розмиття тощо).

Asset Store:

Використання готових асетів з Unity Asset Store для прискорення розробки (наприклад, шаблони UI, звуки, анімації).

РОЗДІЛ 3. РОЗРОБКА ВІДЕОГРИ FACTORY AUTOBATTLER ЗА ДОПОМОГОЮ СЕРЕДОВИЩА UNITY

3.1 Розробка концептуального геймплану відеогри Factory autobattler за допомогою середовища Unity: структура рівнів, взаємодія юнітів і систем

Для початку, користуючись раніше заготовленим флоучартом я почав з розробки карти

```
private void Start()
    {
        GenerateGrid();
    }
    void GenerateGrid()
    {
        GameObject gridParent = new GameObject("GridParent");

        for (int x = 0; x<_width; x++)
        {
            for (int y = 0; y < _height; y++)
            {
                var spawnedTile = Instantiate(_tilePrefab, new Vector3(x, 0,
y), Quaternion.identity);

                spawnedTile.transform.SetParent(gridParent.transform);

                spawnedTile.name = $"Tile x:{x}, y:{y}";
                spawnedTile.SetCoordinates(x, y);

                var tile = FindObjectOfType<Tile>();
                //Debug.Log($"Tile coordinates: X={tile.X}, Y={tile.Y}");

                var isOffset = (x%2 == 0 && y%2 != 0) || (x%2 != 0 && y%2 ==
0);
                spawnedTile.Init(isOffset);
            }
        }

        _cam.transform.position = new Vector3((float)_width / 2 - 0.5f,
(float)_camHeight, -3);
    }
```

Таким чином у нас генерується карта з тайлів за допомогою функції `GenerateGrid()`, розмір якої можна змінити

Вигляд карти представлено на рисунку 3.1.

Наступним кроком, я розробив мехініку будівництва фабрики, але перед тим, щоб покращити ігровий досвід, я зробив підсвітку тайлів, щоб гравець інтуітивно краще розумів, на якому тайлі знаходиться його курсор:

```
private void OnMouseEnter()
    {

        if (BuildRule._isBuilding)
```

```
        {
            highlight.SetActive(true);
            HighlightNeighbors(highlightId, X, Y);
            BuildRule.CurrentTileX = X;
            BuildRule.CurrentTileY = Y;
        }
        else
        {
            BuildRule.BuildingState = false;
            highlightBuildDeny.SetActive(true);
        }

    }
    private void OnMouseExit()
    {
        BuildRule.DisableAllHighlights();
    }
```

Функції **OnMouseEnter()** та **OnMouseExit()** вбудовані в інтерфейс Unity, та зчитують інформацію, коли курсор знаходиться над певним об'єктом, та залишає зону над певним об'єктом, підсвічування зображене на рисунку 3.2.

В функції присутнє розгалуження, яке відповідає за колір підсвітки тайлу, коли можна будувати, а коли – ні. Колір підсвітки, коли дозволено будувати – зображено на рисунку 3.3.

Після цього я додав декілька типів фабрик з різними розмірами:

```
Dictionary<int, List<Vector2Int>> neighborOffsets = new Dictionary<int, List<Vector2Int>>()
 {
     { 0, new List<Vector2Int> { new Vector2Int(1, 0), new Vector2Int(1, -1) } },
     { 1, new List<Vector2Int> { new Vector2Int(0, -1) } },
     { 2, new List<Vector2Int> { new Vector2Int(0, -1), new Vector2Int(-1, -1), new Vector2Int(-2, -1) } }
 };
```

Цифри в дужках відповідають за сусідні тайли, відступи по координатам X та Y.

Від чого і відштовхувався, при підсвітці сусідніх тайлів, коли ми хочемо поставити одну із фабрик. Підсвічування тайлів коли ми хочемо поставити фабрику зображено на рисунку 3.4.

Для того щоб протестувати виставляння фабрик та взагалі зробити механізм будівництва, я використав UI інструменти Unity, та прив'язав до кожної кнопки відповідний функціонал, меню зображено на рисунку 3.5.

Одна з функцій, яка прив'язана до кнопки, що передає данні до скрипта, в якому оперується вся інформація пов'язана з будуванням:

```
    public void BuildMechPlant()
```

```
    {
        TileFinder(0);
        plantBuildRule.BuildingState = true;
    }
```

І на кінець, для будування фабрик, я використав технологію Raycast, яка створює промінь від центру камери до будь якої іншої точки, в моєму випадку, де була натиснута ліва кнопка миші:

```
void Update()
    {
        if (Input.GetKeyUp(KeyCode.B))
        {
            BuildMode();
        }

        if (Input.GetKeyUp(KeyCode.Mouse0) && BuildingState)
        {
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;

            if (Physics.Raycast(ray, out hit))
            {
                Tile selectedTile = hit.collider.GetComponent<Tile>();
                if (selectedTile != null && StartLevel.scrapCount >=
buildingCost)
                {
                    CurrentTileX = selectedTile.X;
                    CurrentTileY = selectedTile.Y;
                    BuildPlant();

                }
            }
        }
    }
Функція BuildPlant():
public void BuildPlant()
{


    EnsureFactoryContainerExists();

    GameObject newFactory = Instantiate(ChoosePlant(), new
Vector3(CurrentTileX, 1, CurrentTileY), Quaternion.identity);

    // Додаємо фабрику до контейнера
    if (newFactory != null)
    {
        newFactory.transform.SetParent(factoryContainer.transform);
        StartLevel.AddScrap(-buildingCost);
    }
    BuildingState = false;
    Debug.Log("Factory with ID: " + factoryPrefabID + ". On tile X: " +
CurrentTileX + " Y: " + CurrentTileY);

    ResetTileHighlightId();
    if (buildingMechanic != null)
    {
        buildingMechanic.IterateFactoriesAndBlocks();
    }
}
```

Ключовий принцип в цій функції це вбудований інструмент `Instantiate()`, який дозволяє відобразити на сцені певні елементи. Побудована фабрика відображена на рисунку 3.6.

Потім був зроблений перегляд початкового флоучарту, і було зроблене рішення для початку зробити базову механіку грошей та ходу:

```
void Start()
{
    AddScrap(3);
    UpdateScrapText();
}

public void AddScrap(int amount)
{
    scrapCount += amount;
    UpdateScrapText();
}

private void UpdateScrapText()
{
    scrapCountText.text = $"{scrapCount}";
}
```

Та візуальний інтрефейс для них, що зображений на рисунку 3.7.

Після цього переглянув механіку будівництва фабрики, щоб можна було будувати, лише коли вистачає ресурсів:

```
StartLevel.AddScrap(-buildingCost);
```

Умова:

```
if (selectedTile != null && StartLevel.scrapCount >= buildingCost)
```

Таким чином, я повністю розробив карту, перші фарики, будівництво, базову механіку ходу, та ресурси.

3.2 Моделювання архітектури відеогри Factory autobattler за допомогою середовища Unity

Архітектура гри будується на принципах модульності та чіткого розділення обов’язків між компонентами. Основні частини архітектури:

Game Manager (Менеджер гри):

Відповідає за загальну логіку гри, таку як ініціалізація сесій, управління станом гри (початок, пауза, завершення) та збереження прогресу. Реалізується через клас GameManager, який є центральним контролером.

Unit System (Система юнітів):

Включає класи для створення, управління та взаємодії юнітів.

Приклад: Клас Unit для базових характеристик юнітів, клас UnitController для
керування поведінкою.

Battle System (Система бою):

Відповідає за автоматичний бій, обчислення пошкоджень та відстеження результатів. Реалізується через клас BattleManager.

Resource System (Система ресурсів):

Управління ресурсами гравця. Реалізується через клас ResourceManager.

UI System (Система інтерфейсу):

Відповідає за взаємодію з гравцем через меню, панелі управління та індикатори. Реалізується через Canvas та UI компоненти (наприклад, UIManager).

Data Management (Керування даними):

Зберігання та використання даних про юніти, ресурси, синергії тощо.

Реалізується через Scriptable Objects та Firebase Integration.

Деталізація основних компонентів архітектури

Game Manager

Game Manager є центральним компонентом архітектури, який координує взаємодію між усіма системами гри.

Функціональність:

- Ініціалізація гри (завантаження налаштувань, створення об’єктів).
- Відстеження стану гри (активна сесія, пауза, завершення).
- Збереження та завантаження прогресу гравця.
- Обробка подій (наприклад, закінчення ходу, перемога/поразка).

Unit System

Юніти є основними акторами гри, тому їх система має бути добре структурованою.

Функціональність:

- Створення юнітів на основі даних (Scriptable Objects).

- Управління характеристиками (здоров’я, сила атаки, тип).
- Реалізація механік синергій.

Battle System

Система бою реалізує механіку автоматичного бою та обробляє взаємодії між юнітами.

Функціональність:

- Визначення черги атаки.
- Обчислення пошкоджень та відстеження смерті юнітів.
- Відображення результатів бою.

Resource System

Система ресурсів дозволяє гравцеві керувати виробництвом та використанням ресурсів.

Функціональність:

- Відстеження доступних ресурсів.
- Механіки витрати ресурсів.
- Взаємодія з фабриками для покращення виробництва.

UI System

Інтерфейс гри є ключовим елементом взаємодії з гравцем.

Функціональність:

- Відображення ресурсів, юнітів та інших даних.
- Меню управління грою (пауза, налаштування).
- Таблиця лідерів та рейтингова система.

3.3 Тестування відеогри Factory autobattler розробленої за допомогою середовища Unity

Тестування є невід’ємною частиною процесу розробки відеогри, оскільки воно забезпечує якість, стабільність та користувацький досвід. У цьому розділі розглядаються методи, інструменти та результати тестування відеогри "Factory autobattler", розробленої у середовищі Unity. Мета тестування полягає у виявленні помилок (багів), оптимізації продуктивності

та забезпеченні відповідності гри очікуванням гравців.

1. Цілі тестування

Основними цілями тестування є:

- Перевірка функціональності механік гри.
- Виявлення та усунення багів або непрацюючих елементів.
- Оцінка продуктивності гри на різних пристроях.
- Забезпечення інтуїтивності та зручності інтерфейсу для гравців.
- Гарантія стабільності гри під час довгих сесій.

2. Етапи тестування

Тестування проводиться на кількох етапах, кожен з яких має свої специфічні завдання.

2.1. Функціональне тестування

Функціональне тестування перевіряє, чи всі механіки гри працюють коректно.

Перевірка механік:

Чи правильно обчислюються пошкодження? Чи відбувається синхронізація між юнітами? Чи активуються унікальні механіки при виконанні умов? Чи правильно збираються та витрачаються ресурси?

Методи тестування:

Ігрові сценарії: Створення тестових ситуацій для перевірки механік.

Play Mode: Використання режиму тестування в Unity для швидкої перевірки.

2.2. Тестування інтерфейсу (UI)

UI тестування перевіряє, чи інтерфейс гри є зручним та інтуїтивним.

Перевірка UI:

Чи правильно показуються ресурси, юніти та інші параметри? Чи працюють кнопки, меню та інші елементи управління? Чи коректно виглядає інтерфейс на різних пристроях (PC, мобільні телефони)?

Методи тестування:

Візуальна перевірка: Використання Canvas в Unity для перевірки розміщення елементів.

Користувацьке тестування: Залучення тестерів для оцінки зручності інтерфейсу.

2.3. Бета-тестування

Бета-тестування проводиться з участю реальних гравців для отримання зворотного зв’язку.

Цілі бета-тестування:

- Оцінка загального досвіду гри.
- Виявлення проблем, які не були помічені на попередніх етапах.
- Збір відгуків про механіки, інтерфейс та графіку.

Методи:

Краудтестинг: Розміщення гри на платформах, таких як Steam Early Access або Google Play Beta.

Опитування: Збір відгуків від гравців через онлайн-форми.

На даний момент були тестування в ході розробки, було виявлено декілька багів пов’язані з будівництвом, де неправильно обчислювалися сусідні клітинки, та коли ми обирали фабрику в меню будівництва, вона відразу ставилася, зараз такого немає, ці баги були виправлені.

ВИСНОВКИ

Для реалізації механік відеогри "Factory autobattler" у середовищі Unity було обрано комплекс інструментів, які забезпечують гнучкість, ефективність та масштабованість проекту. Основними інструментами є скриптинг (C#), Animation System, UI Elements, Firebase Integration та інші компоненти Unity.

Унікальні особливості відеогри "Factory autobattler" можна визначити через аналіз цільової аудиторії, порівняння з існуючими іграми та впровадження інноваційних механік. Основними USP гри є: Фабрична тематика та індустріальний дизайн, управління ресурсами та виробництвом юнітів, унікальні механіки синергій, пов’язані з виробництвом.

Ці особливості роблять гру привабливою для цільової аудиторії та виділяють її на тлі конкурентів.

Моделювання архітектури відеогри включає розробку чітко структурованих систем, таких як Game Manager, Unit System, Battle System, Resource System та UI System. Архітектура базується на принципах модульності, що забезпечує легкість підтримки та масштабування проекту. Реалізація кожної системи в Unity проводиться за допомогою скриптів (C#), Scriptable Objects та інструментів для створення інтерфейсу. Такий підхід дозволяє створити стабільну, ефективну та захоплюючу гру, яка задовольнить очікування гравців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Michael Moore. *Basics of Game Design*. CRC Press, 2016. 400 p.
2. Mark J. P. Wolf. *Encyclopedia of Video Games: A-L*. ABC-CLIO, 2012. 763 p.
3. В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.
4. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.
5. Tianyi Gu (1 квітня, 2019). Auto Chess Mod Boosts Dota 2's Core PC Player Base by 23% in Two Months and Spawns New Genre. newzoo. https://newzoo.com/resources/blog/auto-chess-mod-boosts-dota-2s-core-pc-player-base-by-23-in-two-months-and-spawns-new-genre
6. Understanding Game Architecture. study tonight. https://www.studytonight.com/3d-game-engineering-with-unity/game-development-architecture
7. Unity Real-Time Development Platform. Unity. https://unity.com/
8. Welcome to Unity Learn. Unity Learn. https://learn.unity.com/
9. Unity 6 User Manual. Unity Documentation. https://docs.unity3d.com/6000.0/Documentation/Manual/UnityManual.html
10. Teamfight Tactics autobattler is played by 33 million people every month https://app2top.com/industry/teamfight-tactics-autobattler-is-played-by-33-million-people-every-month-150927.html
11. SteamBase https://steambase.io/games
12. Results and Trends of the Gaming Market in 2024 https://games.logrusit.com/en/news/game-industry-trends/

13. Top Game Genres That Are Dominating the Industry in 2024
https://exeengineering.co.uk/top-game-genres-that-are-dominating-the-industry-in-2024/
14. Most Popular Video Game Genres in 2024: Revenue, Statistics
https://mobilehms.co.in/most-popular-video-game-genres-in-2024-revenue-statistics/
15. Most Popular Video Game Genres in 2025: Revenue, Statistics
https://rocketbrush.com/blog/most-popular-video-game-genres-in-2024-revenue-statistics-genres-overview
16. Auto-Battler Genre Evolution and Variants
https://magicspecialevents.com/2025/01/29/auto-battler-genre-evolution-and-variants/
17. How Dota 2 pioneered and flopped the Auto Battler genre
https://ballers.ph/esports/how-dota-2-pioneered-and-flopped-the-auto-battler-genre/
18. Auto Battler Game Development Guide
https://ilogos.biz/auto-battler-game-development-guide/
19. Design and implementation of an Intelligent Agent for Auto Battler Games UsingReinforcement Learning
https://lutpub.lut.fi/bitstream/handle/10024/167789/BachelorThesis_Wenyue_Zhang.pdf;jsessionid=76B702D72857B48C76C212A888356A9D?sequence=1
20. The system design of the auto battler game
https://www.researchgate.net/figure/The-system-design-of-the-auto-battler-game_fig1_390043527
21. «Any tips on designing an autobattler system?»
https://www.reddit.com/r/gamedesign/comments/13snufn/any_tips_on_designing_an_autobattler_system/?rdt=43029
22. Game design deep dive: what makes autobattlers fun?
https://www.youtube.com/watch?v=MgvZ3ohbveU

23. Metagame Autobalancing for Competitive Multiplayer Games
https://scispace.com/pdf/metagame-autobalancing-for-competitive-multiplayer-games-bfctvi2cwr.pdf
24. We Need to Talk About Auto-Battlers
https://www.youtube.com/watch?v=_v-syxwPfyM
25. Designing games for no one
https://benal.itch.io/blog/devlog/484541/designing-games-for-no-one?utm_source=chatgpt.com
26. Functional Unity Architecture: A Developer's Guide
https://medium.com/@bada/functional-unity-architecture-a-developers-guide-359e5111c5c2
27. Unity Game Development Engine: A Technical Survey
https://www.researchgate.net/publication/348917348_Unity_Game_Development_Engine_A_Technical_Survey
28. Unity Architecture: GameObject Component Pattern
https://medium.com/@simon.nordon/unity-architecture-gameobject-component-pattern-34a76a9eacfb
29. «Auto Battle Algorithm ? »
https://discussions.unity.com/t/auto-battle-algorithm/814046
30. Pathfinding Algorithms & Implementations In Unity
https://aliemreonur.medium.com/pathfinding-algorithms-implementations-in-unity-6067bc4c745b
31. Unity Advanced Tutorial - How to make an Auto-Battle game
https://www.youtube.com/watch?v=AO-ik5fKIuI
32. Setting Up Combat Architecture In Unity
https://chandler-lane.medium.com/setting-up-combat-architecture-in-unity-996096620ef6
33. Template Concept Document for Auto-Battler Games
https://medium.com/gaming-industry-documents-for-every-game-genre/template-concept-document-for-auto-battler-games-2ea7b80cd99c

ДОДАТКИ

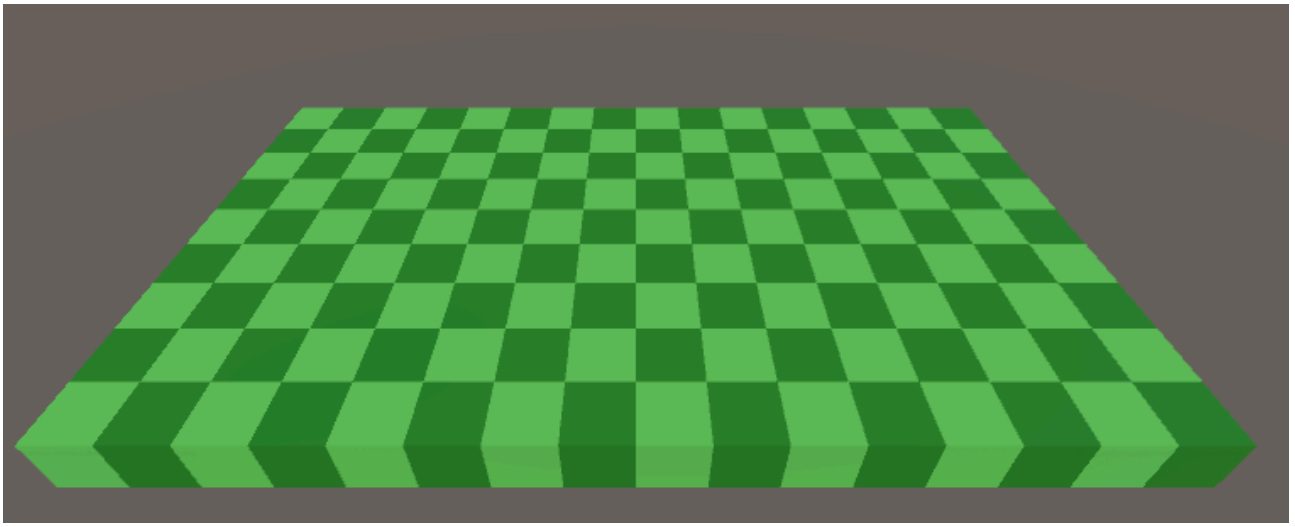

Рис. 3.1 Вигляд карти розміром 16x9

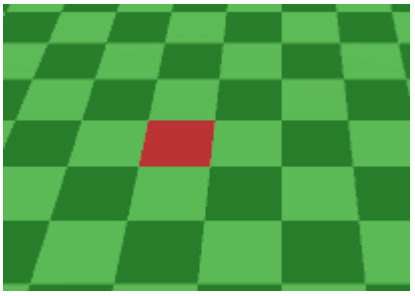

Рис. 3.2 Відображення підсвітки тайлу

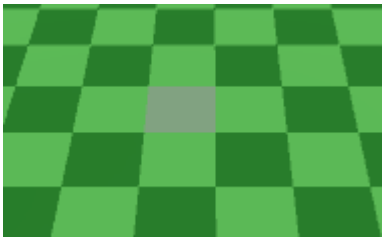

Рис. 3.3 Відображення підсвітки з дозволом будування

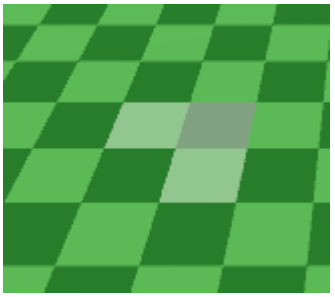

Рис. 3.4 Підсвітка тайлів, коли ставимо Melle фабрику

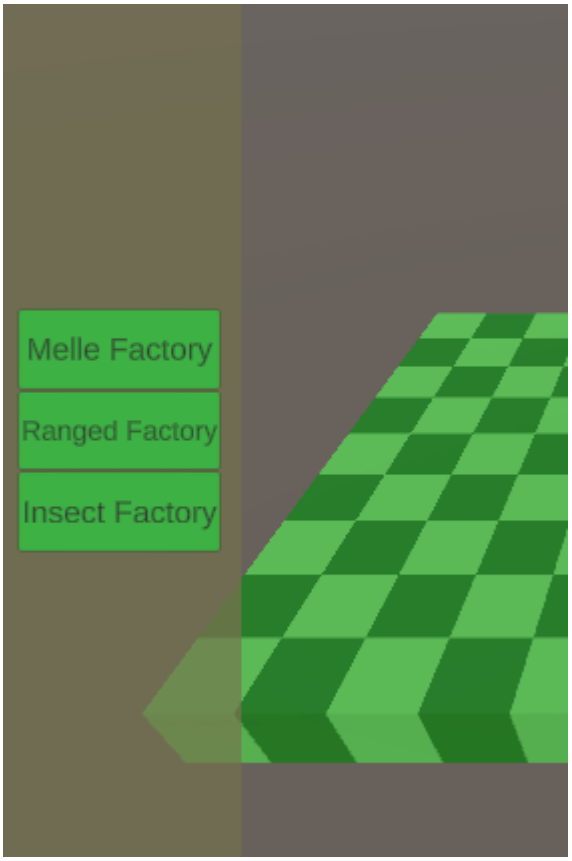

Рис. 3.5 Меню будівництва

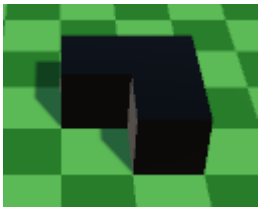

Рис. 3.6 Побудована фабрика

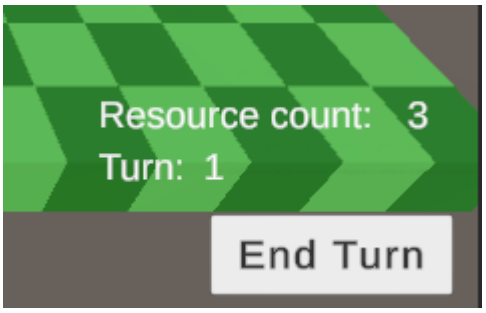

Рис. 3.7 Відображення функціоналу ресурсів, ходу