ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «РОЗРОБКА ГРИ В ЖАНРІ ПОКРОКОВА СТРАТЕГІЯ ЗА ДОПОМОГОЮ ІГРОВОГО РУШІЯ UNITY МОВОЮ C#»

Виконав: студентка 4 курсу, групи ПД–44 спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності/спеціалізації)

Ярошевський О.В.

(прізвище та ініціали)

Керівник Дібрівний О.А.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____ Негоденко О.В.
"____" _____ 2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

ЯРОШЕВСЬКОМУ ОЛЕКСАНДРУ ВІКТОРОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка гри в жанрі Покрокова стратегія за допомогою ігрового рушія Unity мовою C#»

Керівник роботи: Дібрівний Олесь Андрійович, доктор філософії, доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26.

2. Строк подання студентом роботи «1» червня 2023 року

3. Вхідні дані до роботи
   3.1. Методи розробки відеоігрових програмних продуктів:
   3.2. Офіційна документація Unity:
   3.3. Офіційна документація мови програмування C#.

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити)
   4.1. Дослідження та аналіз існуючих ігор в жанрі Покрокова стратегія.
   4.2. Визначення вимог до розробки гри.
   4.3. Проектування архітектури гри.
   4.4. Реалізація механік гри з використанням ігрового рушія Unity та мови програмування C#.
   4.5. Розробка інтерфейсу та графічного відображення гри.
   4.6. Тестування гри.

5. Перелік демонстраційного матеріалу (назва основних слайдів)
   5.1. Мета, об’єкт та предмет дослідження
   5.2. Задачі дипломної роботи
   5.3. Аналіз аналогів
   5.4. Концепт гри
   5.5. Вимоги до гри
   5.6. Програмні засоби реалізації
   5.7. Діаграма класів
   5.8. Діаграма діяльності
   5.9. Блок-схема алгоритму основної механіки
   5.10. Екранні форми
   5.11. Апробація результатів дослідження
   5.12. Висновки

6. Дата видачі завдання «24» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	24.02.2023	Виконано
2	Дослідження аналогів та актуальності гри	01.03.2023	Виконано
3	Аналіз та вибір інструментів для розробки гри	10.03.2023	Виконано
4	Проектування та реалізація	20.03.2023	Виконано
5	Вступ, висновки, реферат	21.04.2023	Виконано
6	Розробка обов‘язкових демонстраційних матеріалів	22.05.2023	Виконано
7	Попередній захист роботи	26.05.2023	Виконано
8	Здача роботи	01.06.2022	

Студент _____ Ярошевський О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дібрівний О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи с. 60, 10 рис.,18 джерел.

Ключові слова: відеогра, геймплей, жанр, механіка, ігровий двигун, покрокова стратегія.

Об'єкт дослідження – геймплей в жанрі Покрокова стратегія.

Предмет дослідження – програмне забезпечення для реалізації геймплею гри в жанрі Покрокова стратегія.

Мета роботи – підвищення якості гри в жанрі Покрокова стратегія за допомогою ігрового рушія Unity та мови програмування C#.

Наукова новизна – розроблені алгоритми реалізації гри в жанрі Покрокова стратегія.

В роботі проведено детальний аналіз ринку комп'ютерних ігор в жанрі покрокових стратегій. Були виявлені та проаналізовані переваги та недоліки різних інструментів, використовуваних для розробки таких ігор.

Для розробки комп'ютерної гри в жанрі покрокових стратегій був використаний ігровий рушій Unity. Він надає широкі можливості для створення складної логіки та графічного представлення гри. Код гри написаний з використанням мови програмування C# в інтегрованому середовищі розробки Visual Studio 2022.

Галузь використання – розвага для геймерів, розвиток стратегічного мислення, розвиток соціальної взаємодії.

ЗМІСТ

ВСТУП

У сучасному світі комп'ютерні ігри є однією з найпопулярніших форм розваги та дозвілля. Вони здатні не лише надати емоційний відпочинок, але й стати засобом розвитку та навчання. Жанр покрокових стратегій вимагає від гравця стратегічного мислення, планування та прийняття важливих рішень.

Гра спрямована на забезпечення захоплюючого та цікавого геймплею, в якому гравці можуть розвивати своє стратегічне мислення та приймати важливі рішення.

Дипломна робота передбачає вивчення методів розробки відеоігрових програмних продуктів, аналіз науково-технічної літератури з питань, пов'язаних з програмним забезпеченням для розробки відеоігор, а також використання офіційної документації Unity, Microsoft Visual Studio та мови програмування C#. Велика увага приділена розробці логіки гри, реалізації графічного інтерфейсу та тестуванню.

Результати даної роботи сприятимуть подальшому розвитку та удосконаленню жанру покрокових стратегій у сфері комп'ютерних ігор. Розроблена гра може бути використана як засіб ненадовго відволіктися та витратити час на захоплюючий та нескладний процес геймплею, сприяючи розвитку стратегічного мислення.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1.Визначення покрокових стратегій в комп'ютерних іграх

Покрокові стратегії в комп'ютерних іграх є піджанром стратегічних ігор, де гравець виконує свої дії у послідовність окремих кроків або ходів. Основною особливістю покрокових стратегій є те, що гра проходить не в режимі реального часу, а за покроковими інтервалами, де кожен гравець робить свій хід, а потім гра переходить до наступного ходу. Це надає гравцеві можливість уважно обдумати кожен свій крок, розглянути можливі варіанти та наслідки своїх рішень.

Покрокові стратегії широко застосовуються у різних жанрах комп'ютерних ігор, включаючи військові стратегії, фантастичні світи, науково-фантастичні симулятори, ігри з елементами будівництва та економіки тощо. Такі ігри часто вимагають глибокого стратегічного мислення, планування та розробки довгострокових стратегій для досягнення поставлених цілей. Вони створюють можливість для гравців керувати власними імперіями, арміями, населенням або виконувати інші стратегічні завдання в умовах віртуального світу.

Основними елементами покрокових стратегій є карта або територія, на якій розгортається гра, та різні одиниці, що представляють гравця або його сили. Гра часто передбачає управління ресурсами, дипломатичні відносини, воєнні конфлікти, будівництво та розвиток, торгівлю та інші аспекти, які впливають на розвиток ігрового процесу. Геймплей покрокових стратегій може бути як вирішальним, так і складним, залежно від конкретної гри і її правил.

З розвитком технологій комп'ютерних ігор та доступністю інструментів для розробки, покрокові стратегії стали все популярнішими серед гравців. Їх варіативність та можливості привертають увагу широкої аудиторії, незалежно від вікової категорії та ігрового досвіду. Завдяки ігровим рушіям, таким як Unity, розробники отримали зручні та потужні інструменти для створення покрокових стратегій з високою якістю графіки, звуковим супроводом та іншими елементами, що роблять гру більш захоплюючою та реалістичною.

Крім того, покрокові стратегії відкривають широкі можливості для творчості та інновацій у галузі розробки ігор. Розробники можуть експериментувати з новими механіками, системами управління та геймплеєм, створюючи унікальні та цікаві ігрові вироби. Це сприяє розвитку індустрії та забезпечує постійний потік нових ігор для задоволення потреб геймерів.

Висока популярність жанру покрокові стратегії свідчить про постійний попит на ігри, які пропонують гравцям глибокий стратегічний досвід та емоційне задоволення від планування, розвитку та досягнення ігрових цілей. Велика кількість успішних покрокових стратегій на ринку свідчить про їх комерційний потенціал та здатність привернути велику кількість гравців.

Розробники намагаються вдосконалювати геймплей, графічну складову, штучний інтелект та інші аспекти гри для надання неперевершеного ігрового досвіду. Ігри в цьому жанрі можуть включати різноманітні сценарії, режими гри, мультиплеєрні можливості та інші функції, що збагачують ігровий процес та забезпечують довготривалу захоплюючу гру. Аналіз таких аспектів розвитку та успіху покрокових стратегій має важливе значення при розробці власної гри в жанрі покрокової стратегії з використанням рушія Unity та мови програмування C#.

1.2.Історія та еволюція жанру покрокових стратегій

Жанр покрокових стратегій має багату історію, яка починається ще в ранній епосі комп'ютерних ігор. Перші покрокові стратегії з'явилися у 1970-х роках, коли комп'ютери стали доступні широкій публіці. Одним з найвідоміших прикладів є гра "Civilization" від Sid Meier, яка вийшла у 1991 році і заслужила велику популярність. "Civilization" встановила стандарти для жанру покрокових стратегій, пропонуючи глибокий стратегічний геймплей, де гравець керує розвитком своєї цивілізації від примітивних часів до сучасності.

З плином часу покрокові стратегії почали розвиватися та еволюціонувати. Розробники ставили нові завдання та виклики перед гравцями, розширювали можливості гри та вдосконалювали геймплей. З'явилися нові ігри, які надали свої

унікальні особливості та ідеї до жанру покрокових стратегій. Наприклад, серія "Total War" від Creative Assembly поєднала елементи глобальної стратегії з реалістичними битвами в режимі реального часу. "XCOM: Enemy Unknown" від Firaxis Games зробив акцент на тактичних боях та менеджменті бази. Ці ігри додали свіжого подиху в жанр покрокових стратегій та зарекомендували себе як успішні та захоплюючі проекти.

З появою сучасних технологій та розвитком індустрії комп'ютерних ігор, покрокові стратегії продовжують еволюціонувати і пристосовуватися до змінних вимог гравців. Однією з найважливіших тенденцій у розвитку жанру є збільшення уваги до графіки, звукового оформлення та інших аспектів візуального досвіду. Сучасні покрокові стратегії надають деталізовані та реалістичні графічні образи, використовують високоякісний звуковий супровід, що дозволяє гравцям ще більше погрузитися у гру і відчути її атмосферу.

Також варто зазначити, що з'явлення інтернету та мультиплеєрних можливостей суттєво змінили підхід до покрокових стратегій. Гравці тепер можуть змагатися або співпрацювати з іншими гравцями з усього світу, створюючи різні онлайн-соціальні взаємодії. Це розширило можливості гри та надало їй більш динамічний та соціальний характер. Такий розвиток дозволяє гравцям насолоджуватися покроковими стратегіями не лише як індивідуальним досвідом, але й як спільною грою з друзями або випробуванням своїх стратегічних навичок у змаганнях з суперниками з усього світу.

Зростання популярності мобільних платформ також вплинуло на розвиток жанру покрокових стратегій. Сучасні смартфони та планшети забезпечують мобільність та доступність для гравців, дозволяючи їм насолоджуватися покроковими стратегіями в дорозі, у будь-якому місці та в будь-який час. Багато покрокових стратегій випущені на мобільних платформах, пропонують спеціально адаптований інтерфейс та геймплей для екранів сенсорних пристроїв. Це сприяє збільшенню аудиторії та популярності жанру, оскільки він стає доступним для більш широкої групи гравців.

Загалом, історія та еволюція жанру покрокових стратегій свідчить про його

стійкість та здатність адаптуватися до змін у геймінговій індустрії. Розвиток технологій, збільшення уваги до візуального та звукового досвіду, поява онлайн-ігрових можливостей та мобільний геймінг вносять свої внески у жанр, розширюючи можливості та привертаючи нову аудиторію. Це свідчить про постійний розвиток та актуальність покрокових стратегій як жанру, який продовжує викликати інтерес гравців усього світу.

1.3.Основні елементи та механіки покрокових стратегій

Покрокові стратегії мають свої основні елементи та механіки, які визначають їхню унікальність та геймплей. Один з ключових елементів - це покрокова система, де гравець має можливість здійснювати свої дії в окремих ходах або кроках. Кожен хід може включати такі дії, як переміщення одиниць, будівництво споруд, дослідження нових технологій, ведення бою та багато іншого. Гравець має враховувати різні фактори та стратегічні розрахунки при прийнятті рішень у своїх ходах.

Інший важливий елемент покрокових стратегій - це управління ресурсами. Гравець повинен ефективно керувати різними ресурсами, такими як гроші, сировина, енергія, населення та інші, для забезпечення розвитку своєї цивілізації або фракції. Ефективне використання ресурсів дозволяє гравцю розширювати свої можливості, вдосконалювати війська, будувати нові споруди та забезпечувати стійкий економічний розвиток. Управління ресурсами є ключовим аспектом стратегічного планування та прийняття рішень у покрокових стратегіях.

Крім того, покрокові стратегії часто мають систему розвитку та прогресії. Гравець може покращувати свої одиниці, персонажів або цивілізацію, набуваючи нові навички, технології або ресурси. Це дозволяє гравцю створювати більш потужні армії, розблоковувати нові можливості та стратегічні варіанти, а також забезпечувати більш глибокий та варіативний геймплей. Прогресія відображає досягнення гравця та його зростання протягом гри.

Комунікація та дипломатія є ще однією важливою складовою покрокових

стратегій. Гравці можуть взаємодіяти між собою шляхом укладання союзів, проведення перемовин, встановлення торгових відносин або навіть ведення війни. Успішне використання дипломатичних вмінь може допомогти гравцю отримати підтримку інших фракцій, уникнути конфліктів або навіть вплинути на вирішення глобальних подій у грі. Комунікація та дипломатія вносять елемент соціальної взаємодії та стратегічного планування в покрокові стратегії, розширюючи геймплей та створюючи нові можливості для гравця.

Додатковою важливою механікою покрокових стратегій є бойова система. Гравець може вести битви з іншими гравцями або комп'ютерними противниками, використовуючи свої війська та стратегічні здібності. Бойова система може мати різноманітні елементи, такі як шахові або тактичні ходи, розміщення військ на полі бою, використання спеціальних або магічних здібностей та багато іншого. Гравець повинен ретельно розраховувати свої дії, враховуючи силу та слабкі сторони своїх військ та противника, що впливають на вихід битви.

Крім того, покрокові стратегії часто мають складні системи керування та управління. Гравець має можливість керувати не лише військами, але й цивілізацією, економікою, дипломатією, наукою та іншими аспектами гри. Управління може включати прийняття стратегічних рішень, виділення ресурсів на певні цілі, розподіл завдань між одиницями та управління процесами розвитку. Складність системи керування може варіюватися в залежності від глибини та складності гри, і гравець повинен володіти хорошим розумінням механік та вміти ефективно керувати різними аспектами гри для досягнення успіху.

1.4.Популярність та комерційний успіх покрокових стратегій

Покрокові стратегії є одним з найвпізнаваніших та найуспішніших жанрів в ігровій індустрії. Вони здобули широку популярність серед гравців та отримали визнання за їхню глибину, стратегічну складність та захоплюючий геймплей. Багато покрокових стратегій стали легендарними, завдяки якості реалізації та інноваційним ідеям, що вони пропонують.

Відомі покрокові стратегії, такі як Civilization, Total War, XCOM та Heroes of Might and Magic, зарекомендували себе як справжні класики жанру та отримали велику кількість прихильників по всьому світу. Ці ігри вирізняються не тільки захоплюючим геймплеєм, але й великою кількістю стратегічних варіантів, складністю прийняття рішень та глибиною гри. Багато з них також мають велику кількість доповнень та продовжень, які додають ще більше можливостей та варіативності до геймплею, збільшуючи тривалість ігрового досвіду для гравців. Завдяки своєму комерційному успіху та відданій фанатській базі, покрокові стратегії часто отримують подальшу підтримку та розвиток від розробників, що сприяє подальшому росту популярності цього жанру.

Покрокові стратегії також знаходять своє місце в електронному спорті, отримуючи визнання та статус конкурентоспроможного ігрового жанру. Великі турніри та чемпіонати з покрокових стратегій збирають професійних гравців з усього світу, які змагаються за грошові призи та звання найкращих у своєму жанрі. Це свідчить про значимість та популярність цих ігор серед гравців і сприяє подальшому просуванню та розвитку покрокових стратегій.

Також варто зазначити, що покрокові стратегії мають потенціал для успішного комерційного використання. Завдяки великому числу шанувальників жанру та здатності до довготривалої гри, ці ігри приваблюють увагу видавців та розробників. Реліз успішної покрокової стратегії може мати значний комерційний успіх, забезпечуючи прибуток через продажі гри та супутніх товарів, таких як доповнення, колекційні видання, фігурки персонажів та інше. Крім того, покрокові стратегії можуть бути успішно випущені на різних платформах, включаючи персональні комп'ютери, консолі та мобільні пристрої, що розширює аудиторію гравців та можливості для комерційного успіху.

На сучасному ринку ігор покрокові стратегії займають важливе місце, пропонуючи гравцям глибокий інтелектуальний виклик та непередбачувані ситуації. Ці ігри зазвичай мають довгу тривалість геймплею, що привертає гравців, які насолоджуються стратегічним мисленням та плануванням на довгу перспективу. Крім того, покрокові стратегії можуть надати відчуття повноцінного

керівництва та контролю над грою, що приваблює гравців, які насолоджуються процесом створення і розвитку своїх віртуальних цивілізацій, армій чи імперій.

Крім того, покрокові стратегії мають потужний потенціал для розвитку та інновацій. Розробники постійно працюють над вдосконаленням геймплею, впроваджуючи нові механіки, функції та ідеї, щоб зробити гру більш цікавою та захоплюючою для гравців. Видавці активно підтримують покрокові стратегії шляхом випуску доповнень та оновлень, що додає ще більше контенту та варіацій до гри. Такий постійний розвиток сприяє залученню нових гравців та збереженню інтересу до покрокових стратегій на протязі тривалого часу.

1.5.Аналіз існуючих покрокових стратегій на ринку

На сучасному ринку існує велика кількість покрокових стратегій, які пропонують різноманітні геймплейні варіанти та тематики. Один з найвідоміших представників цього жанру є серія Civilization, розроблена компанією Firaxis Games. Гра пропонує гравцям можливість керувати цивілізацією, вести її від початкових етапів розвитку до сучасності та навіть в майбутнє. Гравці змагаються з іншими цивілізаціями за володарство над світом, використовуючи стратегічне мислення, дипломатію та військову тактику. Велика кількість варіантів розвитку, широкий вибір громадських будівель, технологій та лідерів роблять гру надзвичайно глибокою та різноманітною.

Інша популярна покрокова стратегія - серія Total War від Creative Assembly. Ці ігри комбінують глобальну стратегію з воєнною тактикою в реальному часі. Гравці мають можливість керувати великими арміями та військовими формуваннями на полі бою, використовуючи різноманітні стратегічні прийоми, тактику та уміння. Крім того, гра пропонує глобальну карту світу, на якій гравці можуть керувати своїми територіями, розвивати економіку та дипломатичні відносини з іншими фракціями. Відмінна деталізація, реалістична графіка та епічні битви роблять гру вельми привабливою для шанувальників стратегій та історичних симуляцій.

Ще однією визначною покроковою стратегією є серія XCOM, розроблена компанією Firaxis Games. Гра пропонує гравцям можливість очолити організацію, що бореться з прибульцями, використовуючи команду солдатів із різними вміннями та військовими технологіями. Гравці вступають в пошуках ресурсів, досліджують нові технології та ведуть битви в пошуках зброї та інформації. Вибір кожного кроку має велике значення, оскільки гра базується на пошарових боях та стратегічному плануванні. Різноманітність ворогів, несподівані події та постійний ризик втрати солдатів роблять гру надзвичайно напруженою та захоплюючою.

Крім того, варто згадати про серію Heroes of Might and Magic, що поєднує покрокову стратегію з елементами фентезі та RPG. Гравці керують героями, які провадять свої армії через фантастичні світи, здобуваючи ресурси, розвиваючи міста та ведучи битви з іншими героями. Ці ігри відомі своїми великими картами, великим розмаїттям магічних сил та різноманітними варіантами стратегічного планування. Гравці можуть вибирати різні шляхи розвитку, використовуючи різні стратегічні підходи для перемоги над супротивниками.

Monopoly - це класична настільна гра, в якій гравці купують, продують та обмінюють нерухомість з метою отримання більшої кількості грошей та встановлення монополії на ринку. Гра має чіткі правила та механіки, які дозволяють гравцям планувати свої наступні кроки та розраховувати свої дії.

Monopoly підходить для вирішення задачі розробки покрокової стратегії завдяки своїй добре продуманій ігровій механіці, яка дозволяє гравцям планувати свої наступні кроки та дії. Гра має чіткі правила, що робить її простою та зрозумілою для гравців, і в той же час має достатньо глибини, щоб гравці могли розвивати свої вміння та стратегії.

Однією з переваг Monopoly є те, що гра має велику кількість різноманітних правил та механік, що дозволяє гравцям налаштувати гру під свої потреби та вподобання.

Однак, є деякі недоліки у грі Monopoly, які потрібно враховувати при розробці власної гри в жанрі покрокових стратегій. По-перше, досить складна механіка гри, яка може відлякувати новачків від гри. По-друге, гра має досить довгу

тривалість, що може знизити зацікавленість користувачів. По-третє, гра не має досить складний інтерфейс, що може призвести до швидкої втрати цікавості до гри. Приклад інтерфейсу продемонстровано на рисунку 1.1.

Рисунок 1.1 – Приклад інтерфейсу гри Monopoly

Pummel Party є іншою покроковою стратегією, яка більше орієнтована на битви з іншими гравцями, ніж на економіку або ресурси. Гра містить багато випадкових подій, які можуть суттєво змінити хід гри і зробити її більш непередбачуваною. У грі є багато різноманітних видів зброї, пасток та інших предметів, які гравці можуть використовувати для атаки один на одного.

Одна з головних переваг Pummel Party - це його мультиплеєрний режим. Гравці можуть грати як локально, так і онлайн, що дозволяє збиратися з друзями та битися один з одним у віртуальному світі. Крім того, гра має цікаву та яскраву графіку, що робить її більш привабливою для молодшої аудиторії. Приклад графіки продемонстровано на рисунку 1.2.

Однак, деякі гравці відзначають, що гра може стати досить одностайною через відсутність різноманітності у режимах гри та малий вибір карт. Крім того, гра

може вимагати дуже багато часу для того, щоб успішно пройти її до кінця, що може бути важко для тих, хто має обмежений час на гру.

Рисунок 1.2 – Приклад графіки гри Pummel Party

"For the King" - це ще один добре відомий ігровий продукт в жанрі покрокової стратегії, який був розроблений в 2018 році. Гра містить елементи ролевої гри, а також детально розроблену систему боїв.

У грі "For the King" гравці зможуть створювати свої команди та вирушати у подорож через віртуальний світ з метою боротьби зі злом та рятування цього світу від загибелі. Ця гра має значно складнішу механіку, ніж Monopoly або Pummel Party, і вимагає від гравців більше стратегічного мислення.

Головними перевагами гри є багатофункціональність, хороша графіка та звукове супроводження. Проте, серед недоліків можна виділити високий рівень складності та вимог до досвіду гравця, що може здатися надто важким для новачків. Інтерфейс гри є складним для розуміння. Приклад інтерфейсу продемонстровано на рисунку 3.3. Також, багато гравців скаржаться на нестабільну роботу гри та наявність деяких багів та глюків. Інтерфейс гри є складним для розуміння

Рисунок 1.3 – Приклад інтерфейсу гри For the King

Зведені результати аналізу характеристик розглянутих додатків наведено у таблиці 1.1.

Таблиця 1.1 – Зведені результати аналізу покрокових стратегій

Критерії порівняння	Monopoly	Pummel Party	For the King
Системні вимоги	8 GB RAM; Intel i7-6700; NVidia GeForce GTX 960	3 GB RAM; Intel Core i5; GeForce 8800 GT	4 GB RAM; Intel Core i5; GeForce GTX 750 Ti
Складність геймплею	Середня	Висока	Висока
Тривалість ігрової партії	1 – 6 годин	1 – 3 години	1 – 6 годин
Складність сприйняття ігрового інтерфейсу	Висока	Середня	Висока
Механіка купівлі власності	Наявна	Відсутня	Відсутня

Механіка нанесення шкоди противнику	Відсутня	Наявна	Наявна
Стабільність частоти кадрів	Від 60 до 35 к/с	Від 60 до 30 к/с	Від 60 до 20 к/с
Наявність ігрових дефектів	Візуальний дефект при пересуванні гравця	Дефект управління персонажем	Затримка передачі ходу від 3 до 10 секунд

1.6.Виклики та тенденції у розробці покрокових стратегій

У розробці покрокових стратегій існують деякі виклики та тенденції, які впливають на процес створення ігор цього жанру. Один з головних викликів полягає у збалансованості гри. Покрокові стратегії вимагають уважного налаштування параметрів гри, щоб забезпечити різноманітні можливості та стратегічні варіанти. Важливо забезпечити, щоб гра була не надто простою або складною, щоб зберегти захоплюючий ігровий процес та інтелектуальний виклик для гравців.

Іншим викликом є забезпечення реалістичності та глибини гри. Гравці прагнуть до емоційного занурення у віртуальний світ покрокової стратегії. Тому важливо розробляти складні ігрові механіки, зробити їх досить реалістичними та варіативними. Додавання елементів, таких як різноманітність умов та територій, можливості стратегічного планування та тактичного розрахунку, може підвищити глибину гри та відчуття реалізму для гравців.

Однією з тенденцій у розробці покрокових стратегій є збільшення використання штучного інтелекту. Розробники стараються вдосконалити алгоритми поведінки ворогів та союзників у грі, щоб забезпечити більш реалістичну гру. Штучний інтелект може аналізувати ситуації, приймати стратегічні рішення, реагувати на дії гравця та виконувати складні алгоритми. Це додає грі глибину та розширює можливості гравців у виборі тактик та стратегій.

Ще однією тенденцією є розширення онлайн-функцій та багатокористувацького режиму. Все більше покрокових стратегій пропонують можливість грати з іншими гравцями онлайн, об'єднуватися в команди або змагатися один з одним. Це відкриває нові можливості для взаємодії та соціального виміру гри, дозволяючи гравцям змагатися з іншими стратегами з усього світу. Онлайн-режим також забезпечує можливість оновлення та додавання нового контенту до гри, що розширює її тривалість та цікавість для гравців.

Не можна також не зазначити зростання популярності мобільних платформ серед гравців. Все більше покрокових стратегій стають доступними на мобільних пристроях, що дозволяє гравцям грати в них в будь-який час і в будь-якому місці. Мобільні платформи пропонують нові можливості для геймплею, використання сенсорних елементів управління та використання геолокації для розширення взаємодії з оточенням. Це розширює аудиторію гравців і забезпечує більш широку доступність покрокових стратегій для людей усіх вікових груп та з різними ігровими вподобаннями.

Ще однією важливою тенденцією у розробці покрокових стратегій є зосередженість на нелінійності та свободі вибору. Сучасні гравці все більше цінують можливість впливати на хід гри та розвивати власну стратегію відповідно до своїх вподобань. Тому розробники прагнуть створювати ігри з великою кількістю альтернативних шляхів розвитку подій, різноманітними варіантами вирішення завдань та необмеженими можливостями для гравця. Це надає більшу свободу та гнучкість у геймплею, роблячи гру більш цікавою та повторюваною.

Додатковою тенденцією є поєднання покрокових стратегій з іншими жанрами та елементами геймплею. Розробники все частіше експериментують з поєднанням покрокових стратегій з елементами рольових ігор (RPG), екшену або навіть симуляторів. Це дає можливість створити унікальний гібридний геймплей, який поєднує різноманітні механіки та задовольняє потреби різних груп гравців. Такі поєднання забезпечують нові враження та можливості для творчості у розробці покрокових стратегій та привертають більше уваги гравців.

2. ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Огляд ігрового рушія Unity

Unity - це ігровий рушій та інтегроване середовище розробки (IDE), яке широко використовується для створення різноманітних видів комп'ютерних ігор. Розроблений компанією Unity Technologies, цей рушій надає потужні інструменти та ресурси для створення як 2D, так і 3D ігор з різними жанрами та механіками. Одна з основних переваг Unity полягає в його доступності та придатності як для початківців, так і для досвідчених розробників.

2.1.1 Unity та його особливості

Однією з особливостей Unity є його крос-платформеність. Це означає, що ігри можуть бути розгорнуті на різних платформах, таких як ПК, консолі, мобільні пристрої та веб. Це забезпечує широку аудиторію та можливості для комерційного успіху, оскільки гра може бути доступна для гравців на різних пристроях. Unity також підтримує різні операційні системи, такі як Windows, macOS і Linux, що робить його універсальним інструментом для розробки.

Іншою важливою особливістю Unity є його широкий набір функцій та інструментів, спрямованих на полегшення процесу розробки ігор. Unity надає розробникам доступ до готових компонентів, скриптів та ресурсів, що дозволяє прискорити створення ігрових об'єктів, фізичної моделі, штучного інтелекту та інших елементів гри. Крім того, Unity має велику спільноту розробників, яка активно ділиться знаннями, документацією та ресурсами, що сприяє підтримці та вирішенню проблем під час розробки.

Ще однією вагомою перевагою Unity є його вбудована підтримка різних форматів файлів, таких як графічні файли, аудіо, відео тощо. Це дозволяє розробникам легко імпортувати та використовувати різноманітні медіа-ресурси в своїх іграх. Unity також підтримує різні мови програмування, зокрема C#, що є потужним і популярним інструментом для розробки ігор. Ця мова програмування

дозволяє розробникам створювати скрипти, контролювати поведінку об'єктів, реалізовувати ігрові логіку та багато іншого. За допомогою Unity і мови програмування C# розробники мають гнучкість та потужність для створення різноманітних елементів та механік покрокових стратегій.

Unity також пропонує багато інструментів для візуального редагування та розробки ігрових сцен. За допомогою редактора Unity, розробники можуть створювати та налаштовувати об'єкти, розміщати їх у сценах, налаштовувати фізику, світло, камери та інші параметри гри. Це дозволяє розробникам швидко експериментувати зі складовими гри, переглядати результати в реальному часі та вносити необхідні зміни. Редактор Unity має інтуїтивний інтерфейс та потужні функції, що допомагають зосередитися на самому процесі розробки, забезпечуючи швидкий та ефективний шлях до створення покрокової стратегії.

Ще однією особливістю Unity є його можливості для розробки мобільних ігор. Unity підтримує розробку ігор для різних мобільних платформ, таких як iOS та Android. Розробники можуть використовувати спеціальні інструменти та налаштування, щоб оптимізувати гру під мобільні пристрої, забезпечуючи оптимальну продуктивність та відповідну якість зображення. Це дає змогу створювати покрокові стратегії, які можна грати на смартфонах та планшетах, розширюючи аудиторію та забезпечуючи більш широкий охоплення ринку. Багато успішних мобільних ігор жанру покрокових стратегій були створені з використанням Unity, демонструючи потужність та ефективність цього рушія у розробці ігор для мобільних платформ.

2.1.2 Можливості рушія Unity для розробки покрокових стратегій

Один з основних аспектів, який робить Unity привабливим для розробки покрокових стратегій, - це його потужна система управління об'єктами та компонентами. У Unity об'єкти в грі представляються як сутності, які мають набір компонентів. Кожен компонент відповідає за конкретну функціональність об'єкта, таку як рух, графіка, колізія, штучний інтелект тощо. Це дозволяє розробникам створювати складні системи, в яких об'єкти взаємодіють між собою та зовнішнім

середовищем. Для покрокових стратегій, де важлива координація багатьох об'єктів та їх взаємодія, така система управління об'єктами є незамінною.

Unity також надає широкий набір інструментів та ресурсів, які полегшують розробку покрокових стратегій. Вбудовані функції фізики, анімації, звуку та графіки допомагають створювати реалістичні та живі ігрові світи. Unity також має розширену підтримку штучного інтелекту, що дозволяє розробникам створювати складні стратегічні алгоритми та поведінку для ігрових персонажів. Крім того, Unity має широкий вибір розширень та активів, які можна використовувати для розробки покрокових стратегій, такі як набори графічних ефектів, анімаційні редактори, системи розміщення об'єктів та багато іншого. Все це робить Unity потужним інструментом для розробки покрокових стратегій, надаючи розробникам необхідні можливості для створення ігор з багатими функціональним.

Однією з ключових переваг Unity для розробки покрокових стратегій є його кросплатформова підтримка. Unity дозволяє розробляти ігри, які працюють на різних платформах, включаючи ПК, мобільні пристрої, консолі, віртуальну реальність та інші. Це дозволяє розробникам зробити свої покрокові стратегії доступними для широкої аудиторії гравців на різних пристроях. Крім того, Unity надає зручні інструменти для розгортання та публікації гри на різних платформах, спрощуючи процес релізу та поширення гри.

Ще однією важливою особливістю Unity є його велика спільнота розробників. Unity має активну спільноту, яка складається з розробників, форумів, ресурсів та документації. Це надає розробникам доступ до багатої бази знань, досвіду та підтримки від інших фахівців. Розробники можуть задавати питання, отримувати поради та ділитися своїми досягненнями з іншими членами спільноти. Це сприяє взаємному навчанню та зростанню, а також робить розробку покрокових стратегій в Unity більш доступною і підтримуваною.

2.1.3 Переваги та обмеження використання Unity для розробки покрокових стратегій

Перевагами використання Unity для розробки покрокових стратегій є його

широкі можливості в графічному та звуковому відображенні. Unity надає потужні інструменти для створення високоякісних графічних ефектів, анімації персонажів, динамічного освітлення та інших візуальних елементів. Розробники можуть створювати живописні світи, деталізовані моделі та ефектні бойові сцени, що додає до естетичного враження від гри та підсилює іммерсивність гравця. Крім того, Unity підтримує різноманітні аудіофункції, дозволяючи додати реалістичні звуки, музику та голосові ефекти до гри.

Проте, використання Unity для розробки покрокових стратегій також має свої обмеження. Одним з них є обмежена підтримка масштабування. У покрокових стратегіях можуть бути великі світи з багатьма одночасно діючими одиницями, що може вимагати значних обчислювальних ресурсів. Unity може стикатися з обмеженнями процесора та оперативної пам'яті, особливо на мобільних пристроях. Розробники повинні бути обережними при плануванні та оптимізації гри, щоб уникнути проблем з продуктивністю та завантаженістю системи. Крім того, Unity не надає вбудованих інструментів для автоматизованої роботи зі штучним інтелектом (ШІ), що може бути необхідним для складних стратегічних алгоритмів та поведінки ігрових персонажів. Розробники повинні використовувати додаткові рішення або самостійно розробляти компоненти ШІ.

2.2 Огляд мови програмування C#

Мова програмування C# є однією з найпопулярніших мов для розробки ігрових додатків у середовищі Unity. Вона має синтаксис, схожий на мови C і C++, що робить її зрозумілою для багатьох розробників. C# володіє потужними можливостями об'єктно-орієнтованого програмування, дозволяючи створювати класи, об'єкти, успадкування, поліморфізм та інші принципи ООП. Це робить його ідеальним вибором для розробки складних систем у покрокових стратегіях, де важливо використовувати об'єктно-орієнтований підхід для моделювання геймплею, юнітів та інших компонентів гри.

Однією з важливих переваг мови C# є його інтеграція з інструментами

розробки Unity. Unity використовує спеціальну версію C#, відому як Unity Scripting API, яка надає доступ до всіх функцій і функціональностей рушія. Це включає роботу зі сценами, об'єктами, фізикою, анімацією, звуком та багатьма іншими аспектами гри. Крім того, C# підтримує багатопотоковість, що дозволяє розробникам ефективно використовувати паралельні процеси та оптимізувати продуктивність гри. Усі ці особливості роблять мову C# потужним інструментом для розробки покрокових стратегій в середовищі Unity.

2.2.1 Основні характеристики мови програмування C#

Мова програмування C# має ряд основних характеристик, які роблять її потужним інструментом для розробки покрокових стратегій в середовищі Unity. Перш за все, C# є типізованою мовою, що означає, що кожна змінна має визначений тип даних. Це сприяє визначенню чітких інтерфейсів, дозволяє проводити перевірку типів на етапі компіляції та зменшує кількість помилок у програмі. Крім того, C# підтримує явну та неявну конвертацію типів, що дозволяє зручно працювати з різними типами даних та забезпечує гнучкість в програмуванні.

Другою важливою характеристикою мови C# є його підтримка об'єктно-орієнтованого програмування (ООП). Це означає, що ви можете створювати класи, об'єкти, успадковувати властивості та методи, використовувати поліморфізм і інші принципи ООП для організації та моделювання складних систем. ООП дозволяє розбити програму на логічні модулі, що спрощує розробку та підтримку коду. Крім того, C# підтримує інкапсуляцію, що дозволяє приховувати деякі деталі реалізації класу та забезпечує контроль доступу до його членів.

Крім того, мова програмування C# має потужну систему керування пам'яттю, що забезпечує автоматичне управління пам'яттю та звільнення ресурсів, що більше не використовуються. Це полегшує завдання розробників, оскільки вони можуть уникнути багатьох проблем, пов'язаних з керуванням пам'яттю, таких як витоки пам'яті або неявне звільнення ресурсів.

Іншою важливою характеристикою мови C# є наявність багатофункціональної базової бібліотеки класів .NET Framework. Ця бібліотека

надає розробникам широкий спектр готових рішень і функцій, що значно спрощує розробку покрокових стратегій. Вона містить класи та методи для роботи зі звуком, графікою, мережами, базами даних, файлами і багатьма іншими аспектами розробки програмного забезпечення.

Однією з ключових особливостей мови C# є підтримка асинхронного програмування за допомогою ключових слов async/await. Ця функціональність дозволяє розробникам ефективно управляти асинхронними операціями, такими як мережеві виклики або завантаження ресурсів, без блокування основного потоку виконання програми. Асинхронне програмування сприяє покращенню продуктивності гри, забезпечує більш плавну і реактивну геймплей, а також покращує загальний досвід користувача.

2.2.2 Використання мови C# у розробці ігор на рушії Unity

Використання мови програмування C# у розробці ігор на рушії Unity є однією з найпоширеніших практик. C# виступає як основна мова програмування для розробки грального функціоналу, логіки гри та взаємодії з ігровим середовищем. Unity має потужну інтеграцію з C#, надаючи розробникам доступ до багатого набору класів, бібліотек та інструментів для створення ігрового досвіду.

З використанням C# у розробці Unity-ігор розробники можуть створювати складну логіку гри, здійснювати обробку вхідних даних, керувати поведінкою персонажів та рухом об'єктів. Мова C# пропонує зручний синтаксис і підтримку об'єктно-орієнтованого програмування, що дозволяє розробникам ефективно структурувати та організовувати код гри. Крім того, C# надає доступ до широкого спектру функцій, що допомагають реалізувати різноманітні механіки гри, включаючи фізичну симуляцію, штучний інтелект, мережеву взаємодію та багато іншого.

Unity забезпечує глибоку інтеграцію з мовою C#, що дозволяє розробникам легко використовувати всі можливості обох інструментів разом. У розробці ігор на Unity з використанням C# розробники можуть використовувати класи, методи, події, властивості та інші конструкції мови C#, щоб створювати розширений

функціонал гри. Крім того, Unity надає спеціальні компоненти та інструменти, які полегшують роботу з C# і прискорюють процес розробки. Наприклад, Unity Editor має інтегроване середовище розробки, яке дозволяє розробникам швидко та зручно писати та тестувати код C# прямо всередині ігрового проекту.

Використання мови C# у розробці ігор на Unity також дозволяє розробникам ефективно працювати з різними платформами. Unity підтримує кросплатформену розробку, що означає, що один і той же код C# може бути використаний для створення ігор на різних платформах, таких як ПК, консолі, мобільні пристрої та веб. Це робить мову C# універсальним інструментом для розробки покрокових стратегій на Unity, дозволяючи розробникам досягати більшої аудиторії гравців та забезпечувати досвід гри на різних платформах.

2.2.3 Переваги та обмеження використання мови C# у розробці покрокових стратегій

Переваги використання мови C# у розробці покрокових стратегій на рушії Unity включають його простоту та легкість в освоєнні. Мова C# має чистий синтаксис та велику кількість готових бібліотек, що полегшує розробку гри. Розробники можуть швидко створювати код, використовуючи готові шаблони та класи, що значно прискорює процес розробки.

Ще однією перевагою використання мови C# є її інтеграція з іншими інструментами та середовищами розробки. Наприклад, розробники можуть використовувати Microsoft Visual Studio, яке надає потужні інструменти для програмування на C#. Також Unity надає своє інтегроване середовище розробки, що сприяє зручному редагуванню коду C#. Ця інтеграція дозволяє розробникам ефективно використовувати всі можливості мови C# у розробці покрокових стратегій.

Однак, використання мови C# також має деякі обмеження. Наприклад, вона обмежена до екосистеми Unity, що може створити певні обмеження у виборі інших інструментів або платформ. Крім того, деякі розробники можуть відчувати обмеження в швидкодії, оскільки C# є мовою високого рівня.

3. ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до програмного продукту

Визначення основних функціональних та нефункціональних вимог до програми. Функціональні вимоги описують, які функції та можливості має мати програмний продукт. Це включає розробку інтерфейсу користувача, систему управління даними, алгоритми обробки інформації та інші специфічні функції, які потрібні для виконання поставлених завдань.

Нефункціональні вимоги визначають якість програмного продукту та умови його використання. Це можуть бути вимоги до продуктивності, надійності, безпеки, масштабованості, зручності використання та інші аспекти, які впливають на загальну якість програми. Нефункціональні вимоги також можуть включати вимоги до сумісності з іншими системами, архітектурних обмежень та технічних характеристик, що впливають на роботу програмного продукту.

Процес визначення вимог до програмного продукту вимагає ретельного аналізу потреб користувачів, розуміння контексту використання, а також урахування технічних обмежень та можливостей розробки. Вимоги до програмного продукту слід формулювати ясно, щоб забезпечити чітке розуміння та оцінку виконання цих вимог на різних етапах розробки.

3.1.1 Функціональні вимоги

Функціональні вимоги:

1. Головне меню:
   1.1.Гравець може почати гру.
   1.2.Гравець може вийти з гри.
2. Гра:
   2.1.Гра має двох гравців.
   2.2.Гравці можуть кидати кубик для визначення кількості кроків.
   2.3.Гравці рухаються по ігровому полю відповідно до кількості кроків.

2.4. Гравці можуть купувати поля та збирати монети.

2.5. Кожне поле має властивості, такі як сила атаки, вартість і кількість монет, що забирається в противника при атаці.

2.6. Гравці можуть атакувати один одного, застосовуючи різні типи атак своїх полів.

2.7. Гра закінчується, коли один з гравців втрачає всі свої життя.

2.8. Після закінчення гри гравці мають можливість перейти до головного меню.

3. Інтерфейс користувача:

3.1. Гра має графічний інтерфейс, який відображає ігрове поле, стан гравців, властивості поточного поля, кількість монет.

3.2. Гравці можуть взаємодіяти з грою за допомогою клавіш та елементів інтерфейсу.

4. Графіка та звук:

4.1. Гра має Low poly графіку, яка відображає ігрові об'єкти і ефекти.

4.2. Гра має звукові ефекти, що підсилюють геймплей і створюють належну атмосферу.

3.1.2 Нефункціональні вимоги

Нефункціональні вимоги:

1. Ефективність та продуктивність:

1.1. Гра повинна бути ефективною та продуктивною, забезпечуючи швидку відповідь на дії гравців. Це означає, що гра повинна мінімізувати затримки, зависання та переривання в процесі гри. Вона повинна забезпечувати плавний геймплей навіть при великій кількості графічних та обчислювальних обробок. Крім того, гра повинна ефективно використовувати ресурси комп'ютера, такі як процесор, оперативна пам'ять та графічна підсистема, для забезпечення оптимальної продуктивності.

1.2. Забезпечення стабільного оновлення частоти кадрів в межах 50 – 60к/с.

1.3. Гра повинна бути сумісна з операційною системою Windows. Крім того, гра повинна бути адаптованою до різних розмірів екранів та роздільних

здатностей, щоб забезпечити зручну та зрозумілу графіку для користувачів на різних пристроях. Також важливо, щоб гра мала низький рівень вимог до апаратного забезпечення, щоб вона могла працювати на різних пристроях, навіть з обмеженими ресурсами.

1.4. Мінімальні системні вимоги: RAM: 1 GB; Процесор: 2 GHz Dual Core CPU; Відеокарта: Intel HD Graphics 4000.

1.5. Рекомендовані системні вимоги: RAM: 2GB; Процесор: 2.5 GHz Dual Core CPU; Відеокарта: Geforce GTX 970.

2. Надійність:

2.1. Гра повинна бути надійною та стабільною, щоб уникнути непередбачуваних помилок або аварійного завершення гри.

3.2. Розробка діаграми класів

Діаграма класів є важливим інструментом в об'єктно-орієнтованому програмуванні, який дозволяє візуалізувати структуру класів, їх взаємозв'язки та основні атрибути і методи. Це допомагає краще зрозуміти внутрішню логіку програми та спроектувати його ефективно.

У діаграмі класів використано нотацію UML (Unified Modeling Language). UML є стандартною мовою моделювання, яка дозволяє описувати різні аспекти системи, включаючи її структуру, поведінку та взаємодію між об'єктами. Використання UML дозволить нам створити чітку та зрозумілу діаграму класів, яка буде використовуватися як вихідний пункт для подальшої розробки гри.

Для розробки діаграми класів, проаналізовано вимоги до гри. Згідно з аналізом, існує кілька основних класів. Один з цих класів - PlayerMover, відповідає за представлення гравця в грі. Він міститиме дані про гравця, такі як його ім'я, рівень, життя тощо. Клас PlayerMover також матиме методи для зміни стану гравця.

Ще одним важливим класом є Node, який представляє поле гри. Клас Node буде мати властивості, які визначатимуть його розмір та структуру. Він також буде містити методи для зміни стану поля, розташування об'єктів на полі, перевірка

наявності колізій тощо. Об'єкти класу Node будуть використовуватися для відображення гри на екрані та обробки взаємодії гравця з гральним полем.

Ще один клас, який присутній в коді гри, це клас "Route". Цей клас відповідає за створення та управління ігровим полем, яке складається з різних полів. Клас "Route" має методи для ініціалізації та розміщення полів на дошці, а також використовується для переміщення гравців по маршруту згідно результатів кидка кубика.

GameManager. Цей клас відповідатиме за управління грою в цілому. Він має методи для початку нової гри, завершення гри, завантаження стану гри, а також для взаємодії з іншими класами, наприклад, для передачі даних між класами.

Крім основних класів, в коді гри також присутні допоміжні класи та класи-контролери, які відповідають за різні аспекти гри, такі як управління інтерфейсом користувача, обробка дій гравців та збереження стану гри. Ці класи забезпечують взаємодію між основними класами та забезпечують правильну роботу гри.

У діаграмі класів можна використовувати різні види зв'язків для відображення взаємозв'язків між класами. Три основні види зв'язків, які часто використовуються, це агрегація, композиція і асоціація.

Агрегація вказує на те, що один клас є частиною іншого класу, але може існувати і самостійно. Зазвичай це відображається за допомогою ромбової стрілки від батьківського класу до дочірнього класу.

Композиція також вказує на те, що один клас є частиною іншого класу, але в цьому випадку підклас не може існувати без батьківського класу. Зазвичай це відображається за допомогою заповненого ромба з стрілкою від батьківського класу до дочірнього класу.

Асоціація вказує на зв'язок між двома або більше класами, коли один клас використовує інший клас або взаємодіє з ним. Це може бути зв'язок типу "має" або зв'язок типу "взаємодіє". В діаграмі класів асоціація зазвичай відображається за допомогою лінії , яка показує напрямок зв'язку.

Діаграму класів зображено на рисунку 3.1.

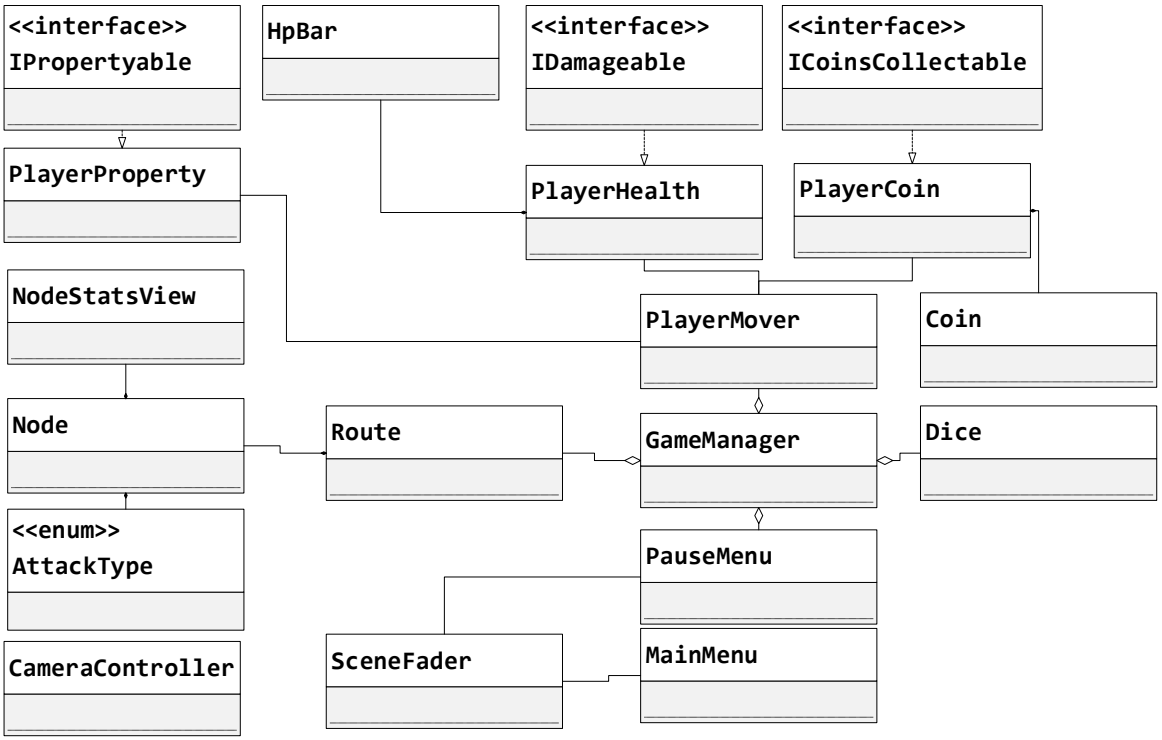


Рисунок 3.1 — Діаграма класів проекту

3.3.Розробка основних механік

Клас GameManager відповідає за керування основними механіками гри і є одним з ключових класів системи. Він відповідає за ініціалізацію гри, управління потоком гри, обробку взаємодії з гравцями та іншими компонентами системи.

Основною відповідальністю класу GameManager є встановлення початкового стану гри і забезпечення її коректної роботи протягом всієї сесії. Він контролює потік гри, забезпечує правильну послідовність ходів гравців, керує зміною станів гри і взаємодіє з різними компонентами системи для забезпечення їх спільної роботи.

У класі GameManager можуть бути методи, що дозволяють стартувати нову гру, завершувати поточну гру, переходити до наступного ходу, обробляти дії гравців і виконувати необхідні обчислення.

Загалом, клас GameManager є центральним елементом гри, який управляє всіма основними механіками і забезпечує їх правильне функціонування. Це

важливий клас, який вимагає уваги при розробці системи, оскільки він визначає основний потік гри і забезпечує її взаємодію з гравцями та іншими компонентами. Код класу «GameManager»:

```
public class GameManager : MonoBehaviour
{
    [SerializeField] private PlayerMover _player1;
    [SerializeField] private PlayerMover _player2;
    [SerializeField] private Dice _dice;

    [SerializeField] private Button _buttonRollDice;
    [SerializeField] private Button _buttonEndMove;

    [SerializeField] private Button _buttonBuyField;
    [SerializeField] private Button _buttonBlood;
    [SerializeField] private Button _buttonCoin;

    [SerializeField] private TMP_Text _textPlayerWin;
    [SerializeField] private Canvas _canvasTextPlayerWin;

    [SerializeField] private SceneFader _sceneFader;
    [SerializeField] private string _levelToLoad = "MainMenu";

    [SerializeField] private NodeStatsView _nodeStatsView;

    private PlayerMover _currentPlayerMover;

    private void Start()
    {
        _player1.GetComponent<PlayerHealth>().PlayerDead += EndGame;
        _player2.GetComponent<PlayerHealth>().PlayerDead += EndGame;

        _player1.PassedCircle += AddCoin;
        _player2.PassedCircle += AddCoin;

        _player1.FinishedMove += NodeViewUpdate;
        _player2.FinishedMove += NodeViewUpdate;

        _currentPlayerMover = _player1;
        _currentPlayerMover.LightOn();

        _buttonRollDice.onClick.AddListener(RollDice);
        _buttonEndMove.onClick.AddListener(ChangeCurrentPlayer);
        _buttonBuyField.onClick.AddListener(BuyField);
        _buttonBlood.onClick.AddListener(Blood);
        _buttonCoin.onClick.AddListener(Coin);

        _buttonEndMove.enabled = false;
        _canvasTextPlayerWin.enabled = false;
        _nodeStatsView.Hide();
    }

    private void BuyField()
    {
        if (!_player1.IsMoving && !_player2.IsMoving)
        {
            _currentPlayerMover.GetComponent<IPropertyable>().BuyField();
        }
    }
```

```
private void Blood()
{
    if (!_player1.IsMoving && !_player2.IsMoving)
    {
        _currentPlayerMover.GetComponent<IPropertyable>().SetBlood();
    }
}

private void Coin()
{
    if (!_player1.IsMoving && !_player2.IsMoving)
    {
        _currentPlayerMover.GetComponent<IPropertyable>().SetCoin();
    }
}

private void RollDice()
{
    var numberOnDice = _dice.RollDice();
    _buttonRollDice.enabled = true;
    if (!_player1.IsMoving && !_player2.IsMoving)
    {
        GoMoveCurrentPlayer(numberOnDice);
        _buttonRollDice.enabled = false;
        _buttonEndMove.enabled = true;
    }
}

private void GoMoveCurrentPlayer(int numberOnDice)
{
    _currentPlayerMover.GoMove(true, numberOnDice);
}


private void ChangeCurrentPlayer()
{
    _nodeStatsView.Hide();
    if (_currentPlayerMover == _player1)
    {
        _currentPlayerMover.LightOff();
        _currentPlayerMover = _player2;
        _currentPlayerMover.LightOn();
    }
    else if (_currentPlayerMover == _player2)
    {
        _currentPlayerMover.LightOff();
        _currentPlayerMover = _player1;
        _currentPlayerMover.LightOn();
    }

    _buttonRollDice.enabled = true;
    _buttonEndMove.enabled = false;
}

private void AddCoin()
{
    _currentPlayerMover.GetComponent<ICoinsCollectable>().AddCoins(50);
}

private void EndGame()
{
    _canvasTextPlayerWin.enabled = true;
```

```
        _textPlayerWin.text = $"Player {_currentPlayerMover.PlayerName} lost";

        Invoke(nameof(LoadNextScene), 5f);
    }

    private void NodeViewUpdate()
    {
        _nodeStatsView.ViewUpdate(_currentPlayerMover.CurrentNode);
    }

    private void LoadNextScene()
    {
        _sceneFader.FadeTo(_levelToLoad);
    }
}
```

Клас GameManager може використовувати інші класи для виконання певних завдань, наприклад, класи для обробки дій гравців, обчислення руху по полю гри, збереження стану гри тощо. Він може викликати методи цих класів у відповідний момент для забезпечення потрібного функціоналу.

Однією із основних механік являється механіка купівлі власності. Її алгоритм відображено на блок-схемі (рисунок 3.2).

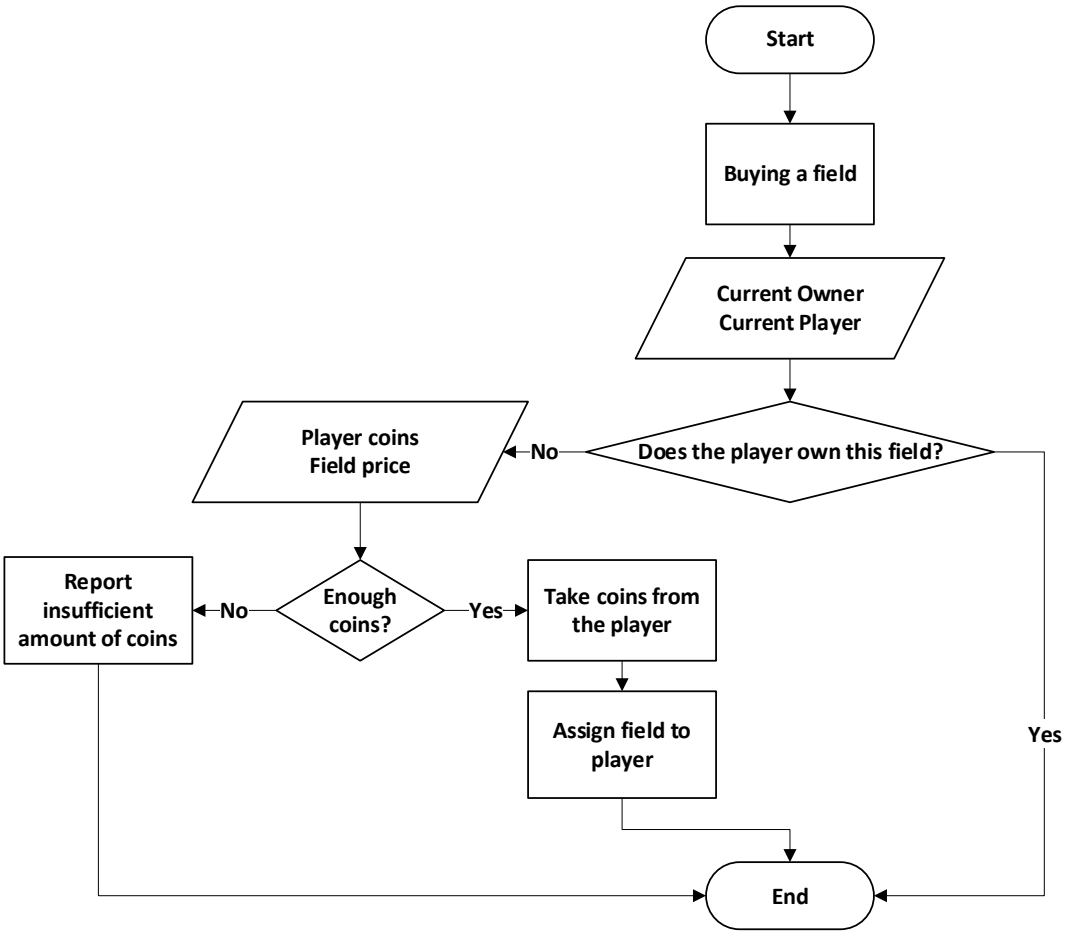


Рисунок 3.2 – Блок-схема алгоритму купівлі власності.

Також розроблена діаграма діяльності, що зображена на рисунку 3.3.

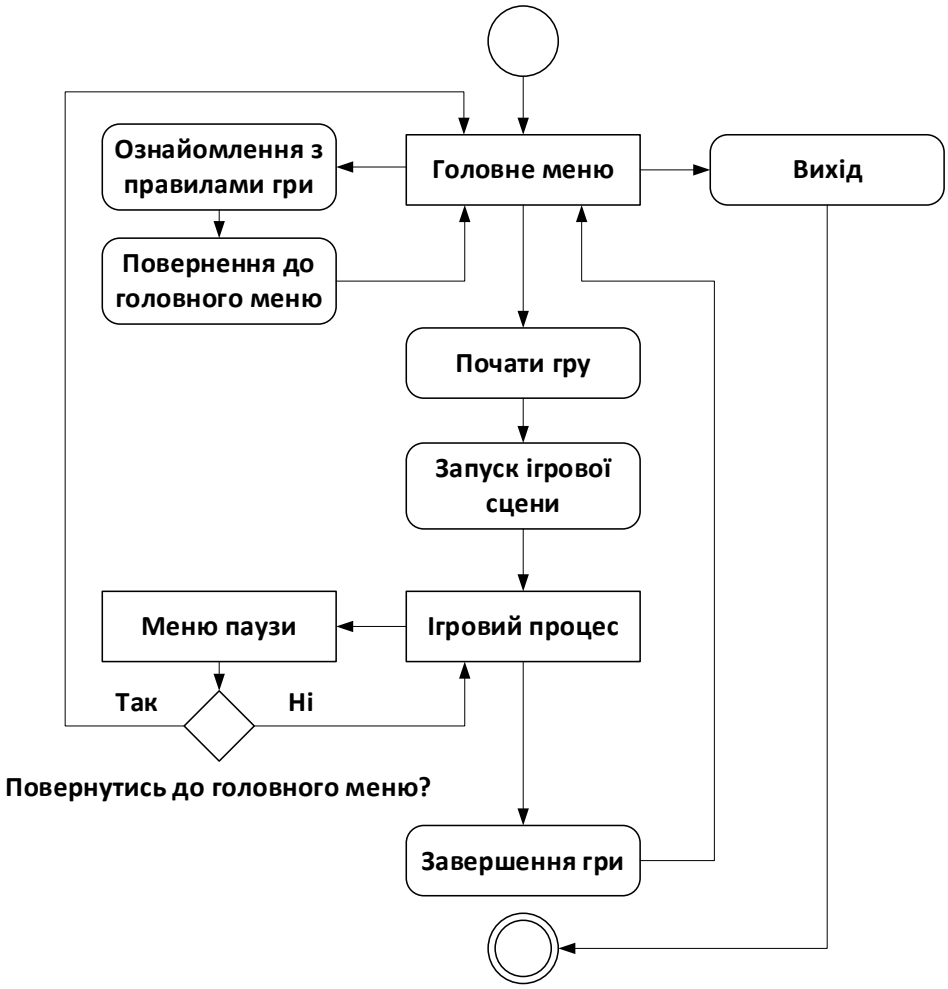


Рисунок 3.3 – Діаграма діяльності

3.4.Розробка графічного інтерфейсу

Графічний інтерфейс (GUI) відіграє важливу роль у сприйнятті гри гравцями, дозволяючи їм взаємодіяти з грою шляхом візуальних елементів, кнопок, меню та інших графічних компонентів.

Для реалізації графічного інтерфейсу використовуються відповідні інструменти та технології, такі як бібліотеки для створення графічних елементів, редактори інтерфейсу, мови програмування та інші інструменти, які надають зручні та потужні засоби для розробки GUI.

Гра розпочинається з головного меню яке зображено на рисунку 3.4.

Рисунок 3.4 — Головне меню

Ігровий процес відбувається на ігровому полі яке зображено на рисунку 3.5. Тут можна побачити елементи графічного інтерфейсу такі як:

1. Кількість запасу здоров’я гравців.
2. Кількість монет гравців.
3. Кнопки взаємодії з грою:
   3.1. Roll Dice – кидок кубика.
   3.2. Buy field – купівля поля.
   3.3. Coin – налаштування типу атаки поля, що забере монети в противника.
   3.4. Blood - налаштування типу атаки поля, що забере життєву силу в противника.
   3.5. End of move – завершення ходу.
4. Кнопка паузи, що відкриває вікно зображене на рисунку 3.6.

Рисунок 3.5 — Ігрове поле

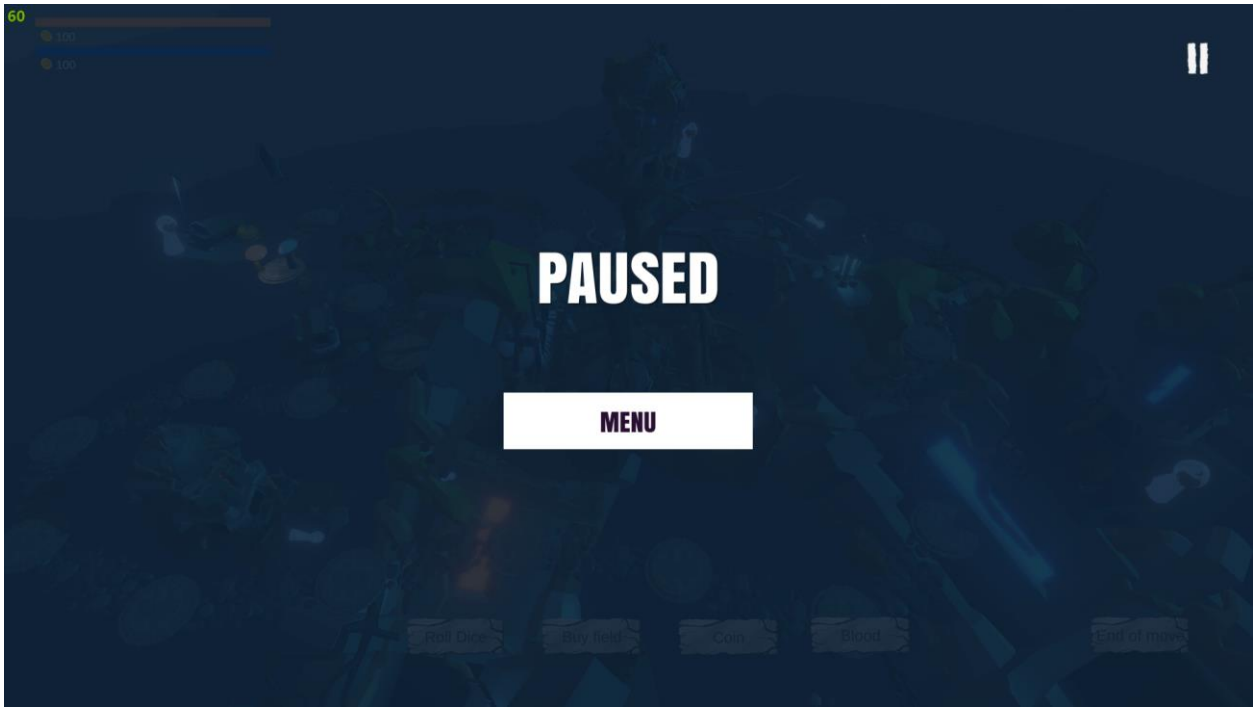


Рисунок 3.6 — Меню паузи

Один з важливих елементів графічного інтерфейсу гри - підсвічування власності поля відповідним кольором гравця під час процесу купівлі. Це важливий візуальний ефект, який допомагає гравцям швидко розпізнати, які поля вони вже придбали, а які залишаються вільними. (рис. 3.7)

Рисунок 3.7 — Демонстрація придбаних полів

Одним з елементів графічного інтерфейсу гри є відображення характеристик поточного поля, на якому знаходиться гравець, в лівому нижньому куті інтерфейсу. Ця функціональність допомагає гравцям отримувати важливу інформацію про власність та стан поля без необхідності шукати цю інформацію в інших частинах інтерфейсу.

Таке відображення забезпечує легкість сприйняття інформації, поліпшує навігацію та збільшує зручність геймплею. Гравці можуть швидко оцінювати важливі аспекти гри і приймати відповідні рішення без зайвих зусиль.

4. ТЕСТУВАННЯ

Тестування є процесом перевірки та оцінки програмного забезпечення з метою виявлення помилок, недоліків та відхилень від вимог. Це систематичний підхід, який дозволяє перевірити, чи працює програма відповідно до очікувань та чи задовольняє вона вимогам користувачів.

Тестування допомагає виявити помилки у програмному забезпеченні та забезпечити його якість. Це включає в себе перевірку функціональності, ефективності, надійності, безпеки та інших аспектів програми. Тестування може бути автоматизованим або виконуватися вручну, в залежності від характеру проекту та ресурсів, доступних для тестування.

Мета тестування полягає в забезпеченні якості програмного забезпечення та зниженні ризиків використання неповного або помилкового програмного забезпечення. В процесі тестування виявляються помилки, що дозволяє розробникам виправити їх до випуску продукту. Також тестування допомагає підтвердити відповідність програмного забезпечення вимогам та забезпечити задоволення потреб користувачів.

Тестування включає в себе розробку тестових сценаріїв, виконання тестів, аналіз результатів і виправлення помилок. Це важливий етап розробки програмного забезпечення, оскільки допомагає підтвердити його якість та забезпечити високу рівень надійності та задоволення користувачів.

4.1.Типи тестів

У процесі тестування програмного забезпечення застосовуються різноманітні типи тестів, кожен з яких спрямований на перевірку певних аспектів функціональності, якості та надійності програми. Нижче наведені деякі з найпоширеніших типів тестів, які можуть бути використані в процесі розробки гри:

1. Функціональні тести. Цей тип тестування перевіряє, чи працює програмне забезпечення відповідно до очікувань та відповідає вимогам. Функціональні тести перевіряють коректність реалізації функцій, модулів та взаємодії між ними.

2. Інтеграційні тести. Цей тип тестування перевіряє взаємодію між різними компонентами або модулями програми. Вони дозволяють виявити проблеми, що можуть виникнути під час злиття різних частин програмного забезпечення та їх взаємодії.

3. Одиничні тести. Цей тип тестування перевіряє правильність роботи окремих функцій, методів або модулів програми. Одиничні тести виконуються на рівні окремих компонентів з метою виявлення помилок ізольовано від інших частин системи.

4. Стрес-тести. Цей тип тестування перевіряє стійкість та витривалість програмного забезпечення під навантаженням. Стрес-тести виконуються з максимальними навантаженнями, великою кількістю користувачів або інтенсивними операціями з метою виявлення слабких місць та перевірки стійкості системи.

5. Тести на безпеку. Цей тип тестування перевіряє вразливості програмного забезпечення та його здатність відстоятися від різних атак. Включає проведення тестів на перехоплення даних, злам системи, перевірку на вразливості безпеки мережі та інші типи атак. Мета таких тестів - забезпечити високий рівень безпеки системи та захисту конфіденційної інформації.

6. Тести на продуктивність. Цей тип тестування оцінює продуктивність програми в різних ситуаціях навантаження. Включає в себе тестування швидкодії, завантаження пам'яті, відповіді на запити користувачів та інші аспекти продуктивності. Ці тести допомагають виявити можливі проблеми, такі як перевантаження, медлену відповідь або низьку продуктивність системи.

7. Тести на відновлення. Цей тип тестування перевіряє здатність програмного забезпечення відновлюватися після виникнення помилок або аварійних ситуацій. Включає проведення тестів на відновлення після збоїв, відновлення даних після втрати та інші випадки відновлення системи до працездатного стану.

8. Тести на сумісність. Цей тип тестування перевіряє сумісність програми з різними операційними системами, пристроями, браузерами та іншими компонентами програмного середовища. Ці тести допомагають забезпечити, що програмне забезпечення буде працювати на різних платформах та в різних конфігураціях, не порушуючи його функціональності та якості.

Користуючись різними типами тестів, розробники гри можуть провести комплексне тестування, яке охопить різні аспекти функціональності, якості та надійності програми. Кожен тип тесту має свої особливості і спрямований на виявлення конкретних проблем інтерфейсу, функцій, продуктивності або безпеки.

Під час тестування важливо мати чіткі критерії успішності, які визначають, коли тест вважається пройденим або непройденим. Тестові випадки повинні бути детально задокументовані, щоб забезпечити повторюваність і послідовність виконання. Крім того, для ефективного тестування можуть бути використані автоматизовані інструменти та фреймворки, що допомагають автоматизувати виконання тестів і забезпечити більш швидке виявлення помилок.

Тестування є важливою складовою процесу розробки гри, оскільки допомагає виявити помилки та проблеми, покращити якість продукту і забезпечити задоволення користувачів. Ефективне тестування допомагає зменшити ризики, пов'язані з випуском гри, і забезпечує більш високу впевненість в якості та надійності програми.

4.2 Тестові сценарії

Тестові сценарії визначають послідовність дій, які будуть виконуватися для перевірки різних аспектів гри. Нижче наведені декілька прикладів тестових сценаріїв, які були розроблені для гри.

"Переміщення гравця". У цьому сценарії тестується правильність роботи механізму переміщення гравця по ігровому полю. Сценарій передбачає кидання кубика, отримання випадкового значення і виконання відповідної кількості кроків на ігровій дошці. Тестування проводиться для різних значень кубика та різних

початкових позицій гравця, щоб переконатися, що гравець рухається правильно та досягає

"Купівля власності". У цьому сценарії перевіряється правильність роботи механізму купівлі власності. Сценарій включає в себе ситуацію, коли гравець досягає вільного поля на ігровій дошці та має достатню кількість монет для його придбання. Тестування проводиться для різних полів та різних сум, щоб переконатися, що процес купівлі власності відбувається правильно та гравець правильно отримує володіння над полем.

"Нанесення шкоди противнику". У цьому сценарії перевіряється правильність роботи механізму нанесення шкоди противнику. Сценарій включає в себе ситуацію, коли гравець стає на поле, яке належить його суперника. Гравець має можливість обрати тип шкоди, яку він хоче завдати противнику - відняти його життєву силу або забрати певну кількість монет. Тестування проводиться для різних комбінацій типів шкоди та різних значень життєвої сили та монет у противника, щоб переконатися, що механізм нанесення шкоди працює правильно та відбувається відповідне зменшення життєвої сили або монет у противника.

"Отримання монет". У цьому сценарії перевіряється правильність роботи механізму отримання монет у грі. Сценарій передбачає проходження гравцем крізь різні поля на ігровій дошці, які мають властивість приносити монети. Тестування проводиться для різних полів та різних значень монет, щоб переконатися, що гравець отримує відповідну кількість монет при проходженні через такі поля. Також перевіряється правильність відбирання монет у конкурента, коли гравець потрапляє на поле, яке належить іншому гравцю.

"Завершення гри". Цей сценарій перевіряє правильність завершення гри після досягнення певної умови. У даному випадку, умовою є втрата всієї життєвої сили одним із гравців. Тестування проводиться шляхом встановлення початкових значень життєвої сили гравців, а потім виклику подій, які призводять до зменшення життєвої сили гравця до нуля. Після цього перевіряється правильність завершення гри та відображення відповідного повідомлення про перемогу одного з гравців.

"Інтерфейс користувача". У цьому сценарії перевіряється коректність роботи інтерфейсу користувача гри. Тестування проводиться шляхом спроб взаємодії з різними елементами інтерфейсу, такими як кнопки, поля вводу, віджети тощо. Перевіряється відповідність реакцій на дії користувача, правильність відображення інформації, а також забезпечення зручності та логічності інтерфейсу. Також перевіряється адаптивність інтерфейсу до різних розмірів екрану та пристроїв, що дозволяє забезпечити зручну гру для користувачів на різних платформах.

Тестовий сценарій - "Переміщення гравця"

Кроки:

1. Початок гри з встановленою початковою позицією гравця на ігровій дошці.
2. Виконання кроку переміщення гравцем на певну кількість полів.
3. Перевірка коректності переміщення гравця шляхом перевірки його нової позиції на ігровій дошці.
4. Виконання додаткових перевірок, таких як перевірка наявності перешкод або обмежень для переміщення на певні поля.
5. Повторення кроків 2-4 для різних варіантів переміщення, включаючи переміщення вперед, назад, вліво та вправо.
6. Перевірка реакції гри на некоректні дії, наприклад на зайняте поле.
7. Завершення тестового сценарію та перевірка правильності результатів переміщення гравця.

Цей тестовий сценарій дозволяє перевірити коректність та надійність механізму переміщення гравця в грі. Виконуючи різні кроки переміщення і перевіряючи результати, можна переконатись у правильності розрахунків та точності переміщення гравця на ігровій дошці.

Тестовий сценарій - "Купівля власності"

Кроки:

1. Початок гри з гравцем, який має певну кількість монет та знаходиться на полі, яке можна придбати.
2. Виконання кроку купівлі власності гравцем, який придбуває вільне поле.

3. Перевірка коректності купівлі власності шляхом перевірки статусу поля після покупки.
4. Перевірка зменшення кількості монет у гравця після купівлі власності.
5. Виконання додаткових перевірок, таких як перевірка можливості купівлі власності на зайнятому полі або полі, яке належить іншому гравцю.
6. Повторення кроків 2-5 для різних варіантів купівлі власності.
7. Перевірка реакції гри на некоректні дії, наприклад, спробу купити власність без достатньої кількості монет.
8. Завершення тестового сценарію та перевірка правильності результатів купівлі власності.

Цей тестовий сценарій дозволяє перевірити правильність та надійність механізму купівлі власності в грі. Виконуючи кроки купівлі та перевіряючи результати, можна переконатись у коректності розрахунків та успішності купівлі власності гравцем.

Тестовий сценарій - "Нанесення шкоди противнику"

Кроки:

1. Початок гри з двома гравцями, розташованими на різних полях.
2. Виконання кроку переміщення одного з гравців на поле, що належить противнику.
3. Вибір гравцем типу шкоди, яку він хоче нанести противнику.
4. Застосування вибраної шкоди до противника та зменшення життєвої сили або кількості монет противника відповідно до типу шкоди.
5. Перевірка коректності нанесення шкоди шляхом перевірки статусу противника після здійснення шкоди.
6. Перевірка реакції гри на некоректні дії, наприклад, спробу нанести шкоду на власність або на поле, яке не належить противнику.
7. Повторення кроків 2-6 для різних варіантів нанесення шкоди противнику.
8. Завершення тестового сценарію та перевірка правильності результатів нанесення шкоди.

Цей тестовий сценарій дозволяє перевірити правильність та ефективність механізму нанесення шкоди противнику в грі. Виконуючи кроки нанесення шкоди та перевіряючи результати, можна переконатись у коректності логіки гри та правильному врахуванні шкоди, що наноситься противнику.

Тестовий сценарій - "Отримання монет"

Кроки:

1. Початок гри з двома гравцями, розташованими на різних полях.
2. Виконання кроку переміщення гравця по ігровій дошці, що призводить до досягнення поля з монетами.
3. Збирання монет гравцем, які знаходяться на полі, на якому він зупинився.
4. Перевірка коректності збирання монет шляхом перевірки зміни кількості монет гравця після збирання.
5. Перевірка реакції гри на некоректні дії, наприклад, спробу збирати монети з поля, яке не містить монет або зайняте іншим гравцем.
6. Повторення кроків 2-5 для різних полів з монетами та різних ситуацій.
7. Завершення тестового сценарію та перевірка правильності результатів отримання монет.

Цей тестовий сценарій дозволяє перевірити правильність та ефективність механізму отримання монет в грі. Виконуючи кроки збирання монет та перевіряючи результати, можна переконатись у коректності логіки гри та правильному врахуванні монет, які гравець отримує при проходженні полів з монетами.

Тестовий сценарій - "Завершення гри"

Кроки:

1. Початок гри з двома гравцями, кожен з них має початкову життєву силу та кількість монет.
2. Виконання кроків гри, таких як переміщення гравців, купівля власностей, нанесення шкоди противникові та отримання монет.
3. Перевірка стану гри після кожного кроку для визначення, чи є гра ще в процесі.

4. Перевірка умови завершення гри, яка може бути досягнута, наприклад, коли один із гравців втрачає всю життєву силу.
5. Завершення гри та оголошення переможця або нічиєї, залежно від умов завершення.
6. Перевірка правильності визначення переможця, зокрема перевірка життєвої сили та кількості монет у гравців.
7. Перевірка правильності показу результатів гри, включаючи відображення переможця та відповідну інформацію про нього.

Цей тестовий сценарій дозволяє перевірити коректність та правильність механізму завершення гри. Виконуючи кроки гри та перевіряючи умови завершення, можна переконатись у правильному визначенні переможця та коректному закінченні гри.

Тестовий сценарій - "Інтерфейс користувача"

Кроки:

1. Запуск гри та відображення початкового екрану інтерфейсу користувача.
2. Перевірка наявності основних елементів інтерфейсу, таких як кнопки, поля вводу, текстові блоки, ігрова дошка тощо.
3. Взаємодія з елементами інтерфейсу, наприклад, натискання кнопок, введення даних у поля вводу, переміщення по ігровій дошці тощо.
4. Перевірка правильності реакції інтерфейсу на дії користувача, наприклад, зміна відображення після натискання кнопок, відображення результатів дій користувача на ігровій дошці тощо.
5. Перевірка функціональності елементів інтерфейсу, наприклад, перевірка правильності введених даних, відображення вірного стану гри на ігровій дошці тощо.
6. Перевірка візуального відображення інтерфейсу, зокрема перевірка зручності сприйняття інтерфейсу, логічності розміщення елементів, чіткості та зрозумілості відображення інформації.
7. Перевірка реагування інтерфейсу на зміни в стані гри, наприклад, перевірка оновлення інформації після виконання дій користувача.

Цей тестовий сценарій дозволяє перевірити коректність та зручність використання інтерфейсу користувача. Виконуючи кроки тестового сценарію, можна переконатись у правильній реакції інтерфейсу на дії користувача, правильному відображенні інформації та зручності сприйняття інтерфейсу гравцем.

Тестовий сценарій - "Продуктивність"

Кроки:

1. Запуск гри з повним функціоналом і великою кількістю елементів на ігровій дошці.
2. Відстеження частоти кадрів (FPS) під час активної гри.
3. Перевірка стабільності частоти кадрів під час різних дій гравця, таких як переміщення, купівля власності, нанесення шкоди противнику тощо.
4. Перевірка відсутності помітних затримок або просідань частоти кадрів при виконанні складних операцій, наприклад, при великій кількості одночасних розрахунків або великому обсязі графічних об'єктів на екрані.
5. Перевірка завантаження системних ресурсів під час гри, таких як використання процесорного часу, пам'яті та інших ресурсів.
6. Перевірка наявності можливості запуску гри на різних пристроях з різними характеристиками (наприклад, настільні комп'ютери, ноутбуки, мобільні пристрої) і перевірка продуктивності гри на цих пристроях.
7. Аналіз результатів тестування для оцінки продуктивності гри та виявлення можливих проблем, які можуть вплинути на гравців, таких як низька частота кадрів, затримки або нестабільна робота.

Цей тестовий сценарій дозволяє оцінити продуктивність гри та переконатись, що вона працює ефективно та стабільно на різних пристроях і в різних ситуаціях.

Після завершення процесу тестування, гра пройшла успішно всі основні тести, що відповідають вимогам гри. Всі функціональні та нефункціональні вимоги були перевірені і виконані згідно заданих критеріїв успішності.

ВИСНОВКИ

Під час дослідження та аналізу існуючих ігор в жанрі Покрокова стратегія були виявлені їхні особливості та ігрові механіки. Це дало можливість визначити ключові вимоги до розробки гри. Використання ігрового рушія Unity та мови програмування C# дозволило успішно реалізувати необхідні механіки гри, забезпечивши стабільну роботу та реалістичність ігрового процесу.

Проектування архітектури гри було виконано з урахуванням визначених вимог. Були розроблені основні компоненти гри, включаючи головний логічний модуль, систему керування гравцем, механіку руху та пошкоджень.

Розробка інтерфейсу користувача була успішно виконана, дозволяючи зручно взаємодіяти з грою та контролювати хід подій. Візуальне відображення гри було реалізовано на високому рівні, забезпечуючи естетичний та привабливий вигляд.

Тестування гри було проведено згідно з запланованими тестовими сценаріями. Гра пройшла успішно всі тести, включаючи перевірку функціональності, взаємодії гравців, правильності розрахунків та графічного відображення. Виявлено та виправлено помилки на ранніх етапах розробки.

У результаті дослідження, проектування та реалізації гри в жанрі Покрокова стратегія з використанням ігрового рушія Unity та мови програмування C#, було створено функціональну, естетично привабливу гру та підвищено якість гри у жанрі Покрокова стратегія.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жанри відеоігор [Електронний ресурс] // TV Tropes. – 2023. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Жанри_відеоігор.
2. Unity User Manual [Електронний ресурс] // Unity Technologies. – 2023. – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/index.html.
3. Post Processing Stack [Електронний ресурс] // Unity Technologies. – 2022. – Режим доступу до ресурсу: https://docs.unity3d.com/Packages/com.unity.postprocessing@3.2/manual/index.html.
4. Brown F. The history of the strategy game [Електронний ресурс] / Fraser Brown // pcgamer. – 2021. – Режим доступу до ресурсу: https://www.pcgamer.com/the-history-of-the-strategy-game/.
5. What Is Game Development? [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.freecodecamp.org/news/what-is-game-development/.
6. What is game development? Everything you need to know [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.pulsecollege.com/what-is-game-development-everything-you-need-to-know/
7. Schell J. The Art of Game Design / Schell J. - Florida : CRC Press, 2008. - 520p.
8. Video Game Genres: Everything You Need to Know [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.hp.com/us-en/shop/tech-takes/video-game-genres.
9. Hosch W. electronic strategy game [Електронний ресурс] / William Hosch – Режим доступу до ресурсу: https://www.britannica.com/topic/electronic-strategy-game.

10. Unity [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://unity.com.

11. The Best Gaming Engines You Should Consider for 2023 [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.incredibuild.com/blog/top-gaming-engines-you-should-consider.

12. Visual Studio [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://visualstudio.microsoft.com.

13. C# Language Documentation [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/.

14. What is Microsoft Visual Studio? [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://softwarekeep.com/help-center/what-is-microsoft-visual-studio-where-can-i-download-it.

15. Hamilton T. Game Testing: Types & How to Test Mobile/Desktop Apps [Електронний ресурс] / Thomas Hamilton. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/game-testing-mobile-desktop-apps.html.

16. Game Testing 101: Basic Tips and Strategies [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://starloopstudios.com/game-testing-101-tips-and-strategies/.

17. UML Use Case Diagram Tutorial [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram.

18. Unity in Action: Multiplatform Game Development in C# with Unity 5/ Joe Hocking//2018 p.-80 c.

ДОДАТОК А

ПРЕЗЕНТАЦІЯ ДО ЗВІТУ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО -НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ГРИ В ЖАНРІ ПОКРОКОВА СТРАТЕГІЯ ЗА ДОПОМОГОЮ ІГРОВОГО РУШІЯ UNITY МОВОЮ C#

Виконав студент 4 курсу
групи ПД-44
Ярошевський Олександр Вікторович

Керівник роботи
доктор філософії, доцент кафедри ІПЗ Дібрівний Олесь Андрійович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - підвищення якості гри в жанрі Покрокова стратегія за допомогою ігрового рушія Unity та C#.
- **Об’єкт дослідження –** геймплей в жанрі Покрокова стратегія.
- **Предмет дослідження –** програмне забезпечення для реалізації геймплею гри в жанрі Покрокова стратегія.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Дослідження та аналіз існуючих ігор в жанрі Покрокова стратегія.
2. Визначення вимог до розробки гри.
3. Проектування архітектури гри.
4. Реалізація механік гри з використанням ігрового рушія Unity та C#.
5. Розробка інтерфейсу та графічного відображення гри.
6. Тестування гри.

АНАЛІЗ АНАЛОГІВ

Критерії порівняння	Monopoly	Pummel Party	For the King	Step-by-step strategy
Системні вимоги	8 GB RAM; Intel i7-6700; NVidia GeForce GTX 960	3 GB RAM; Intel Core i5; GeForce 8800 GT	4 GB RAM; Intel Core i5; GeForce GTX 750 Ti	2GB RAM; 2.5 GHz Dual Core CPU; Geforce GTX 970
Складність геймплею	Середня	Висока	Висока	Низька
Тривалість ігрової партії	1 – 6 годин	1 – 3 години	1 – 6 годин	20 – 40 хвилин
Складність сприйняття ігрового інтерфейсу	Висока	Середня	Висока	Низька
Механіка купівлі власності	Наявна	Відсутня	Відсутня	Наявна
Механіка нанесення шкоди противнику	Відсутня	Наявна	Наявна	Наявна
Стабільність частоти кадрів	Від 60 до 35 к/с	Від 60 до 30 к/с	Від 60 до 20 к/с	Від 60 до 55 к/с
Наявність ігрових дефектів	Візуальний дефект при пересуванні гравця	Дефект управління персонажем	Затримка передачі ходу від 3 до 10 секунд	Відсутні

КОНЦЕПТ ГРИ

- Два гравці рухаються по ігровій дошці, кидаючи кубик, що визначає кількість можливих кроків.
- В ході гри відбувається протистояння між суперниками за володіння полями.
- Ставши на вільне поле, гравець може його придбати за наявності достатньої кількості монет.
- Заволодівши полем, гравець обирає яку шкоду нанести противнику, що стане на його поле: відняти життєву силу або забрати певну кількість монет, що передбачена властивістю поля.
- Повторно ставши на свою власність, гравець може змінити тип шкоди.
- Отримати монети можна пройшовши коло або відібравши їх у конкурента.
- Гра закінчується, коли один із гравців втрачає всю життєву силу.

ВИМОГИ ДО ГРИ

Функціональні можливості гравця:

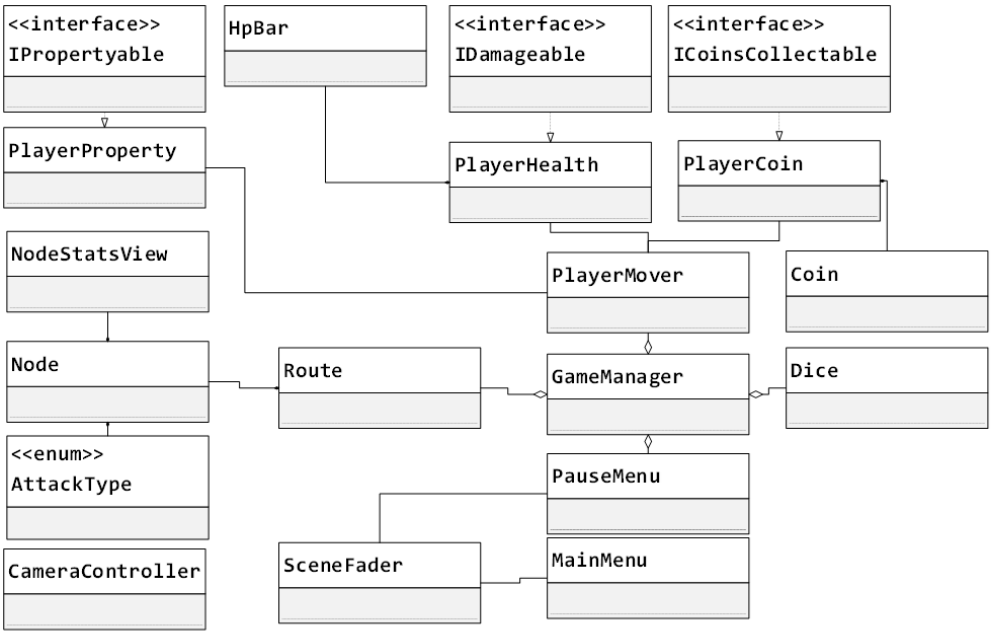
1. Можливість почати та закінчити гру.
2. Кидок кубика для визначення кількості кроків.
3. Рух по ігровому полю відповідно до кількості кроків.
4. Купівля поля та збирання монет.
5. Нанесення фізичної шкоди супернику за допомогою власних полів.
6. Взаємодія з грою за допомогою клавіш та елементів інтерфейсу .

Нефункціональні вимоги до гри

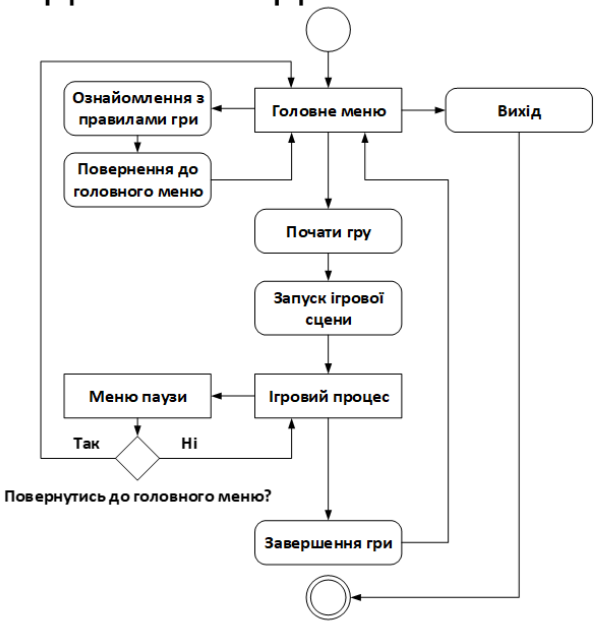
1. Сумісність з операційною системою Windows.
2. Забезпечення стабільно го оновлення частоти кадрів в межах 50 – 60к/с.
3. Мінімальні системні вимоги: RAM: 1 GB; Процесор: 2 GHz Dual Core CPU; Відеокарта: Intel HD Graphics 4000.
4. Рекомендовані системні вимоги: RAM: 2GB; Процесор: 2.5 GHz Dual Core CPU; Відеокарта: Geforce GTX 970.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

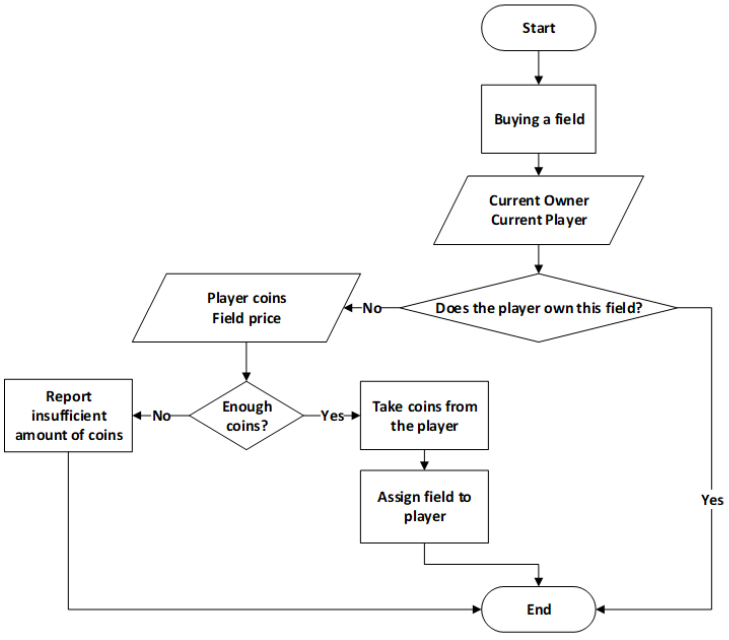
ДІАГРАМА КЛАСІВ



ДІАГРАМА ДІЯЛЬНОСТІ



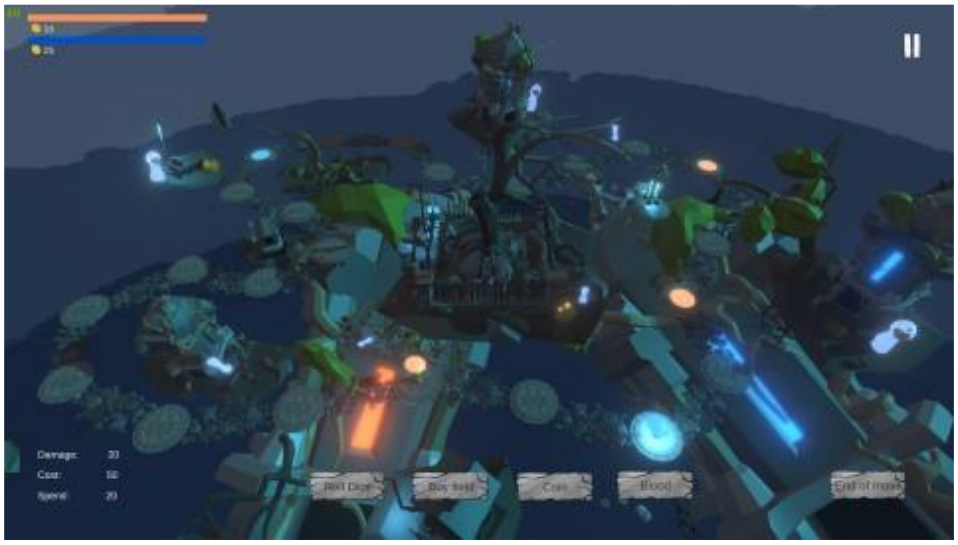
БЛОК-СХЕМА АЛГОРИТМУ ОСНОВНОЇ МЕХАНІКИ



ЕКРАННІ ФОРМИ

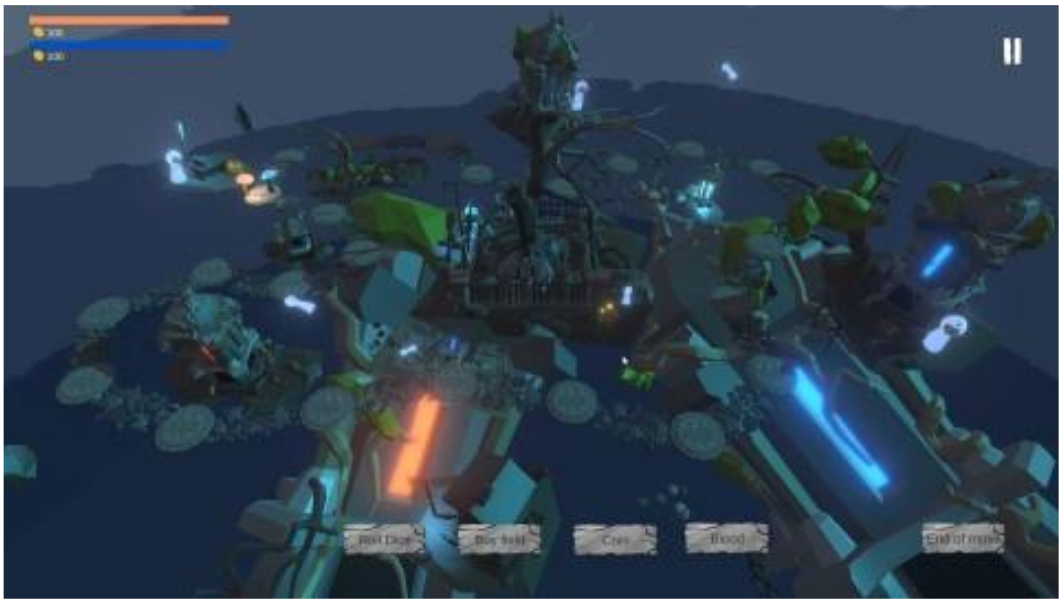
Головне меню

ЕКРАННІ ФОРМИ



Сцена ігрового поля

ДЕМОНСТРАЦІЯ ГРИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Ярошевський О.В. Розробка гри монополія / Дібрівний О.А, Ярошевський О.В. // Застосування програмного забезпечення в інфокомунікаційних технологіях: матеріали всеукраїнської науково-технічної конференції. Збірник тез. 20 квітня 2023 року, ДУТ, м. Київ – К: ДУТ, 2023. – С. 123 – 124.

2. Ярошевський О.В. Покрокова стратегія / Ярошевський О.В. // Хакатон «Start Smart» 25 травня 2023. ДУТ, м. Київ: Сертифікат за участь Ярошевський Олександр Вікторович в категорії «GameDev».

ВИСНОВКИ

1. Досліджено та проаналізовано ігри в жанрі Покрокова стратегія щодо їх особливостей та ігрових механік. Визначено основні переваги та недоліки. Встановлено ключові вимоги до функціональності та інтерфейсу гри.
2. Розроблено гру з використанням Unity та C#. Реалізовано механіки купівлі власності та нанесення шкоди противнику. Забезпечено стабільність частоти кадрів, що дозволило підвищити якість гри в жанрі Покрокова стратегія.
3. Розроблено інтерфейс користувача, що дає змогу взаємодіяти з грою за допомогою його елементів.
4. Проведено тестування гри з урахуванням вимог основних компонентів, яке виявило помилки та надало змогу усунути їх на ранніх етапах розробки.

ДЯКУЮ ЗА УВАГУ!