Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Ігровий застосунок у жанрі Strategy на основі патерну MVVM та технології Unity

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-1
Андрій СЄРОВ
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Олександр ФОМІЧОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сєрову Андрію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий застосунок у жанрі Strategy на основі патерну MVVM та Технології Unity

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи

1) Документація мови програмування C#
2) Документація Unity
3) Інтегроване середовище: Cursor

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз предметної області
2) патерн MVVM
3) реалізація програмного застосунку в жанрі Strategy
4) інструкція користувача
5) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація - 20 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	30.05.25 - 31.05.25	
2	Формування переліку вимог до програми	01.06.25 - 02.06.25	
3	Вибір технології розробки та	03.06.25-05.06.25	
	інструментальних засобів		
4	Розробка та тестування програми	06.06.25 - 11.06.25	
5	Оформлення матеріалів кваліфікаційної роботи	11.06.25 - 15.06.25	
6	Подання квіліфікаційної роботи керівникові та	16.06.25 - 19.06.25	
	її попередній захист		
7	Подання кваліфікаційної роботи на	19.06.25 - 20.06.25	
	рецензування		

Дата видачі завдання " 29 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Олександр ФОМІЧОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 62 с., 9 рис., 0 табл., 1 дод., 20 джерел.

СТРАТЕГІЯ, Unity, MVVM, ГЕЙМДЕВ, C#, ІГРОВИЙ РУШІЙ, РЕСУРС-МЕНЕДЖМЕНТ, ІНТЕРФЕЙС, АРХІТЕКТУРА, ІГРОВА ЛОГІКА.

Метою кваліфікаційної роботи є створення стратегічної гри з використанням Unity та архітектурного шаблону MVVM

Під час кваліфікаційної роботи я розробив гру, яка містить такі ключові механізми, як управління ресурсами, будівництво будівель, зростання населення та зручний інтерфейс. Я зосередився на структурі проекту. Я чітко розділив логіку, візуальні елементи та дані, що полегшує підтримку та розширення гри. Щоб зібрати ідеї та рішення, я грав у схожі ігри для натхнення та аналізу. Більшість ресурсів та інструментів, які я використовував, були безкоштовними, що робило проект доступним для будь-якого ентузіаста. Фонова музика надійшла з іншої гри і не може використовуватися в комерційних цілях. Загалом, це стратегічна гра, яка демонструє мої навички програмування та здатність Unity створювати складні ігрові системи.

ABSTRACT

Bachelor’s thesis: 62 pages, 9 figures, 0 tables, 1 appendices, 20 sources.

STRATEGY, UNITY, MVVM, GAMEDEV, C#, GAME ENGINE, RESOURCE MANAGEMENT, INTERFACE, ARCHITECTURE, GAME LOGIC.

The major goal of this thesis is to create a strategy game using Unity and the MVVM architectural pattern

In order to developed a game that features key mechanics like resource management, building construction, population growth, and a user-friendly interface. I focused closely on the project structure. I clearly separated the logic, visuals, and data, which makes it easier to support and expand the game. To gather ideas and solutions, I played similar games for inspiration and analysis. Most of the resources and tools I used were free, making the project accessible to any enthusiast. The background music came from another game and cannot be used for commercial purposes. In summary, this is a strategy game that showcases my programming skills and Unity's ability to create complex game systems.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ - програмне забезпечення

ПК - персональний комп’ютер

ШІ - штучний інтелект

Asset - ресурс або об’єкт у грі (модель, текстура, звук тощо)

Blueprint - візуальна система програмування в Unreal Engine

CPU - центральний процесор (Central Processing Unit)

C# - мова програмування C Sharp

GPU - графічний процесор (Graphics Processing Unit)

MVVM - Model-View-ViewModel (архітектурний патерн)

MVP - Model-View-Presenter (архітектурний патерн)

MVC - Model-View-Controller (архітектурний патерн)

Prefab - підготовлений ігровий об’єкт в Unity

RAM - оперативна пам’ять (Random Access Memory)

RTS - Real-Time Strategy (жанр стратегій у реальному часі)

UI - інтерфейс користувача (User Interface)

UE5 - Unreal Engine 5

ВСТУП

Сфера розробки відеоігор протягом останніх двадцяти років активно розвивається, демонструючи стабільне зростання як у технічному, так і в креативному плані. Цей напрямок індустрії став одним із найважливіших складників сучасної цифрової культури. Проте за останні п’ять років мене дедалі більше почала розчаровувати ситуація, пов’язана з використанням штучного інтелекту - зокрема, для генерації зображень, кадрів та масштабування графіки у відеоіграх. Хоча з технічної точки зору ці інструменти можуть пришвидшити розробку, часто вони призводять до зниження якості, однотипності контенту та втрати унікального художнього стилю, що раніше був характерний для окремих студій.

Сьогодні ігри стали частиною життя молоді. Багато хто сприймає їх не лише як засіб розваги, а й як один із основних способів спілкування, самовираження та навіть навчання. Особливо яскраво цей тренд проявився під час пандемії COVID-19, коли велика частина людей, що раніше не цікавилися іграми, почали шукати нові форми соціалізації в умовах ізоляції. Онлайн-ігри стали зручним і веселим способом комунікації, який не вимагає фізичної присутності.

Однак варто згадати і про тих, хто стоїть за створенням ігор - про розробників, команди та студії. На жаль, сьогодні існує велика кількість компаній, що фокусуються виключно на прибутках, створюючи проєкти, головною метою яких є витягування грошей із користувачів без особливих зусиль щодо якості чи оригінальності. Особливо це стосується мобільного сегменту - близько 90% мобільних ігор використовують агресивну рекламу як основний або додатковий метод монетизації.

На фоні цього комп’ютерні ігри залишаються відносно більш контрольованим середовищем. Великим чином це пов’язано з політикою платформи Steam, яка встановлює чіткі правила щодо якості контенту,

забороняє розміщення рекламних оголошень у самих іграх та має ефективну систему рейтингів і відгуків. Саме завдяки цьому гравці можуть орієнтуватися на думки інших користувачів і уникати неякісних або шкідливих проєктів.

Серед найпопулярніших на сьогодні ігор переважають змагальні жанри, зокрема такі тайтли як Counter-Strike 2, Dota 2, Valorant та PUBG. Такі ігри приваблюють користувачів швидкою динамікою, командною взаємодією та постійним оновленням. Однак водночас все ще зберігається популярність стратегічного жанру. Стратегії дозволяють гравцям заглибитися в складні сценарії , в яких необхідно приймати зважені рішення, керувати ресурсами, будувати бази або армії, що вимагає планування і тактичного мислення.

Сучасна індустрія розробки ігор спирається на широкий набір технологій і інструментів. Серед них особливе місце займає Unity - один з найпопулярніших ігрових рушіїв, що дозволяє створювати як 2D, так і 3D ігри під різноманітні платформи від ПК до мобільних пристроїв і навіть web . Основними перевагами Unity є велика спільнота розробників, потужна документація, наявність Asset Store з безліччю готових рішень та підтримка сучасних архітектурних патернів [8].

У рамках цієї дипломної роботи буде розроблено гру в жанрі стратегії з використанням архітектурного патерну MVVM (Model-View-ViewModel). Такий підхід дозволяє чітко розділити логіку інтерфейсу користувача та внутрішню ігрову логіку, що позитивно впливає на модульність і підтримку проєкту. Це особливо корисно при роботі в команді або в умовах розширення проєкту в майбутньому. Обрана тема є актуальною, оскільки поєднує в собі практичне застосування ігрового рушія Unity з сучасним підходом до програмування, дозволяючи створити гнучкий та добре структурований код . Головною метою є реалізація простої, але захоплюючої стратегічної гри, яка демонструватиме основні механіки жанру, водночас слугуючи прикладом грамотної архітектури програмного продукту [19].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Стратегія як тип програмного продукту

Ігри жанру стратегії займають важливе місце в індустрії відеоігор завдяки своїй здатності поєднувати розважальний процес із розвитком когнітивних навичок гравців [17]. Вони вимагають від користувача не лише швидкої реакції, а й глибокого стратегічного мислення, планування та управління ресурсами. Цей жанр охоплює різноманітні піджанри, кожен з яких має свої особливості та механіки.

Згідно з класифікацією, запропонованою QATestLab, стратегічні ігри можна поділити на кілька основних піджанрів:

- Економічні стратегії (Economic strategy): У цих іграх гравець виступає в ролі власника міста, країни або підприємства, завданням якого є розвиток території через видобуток ресурсів, виробництво товарів та їх продаж. Прикладом такої гри є Capitalism II;
- Ігри «захисні башти» (Tower defense): Гравець будує захисні споруди для автоматичного знищення ворогів. Ці ігри вимагають швидкої реакції та стратегічного планування для ефективного розміщення оборонних споруд. Прикладом є Plants vs. Zombies;
- Військові стратегії (Wargame): Гравець керує бойовими підрозділами, виконуючи місії з метою досягнення стратегічних цілей. Ігри можуть бути як покроковими, так і в реальному часі. Прикладом є Blitzkrieg .
- Карткові стратегії (Cardgame): Гравець використовує набір карт для битви з опонентами, де кожна карта представляє окремий підрозділ або дію. Прикладом є Hearthstone;

Стратегічні ігри, як програмний продукт, мають низку характерних рис:

- Складна логіка та механіка: Реалізація різноманітних ігрових

механік, таких як управління ресурсами, штучний інтелект супротивників, економічні системи та інші, вимагає ретельного планування та оптимізації коду;

- Масштабність та модульність: Багато стратегічних ігор мають великі карти, численні одиниці та складні системи, що потребують ефективної архітектури програмного забезпечення для забезпечення стабільної роботи;
- Інтерактивний інтерфейс: Розробка зручного та інтуїтивно зрозумілого інтерфейсу є важливою складовою, оскільки гравець повинен мати змогу швидко реагувати на зміни в ігровому середовищі;
- Мультимедійні компоненти: Високоякісна графіка, анімація, звук та музика сприяють зануренню гравця в ігровий процес, підвищуючи його задоволення від гри.

1.2 Розгляд аналогічних продуктів або рішень

Перед розробкою власного продукту доцільним є розгляд вже існуючих ігор в жанрі стратегії, зокрема тих що подібні за механіками, візуальним стилем або цільовою аудиторією.

Anno (Ubisoft Blue Byte) - цей проект предствляє собою серію ігор в якій кожна окрема гра повторює попередню змінюючи сюжет та час дій. Основні елементи моєї гри будуть взяті або натхненні Anno 1800 [18].

Ця серія ігор акцентує увагу на управлінні містом, логістиці, торгівлі та задоволенні потреб населення. Гравець будує виробничі ланцюги, управляє економікою і дипломатією.

На що варто звернути увагу:

- Детальна система ресурсів і виробництва;
- Поступовий розвиток населення через соціальні рівні;
- Сильна прив’язка до візуального фідбеку (акцент на зовнішній вигляд міста);
- Майже відсутність бойової складової як основного елементу

геймплею.

Cossacks(GSC Game World) - одна з найвідоміших стратегій українського виробництва, яка зосередженна на масштабних битвах, мікроконтролі юнітів та економічному менеджменті.

На що треба звернути увагу:

- Підтримка до 8000 юнітів на карті;
- Ресурсозалежна економіка, де кожна дія потебує певних запасів;
- Сильний історичний компонент, що впливає на геймдизайн та баланс;
- Відокремлення бойових та цивільних юнітів.

Age of Empires - гра охоплює розвиток цивілізації від темних віків до Імперської епохи. Вона поєднує збір ресурсів, будівництво баз, тренування армії та дослідження технологій [19].

На що варто звернути увагу:

- Чітко збалансовані цивілізації з унікальними юнітами та технологіями;
- Класичне дерево розвитку;
- Підтримка одиночної кампанії та мультиплеєру;
- Проста, але ефективна система контролю.

1.3 Визначення обмежень

Обмеження, визначенні на етапі планування, дозволяють щосередитись на головному: реалізації ігрової механіки та архітектури в рамках можливостей. Це важливо як для ефективного розподілу часу, так і для формування досяжних цілей у дипломному проєкті.

Основними технічними обмеженнями можна назвати такі:

- Цільова платформа - Windows, з можливим портуванням на Linux та MacOS. Це означає, що інтерфейс, управління та продуктивність оптимізуються насамперед для ПК. З портуванням на Linux є обмеження в

підтримці так як існує багато дистрибутивів і перевірити кожен неможливо. Також згадуючи MacOS необхідно мати цей пристрій не просто для тестування, а також для збірки проекту та мати на увазі, що ігрові можливості в цих пристроях дуже скромні та не мають великох кількості функцій як от масштабування або трасування променів. На даний момент найефективнішим варіантом буде зосередитись лише на Windows 11 так як попередні версії вже вийшли з підтримки;

- Ігровий рушій - Unity, який забезпечує достатній функціонал, але потребує врахування продуктивності при роботі зі складними сценами або великою кількістю об’єктів. Порівнюючи його з іншими можна сказати, що його достатньо для непоганої гри, що не має на меті високоякісну та реалістичну графіку;

- Мова програмування - C#, що задає певні обмеження в реалізації настандартної логіки, але є стандартом для Unity, та використовується в великій кількості рушіїв та як самостійний інструмент розробки. Ця мова займає 5-те місце з рейтингом 4.98% згідно з індексом TIOBE за листопад 2024;

- Цільовий ПК для розробки - Intel Core i5-8400, 32 GB RAM, GTX 1660. Цей рівень продуктивності обмежує обсяг і деталізацію графіки, використання високоякісних ассетів або складних фізичних симуляцій. Потрібно врахувати що при розробці гра є неоптимізованою тому розробник повинен мати систему потужнішу за рекомендовану систему користувача. На це впливає відсоток користувачів, що зможуть пограти в гру. Наприклад за даними опитування на платформі Steam станом на грудень 2024 року 10-15% користувачів мають слабшу відеокарту ніж GTX 1660. Таким чином вони можуть мати проблеми з продуктивністю. З цього випливає що всі хто має потужніші відеокарти не зможуть отримати кращого зображення, незважаючи на те що залізо спроможне дати краще зображення [5].

Основнимі часові обмеження, що треба враховувати при розробці:

- Тривалість розробки - приблизно 3 місяці активної роботи, включно

з проєктуванням, реалізацією, тестуванням та написанням супровідної документації. Також необхідно врахувати час на вивчення нових інструментів таких як ігровий рушій та програма для створення 3д моделей;

- Наявність інших навчальних навантажень - дипломна робота виконується паралельно з іншими предметами та підготовкою для екзаменів, що обмежує кількість годин, яку можна стабільно виділяти щодня;

- Реалістичне планування - реалізація гри має обмежитися базовим набором функцій, як побудова та ресурс-менеджмент.

Основними ресурсними обмеженнями є:

- Людські ресурси - роботи виконується індивідуально, тому відсутня можливість паралельної розробки (наприклад, хтось окремо займається арт-дизайном, тестуванням, моделюванням, гейм-дизайном тощо.);

- Ассети та інструменти - використовуються виключно безкоштовні ресурси з Unity Asset Store, AmbientCG або аналогічних платформ. Це впливає на рівень унікальності візуальної частини, а також на використання інструментів для полегшення та пришвидшення розробки, наприклад для генерації рівня або готовий ШІ для юнітів;

- Відсутність комерційної підтримки - проєкт розробляється як навчальний, тому немає можливості придбати ліцензії на просунуті плагіни чи підключити зовнішніх фахівців.

1.4 Постановка задачі

Метою проекту є створення застосунку в жанрі стратегії з використанням архітектурного патерну MVVM та технології Unity. Ці вимоги не остаточні та можуть бути змінені залежно від обмежень технічних, часових або ресурсних.

Основний для ігрового застосунку жанру стратегії функціонал:

- Управління ресурсами - користувач повинен мати змогу збирати, розподіляти та використовувати ресурси для розвитку, вони мають бути

рушієм та перешкодою гравця;

- Реалізація ігрових механік жанру - будівництво, дослідження технологій, керування юнітами та ведення бойових дій ;

- Інтуїтивно зрозумілий інтерфейс - зручний для взаємодії користувача з грою, що дозволяє швидко приймати рішення та не замислюючись знаходити необхідну дію;

- Збереження прогресу - можливість зберігати та завантажувати стан гри, мажливість мати декілька акаунтів для різних гравців на єдиному пристрої;

- Можливість масштабування - додавання нових рівнів, механік або оновлень без певної переробки коду, додавання підтримки декількох платформ та мережева гра [6, 10].

Також важливою частиною проекту є вимоги до архітектури, а саме використання паттерну MVVM та використання компонентної архітектури Unity.

Патерн MVVM має містити такі компоненти:

- Model, що містить ігрову логіку, дані ресурсів, об’єктів, стан гри тощо;

- View, який реалізується через Unity UI, тобто панелі, кнопки, текст, іконки тощо;

- ViewModel, що є посередником який отримує та обробляє команди від UI, змінює стан моделі та забезпечує оновлення інтерфейсу при змінах.

Використання MVVM дозволяє забезпечити гнучку, розширювану та модульну структуру коду, що полегшує підтримку проекту, тестування та його масштабування.

Архітектура Unity має містити такі компоненти як об’єкт, що є базовим елементом рушія. Він складається з різноманітного набору компонентів і може виступати як в ролі будівлі на землі так і просто контейнером для скрипта, що буде виконуватись під час запуску гри. Вся логіка прив’язана до MonoBehaviour-скриптів, що є основою всіх скриптів, які виконуються у

сцені, тобто вони наслідуються від класу MonoBehaviour. Основною мовою програмування має бути C#, яка є єдиною мовою, що підримує рушій Unity, на даний момент. Також необхідно використовувати безкоштовні ассети. Їх можна взяти з Asset Store або інших відкритих джерел.

2 ВИБІР ТА ОБГРУНТУВАННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ

Наразі розробники, що створюють ігри діляться на 2 типи. Ті, що пишуть свій двигун та ті, що використовують готовий. В моєму випадку я належу до другого типу через раніше описані обмеження. Серез готових двигунів найпопулярнішими вважають Unity, Unreal Engine та Godot. Розглянемо їх детальніше.

2.1 Unity

Unity - це потужний і мегапопулярний багатоплатформний двигун для створення комп'ютерних ігор, а також інтерактивних 2D і 3D-додатків [8]. Він надає розробникам все необхідне для створення ігор. Саме тому його використовують фахівці з усього світу. А багато хто з них вважає Unity номером 1 у світі розробки ігор.

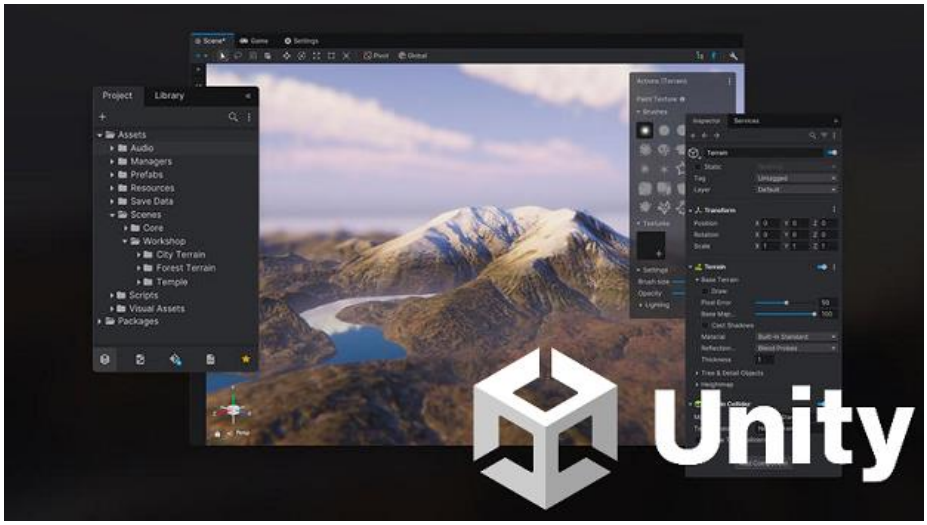

Рисунок 1 - інтерфейс та логотип ігрового рушія Unity

Основні особливості на які зверають увагу при виборі фреймворків або двигунів перед початком або в процесі розробки:

- Багатоплатформеність - за допомогою дигуна розробники можуть

розробляти проекти для багатьох платформ. Наприклад, для PC, мобільних пристроїв ,консолей та віртуальної реальності. Загалом він може запускатися на 25 платформах [8];

- Графіка та анімація - у Unity дуже сильний графічний двигун. Він підтримує рендеринг у реальному часі, освітлення, ефекти, анімацію персонажів та об’єктів. Розробники можуть створювати бависті та реалістичні світи;

- Компонентна система - Unity використовує компонентну систему, що робить розробку більш гнучкою та зручною. Розробники можуть створювати компоненти та легко застосовувати їх до різних об’єктів у сцені. При цьому не потрібно прописувати код;

- Фізика - У движку є вбудована підтримка фізики, що дозволяє створювати реалістичну поведінку об’єктів в ігровому світі. Це включає колізії, гравітацію, сили та багато іншого;

- Скрипт - для роботи, керування поведінкою ігрових об’єктів та написання скриптів використовується мова програмування C# [1, 2]. Це робить розробку більш простою та гнучкою для досвідчених програмістів, а також дозволяє розробникам створювати складну логіку та поведінку ігрових об’єктів. Вбудовані інструменти фізичної симуляції, управління анімаціями та аудіо дозволяють створювати багатий та інтерактивний ігровий досвід;

- Активне співтовариство - у двигуна є велика спільнота розробників. Усі вони активно діляться своїм досвідом, рішення та ремсурсами. Це дозволяє новачкам швидше освоїтися з двигуном і отримати допомогу у разі виникнення проблем.

Принцип роботи в Unity є послідовністю кроків, які розробники роблять при створенні гри. Ось основні з них:

- Створення сцени - сцена це простір, де розміщуються та взаємодіють об'єкти гри. Їх може бути декілька на кожному рівні. У Unity можна створювати та редагувати сцени, визначати компоненти оточення, задавати освітлення, камери та ін.;

- Додавання об’єктів - у Unity об'єкти це основні будівельні блоки гри. Тобто необхідно додавати різні об'єкти до сцени, такі як персонажі, предмети, перешкоди тощо. Також на цьому етапі визначається їх взаємодія та поведінка;

- Скрипти та кодування - вся робота з Unity здійснюється мовою C#. За допомогою скриптів необхідно створювати поведінку об'єктів та керувати ігровою логікою. За допомогою коду можна створювати унікальні функції, визначити умови перемоги або поразки, обробляти введення користувача та багато іншого;

- Тестування та оптимізація - коли гра створена, її необхідно протестувати на різних пристроях та платформах. Тестування допомагає виявити помилки, покращити геймплей та оптимізувати продуктивність гри, щоб вона працювала ефективно на різних пристроях;

- Публікація - гра публікується на вибраних платформах, таких як комп'ютер, мобільні пристрої, ігрові приставки та інше. Unity надає інструменти для експорту та публікації гри на різних платформах, що дозволяє досягти широкої аудиторії гравців.

На цьому двигуні були створені такі ігри як:

- Hollow Knight - атмосферна 2D-метроїдванія від студії Team Cherry, яка вирізняється глибоким світом, складними босами та плавною анімацією. Unity дозволив розробникам реалізувати складну бойову систему та детально пророблений світ без потреби у великій команді;

- Cuphead - гра в стилі ретро-мультиків 1930-х років, створена Studio MDHR. Хоча її візуальний стиль повністю мальований вручну, саме Unity забезпечив ефективне втілення цієї унікальної естетики в інтерактивній формі, зокрема завдяки підтримці 2D-анімації та точній синхронізації аудіо та графіки;

- Subnautica - підводна survival-гра з відкритим світом від студії Unknown Worlds Entertainment. Unity використано для реалізації великого процедурно згенерованого океанічного середовища з різноманітними

біомами, фауною і базовим будівництвом. Рушій добре впорався з великими локаціями та плавною зміною глибини і світла;

- Among us - соціальна мультиплеєрна гра від InnerSloth, що стала вірусною у 2020 році. Завдяки Unity, розробникам вдалося швидко адаптувати гру до мобільних платформ та ПК, реалізувати онлайн-гру з кросплатформеністю, а також легко оновлювати функціонал;

- Rust - спочатку розроблена на Unity студією Facepunch Studios, ця виживальна онлайн-гра у відкритому світі поєднує будівництво, PvP і ресурс-менеджмент. Хоч згодом частина систем була переписана, Unity на ранніх етапах дав можливість швидко прототипувати основні механіки та створити базову багатокористувацьку інфраструктуру.

- Також деякі розробники використовують його на ранніх етапах розробки як дешевий варіант після чого створюють свій двигун та переносять гру на нього.

Unity є одним з основних кандидатів при виборі двигуна для розробки ігор. Завдяки своїй потужності, гнучкості та доступності він популярний і затребуваний у всьому світі. Двигун підходить як для розробників-початківців, так і для досвідчених команд, які бажають створити високоякісні та захоплюючі ігри.

2.2 Unreal Engine

Unreal Engine - це сучасний ігровий рушій, розроблений компанією Epic Games, який широко використовується для створення високоякісного 3D-контенту в ігровій індустрії, кіновиробництві, архітектурі та інших сферах.

Unreal Engine 5 (UE5) став справжнім проривом у світі розробки ігор. Цей потужний інструмент від Epic Games не лише відкриває нові горизонти для розробників, але й ставить перед ними нові виклики.

Рисунок 2 - логотип ігрового рушія Unreal Engine

Після виходу 5-ї версії цього двигуна він сильно преобразився, тому в подальшому буде розлядатися Unreal Engine 5. Цей двигун представляє собою схожі можливості з Unity але має низку революційних функцій, які значно розширюють моливості розробників ігор і візуалізації. Серед них ключовими є Nanite, Lumen, Chaos Physics and Destructions, MetaHuman Creator і World Partition.

Компанія Epic Games створила бібліотеки та фреймворки, що безповоротно змінили розробку ігор, так як алтернативи просто не існує. Ось декілька з них:

- Nanite (Віртуалізована геометрія) - ця технологія дозволяє розробникам використовувати міль’ярди полігонів, створюючи детальні та реалістичні сцени без необхідності оптимізації полігональних моделей;

- Lumen (Динамічне глобальне освітлення) - це нова система глобального освітлення в реальному часі, яка дозволяє створювати динамічне освітлення, що змінюється відповідно до умов сцени. Ця технологія усуває необхідність у тривалому процесі бейкінгу освітлення, роблячи процес створення освітлення швидшим і гнучкішим;

- Chaos Physics and Destruction (Нова система фізики) - це нова система фізики та руйнувань, яка забезпечує реалістичну взаємодію об’єктів у сцені. Вона дозволяє створювати складні симуляції фізичних взаємодій, включаючи руйнування будівель, зіткнення та інші ефекти;

- MetaHuman Creator (Створення реалістичних персонажів) - це інструмент для створення високоякісних 3D-персонажів. Він дозволяє розробникам швидко створювати реалістичних людей з високою деталізацією та анімацією;

- World Partition (Управління великими світами) - це нова система управління великими світами, яка спрощує роботу з великими проектами. Вона автоматично розділяє світ на сектори та завантажує тільки ті з них, які необхідні в даний момент.

Також цей двигун пропонує безліч інструментів та плагінів, які значно розширюють його функціонал і допомагають розробникам створювати високоякісні ігри та інші проекти.

Unreal Engine Marketplace - це платформа, де розробники можуть придбати або завантажити безкоштовні ресурси, такі як 3D-моделі, анімації, текстури, звукові ефекти та інші інструменти. Marketplace значно полегшує життя розробникам, дозволяючи їм зосередитися на ключових аспектах гри, не витрачаючи час на створення кожного елемента з нуля. Також доволі часто роздають безкоштовні набори текстур або плагінів, що в іншому випадку можуть коштувати навіть $100. Але і без роздачі маркетплейс має великий вибір моделей, текстур, та ресурсів для демонстрацій що мають високу якість яку іноді не відрізниш від фотографії.

UE5 підтримує як візуальне програмування за допомогою Blueprints, так і традиційне програмування на C++. Обидва підходи мають свої переваги та можуть використовуватися в залежності від потреб проекту.

Blueprints особливо корисна для швидкого прототипування та створення простих механік. А мова програмування C++ надає більшу гнучкість та продуктивність та використовується для створення складних систем та оптимізації продуктивності гри.

Однією з привабливих особливостей UE5 є його цінова політика. Використання Unreal Engine 5 безкоштовне для будь-якого розробника до моменту, коли ваш проект почне приносити прибуток. Epic Games бере лише

5% роялті з доходів, отриманих від комерційного використання вашої гри, але тільки після досягнення порогу в 1 мільйон доларів.

Це означає, що ви можете експериментувати, створювати прототипи та навіть випустити невеликі ігри, не витрачаючи грошей на ліцензію. Така модель робить UE5 доступним для інді-розробників та малих студій, які мають обмежені бюджети.

Якщо казати про вибір двигуна розробки то цей має великий недолік який стосується лише моєї ситуації. Для комфортної роботи з UE5, Epic Games рекомендує використовувати відеокарти рівня NVIDIA GeForce RTX 2070 або вище. Ці карти забезпечують достатню продуктивність для обробки складних сцен та використання новітніх технологій, таких як Nanite та Lumen, які є одними з ключових особливостей UE5.

Процесор - ще один ключовий елемент для продуктивної роботи з UE5. Оскільки Unreal Engine 5 активно використовує багатопоточність, важливо мати процесор з великою кількістю ядер. Оптимальним вибором будуть процесори рівня AMD Ryzen 7 3700X або Intel Core i7-9700K. Ці процесори забезпечують високу продуктивність та дозволяють ефективно обробляти великі обсяги даних.

З цього можна зробити висновок, що я не можу розробляти на Unreal Engine 5 так як мій ПК не достатньо потужний для цього, але в усьому іншому він перевершує всіх своїх конкурентів на роки вперед.

Порівнюючи його до інших цей двигун має такі переваги:

- Висока якість графіки - завдяки технологіям Nanite та Lumen, UE5 забезпечує фотореалістичну графіку, яка раніше була доступна лише в офлайн-рендерах;

- Кросплатформеність - Unreal Engine підтримує розробку для різних платформ, включаючи Windows, macOS, Linux, iOS, Android, а також консолі PlayStation, Xbox та Nintendo Switch;

- Гнучка система скриптів - рушій підтримує як традиційне програмування на C++, так і візуальне скриптування за допомогою системи

Blueprint, що дозволяє розробникам з різним рівнем досвіду ефективно працювати над проектами;

- Широке застосування - окрім ігрової індустрії, Unreal Engine активно використовується в кіновиробництві, архітектурі, автомобілебудуванні та інших галузях для створення візуалізацій, симуляцій та віртуальних середовищ.

З моменту виходу 5-ї версії цього двигуна було створено багато неймовірно гарних та реалістичних проектів. Таких як:

- Fortnite - один із найвідоміших battle royale-проєктів від Epic Games. З переходом на Unreal Engine 5 гра отримала значне покращення візуальної якості, зокрема завдяки технологіям Nanite та Lumen, що забезпечують реалістичні матеріали, освітлення та масштабованість великого відкритого світу;

- S.T.A.L.K.E.R. 2: Heart of Chornobyl - довгоочікуване продовження легендарної серії від української студії GSC Game World. Unreal Engine 5 дозволив створити деталізований постапокаліптичний світ Зони з динамічним освітленням і реалістичною фізикою, що суттєво підвищує рівень занурення гравця;

- Lords of the Fallen (2023) - рімейк гри у жанрі dark fantasy action-RPG. Завдяки UE5 творці реалізували складну систему освітлення, тіней і частинок, які формують похмуру атмосферу гри, а також плавну анімацію бойових дій з високою деталізацією оточення;

- Black Myth: Wukong - китайська action-RPG на основі народних легенд. Рушій дозволив студії Game Science створити яскравий, кінематографічний світ з фотореалістичною графікою, природною анімацією персонажів та бойовою системою, що використовує сучасні технології фізики й освітлення.

Якби я мав достатньо потужний ПК то працював на цьому рушії без сумнівів. Незважаючи на велику кількість інструментів, що треба вивчити без них набагато складніше зробити щось якісне та гарне. Я як людина що

любить гарне зображення не можу не хвалити цей двигун та його розробників.

2.3 Godot

Godot - це безкоштовний ігровий рушій з відкритим вихідним кодом, який набув значної популярності серед інді-розробників, студентів та невеликих студій. Його розвиток ведеться спільнотою ентузіастів та розробників у межах відкритого проєкту під ліцензією MIT. Godot вирізняється гнучкою архітектурою, підтримкою кількох мов програмування, широкими можливостями для 2D та 3D розробки, а також простим у використанні редактором, що робить його привабливим інструментом як для новачків, так і для досвідчених розробників.

Рисунок 3 - логотип ігрового рушія Godot

Однією з ключових технічних переваг Godot є чітке розділення 2D та 3D підсистем. Це означає, що 2D-рушій не залежить від 3D-компонентів, що дозволяє оптимізувати продуктивність у проектах, які орієнтовані виключно на 2D. Починаючи з версії 4.0, рушій отримав підтримку API Vulkan, що значно покращує візуальні можливості, а також забезпечує підвищену продуктивність для сучасних платформ.

Окрім того, Godot має вбудований фізичний рушій, який дозволяє реалізувати реалістичну симуляцію зіткнень, гравітації, маси та інших властивостей фізичних об'єктів у грі. Для 3D сцен підтримується як власна фізика, так і інтеграція сторонніх бібліотек.

Основні переваги Godot:

- Відкритий код - Рушій розповсюджується за ліцензією MIT, що дозволяє вільно використовувати його у комерційних і некомерційних проєктах, модифікувати та поширювати;

- Підтримка кількох мов програмування - Godot підтримує GDScript (мова, схожа на Python), C#, C++ та інші мови через систему GDNative, що надає розробникам гнучкість у виборі інструментів;

- Інтуїтивно зрозумілий інтерфейс - Візуальний редактор має логічну структуру, підтримує drag-and-drop, автозбереження, дебаг, профілювання та роботу з анімаціями, що пришвидшує розробку;

- Кросплатформеність: Можливість експортувати проєкти на Windows, Linux, macOS, Android, iOS, Web (HTML5), а також окремо - на Steam Deck, Raspberry Pi, консолі (через сторонні рішення);

- Активна спільнота - Хоча розмір спільноти поступається таким гігантам, як Unity або Unreal, вона активно зростає. Створюються численні плагіни, шаблони, туторіали та офіційна документація.

Серед основних технічних можливостей рушія:

- Рендеринг - Після релізу версії 4.0 рушій перейшов на графічне API Vulkan, що дозволяє використовувати сучасні графічні ефекти - тіні, постобробку, PBR (Physically Based Rendering) матеріали;

- Сцени як базовий будівельний блок - Вся логіка в Godot побудована на концепції «сцен» і «вузлів», де кожен об’єкт гри - це вузол певного типу. Це дозволяє легко організовувати ієрархії та повторно використовувати компоненти;

- Фізика - Godot має вбудовані рушії для 2D та 3D фізики з підтримкою колізій, сил, мас, гравітації, а також системи частинок;

- Інтерфейс користувача (UI) - у рушії вбудована система UI-елементів з підтримкою анімації, адаптивного розташування та взаємодії з подіями.

Незважаючи на всі переваги, Godot має ряд обмежень. Насамперед, його 3D-можливості все ще поступаються великим рушіям, зокрема за якістю рендерингу, підтримкою фотореалізму, складних шейдерів та оптимізації великих сцен. Також слід враховувати відносно невелику кількість готових рішень та обмежений маркетплейс ресурсів, що може уповільнити розробку великих або технічно складних проєктів.

Хоча рушій не такий поширений, як Unity чи Unreal, вже існує чимало успішних ігор, створених на ньому:

- Kingdoms of the Dump - класична RPG в стилі 16-бітної епохи;
- The Garden Path - спокійна пригодницька гра з глибокими елементами симуляції;
- Lumencraft - науково-фантастична гра, яка поєднує елементи шутера та виживання;
- Cassette Beasts - покрокова RPG з покемоно-подібним геймплеєм, опублікована на Steam та Xbox.

До речі я вперше чую про такі ігри, всеж не просто так порівнюючи ігрові двигуни про Godot майже не згадують.

Godot - це потужний, гнучкий та безкоштовний рушій, який чудово підходить для розробки 2D та нескладних 3D ігор. Завдяки відкритому коду, зручному інтерфейсу та широкій платформній підтримці, він стає дедалі популярнішим серед незалежних розробників.

2.4. Огляд поширених трирівневих архітектурних патернів

Головним питанням при розробці великого проекту є вибір правильного підходу та структури. Навіть якщо проект створений однією людиною це не значить що в коді не заплутаєшся, та і розширення набагато легше йде коли маєш план дій.

Коли мова заходить про створення ігрового або будь-якого іншого застосунку, дуже важливо з самого початку визначити, якою буде структура проєкту. Від цього залежить не тільки зручність у написанні коду, а й наскільки легко буде підтримувати, оновлювати або масштабувати продукт у майбутньому. Особливо це важливо, якщо планується командна робота або довготривалий розвиток гри. Тут і з’являється необхідність в патернах.

Найпопулярніші серед них - це MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel). Вони вже роками застосовуються в реальних проєктах і допомагають уникнути плутанини в коді, де логіка, інтерфейс і дані змішані між собою.

MVC - це, напевно, найвідоміший із цих підходів. Його суть у тому, щоб розділити код на три частини: модель (тобто дані, наприклад, кількість очок, ресурси чи позиції об’єктів), вигляд (інтерфейс, що бачить користувач), і контролер (логіка, яка з'єднує ці дві частини). Це досить простий і зрозумілий підхід, і його зручно використовувати у веб-розробці, але в ігрових проєктах, особливо коли інтерфейс складний або динамічний, він не завжди підходить.

MVP - це щось середнє між MVC та MVVM. Він передбачає наявність окремого об'єкта, який називається Presenter. Саме він отримує дані від моделі та "керує" тим, що має показати інтерфейс. Це вже трохи зручніше, бо інтерфейс не залежить від даних напряму, але все одно є певні обмеження, особливо якщо потрібно реалізувати двосторонній зв’язок.

MVVM - це саме той патерн, який найчастіше використовують у сучасних UI-орієнтованих застосунках [6, 7]. Особливо він став популярним у середовищах, які мають можливість зв’язувати змінні у коді з елементами інтерфейсу, як наприклад, у WPF, Xamarin або навіть у Unity (з певними підходами). Основна ідея полягає в тому, що між View (тобто візуальною частиною) і Model (логікою та даними) додається ще один рівень - ViewModel. Це посередник, який відповідає за передачу даних між логікою гри та її інтерфейсом. І головне - він дозволяє підтримувати інтерфейс

актуальним автоматично, без написання додаткової логіки оновлення.

У моєму випадку, коли я працюю над грою в жанрі стратегії, структура проєкту відіграє ключову роль. Тут багато даних, з якими потрібно працювати: ресурси, будівлі, обробка подій тощо. Якщо все це тримати в одному файлі - з часом буде важко розібратись навіть самому. Тому я вирішив, що MVVM - це саме той патерн, який дозволить мені зберегти порядок у коді та водночас гнучко керувати як логікою, так і інтерфейсом.

Патерн MVVM (Model-View-ViewModel) - це один із сучасних підходів до архітектури програмного забезпечення, який особливо популярний для розробки додатків з графічним інтерфейсом. Його суть полягає в розділенні логіки програми на три основні частини:

- Model - це, по суті, серце програми, де зберігаються дані і бізнес-логіка. У грі це можуть бути параметри гравця, характеристики юнітів, ресурси, стани об’єктів тощо. Model не знає нічого про те, як саме ці дані будуть показані користувачеві;
- View - це все, що бачить користувач на екрані. Кнопки, панелі, ігрові елементи інтерфейсу, меню. Вона відповідає лише за відображення і не містить логіки обробки даних;
- ViewModel - це проміжний рівень, який виконує роль “перекладача” між Model і View. Саме ViewModel відповідає за те, щоб підготувати дані у вигляді, зручному для відображення, а також приймати команди від користувача і передавати їх у Model.

Основна перевага MVVM у тому, що View та Model не залежать безпосередньо одна від одної - вони контактують через ViewModel. Це значно полегшує підтримку коду, дозволяє змінювати інтерфейс без переробки логіки і навпаки. Такий підхід також дуже допомагає, коли працюєш у команді, дизайнери можуть паралельно працювати над інтерфейсом, а програмісти над логікою.

Ще одна важлива риса MVVM це підтримка двонаправленого зв’язку (data binding). Це означає, що зміни в даних автоматично відображаються у

View без необхідності писати додатковий код для оновлення елементів інтерфейсу. І навпаки взаємодія користувача, наприклад натискання кнопок чи введення тексту, безпосередньо впливає на Model через ViewModel. Така автоматизація значно скорочує час розробки і знижує кількість помилок.

2.5. Особливості реалізації патерну MVVM в Unity проектах

Перш за все, Unity базується на компонентно-орієнтованій архітектурі, де логіка гри реалізується через скрипти, прикріплені до об’єктів сцени. Зазвичай, без патернів коду в проєкті може швидко накопичуватись хаос. Використання MVVM допомагає розділити сфери відповідальності та полегшити цей процес.

Model у Unity може бути реалізований як клас або набір класів, які зберігають основні дані гри - наприклад, стан юнітів, кількість ресурсів, рівень розвитку. Ці дані не залежать від того, як вони відображаються або взаємодіють з гравцем.

View це всі компоненти інтерфейсу користувача, які Unity реалізує через UI-систему: кнопки, панелі, текстові поля, ігрові об’єкти, що відображають інформацію. У цьому шарі немає бізнес-логіки, тут тільки візуальне представлення даних.

ViewModel це проміжний шар, який зв’язує Model і View. Він відповідає за обробку подій користувача, оновлення даних у Model і підготовку даних для відображення у View. У Unity ViewModel зазвичай реалізується через скрипти, які мають властивості і методи, що зв’язують UI з внутрішньою логікою.

Важливо, що MVVM у Unity часто реалізується з використанням data binding - механізму, що автоматично оновлює інтерфейс при зміні даних і навпаки. Хоча Unity не має вбудованої підтримки двонаправленого binding, існують готові бібліотеки (наприклад, UniRx, MVVM Toolkit), які спрощують це завдання [10].

Використання MVVM у Unity дає такі переваги:

- Чітке розділення коду - зручно працювати над логікою і UI окремо, що прискорює розробку і полегшує внесення змін;

- Покращена підтримка і масштабованість - проєкт легше підтримувати, особливо при збільшенні обсягу коду чи при роботі команди;

- Зручність тестування - Model і ViewModel можна тестувати незалежно від UI, що підвищує якість коду;

- Гнучкість - зміни в UI не впливають на внутрішню логіку і навпаки, що важливо для адаптації гри під різні платформи або оновлення.

Таким чином, патерн MVVM ідеально підходить для розробки стратегії в Unity, де є багато взаємопов’язаних елементів: ресурси, юніти, події, інтерфейс. Такий підхід допомагає утримувати проєкт у впорядкованому стані і полегшує його подальше розширення.

3 РЕАЛІЗАЦІЯ ІГРОВОГО ПРОЕКТУ

3.1 Загальна структура проекту

Проєкт умовно можна поділити на логіку та ресурси. У пункті з логікою знаходяться скрипти, що уособлюють собою основну логічну структуру гри, її правила та обмеження. До ресурсів можна віднести такі речі як сцена, моделі, текстури, бібліотеки та різного роду налаштування.

Логіка - це усі файли, що відносяться до ігрової логіки. Вони розташовані в папці Scripts.

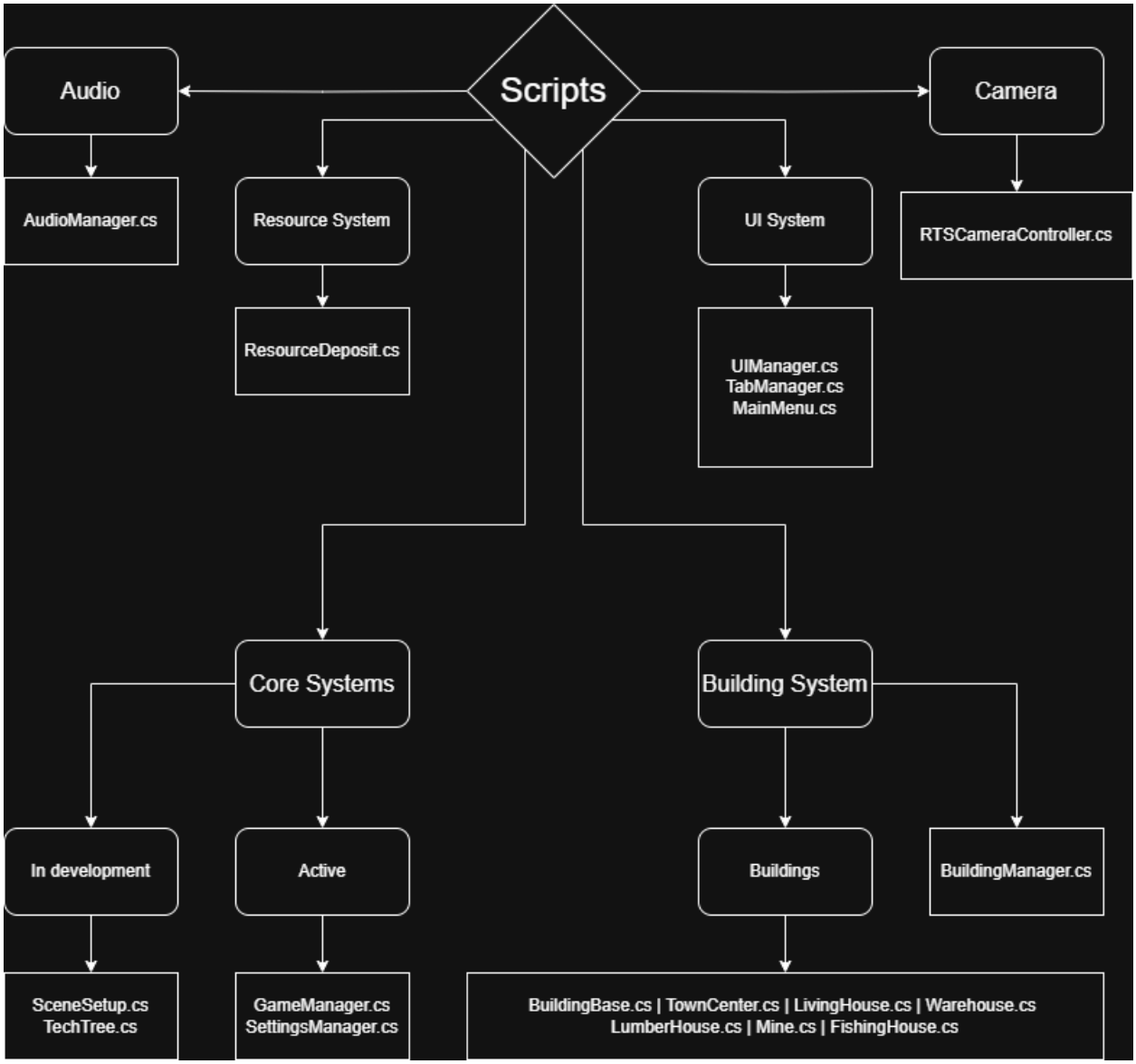

Рисунок 4 - Структура логіки в проєкті

Дана схема не відображає реальну структуру файлів. Насправді усі файли знаходяться на одному рівні, окрім папки Buildings в якій розташовані скрипти для кожного типу будівлі та базової будівлі.

Детальніше про скрипти:

- GameManager - головний менеджер гри, що упрвляє ресурсами гри (їжа, дерево, золото, камінь, населення), контролює ємність цих ресурсів, управляє ростом населення, має систему подій для сповіщення про зміни ресурсів та контролює паузу гри [4, 9];

- BuildingBase - базовий клас для всіх будівель. Він контролює розміщення будівель з валідацією, підсвічує будівлю залежно від моливості її розміщення в даному місці,реалізує систему рівнів будівель та вартість розміщення та апгрейду, реалізує систему виробництва ресурсів та зміну моделі при піднятті її рівня(наразі не працює). Цей клас є вабстрактним і його настідують такі типи будівель як міський центр, житловий будинок, рибальський будинок, лісопильня, шахта та склад;

- UIManager - менеджер інтерфейсу, що управляє усіма елементами UI, відображає інформацію про ресурси, керує меню будівництва та відображає інформацію про будівлі;

- RTSCameraController - контроллер камери, що реалізує камеру в стилі RTS(Real-Time Strategy). Він також керує мастабуванням, переміщенням та обертанням камери;

- SettingsManager - менеджер налаштувань, що зберігає налаштування гри, управляє графічними налаштуваннями та зберігає налаштування звуку;

- ResourceDeposit - Скрипт, що описує логіку роботи ресурсних родовищ, визначає кулькість ресурсів та контролює відновлення ресурсів;

- AudioManager - менеджер звуку, що управляє звуковими ефектами, контролює музику та налаштовує гучність;

- TabManager - менеджер вкладок, що управляє вкладками у UI. Надає моливість перемикання між різними меню налаштувань. Наприклад між налаштуванням графіки та звуку;

- MainMenu - головне меню гри, яке зустрічає гравця на початку гри. В ньому можна перейти до самої гри або зайти до налаштувань чи вийти з гри;

- LightingManager - менеджер освітлення, що контролює денний/нічний цикл (на випадок якщо в майбутньому в грі буде ніч), цправляє освітленням сцени та налаштовує атмосферні ефекти.

Ресурси - простіше кажучи це всі інші елементи гри, що не можна прив’язати до скриптів але вони не менш важливі, серед них можна виділити такі:

- Сцени - тут описано сцени та їх наповнення, тобто ігров об’єкти та менеджери;

- Моделі - тут лежаль створені 3Д об’єкти що не пройшли обробки для використання у сцені;

- Матеріали - тут лежать зображення текстур та різні кольори, що використовуються у грі;

- Префаби - обработані моделі, на які наклали скріпти та текстури, що можуть бути використані у грі;

- Аудіо - тут розташовано музичні треки

- Іконки - тут розташовані іконки та малі зображення які можна використати в UI;

- Налаштування - системні налаштування, які Unity створює автоматично, вони відповідають за освітлення, звук та пресети налаштувань для різних пристроїв;

- TextMesh Pro - фрейморк, що використовується як страндартний в Unity для відображення тексту;

- StarterAssets - фреймворк з базовими ресурсами, моделями та скріптами для 3д гри;

- TerrainSampleAssets - фреймворк з ресурсами для створення оточення;

- Tree9 - текстури та моделі дерева;

- Stylized Wood Textures - текстури для дерев’яних стін та підлоги;

- Stylize Roof Texture - текстури для даху;

3.2 Взаємодія View - ViewModel - Model

Дана взаємодія була реалізована в скриптах UIManager та GameManager. Так як проєкт невеликий то цей паттер було виконано не за всіма правила в угоду іншим паттернам та правилам. У файлі GameManager в цьому уривку описано змінні та створення підписки на зміни:

```
public event Action OnResourcesChanged;

// Properties with events
public int Food
{
    get => _food;
    set
    {
        _food = value;
        OnResourcesChanged?.Invoke();
    }
}

public int Wood
{
    get => _wood;
    set
    {
        _wood = value;
        OnResourcesChanged?.Invoke();
    }
}

public int Gold
{
    get => _gold;
    set
    {
        _gold = value;
        OnResourcesChanged?.Invoke();
    }
}

public int Stone
{
    get => _stone;
    set
    {
        _stone = value;
```

```
            OnResourcesChanged?.Invoke();
        }
    }

    public int Population
    {
        get => _population;
        set
        {
            _population = Mathf.Clamp(value, 0,
_populationCapacity);
            OnResourcesChanged?.Invoke();
        }
    }
```

А у файлі UIManager на ці зміни підписуються, в результаті чого при зміні ресурсів буть-де це автоматично оновить відображення відповідних полів з ресурсами в UI.

3.3 Використання Design Patterns у проекті

Патерн Singleton гарантує, що клас має лише один екземпляр, і надає глобальну точку доступу до нього. Часто використовується для менеджерів, які мають зберігати стан гри або керувати ресурсами [9].

Код:

```
public class GameManager : MonoBehaviour
{
    public static GameManager Instance { get; private set; }

    void Awake()
    {
        if (Instance != null && Instance != this)
        {
            Destroy(this.gameObject);
        }
        else
        {
            Instance = this;
            DontDestroyOnLoad(this.gameObject);
        }
    }
    // ... інший код ...
}
```

У моєму проекті GameManager, AudioManager, UIManager - це класи,

які мають бути лише в одному екземплярі. Singleton гарантує, що не буде дублювання логіки чи стану, а доступ до менеджера відкритий для всього проекту через `GameManager.Instance`

Template Method визначає скелет алгоритму в базовому класі, дозволяючи підкласам перевизначати окремі кроки алгоритму без зміни його структури. Зручно для ієрархії будівель, юнітів тощо [9].

Код:

```
public class Warehouse : BuildingBase
{
    protected override void Awake()
    {
        // Специфічна ініціалізація для складу
    }

    protected override void Start()
    {
        // Дії при старті для складу
    }

    protected override void OnPlacementConfirmed()
    {
        // Дії після підтвердження розміщення складу
    }

    protected override void ApplyUpgrade()
    {
        // Логіка покращення складу
    }

    public override void Demolish()
    {
        // Логіка знесення складу
    }
}
```

В моєму проекті існує клас BuildingBase та його похідні (LumberHouse, Warehouse тощо). Warehouse наслідує базовий клас BuildingBase і перевизначає ключові методи. Це класичний Template Method - базовий клас задає загальний процес будівництва, а деталі реалізуються у підкласах для кожної конкретної будівлі. Це спрощує додавання нових типів будівель і гарантує, що всі вони будуються за єдиною схемою та мають однакові базові параметри та методи.

3.4 Ігрова механіка проекту

Ця гра надихалась такими проєктами як Anno 1800, Cossacks, Age of Empires та 0 A.D. Єдина відмінність моєї гри лише в тому, що в ній немає прямої взаємодії гравця з юнітами та відсутні будь-які війни або сутички.

Так в цій грі можна виділити такі механіки як:

- Система ресурсів - існують такі ресурси як їжа, дерево, камінь та золото. Вони в свою чергу можуть витрачатись, зберігатись та накопичуватись. Вони мають обмежену кількість зберігання, але можуть бути розширені за допомогою нового складу. Виробництвом ресурсів займаються відповідні будівлі. Наприклад шахта може добувати камінь або золото а рибальська хижка ловить рибу;

- Система населення - ця система основана на постійному розвитку та є основною метою гри, при дотриманні всіх необхідних умов населення буде зростати та навпаки при недотриманні зменшуватись(наразі тільки збільшується). Існує 2 способи збільшити населення - перший і найочевидніший це побудувати житловий будинок та другий це дотримуватись умов для розмноження, наразі це лише небхідність достатку в їжі, але треба зауважити, що чим більше населення тим більше воно їсть тому потрібно постійно збільшувати виробництво. А щоб виробництво їжі збільшити потрібні інші ресурси, що потребує будівництво шахт та лісопилок. Таким чином створюється залежність усіх механік між собою;

- Система будівництва - у грі існують такі будівлі як TownCenter, LivingHouse, FishingHouse, LumberHouse, Mine та Warehouse. Кожна має свою особливість та мету. Міський центр представляє з себе маленьке мість що може розміщувати населення, добувати дерево та вирощувати їжу, але має високу вартість будівництва. Житловий будинок вміщує в собі людей тим самим збільшуючи вмістимість населення. Рибальський будинок має на меті лише рибалити а тому розміщується біля водойми та добуває їжу. Лісопилка добуває деревину залежно від розташування, таким чином якщо

поруч є дерева то швидкість видобутку збільшується. Шахта є особюливою так як може бути встановленна лише на родовищі та залежно від типу родовища може добувати або камінь або золото. Склад слугує лише для зберігання всіх ресурсів тому просто збільшує максимальну вмістистимість. Це може слугувати обмеженням для будівництва нових або вдоскналення старих будівель;

- Система оновлень будівель - будівлі можуть буди оновленні до більших або досконаліших версій за додаткову платню. Таким чином можна зекономити простір та мати змогу розвиватись після того як воно закінчиться чи для естетичного вигляду, який можна поєднати з практичністю;
- Система розміщень будівель - будівлі як і в реальному житті мають обмеження щодо місця їх будівництва. Таким чином рибацький будинок має буди біля води, лісопилка біля лісу та шахта біля родовища;
- Система виробництва - деякі будівлі мають змогу добувати ресурси, що необхідно для подальшого розвитку чи просто існування.

3.5 Організація взаємодії з користувачем

На початку гри нас зустрічає головне меню.В ньому можна перейти до самої гри або до налаштувань. Фонове зображення було згенероване за допомогою ChatGPT.

Рисунок 5 - Головне меню гри

В налаштуваннях є вкладка графіки та звуку, які можна перемикати між собою.

Рисунок 6 - Вікно налаштувань гри, вкладка з графічними налаштуваннями

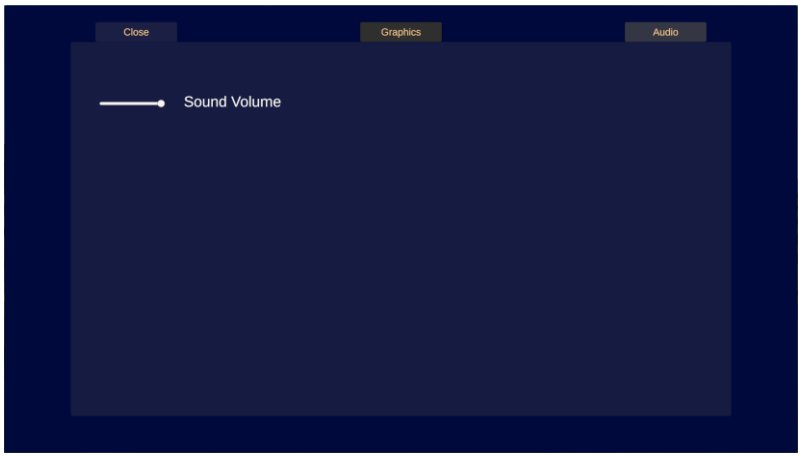

Рисунок 7 - Вікно налаштувань гри, вкладка з налаштуваннями звуку

Наразі налаштування графіки працюють доволі обмеженно. Рівень якості є лише один, а тому на швидкодію впливає лише роздільна здатність та якість тіней.

У самій грі ми маємо ігровий світ та елементи інтерфейсу з інформацією.

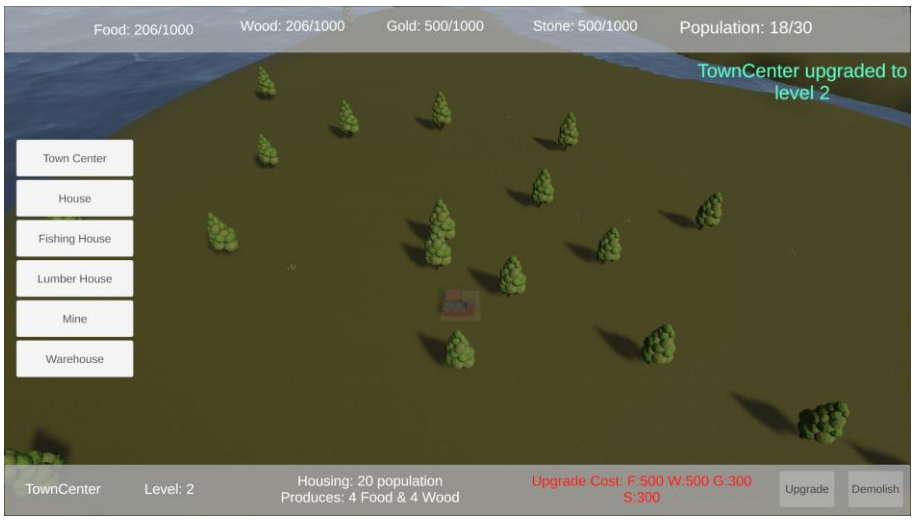

Рисунок 8 - Зображення геймплею

Нагорі розташовано інтерфейс ресурсів, він відображає поточні та об’єм ресурсів у реальному часі. Під ним блакитним кольором з’являється інтерфейс підказок, що надає додаткову інформацію для гравця. З лівого краю розташовано меню будівництва, що надає можливість гравцю будувати та розвиватись. Знизу розташована інформаційна панель, що надає інформацію щодо вибраної будівлі. Коли жодної будівлі не розташовано ця панель схована.

Також в грі присутнє меню паузи на якому розташовано кнопки продовження, налаштувань та виходу.

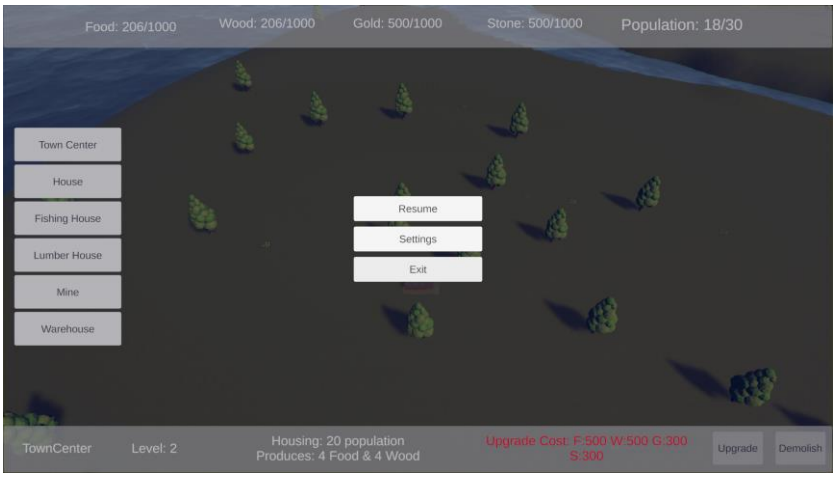

Рисунок 9 - Меню паузи

Ці кнопки мають ту ж саму функціональність що і головне меню. Під час паузи усі процеси всередині гри припиняються та фон стає прозоро-

синім.

У гри немає перемоги або програшу так само як і кінця. Також збереження теж немає, його можна додати потім, але це не точно.

В наш час ігри мають один і той самий шаблон. Моєю ціллю було створити гру яка буде мене радувати. Іноді не потрібно мати на меті виграти або намагатись не програти. В наш час ігрові студії або окремі розробники намагаються розтягнути час проведений у грі по максимуму не зважаючи на користувачів. Моя гра має на меті дати дозвілля на один день, одни вечір або навіть на 1 годину. Головне не те що ти виграв або витратив на гру 100 годин. Головне задоволення якого ти від неї очікуєш. Мені вона приносить додаткове задоволення так як це моя гра - те ,що я створив своїми руками.

3.6 Тестування ігрового проекту

Існує багато різних способів протестувати проєкт. В різних сферах програмування вони мають схожі типи. Маючи досвід створення тестів для web-додатку можна дійти висновку, що тестування для різних програм та мов промування є єдиним для ручного тестування і залежить лише від фреймворку для автоматичного тестування [20]. Серед способів тестування можна виділити такі:

- Unit-тести - це перевірка окремих елементів логіки гри таких як методів або класів без прив’язки до Unity-сцени. Воно дозволяє ізолювати логіку (наприклад, систему будівництва або підрахунок ресурсів) і переконатись, що вона працює правильно при різних вхідних даних;
- Інтеграційні тести - це тестування в контексті Unity-сцени, коли вже є GameObject, компоненти та взаємодія між ними. Проводиться в середовищі Play Mode Unity. Воно дає змогу перевірити, як логіка працює разом з візуальною частиною, фізикою, анімаціями тощо;
- Ручне тестування в Editor (лише для Unity) - це найпростіший і найбільш інтуїтивний спосіб перевірити, як працює гра шляхом запуску її

вручну в редакторі або збірці. Тестувальник або розробник грає в гру, намагається викликати помилки, натискає на кнопки, дивиться на реакції системи;

- Логування і дебаг - цей метод включає в себе використання логів Debug.Log, інструментів Unity Profiler, Frame Debugger і Visual Debug (напр. лінії, текстові підказки). Він дозволяє зрозуміти, що відбувається в грі на рівні внутрішніх значень змінних та станів;

- Автоматизовані сценарії - це створення скриптів або ботів, які автоматично відтворюють поведінку гравця таких як натискання кнопок, рухи, побудову бази тощо. Потрібно для перевірки складних сценаріїв гри без участі людини;

- Бета-тестування- це тестування реальними гравцями, які надають фідбек щодо геймплею, знаходять баги або повідомляють про помилки, які не виявлені в інструментальних тестах. Часто супроводжується логуванням або аналітикою.

Наразі тестів до проєкту немає. Усе тестування функціоналу проводилися за необхідності. Якщо після написання скрипту гра працювала не так як очікувалося то проводилися заходи по додаванню логів до проблемних моментів коду. Також до тестування були залучені друзі та родичі. З цих даних було знайдено та виправлено декілька багів, це можна назвати бета-тестом.

Однією з головних проблем було відстеження миші при роміщенні будівлі. Проблема заключалась у тому що деякі елементи інтерфейсу не правильно працювали разом із ігровою сценою. Наприклад при виборі будівлі гравець мав натиснути на місце куди її треба поставити, але що якщо він передумав і захотів вибрати іншу будівлю. В такому разі натискаючи на кнопку іншої будівлі на інтерфейсі поточна будівля ставилась на місце під тією кнопкою. Ця проблема була виправлена додаванням умови, що якщо було натиснуто на кнопку вибору будівлі то треба прервати поточне розміщення будівлі та почати нове залежно від натиснутої кнопки. Це важко

назвати помилкою так як існує спосів вийти з режиму розміщення, а тому це більше відноситься до інтуітивності інтерфейсу.

Також була проблема з пошуком води поблизу, так як вода у грі виконана як великий квадрат, що розташований нижче рівня землі та в місцях де земля нижча її перекриває вода. Через це було декілька проблем бо незважаючи на те що будівля була на землі її не можна було встановити бо на кілька одиниць нижче знаходиться вода. Ця проблема вирішилась таким чином, що при перевірці функція брала найпершу згору поверхню.

Всі ці помилки було вирішено за допомогою дебагу та логів. Через брак часу тести писати неможливо. Також цей проєкт хоч і можна вважати завершеним йому ще можна багато чого додати. Про це детальніше написано у відповідному розділі.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Системні вимоги

Для корректної роботи гри з комфортною кількістю кадрів на секунду необхідно щоб пристій на якому буде працювати дана гра мала вказані характеристики або кращі [3].

Мінімальні:

- Операційна система: Windows 10 (64-bit) або Linux (Ubuntu 20.04 / аналогічні)
- Процесор: Intel Core i3-6100 або AMD Ryzen 3 3200U
- Оперативна пам’ять: 4 GB
- Графіка: Intel HD Graphics 620 або Radeon Vega 3
- Вільне місце на диску: 200 MB
- DirectX: версія 11

Рекомендовані:

- Операційна система: Windows 10 (64-bit) або Linux (Ubuntu 20.04 / аналогічні)
- Процесор: AMD Ryzen 7 5700U
- Оперативна пам’ять: 8 GB
- Графіка: Radeon Vega 8 або Nvidia GTX 1050
- Вільне місце на диску: 200 MB
- DirectX: версія 11 або вище

4.2 Особливості інсталяції

Завантажити архів з грою та розпакувати його у будь-яке зручне місце та запустити застосунок з назвою “Strategy_v2” (він також може мати

розширення exe в назві). Також можна створити посилання на цей файл для швидшого доступу.

4.3 Опис основного ігрового процесу

Гравець бере на себе роль мера на острові, де головною метою є будівництво поселення, збір ресурсів та заселення острова максимальною кількістю населення.

Гра відбувається в режимі реального часу, де користувач має вирішувати, що небхідно для виживання, будувати споруди та добувати їжу разом з іншими корисними матеріалами.

Кожна будівля має унікальну механіку. Що це за механіки гравець має самостійно з’ясувати, але це все доволі логічно тому можна здогадатись.

Щоб почати гру на головному меню (Рисунок 5) потрібно натиснути на кнопку Start. Перед гравцем з’явиться ігровий світ та інтерфейс (Рисунок 8), для взаємодії з ігровим світом в пункті 4.4 описано базове управління. На початку гравцеві доступна деяка кількість ресурсів. Це відображається в верхній частині екрану.

Щоб почати розвиток міста необхідно побудувати будівлю. Для цього в лівій частині екрану потрібно натиснути на кнопку з необхідним типом будівлі та перемістивши мишу на потрібне місце написнути ЛКМ для встановлення.

Також при встановленні треба зважати на колір будівлі. Якщо він зелений то будівництво можливе, якщо ж червоний то треба пошукати інше місце. Якщо під час встановлення гравець передумав то він може відмінити встановлення натиснувши ПКМ, це діє лише поки будівля ще не встановленна.

Натиснувши на раніше встановленну будівлю ЛКМ можна побачити детальну інформацію про неї в нижній частині екрану. На цій панелі також можна прокачати або зруйнувати будівлю. Прокачуючи будівлю її

характеристики покращуються. Наприклад житловий будинок зможе вмістити більшу кількість людей.

Якщо небхідно поставити паузу то треба натиснути на Esc. Відкриється меню паузи (Рисунок 9) в якому можна продовжити, вийти або змінити налаштування. Відкривши меню налаштувань (Рисунок 6) можна змінити налаштування графіки. Якщо ж необхідно змінити нашатування звуку то потрібно натиснути на відповідну кнопку в меню налаштувань. Меню налаштувань звуку (Рисунок 7) має слайдер за допомогою якого можна змінити гучність гри, а якщо точніше то фонової музики, так як інших звуків у грі немає.

4.4 Керування

Керування в грі здійснюється за допомогою клавіатури та миші.

Основні елементи управління:

- W, A, S, D - переміщення камери по карті, також можна підвести мишу до краю екрану щоб камера почала рухатись у відповідному напрямку;
- Ліва кнопка миші (ЛКМ) - вибір будівлі або її встановлення
- Права кнопка миші (ПКМ) - відміна встановлення будівлі в режимі редагування
- Esc - відкриття/закриття меню паузи
- Q, E - горизонтальний поворот камери
- Колесо миші - масшабування камери

ВИСНОВКИ

У ході виконання даного проєкту було розроблено гру в фанрі стратегії в реальному часі, яка демонструє ключові аспекти проєктування та реалізації сучасного програмного забезпечення в галузі ігрової індустрії. В роботі виконано аналіз предметної області, обгрунтовано вибір інструментів розробки та архітектурного паттерну MVVM, що забезпечує гнучкість та масштабованість проєкту [6, 7].

Функціональність гри охоплює базові механіки керування, побудову структуру та збір ресурсів. У процесі ручного тестування було перевірено коректність реалізації основних сценаріїв гри, виявлено й оброблено окремі випадки збоїв, що дозволило покращити стабільність проєкту.

Було вивчено такі програми як Unity- ігровий рушій, BlockBench - 3д редактор для створення моделей, Cursor - це текстовий редактор на основі VS Code для написання коду та автоматичної генерації коментарів, також він використовувався як інструмент для автоматичного дописування коду.

Загалом, результати роботи підтверджують доцільність використання сучасних архітектурних підходів та інструментів у розробці ігор, а також демонструють практичну реалізацію теоретичних знань, отриманих у процесі навчання.

Даної реалізації, на мою думку, не достатньо для випуску її на торгову платформу. Деякі речі як от оточення, звуки та баланс не дають їй змоги бути завершеною. Втім для навчального проєкту це достатній рівень.

Однак, кажучи про подальший розвиток, для гри такого жанру існує багато варіантів для доповнень. Перше що приходить на думку це нові типи будівель, нові локації та додаткові механіки розвитку як от “дерево технологій”. В проєкті є скрипт для реалізації дерева розвитку, але він не дороблений та не доданий до сцени. Ця механіка, що присутня в більшості ігор такого жанру і майже вважається необхідністю особливо добре підійде

для подальшого розвитку проєкту.

Також можна згадати що в моїй грі немає юнітів або людей. Їх можна додати як інтерактивних персонажів або просто як елемент декорації. І навіть якщо немає функції збереження можна сворити кілька рівнів з різними рельєфами та предвстановленими будівлями.

Також можна додати сюжет. Його можна реалізувати таким самим чином як в грі Козаки. Невеличкі історії на різних островах. Цей підхід дуже добре вписується в мій концепт гри.

У сучсних іграх нормою є наявність мультиплеєра в тому чи іншому його вигляді. Деякі грають як суперники, інші ж як союзники. Іноді мультиплеєр реалізовано як у Elden Ring. Доволі обмежено та має декілька видів. З одного боку можна дати змогу гравцям взаємодіяти в реальному часі або дати можливість іншим гравцям залиши послання наступним. Також деякі змагаються за допомогою досягнень.

Якщо казати про мультиплеєр в моїй грі я бачу це як конкурс в якому люди потрапляють до островів інших гравців та порівнюють їх за різними критеріями. Наприклад хто створив найгарніше або найпрактичніше місто. Можна також створити режим змагань в якому гравці на час мають відтворити якесь реальне місто чи село. Зважаючи на особливості мультиплеєра додавання ШІ як опонента не має сенсу так як важливим його аспектом є почуття та фантазія.

За останні кілька років мультиплатформеність стає майже необхідністю у зв’язку з популярністю SteamOS та Linux у загалом. За даними платформи Steam в цьому році кількість гравців на linux значно зросла. Наразі з використанням Proton кількість ігор що можна запустити на linux досягла 80%. Через що багато хто переходить до linux як до основної операційної системи. Я і сам маю SteamOS та Windows на двох моїх пристроях. На даний час гра працює на SteamOS та Windows 11 без відмінностей, але це може змінитись після додавання нових технологій рендерингу чи масштабування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Глінка, С. М. Основи програмування мовою C# / С. М. Глінка. - Київ : Наукова думка, 2020. - 412 с.

2. Френк Д. Луна. Програмування ігор з використанням C# і Unity / Ф. Д. Луна. - Київ : Видавництво "Ліра-К", 2021. - 380 с.

3. Кузьмінський, О. Ю. Принципи об’єктно-орієнтованого програмування / О. Ю. Кузьмінський. - Львів : Видавництво ЛНУ, 2018. - 252 с.

4. Прокопенко, О. І. Архітектура програмного забезпечення / О. І. Прокопенко. - Харків : ХНУРЕ, 2022. - 198 с.

5. Microsoft. The C# Programming Language. 4th Edition / Anders Hejlsberg, Scott Wiltamuth, Peter Golde. - Pearson Education, 2010. - 850 p.

6. MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF / Ryan Vice. - Packt Publishing, 2012. - 330 p.

7. Freeman, A. Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4 / A. Freeman. - Apress, 2010. - 1080 p.

8. Unity in Action: Multiplatform Game Development in C# with Unity / Joe Hocking. - Manning Publications, 2018. - 360 p.

9. Game Programming Patterns / Robert Nystrom. - Genever Benning, 2014. - 354 p.

10. Souvik Paul. MVVM Design Pattern in Unity / S. Paul. - Independently published, 2021. - 168 p.

11. Jesse Freeman. Unity UI Essentials / J. Freeman. - Packt Publishing, 2014. - 188 p.

12. Jon Duckett. HTML and CSS: Design and Build Websites / J. Duckett. - Wiley, 2011. - 490 p.

13. Bob Nystrom. Crafting Interpreters / B. Nystrom. - G. Benning, 2021. - 566 p.

14. Marschner, S. Fundamentals of Computer Graphics / S. Marschner, P. Shirley. - CRC Press, 2020. - 1000 p.

15. David Eberly. Game Physics / D. Eberly. - CRC Press, 2010. - 752 p.

16. Eric Lengyel. Mathematics for 3D Game Programming and Computer Graphics / E. Lengyel. - Cengage Learning, 2012. - 576 p.

17. Jesse Schell. The Art of Game Design: A Book of Lenses / J. Schell. - CRC Press, 2019. - 600 p.

18. Ernest Adams, Andrew Rollings. Fundamentals of Game Design / E. Adams. - New Riders, 2014. - 496 p.

19. Tracy Fullerton. Game Design Workshop: A Playcentric Approach to Creating Innovative Games / T. Fullerton. - CRC Press, 2018. - 535 p.

20. Steve Rabin. Introduction to Game Development / S. Rabin. - Cengage Learning, 2010. - 1040 p.