ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтернет гра на основі технологій Unity»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Дмитро Буркаль
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Дмитро Буркаль
група КНДМ-63

Керівник: Андрій Чичур
науковий ступінь, вчене звання к.т.н. доцент

Рецензент:
науковий ступінь, вчене звання
_____
(Ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Буркалю Дмитру Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтернет гра на основі технологій Unity»

керівник кваліфікаційної роботи Андрій Чичур доктор філософії,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320

2. Строк подання кваліфікаційної роботи «15» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з питань, пов'язаних з темою роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1. Реалізація інтерактивної карти України з поділом на регіони
   4.2. Використання бібліотеки Mirror для багатокористувацької взаємодії та синхронізації даних.
   4.3. Розробка механіки гри: система питань, захоплення регіонів та підрахунок очок.

5. Перелік графічного матеріалу: *презентація*
 5.1 Інтерактивна карта України з виділеними регіонами.
 5.2 Приклад інтерфейсу з питанням та варіантами відповідей.
 5.3 Схема архітектури клієнт-серверної моделі з використанням Mirror.
6. Дата видачі завдання «16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.09-27.09.24	
2	Аналіз обраної теми та розробка концепції гри	30.09-11.10.24	
3	Розробка основної механіки гри	14.10-25.10.24	
4	Впровадження бібліотеки Mirror для багатокористувацького режиму.	28.10-08.11.24	
5	Тестування гри та оптимізація механік	11.11-22.12.24	
6	Виправлення помилок	25.11-29.11.24	
7	Оформлення роботи: вступ, висновки, реферат	02.12-06.12.24	
8	Розробка демонстраційних матеріалів	09.12-13.12.24	

Здобувач вищої освіти _____ Дмитро БУРКАЛЬ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій ЧИЧУР
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 53 стор., 16 рис., 15 джерел.

Наукове завдання – розробка багатокористувацької інтернет-гри з інтерактивною картою України на основі Unity та бібліотеки Mirror.

Мета роботи – створення прототипу освітньої гри, яка поєднує елементи стратегій і навчання з можливістю багатокористувацької взаємодії.

Об'єкт дослідження – процес розробки багатокористувацьких ігор з інтерактивним інтерфейсом.

Предмет дослідження – технології створення ігор із використанням Unity, C#, і бібліотеки Mirror.

Методи дослідження – аналіз сучасних технологій розробки ігор, застосування клієнт-серверної архітектури для синхронізації гравців у реальному часі, реалізація інтерактивної карти та алгоритмів обробки даних.

Короткий зміст роботи: Проведено аналіз сучасних технологій для створення багатокористувацьких ігор, таких як Unity та Mirror. Розглянуто основні переваги та недоліки обраних інструментів і бібліотек. Розроблено прототип освітньої гри, яка базується на карті України з поділом на регіони. У грі реалізовано механіку питань і відповідей, таймер для обмеження часу, динамічну зміну кольору регіонів та підрахунок очок. Також інтегровано функціонал для налаштувань, що дозволяє гравцям вводити свої імена та змінювати гучність звуку.

Результатом розробки є прототип багатокористувацької освітньої гри, яка дозволяє гравцям змагатися за контроль регіонів через відповіді на освітні питання. Гра має потенціал для інтеграції в освітні програми та може бути розширена для мобільних платформ.

Ключові слова: Unity, Mirror, багатокористувацька гра, інтерактивна карта, освітня гра, клієнт-серверна архітектура.

ABSTRACT

Text part of the master's qualification work: 53 pages, 16 pictures, 15 sources.

Scientific task is develop a multiplayer online game with an interactive map of Ukraine based on Unity and the Mirror library.

Goal of the work is create a prototype of an educational game that combines elements of strategy and training with the possibility of multiplayer interaction

Object of research – the process of developing multiplayer games with an interactive interface.

Subject of research – the technologies of creating games using Unity, C#, and the Mirror library.

Research methods – analysis of modern game development technologies, application of client-server architecture for real-time synchronisation of players, implementation of an interactive map and data processing algorithms.

Summary of the work: An analysis of modern technologies for creating multiplayer games, such as Unity and Mirror, was carried out. A prototype of an educational game based on a map of Ukraine with a division into regions was developed. The game implements the mechanics of questions and answers, a timer for time limitation, dynamic colour change of regions, and scoring.

The result of the development is a prototype of a multiplayer educational game. The game has the potential to be integrated into educational programmes and can be extended for mobile platforms.

Keywords: Unity, Mirror, multiplayer game, interactive map, educational game, client-server architecture.

ЗМІСТ

ВСТУП

Сучасний світ неможливо уявити без комп'ютерних технологій та інтерактивних ігор, які займають значне місце не лише в індустрії розваг, але й в освіті, комунікації та розвитку навичок. Розробка онлайн-ігор стала одним з найбільш динамічних і популярних напрямків програмування, що поєднує в собі використання сучасних інструментів, мов програмування і технологій для створення зручних і функціональних продуктів.

Вибір Unity як платформи для розробки онлайн-ігор виправданий її універсальністю, простотою використання та широкими можливостями для створення як 2D, так і 3D додатків. Мережева технологія Mirror дозволяє реалізувати багатокористувацьку взаємодію в режимі реального часу, що є важливим для розробки змагальних та спільних ігор. Мова програмування C# забезпечує високу продуктивність, стабільність та простоту написання коду, що є особливо актуальним для розробників у сучасному ігровому середовищі.

Актуальність дослідження визначається зростаючим попитом на якісні багатокористувацькі онлайн-ігри, які не тільки розважають користувачів, але й сприяють розвитку інтелектуальних та соціальних навичок. Реалізація освітньої гри на базі платформи Unity з використанням мережевої технології Mirror є важливим кроком у розвитку нових освітніх методологій, заохочуючи молодь до навчання через інтерактивний та приємний формат.

1 АНАЛІЗ ОБРАНОЇ ТЕМИ

1.1 Обґрунтування актуальності обраної теми

Останніми роками онлайн-ігри стали однією з найпомітніших тенденцій в індустрії розробки програмного забезпечення. Вони перестали бути просто формою розваги, а все частіше використовуються в освіті, командоутворенні та розвитку когнітивних навичок. Це робить вивчення та створення онлайн-ігор не лише захопливим, але й впливовим для різних галузей.

Зростання онлайн-ігор зумовлене розвитком мережевих технологій, платформ для розробки ігор та мов програмування. Серед них Unity зарекомендувала себе як провідна платформа для створення інтерактивних ігор завдяки простоті використання, гнучкості та кросплатформенності. У поєднанні з мережевими рішеннями, такими як Mirror, Unity забезпечує потужне середовище для розробки багатокористувацьких ігор у реальному часі.

Освітні ігри, зокрема, набувають все більшої популярності як інструменти для покращення навчального процесу. Вони пропонують інтерактивні та захоплюючі методи вивчення складних тем, що робить їх придатними як для традиційної, так і для нетрадиційної освіти. Розробка гри, що поєднує багатокористувацьку взаємодію в режимі реального часу, механіку перевірки знань та зручний дизайн, може задовольнити зростаючий попит на інноваційні освітні інструменти.

Актуальність цього дослідження полягає в тому, що воно зосереджене на поєднанні освіти та технологій через багатокористувацьку онлайн-гру. Використовуючи сучасні інструменти, такі як Unity та Mirror, цей проект демонструє, як ігри можуть слугувати як розважальним, так і освітнім середовищем. Практичний результат - повнофункціональна онлайн-гра - демонструє потенціал технологій для створення змістовного користувацького досвіду та задоволення зростаючих потреб як розважального, так і освітнього секторів.

1.2 Мета та завдання дослідження

Основною метою цього дослідження є розробка 2D багатокористувацької онлайн-гри з використанням платформи Unity та мережевої технології Mirror. Ця гра покликана продемонструвати клієнт-серверну взаємодію в реальному часі,

забезпечуючи захоплюючий та інтерактивний досвід для гравців. Інтегруючи освітні елементи в ігровий процес, проект прагне поєднати розваги та навчання, пропонуючи змістовний та інноваційний ігровий досвід.

Для досягнення цієї мети дослідження фокусується на кількох ключових аспектах. Воно починається з вивчення мови програмування C#, підкреслюючи її особливості та переваги в контексті розробки ігор. Дослідження також заглиблюється у можливості Unity як універсальної платформи для створення інтерактивних та крос-платформних додатків. Крім того, досліджується мережева бібліотека Mirror для забезпечення надійної та безперебійної багатокористувацької функціональності в реальному часі.

Ретельно спланована архітектура, заснована на клієнт-серверній моделі, розробляється для підтримки масштабованості та стабільності, формуючи кістяк гри. Потім проект переходить до фази практичної реалізації, де реалізуються основні функції, такі як користувацький інтерфейс, ігрова логіка та мережева синхронізація. Всебічне тестування забезпечує функціональність гри та вирішує потенційні проблеми, забезпечуючи відшліфований кінцевий продукт. Насамкінець пропонуються рекомендації щодо розширення можливостей гри та її адаптації до різних платформ, що забезпечить її актуальність та зручність використання в різних середовищах.

1.3 Огляд структури роботи

Це дослідження ретельно структуроване, щоб забезпечити ретельне вивчення дизайну, розробки та аналізу багатокористувацької онлайн-гри, створеної за допомогою технологій Unity та Mirror. Структура забезпечує безперервний потік інформації, спрямовуючи читача від фундаментальних концепцій до практичної реалізації та оцінки проекту.

Вступ задає тон, визначаючи актуальність обраної теми та окреслюючи мету і завдання дослідження. Він висвітлює значення багатокористувацьких ігор у сучасному технологічному та освітньому ландшафті, підкреслюючи їхній потенціал для поєднання розваг зі змістовним навчальним досвідом.

Перший розділ заглиблюється в аналітичний погляд на тему. У ньому розглядається роль онлайн-ігор у сучасному суспільстві, досліджуються технологічні досягнення, що формують цю сферу, та обговорюються освітні можливості, які відкривають багатокористувацькі ігри. У цьому розділі також

підкреслюється зростаючий попит на інноваційні, інтерактивні та освітньо насичені ігрові рішення.

Другий розділ присвячений мові програмування C#, в ньому подано огляд її еволюції, ключових особливостей, сильних і слабких сторін та обмежень. Крім того, у ньому розглядається застосування C# у різних сферах, з особливим акцентом на її інтеграції з Unity для розробки ігор. Ця глава позиціонує C# як ідеальний вибір для реалізації надійної та ефективної ігрової механіки.

Третій розділ переходить до детального розгляду інструментів і технологій, використаних у проекті. Досліджується універсальність Unity як платформи для розробки ігор, а також можливості мережевої бібліотеки Mirror для полегшення багатокористувацької взаємодії в реальному часі. Цей розділ також проливає світло на те, як ці інструменти синергічно поєднуються з C# для створення масштабованого та інтерактивного ігрового досвіду.

Четвертий розділ присвячений етапу проектування гри, де детально описано концептуальну основу та механіку ігрового процесу. Описується архітектурний підхід, з акцентом на клієнт-серверну модель, та обговорюється вибір інструментів, які відповідають цілям гри. Ця глава слугує планом для подальшого процесу розробки.

П'ятий розділ присвячений реалізації та розробці гри. У ньому представлено створення графічного інтерфейсу, проектування та інтеграцію ігрової логіки, а також реалізацію мережевої функціональності для забезпечення безперешкодної багатокористувацької взаємодії. Також обговорюються результати тестування, висвітлюється вирішення проблем для забезпечення оптимальної продуктивності та користувацького досвіду.

У заключному розділі пропонується критичний аналіз розробленої гри та надаються рекомендації щодо її вдосконалення. Надано пропозиції щодо розширення її функціональності, оптимізації продуктивності та адаптації гри для різних платформ, включаючи мобільні пристрої, веб-браузери та десктопні системи.

У висновках узагальнюються результати дослідження, осмислюються його практичні досягнення та майбутній потенціал. Список використаних джерел та додатки містять додаткові матеріали, що забезпечують повноту та достовірність дослідження.

Додаток А містить додаткові матеріали, необхідні для розуміння та відтворення проекту. Сюди входять слайди презентації, що підсумовують процес дослідження та розробки, а також повний програмний код гри.

2 ОГЛЯД МОВИ ПРОГРАМУВАННЯ C#

2.1 Історія розвитку та основні особливості

C# - це сучасна мова програмування високого рівня, розроблена компанією Microsoft у 2000 році як частина фреймворку .NET. Розроблена Андерсом Хейлсбергом (Anders Hejlsberg), C# була покликана подолати розрив між ефективністю низькорівневих мов, таких як C++, і простотою високорівневих мов, таких як Java. За роки свого існування вона стала однією з найпопулярніших та найуніверсальніших мов у світі, яку широко використовують для створення десктопних додатків, веб-сервісів, хмарних рішень, мобільних додатків тощо.

C# є об'єктно-орієнтованою та безпечною за типами, що робить її добре придатною для розробки масштабного програмного забезпечення, яке легко підтримується. При її розробці керувалися принципами простоти, надійності та масштабованості, які продовжують визначати її розвиток.

Кросплатформені можливості мови значно розширилися з появою .NET Core, що дозволило розробникам писати додатки, які без проблем працюють на Windows, macOS та Linux. Зростання екосистеми C# також включає Xamarin для мобільної розробки, Blazor для створення інтерактивних веб-додатків та інтеграцію з Azure для хмарних рішень.

Ключовим моментом в історії C# стало його прийняття ігровим рушієм Unity у 2005 році, що зробило його основною мовою сценаріїв для розробки ігор. Ця інтеграція відкрила C# мільйонам розробників завдяки домінуванню Unity в індустрії розробки ігор. Скриптовий API Unity, написаний на C#, дозволяє розробникам створювати інтерактивні, високопродуктивні 2D та 3D ігри, закріплюючи актуальність мови в сучасному геймінгу.

Ключові особливості мови C# включають в себе:

- Об'єктно-орієнтоване програмування: C# повністю підтримує принципи ООП, такі як успадкування, інкапсуляція та поліморфізм, що дозволяє розробникам писати модульний, багаторазовий та легко підтримуваний код.
- Керування пам'яттю: Автоматичне збирання сміття забезпечує ефективне керування пам'яттю без необхідності ручного втручання, зменшуючи ризик витоку пам'яті.

- Багата стандартна бібліотека: Бібліотека .NET надає широкі можливості для виконання таких завдань, як робота з файлами, мережева робота, операції з базами даних та графічні інтерфейси.
- LINQ (Language Integrated Query): C# пропонує інтуїтивно зрозумілий спосіб запитувати колекції та бази даних безпосередньо в мові, що спрощує завдання маніпулювання даними.
- Асинхронне програмування: Завдяки підтримці асинхронізації/очікування, C# спрощує обробку паралельних операцій, що є важливим для адаптивних додатків.
- Крос-платформна розробка: Використовуючи такі фреймворки, як .NET Core, розробники можуть створювати додатки, які працюють на різних платформах і пристроях.

C# знайшла застосування у багатьох сферах, включаючи корпоративне програмне забезпечення, фінансові системи, розробку ігор, веб-додатків та пристроїв Інтернету речей. Її універсальність у поєднанні зі сприятливим середовищем розробки та активною спільнотою зміцнила її позиції як наріжного каменю сучасної розробки програмного забезпечення.

На рисунку 2.1 зображений логотип мови програмування C#.

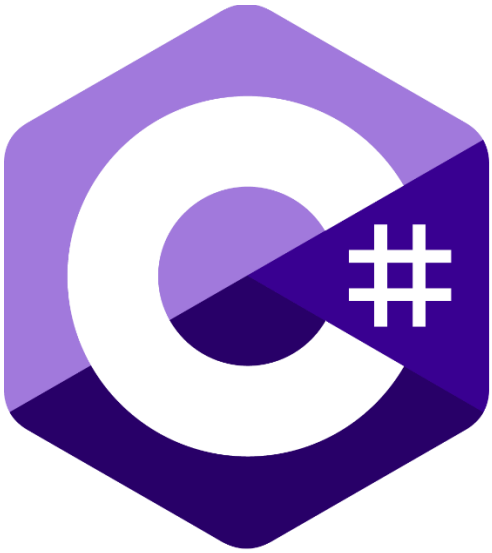


Рисунок 2.1 – Логотип мови програмування C#

2.2 Переваги та недоліки

Переваги C#:

- Простота використання: Чистий, послідовний синтаксис і потужна підтримка спільноти роблять C# зручною для початківців.

- Об'єктно-орієнтоване проектування: Дозволяє створювати модульні, підтримувані та масштабовані додатки.
- Багата екосистема: Фреймворк .NET надає бібліотеки для таких завдань, як робота з мережею, введення/виведення файлів та доступ до баз даних.
- Крос-платформна підтримка: Завдяки .NET Core додатки на C# працюють на Windows, macOS та Linux.
- Автоматичне керування пам'яттю: Збір сміття спрощує керування ресурсами та зменшує витоки пам'яті.
- Розширені засоби розробки: IDE, такі як Visual Studio, підвищують продуктивність завдяки інструментам налагодження та тестування.
- Асинхронне програмування: Вбудований асинхронний режим (async/await) підтримує швидку та ефективну роботу додатків.

Недоліки C#:

- Залежність від .NET: C# значною мірою залежить від екосистеми .NET, яка може бути непридатною для використання в середовищах, відмінних від .NET.
- Накладні витрати на продуктивність: C# працює повільніше, ніж мови нижчого рівня, такі як C++, у критичних до апаратного забезпечення додатках.
- Використання ресурсів: Програми можуть споживати більше пам'яті порівняно з програмами, написаними на рідній мові.
- Платформо-специфічні особливості: Деякі бібліотеки та функції орієнтовані на Windows, що в певних випадках обмежує переносимість.

2.3 Unity як платформа для створення ігор

Unity було обрано в якості платформи для розробки цього проекту завдяки його універсальності, простоті використання та широкій підтримці розробки 2D та багатокористувацьких ігор. У порівнянні з іншими популярними ігровими рушіями, такими як Unreal Engine та Godot, Unity має явні переваги, які роблять його ідеальним вибором для створення цього проекту.

На рисунку 2.2 зображений логотип ігрового рушія Unity.

Рисунок 2.2 – Логотип ігрового рушія Unity

Порівняння з Unreal Engine

Unreal Engine відомий своїми потужними можливостями рендерингу і часто є кращим вибором для AAA ігор та візуально інтенсивних 3D проектів. Однак для менших за масштабом 2D-проектів, таких як у цьому дослідженні, Unreal Engine може бути невиправдано складним та ресурсоємним. Unity, навпаки, є легким і надає інструменти, спеціально оптимізовані для розробки 2D ігор, такі як спеціальний 2D фізичний рушій та система управління спрайтами. Крім того, залежність Unreal від C++ як основної мови сценаріїв призводить до більш крутої кривої навчання порівняно з використанням C# в Unity, який є простішим у вивченні та більш поблажливим для розробників.

На рисунку 2.3 зображений логотип ігрового рушія Unreal Engine.

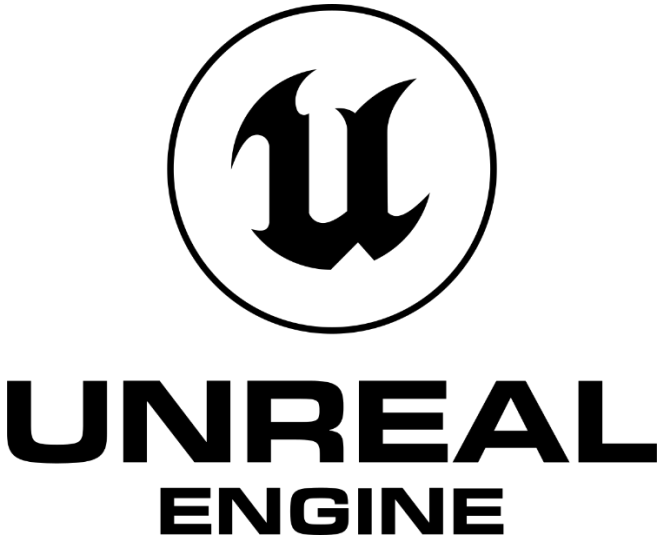


Рисунок 2.3 – Логотип ігрового рушія Unreal Engine

Порівняння з Godot

Godot є зростаючим конкурентом у сфері розробки ігор, пропонуючи рішення з відкритим вихідним кодом та простим, легким рушієм. Хоча Godot приваблює інді-розробників своєю економічністю та простотою, йому бракує надійності та підтримки в масштабах індустрії, як у Unity. Unity пропонує більш розвинену екосистему, включаючи всеосяжний Asset Store та широкі можливості інтеграції зі сторонніми розробниками. Крім того, мережеві можливості Unity, розширені такими інструментами, як Mirror, є більш усталеними та надійними для розробки багатокористувацьких ігор у порівнянні з відносно новими мережевими можливостями Godot.

На рисунку 2.4 зображений логотип ігрового рушія Godot.

Рисунок 2.4 – Логотип ігрового рушія Godot

Основні сильні сторони Unity включають:

- Крос-платформна підтримка: Unity дозволяє розробникам орієнтуватися на різні платформи, включаючи настільні, мобільні, веб та консолі, з мінімальними додатковими зусиллями.
- Простота використання: Інтуїтивно зрозумілий інтерфейс Unity у поєднанні з підтримкою мови програмування C# робить його доступним для розробників усіх рівнів.
- Розвинена екосистема: Unity Asset Store надає доступ до величезної бібліотеки ресурсів, плагінів та інструментів, які можуть прискорити розробку.

- Інструменти для 2D розробки: Спеціальні інструменти Unity для 2D ігор, такі як редактор 2D тайлмапів та управління спрайтами, є більш досконалими, ніж ті, що пропонуються Unreal, та співставні з можливостями Godot.
- Спільнота та документація: Unity може похвалитися однією з найбільших спільнот розробників та обширною документацією, що гарантує, що розробники можуть знайти підтримку практично для будь-якої проблеми, з якою вони стикаються.
- Багатокористувацькі можливості: Завдяки таким інструментам, як Mirror, Unity спрощує реалізацію багатокористувацької функціональності, що робить його найкращим вибором для вимог цього проекту до онлайн-ігор.

2.4 Технологія Mirror для мережевої взаємодії

Mirror - це потужна мережева бібліотека з відкритим вихідним кодом для Unity, яка спрощує розробку багатокористувацьких ігор, пропонуючи комплексний фреймворк для клієнт-серверної взаємодії в реальному часі. Її було обрано для цього проекту завдяки безшовній інтеграції з Unity та надійним функціям, включаючи ефективну синхронізацію даних, підтримку клієнт-серверної архітектури та масштабованість для багатокористувацьких середовищ.

На рисунку 2.5 зображений логотип мережевої бібліотеки Mirror.

Рисунок 2.5 – Логотип мережевої бібліотеки Mirror

Ключові можливості Mirror:

- Синхронізація даних: Mirror чудово синхронізує стани гри між клієнтами в режимі реального часу. Використовуючи такі інструменти, як SyncVar для змінних і NetworkTransform для позицій об'єктів, розробники можуть гарантувати, що всі гравці відчувають узгодженість і актуальність ігрового середовища без необхідності складного ручного написання сценаріїв.
- Підтримка клієнт-серверної архітектури: Mirror працює на основі безпечної та ефективної клієнт-серверної моделі. Ця архітектура призначає один екземпляр гри в якості сервера, керуючи логікою та станом гри, в той час як інші гравці підключаються як клієнти. Ця модель ідеально підходить для масштабованих і безпечних багатокористувацьких ігор, незалежно від того, розміщені вони на виділених серверах чи через однорангові з'єднання.
- Масштабованість для багатокористувацьких ігор: Mirror призначений для роботи з проектами різного масштабу, від невеликих багатокористувацьких ігор з кількома гравцями до масштабних ігор з великою кількістю одночасних підключень. Його архітектура дозволяє розробникам оптимізувати продуктивність відповідно до конкретних потреб їхньої гри.
- Простота інтеграції з Unity: Mirror інтегрується безпосередньо в середовище розробки Unity, надаючи такі важливі компоненти, як NetworkManager та NetworkManagerHUD. Ці інструменти дозволяють розробникам швидко додавати мережеві можливості до своїх проектів без значних додаткових налаштувань.
- Гнучкість та кастомізація: Mirror дозволяє розробникам налаштовувати його функції відповідно до конкретних вимог проекту. Незалежно від того, чи це тонка настройка синхронізації даних або налаштування протоколів мережевого зв'язку, Mirror забезпечує гнучкість, необхідну для створення унікального багатокористувацького досвіду.
- Активна спільнота та документація: Як проект з відкритим вихідним кодом, Mirror має активну спільноту розробників, які постійно оновлюють бібліотеку, надають детальну документацію та пропонують підтримку для усунення несправностей і налаштування.

Можливості Mirror ідеально відповідають вимогам цього проекту. Інструменти синхронізації даних спрощують управління спільними ігровими станами, забезпечуючи безперебійний ігровий процес у реальному часі для всіх

гравців. Клієнт-серверна архітектура забезпечує надійність і безпеку, а її масштабованість гарантує, що гра зможе прийняти більшу кількість гравців у майбутніх ітераціях. Крім того, інтуїтивна інтеграція Mirror з Unity скорочує час і складність розробки, дозволяючи більш сфокусовано підходити до механіки та дизайну гри.

Хоча такі альтернативи, як Photon або Netcode for GameObjects від Unity, пропонують мережеві рішення, Mirror вирізняється поєднанням гнучкості, простоти використання та економічної ефективності. На відміну від Photon, який працює за моделлю підписки, Mirror має повністю відкритий вихідний код. У порівнянні з Netcode for GameObjects, який все ще розвивається, Mirror пропонує більш усталене і багатофункціональне рішення.

2.5 Особливості мови програмування C# у розробці ігор

C# є наріжним каменем сучасної розробки ігор, особливо в екосистемі Unity, де її гнучкість і потужність сяють. Її чистий синтаксис, потужна типізація та об'єктно-орієнтована природа роблять її ідеальною для створення структурованих та масштабованих ігрових систем.

Однією з визначних особливостей C# є її безшовна інтеграція з Unity, що дозволяє розробникам ефективно визначати ігрову механіку, фізику, анімацію та взаємодію з користувачем за допомогою фреймворку MonoBehaviour. Підтримка мовою принципів об'єктно-орієнтованого програмування (ООП) гарантує модульність та багаторазове використання, що має вирішальне значення для управління складними елементами ігрового процесу.

C# також чудово підходить для асинхронного програмування завдяки таким інструментам, як async та await, які оптимізують продуктивність, дозволяючи виконувати такі завдання, як завантаження ресурсів або керування мережевим зв'язком, не перериваючи ігровий процес. Крім того, потужна бібліотека .NET надає готові рішення для роботи з файлами, мережею та серіалізацією даних, що спрощує розробку.

Кросплатформені можливості мови в поєднанні з Unity дозволяють розробникам орієнтуватися на різні платформи, включаючи настільні, мобільні та консолі, з мінімальними налаштуваннями. Потужна підтримка спільноти та великі ресурси ще більше підвищують привабливість C#, роблячи її доступною для розробників з будь-яким рівнем досвіду.

Таким чином, C# поєднує в собі простоту з розширеними можливостями, що робить її незамінним інструментом для розробки ігор. Інтеграція з Unity забезпечує ефективну реалізацію основних функцій, дозволяючи розробникам зосередитися на створенні захопливих та відшліфованих ігрових вражень.

3 ПРОЕКТУВАННЯ ІНТЕРНЕТ-ГРИ

3.1 Концепція гри: опис ігрового процесу

Гра, розроблена в рамках цього проекту, - це 2D багатокористувацька освітня стратегія, покликана поєднати навчання з розвагами. Основна концепція полягає в тому, що гравці змагаються, відповідаючи на запитання та отримуючи контроль над регіонами України на карті. Ігровий процес побудований таким чином, щоб сприяти як співпраці, так і змаганню, створюючи динамічний та захоплюючий досвід для гравців. Гра буде дуже цікава в першу чергу як для самих українців, так і для іноземців, зацікавлених нашою культурою та історією.

Мета

Основна мета гри полягає в тому, щоб гравці правильно відповідали на запитання, щоб отримати регіони країни на ігровій карті. Переможцем стає гравець, який до кінця гри отримає найбільшу кількість регіонів під своїм контролем.

Ігрова механіка

Покрокова взаємодія: гра покрокова, де гравці по черзі обирають регіон на карті. Після того, як регіон обрано, гравцям пропонується питання, пов'язане з цим регіоном або загальною темою.

Система запитань і відповідей

Гравці відповідають на запитання з кількома варіантами відповідей або з короткою відповіддю. Правильна відповідь дозволяє гравцеві претендувати на обраний регіон, який потім візуально позначається як такий, що знаходиться під його контролем. Неправильні відповіді дають можливість гравцю-супернику претендувати на цей регіон.

Система підрахунку балів

Бали нараховуються на основі кількості правильних відповідей на питання та контрольованих регіонів. Бонусні бали можуть нараховуватися за послідовні правильні відповіді або за контроль над більшістю сусідніх регіонів.

Інтерактивна карта

Ігрова карта слугує основним ігровим полем. Регіони є інтерактивними, що дозволяє гравцям натискати на них і вибирати. Кожен регіон змінює колір, щоб відобразити право власності, створюючи чітке візуальне представлення прогресу.

Багатокористувацька взаємодія

Гравці з'єднуються в режимі реального часу через багатокористувацьку мережу на базі Mirror. Ця система гарантує, що обидва гравці відчувають синхронізований ігровий процес, а оновлення стану гри миттєво відображаються на всіх клієнтах.

Динамічна складність

Для підвищення зацікавленості складність запитань може зростати в міру проходження гри, забезпечуючи збалансований та цікавий досвід для гравців з різними рівнями підготовки.

Освітній аспект

Гра включає в себе освітній компонент, використовуючи систему запитань для охоплення таких тем, як географія, історія або загальні знання. Це не тільки робить ігровий процес інтелектуально стимулюючим, але й додає цінності, сприяючи навчанню через розваги.

Мистецтво та стиль

Гра виконана в мінімалістичному стилі 2D-арту, що підкреслює простоту та зрозумілість. Карта візуально розділена на окремі регіони, кожен з яких легко ідентифікується. Анімація та візуальні ефекти використовуються помірковано, щоб зосередити увагу на ігровому процесі, одночасно підвищуючи інтерактивність.

Цільова аудиторія

Гра призначена для гравців, які шукають невимушену, але інтелектуально захоплюючу гру. Вона розрахована на людей, які цікавляться дрібницями, стратегічними іграми та змагальним багатокористувацьким ігровим процесом. Її освітній характер робить її придатною для молодшої аудиторії, студентів або будь-кого, хто прагне розширити свої знання в приємному форматі.

3.2 Архітектура проекту: клієнт-серверна модель

У грі використовується клієнт-серверна архітектура, яка вважається найбільш підходящою для багатокористувацьких ігор завдяки своїй надійності, масштабованості та можливості керувати синхронізацією стану гри.

Огляд архітектури

У клієнт-серверній моделі сервер виступає в ролі центрального органу, керуючи логікою, станом і синхронізацією гри. Клієнти підключаються до сервера, щоб надсилати вхідні дані (наприклад, дії гравців) та отримувати оновлення про стан гри (наприклад, оновлені кольори мапи, бали гравців).

Така архітектура забезпечує узгодженість ігрового процесу на всіх клієнтах і запобігає шахрайству, зберігаючи чутливу логіку на стороні сервера.

Компоненти архітектури

1. Сервер

Сервер є основою гри і відповідає за:

- Управління станом гри: Сервер підтримує поточний стан гри, включаючи володіння регіонами, бали гравців та поточні ходи.
- Обробка вхідних даних: Сервер обробляє дії гравців, перевіряє їх і відповідно оновлює стан гри.
- Синхронізація: Сервер надсилає оновлення всім підключеним клієнтам, щоб гарантувати, що кожен гравець має однакове уявлення про гру.

2. Клієнти

Клієнти представляють пристрої гравців та ручку:

- Користувацьке введення: Гравці взаємодіють з грою, натискаючи регіони, відповідаючи на запитання або виконуючи інші дії.
- Візуальні оновлення: Клієнти відображають поточний стан гри (наприклад, оновлені кольори мапи) на основі даних, отриманих від сервера.
- Спілкування з сервером: Клієнти надсилають дії (наприклад, заявляють регіон) на сервер і отримують оновлення у відповідь.

Реалізація з використанням Mirror

Гра використовує мережеву бібліотеку Mirror для реалізації клієнт-серверної архітектури в Unity. Mirror спрощує розробку багатокористувацьких ігор, надаючи готові інструменти та компоненти.

Ключові можливості Mirror в архітектурі:

- NetworkManager: NetworkManager у Mirror відповідає за клієнт-серверні з'єднання, підбір гравців та керування сесіями.
- SyncVar: Ця функція забезпечує автоматичну синхронізацію змінних (наприклад, приналежність регіону) між сервером і клієнтами.
- Команди і RPC: Команди використовуються для надсилання клієнтських дій на сервер, тоді як виклики віддалених процедур (RPC) дозволяють серверу транслювати оновлення клієнтам.

3.3 Вибір інструментів для реалізації

Для цього проекту було ретельно відібрано декілька інструментів та бібліотек, які відповідають вимогам створення 2D багатокористувацької гри з інтерактивною картою України.

1. Ігровий рушій Unity

Unity було обрано як основну платформу для розробки гри завдяки її універсальності, потужному набору функцій та підтримці розробки 2D ігор.

Основні причини вибору Unity включають:

- Простота використання: Інтуїтивно зрозумілий інтерфейс Unity та вичерпна документація роблять його доступним для розробників усіх рівнів.
- Крос-платформна підтримка: Unity дозволяє розгортання на різних платформах, включаючи Windows, macOS, Android, iOS та веб-браузери.
- Інтеграція з C#: Unity підтримує написання сценаріїв на C#, мові, відомій своєю простотою та ефективністю у розробці ігор.
- Багата екосистема: Unity Asset Store надає готові ресурси, інструменти та плагіни, які прискорюють розробку.

2. Мережева бібліотека Mirror

Для реалізації багатокористувацької функціональності було обрано мережеву бібліотеку Mirror завдяки її безшовній інтеграції з Unity та потужним можливостям для клієнт-серверної взаємодії.

Mirror спрощує розробку багатокористувацьких ігор завдяки таким можливостям, як:

- Легке налаштування: Mirror надає готові компоненти, такі як NetworkManager, для керування з'єднаннями.
- Синхронізація даних: Такі інструменти, як SyncVar та віддалений виклик процедур (RPC), забезпечують синхронізацію оновлень стану гри на всіх клієнтах у режимі реального часу.
- Гнучкість: Mirror підтримує широкий спектр багатокористувацьких сценаріїв, від невеликих однорангових мереж до великих виділених серверів.
- Відкритий вихідний код: Mirror є безкоштовним у використанні та має активну спільноту, що забезпечує постійні оновлення та підтримку.

3. Visual Studio Code

Visual Studio Code було обрано як інтегроване середовище розробки (IDE) для написання та налагодження скриптів на C#.

Основні характеристики включають:

- Легкість та швидкість: ідеально підходить для роботи над малими та середніми проектами без зайвих накладних витрат.
- Широкі розширення: Підтримка специфічних для Unity плагінів, таких як IntelliSense для Unity, підвищує продуктивність.
- Інтегрована налагодження: Вдосконалені інструменти налагодження дозволяють швидко виявляти та вирішувати проблеми з кодуванням.

4. Інструменти для дизайну

Adobe Photoshop/Illustrator (або еквівалент):

- Використовується для підготовки та редагування двовимірної карти України, забезпечуючи її відповідність дизайну та функціональним вимогам гри.
- Допомагає розділити карту на регіони для інтерактивності.

Редактор Sprite Unity:

- Використовується для розбиття карти на окремі регіони, що забезпечує індивідуальну інтерактивність для кожної області.

5. Контроль версій: Git

Контроль версій необхідний для відстеження змін та ефективної співпраці під час розробки.

Git, а також репозиторій на GitHub або GitLab, використовується для:

- Відстежувати зміни: Відстежуйте та керуйте оновленнями кодової бази та ресурсів.
- Співпраці: Дозволити декільком розробникам працювати над проектом одночасно без конфліктів.
- Резервне копіювання: Переконайтеся, що проект надійно захищений резервною копією і може бути відновлений у разі виникнення проблем.

6. Інструменти тестування та налагодження

- Unity Profiler: Використовується для моніторингу продуктивності гри, виявлення вузьких місць та оптимізації використання ресурсів.
- Mirror Debugging Tools: Mirror надає вбудовані налагоджувальні компоненти для тестування багатокористувацької функціональності, наприклад, з'єднання гравців і синхронізації даних.
- Unity Play Mode: Дозволяє тестувати ігрову механіку та багатокористувацьку взаємодію в режимі реального часу.

3.4 Створення основної сцени та графічного інтерфейсу

Основна сцена та графічний інтерфейс (GUI) складають основу гри, надаючи гравцям візуально привабливе та інтерактивне середовище для взаємодії з ігровим процесом. У цьому розділі описано процес створення головної сцени та проектування графічного інтерфейсу, що забезпечує цілісний та зручний для користувача досвід.

Проектування головної сцени

Основна сцена зосереджена навколо двомірної карти України, розділеної на регіони, які слугують основними інтерактивними елементами. Ключові елементи головної сцени включають

Розміщення карти

Двомірна карта України, підготовлена та імпортована у вигляді спрайту, розміщується в центрі сцени.

Кожна область окреслена для наочності та інтерактивності. Регіони розділяються за допомогою редактора спрайтів Unity або зовнішніх інструментів дизайну, таких як Photoshop.

Взаємодія регіонів

Кожен регіон налаштовується за допомогою PolygonCollider2D для забезпечення точної взаємодії.

Спрайтам регіонів призначено скрипти для обробки кліків, підсвічування та зміни кольору на основі дій гравця.

Фон та стилізація

Чистий, мінімалістичний фон використовується для того, щоб зосередити увагу на карті, забезпечуючи при цьому візуально приємний інтерфейс. Тонкі градієнти або візерунки можуть бути додані для покращення естетики, не відволікаючи від ігрового процесу.

Створення графічного інтерфейсу

Графічний інтерфейс призначений для надання гравцям необхідної інформації та елементів керування в інтуїтивно зрозумілій формі. Ключові компоненти включають:

Табло

Табло відображає поточні результати кожного гравця, кількість контрольованих регіонів та загальну кількість очок. Розташовується вгорі або збоку екрана для зручності перегляду.

Індикатор ходу

Індикатор ходу показує, який зараз хід гравця, забезпечуючи чітку комунікацію про стан гри.

Панель дій

Панель дій дозволяє гравцям переглядати свої можливості (наприклад, заявити регіон, відповісти на питання).

Ця панель динамічно оновлюється залежно від поточного ходу та стану гри.

Спливаюче вікно з питанням

Коли гравець обирає регіон, у спливаючому вікні відображається питання для цього регіону.

У спливаючому вікні можна вибрати декілька варіантів відповідей або ввести текст, а також натиснути кнопки «Надіслати» та «Скасувати».

Екран закінчення гри

Після завершення гри на екрані відображається переможець і ключова статистика, наприклад, контрольовані регіони і загальна кількість очок.

Реалізація в Unity

Налаштування полотна

Unity Canvas створюється для розміщення всіх елементів графічного інтерфейсу. Компоненти інтерфейсу, такі як текст, кнопки та панелі, додаються та позиціонуються в межах полотна.

Адаптивний дизайн

Полотно налаштовується для масштабування відповідно до роздільної здатності екрану, забезпечуючи однаковий вигляд графічного інтерфейсу на різних пристроях.

Динамічні оновлення

До елементів інтерфейсу додаються скрипти, які динамічно оновлюють їхній вміст залежно від ігрових подій. Наприклад, табло оновлюється щоразу, коли гравець претендує на регіон.

Інтеграція з ігровою логікою

Графічний інтерфейс пов'язаний з логічними скриптами гри, забезпечуючи безперебійну взаємодію між інтерфейсом та механікою ігрового процесу.

Тестування сцени та інтерфейсу

- Юзабіліті-тестування (зручність використання): Інтерфейс перевіряється на зрозумілість, чуйність і простоту використання.
- Тестування взаємодії: Кожен регіон тестується на точність зміни кольору та спливаючих вікон.
- Тестування продуктивності: Сцена профілюється на продуктивність, щоб забезпечити безперебійну взаємодію навіть на пристроях нижчого класу.

3.5 Реалізація ігрової логіки

У цьому розділі описані ключові компоненти ігрової логіки, включаючи управління ходами гравців, володіння регіонами, механіку запитань-відповідей та підрахунок очок.

Основні компоненти ігрової логіки

1. Покрокова система

Для забезпечення справедливості та структурованості ігрового процесу в грі використовується покрокова система:

- Управління ходами: Скрипт управління ходами відстежує активного гравця і чергує ходи після кожної дії (наприклад, вибору регіону або відповіді на питання).
- Візуальний зворотний зв'язок: Інтерфейс містить індикатор ходу, який показує, чий зараз хід, забезпечуючи чіткість для обох гравців.
- Перевірка ходу: Гравці можуть виконувати дії тільки під час призначеного їм ходу, що забезпечується логікою в менеджері ходів.

2. Вибір регіону та володіння ним

Кожен регіон на карті є взаємодіючим і на нього можуть претендувати гравці:

- Взаємодія регіонів:

- Регіони конфігуруються за допомогою колайдерів для виявлення кліків.
- Скрипт обробляє вибір регіону, змінюючи його колір залежно від гравця, який претендує на нього (наприклад, червоний для гравця 1, жовтий для гравця 2).

- Присвоєння прав власності:
  - Приналежність регіону зберігається у змінній, що гарантує, що тільки незатребувані регіони можуть бути обрані.
  - Право власності синхронізується між усіма клієнтами в багатокористувацькому режимі за допомогою Mirror's SyncVar.

3. Система запитань-відповідей

Запитання є основним механізмом для визначення регіональної приналежності:

- Відображення питання:
  - Коли гравець обирає регіон, у спливаючому вікні з'являється питання, пов'язане з цим регіоном.
  - Питання беруться з попередньо визначеної бази даних або списку.
- Перевірка відповідей:
  - Гравці надсилають відповіді через інтерфейс.
  - Логіка гри перевіряє, чи відповідь правильна, і відповідно оновлює стан регіону.
- Питання з таймером (необов'язково):
  - Таймер може бути доданий для заохочення швидкого прийняття рішень і підтримки темпу гри.

4. Система підрахунку балів

Для визначення переможця відстежуються бали гравців:

- Підрахунок балів:
  - Гравці заробляють бали за кожну правильну відповідь на питання і за кожен регіон, на який вони претендують.
  - Бонусні бали можуть бути нараховані за послідовні правильні відповіді або стратегічні досягнення, такі як контроль над сусідніми регіонами.
- Відображення балів:

- Табло в інтерфейсі динамічно оновлюється, щоб відображати поточні бали обох гравців.

3.6 Впровадження мережевої синхронізації за допомогою Mirror

Для реалізації багатокористувацької функціональності було обрано бібліотеку Mirror Networking завдяки її потужним можливостям для синхронізації в реальному часі в Unity. У цьому розділі описано методи, що використовуються для забезпечення узгодженого ігрового процесу для кількох гравців, з акцентом на синхронізацію ігрових станів, дій гравців та володіння регіонами.

Архітектура багатокористувацької гри

У грі використовується клієнт-серверна модель, де сервер виступає центральним органом управління ігровою логікою, взаємодією гравців та володінням регіонами. Всі дії клієнта перевіряються і обробляються на сервері, забезпечуючи узгодженість і запобігаючи конфліктам.

Синхронізація ігрових елементів

1. Приналежність регіону

Кожен регіон на ігровій карті є інтерактивним і може бути заявлений гравцем. Щоб переконатися, що всі клієнти відображають однаковий стан:

- SyncVar для володіння регіонами: Приналежність кожного регіону зберігається як синхронізована змінна (SyncVar) на сервері. Ця змінна автоматично оновлюється на всіх клієнтах щоразу, коли запитується регіон.
- Оновлення в режимі реального часу: Виклик віддаленої процедури (RPC) використовується для трансляції візуальних оновлень, таких як зміна кольору регіону, щоб відобразити нове право власності.

2. Керування чергами

Покроковий ігровий процес вимагає, щоб усі клієнти знали, чия черга настала в кожен момент часу. Це досягається за допомогою:

- SyncVar для поточного гравця: Сервер підтримує синхронізовану змінну для відстеження активного гравця.
- Оновлення індикатора ходу: ClientRpc використовується для оновлення індикатора черги на кожному клієнті, коли черга змінюється.

3. Механіка запитань-відповідей

Система запитань синхронізується, щоб гарантувати, що всі гравці бачать одне й те саме запитання та отримують однакові відповіді. Це включає в себе:

- Командний і RPC робочий процес: Запитання надсилаються клієнтам за допомогою ClientRpc, а відповіді гравців перевіряються на сервері за допомогою команд.
- Синхронізований зворотній зв'язок: Сервер повідомляє результат (правильний чи неправильний) усім клієнтам у режимі реального часу, забезпечуючи послідовність у проходженні гри.

Реалізація в Unity за допомогою Mirror:

- Налаштування NetworkManager: Компонент Mirror NetworkManager було використано для керування клієнт-серверними з'єднаннями та управління сесіями. Це забезпечило безперебійний зв'язок між гравцями та дозволило серверу ефективно керувати станом гри.
- Синхронізація за допомогою SyncVars: Синхронізовані змінні використовувалися для критично важливих елементів гри, таких як володіння регіонами та ходи гравців. SyncVars автоматично поширює зміни з сервера на всіх клієнтів.
- Оновлення в реальному часі за допомогою RPC: Виклики віддалених процедур (RPC) транслюють оновлення, такі як зміни кольору регіону або індикаторів ходів, гарантуючи, що всі клієнти відображають однакову інформацію.
- Потік взаємодії:
    - Клієнт натискає на регіон, щоб заявити його.
    - Дія надсилається на сервер для перевірки.
    - Сервер оновлює стан гри і транслює зміни всім клієнтам.

Проблеми та рішення:

- Проблеми із затримками: Мережеву затримку було мінімізовано шляхом зменшення частоти та розміру пакетів даних, що надсилаються між клієнтами та сервером.
- Відключення гравців: Реалізовано логіку, що дозволяє м'яко обробляти відключення гравців, наприклад, перепризначати їхні регіони в нейтральний стан.
- Навантаження на сервер: Сервер було оптимізовано для обробки декількох одночасних з'єднань, ефективно керуючи мережевими повідомленнями та обробляючи лише найважливіші оновлення.

Синхронізація багатокористувацького ігрового процесу за допомогою Mirror забезпечує послідовний і цікавий досвід для всіх гравців. Завдяки використанню інструментів Mirror для синхронізації в реальному часі, гра забезпечує безперебійне оновлення стану гри, точне управління ходами та синхронізовану механіку запитань-відповідей. Ці функції є критично важливими для забезпечення відшліфованого та конкурентного багатокористувацького досвіду.

3.7 Результат проекту

Взаємодія з регіонами

Кожен регіон України представлений у вигляді окремого інтерактивного об'єкта. Гравець може натиснути на будь-який регіон, якщо він ще не захоплений. Після натискання з'являється запитання, залежно від відповіді на яке визначається доля регіону.

Карта України розділена на регіони. При натисканні на регіон він підсвічується, а інші залишаються неактивними.

Запитання та відповіді

Під час натискання на регіон відкривається вікно із запитанням і кількома відповідями. Якщо гравець відповідає правильно, регіон захоплюється. Якщо два гравці відповідають одночасно, то переможець визначається за швидкістю відповіді.

Панель із запитаннями відкривається поверх карти. Кнопки відповідей розташовуються вертикально або горизонтально.

Захоплення регіону із випадковим кольором.

Коли гравець захоплює регіон, його колір автоматично змінюється на випадковий колір, відмінний від стандартного кольору карти. Регіон змінює колір після захоплення, додаючи карті динамічності.

Додавання запитань

Розробник може додавати власні запитання та відповіді тільки через програмний код, які будуть задіяні тільки після оновлення гри. В подальшому можливе користувацьке додавання питань та варіантів відповідей.

Запитання з множинним вибором

Гравець обирає одну з кількох відповідей. Якщо відповідь правильна, регіон захоплюється. Кнопки множинного вибору яскраві, добре помітні на тлі.

Запитання з числовим введенням

Деякі запитання вимагають введення точної відповіді (наприклад, «У якому році Україна проголосила свою незалежність?»). Поле введення розташоване на панелі запитань, а під ним - кнопка «Підтвердити».

Таймер для відповіді

Гравцям дається обмежений час на відповідь (наприклад, 15 секунд). Після закінчення таймера відповідь автоматично вважається неправильною.

Таймер відображається у вигляді прогрес-бара або текстового зворотного відліку.

Накопичення очок

Кожен захоплений регіон додає гравцеві 500 очок. Якщо регіон захоплено в іншого гравця, 500 очок списуються з попереднього власника. Таблиця з очками кожного гравця відображається поруч із картою.

Поперемінні ходи

Гравці по черзі відповідають на запитання. Коли хід одного гравця завершено, хід передається іншому. На екрані відображається текст, наприклад «Хід гравця 1» або «Хід гравця 2».

Один або обидва гравці можуть змагатися за регіон

У деяких ситуаціях лише один гравець відповідає на запитання, а інший спостерігає. В інших - обидва гравці змагаються, хто швидше і правильніше відповість на запитання. У випадку спостереження, інший гравець бачить статус: «Суперник відповідає».

Завершення гри

Гра закінчується, коли один з гравців захопив усі регіони. Переможець отримує повідомлення «Ви перемогли», а той, хто програв, отримує повідомлення «Ви програли».

З'являється екран завершення гри з остаточними результатами та текстовим повідомленням.

Дизайн та візуальні аспекти гри

- Карта: Стилізована карта України з чітко окресленими регіонами.
- Інтерфейс: Мінімалістичний, з простими кольорами, які не відволікають.
- Ефекти: Анімація підсвічування регіонів при натисканні, плавне затемнення областей, якими не користуються.
- Музика: фонова музика та звукові ефекти під час взаємодії.

Механіка для однокористувацького та багатокористувацького режимів

- Однокористувацький режим: гравець змагається з комп'ютером, який автоматично відповідає на запитання, або вірно, або з помилкою.
- Багатокористувацький режим: гравці підключаються через сервер, відповідають на запитання та змагаються за регіони в режимі реального часу.

3.8 Перспективи розширення функціоналу гри

Багатокористувацька освітня гра забезпечує міцну основу для інтерактивного навчання та ігрового процесу, але існує безліч можливостей

розширити її функціональність, щоб підвищити залученість та зручність використання.

Одним із ключових напрямків є урізноманітнення контенту гри. Розширюючи тематику запитань, включаючи такі предмети, як історія, культура, наука та загальні знання, гра може зацікавити ширшу аудиторію. Крім того, інтеграція динамічної генерації запитань, де складність змінюється залежно від результатів гравця, забезпечить індивідуальний та цікавий досвід для всіх рівнів кваліфікації.

Впровадження додаткових ігрових режимів, таких як кооперативна гра, завдання на час або структурована кампанія, може ще більше урізноманітнити ігровий процес і зробити його довговічнішим.

Розширення підтримки платформи на мобільні пристрої, веб-браузери та ігрові консолі також зробить гру більш доступною для гравців на різних пристроях. Ці зміни забезпечили б збереження актуальності та привабливості гри на конкурентному ринку, а також сприяли б довготривалому залученню та задоволенню від гри.

3.9 Рекомендації щодо оптимізації продуктивності

Щоб підтримувати плавний і чуйний ігровий процес, важливо оптимізувати продуктивність гри. Це починається з мінімізації мережевих затримок у багатокористувацькій архітектурі шляхом оптимізації передачі даних і зменшення кількості непотрібних оновлень між клієнтами та сервером. Використання вбудованих інструментів Mirror для ефективного управління пропускною здатністю забезпечує стабільну синхронізацію в реальному часі для всіх гравців.

Графічні ресурси гри також можна оптимізувати для підвищення продуктивності на різних пристроях. Стиснення текстур і використання атласів спрайтів для 2D-карти та інших візуальних елементів зменшує використання пам'яті та кількість викликів до малювання. Крім того, ефективне використання вбудованих систем фізики та рендерингу Unity може мінімізувати навантаження на CPU та GPU.

Керування пам'яттю є ще одним важливим фактором. Реалізація пулу об'єктів для часто використовуваних об'єктів, таких як елементи інтерфейсу та підсвічування регіонів, забезпечує плавні переходи без зайвих інстанцій. Тестування та профілювання гри на різних пристроях дозволить виявити вузькі

місця, що дасть змогу провести цілеспрямовану оптимізацію для забезпечення стабільної продуктивності як на низькопродуктивних, так і на високопродуктивних системах.

3.10 Можливість адаптації гри для різних платформ

Розширення гри на додаткові платформи значно збільшить її доступність та кількість користувачів. Кросплатформені можливості Unity дозволяють відносно легко адаптувати гру для мобільних пристроїв, веб-браузерів та ігрових консолей. Для мобільних платформ інтерфейс та елементи керування потрібно оптимізувати для сенсорних екранів, скоригувавши розміри кнопок та макет для менших екранів. Впровадження оптимізації продуктивності, наприклад, зменшення графічної складності та використання пам'яті, забезпечить безперебійний ігровий процес на мобільних пристроях.

Веб-версія знизила б бар'єри для входу, дозволивши гравцям отримати доступ до гри безпосередньо у браузері без необхідності інсталяції. Цього можна досягти за допомогою опції збірки Unity's WebGL. Для ігрових консолей інтерфейс гри та елементи керування мають бути адаптовані до геймпадів, що забезпечить інтуїтивно зрозумілий досвід для гравців, які звикли до консольних ігор.

Адаптація гри для різних платформ дозволяє охопити ширшу аудиторію, підвищити її освітній вплив і загальну привабливість. Кожна платформа надає унікальні можливості для залучення певних демографічних груп, що в кінцевому підсумку сприяє успіху та довголіттю гри.

ВИСНОВКИ

Розробка багатокористувацької освітньої гри з використанням Unity демонструє ефективну інтеграцію технологій та навчання для створення цікавого та інтерактивного досвіду. Цей проект вдало поєднує освітню цінність дрібниць зі стратегічним ігровим процесом, пропонуючи гравцям платформу як для розваги, так і для інтелектуального зростання. Процес розробки, впровадження та вдосконалення гри дав уявлення про сучасні практики розробки ігор, а також висвітлив можливості для майбутніх інновацій.

Обрані технології, зокрема Unity як основний рушій розробки та мережева бібліотека Mirror для багатокористувацької функціональності, виявилися дуже важливими для досягнення цілей проекту. Універсальність Unity дозволила без проблем створити візуально привабливе та функціональне 2D-середовище, а Mirror спростила реалізацію синхронізації між гравцями в реальному часі, забезпечивши послідовний та конкурентний досвід. Використання C# для написання сценаріїв ще більше спростило процес розробки, полегшивши створення ігрових механік, користувацьких інтерфейсів та багатокористувацької взаємодії.

Дипломний проект підкреслив важливість клієнт-серверної архітектури для забезпечення надійності та безпеки багатокористувацьких ігор. Завдяки централізації ігрової логіки на сервері та синхронізації оновлень з клієнтами, гра досягає узгодженості ігрового процесу, мінімізуючи ризик невідповідності даних або шахрайства. Ретельна оптимізація графічних ресурсів та мережевого зв'язку забезпечила безперебійну роботу на різних пристроях, демонструючи важливість технічної ефективності в сучасній розробці ігор.

Дизайн та реалізація гри також надали цінні уроки щодо користувацького досвіду. Такі функції, як інтерактивні карти, динамічні системи запитань-відповідей та візуальний зворотній зв'язок, сприяли створенню зручного інтерфейсу, що підвищує залученість гравців. Крім того, інтеграція покрокової механіки та системи підрахунку балів збалансувала конкуренцію та співпрацю, збагативши загальний ігровий досвід.

Забігаючи наперед, можна сказати, що проект має багато можливостей для розширення та вдосконалення. Майбутні версії гри можуть включати додаткові теми питань, нові ігрові режими, вдосконалені функції та адаптація гри для інших пристроїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gregory, J. (2018). *Game Engine Architecture*. CRC Press.
2. Adams, E., & Rollings, A. (2014). *Fundamentals of Game Design*. New Riders.
3. Rabin, S. (2015). *Introduction to Game Development*. Cengage Learning.
4. Nystrom, R. (2014). *Game Programming Patterns*. Genever Benning.
5. Unity Technologies. (2023). *Unity Documentation*. Retrieved from https://docs.unity3d.com/
6. Mirror Networking. (2023). *Mirror Documentation*. Retrieved from https://mirror-networking.com/
7. Hejlsberg, A., Wiltamuth, S., & Golde, P. (2003). *The C# Programming Language*. Addison-Wesley.
8. Sweeney, T. (1999). *Unreal Engine Overview*. Retrieved from https://www.unrealengine.com/
9. O'Reilly Media. (2019). *Learning C# by Developing Games with Unity*. Packt Publishing.
10. Cook, D. (2013). *Designing Games: A Guide to Engineering Experiences*. O'Reilly Media.
11. Литвиненко, В. (2018). *Розробка ігор в середовищі Unity 3D*. Київ: Освітня література.
12. Тарасов, В. А. (2020). *Основи програмування на мові C#*. Харків: Фоліо.
13. Коваленко, С. М., & Мартинюк, Ю. (2021). *Введення у розробку мережевих ігор*. Львів: Видавничий дім "Академія".
14. Unity Україна. (2023). *Розробка ігор в Unity: офіційна документація*. Отримано з https://unity3d.com/uk
15. Бойко, О. Г. (2022). *Інформаційні технології в освіті: використання ігрових платформ*. Одеса: НДІ Інформатика.

ДОДАТОК А

КОД ДОДАТКУ

Код для музики:

```
using UnityEngine;

public class BackgroundMusic : MonoBehaviour
{
   private static BackgroundMusic instance;

   void Awake()
   {
      if (instance == null)
      {
         instance = this;
         DontDestroyOnLoad(gameObject); // Зберігаємо музику при переході між
сценами
      }
      else
      {
         Destroy(gameObject); // Уникаємо дублювання музики
      }
   }
}
```

Код для кнопки налаштувань, де є зміна гучності звуку:

```
using UnityEngine;
using UnityEngine.UI;

public class SettingsMenu : MonoBehaviour
{
   public Slider volumeSlider; // Повзунок для гучності

   void Start()
   {
      volumeSlider.value = AudioListener.volume; // Ініціалізуємо значення
повзунка
   }

   public void AdjustVolume(float volume)
```

```
    {
        AudioListener.volume = volume; // Зміна глобального рівня звуку
        Debug.Log($"Гучність змінено на: {volume}");
    }

    public void CloseSettings()
    {
        gameObject.SetActive(false); // Закриття панелі налаштувань
    }
}
```

Після натискання кнопки "Старт" з’являється панель із вибором режиму:

```
using UnityEngine;

public class GameModeSelector : MonoBehaviour
{
    public GameObject modeSelectionPanel; // Панель вибору режиму

    public void ShowModeSelection()
    {
        modeSelectionPanel.SetActive(true); // Відкриття панелі вибору режиму
    }

    public void StartSinglePlayer()
    {
        Debug.Log("Розпочинається однокористувацька гра...");
        // Завантаження сцени однокористувацької гри

UnityEngine.SceneManagement.SceneManager.LoadScene("SinglePlayerScene");
    }

    public void StartMultiplayer()
    {
        Debug.Log("Розпочинається багатокористувацька гра...");
        // Завантаження сцени багатокористувацької гри

UnityEngine.SceneManagement.SceneManager.LoadScene("MultiplayerScene");
    }

    public void CloseModeSelection()
    {
        modeSelectionPanel.SetActive(false); // Закриття панелі вибору режиму
    }
```

```
}
```

Повна структура головного меню:

```
using UnityEngine;

public class MainMenu : MonoBehaviour
{
    public GameObject settingsPanel; // Панель налаштувань
    public GameObject modeSelectionPanel; // Панель вибору режиму

    public void OpenSettings()
    {
        settingsPanel.SetActive(true); // Відкриття налаштувань
    }

    public void CloseSettings()
    {
        settingsPanel.SetActive(false); // Закриття налаштувань
    }

    public void ShowModeSelection()
    {
        modeSelectionPanel.SetActive(true); // Відкриття вибору режиму
    }

    public void CloseModeSelection()
    {
        modeSelectionPanel.SetActive(false); // Закриття вибору режиму
    }

    public void ExitGame()
    {
        Debug.Log("Вихід з гри...");
        Application.Quit(); // Завершення гри
    }
}
```

Скрипт для створення та збереження нікнейму в налаштуваннях для гравця в подальших іграх:

```
using UnityEngine;
using UnityEngine.UI;
```

```
public class PlayerSettings : MonoBehaviour
{
   public InputField nameInputField; // Поле для введення імені
   public Text displayNameText; // Текст для відображення імені
   private string playerName;

   void Start()
   {
      // Завантажуємо ім'я гравця з налаштувань
      playerName = PlayerPrefs.GetString("PlayerName", "Гравець");
      nameInputField.text = playerName;
      displayNameText.text = $"Ім'я: {playerName}";
   }

   public void SaveName()
   {
      playerName = nameInputField.text; // Зчитуємо ім'я з поля
      PlayerPrefs.SetString("PlayerName", playerName); // Зберігаємо ім'я
      PlayerPrefs.Save(); // Зберігаємо всі налаштування
      displayNameText.text = $"Ім'я: {playerName}"; // Оновлюємо текст
      Debug.Log($"Ім'я збережено: {playerName}");
   }
}
```

Цей код відповідає за обробку кліків на регіонах і передачу управління системі питань.

```
using UnityEngine;

public class RegionInteraction : MonoBehaviour
{
   public string regionName; // Назва регіону
   public bool isCaptured = false; // Захоплено чи ні
   private SpriteRenderer spriteRenderer;

   void Start()
   {
      spriteRenderer = GetComponent<SpriteRenderer>();
   }

   void OnMouseDown()
   {
      if (!isCaptured)
      {
```

```
            Debug.Log($"Регіон {regionName} натиснуто!");
            FindObjectOfType<QuestionManager>().ShowQuestion(this); // Викликаємо
питання
        }
        else
        {
            Debug.Log($"Регіон {regionName} вже захоплений.");
        }
    }

    public void CaptureRegion()
    {
        isCaptured = true;
        Color randomColor = new Color(Random.value, Random.value,
Random.value); // Рандомний колір
        spriteRenderer.color = randomColor;
        Debug.Log($"Регіон {regionName} захоплено!");
    }
}
```

Цей код відображає питання і враховує, хто перший натиснув на правильну відповідь.

```
using UnityEngine;
using UnityEngine.UI;

public class QuestionManager : MonoBehaviour
{
    public GameObject questionPanel; // Панель із запитанням
    public Text questionText; // Текст запитання
    public Button[] answerButtons; // Кнопки відповідей
    private RegionInteraction currentRegion; // Поточний регіон
    private bool winnerDecided = false;

    public void ShowQuestion(RegionInteraction region)
    {
        currentRegion = region;
        questionPanel.SetActive(true);
        winnerDecided = false;

        string question = "Що є столицею України?";
        string[] answers = { "Київ", "Львів", "Одеса", "Харків" };
        string correctAnswer = "Київ";
```

```
        questionText.text = question;

        for (int i = 0; i < answerButtons.Length; i++)
        {
            answerButtons[i].GetComponentInChildren<Text>().text = answers[i];
            int index = i;
            answerButtons[i].onClick.AddListener(() => CheckAnswer(answers[index],
correctAnswer));
        }
    }

    public void CheckAnswer(string playerAnswer, string correctAnswer)
    {
        if (!winnerDecided && playerAnswer == correctAnswer)
        {
            Debug.Log("Правильна відповідь!");
            currentRegion.CaptureRegion();
            winnerDecided = true;
        }
        else if (!winnerDecided)
        {
            Debug.Log("Неправильна відповідь.");
        }

        if (winnerDecided)
        {
            questionPanel.SetActive(false);
            foreach (var button in answerButtons)
            {
                button.onClick.RemoveAllListeners();
            }
        }
    }
}
```

Код зі списком питань, який можна легко змінювати.

```
[System.Serializable]
public class QuestionData
{
    public string question;
    public string[] answers;
    public string correctAnswer;
}
```

```
public class QuestionDatabase : MonoBehaviour
{
   public QuestionData[] questions;

   public QuestionData GetRandomQuestion()
   {
      return questions[Random.Range(0, questions.Length)];
   }
}
```

Реалізація кольору захопленого регіону через генерацію випадкових значень для RGB.

```
using UnityEngine;

public class Region : MonoBehaviour
{
   public bool isCaptured = false; // Стан регіону
   private SpriteRenderer spriteRenderer;

   void Start()
   {
      spriteRenderer = GetComponent<SpriteRenderer>();
   }

   public void CaptureRegion()
   {
      if (!isCaptured)
      {
         isCaptured = true;
         // Генеруємо рандомний колір
         Color randomColor = new Color(
            Random.Range(0.2f, 1.0f), // Червоний компонент (більше 0.2 для уникнення надто темних кольорів)
            Random.Range(0.2f, 1.0f), // Зелений компонент
            Random.Range(0.2f, 1.0f)  // Синій компонент
         );

         // Призначаємо цей колір спрайту регіону
         spriteRenderer.color = randomColor;
         Debug.Log($"Регіон захоплено! Колір: {randomColor}");
      }
      else
```

```
        {
            Debug.Log("Регіон вже захоплений!");
        }
    }
}
```

Питання з введенням чисел

```
public void ShowNumericQuestion(string question, int correctNumber)
{
    questionPanel.SetActive(true);
    questionText.text = question;
    // Тут логіка введення числа через InputField
}
```

Код з таймером

```
public class Timer : MonoBehaviour
{
    public float timeLimit = 30f;
    public Text timerText;
    private float remainingTime;
    private bool isActive = false;

    public void StartTimer()
    {
        isActive = true;
        remainingTime = timeLimit;
    }

    void Update()
    {
        if (isActive && remainingTime > 0)
        {
            remainingTime -= Time.deltaTime;
            timerText.text = Mathf.Round(remainingTime).ToString();
        }
        else if (isActive)
        {
            Debug.Log("Час вийшов!");
            isActive = false;
        }
    }
}
```

Накопичення очок

```
public class ScoreManager : MonoBehaviour
{
   public int player1Score = 0;
   public int player2Score = 0;

   public void UpdateScore(int player, bool regionCaptured)
   {
      if (regionCaptured)
      {
         if (player == 1)
         {
            player1Score += 500;
            player2Score -= 500;
         }
         else
         {
            player2Score += 500;
            player1Score -= 500;
         }
      }

      Debug.Log($"Очки: Гравець 1 - {player1Score}, Гравець 2 -
{player2Score}");
   }
}
```

Гравці по черзі відповідають

```
public class TurnManager : MonoBehaviour
{
   public int currentPlayer = 1;

   public void EndTurn()
   {
      currentPlayer = (currentPlayer == 1) ? 2 : 1;
      Debug.Log($"Хід гравця {currentPlayer}");
   }
}
```

Один відповідає, другий спостерігає (якщо обидва не борються за регіон)

```
public class ObserverMode : MonoBehaviour
{
   public void ShowObserverView(int observingPlayer)
   {
      Debug.Log($"Гравець {observingPlayer} спостерігає...");
   }
}
```

Закінчення гри

```
public class GameOverManager : MonoBehaviour
{
   public void CheckGameOver(int player1Regions, int player2Regions, int
totalRegions)
   {
      if (player1Regions + player2Regions == totalRegions)
      {
         string winner = (player1Regions > player2Regions) ? "Гравець 1" : "Гравець
2";
         Debug.Log($"Гра закінчена, {winner} виграв!");
      }
   }
}
```

Створення скрипту для запуску серверу, підключення клієнта або створення локального хоста.

```
using Mirror;
using UnityEngine;

public class MultiplayerManager : MonoBehaviour
{
   public NetworkManager networkManager;

   public void StartHost()
   {
      Debug.Log("Запуск хоста...");
      networkManager.StartHost(); // Запускає сервер і клієнта одночасно
   }

   public void StartClient()
   {
      Debug.Log("Підключення до сервера...");
      networkManager.StartClient(); // Підключення клієнта до сервера
```

```
    }

    public void StartServer()
    {
        Debug.Log("Запуск сервера...");
        networkManager.StartServer(); // Тільки сервер
    }
}
```

Додавання кнопки "Хост", "Підключитись" і "Сервер" для керування мережевими з'єднаннями.

```
using UnityEngine;

public class MultiplayerMenu : MonoBehaviour
{
    public MultiplayerManager multiplayerManager;

    public void OnHostButtonPressed()
    {
        multiplayerManager.StartHost();
    }

    public void OnClientButtonPressed()
    {
        multiplayerManager.StartClient();
    }

    public void OnServerButtonPressed()
    {
        multiplayerManager.StartServer();
    }
}
```

Скрипт для регіонів із синхронізацією

```
using Mirror;
using UnityEngine;

public class NetworkRegion : NetworkBehaviour
{
    [SyncVar] public bool isCaptured = false; // Стан регіону синхронізується
    [SyncVar] public Color regionColor; // Колір регіону синхронізується
    private SpriteRenderer spriteRenderer;
```

```
    void Start()
    {
        spriteRenderer = GetComponent<SpriteRenderer>();
        spriteRenderer.color = regionColor; // Встановлення кольору
    }

    [Server] // Метод доступний тільки на сервері
    public void CaptureRegion()
    {
        if (!isCaptured)
        {
            isCaptured = true;
            regionColor = new Color(Random.value, Random.value, Random.value); //
Рандомний колір
            RpcUpdateRegionColor(regionColor);
            Debug.Log($"Регіон захоплено: {regionColor}");
        }
    }

    [ClientRpc] // Оновлення на всіх клієнтах
    void RpcUpdateRegionColor(Color color)
    {
        spriteRenderer.color = color;
    }
}
```

Коли гравець натискає на регіон, запит відправляється на сервер

```
using Mirror;
using UnityEngine;

public class NetworkRegionInteraction : NetworkBehaviour
{
    void OnMouseDown()
    {
        if (isLocalPlayer) // Перевірка: тільки локальний гравець
        {
            CmdTryCaptureRegion();
        }
    }

    [Command] // Відправка запиту на сервер
    void CmdTryCaptureRegion()
    {
```

```
        GetComponent<NetworkRegion>().CaptureRegion(); // Захоплення регіону
    }
}
```

Коли один із гравців захоплює всі регіони, гра завершується

```
using Mirror;
using UnityEngine;

public class GameOverManager : NetworkBehaviour
{
    [SyncVar] public int player1Regions = 0;
    [SyncVar] public int player2Regions = 0;
    public int totalRegions;

    void Update()
    {
        if (player1Regions + player2Regions == totalRegions)
        {
            RpcEndGame(player1Regions > player2Regions ? 1 : 2);
        }
    }

    [ClientRpc]
    void RpcEndGame(int winningPlayer)
    {
        if (winningPlayer == 1)
        {
            Debug.Log("Гра завершена: Гравець 1 виграв!");
        }
        else
        {
            Debug.Log("Гра завершена: Гравець 2 виграв!");
        }
    }
}
```

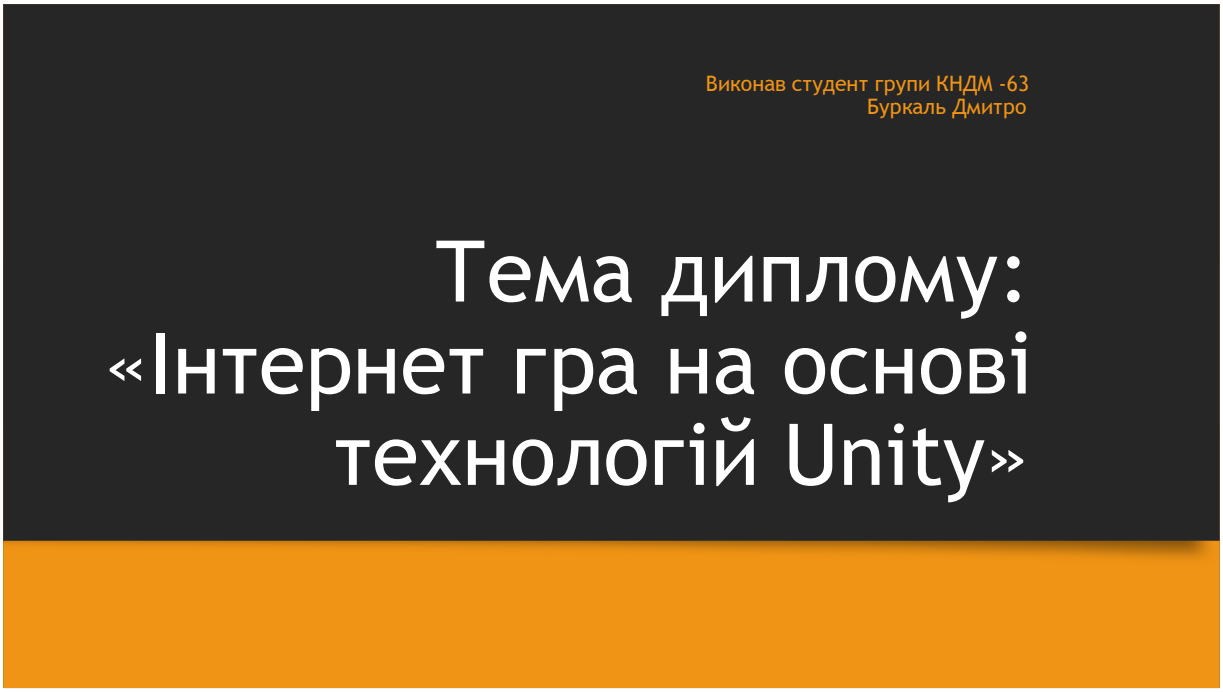
Скрипт для додавання фону:

```
using UnityEngine;

public class BackgroundManager : MonoBehaviour
{
    public SpriteRenderer backgroundRenderer; // SpriteRenderer для фону
```

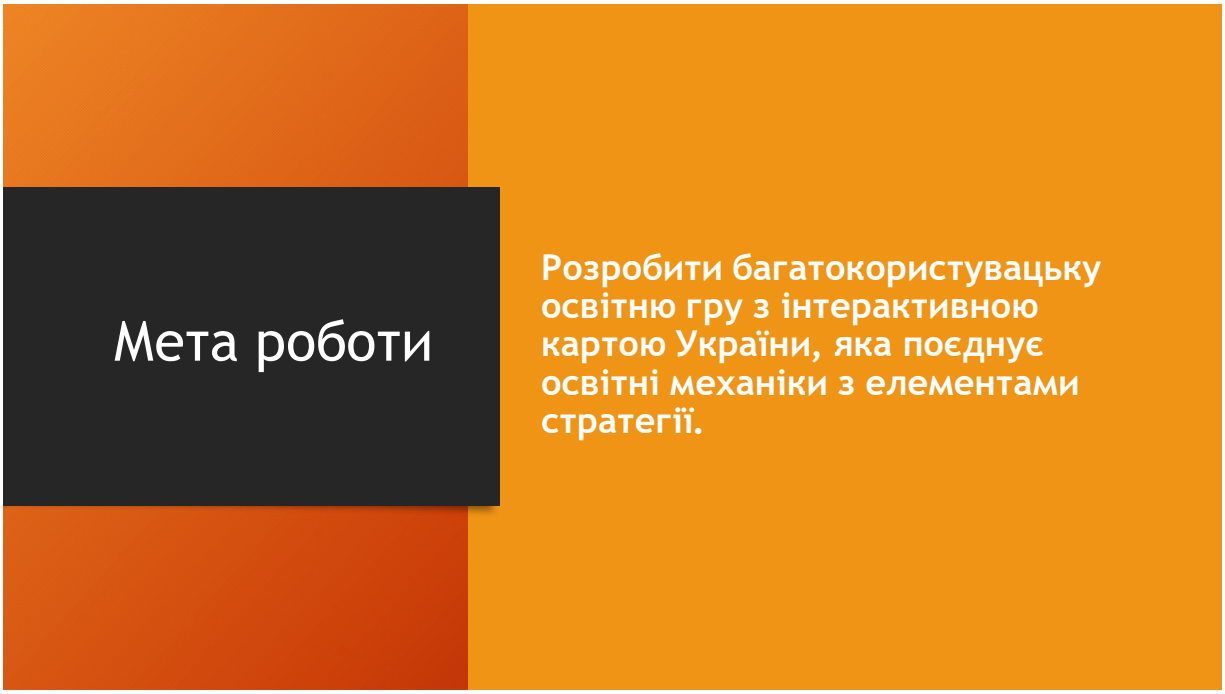
```
    public Sprite[] backgroundImages; // Массив фонових зображень

    public void SetBackground(int index)
    {
        if (index >= 0 && index < backgroundImages.Length)
        {
            backgroundRenderer.sprite = backgroundImages[index]; // Встановлюємо
фон
            Debug.Log($"Фон змінено на: {backgroundImages[index].name}");
        }
        else
        {
            Debug.LogWarning("Індекс фону виходить за межі масиву!");
        }
    }
}
```

ДОДАТОК Б

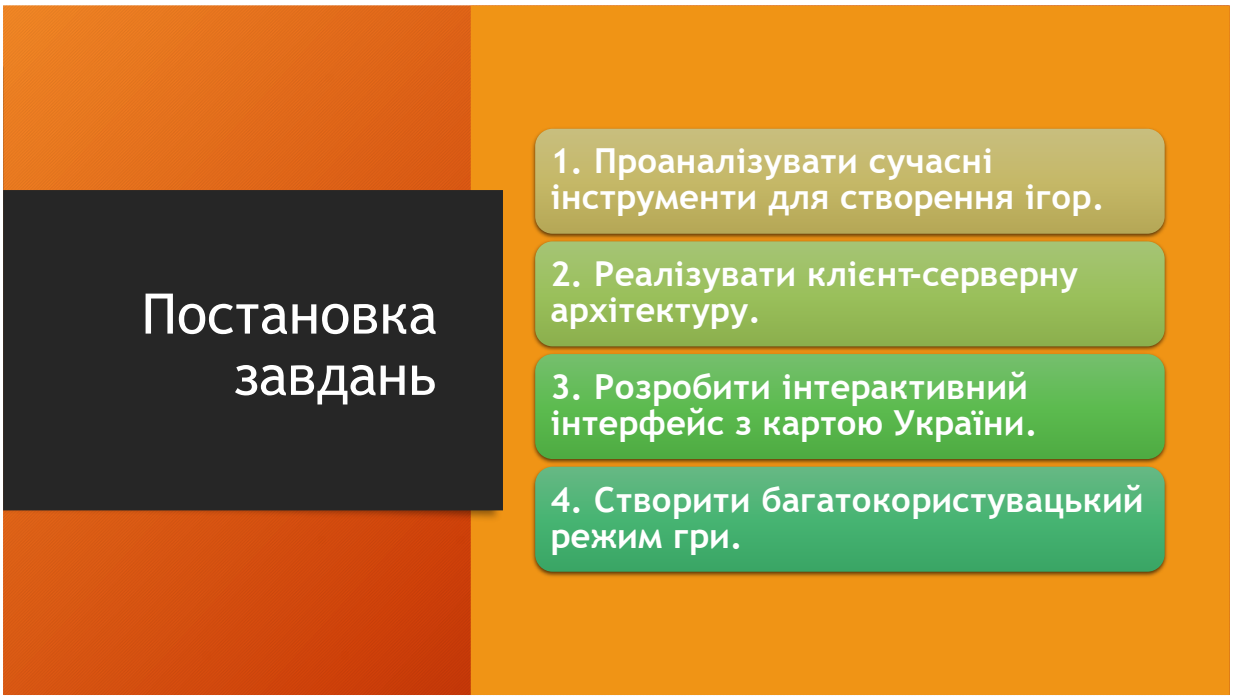
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



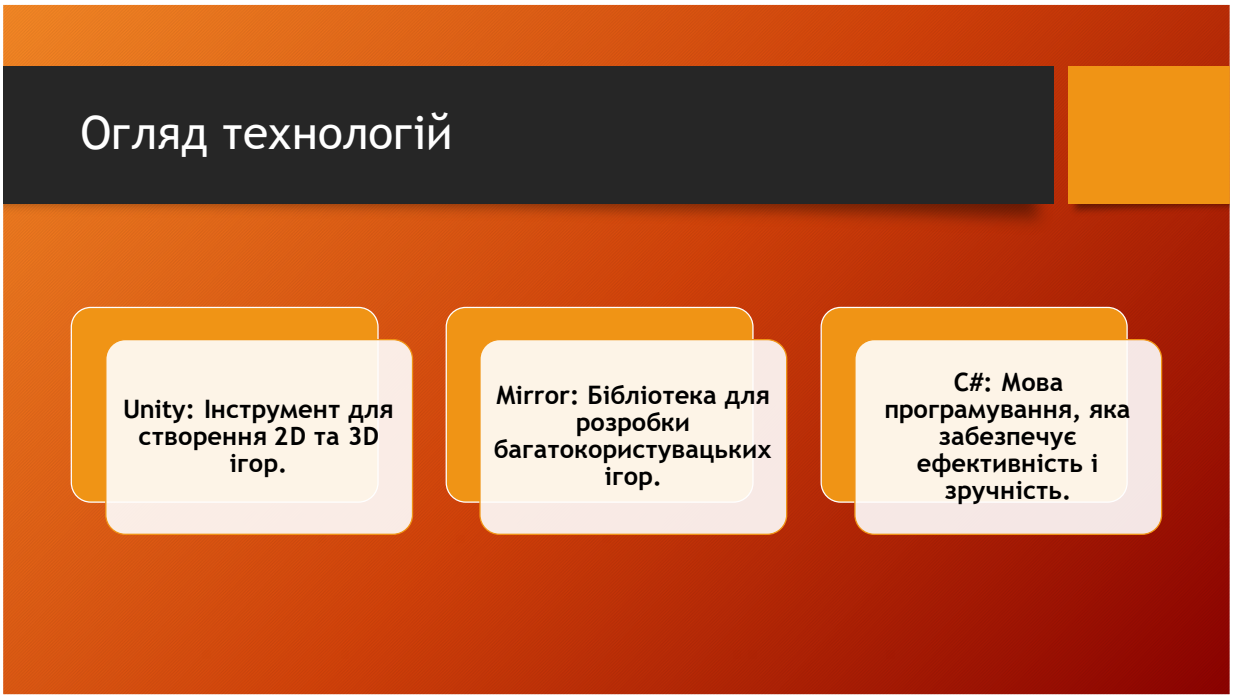
Слайд 1



Слайд 2



Слайд 3



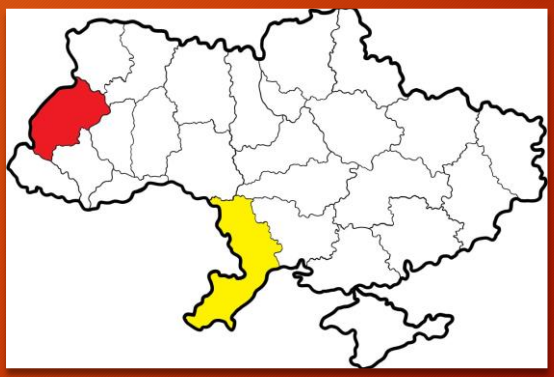
Слайд 4

Опис гри

Ціль гри: гравці змагаються за контроль регіонів, відповідаючи на запитання.

Головна механіка: інтерактивна карта України з підрахунком балів.

Цільова аудиторія: студенти, молодь, освітні заклади.



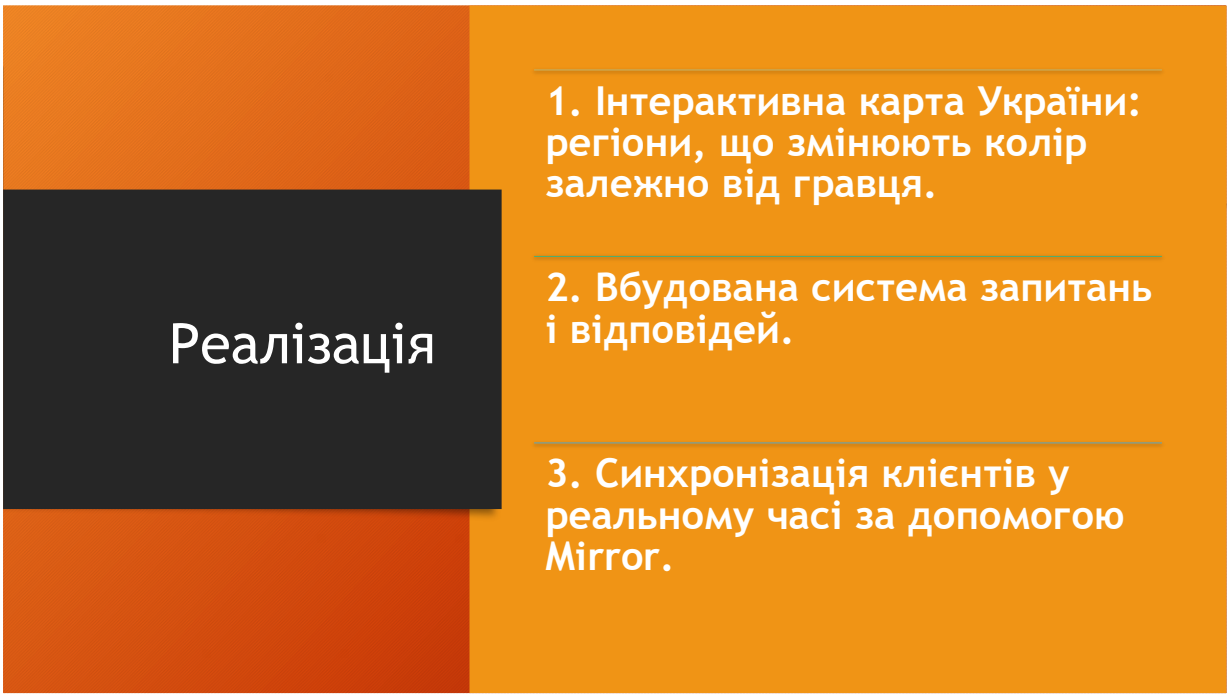
Слайд 5

Архітектура гри

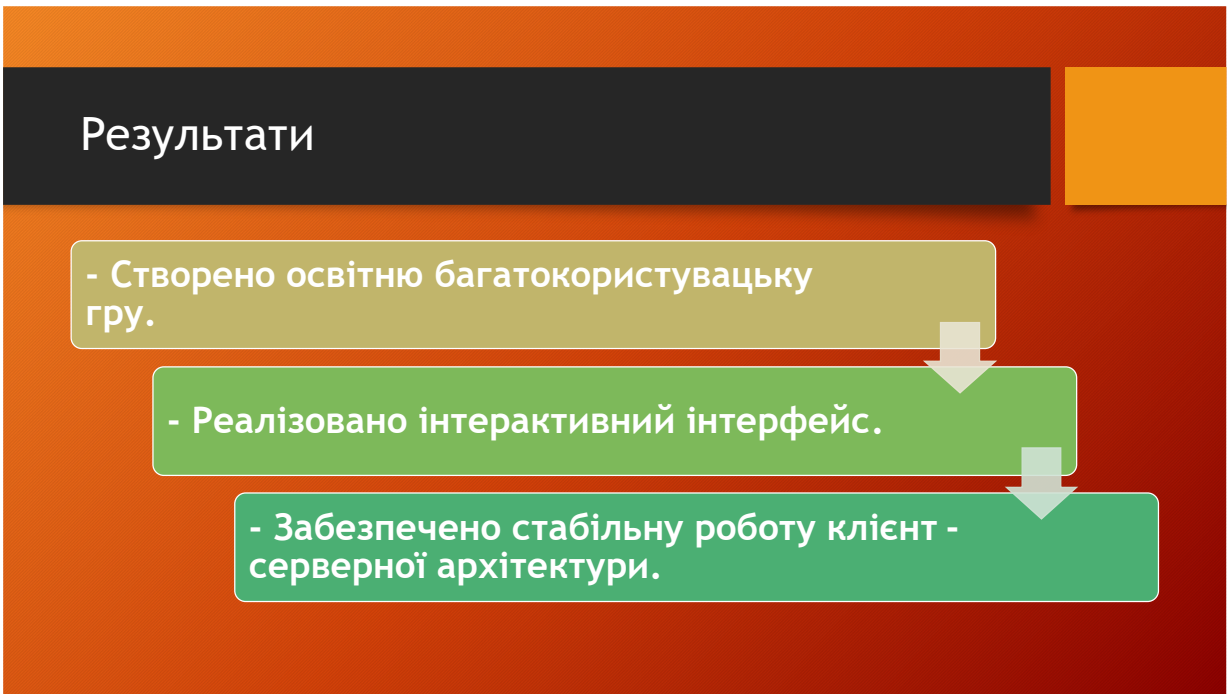
Гра побудована на клієнт-серверній моделі:

- **Сервер обробляє всі дії та синхронізує стан гри.**
- **Клієнти виконують дії гравців та отримують оновлення в реальному часі.**

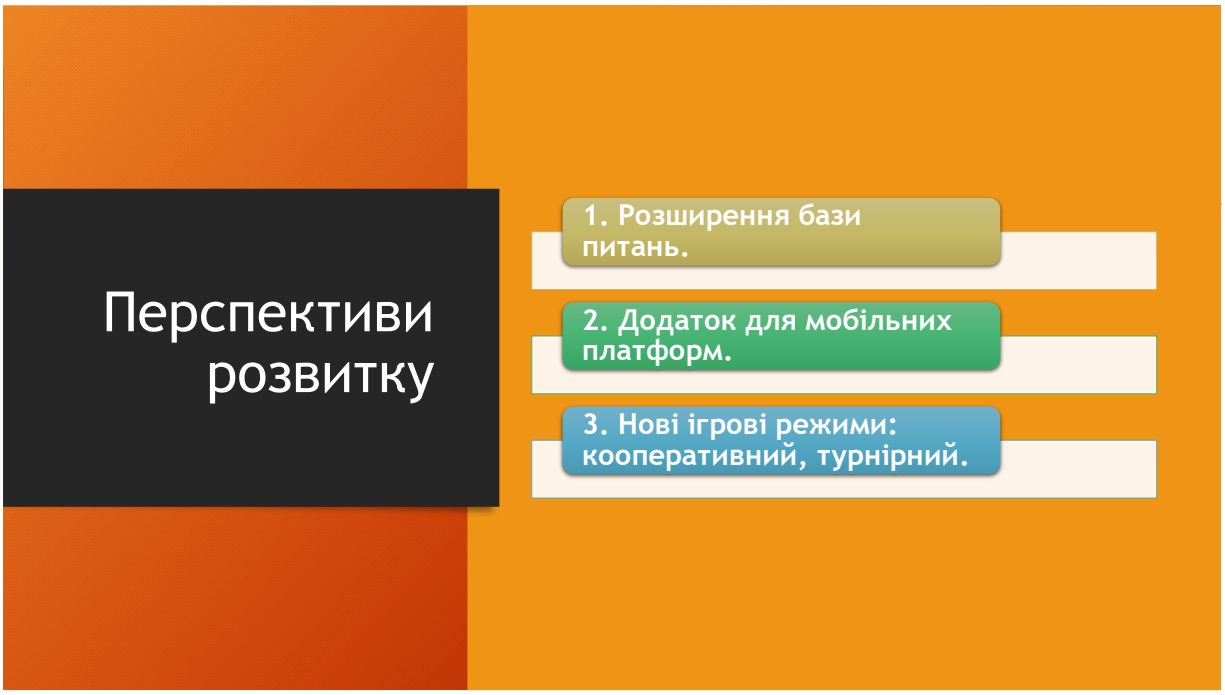
Слайд 6



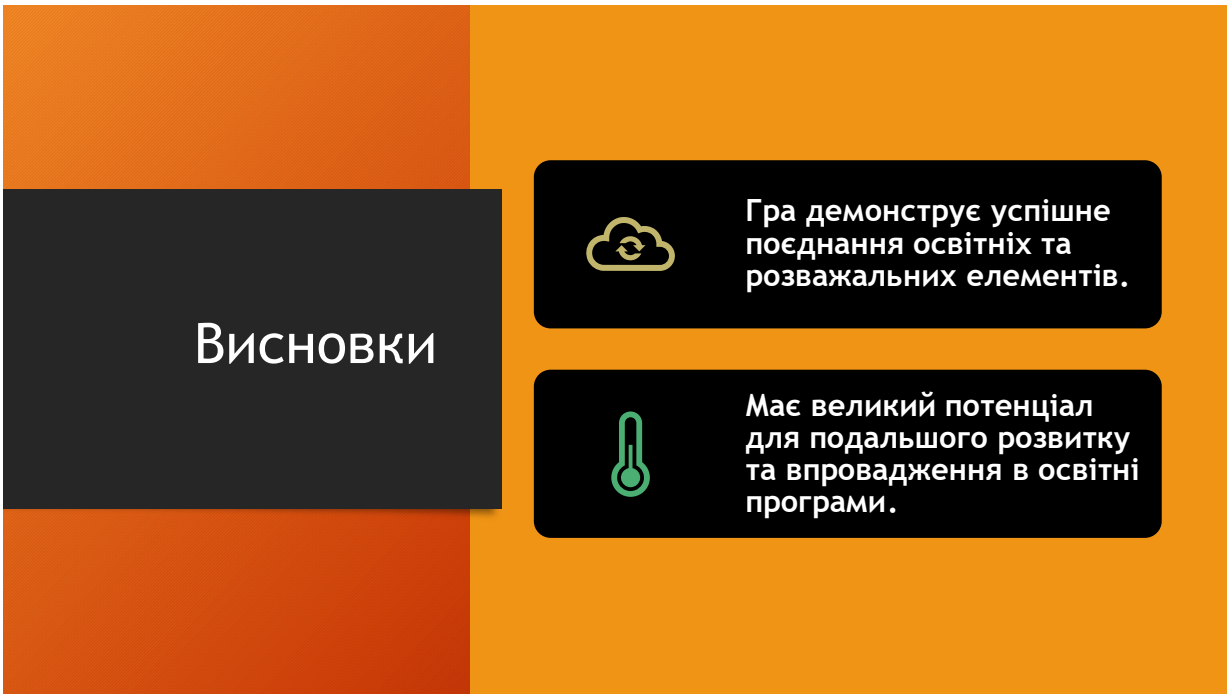
Слайд 7



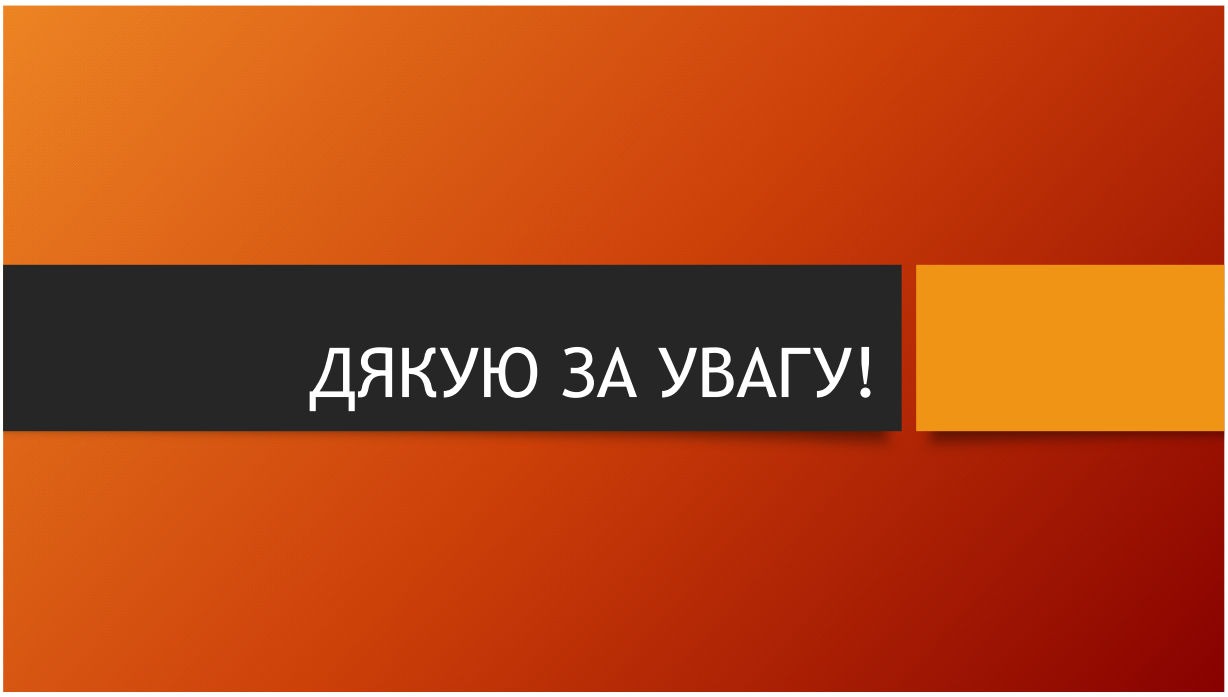
Слайд 8



Слайд 9



Слайд 10



Слайд 11