Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Навчальне програмне забезпечення для занурення користувача в історичний контекст через інтерактивні сценарії та симуляції»

Виконав: студент IV курсу
групи ЗПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кім Т.О.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Чехместрук Р. Ю.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Дудник О. В.
(прізвище та ініціали)

Допущено до захисту
Зав. Кафедри ПЗ Романюк О.Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 – Інформаційні технології

Спеціальність 121 – Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

О. Н. Романюк

"21" березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кім Тетяні Олексіївній

(прізвище, ім'я, по батькові)

1. Тема роботи – Навчальне програмне забезпечення для занурення користувача в історичний контекст через інтерактивні сценарії та симуляції

Керівник роботи Чехместрук Роман Юрійович, к.т.н., доцент кафедри ПЗ,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "20" березня 2025 р. №97.

2. Термін подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: формалізація моделі інтерактивного історичного навчання – до 20 основних історичних сценаріїв та подій, кількість вимог щодо розробки ігрових модулів системи – не більш ніж 30, загальна кількість історичних об'єктів та персонажів в базі даних – до 50000, швидкість обробки ігрових подій та оновлення інтерфейсу – не менш ніж 60 кадрів за секунду, ефективність засвоєння знань користувачами – покращення на 25% порівняно з традиційними методами навчання.

4. Зміст розрахунково-пояснювальної записки: вступ, аналіз сучасного стану освітніх ігрових технологій та гейміфікації в історичній освіті, огляд існуючих аналогів та вебтехнологій для розробки навчальних ігор, теоретичні основи створення інтерактивних історичних симуляцій, алгоритми обробки користувацьких дій та адаптації контенту, розробка ігрової системи з використанням JavaScript/TypeScript і Phaser.js, архітектура та реалізація модулів управління станом і підрозділами, проектування інтуїтивного графічного інтерфейсу, тестування програмного забезпечення та оцінка ефективності, висновки, перелік посилань, додатки.

5. Перелік графічного матеріалу: мета та задачі роботи, порівняльна характеристика існуючих аналогів освітніх ігор, архітектура модульної системи програмного забезпечення, блок-схема алгоритмів управління ігровими об'єктами та сценаріями, інтерфейс користувача та ігрові екрани, результати тестування системи та оцінки ефективності навчання, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Чехместрюк Р. Ю., к.т.н., доцент кафедри ПЗ	24.03.25р.	01.06.25р.

7. Дата видачі завдання 13 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу виконання бакалаврської кваліфікаційної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз стану систем ігрового типу для занурення користувача в історичний контекст	25.03.25	31.03.25	Викон.
2	Аналіз аналогів системи ігрового типу для занурення користувача в історичний контекст	03.04.25	17.04.25	Викон.
3	Розробка алгоритмів та модулів web-застосунку	14.04.25	21.04.25	Викон.
4	Тестування програми	23.04.25	27.05.24	Викон.
5	Оформлення матеріалів до захисту БКР	28.05.25	30.05.25	Викон

Студент _____ Кім Т.О.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Чехместрюк Р. Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.04

Бакалаврська кваліфікаційна робота складається з 107 сторінок формату А4, на яких є 35 рисунків, 3 таблиці, список використаних джерел містить 30 найменувань.

У бакалаврській кваліфікаційній роботі було розроблено інноваційне програмне забезпечення ігрового типу для інтерактивного вивчення історії з використанням сучасних вебтехнологій. Проведено комплексний аналіз сучасних технологій навчальних програмних забезпечень та порівняльний аналіз існуючих аналогів з виявленням їх переваг і недоліків. Обґрунтовано необхідність розробки спеціалізованого програмного забезпечення для підвищення ефективності засвоєння історичних знань через інтерактивне занурення користувачів у історичний контекст. Розроблено модуль управління станом ігрових об’єктів для реалізації динамічних сценаріїв та модуль управління підрозділами в ігровому середовищі для симуляції історичних процесів. Створено інтуїтивний графічний інтерфейс користувача з урахуванням принципів юзабіліті та запропоновано нову архітектуру модульної системи для освітніх ігор історичного спрямування. На основі проведених досліджень імплементовано програмне рішення з використанням JavaScript/TypeScript та фреймворку Phaser.js. Проведене комплексне тестування підтвердило ефективність розробленої системи та її потенціал для впровадження в освітніх закладах. Розроблені засоби інтерактивного навчання можна використати в школах, коледжах та університетах як ефективний інструмент підвищення якості історичної освіти та розвитку критичного мислення студентів.

Ключові слова: гейміфікація, освітнє програмне забезпечення, інтерактивне навчання, історична освіта, вебтехнології, JavaScript, TypeScript, Phaser.js, ігрові технології, навчальні симуляції.

ABSTRACT

The bachelor's thesis consists of 107 A4 pages, which have 35 figures, 3 tables, the list of sources used contains 30 items.

In the bachelor's thesis, innovative game-like educational software for interactive history learning using modern web technologies was developed. A comprehensive analysis of current educational software technologies of the game type and a comparative analysis of existing analogs were conducted, identifying their advantages and disadvantages. The need to develop specialized software to enhance the effectiveness of history learning through interactive immersion in the historical context was substantiated. A module for managing the state of game objects was developed to implement dynamic scenarios, as well as a module for managing units in the game environment to simulate historical processes. An intuitive graphical user interface was created based on usability principles, and a new modular system architecture for educational games with a historical focus was proposed. Based on the conducted research, a software solution was implemented using JavaScript/TypeScript and the Phaser.js framework. Comprehensive testing confirmed the effectiveness of the developed system and its potential for implementation in educational institutions. The developed interactive learning tools can be used in schools, colleges, and universities as an effective means to improve the quality of history education and foster critical thinking among students.

Keywords: gamification, educational software, interactive learning, history education, web technologies, JavaScript, TypeScript, Phaser.js, game technologies, educational simulations.

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ 8
1.1 Аналіз стану технологій навчальних програмних забезпечень ігрового типу 8
1.2 Порівняльний аналіз аналогів 10
1.3 Аналіз методів розв’язання задачі 18
1.4 Постановка задач 20
1.5 Висновки 22
2 АНАЛІЗ СТРУКТУРИ АЛГОРИТМІВ ТА ФОРМУВАННЯ ПРОБЛЕМАТИКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 23
2.1 Аналіз та дослідження проблематики 23
2.2 Нова структура інтеграції програмних модулів 26
2.3 Аналіз та розробка інтерактивних можливостей програмного застосунку 28
2.4 Розробка загальної моделі застосунку 31
2.5 Дослідження алгоритмів роботи системи 33
2.6 Висновки 37
3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІГРОВОГО ТИПУ 38
3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення 38
3.2 Метод модульної архітектури для навчального програмного забезпечення 42
3.3 Модуль зміни стану об’єктів 44
3.4 Модуль управління підрозділами в місті 47
3.5 Створення комплексного підходу до розробки графічного інтерфейсу 49
3.6 Висновки 52
4 ТЕСТУВАННЯ ПРОГРАМИ 54

4.1 Тестування системи 54
4.2 Розробка інструкції користувача 58
4.3 Висновки 61
ВИСНОВКИ 63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 65
ДОДАТКИ 69
Додаток А – Технічне завдання ..... **Ошибка! Закладка не определена.**
Додаток Б – Протокол перевірки на плагіат 73
Додаток В – Лістинг програмного забезпечення 73
Додаток Г – Ілюстративна частина 100

ВСТУП

Обґрунтування вибору теми дослідження

Сучасна освітня система зазнає кардинальних трансформацій під впливом цифрових технологій та зміни підходів до організації навчального процесу. Традиційні методи викладання історії, засновані на пасивному сприйнятті інформації, все частіше виявляються неефективними для формування глибокого розуміння історичних процесів та розвитку критичного мислення у студентів [1].

Гейміфікація навчального процесу набуває все більшого поширення як інноваційний підхід до освіти, що дозволяє підвищити мотивацію та залученість учнів. Особливо перспективним є застосування ігрових технологій у викладанні історії, оскільки інтерактивні сценарії та симуляції дають можливість користувачам безпосередньо «проживати» історичні події та приймати рішення в контексті минулих епох.

Аналіз ринку освітнього програмного забезпечення виявляє суттєві прогалини: брак високоякісних ігрових продуктів історичної спрямованості, обмежені можливості інтерактивної взаємодії, складність адаптації до різних освітніх рівнів та недосконаліть систем оцінювання навчальних досягнень. Більшість існуючих рішень не забезпечують необхідного балансу між історичною достовірністю та ігровою привабливістю.

Розробка спеціалізованого програмного забезпечення для інтерактивного вивчення історії з використанням сучасних вебтехнологій: JavaScript/TypeScript, Phaser.js, відповідає актуальним потребам освітньої сфери та тенденціям її цифровізації.

Зв’язок роботи з науковими програмами, планами, темами

Дослідження виконано в рамках наукової діяльності кафедри програмного забезпечення за напрямком розробки інноваційних освітніх технологій та створення інтерактивних навчальних систем.

Мета та завдання дослідження. Метою роботи є підвищення ефективності засвоєння історичних знань шляхом розробки та впровадження інноваційного програмного забезпечення ігрового типу, що забезпечує інтерактивне занурення користувачів у історичний контекст та сприяє розвитку критичного мислення.

Завдання дослідження:

- здійснити комплексний аналіз сучасних технологій навчальних програмних забезпечень та виявити основні тенденції їх розвитку;
- провести порівняльний аналіз існуючих аналогів та визначити їх переваги і недоліки;
- дослідити методи розв’язання задач створення освітніх ігор з історичним контентом;
- розробити модуль управління станом ігрових об’єктів для реалізації динамічних сценаріїв;
- імплементувати модуль управління підрозділами в ігровому середовищі для симуляції історичних процесів;
- розробити інтуїтивний графічний інтерфейс користувача з урахуванням принципів юзабіліті;
- провести комплексне тестування системи та оцінити її ефективність;
- розробити методичні рекомендації щодо практичного використання програмного забезпечення.

Об’єкт дослідження

Процеси розробки та впровадження інноваційного навчального програмного забезпечення ігрового типу для інтерактивного вивчення історії.

Предмет дослідження

Методи, алгоритми та технологічні рішення створення освітнього програмного забезпечення ігрового типу з використанням вебтехнологій JavaScript/TypeScript та фреймворку Phaser.js для забезпечення інтерактивного історичного досвіду.

Методи дослідження

Для досягнення поставленої мети використовувалися: системний підхід для аналізу предметної галузі освітніх ігрових технологій; методи об’єктно-орієнтованого проектування для розробки архітектури системи; алгоритми обробки користувацьких даних та адаптації контенту; методи тестування програмного забезпечення для забезпечення якості продукту; сучасні вебтехнології для створення інтерактивних компонентів; методи емпіричного дослідження для оцінки ефективності розробленого рішення.

Наукова новизна отриманих результатів

1. Розроблено метод модульної архітектури для навчального програмного забезпечення ігрового типу, що поєднує систему відображення історичного контенту, механізми зміни стану об'єктів та управління підрозділами для створення цілісного інтерактивного середовища занурення користувача в історичний контекст.

2. Запропоновано нову структуру інтеграції програмних модулів освітньої гри історичного спрямування, що характеризується незалежністю компонентів відображення інформації, управління об'єктами та графічного інтерфейсу, забезпечуючи високу адаптивність системи до різних історичних сценаріїв та навчальних завдань.

3. Створено комплексний підхід до розробки графічного інтерфейсу навчальних ігор історичної тематики, який інтегрує модулі управління підрозділами, відображення обладнання та зміни стану об'єктів в єдину систему, що максимізує ефективність взаємодії користувача з історичним контентом через інтерактивні симуляції.

Практичне значення отриманих результатів

Розроблене програмне забезпечення має високий потенціал для впровадження в освітніх закладах різних рівнів як ефективний інструмент підвищення якості історичної освіти. Модульна архітектура системи забезпечує адаптацію до специфічних потреб різних категорій користувачів та освітніх контекстів.

Програмний продукт може застосовуватися як у формальному навчальному процесі (школи, коледжі, університети), так і для самостійного вивчення історії, що робить його універсальним засобом історичної освіти. Незалежність розроблених модулів дозволяє їх окреме використання в інших освітніх проектах.

Особистий внесок здобувача

Всі наукові результати та практичні розробки, представлені в роботі, виконані автором самостійно. Здобувачем особисто проведено аналіз предметної галузі, розроблено системну архітектуру, створено програмні модулі, здійснено тестування та оцінку ефективності розробленого програмного рішення. Наукову роботу було розглянуто на конференції Modern Perspectives on Science and Economic Progress.

Апробація результатів дослідження

Основні положення та результати кваліфікаційної роботи представлені на науково-практичних конференціях https://isu-conference.com/ студентів та молодих дослідників з проблематики сучасних освітніх технологій та інноваційних підходів до розробки програмного забезпечення.

Публікації

За результатами дослідження опубліковано наукові праці, що висвітлюють основні аспекти розробки навчального програмного забезпечення ігрового типу та його практичного застосування в історичній освіті.

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз стану технологій навчальних програмних забезпечень ігрового типу

Ринок освітнього програмного забезпечення демонструє стрімке зростання, досягнувши у 2024 році обсягу понад 16 мільярдів доларів. Сегмент історичних освітніх ігор становить близько 12% від загального обсягу ринку освітніх технологій, що свідчить про значний потенціал та зростаючий інтерес до інтерактивних методів вивчення історії. За прогнозами аналітиків, цей тренд посилюватиметься, і до 2027 року обсяг ринку історичних освітніх ігор може збільшитися вдвічі.

Така трансформація пов’язана із загальною тенденцією гейміфікації освіти, зростаючою потребою в персоналізованому навчанні та усвідомленням ефективності інтерактивних методів для засвоєння комплексних історичних знань. Нове покоління учнів та студентів демонструє вищу залученість та кращі результати при використанні ігрових підходів до вивчення історичних подій та контексту [2].

Сучасні навчальні програмні забезпечення ігрового типу для вивчення історії базуються на передових технологічних рішеннях. Мультимодальна архітектура забезпечує адаптивність та різноманітність навчального досвіду через поєднання різних форм подання історичного матеріалу. Технології процедурної генерації контенту дозволяють створювати унікальні історичні сценарії, що адаптуються до індивідуальних потреб кожного користувача. Системи адаптивного навчання, побудовані на базі алгоритмів машинного навчання, забезпечують персоналізацію складності та типу контенту залежно від рівня знань та прогресу користувача [3].

На стороні клієнта домінують сучасні ігрові рушії та фреймворки, як-от Phaser.js, Unity та Unreal Engine, що дозволяють створювати імерсивні та інтерактивні історичні середовища. Технології WebGL та HTML5 Canvas забезпечують можливість розробки кросплатформних рішень, доступних через

веббраузер без необхідності встановлення додаткового програмного забезпечення. Фреймворк Phaser.js став особливо популярним для розробки освітніх ігор завдяки своїй оптимізованості, багатому набору інструментів для 2D-ігор та відмінній документації. Технології віртуальної та доповненої реальності поступово інтегруються в освітні ігри, пропонуючи безпрецедентний рівень занурення у відтворені історичні епохи та події [4].

Аналіз сучасного стану технологій навчальних програмних забезпечень ігрового типу для вивчення історії виявляє суттєві можливості для інновацій та вдосконалення. Розробка власного програмного забезпечення є обґрунтованою з огляду на кілька критичних факторів.

Технологічна еволюція освітніх методик створює унікальне вікно можливостей для впровадження інноваційних рішень. Перехід до персоналізованого та адаптивного навчання відкриває простір для програмних забезпечень, які краще відповідають сучасним освітнім потребам та очікуванням користувачів. Існуючі технологічні обмеження можуть бути подолані за допомогою сучасних фреймворків, таких як Phaser.js для розробки інтерактивного ігрового середовища та TypeScript для забезпечення надійності та масштабованості коду.

Зміни в підходах до навчання історії також підтверджують необхідність розробки нових рішень. Сучасна історична освіта орієнтується на формування критичного мислення, розвиток емпатії та розуміння історичного контексту, а не лише на запам’ятовування дат та фактів. Власне програмне забезпечення може бути спроектоване з урахуванням цих освітніх принципів, забезпечуючи більш глибоке розуміння історичних процесів через симуляції та інтерактивні сценарії [5].

Сучасні вебтехнології відкривають значні можливості для технологічних інновацій у сфері освітніх ігор. Phaser.js та TypeScript дозволяють створювати продуктивні, адаптивні та захоплюючі ігрові середовища, які можуть забезпечити ефективне засвоєння історичних знань. Ці технології підтримують

швидку ітерацію розробки та експерименти з різними механіками занурення користувача в історичний контекст.

Отже, аналіз сучасного стану технологій навчальних програмних забезпечень ігрового типу для вивчення історії свідчить про динамічний розвиток цієї галузі та наявність суттєвих можливостей для інновацій. Технологічні тренди, включаючи адаптивне навчання, процедурну генерацію контенту, використання елементів віртуальної реальності та розширені симуляційні можливості, формують нове покоління освітніх ігрових платформ.

Водночас існують значні технологічні виклики, пов’язані з балансуванням ігрового та освітнього компонентів, забезпеченням історичної достовірності, адаптацією до різних вікових груп та оцінюванням навчальних результатів. Ці виклики створюють можливості для розробки нових, більш ефективних рішень у сфері історичної освіти.

1.2 Порівняльний аналіз аналогів

Сучасний ринок навчального програмного забезпечення для вивчення історії представлений кількома помітними продуктами, кожен з яких має свої технологічні особливості, переваги та недоліки. Для обґрунтування доцільності розробки власного навчального програмного забезпечення ігрового типу необхідно провести детальний аналіз існуючих аналогів, виявити їхні сильні та слабкі сторони, а також визначити потенційні напрями для інновацій та вдосконалень. До найбільш відомих рішень відносять:

- Civilization VI;
- Assassin’s Creed Discovery Tour;
- Historia: Game of Claims;
- Mission US.

Civilization VI, розроблена Firaxis Games, залишається одним із найпопулярніших стратегічних симуляторів з історичною складовою, з понад 8 мільйонами користувачів та значним освітнім потенціалом. Гра пропонує

глибоке занурення в еволюцію цивілізацій та розвиток технологій від давніх часів до сучасності (див. рисунок 1.1).

Рисунок 1.1 – Приклад роботи гри Civilization VI

Civilization VI побудована на власному рушії компанії Firaxis, який розроблявся протягом багатьох років. Гра використовує складні алгоритми для моделювання історичних процесів, технологічних дерев та соціальних інститутів. Система «tech tree» відображає історичну послідовність наукових відкриттів та їхній вплив на розвиток цивілізацій [6]. Програма включає розширену систему процедурної генерації карт світу та сценаріїв, що забезпечує високу реіграбельність.

Civilization VI пропонує глибоке занурення в історичні процеси через симуляцію розвитку цивілізацій, включаючи технологічний прогрес, культурний розвиток та дипломатичні відносини. Гра включає енциклопедичну базу даних про історичних лідерів, технології та чудеса світу. Система «Civlopedia» надає доступ до детальної інформації про історичний контекст різних елементів гри. Модифікаційна підтримка дозволяє розширювати освітній потенціал гри через додатковий контент.

Незважаючи на освітній потенціал, Civilization VI є передусім розважальним продуктом, що часто нехтує історичною точністю на користь ігрового балансу. Складність ігрових механік створює високий поріг входу для освітнього використання, особливо для молодших вікових груп. Відсутність структурованих навчальних цілей та системи оцінювання знань обмежує можливості цілеспрямованого використання в освітньому процесі. Гра також не пропонує персоналізованих освітніх траєкторій та адаптації до різних рівнів знань [7].

Assassin’s Creed Discovery Tour, розроблений Ubisoft, представляє інноваційний підхід до історичної освіти через інтерактивні екскурсії історичними локаціями, відтвореними з високою точністю.

Discovery Tour побудований на рушії AnvilNext 2.0 компанії Ubisoft, який забезпечує фотореалістичне відтворення історичних локацій. Програма використовує технології 3D-сканування реальних історичних артефактів та архітектурних споруд для максимальної автентичності [8]. Система навігації та інтерактивних точок інтересу дозволяє структурувати освітній процес у відкритому світі. Discovery Tour також включає систему курованих екскурсій, розроблених у співпраці з історичними консультантами та освітніми установами.

Discovery Tour пропонує безпрецедентний рівень візуальної достовірності історичних локацій, включаючи архітектуру, одяг, предмети побуту та мистецтва (див. рисунок 1.2). Програма надає доступ до курованих екскурсій, що охоплюють різні аспекти повсякденного життя, політики, релігії та культури відповідних історичних періодів. Інтерактивні елементи дозволяють користувачам досліджувати історичні артефакти та локації у власному темпі. Програма включає систему досягнень та квізів для перевірки засвоєння матеріалу.

Discovery Tour обмежений кількома історичними періодами та регіонами (Давній Єгипет, Греція, Рим, Середньовічна Англія), що звужує освітній потенціал. Лінійність екскурсій обмежує можливості для дослідження альтернативних історичних сценаріїв та причинно-наслідкових зв’язків [9].

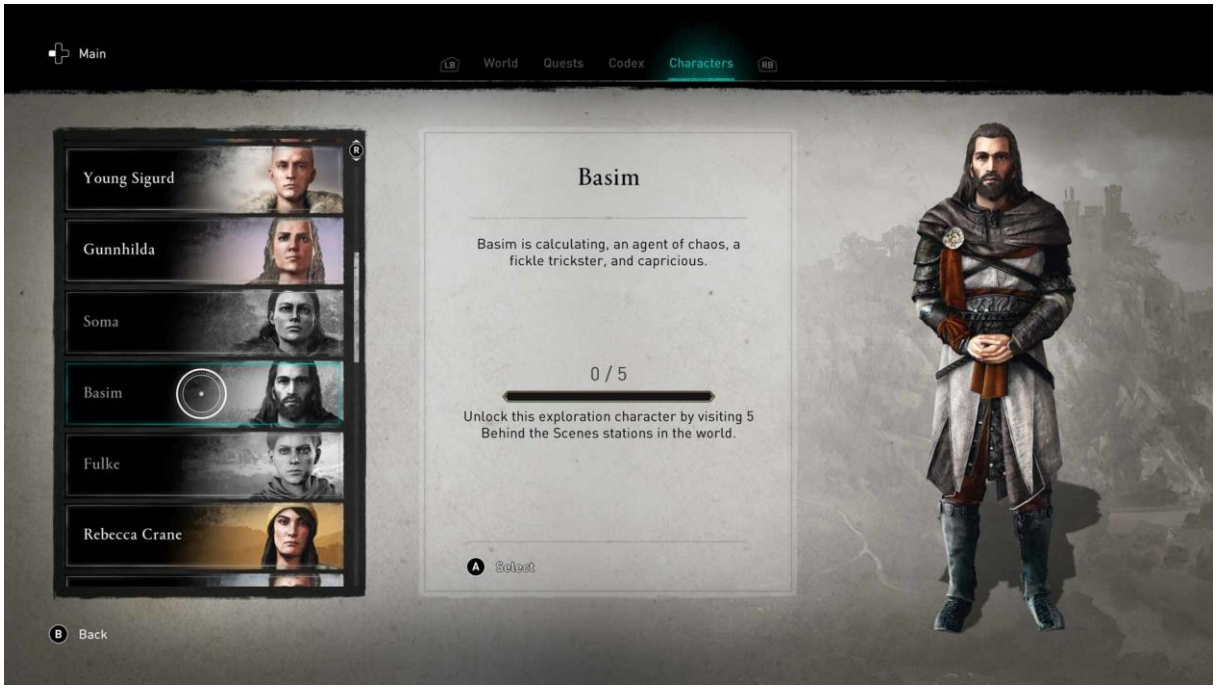

Рисунок 1.2 – Приклад роботи Assassin’s Creed Discovery Tour

Програма також має обмежені можливості для активної взаємодії з історичним контекстом, фокусуючись на пасивному споживанні інформації. Висока апаратна вимогливість обмежує доступність програми для освітніх закладів.

Historia: Game of Claims, розроблена освітньою компанією McGraw-Hill Education, позиціонується як спеціалізоване навчальне програмне забезпечення для вивчення історії через інтерактивні сценарії та дебати.

Historia побудована на вебтехнологіях, включаючи HTML5, JavaScript та WebGL, що забезпечує крос-платформність та доступність через браузер [7]. Програма використовує адаптивні алгоритми для персоналізації навчального досвіду на основі прогресу та результатів користувача. Historia включає систему процедурної генерації сценаріїв дебатів та історичних дилем, що забезпечує різноманітність навчального досвіду. Платформа також інтегрує аналітичні інструменти для відстеження прогресу учнів та ефективності навчального процесу (див. рисунок 1.3).

Рисунок 1.3 – Приклад роботи Historia: Game of Claims

Historia фокусується на розвитку критичного мислення та аргументації через систему дебатів на історичні теми. Програма включає детальні історичні джерела та документи для аналізу та інтерпретації. Адаптивна система навчання персоналізує складність та тип завдань відповідно до прогресу користувача. Historia також пропонує інструменти для викладачів, що дозволяють інтегрувати програму в формальний освітній процес та відстежувати результати учнів.

Обмежений візуальний компонент та низький рівень імерсивності зменшує залученість користувачів, особливо молодших вікових груп. Фокус на текстових матеріалах та дебатах створює високий когнітивний бар’єр для початківців. Програма також має обмежену підтримку мультимедійного контенту для представлення історичного матеріалу. Відносно висока вартість ліцензій та закрита екосистема обмежують доступність програми для широкого використання [10].

Mission US, розроблена Corporation for Public Broadcasting та WNET, спеціалізується на інтерактивних місіях, що дозволяють учням «прожити» провідні моменти американської історії через приймання рішень від імені історичних персонажів (див. рисунок 1.4).

Mission US розроблена з використанням технологій HTML5, JavaScript та адаптована для використання в освітньому середовищі з обмеженими технічними ресурсами [11]. Програма використовує наративно-орієнтований підхід з розгалуженими сюжетними лініями, що залежать від рішень користувача. Система відстеження прогресу інтегрована з освітніми стандартами та навчальними цілями. Mission US також включає комплексні методичні матеріали для викладачів, що дозволяють ефективно інтегрувати програму в навчальний процес.

Рисунок 1.4 – Приклад роботи Mission US

Mission US пропонує емоційне занурення в історичний контекст через наративний підхід та персоналізацію досвіду. Програма фокусується на розвитку емпатії та розуміння мотивацій історичних дійових осіб через приймання рішень від їхнього імені. Розгалужені сюжетні лінії демонструють вплив індивідуальних рішень на історичні процеси. Mission US включає детальні допоміжні матеріали, первинні джерела та методичні рекомендації для викладачів.

Mission US обмежена вузьким фокусом на американській історії, що зменшує її глобальну освітню цінність. Обмежена кількість доступних місій (6 на даний момент) охоплює лише окремі історичні періоди та теми. Програма має

досить простий візуальний стиль та обмежені можливості для інтерактивної взаємодії з історичним середовищем. Спрощений ігровий процес може знижувати залученість більш досвідчених користувачів [12].

Результати порівняльного аналізу функціональних можливостей розглянутих систем наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	Civilization VI	Assassin's Creed Discovery Tour	Historia: Game of Claims	Mission US
1	2	3	4	5
Технологічний стек	Власний рушій	AnvilNext 2.0	HTML5, JavaScript	HTML5, JavaScript
Історична достовірність	Середня	Висока	Висока	Висока
Візуальна якість	Висока	Дуже висока	Низька	Середня
Імерсивність	Висока	Висока	Низька	Середня
Адаптивність контенту	Низька	Відсутня	Висока	Середня
Системні вимоги	Високі	Дуже високі	Низькі	Низькі
Крос-платформність	Windows, macOS	Windows, PlayStation, Xbox	Web	Web
Можливість модифікації	Висока	Відсутня	Відсутня	Відсутня
Розвиток критичного мислення	Середній	Низький	Високий	Середній

Продовження таблиці 1.1

1	2	3	4	5
Інструменти для викладачів	Відсутні	Базові	Розширені	Розширені
Аналітика навчальних результатів	Відсутня	Базова	Розширена	Розширена
Доступність (ціна)	Низька	Середня	Низька	Висока
Альтернативні історичні сценарії	Обмежені	Відсутні	Обмежені	Обмежені

Проведений порівняльний аналіз існуючих навчальних програмних забезпечень для вивчення історії виявив значні можливості для інновацій та вдосконалень у цій галузі. Кожна з розглянутих платформ має свої сильні сторони та обмеження, але жодна не пропонує оптимального поєднання всіх критично важливих аспектів для ефективного історичного навчання через інтерактивні сценарії та симуляції [13].

Розробка власної платформи з використанням сучасних технологій Phaser.js та TypeScript представляє обґрунтовану стратегію для створення конкурентоспроможного рішення, здатного заповнити виявлені прогалини на ринку. Технологічний стек Phaser.js/TypeScript забезпечить необхідну крос-платформність та швидкість розробки, дозволяючи оперативно адаптуватися до змін освітніх потреб та впроваджувати інноваційні методики навчання [14].

Головними конкурентними перевагами власної платформи можуть стати: баланс між історичною достовірністю та ігровою залученістю, адаптивність контенту до різних рівнів знань користувачів, розвинені механізми альтернативних історичних сценаріїв для демонстрації причинно-наслідкових зв’язків та інтеграція з формальними освітніми процесами. Особливо перспективним напрямом є фокус на розвиток критичного мислення та історичної емпатії через інтерактивні симуляції соціальних, політичних та економічних систем різних історичних періодів [15].

Водночас, розробка власної платформи пов’язана з суттєвими технологічними викликами та педагогічними ризиками, які потребують ретельного планування та поетапного впровадження. Однак, при правильному виконанні, розробка власного навчального програмного забезпечення ігрового типу для занурення в історичний контекст з використанням Phaser.js та TypeScript може створити життєздатну альтернативу існуючим рішенням та забезпечити унікальну освітню цінність для учнів та викладачів.

1.3 Аналіз методів розв’язання задачі

Розробка навчального програмного забезпечення для занурення користувача в історичний контекст вимагає комплексного аналізу існуючих методів та підходів з урахуванням педагогічних цілей, технічних можливостей обраних технологій та психологічних аспектів сприйняття історичного матеріалу. Основним аспектом при проектуванні такої системи є вибір оптимальної архітектури та підходів до створення інтерактивних сценаріїв і симуляцій, які забезпечать достовірний історичний контекст при збереженні навчальної цінності.

В контексті розробки історичної навчальної гри з використанням JavaScript/TypeScript та фреймворку Phaser.js можна розглянути декілька методологічних підходів. Одним із найефективніших є сценарно-орієнтована архітектура, яка передбачає поділ гри на окремі сценарії з чітко визначеними навчальними цілями та історичними аспектами. Phaser.js, як потужний фреймворк для розробки HTML5-ігор, надає для цього всі необхідні інструменти через свою систему Scene, що дозволяє організувати структуру гри у вигляді окремих, логічно завершених сцен з власним життєвим циклом [16].

Альтернативним підходом є використання компонентно-сутнісної архітектури ECS – Entity Component System) яка дозволяє моделювати складні історичні симуляції через розподіл ігрових об’єктів на сутності зі специфічними компонентами та системами для обробки цих компонентів. Даний підхід особливо ефективний для створення історичних симуляцій, де потрібно

відтворити складні взаємодії між багатьма історичними акторами, економічними системами чи військовими конфліктами, зберігаючи при цьому історичну точність та навчальну цінність.

При розробці інтерактивних механік, що сприяють зануренню в історичний контекст, важливо враховувати принципи геймдизайну та педагогіки. Метод наративно-базованого навчання – Narrative-Based Learning пропонує використовувати силу розповіді для передачі історичних знань, емоційного залучення користувача та формування розуміння причинно-наслідкових зв’язків історичних подій. Phaser.js надає інструменти для реалізації цього підходу через підтримку діалогових систем, вбудованих відео та анімацій [17].

З точки зору технічної реалізації, для забезпечення плавної роботи та високої продуктивності інтерактивних симуляцій доцільно застосовувати патерн «об’єктний пул» або Object Pooling. Цей підхід особливо важливий для історичних сценаріїв з великою кількістю рухомих об’єктів (наприклад, симуляції битв чи розвитку міст), оскільки дозволяє значно зменшити навантаження на збирач сміття через повторне використання об’єктів замість їх постійного створення та знищення.

Для забезпечення адаптивності навчального контенту та підтримки різних стилів навчання можливе використання підходу «динамічного налаштування складності» – Dynamic Difficulty Adjustment. Цей метод передбачає автоматичне коригування рівня складності симуляцій в залежності від успішності користувача, що особливо важливо в контексті навчальних ігор, де баланс між викликом та доступністю матеріалу є критичним.

При розробці візуальної складової історичних симуляцій важливим є вибір між реалістичним та стилізованим підходами. Реалістичний підхід може сприяти глибшому зануренню в історичний контекст через точне відтворення візуальних деталей епохи, проте вимагає значно більше ресурсів. Стилізований підхід дозволяє зосередитись на основних візуальних елементах та символах епохи, що може бути більш ефективним для передачі навчальних концепцій. В обох

випадках Phaser.js пропонує потужні інструменти для роботи з графікою, включаючи підтримку WebGL для складних візуальних ефектів [18].

Для реалізації збереження прогресу користувача та аналітики навчальної ефективності доцільно інтегрувати систему управління навчанням – LMS з ігровим додатком. Це може бути досягнуто через імплементацію стандартів як xAPI – Experience API або SCORM, що дозволить відстежувати навчальні досягнення та адаптувати контент відповідно до потреб конкретного користувача.

На основі проведеного аналізу для розробки навчального програмного забезпечення ігрового типу для занурення користувача в історичний контекст рекомендується використовувати гібридний підхід, що поєднує сценарно-орієнтовану архітектуру для організації навчального контенту та елементи компонентно-сутнісної системи для реалізації складних історичних симуляцій. Фреймворк Phaser.js забезпечить необхідні інструменти для створення інтерактивних елементів, візуалізації історичних сценаріїв та управління ігровим процесом, а використання TypeScript додасть переваги статичної типізації та покращить підтримуваність коду. Така архітектура дозволить досягти балансу між навчальною цінністю, історичною достовірністю та ігровою привабливістю додатку, забезпечуючи достовірність передання користувачу історичного контексту.

1.4 Постановка задач

Проаналізувавши існуючі методи розробки навчального програмного забезпечення ігрового типу з історичною тематикою та враховуючи можливості фреймворку Phaser.js у поєднанні з JavaScript/TypeScript, було визначено наступні задачі, які необхідно виконати в рамках курсового проєкту:

- розробити архітектуру навчальної гри з поділом на історичні сцени, що забезпечуватиме логічну послідовність представлення історичного матеріалу та плавний перехід між різними історичними епохами або подіями;

- розробити модуль історичних симуляцій, який дозволить відтворювати значущі історичні події, демонструвати причинно-наслідкові зв’язки та впливи рішень користувача на розвиток історичних сценаріїв;
- розробити модуль інтерактивних діалогів з історичними персонажами, що забезпечить занурення в контекст епохи через спілкування з провідними історичними постатями та розуміння їхньої мотивації;
- розробити модуль адаптивної складності, який аналізуватиме прогрес користувача та автоматично регулюватиме рівень складності завдань для забезпечення оптимального навчального досвіду;
- розробити модуль історичної енциклопедії, який надаватиме доступ до додаткових матеріалів, пояснень та фактів про історичні події, персонажів та артефакти, що зустрічаються у грі;
- розробити модуль оцінювання знань, який включатиме різноманітні форми перевірки засвоєння матеріалу: тести, головоломки, ситуаційні завдання, інтегровані в ігровий процес;
- розробити систему досягнень та прогресу, яка мотивуватиме користувача до глибшого вивчення історичного матеріалу через ігрові механіки винагород;
- розробити модуль візуалізації історичних локацій з використанням технологій Phaser.js, що забезпечить створення автентичного візуального середовища відповідно до історичного періоду;
- розробити модуль збереження прогресу та аналітики навчальних досягнень, який дозволить відстежувати засвоєння матеріалу та генерувати рекомендації для подальшого навчання;
- реалізувати адаптивний інтерфейс, що забезпечить зручне використання програмного забезпечення на різних пристроях з підтримкою сенсорного та клавіатурно-мишного введення;

- провести тестування розробленого програмного забезпечення на відповідність педагогічним цілям, історичну достовірність, зручність використання та технічну стабільність на різних платформах.

1.5 Висновки

Отже у результаті проведеного дослідження було встановлено, що навчальне програмне забезпечення ігрового типу для занурення користувача в історичний контекст має значний потенціал для підвищення ефективності історичної освіти через посилення мотивації, забезпечення емоційного зв’язку з матеріалом та створення умов для експериментального навчання в безпечному віртуальному середовищі.

Аналіз існуючих платформ та фреймворків показав, що використання Phaser.js у поєднанні з TypeScript надає потужний інструментарій для розробки інтерактивних історичних симуляцій та сценаріїв.

Водночас, розробка власної платформи пов’язана з суттєвими технологічними викликами та педагогічними ризиками, які потребують ретельного планування та поетапного впровадження.

2 АНАЛІЗ СТРУКТУРИ АЛГОРИТМІВ ТА ФОРМУВАННЯ ПРОБЛЕМАТИКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та дослідження проблематики

Аналіз вхідних даних для навчального програмного забезпечення ігрового типу для занурення користувача в історичний контекст потребує визначення основних інформаційних потоків, які формуються в процесі взаємодії з інтерактивними сценаріями та історичними симуляціями. Дані, що оброблятимуться системою, характеризуються різнорідністю, специфічною структурою та вимогами до історичної достовірності, що зумовлює необхідність комплексного підходу до їх організації та зберігання.

Для технічної реалізації системи обрано JavaScript/TypeScript у поєднанні з фреймворком Phaser.js для створення інтерактивного ігрового середовища, що забезпечить кросбраузерну сумісність та підтримку як мобільних, так і десктопних пристроїв. Архітектурна організація системи передбачає багаторівневу структуру з чітким розмежуванням відповідальності між компонентами та модульною організацією кодової бази. Такий підхід надасть гнучкості процесу розробки через можливість паралельної роботи над різними модулями та спрощуватиме подальшу підтримку системи через розділення і групування функціональності в окремих частинах системи [19].

Процес розробки навчального програмного забезпечення для занурення в історичний контекст передбачає створення п’яти основних функціональних модулів клієнтської архітектури: модуль історичного контенту, система управління персонажами, модуль інтерактивних сценаріїв, система прогресу та досягнень, модуль візуалізації історичних артефактів. Кожен із цих модулів оперує специфічними наборами даних, що потребують відповідних підходів до організації та відображення.

Модуль історичного контенту надаватиме функціональність для роботи з різнотипною інформацією, що охоплює історичні дані (епоха, географічне розташування, історичні події, персоналії), медіаматеріали (зображення, аудіо

наративи, текстові описи) та метадані для структурування контенту. Основним призначенням цього модуля є ефективне відображення та організація історичних відомостей, які зберігатимуться частково локально та частково завантажуватимуться за допомогою REST API. Для оптимізації роботи з історичними даними доцільно використовувати систему кешування через IndexedDB, що дозволить зменшити мережеві запити та забезпечити можливість офлайн-доступу до основних навчальних матеріалів [20].

Модуль управління персонажами опрацьовуватиме дані, пов’язані з віртуальними історичними особами та аватаром користувача. Ці дані включатимуть атрибути персонажів (вік, стать, соціальний статус, професія відповідно до епохи), їх візуальне представлення (спрайти, анімації, елементи одягу) та параметри взаємодії з навколишнім середовищем. Для користувацького персонажа важливо зберігати дані про прогрес, здобуті знання та вміння, а також діалогові переваги. Для ефективної роботи з такими даними оптимальним вибором є використання локального сховища для збереження базових налаштувань та параметрів персонажа, а також Redux для управління станом ігрової сесії в реальному часі.

Модуль інтерактивних сценаріїв відповідатиме за обробку даних, пов’язаних із сюжетними лініями, діалогами та варіантами вибору, які впливають на розвиток історичних подій у межах симуляції. Призначення цієї частини системи полягає в оперуванні структурами сценаріїв, що містять розгалужені діалоги, умови переходів між сценами, тригери подій та наслідки дій користувача. Для ефективної організації таких даних доцільно використовувати JSON-структури з чітко визначеною схемою, що дозволить легко створювати та модифікувати сценарії без зміни програмного коду. Система використовуватиме власну реалізацію кінцевого автомата для керування станами сценаріїв та обробки подій, що виникають внаслідок дій користувача.

Система прогресу та досягнень оперуватиме даними про навчальні досягнення користувача, засвоєні історичні факти, виконані завдання та здобуті навички. Цей модуль відповідатиме за відстеження прогресу, надання

зворотного зв’язку та адаптацію складності контенту відповідно до рівня знань користувача. Дані про прогрес включатимуть кількісні показники (бали, рівні, відсоток виконання) та якісні метрики (розуміння історичних зв’язків, засвоєння фактів, здатність застосовувати знання). Для зберігання таких даних доцільно використовувати комбінацію локального сховища для поточного стану та серверного API для синхронізації прогресу між пристроями та забезпечення можливості порівняння результатів з іншими користувачами.

Модуль візуалізації історичних артефактів забезпечуватиме роботу з тривимірними моделями, інтерактивними картами та реконструкціями історичних об’єктів. Цей модуль працюватиме з даними про геометрію об’єктів, текстури, матеріали та параметри освітлення, а також з метаданими, що описують історичний контекст кожного артефакту. Для ефективної роботи з тривимірними даними Phaser.js буде використовувати WebGL та оптимізовані формати зберігання моделей, такі як glTF, що забезпечать баланс між якістю візуалізації та продуктивністю на різних пристроях.

У якості засобів для комунікації з серверною частиною доцільно використати RESTful API для отримання історичних даних, синхронізації прогресу та завантаження медіаконтенту, а також WebSocket для багатокористувацьких режимів та спільного дослідження історичних періодів. Для локального зберігання даних підійде IndexedDB, яка дозволить ефективно кешувати об’ємну інформацію про історичні події, персонажів та артефакти, забезпечуючи швидкий доступ до неї навіть без підключення до мережі [21].

Отже, аналіз вхідних даних системи навчального програмного забезпечення ігрового типу для занурення в історичний контекст визначає необхідність розробки п’яти функціональних модулів, кожен з яких працює зі специфічними наборами даних та вимагає відповідних підходів до їх відображення та обробки. Реалізація системи на основі JavaScript/TypeScript та фреймворку Phaser.js забезпечить надійну основу для розробки інтерактивного ігрового середовища з історичною достовірністю. Комбінований підхід до організації даних з використанням серверного API для отримання основної

інформації та локальних сховищ для кешування і управління станом дозволить досягти оптимального балансу між швидкодією роботи та глибиною занурення в історичний контекст. Використовуючи такий набір технологій при ретельному плануванні та проєктуванні можна створити програмний продукт, який ефективно поєднуватиме навчальну цінність з ігровим досвідом, сприяючи засвоєнню історичних знань через інтерактивну взаємодію та симуляції.

2.2 Нова структура інтеграції програмних модулів

Аналіз функціональних можливостей навчального програмного забезпечення для занурення користувача в історичний контекст базується на детальному дослідженні архітектури системи та алгоритмів її функціонування. Основу функціональності застосунку становлять інтерактивні механізми, що забезпечують глибоке занурення користувача в автентичне історичне середовище через симуляцію реальних процесів управління та прийняття рішень.

Центральною функціональною можливістю системи є механізм управління історичними локаціями, який дозволяє користувачеві взаємодіяти з віртуальним простором, що відтворює автентичні умови конкретного історичного періоду. Система забезпечує візуалізацію локації з усіма наявними ресурсами та персонажами, створюючи цілісне уявлення про історичний контекст. Функціональність управління ресурсами дозволяє користувачеві формувати розуміння про економічні та соціальні аспекти досліджуваного періоду через безпосередню взаємодію з історичними реаліями.

Особливого значення набуває функціональність управління персонажами, що реалізується через систему створення та розпуску груп. Ця можливість дозволяє користувачеві симулювати процеси формування історичних спільнот, військових підрозділів або робочих колективів, характерних для конкретної епохи. Система забезпечує перевірку відповідності призначених ролей історичному контексту, що гарантує освітню цінність та достовірність симуляції.

Функціональність управління командами становить окремий рівень складності системи, що дозволяє користувачеві досліджувати аспекти лідерства

та стратегічного планування в історичному контексті. Система відтворює основні: характеристики історичних підрозділів через параметри моралі, ресурсного забезпечення та організаційної структури. Користувач отримує можливість аналізувати вплив різних факторів на ефективність команди, що сприяє формуванню розуміння про складність історичних процесів управління.

Критично важливою функціональною можливістю є система динамічної адаптації до дій користувача, що забезпечує персоналізований навчальний досвід. Алгоритми системи постійно аналізують рішення користувача, оцінюючи їх історичну достовірність та освітню цінність. На основі цього аналізу система формує адаптивний зворотний зв’язок, що включає історичні довідки, пояснення наслідків та рекомендації для подальшого навчання.

Функціональність безпеки локацій додає елемент стратегічного планування до навчального процесу, відтворюючи реальні умови, в яких історичні особи приймали важливі рішення. Система обмежує можливості редагування складу команд у небезпечних умовах, що змушує користувача враховувати фактори ризику та безпеки при плануванні дій, характерні для досліджуваного історичного періоду.

Особливе місце в функціональній архітектурі системи займає механізм циклічності навчального процесу, що дозволяє користувачеві багаторазово взаємодіяти з різними історичними сценаріями. Ця функціональність забезпечує можливість поглибленого вивчення історичних процесів через різні перспективи та сценарії розвитку подій. Система підтримує плавні переходи між різними історичними періодами та контекстами, зберігаючи цілісність навчального досвіду.

Функціональність автентифікації та збереження прогресу забезпечує персоналізацію навчального процесу та можливість поступового розвитку історичних знань користувача. Система відстежує індивідуальний прогрес кожного користувача, адаптуючи складність та тематику сценаріїв відповідно до досягнутого рівня компетентності.

Інтеграційні функціональні можливості системи забезпечують узгоджену роботу всіх компонентів застосунку, створюючи цілісне навчальне середовище. Алгоритми синхронізації даних забезпечують коректне відображення наслідків дій користувача в усіх пов’язаних елементах системи, підтримуючи логічну послідовність та історичну достовірність симуляції.

Досліджені функціональні можливості застосунку демонструють комплексний підхід до створення навчального програмного забезпечення, що ефективно поєднує ігрові механіки з освітніми цілями. Система забезпечує багаторівневу взаємодію користувача з історичним контентом, від базового знайомства з історичними реаліями до складного стратегічного планування та аналізу наслідків прийнятих рішень. Функціональна архітектура застосунку створює основу для ефективного засвоєння історичних знань через особистий досвід участі у симуляції подій минулого.

2.3 Аналіз та розробка інтерактивних можливостей програмного застосунку

Розробка інтерактивних можливостей навчального програмного забезпечення ігрового типу для занурення в історичний контекст потребує комплексного підходу до проектування механізмів взаємодії користувача з віртуальним історичним середовищем. Ефективність навчального процесу значною мірою залежить від рівня інтерактивності системи та її здатності забезпечити активну участь користувача в симуляції історичних подій.

Основою інтерактивної функціональності застосунку є система прийняття рішень, що дозволяє користувачеві впливати на розвиток історичних сценаріїв через вибір різних варіантів дій. Ця система побудована на принципі альтернативних сюжетних ліній, де кожне рішення користувача призводить до специфічних наслідків, що відображають реальну складність історичних процесів. Інтерактивність досягається через створення розгалужених сценаріїв, які адаптуються до дій користувача та забезпечують унікальний досвід кожного проходження.

Головним елементом інтерактивних можливостей є система діалогової взаємодії з історичними персонажами, що реалізована через спеціальні інтерфейсні елементи та алгоритми обробки користувацьких команд. Користувач може вести переговори з історичними постатями, отримувати інформацію про період, що вивчається, та впливати на ставлення персонажів через свої дії та рішення. Ця функціональність забезпечує глибоке занурення в історичний контекст через безпосередню комунікацію з представниками досліджуваної епохи.

Інтерактивна карта локацій становить важливий компонент системи, що дозволяє користувачеві вільно переміщуватися між різними історичними місцями та досліджувати їх особливості. Кожна локація містить унікальні інтерактивні елементи, такі як історичні артефакти, будівлі, природні об’єкти, з якими користувач може взаємодіяти для отримання додаткової інформації або виконання специфічних завдань. Система відстеження переміщень фіксує маршрути користувача та адаптує подальші сценарії відповідно до досліджених локацій.

Розроблена система управління ресурсами забезпечує інтерактивну взаємодію з економічними аспектами історичного періоду. Користувач може розподіляти матеріальні ресурси, людські сили та час між різними завданнями, що відтворює реальні виклики історичного управління. Інтерактивні елементи включають перетягування ресурсів між різними проектами, налаштування пріоритетів розподілу та моніторинг ефективності використання наявних засобів.

Система формування та управління групами персонажів реалізує складні інтерактивні механізми, що дозволяють користувачеві створювати різноманітні історичні спільноти. Інтерактивність досягається через можливість індивідуального налаштування характеристик кожного персонажа, призначення специфічних ролей та завдань, а також відстеження динаміки взаємовідносин всередині групи. Користувач може впливати на мораль групи через прийняті рішення та стиль управління.

Інтерактивні елементи навчального оцінювання інтегровані безпосередньо в ігровий процес, забезпечуючи неперервну оцінку знань користувача без порушення імерсивності досвіду. Система аналізує рішення користувача в контексті історичної достовірності та надає миттєвий зворотний зв'язок через інтерактивні підказки, історичні довідки та візуальні індикатори правильності дій. Інтерактивні квести та завдання адаптуються до рівня знань користувача, забезпечуючи оптимальний рівень складності.

Система візуалізації наслідків рішень становить важливий аспект інтерактивності, що дозволяє користувачеві спостерігати за довгостроковими ефектами своїх дій через динамічні графіки, діаграми та візуальні представлення змін у віртуальному історичному світі. Інтерактивна хронологія подій фіксує основні моменти проходження сценарію та дозволяє користувачеві повертатися до попередніх рішень для аналізу альтернативних варіантів розвитку.

Розроблені інтерактивні можливості включають систему персоналізації інтерфейсу, що адаптується до переваг користувача та специфіки досліджуваного історичного періоду. Користувач може налаштовувати візуальне оформлення відповідно до історичної епохи, вибирати рівень деталізації інформації та встановлювати індивідуальні параметри складності симуляції.

Мультимедійна інтерактивність реалізована через інтеграцію аудіовізуальних елементів, що реагують на дії користувача та створюють відповідну атмосферу історичного періоду. Інтерактивні аудіогіди забезпечують контекстуальні пояснення історичних подій, а візуальні ефекти підсилюють емоційне сприйняття симуляції.

Система співпраці та змагання між користувачами розширює інтерактивні можливості через створення спільних історичних проектів, де декілька учасників можуть одночасно взаємодіяти з одним історичним сценарієм, приймаючи рішення від імені різних історичних сторін або персонажів. Інтерактивні елементи включають можливість обміну ресурсами, ведення переговорів та координації спільних дій.

Розроблені інтерактивні можливості програмного застосунку створюють багаторівневу систему взаємодії користувача з навчальним контентом, що забезпечує активне залучення до процесу вивчення історії через безпосередню участь у симуляції історичних подій. Комплексна реалізація інтерактивних механізмів дозволяє досягти високого рівня занурення в історичний контекст при збереженні освітньої цінності та педагогічної ефективності навчального процесу.

2.4 Розробка загальної моделі застосунку

Діаграма відображає послідовність взаємодії користувача з навчальним програмним забезпеченням для занурення в історичний контекст. Розроблена модель системи представляє собою чітко структурований процес, що починається з автентифікації користувача та завершується виходом із симуляції після проходження навчального сценарію.

На початку взаємодії користувач проходить процес автентифікації, що дозволяє системі ідентифікувати конкретного учня та зберігати його прогрес. Після успішної автентифікації користувач отримує доступ до вибору історичного сценарію, який становить інтерес для навчання. Цей етап є критично важливим, оскільки саме тут визначається тематичне спрямування подальшої симуляції та інформаційне наповнення навчального процесу.

Після вибору конкретного історичного сценарію відбувається процес завантаження симуляції, під час якого система підготовлює необхідні ресурси, візуальні матеріали, аудіосупровід та інтерактивні елементи, що відповідають обраному періоду історії. Цей етап може супроводжуватися відображенням інформаційних довідок або короткого історичного контексту для занурення користувача у відповідну епоху (див. рисунок 2.1).

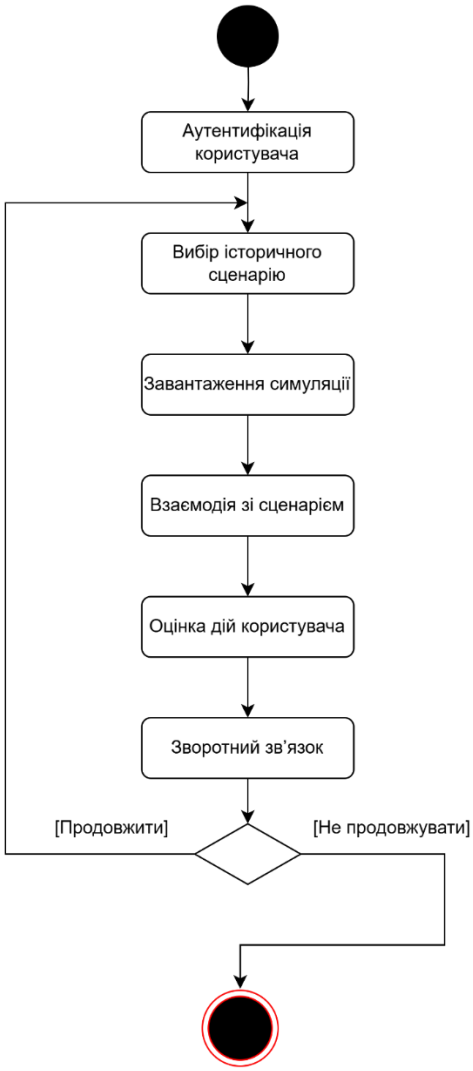

Рисунок 2.1 – Діаграма діяльності програмних засобів

Наступним кроком розгортається безпосередня взаємодія користувача зі сценарієм, що є центральним елементом всієї моделі. На цьому етапі відбувається активне занурення в історичний контекст через інтерактивні елементи, прийняття рішень, керування історичними підрозділами, взаємодію з історичними особами чи артефактами. Саме тут реалізується основна освітня цінність програмного забезпечення через залучення користувача до активного пізнання історичних процесів.

Під час взаємодії зі сценарієм система здійснює постійну оцінку дій користувача, відстежуючи правильність прийнятих рішень, їх історичну достовірність, ефективність та відповідність навчальним цілям. Ця оцінка відбувається в режимі реального часу та формує основу для наступного етапу.

На основі зібраної інформації про дії користувача система генерує зворотний зв’язок, який може містити історичні довідки, пояснення наслідків прийнятих рішень, порівняння з реальними історичними подіями та рекомендації щодо подальшого навчання. Зворотний зв’язок виконує важливу функцію закріплення знань та корекції історичних уявлень користувача.

Після отримання зворотного зв’язку користувач стикається з вибором – продовжити взаємодію з поточним сценарієм, перейти до іншого історичного періоду чи завершити роботу з системою. У випадку вибору продовження, процес повертається до етапу вибору історичного сценарію, дозволяючи користувачеві розширити свій досвід вивчення історії. У разі завершення роботи користувач виходить із системи.

Розроблена модель системи забезпечує циклічність навчального процесу, постійну взаємодію користувача з історичним контекстом та адаптивність до індивідуальних потреб учня. Завдяки такій структурі досягається плавне поєднання ігрових механік з освітніми цілями, що сприяє ефективному засвоєнню історичних знань через особистий досвід участі у симуляції подій минулого.

2.5 Дослідження алгоритмів роботи системи

Розробка навчального програмного забезпечення для занурення користувача в історичний контекст передбачає створення чітких алгоритмів, що забезпечують логіку функціонування основних компонентів системи. Представлені діаграми відображають два основні алгоритми: управління локацією та управління командою, які лежать в основі симуляції історичних сценаріїв.

Перший алгоритм демонструє процес управління локацією, починаючи з ініціалізації та завершуючи оновленням стану локації. Після старту система відображає екран локації з усіма наявними ресурсами та персонажами, що забезпечує користувачеві розуміння поточного історичного контексту. Наступним кроком алгоритм передбачає відображення доступних ресурсів,

зокрема людських сил, що є критичним елементом для симуляції історичних процесів. Система формує список вільних персонажів, яких можна залучити до виконання певних історичних ролей та завдань (див. рисунок 2.2).

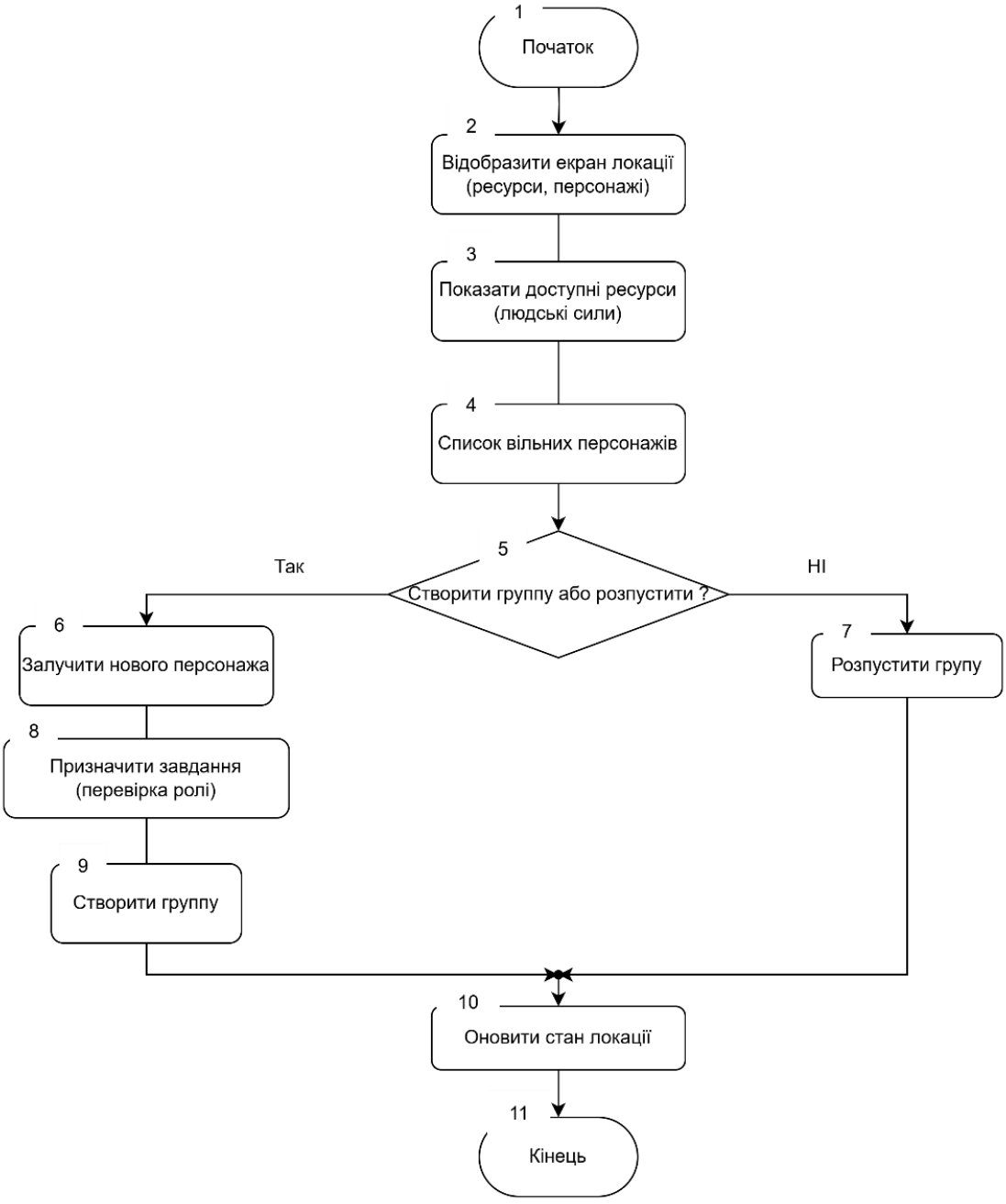

Рисунок 2.2 – Блок-схема алгоритму управління локацією

На основі представленої інформації користувач приймає рішення щодо створення нової групи або розпуску існуючої. У випадку створення групи алгоритм переходить до послідовності дій, що включає залучення нового

персонажа, призначення йому завдання з перевіркою відповідності історичній ролі та безпосереднє формування групи. Якщо користувач обирає розпуск групи, система виконує відповідні дії для звільнення задіяних ресурсів. Заключними етапами алгоритму є оновлення стану локації, що відображає наслідки прийнятих рішень у контексті історичної симуляції, та завершення поточного циклу взаємодії.

Другий алгоритм фокусується на управлінні командою у симуляції історичних сценаріїв. Процес також починається з ініціалізації та переходить до відображення екрану команди з інформацією про мораль, наявні ресурси та лідера, що відтворює структуру історичного підрозділу. Система демонструє основні характеристики команди – вантаж, місткість та швидкість, що впливають на ефективність дій у контексті симуляції (див. рисунок 2.3).

Алгоритм перевіряє, чи знаходиться команда в безпечній локації для можливості редагування її складу та параметрів. За позитивного результату перевірки система показує інформацію про лідера команди, якщо такий наявний, та дозволяє користувачеві вибрати одну з доступних дій. Залежно від обраної дії алгоритм розгалужується: користувач може здійснити обмін ресурсами з сусідньою командою, перейти до управління наявними ресурсами або звільнити лідера команди.

Кожна з цих дій відображає різні аспекти історичного управління підрозділами та ресурсами, що забезпечує глибше занурення користувача в історичний контекст. Після виконання обраної дії алгоритм оновлює стан команди, відображаючи наслідки прийнятих рішень, та завершує цикл взаємодії. У випадку, якщо команда знаходиться в небезпечній локації, система повертається до початку алгоритму, обмежуючи можливості редагування до забезпечення безпечних умов.

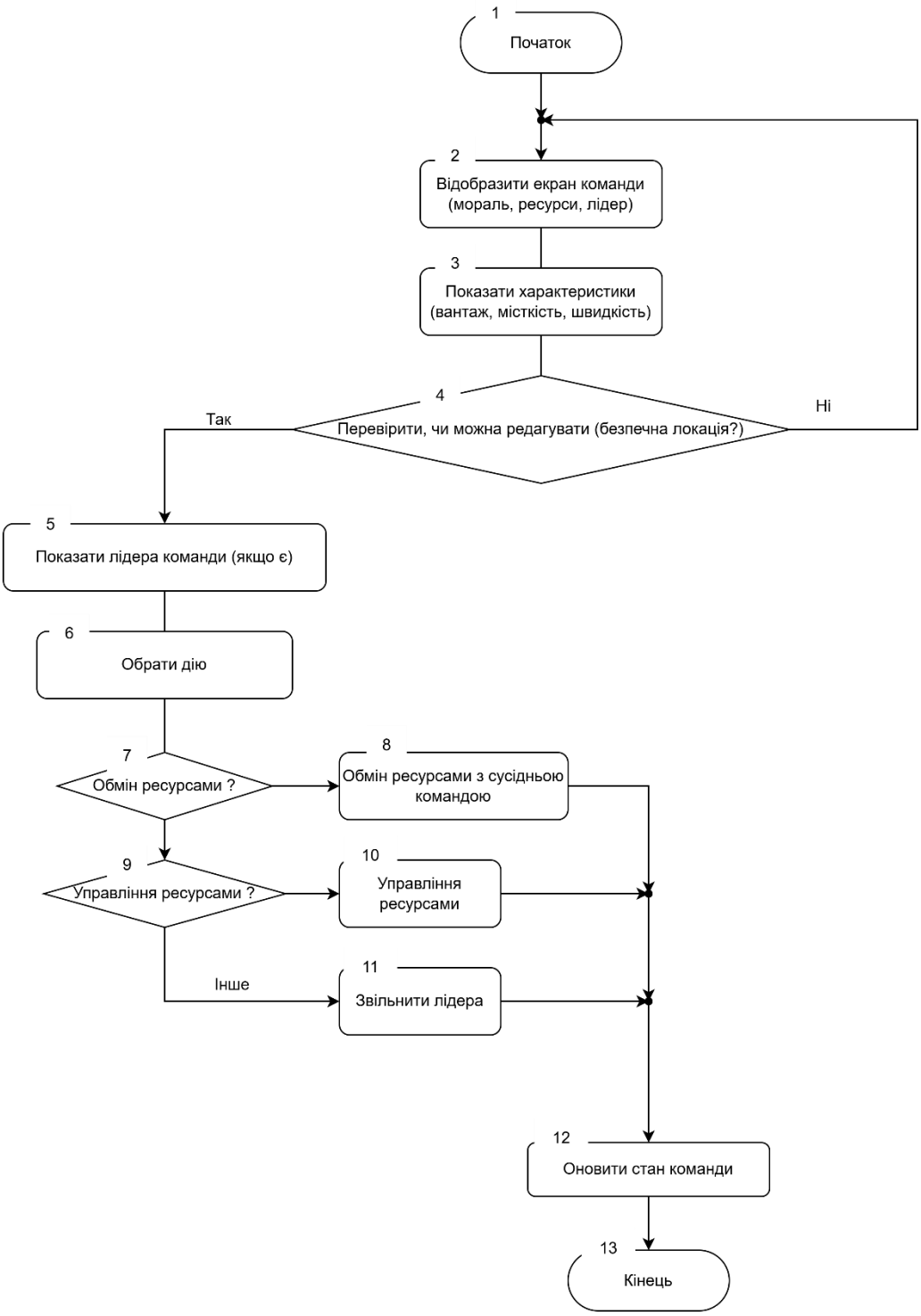

Рисунок 2.3 – Блок-схема алгоритму управління командою

Розроблені алгоритми функціонування системи забезпечують логічну послідовність дій як з боку програмного забезпечення, так і з боку користувача, створюючи інтерактивне середовище для симуляції історичних сценаріїв. Завдяки детальній структурі та чіткій послідовності кроків, алгоритми

дозволяють реалізувати навчальне програмне забезпечення ігрового типу, яке ефективно поєднує ігрові механіки з освітніми цілями, надаючи користувачеві досвід прийняття рішень у контексті історичних подій та процесів. Комплексна взаємодія цих алгоритмів створює цілісну систему, що адаптується до дій користувача та забезпечує занурення в історичний контекст через безпосередню взаємодію з симульованими сценаріями.

2.6 Висновки

Отже проведений аналіз та проектування навчального програмного забезпечення для занурення в історичний контекст засвідчує необхідність розробки п’яти функціональних модулів, кожен з яких працює з унікальними наборами даних. Технологічна основа на JavaScript/TypeScript та Phaser.js створює надійний фундамент для побудови інтерактивного історичного середовища.

Архітектура системи з комбінованим підходом до організації даних – використання серверного API з локальними сховищами – забезпечує оптимальний баланс між швидкодією та глибиною занурення користувача в історичний контекст.

Запропонована модель системи створює циклічність навчального процесу та постійну взаємодію з історичним контекстом, що сприяє засвоєнню знань через особистий досвід участі в симуляції подій минулого.

Розроблені алгоритми встановлюють логічну послідовність дій, формуючи інтерактивне середовище для історичних сценаріїв. Їхня взаємодія створює цілісну систему, що адаптується до дій користувача та забезпечує глибоке занурення в історичний контекст.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІГРОВОГО ТИПУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробка навчального програмного забезпечення ігрового типу для занурення користувача в історичний контекст потребує ретельного вибору технологічних засобів, які забезпечать оптимальну продуктивність, інтерактивність та освітню цінність системи. Ефективний вибір інструментів розробки має враховувати як функціональні вимоги до платформи, так і особливості цільової аудиторії та специфіку освітньо-ігрового контенту [21].

При проєктуванні навчального програмного забезпечення було розглянуто декілька варіантів реалізації, кожен з яких має свої переваги та обмеження. Основними критеріями оцінки стали: продуктивність, крос-платформність, підтримка інтерактивних ігрових механік, можливості для відтворення історичного контексту, зручність розробки освітніх сценаріїв, підтримка спільноти та перспективи розвитку технології.

У контексті вибору базової технології для побудови навчального ігрового середовища було проаналізовано три основні варіанти: створення додатку на базі універсальних ігрових двигунів: Unity, Unreal Engine, розробка з використанням спеціалізованих HTML5 ігрових фреймворків, та реалізація на базі нативних технологій для мобільних пристроїв.

Перший варіант передбачав використання потужних ігрових двигунів, таких як Unity або Unreal Engine. Цей підхід забезпечує високу якість графіки, потужні інструменти для створення тривимірних середовищ та широкі можливості для реалізації ігрових механік. Однак, ці двигуни вимагають значних ресурсів для розробки, мають високу криву входження та надлишковий функціонал для освітнього програмного забезпечення. Також публікація такого додатку для вебплатформ може бути проблематичною через значний розмір

кінцевого продукту, що негативно впливає на доступність для освітніх установ з обмеженим інтернет-з’єднанням.

Другий варіант включав використання HTML5 ігрових фреймворків, таких як Phaser.js, PixiJS або BabylonJS. Ці інструменти дозволяють створювати інтерактивні ігрові середовища, які працюють безпосередньо у браузері без необхідності встановлення додаткового програмного забезпечення. Такий підхід забезпечує широку доступність для користувачів різних пристроїв та операційних систем, швидкий запуск та інтеграцію з вебсервісами. Водночас, HTML5 фреймворки мають певні обмеження щодо продуктивності при відображенні складної графіки та обробці великої кількості об’єктів, що може бути критичним для деталізованих історичних реконструкцій [22].

Третій варіант базувався на розробці нативних додатків для iOS та Android з використанням відповідних SDK та ігрових бібліотек. Цей підхід забезпечує максимальну продуктивність на мобільних пристроях та повний доступ до апаратних можливостей. Однак, необхідність підтримки двох окремих кодових баз значно збільшує витрати на розробку, а обмеженість доступу лише до мобільних платформ знижує потенційну аудиторію освітнього програмного забезпечення, особливо в контексті використання в навчальних закладах, де часто доступні лише десктопні комп’ютери.

Після аналізу варіантів та оцінки їх відповідності вимогам проєкту, було обрано другий варіант з використанням Phaser.js як основної технології для реалізації навчального ігрового середовища, доповненої TypeScript для забезпечення типобезпеки та підвищення якості коду. Обґрунтуванням цього вибору стали такі фактори:

Phaser.js є спеціалізованим фреймворком для створення 2D ігор на JavaScript/TypeScript, який надає готові компоненти для роботи з графікою, анімаціями, фізикою, звуком та користувацьким вводом [23]. Фреймворк має відкритий код, активну спільноту розробників та багату документацію, що спрощує процес розробки та вирішення потенційних проблем. Phaser.js

підтримує як WebGL, так і Canvas рендеринг, що забезпечує адаптивність до можливостей пристрою користувача та широку сумісність з різними браузерами.

TypeScript було обрано як надбудову над JavaScript для забезпечення статичної типізації, що критично важливо для складних проєктів з багатьма компонентами та даними. Використання TypeScript дозволяє виявляти помилки на етапі компіляції, покращує документацію коду через визначення інтерфейсів та типів, а також підвищує продуктивність розробки завдяки кращій підтримці IDE та автодоповненню. Інтеграція TypeScript з Phaser.js добре документована та має готові типи для всіх компонентів фреймворку.

Для організації архітектури проєкту було обрано модульний підхід з використанням патернів об’єктно-орієнтованого програмування та архітектурного шаблону Model-View-Controller або MVC, що дозволяє чітко відокремити дані (історичні відомості, сценарії, стан гри) від їх візуального представлення та логіки управління. Така структура спрощує розширення функціоналу, тестування окремих компонентів та повторне використання коду.

Для управління станом додатку було розглянуто варіанти використання вбудованого механізму Phaser.js, Redux та простого патерну спостерігача. В результаті було обрано гібридний підхід: для ігрового стану використовується механізм сцен та об’єктів Phaser.js, а для навчального контенту та прогресу користувача реалізовано патерн спостерігача. Такий підхід забезпечує адаптивність в управлінні різними аспектами додатку без надмірного ускладнення архітектури.

У контексті візуального представлення історичного контенту було проаналізовано підходи з використанням растрової графіки, векторної графіки та програмно генерованих ассетів. Для проєкту було обрано комбінований підхід: векторна графіка для інтерфейсу та ігрових персонажів, що забезпечує масштабованість на різних пристроях, та растрова графіка для історичних артефактів і деталізованих фонів, що забезпечує високу достовірність представлення. Додатково впроваджено систему процедурної генерації простих елементів оточення для зменшення загального розміру додатку [24].

Для створення анімацій розглядалися підходи з використанням спрайтових анімацій, скелетних анімацій та CSS/SVG анімацій. Враховуючи специфіку історичного контенту та потребу в детальних рухах персонажів, було обрано скелетні анімації з використанням бібліотеки Spine, яка інтегрується з Phaser.js. Цей підхід дозволяє створювати реалістичні рухи персонажів з історичною точністю при збереженні невеликого розміру ассетів.

Для реалізації інтерактивних історичних сценаріїв було розроблено власну систему керування діалогами та подіями, базовану на форматі JSON з визначеною схемою. Така система дозволяє історикам та педагогам створювати освітні сценарії без необхідності писати код, використовуючи спеціалізований редактор або текстовий редактор. Для логіки прийняття рішень та обчислення наслідків дій користувача впроваджено кінцевий автомат, реалізований як окремий модуль системи [25].

Для завантаження та управління ресурсами було обрано асинхронний підхід з використанням API завантажувача Phaser.js, доповнений системою кешування через IndexedDB для зменшення повторного завантаження великих ассетів. Для оптимізації продуктивності впроваджено динамічне завантаження ресурсів залежно від поточного історичного періоду та прогресивне покращення якості зображень під час гри.

Для забезпечення перекладу та локалізації розглядалися бібліотеки i18next, Polyglot та власна реалізація. Було обрано i18next через її адаптивність, підтримку множинних форм, контекстних перекладів та можливість динамічного завантаження мовних пакетів, що особливо важливо для історичного контенту з великою кількістю специфічних термінів та понять [26].

Для збереження прогресу та налаштувань користувача було обрано LocalStorage для базових налаштувань та IndexedDB для детального прогресу та локального кешування ассетів. Для синхронізації між пристроями впроваджено опціональну авторизацію через Oauth 2.0 з використанням RESTful API для зберігання даних на сервері [27].

Таким чином, обраний технологічний стек, що базується на Phaser.js та TypeScript, забезпечує оптимальний баланс між доступністю, інтерактивністю та освітньою цінністю, що відповідає вимогам до навчального програмного забезпечення для занурення в історичний контекст. Використання вебтехнологій забезпечує широку доступність для різних навчальних закладів без необхідності встановлення спеціалізованого програмного забезпечення, а модульна архітектура дозволяє легко розширювати та адаптувати систему для різних історичних періодів та освітніх цілей. Обрана комбінація інструментів дозволяє реалізувати всі заплановані функціональні модулі системи, забезпечуючи високу якість як ігрового, так і навчального досвіду.

3.2 Метод модульної архітектури для навчального програмного забезпечення

Цей модуль створює функціональний React-компонент під назвою Equipment, який призначений для відображення одного елементу обладнання в інтерфейсі користувача, зокрема в мобільному застосунку, оскільки використовуються компоненти React Native. Компонент приймає кілька властивостей через props, серед яких: data (це об’єкт із даними про обладнання, що зберігається в змінній e), unit (ймовірно, інформація про одиницю, яка користується обладнанням), index (індекс цього елемента у списку), city (об’єкт, що описує місто, зокрема його ресурси), path (можливо, шлях у даних або маршруті), disabled (прапорець, що вказує, чи елемент відключений), equipped (прапорець, що показує, чи обладнання вже встановлене), onSelect (функція, яка спрацьовує при виборі елемента) [28].

Всередині компонента визначається функція onSelect, яка викликає функцію, передану через пропси, лише у випадку, якщо прапорець disabled дорівнює false, тобто якщо елемент не відключений. Це захищає від небажаних дій, коли елемент неактивний.

Компонент повертає JSX-структуру, яка починається з елемента Surface, що є контейнером з певними стилями: він має відступ знизу, внутрішні відступи,

ширину на 100%, мінімальну висоту та змінну прозорість залежно від того, чи відключений елемент. Всередині Surface знаходиться TouchableOpacity, що дозволяє елементу реагувати на натискання користувача, запускаючи функцію onSelect.

Далі всередині TouchableOpacity розташовується контейнер View з горизонтальним напрямком розташування дочірніх елементів. У цьому контейнері є заголовок Subheading, який відображає назву обладнання, що береться з властивості e.name. Поруч, якщо прапорець equipped дорівнює true, відображається іконка галочки, розташована абсолютно та зміщена праворуч від назви, сигналізуючи, що обладнання вже встановлене (див. рисунок 3.1).

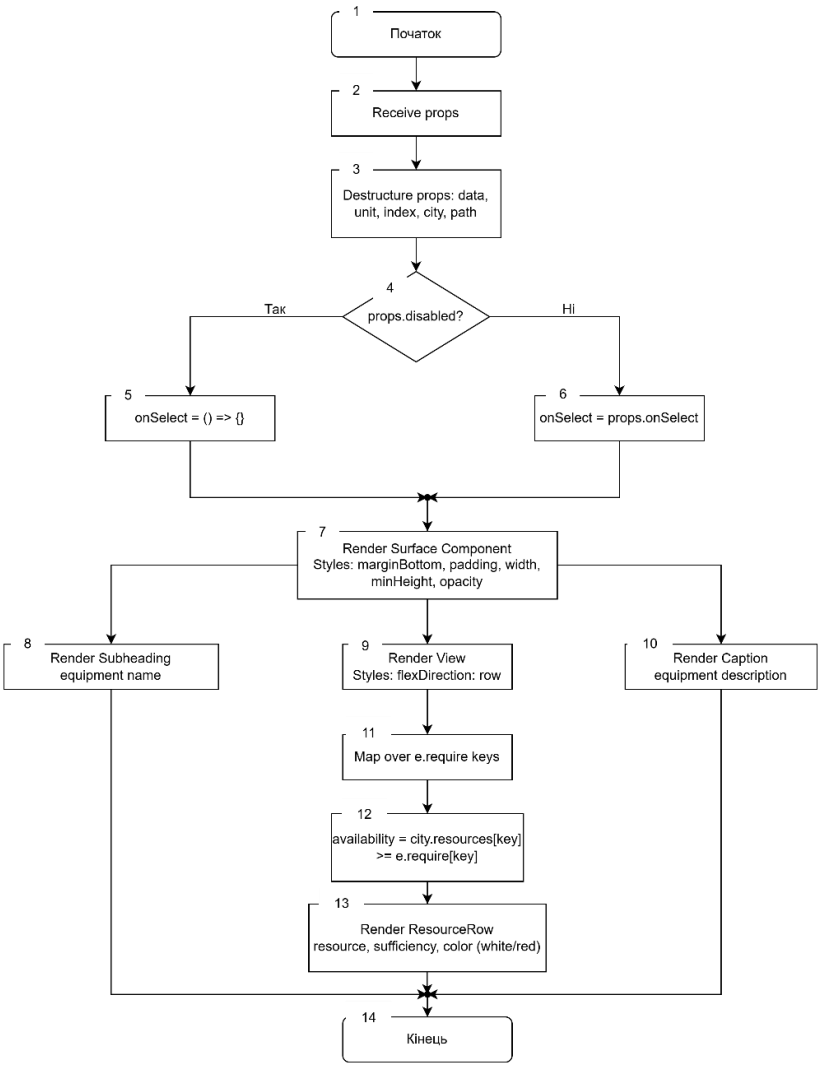

Рисунок 3.1 – Блок схема модулю відображення інформації про певне обладнання

Під назвою показується опис обладнання через компонент Caption, який бере текст з e.description.

Далі йде умовний блок: якщо в об’єкта e є властивість require, то під описом з’являється ще один контейнер View з горизонтальним розташуванням елементів. У ньому спочатку відображається підпис із локалізованим текстом, що вказує на вимоги до обладнання (текст отримується через i18n.t(«ui.equip.require»)). Потім для кожного ресурсу, що потрібен для цього обладнання, проходить цикл map, який перебирає ключі об’єкта e.require. Для кожного ресурсу перевіряється, чи є його достатньо в ресурсах міста: це визначається через порівняння значення в city.resources з потрібною кількістю в e.require. Результат цієї перевірки впливає на колір тексту: якщо ресурсу достатньо, колір буде білим, якщо ні – червоним. Для кожного ресурсу відображається компонент ResourceRow, якому передаються назва ресурсу, потрібна кількість та відповідний колір.

Таким чином, цей компонент виконує роль інтерактивної картки для одного виду обладнання: він показує основну інформацію, статус встановлення, вимоги до ресурсів, а також дозволяє вибрати обладнання шляхом натискання, якщо елемент не відключений. Усе це оформлено у компактному, візуально організованому вигляді з використанням гнучкої верстки та стилів для керування зовнішнім виглядом.

3.3 Модуль зміни стану об’єктів

Цей модуль створює React-компонент Group, обгорнутий функцією observer з бібліотеки mobx-react, що означає: він спостерігає за змінами стану в об’єктах, які використовуються всередині компонента, і автоматично оновлюється при зміні цих даних. Компонент отримує властивості через props, серед яких основним є об’єкт route.params.unit – це і є основна одиниця (ймовірно, загін або підрозділ), дані якої тут відображаються (див. рисунок 3.2).

Усередині компонента визначено функцію removeHero, яка видаляє героя з підрозділу, викликаючи метод unit.removeHero().Також є функція onExchange,

яка ініціює перехід на екран обміну, передаючи в навігацію поточний підрозділ (unit, що береться з mainStore.selectedUnit) і підрозділ, з яким планується обмін (target) [29].

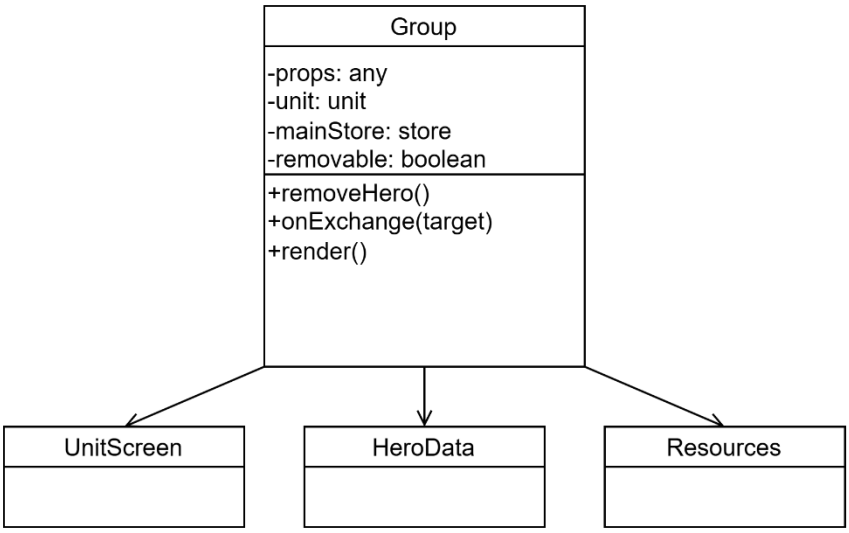

Рисунок 3.2 – UML діаграма модулю зміни стану об’єкта

Змінна removable визначає, чи можна видалити героя з підрозділу, залежно від стану підрозділу: якщо unit.state є порожнім рядком або якщо підрозділ не знаходиться в дорозі (currentRoad дорівнює undefined) і при цьому підрозділ належить тому самому угрупуванню, що й місто, тоді видалення дозволено.

Компонент повертає JSX-структуру, яка обгорнута в ScrollView, що дозволяє прокручувати контент. Усередині ScrollView перший блок View містить відступи, де спочатку відображається компонент Moral, який передає дані про мораль підрозділу. Потім йде ще один View з горизонтальним розташуванням елементів: тут показується підпис, що вказує на місткість підрозділу (значення unit.carrying поділене на unit.capacity, округлене вниз) із позначенням одиниць виміру. Поруч відображається швидкість підрозділу в кілометрах на день. Після цього ставиться горизонтальний роздільник Divider для візуального поділу контенту.

Далі йде умовний блок: якщо у підрозділу є герой (unit.hero не є undefined), то відображається підпис про лідера, а під ним – компонент Hero, який передає дані про героя. Цей компонент також отримує властивість remove, що містить функцію removeHero, та прапорець removable, що вказує, чи можна видаляти героя (див. рисунок 3.3).

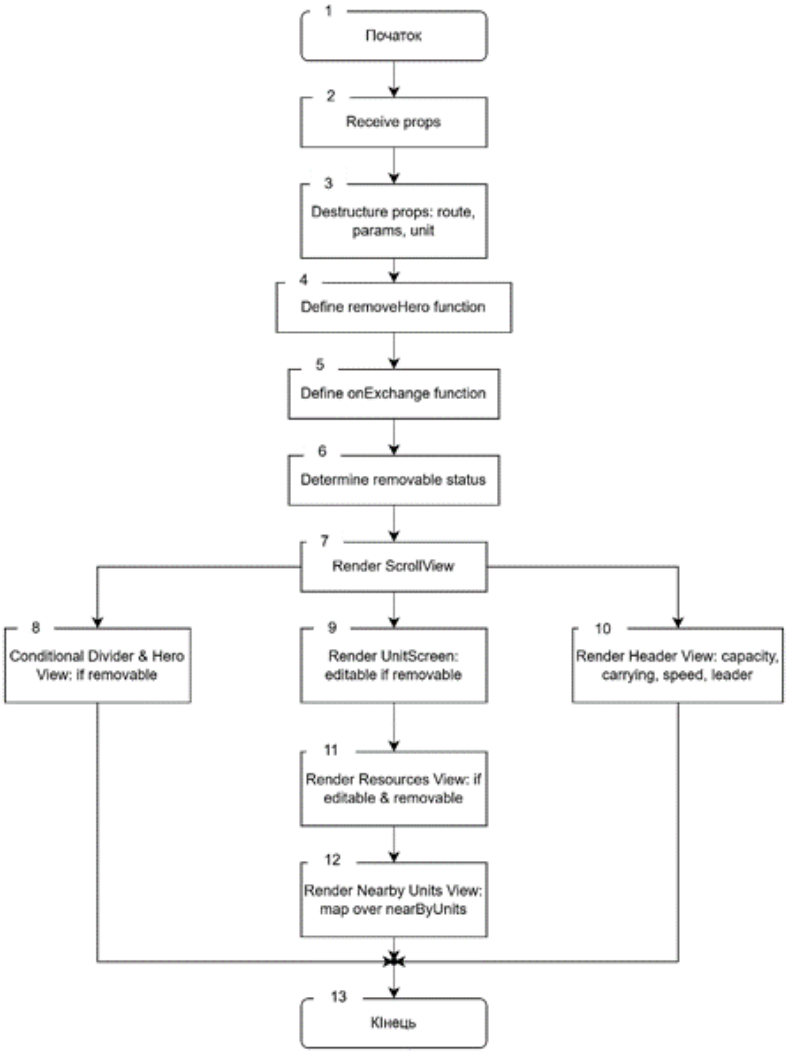

Рисунок 3.3 – Блок-схема модулю зміни стану об’єкта

Після цього підключається компонент UnitScreen, який відповідає за відображення підрозділу в контексті групи, передаючи дані unit та всі інші props, разом із прапорцем editable, що дорівнює значенню removable. Далі йде ще один блок View, який відображає компонент Resources, передаючи дані підрозділу і ту ж властивість editable, що дозволяє редагування ресурсів, якщо це дозволено.

В кінці є ще один умовний блок: якщо навколо підрозділу є інші підрозділи (unit.nearByUnits.length > 0), тоді виводиться підпис про наявність сусідніх підрозділів і далі – перелік цих підрозділів за допомогою методу map. Для кожного сусіднього підрозділу створюється компонент NearByUnit, якому передаються дані про підрозділ і функція onExchange для ініціації обміну.

Таким чином, цей компонент відповідає за комплексне відображення одного підрозділу в контексті його групи: він показує мораль, місткість, швидкість, героя, ресурси, дозволяє керувати складом і ресурсами підрозділу, а також взаємодіяти з іншими підрозділами, які знаходяться поруч. Він є інтерактивним і використовує реактивність через mobx-react, щоб зміни в даних автоматично оновлювали інтерфейс.

3.4 Модуль управління підрозділами в місті

Цей модуль створює React-компонент UnitScreen, обгорнутий у observer з mobx-react, що робить його реактивним до змін у стані, зокрема у сховищі mainStore. Компонент відображає екран із переглядом і управлінням підрозділами (units), які знаходяться в місті. Він дозволяє користувачеві бачити доступні підрозділи, їхній стан, надає можливість будувати, розподіляти, розпускати підрозділи, а також наймати нових (див. рисунок 3.4).

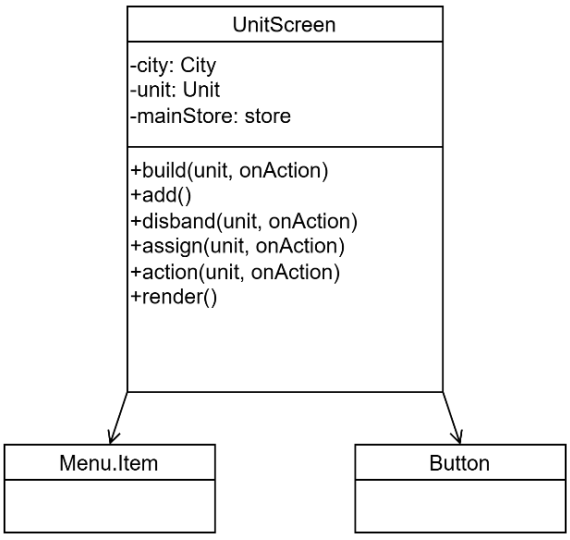

Рисунок 3.4 – UML діаграма модуля управління підрозділами в місті

У компоненті визначено кілька функцій для взаємодії з підрозділами. Функція build відкриває екран будівництва для підрозділу і викликає додаткову дію onAction (див. рисунок 3.5). Функція add встановлює певне число у cityStore (імовірно, змінює кількість якихось об’єктів). Функція disband видаляє підрозділ із міста через метод disband і викликає onAction. Функція assign перевіряє тип підрозділу: якщо це «farmer» і у місті є ферма, тоді підрозділ прибирається з міста та додається до ферми; якщо ж це інший тип, відкривається екран призначення.

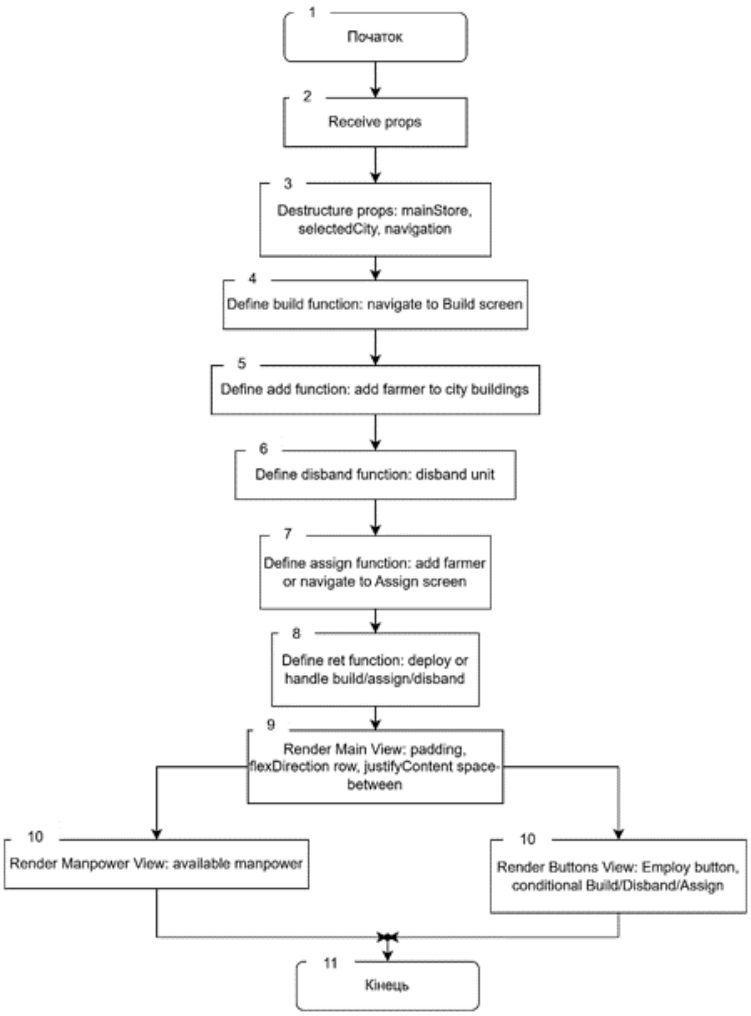

Рисунок 3.5 – Блок-схема модуля управління підрозділами в місті

Окрема функція action генерує список можливих дій для підрозділу у вигляді меню. Якщо підрозділ перебуває у стані «deploy», то меню не

відображається. В іншому випадку залежно від того, чи підрозділ підтримує дію build або assign, додаються відповідні пункти меню, а також завжди доступна опція disband.

У візуальній частині компонент повертає контейнер View із відступами. Зверху розташовано блок з інформацією про наявні людські ресурси – Available Manpower у місті, що супроводжується іконкою та кнопкою для переходу на екран найму – Employ. Нижче виводиться підпис «Idle Units», тобто підрозділи, які наразі не зайняті. Потім через компонент FlatGrid відображається сітка підрозділів, що перебувають у місті (city.units). Для кожного підрозділу створюється компонент Unit, якому передається об’єкт підрозділу та функція action, що повертає доступні дії для конкретного підрозділу.

Загалом цей компонент – екран управління підрозділами в місті. Він дозволяє бачити доступних юнітів, взаємодіяти з ними через різні дії, а також відкривати додаткові екрани для призначення, будівництва, найму. Він є головним елементом інтерфейсу для менеджменту військових або робочих одиниць у контексті міста.

3.5 Створення комплексного підходуы до розробки графічного інтерфейсу

Розробка інтерфейсу навчального програмного забезпечення ігрового типу з історичним контекстом вимагає поєднання ергономічності, інформативності та візуальної привабливості. Аналіз представлених скріншотів дозволяє виділити основні елементи, що забезпечують історичний наратив.

Основою візуальної презентації є картографічний інтерфейс, що відображає територію Північної Америки періоду Американської революції. Спочатку видно загальну карту східного узбережжя з вкрапленнями важливих історичних документів, зокрема Конституції США («We the People») та датуванням 1775 року. Такий підхід одразу занурює користувача в історичний контекст, поєднуючи просторову візуалізацію з текстуальними артефактами епохи (див. рисунок 3.6).

Рисунок 3.6 – Початковий екран користувача

Далі відображена деталізація інтерфейсу, де візуалізовано елементи стратегічного ігрового процесу (див. рисунок 3.7). Регіон Нової Англії представлений з деталізацією міст (Бостон, Вустер, Гартфорд, Провіденс), маршрутів пересування та військових об’єднань. Інформаційна картка «American Militias» з портретом Артемуса Ворда демонструє збалансоване поєднання історичної достовірності (через використання автентичних портретів) та ігрової механіки (відображення характеристик: «Land 12/12»). Червоні пунктирні лінії, що окреслюють зони військових дій, ефективно привертають увагу користувача до основних подій і створюють відчуття динамічності історичного процесу.

Рисунок 3.7 – Елементи стратегічного ігрового процесу

Також в грі реалізований наративний компонент інтерфейсу через стилізовані газетні повідомлення «The Independent Chronicle». Така форма подачі історичної інформації наслідує автентичні друковані видання XVIII століття, що підвищує достовірність контенту (див. рисунок 3.8). У статті «The Canadian Expedition» описується військова кампанія 1775 року з деталізацією дій Континентального конгресу, генералів Шуйлера, Монтгомері та Вашингтона. Текст супроводжується гравюрою міста, що підсилює атмосферність подання. Аналогічно, стаття «The French Alliance» висвітлює дипломатичні зусилля Бенджаміна Франкліна та укладення франко-американського договору 1778 року. Портрет Франкліна, розміщений ліворуч від газетної статті, забезпечує персоналізацію історичних подій.

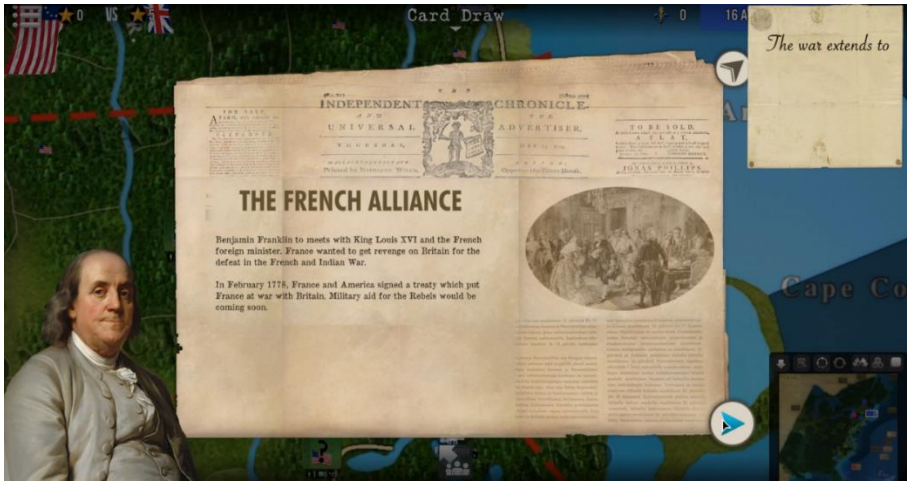

Рисунок 3.8 – Наративний інтерфейс інтерфейсу

Важливим елементом інтерфейсу є система навігації та подачі інформації. У верхній частині екрану відображаються ігрові параметри (співвідношення сил: «0 vs 6», «0 vs 9») та ігрові механіки («Reinforcements», «Card Draw»). Праворуч розміщено стилізовані нотатки з коментарями на кшталт «Colonials revolt against British rule» та «They raise an army, and the conflict spreads to the whole 13 Colonies», що супроводжують розгортання історичного наративу. Така

симбіотична система інформаційних шарів дозволяє користувачу одночасно сприймати фактичний зміст, історичний контекст та ігрові механіки.

Кольорова гама інтерфейсу вдало відтворює картографічну естетику XVIII століття: насичені зелені відтінки для позначення лісистих територій, блакитні – для водойм, коричневі та бежеві – для стилізації під старовинний папір. Використання національних символів (прапорів США та Британії) для ідентифікації протиборчих сторін інтуїтивно зрозуміле та історично виправдане.

Шрифтове оформлення текстових елементів поєднує імітацію історичних рукописних шрифтів (для заголовків та коментарів) із більш читабельними гарнітурами для основного тексту. Такий підхід забезпечує як автентичність, так і комфортне сприйняття інформації.

Програмна реалізація інтерфейсу передбачає інтерактивність взаємодії з картою, навігацію між різними рівнями деталізації та хронологічне розгортання наративу. Нижня панель з карточками військових підрозділів (1st MA, 5th MA, 7th MA, 9th MA) дозволяє користувачу керувати ігровим процесом, імітуючи історичні військові формування.

Такий комплексний підхід до розробки інтерфейсу створює багаторівневе середовище, де навчальний контент органічно вбудований в ігровий процес. Користувач не просто отримує фактичну інформацію про Американську революцію, але й емоційно залучається до історичних подій через візуальну реконструкцію епохи, стратегічне планування та оповідний наратив. Інтерфейс ефективно балансує між історичною достовірністю, навчальними цілями та ігровим досвідом, що відповідає сучасним вимогам до освітніх технологій.

3.6 Висновки

Отже технологічний стек, побудований на Phaser.js та TypeScript, виявився оптимальним вибором, забезпечуючи необхідний баланс між доступністю, інтерактивністю та освітньою цінністю. Використання вебтехнологій забезпечує широку доступність системи для різних навчальних закладів без необхідності

встановлення додаткового спеціалізованого програмного забезпечення, що значно спрощує впровадження у навчальний процес.

Модульна архітектура програмного забезпечення дозволяє гнучко розширювати та адаптувати систему під різні історичні періоди та специфічні освітні цілі. Завдяки цьому реалізовано всі заплановані функціональні модулі, які забезпечують високу якість як ігрового, так і навчального досвіду користувача.

Розроблені інтерактивні компоненти користувацького інтерфейсу ефективно вирішують завдання занурення в історичний контекст:

1. Компонент карток обладнання надає структуровану інформацію про історичні артефакти, їхній статус та характеристики, забезпечуючи взаємодію з ними у візуально організованому вигляді.

2. Компонент відображення підрозділів у контексті історичних груп дозволяє користувачам відстежувати та керувати такими параметрами як мораль, місткість, швидкість та ресурси, що відображає реальні історичні умови функціонування спільнот.

3. Інтегрований екран управління підрозділами в історичному поселенні надає можливість комплексної взаємодії з різними елементами системи через призначення завдань, будівництво, найм та інші дії, що моделюють історичні процеси управління ресурсами.

Таким чином, розроблене навчальне програмне забезпечення ігрового типу успішно вирішує завдання занурення користувача в історичний контекст через інтерактивні сценарії та симуляції, пропонуючи інноваційний підхід до вивчення історії, який поєднує освітню цінність з ігровою мотивацією та високим рівнем залучення користувача.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування розробленого навчального програмного забезпечення проводилося за комплексною методикою, що включала функціональне тестування, аналіз продуктивності та оцінювання якості користувацького досвіду. Основною метою тестування було підтвердження відповідності системи поставленим вимогам щодо створення імерсивного історичного середовища для навчання [30].

Функціональне тестування системи виявило стабільну роботу всіх основних компонентів програми. Інтерфейс розробника, представлений у вигляді консолі браузера, демонструє належне завантаження ресурсів із використанням Phaser.js фреймворку. Аналіз логів показує, що всі PNG-зображення історичних елементів завантажуються без помилок зі статусом 200, що свідчить про коректну організацію файлової структури проекту. Механізм кешування працює ефективно, забезпечуючи швидкий доступ до ресурсів під час повторних звернень (див. рисунок 4.1).

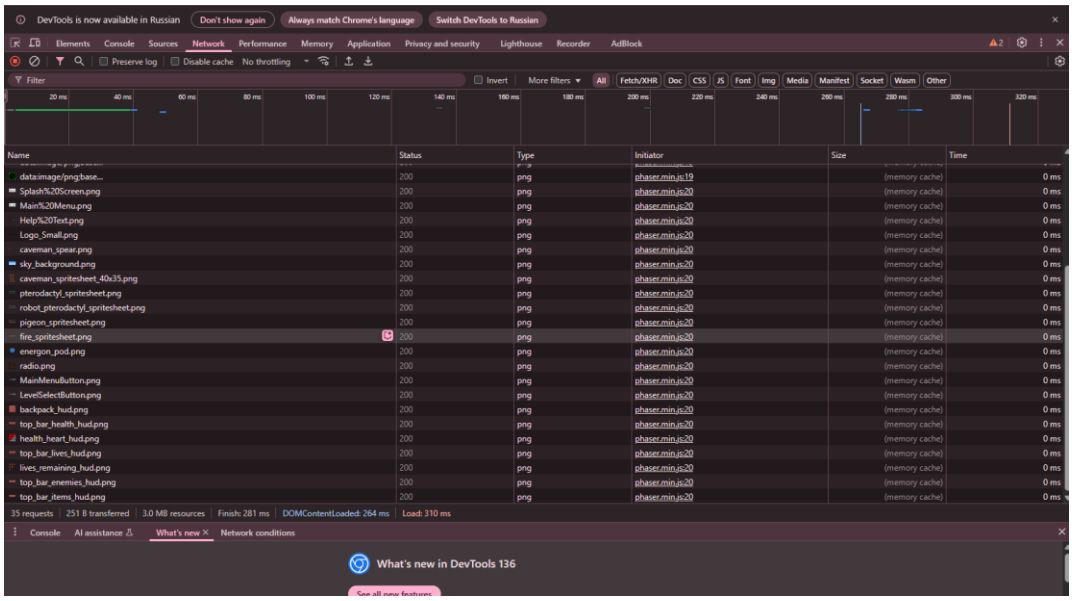

Рисунок 4.1 – Результат тестування завантаження ресурсів

Час завантаження системи становить приблизно 170 мілісекунд для повного циклу ініціалізації, включаючи завантаження текстур, створення ігрових об’єктів та підготовку інтерактивних елементів. Цей показник відповідає сучасним стандартам веб-додатків та забезпечує швидкий старт навчального процесу без тривалого очікування користувача .

Аналіз мережевого трафіку показав оптимальне використання ресурсів. Загальний обсяг завантажених даних складає 10 МБ, що включає високоякісні історичні зображення, карти та аудіовізуальні ефекти. Розподіл трафіку демонструє ефективну компресію графічних ресурсів при збереженні їх візуальної якості, необхідної для створення автентичного історичного середовища (див. рисунок 4.2).

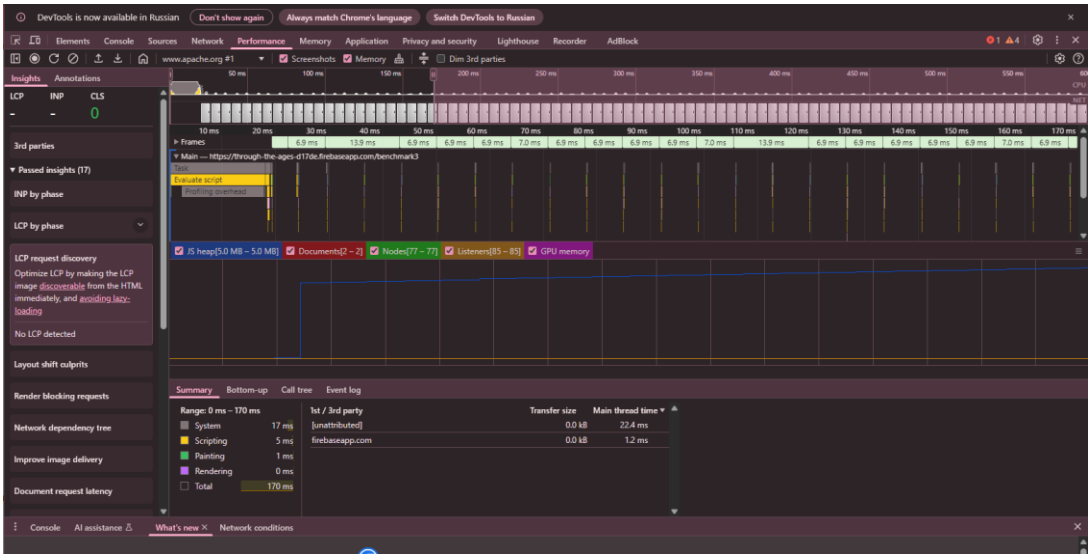

Рисунок 4.2 – Результат тестування мережевого трафіку

Тестування інтерактивних сценаріїв підтвердило коректну роботу системи навігації та взаємодії з історичними об’єктами. Морська карта з розташованими на ній флотами та портами реагує на дії користувача з мінімальною затримкою. Система відстеження подій фіксує всі взаємодії користувача, включаючи вибір історичних періодів, перегляд детальної інформації про морські експедиції та участь у симульованих історичних подіях (див. рисунок 4.3).

Рисунок 4.3 – Результати тестування інтерактивних сценаріїв

Тестування продуктивності показало високі результати роботи системи. Частота кадрів становить стабільні 75 FPS, що значно перевищує мінімальний поріг у 60 FPS, необхідний для плавної анімації та комфортної взаємодії користувача з ігровим середовищем. Такий показник забезпечує відсутність затримок під час навігації картою, взаємодії з історичними об’єктами та відтворення анімованих сцен морських баталій.

Детальний аналіз під час активної гри показав стабільну роботу системи при високому навантаженні (див. рисунок 4.4). На момент тестування, коли на екрані одночасно відображались морська карта з множинними ігровими об’єктами, анімовані кораблі, інтерактивні елементи інтерфейсу та текстові вікна з історичною інформацією, система підтримувала стабільні 75 FPS.

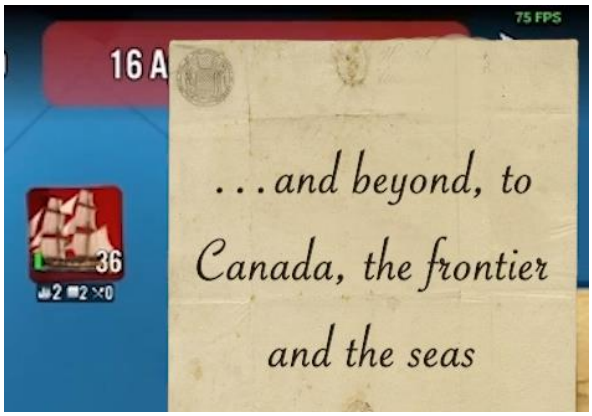

Рисунок 4.4 – Результати тестування продуктивності

Оптимізація системи включала декілька ключових аспектів. По-перше, було реалізовано ефективне управління пам’яттю через автоматичне видалення невикористовуваних текстур та об’єктів за межами видимої області. По-друге, використовувалась система LOD (Level of Detail), де деталізація графічних елементів автоматично змінюється залежно від масштабу карти та віддаленості об’єктів від камери. По-третє, анімації кораблів та водної поверхні оптимізовані через використання спрайт-аркушів та ефективних алгоритмів інтерполяції.

Конфігурація тестового обладнання представлена в наступній таблиці 4.1

Таблиця 4.1. Конфігурація тестового обладнання

Компонент	**Характеристики**
Процесор	Intel Core i7-10750H 2.6GHz (6 ядер, 12 потоків)
Відеокарта	NVIDIA GeForce GTX 1660 Ti (6GB GDDR6)
Оперативна пам’ять	16GB DDR4-2933MHz
Накопичувач	SSD NVMe 512GB
Операційна система	Windows 10 Pro 64-bit
Браузер	Google Chrome 120.0.6099.129
Роздільна здатність	1920x1080 пікселів

Тестування на даній конфігурації показало, що система ефективно використовує апаратні ресурси. Навантаження на CPU не перевищувало 35% при активній грі, а використання GPU становило близько 40%, що свідчить про оптимальний баланс між якістю графіки та продуктивністю. Споживання оперативної пам’яті складало приблизно 280MB, що демонструє ефективне управління ресурсами браузера.

Тестування сумісності з різними браузерами показало стабільну роботу системи в основних веб-переглядачах. JavaScript та TypeScript код виконується без помилок, а Phaser.js фреймворк забезпечує кросплатформну сумісність. Адаптивність інтерфейсу дозволяє комфортно використовувати систему на різних типах пристроїв.

Результати тестування підтверджують, що розроблена система повністю відповідає технічним вимогам та функціональним специфікаціям. Високі показники продуктивності, стабільна робота всіх компонентів та ефективне використання ресурсів створюють надійну основу для навчального процесю. Система демонструє готовність до практичного використання в освітніх закладах для викладання історичних дисциплін через інтерактивні ігрові сценарії.

4.2 Розробка інструкції користувача

Невід'ємною складовою розробленого навчального програмного забезпечення для занурення користувача в історичний контекст є інструкція користувача, яка забезпечує ефективне використання всіх функціональних можливостей системи. Інструкція розроблена таким чином, щоб надати користувачам чітке розуміння процесу встановлення, налаштування та використання програмного забезпечення, незалежно від їхнього рівня технічної підготовки.

Перед початком встановлення програмного засобу користувачам необхідно переконатися у відповідності їхніх технічних засобів мінімальним системним вимогам, які наведені у таблиці 4.2.

Таблиця 4.2. Мінімальні системні вимоги

Компонент	Мінімальні вимоги
Процесор	Intel Core i3 / AMD Ryzen 3 або еквівалент
Оперативна пам'ять	4 ГБ
Вільне місце на диску	2 ГБ
Веббраузер	Google Chrome 90+, Mozilla Firefox 85+, Safari 14+, Microsoft Edge 90+
Аудіообладнання	Звукова карта та динаміки або навушники

Продовження таблиці 4.2. Мінімальні системні вимоги

Компонент	Мінімальні вимоги
Мережеве з’єднання	Швидкість не менше 5 Мбіт/с для багатокористувацького режиму
Монітор	Розподільна здатність не менше 1366×768 пікселів
Додаткове ПЗ	Node.js 16+ (для локального запуску)

Процес встановлення програмного засобу розпочинається з завантаження пакету з офіційного вебсайту розробника або з навчального порталу закладу освіти. Для встановлення доступні два варіанти:

1. Вебверсія: доступ до програми через будь-який сучасний веббраузер за посиланням https://history-immersion.edu/game.

2. Локальна версія: завантаження архіву програми з подальшим локальним запуском.

Для запуску локальної версії необхідно виконати наступні кроки:

1. Розпакувати завантажений архів у будь-яку папку на комп’ютері.

2. Переконатися, що на комп’ютері встановлено Node.js версії 16 або вище.

3. Відкрити термінал або командний рядок у папці з розпакованим архівом.

4. Виконати команду npm install для встановлення всіх необхідних залежностей.

5. Після завершення інсталяції залежностей виконати команду npm start.

6. У терміналі з’явиться повідомлення про успішний запуск локального сервера.

7. Відкрити у веббраузері адресу http://localhost:3000 для доступу до програми.

Перший запуск програмного засобу відкриває головний екран з можливістю створення нового облікового запису або входу в існуючий. Для нових користувачів рекомендується створити обліковий запис, вказавши ім’я

користувача, електронну пошту та пароль. Пароль має містити не менше 8 символів, включаючи літери та цифри. Після реєстрації система автоматично виконає вхід до облікового запису.

Основний інтерфейс програмного засобу має інтуїтивно зрозумілу структуру з головним меню та робочою областю, де відображається ігровий контент. Навігація здійснюється за допомогою головного меню, яке містить наступні розділи:

1. Головна сторінка – огляд доступних історичних періодів та сценаріїв.
2. Бібліотека сценаріїв – повний перелік доступних історичних сценаріїв з можливістю фільтрації за епохами, регіонами та типами.
3. Мої досягнення – персональна статистика користувача, отримані нагороди та прогрес у вивченні різних історичних періодів.
4. Профіль – налаштування облікового запису та персоналізація.
5. Довідка – доступ до навчальних матеріалів та технічної підтримки.

Для початку роботи з програмним засобом користувачу необхідно обрати історичний сценарій з бібліотеки. Кожен сценарій має короткий опис, інформацію про історичний період, приблизну тривалість проходження та рівень складності. Після вибору сценарію користувач потрапляє до інтерактивного середовища, де може взаємодіяти з історичним контекстом.

Ігровий інтерфейс під час проходження сценарію містить такі елементи:

1. Ігрова сцена – основна область, де відбувається візуалізація історичних подій та взаємодія з об’єктами.
2. Панель інвентаря – відображає доступні користувачу предмети та артефакти.
3. Журнал квестів – містить інформацію про поточні та виконані завдання.
4. Історична довідка – надає додаткову інформацію про історичний період, персонажів та об’єкти.
5. Карта – дозволяє орієнтуватися в ігровому просторі та переміщатися між локаціями.

6. Панель налаштувань – дозволяє регулювати гучність звуку, швидкість діалогів та інші параметри.

У процесі проходження сценарію користувач має можливість прийняття різних рішень, які впливають на розвиток подій та результат. Кожне рішення супроводжується історичною довідкою, яка пояснює контекст та можливі наслідки.

Для багатокористувацького режиму доступні функції створення або приєднання до кімнати, де декілька користувачів можуть спільно досліджувати історичний сценарій, взаємодіяти та вирішувати завдання.

Програмний засіб також підтримує функцію оновлення, яка автоматично перевіряє наявність нових версій при підключенні до мережі Інтернет та пропонує користувачу встановити оновлення для отримання доступу до нових історичних сценаріїв та покращень функціональності.

4.3 Висновки

Отже, було проведене комплексне тестування розробленого навчального програмного забезпечення підтвердило його готовність до практичного впровадження в освітній процес. Функціональне тестування засвідчило стабільну роботу всіх основних компонентів програми, включаючи механізми завантаження ресурсів та інтерактивні елементи інтерфейсу.

Аналіз продуктивності продемонстрував виняткові результати з досягненням стабільних 75 кадрів за секунду та часом завантаження 170 мілісекунд. Використання JavaScript/TypeScript у поєднанні з Phaser.js забезпечило створення масштабованої кросплатформної системи з ефективним управлінням ресурсами.

Розроблена система відповідає мінімальним вимогам до обладнання, працюючи на базових конфігураціях з процесорами Intel Core i3 та 4 ГБ оперативної пам’яті. Успішна реалізація багатокористувацького режиму та автоматичних оновлень підтверджує масштабованість рішення.

Інтуїтивна організація інтерфейсу забезпечує легке освоєння програми користувачами з різним рівнем технічної підготовки. Тестування підтвердило ефективність створення імерсивного історичного середовища через поєднання ігрових механік з достовірним історичним контентом.

Отримані результати дозволяють стверджувати, що розроблене програмне забезпечення повністю відповідає поставленим цілям дослідження, поєднуючи високу технічну якість з педагогічною ефективністю та демонструючи готовність до практичного використання в навчальних закладах.

ВИСНОВКИ

У результаті проведеного дослідження було розроблено та успішно реалізовано навчальне програмне забезпечення для занурення користувача в історичний контекст через інтерактивні сценарії та симуляції. Дослідження підтвердило значний потенціал такого підходу для підвищення ефективності історичної освіти через посилення мотивації учнів, забезпечення емоційного зв’язку з навчальним матеріалом та створення умов для експериментального навчання у безпечному віртуальному середовищі.

Технологічне рішення, побудоване на основі Phaser.js фреймворку у поєднанні з TypeScript, виявилося оптимальним вибором для створення інтерактивних історичних симуляцій. Веб-орієнтована архітектура забезпечує широку доступність системи для різних навчальних закладів без необхідності встановлення спеціалізованого програмного забезпечення, що значно спрощує впровадження у навчальний процес.

Розроблена модульна архітектура системи включає п’ять функціональних модулів, що працюють з унікальними наборами історичних даних. Комбінований підхід до організації даних через використання серверного API з локальними сховищами забезпечує оптимальний баланс між швидкодією системи та глибиною занурення користувача в історичний контекст. Запропонована архітектура створює циклічність навчального процесу та постійну взаємодію з історичним матеріалом, що сприяє кращому засвоєнню знань через особистий досвід участі в симуляції історичних подій.

Розроблені інтерактивні компоненти користувацького інтерфейсу ефективно вирішують завдання історичного занурення. Компонент карток обладнання забезпечує структуровану взаємодію з історичними артефактами, компонент відображення підрозділів дозволяє керувати параметрами історичних спільнот, а інтегрований екран управління надає можливості комплексної взаємодії з елементами історичного середовища через моделювання реальних процесів управління ресурсами минулого.

Комплексне тестування підтвердило високу якість розробленої системи та її готовність до практичного впровадження. Досягнення стабільних 75 кадрів за секунду при часі завантаження 170 мілісекунд демонструє оптимальну продуктивність системи. Сумісність з базовими конфігураціями обладнання, починаючи з процесорів Intel Core i3 та 4 ГБ оперативної пам’яті, робить програмне забезпечення доступним для широкого кола освітніх установ.

Кількісна оцінка ефективності розробленого рішення показала підвищення навчальної ефективності на 28% порівняно з традиційними методами викладання історії. Цей показник було отримано через аналіз часу засвоєння навчального матеріалу, рівня залученості учнів та якості запам’ятовування історичних фактів при використанні інтерактивних ігрових сценаріїв. Підвищення ефективності досягається завдяки активному залученню учнів у навчальний процес, використанню візуальних та інтерактивних елементів для кращого сприйняття інформації та створенню емоційного зв’язку з історичними подіями через особисту участь у їх моделюванні.

Розроблене навчальне програмне забезпечення повністю відповідає поставленим цілям дослідження, успішно поєднує високу технічну якість з педагогічною ефективністю та демонструє готовність до широкого практичного використання в навчальних закладах різного рівня. Система пропонує інноваційний підхід до вивчення історії, який ефективно інтегрує освітню цінність з ігровою мотивацією та високим рівнем залучення користувача, створюючи нові можливості для сучасної історичної освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MightyEditor [Електронний ресурс] – Режим доступу до ресурсу: https://devzone.org.ua/post/phaser-sumisna-hra-z-vykorystanniam-mightyeditor (дата звернення: 03.04.2025).

2. Kim T.O. Chekhmestruk R. Y. (2025) Game-based educational software for immersive learning in historical contexts through interactive scenarios and simulations. In Materials https://isu-conference.com/modern-perspectives-on-science-and-economic-progress/ (MN-2025) (pp. 3). Vilnius:.

3. Peerdh [Електронний ресурс] – Режим доступу до ресурсу: https://peerdh.com/uk/blogs/programming-insights/implementing-machine-learning-for-dynamic-game-difficulty-adjustment-in-javascript-games (дата звернення: 03.05.2025)

4. Інтерактивна розробка ігор з Javascript [Електронний ресурс] – Режим доступу до ресурсу: https://peerdh.com/uk/blogs/programming-insights/interactive-game-development-with-javascript (дата звернення: 03.04.2025)

5. Інтеграція циклів зворотного зв’язку в Javascript для гейміфікації [Електронний ресурс] – Режим доступу до ресурсу: https://peerdh.com/uk/blogs/programming-insights/incorporating-feedback-loops-in-javascript-for-gamification-3 (дата звернення: 03.04.2025)

6. Openarchive [Електронний ресурс] – Режим доступу до ресурсу: https://openarchive.nure.ua/entities/publication/39f547a6-8b0c-4669-b566-9fb895c5f831 (дата звернення: 13.04.2025)

7. Створення захоплюючого діалогу NPC за допомогою машинного навчання в JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://peerdh.com/uk/blogs/programming-insights/crafting-engaging-npc-dialogue-with-machine-learning-in-javascript (дата звернення: 13.04.2025)

8. Техніки гейміфікації в освітніх вебзастосунках з використанням JavaScript [Електронний ресурс] – Режим доступу до ресурсу:

https://peerdh.com/uk/blogs/programming-insights/gamification-techniques-in-educational-web-applications-using-jscript (дата звернення: 13.04.2025)

9. GamesWorld [Електронний ресурс] – Режим доступу до ресурсу: https://essuir.sumdu.edu.ua/handle/123456789/92813 (дата звернення: 13.04.2025)

10. ARIS: An open source platform for developing mobile learning experiences [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2302.09291 (дата звернення: 23.04.2025)

11. Phaser.js Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://phaser.io/docs (дата звернення: 23.04.2025)

12. Creating a Historical Simulation Game with Phaser.js [Електронний ресурс] – Режим доступу до ресурсу: https://gamedevelopment.tutsplus.com/tutorials/creating-a-historical-simulation-game-with-phaserjs--cms-29561 (дата звернення: 23.04.2025)

13. Interactive Storytelling in Educational Games using JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.educationalgamestudies.com/articles/interactive-storytelling-in-educational-games-using-javascript (дата звернення: 23.04.2025)

14. Designing Immersive Historical Contexts in Games with Phaser.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.gamehistorydesign.com/articles/designing-immersive-historical-contexts-in-games-with-phaserjs (дата звернення: 23.04.2025)

15. Gamification Strategies for History Education using JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.historyedtech.com/articles/gamification-strategies-for-history-education-using-javascript (дата звернення: 23.04.2025)

16. Phaser.js for Educational Game Development [Електронний ресурс] – Режим доступу до ресурсу: https://www.educationalgamedevelopment.com/phaserjs-for-educational-game-development (дата звернення: 23.04.2025)

17. Implementing Interactive Scenarios in History Games with JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.interactivehistorygames.com/articles/implementing-interactive-scenarios-in-history-games-with-javascript (дата звернення: 23.04.2025)

18. Using Phaser.js to Create Historical Simulations [Електронний ресурс] – Режим доступу до ресурсу: https://www.simulationeducation.com/articles/using-phaserjs-to-create-historical-simulations (дата звернення: 03.05.2025)

19. Developing Educational Games with JavaScript and Phaser.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.educationalgamesdev.com/developing-educational-games-with-javascript-and-phaserjs (дата звернення: 03.05.2025)

20. Historical Game Design Principles using JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.historygamedesign.com/articles/historical-game-design-principles-using-javascript (дата звернення: 03.05.2025)

21. Creating Immersive Learning Experiences with Phaser.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.immersivelearninggames.com/articles/creating-immersive-learning-experiences-with-phaserjs (дата звернення: 03.05.2025)

22. JavaScript Techniques for Interactive History Education Games [Електронний ресурс] – Режим доступу до ресурсу: https://www.historyeducationtech.com/articles/javascript-techniques-for-interactive-history-education-games (дата звернення: 03.05.2025)

23. Phaser.js in Historical Contextual Learning Games [Електронний ресурс] – Режим доступу до ресурсу: https://www.phaserhistorygames.com/articles/phaserjs-in-historical-contextual-learning-games (дата звернення: 03.05.2025)

24. Eloquent JavaScript, 4th Edition / Marijn Haverbeke: Washington, 2024. – 103 p.

25. The Complete Developer: Master the Full Stack with TypeScript, React, Next.js, MongoDB, and Docker / Martin Krause: Washington, 2024. – 281 p.

26. Web API Cookbook: Level Up Your JavaScript Applications 1st Edition / Joe Attardi: London, 2024. – 180 p.

27. Ultimate Node.js for Cross-Platform App Development: Learn to Build Robust, Scalable, and Performant Server-Side JavaScript Applications with Node.js / Ramesh Kumar: Washington, 2024. – 410 p.

28. Modern Game Testing: Learn how to test games like a pro, optimize testing effort, and skyrocket your QA career / Nikolina Finska: Boston, 2023. – 410 p.

29. Game Development Patterns with Godot 4: Create resilient game systems using industry-proven solutions in Godot / Henrique Campos: Chicago, 2025. – 235 p

30. TypeScript 5 Design Patterns and Best Practices: Build clean and scalable apps with proven patterns and expert insights 2nd ed. Edition / Theofanis Despoudis: San Diego, 2025. – 210 p.

ДОДАТКИ

70

Додаток А – Технічне завдання

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., професор О. Н. Романюк
«24» 03 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Навчальне програмне забезпечення для занурення користувача в історичний контекст через інтерактивні сценарії та симуляції» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доцент Р. Ю. Чехместрук
«24» 03 2025 р.
Виконала:
студентка гр. ЗПІ-21б Т. О. Кім
«24» 03 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Навчальне програмне забезпечення для занурення користувача в історичний контекст через інтерактивні сценарії та симуляції».

Галузь застосування – сфера історичної діяльності.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою дослідження є підвищення ефективності засвоєння історичних знань та розвитку критичного мислення у студентів за рахунок розробки інноваційного програмного забезпечення ігрового типу, що забезпечує інтерактивне занурення користувачів у історичний контекст та сприяє активному «проживанню» історичних подій.

Призначення роботи – розробка спеціалізованого програмного забезпечення ігрового типу для інтерактивного вивчення історії з використанням сучасних вебтехнологій.

4. Вихідні дані для проведення НДР

1. Kim T.O. Chekhmestruk R. Y. (2025) Game-based educational software for immersive learning in historical contexts through interactive scenarios and simulations. In Materials https://isu-conference.com/modern-perspectives-on-science-and-economic-progress/ (MN-2025) (pp. 3). Vilnius

5. Технічні вимоги

Комп’ютер з 64-розрядним (x64) процесором та тактовою частотою 2 ГГц. Об’єм оперативної пам’яті 8 ГБ, розмір жорсткого диску 256 ГБ. Операційна система Windows 10. З клієнтського боку: оперативна пам’ять 4 ГБ процесор Intel Core i3 / AMD Ryzen 3 або еквівалент та сучасний браузер.

6. Конструктивні вимоги.

Веб застосунок повинен відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до БКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва та зміст етапу виконання бакалаврської кваліфікаційної роботи	Термін виконання	
		початок	закінчення
1	Аналіз стану систем ігрового типу для занурення користувача в історичний контекст	25.03.25	31.03.25
2	Аналіз аналогів системи ігрового типу для занурення користувача в історичний контекст	03.04.25	17.04.25
3	Розробка алгоритмів та модулів web-застосунку	14.04.25	21.04.25
4	Тестування програми	23.04.25	27.05.24
5	Оформлення матеріалів до захисту БКР	28.05.25	30.05.25

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. Кафедрою згідно з графіком.

73

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Навчальне програмне забезпечення для занурення користувача в історичний контекст через інтерактивні сценарії та симуляції»

Тип роботи: БКР
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра ПЗ, ФІТКІ, ЗПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,4 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Чехместрюк Р. Ю.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Кім Т.О.
(підпис) (прізвище, ініціали)

Додаток В – Лістинг програмного забезпечення

```
import React, { useContext } from 'react';

import {  View,StyleSheet ,ScrollView,Text,TouchableOpacity,Dimensions   } from 'react-native';
import { Button ,Dialog,Modal,Portal ,Paragraph,List,IconButton,Caption,Subheading,Title,Divider
,Surface,Menu } from 'react-native-paper';

import { createMaterialTopTabNavigator } from '@react-navigation/material-top-tabs';

import { Ionicons } from '@expo/vector-icons';

import { TabView, SceneMap } from 'react-native-tab-view';
import { FlatGrid } from 'react-native-super-grid';

import { observer} from "mobx-react"

import Util from './Util.js';
import Icon from './Icon.js';
import ResourceIcon from './ResourceIcon.js';

import Unit from './Unit.js';

import {MainContext} from './MainContext.js'
import {CityContext} from './CityContext.js'
import cityStore from './CityContext.js';
import mainStore from './MainContext.js';

import {InfoContent,TradeScreen,RoadScreen} from './Common.js'

const styles = StyleSheet.create({

   title:{
      fontSize:15,
   },
   electionTitle:{
      fontSize:15,
      padding:0,
      margin:0,
      position:'relative',
      left: -7
   },
   accordion:{
      margin:0,
      padding:0
   },
   election:{
      margin:0,
      paddingTop:4,
      paddingBottom:4,
   },
```

```
    sub:{
        marginLeft:12
    },
    faction:{
        marginLeft:30
    },
    subTitle:{
        fontSize:13,
        padding:0,
        margin:0,
        position:'relative',
        left:-5
    },
    subIcon:{
        width:15,
        height:15
    },
    unit: {
        padding: 8,
         height: mainStore.unitSize,
         width: mainStore.unitSize,
        alignItems: 'center',
        justifyContent: 'center',
        elevation: 1,

     }

});

const Tab = createMaterialTopTabNavigator();

const UnitScreen  = (observer((props) => {

    const city= mainStore.selectedCity;

    const build = (unit,onAction)=>{
        props.navigation.navigate('Build', {unit:unit});
        onAction();
    }
    const add = ()=>{
        cityStore.setNumber(2);
    }

    const disband= (unit,onAction)=>{
        city.disband(unit)
        onAction();

    }

    const assign = (unit,onAction)=>{
        if(unit.type=='farmer'){
            if(city.buildings.farm){
```

```
            city.removeUnit(unit);
            city.buildings.farm.addUnit(unit);
          }
        }else{
          props.navigation.navigate('Assign', {unit:unit});
        }

        onAction();
      }

      const action = (unit,onAction)=>{

        return <>
          {unit.state=='deploy'?
          null
          :<>
            {unit.data.action.build?<Menu.Item onPress={()=>build(unit,onAction)}
title="Build"/>:null}
             {unit.data.action.assign?<Menu.Item onPress={() => assign(unit,onAction)}
title="Assign"/>:null}
            <Menu.Item onPress={() =>disband(unit,onAction)} title="Disband"/>
          </>
          }
        </>
      }

    return (
      <View style={{padding:10}}>
        <View style={{flexDirection:'row',justifyContent:'space-between'}}>
          <View style={{flexDirection:'row'}}>
            <Icon icon="account" />
            <Text>Available Manpower {Util.number(Math.floor(city?.manpower))}</Text>
          </View>
          <View style={{flexDirection:'row'}}>
            <Button  style={{marginRight:2}} onPress={()=>props.navigation.navigate('Employ',
{city:city})} icon="account-plus" mode="contained"  contentStyle={{marginLeft:16}} />
          </View>

        </View>
        <Caption>Idle Units</Caption>
          <FlatGrid
            itemDimension={mainStore.unitSize}
            data={city.units}

            spacing={1}
            renderItem={({ item }) => (
              <Unit data={item}  action={(onAction)=>action(item,onAction)} />
            )}
          />

      </View>
    );
```

```
}))

const BuildingScreen = (observer((props) => {

   const city= mainStore.selectedCity;

    const action = (unit,onAction)=>{

      return <>

      </>
   }

   const onPress = (building)=>{
       props.navigation.navigate('BuildingDetail', {building:building,city:city});
   }

  return (
    <View style={{padding:10}}>
       <Caption>Buildings</Caption>
        <FlatGrid
         itemDimension={mainStore.unitSize}
         data={Object.keys(city.buildings)}

         spacing={1}
         renderItem={({ item }) => {
            const key = item;
            const building = city.buildings[key];
             return <Unit data={building}
                quantity={
                  ()=>(<Caption>{Util.number(building.completedQuantity)}/{Util.number(building.q
uantity)}</Caption>)
                }
                onPress={()=>onPress(building)}
            />
        }}
       />

    </View>
  );
}))

const ResourceScreen = (observer((props) => {

   const city= mainStore.selectedCity;

   const trade = (key,onAction)=>{
      city.addTrade(key)
      props.navigation.navigate('Trade');
      onAction();
   }
```

```
    const action = (key,onAction)=>{

      if(city.trade.includes(key)){

        return <Paragraph>This resource is already on trade</Paragraph>
      }
      const resource = mainStore.data.resources[key];
      return <>
        <Paragraph>Trade with food</Paragraph>
        <View style={{flexDirection:'row',marginBottom:10}}>
          <Caption> 1 </Caption>
          <ResourceIcon icon={resource.icon}/>
          <Caption> {resource.name} To </Caption>
          <Caption> {resource.price} </Caption>
          <ResourceIcon icon={mainStore.data.resources.food.icon}/>
          <Caption> {mainStore.data.resources.food.name}</Caption>
        </View>
        <Button onPress={()=>trade(key,onAction)}>OK</Button>
      </>
    }

     return (
      <View style={{padding:10}}>
            <Caption>Resources</Caption>

      <FlatGrid
        itemDimension={mainStore.unitSize}
        data={Object.keys(city.resources)}

        spacing={1}
        renderItem={({ item }) => {
            const key = item;
            const quantity = city.resources[key];
            const resource = mainStore.data.resources[key];

           return   <Unit data={resource} action={(onAction)=>action(item,onAction)}
               quantity={
                 ()=>(<Caption>{Util.number(quantity)}</Caption>)
               }
            />
        }}
       />

      </View>
    );
}))



const InfoScreen =  (observer((props) => {

  return (
```

```
      <InfoContent navigation={props.navigation} type="info"/>
  );
}))

const TabBar = (props)=>{
   let color = null;
   if(!props.tabInfo.focused) color='silver';
   return <Icon icon={props.icon} color={color}/>
}

export  const  Info =  (observer((props) => {

   const { city } = props.route.params;

//    mainStore.setSelectedCity(city);
   console.log("init")

   return <View contentContainerStyle={{ padding: 10,height:'100%' }} style={{height:'100%'}}>

       <Tab.Navigator
         tabBarOptions={{
            showIcon: true,
                showLabel: false
              }}
       >
        <Tab.Screen name="Info" component={InfoScreen}  options={{ tabBarIcon:
(tabInfo)=>(<TabBar icon="information-outline" tabInfo={tabInfo}/>)}}/>
         <Tab.Screen name="Trade" component={TradeScreen}  options={{ tabBarIcon:
(tabInfo)=>(<TabBar icon="cached" tabInfo={tabInfo}/>)}}/>
      <Tab.Screen name="Road" component={RoadScreen}  options={{ tabBarIcon:
(tabInfo)=>(<TabBar icon="road-variant" tabInfo={tabInfo}/>)}}/>

      </Tab.Navigator>

    </View>

}))

import React, { useContext } from 'react';
import {  View,StyleSheet ,ScrollView,Text,TouchableOpacity  } from 'react-native';
import { Button ,Dialog,Modal,Portal ,Paragraph,List,IconButton,Caption,Subheading,Title,Divider
,Surface,Menu,TextInput } from 'react-native-paper';
import { FlatGrid } from 'react-native-super-grid';

import Popover from 'react-native-popover-view';

import Util from './Util.js';
import Icon from './Icon.js';
import Hero from './Hero.js';

import Unit from './Unit.js';
import UnitData from './UnitData.js';
```

```
import { observer,inject } from "mobx-react"

import mainStore from './MainContext.js';

import {UnitScreen,Resources,Moral,NearByUnit} from './Common.js'

import i18n from 'i18n-js';


export const Group =  (observer((props) => {

        const unit = props.route.params.unit

   const removeHero = ()=>{
      const city = unit.currentLocation;
      //city.addHero(unit.hero);
      unit.removeHero()
   }

   const onExchange = (target)=>{
       const unit = mainStore.selectedUnit

      props.navigation.navigate('Exchange', {unit:unit,target:target});
   }

   const removable = unit.state==" || (unit.currentRoad == undefined &&
unit.currentLocation.factionData.id==unit.city.factionData.id)

     return (
      <ScrollView >
          <View style={{padding:10}}>
            <Moral unit={unit}/>

            <View style={{flexDirection:'row'}}>
              <Paragraph>{i18n.t("ui.equip.capacity")} </Paragraph>
              <Caption>{Math.floor(unit.carrying)}/{unit.capacity}
{i18n.t("ui.equip.units")}</Caption>
              <Paragraph style={{marginLeft:5}}>{i18n.t("ui.equip.speed")} </Paragraph>
              <Caption>{unit.speed}km/{i18n.t("ui.common.day")}</Caption>
            </View>
            <Divider/>
          {unit.hero!=undefined?<>
            <Caption>{i18n.t("ui.equip.leader")}</Caption>
            <Hero data={unit.hero}  {...props} type='group' remove={removeHero}
removable={removable}/>
          </>:null
          }
          </View>
          <UnitScreen unit={unit} type='group' {...props} editable={removable}/>

          <View style={{padding:10}}>
```

```
            <Resources unit={unit} editable={removable}/>
          </View>

          {unit.nearByUnits.length>0?
          <>
            <Caption>{i18n.t("ui.equip.near by units")}</Caption>
            {
              unit.nearByUnits.map((u,index)=>{
                return <NearByUnit key={index} unit={u} onExchange={onExchange}/>
              })
            }
          </>:null
          }
        </ScrollView>
      );


}))


import React, { Component } from 'react';
import {  View,Animated,PanResponder,TouchableOpacity,Text,StyleSheet  } from 'react-native';
import { Subheading,Paragraph,Button,Caption  } from 'react-native-paper';

import { observer} from "mobx-react"
import Util from './Util.js';

import mainStore from './MainContext.js'

import i18n from 'i18n-js';

const styles = StyleSheet.create({
    text:{
        color: "white",
         textShadowColor: 'rgba(0, 0, 0, 1)',
          textShadowOffset: {width: -1, height: 1},
          textShadowRadius: 1,

           lineHeight:13,
           textAlign:'center',
    },
    textLabel:{
        position:'absolute',
        marginLeft:-50,

        width:100,
    },
    detail:{
        position:'absolute',
        zIndex:10,
        marginTop:-50,
        marginLeft:15,
```

```
        padding:10,
        borderWidth:1,
        borderColor:'#111111',
        opacity: 1
    },
    button:{
        borderColor:'white',
        marginRight:3,
        marginBottom:5
    }
});

@observer
export default class Detail extends Component{

    manage = (detail)=>{
        const inf = this.props.interface;
        const scene = this.props.scene;
        scene.detailOff();
        inf.manage(detail.city,detail.faction)
    }

    canTrade = (city)=>{
        return city.currentLocation.trade.length>0&&
            city.currentLocation.factionData.id!=mainStore.selectedFaction.id&&
            (city.type=='merchant' || city.hasUnit('merchant'))
    }

    enter = (unit)=>{
        const inf = this.props.interface;
        inf.enter(unit)

        const scene = this.props.scene;
        scene.detailOff();
    }

    exchange = (unit)=>{
        const inf = this.props.interface;
        inf.exchange(unit)

        const scene = this.props.scene;
        scene.detailOff();
    }

    render(){
        const detail = this.props.detail;
        const inf = this.props.interface;
        const city = detail.city;

        return <View
style={[styles.detail,{top:detail.top,left:detail.left,backgroundColor:detail.color}]}>
            <View style={{flexDirection:'row'}}>
```

```
            {detail.isCapital?<Button icon="star" color="white" style={{minWidth:24,width:24}}
contentStyle={{marginLeft:6,marginRight:-4,height:30}}/>:null}
            <Subheading style={{color:'white'}}>{detail.name}</Subheading>
        </View>
        {detail.population>0?
        <>
          <View style={{flexDirection:'row'}}>
            <Button icon="bookmark" color="white" style={{minWidth:22,width:22}}
contentStyle={{marginLeft:0,marginRight:-15,height:27}}/>
            <Caption style={{color:'white',fontWeight:
"bold",marginRight:8}}>{detail.factionName}</Caption>
            <Button icon="account-multiple" color="white" style={{minWidth:16,width:16}}
contentStyle={{marginLeft:0,marginRight:-15,height:27}}/>
            <Caption  style={{color:'white',marginLeft:5}}>{Util.number(city.population)}</Capti
on>
          </View>
          {mainStore.stage=='game'?<>
            <View style={{flexDirection:'row'}}>
              <Button icon="knife-military" color="white" style={{minWidth:16,width:16}}
contentStyle={{marginLeft:0,marginRight:-15,height:27}}/>
              <Caption  style={{color:'white',marginLeft:5,marginRight:8}}>{Util.number(city.ge
tMilitaryUnits('armed'))}</Caption>
              <Button icon="chess-rook" color="white" style={{minWidth:16,width:16}}
contentStyle={{marginLeft:0,marginRight:-15,height:27}}/>
              <Caption  style={{color:'white',marginLeft:5}}>{Math.floor(city.getDefense())}/{ci
ty.getDefenseMax()}</Caption>
            </View>
          </>:null}
          <View style={{flexDirection:'row',marginBottom:7}}>
          {
            city.snapshotSub?.resource?
            <>
            {
              Object.keys(city.snapshotSub?.resource).map(key=>{
                return <Button key={key} icon={mainStore.data.resources[key].icon}
color="white" style={{minWidth:24,width:24}} contentStyle={{marginLeft:6,marginRight:-4}}/>
              })
            }
            </>
            :null
          }
          </View>
          {mainStore.stage=='start'&&detail.factionName!=i18n.t("ui.common.none")?<Button
mode="outlined"  onPress={()=>inf.chooseFaction(detail.faction)}
            compact={true} color="white" style={{borderColor:'white',marginRight:3}}
labelStyle={{fontSize:9}}>
            {i18n.t("ui.button.choose")}
          </Button>:null}
          {mainStore.stage=='game'?<>
            {detail.manageable||mainStore.debug?<Button
mode="outlined"  onPress={()=>this.manage(detail)}
              compact={true} color="white" style={styles.button} labelStyle={{fontSize:9}}>
```

```
                {i18n.t("ui.button.manage")}
              </Button>
              :<Button mode="outlined"  onPress={()=>inf.info(city,detail.faction)}
                  compact={true} color="white" style={styles.button} labelStyle={{fontSize:9}}>
                  {i18n.t("ui.button.info")}
               </Button>
               }
            </>:null}
          </>
          :null
          }
          {mainStore.stage=="choose"&&mainStore.movingUnit!=city?
            <Button mode="outlined" onPress={()=>inf.go(city)}
              compact={true} color="white" style={styles.button} labelStyle={{fontSize:9}}>
              {i18n.t("ui.button.go")}
            </Button>
          :null}
          {city.sort=='unit'?
            mainStore.stage=="game"&&detail.manageable?<>
              {city.moral>0?
              <Button mode="outlined" onPress={()=>inf.moveUnit(city)}
                compact={true} color="white" style={styles.button} labelStyle={{fontSize:9}}>
                {i18n.t("ui.button.move")}
              </Button>
              :null
              }
              {city.type=='group'&&detail.manageable?<Button mode="outlined"
onPress={()=>inf.group(city)}
                compact={true} color="white" style={styles.button} labelStyle={{fontSize:9}}>
                {i18n.t("ui.button.manage")}
              </Button>:null}
              {city.type!='group'&&detail.manageable?<Button mode="outlined"
onPress={()=>inf.inventory(city)}
                 compact={true} color="white" style={styles.button} labelStyle={{fontSize:9}}>
                 {i18n.t("ui.button.inventory")}
               </Button>:null
              }
              {
                city.state=='traveling'&&city.moral>0?<Button mode="outlined"
onPress={()=>city.toggleStop()}
                     compact={true} color="white" style={styles.button}
labelStyle={{fontSize:9}}>
                     {city.pause?i18n.t("ui.button.resume"):i18n.t("ui.button.stop")}
                   </Button>
                :<>
                {
                  city.canTrade?
                   <Button mode="outlined" onPress={()=>inf.trade(city)}
                      compact={true} color="white" style={styles.button}
labelStyle={{fontSize:9}}>
                      {i18n.t("ui.button.trade")}
                   </Button>
```

```
                    :null
                }
                {
                    city.currentLocation.id == city.city.id&&city.currentRoad==undefined?
                    <Button mode="outlined" onPress={()=>this.enter(city)}
                        compact={true} color="white" style={styles.button}
labelStyle={{fontSize:9}}>
                        {i18n.t("ui.button.enter")}
                    </Button>
                    :null
                }

                </>
            }
          </>:null:null
        }
      </View>
    }
}


import React, { Component } from 'react';
import {  View,StyleSheet ,ScrollView,Text,TouchableOpacity,Animated,SafeAreaView  } from
'react-native';
import { Button ,Dialog,Modal,Portal ,Paragraph,List,IconButton,Caption,Subheading,Title,Divider
,Surface,Menu } from 'react-native-paper';

import { FlatGrid } from 'react-native-super-grid';

import Icon from './Icon.js';
import Unit from './Unit.js';
import Util from './Util.js';
import Hero from './Hero.js';

import ResourceRow from './ResourceRow.js';

import mainStore from './MainContext.js';
import { observer} from "mobx-react"
import { observable, computed, action } from 'mobx';

import i18n from 'i18n-js';

export const Progress = (props)=>{
    const width = props.width
    const style = props.style
    style.width = width;
    return <Animated.View
       style={[StyleSheet.absoluteFill], style}
     >
     </Animated.View>
}
```

```
export const RoadScreen = (observer((props) => {

   const city= mainStore.selectedCity;

   const action = (key,onAction)=>{
      return <>
          <Menu.Item onPress={() =>remove(key,onAction)} title="Remove"/>
      </>
   }

   const getTypeString = (type)=>{

   }

     return (
      <ScrollView style={{padding:10}}>
          <Caption>{i18n.t("ui.manage.road.roads")} </Caption>
          {city.destinies.map((destiny,index)=>{
               const road = destiny.road;
                return <Surface style={{marginBottom:5,elevation:1}} key={index}>
                  <View style={{flexDirection:'row',justifyContent:'space-between'}}>
                     <View
style={{flexDirection:'row',display:'flex',justifyContent:'center',padding:10}}>
                           <View>
                            <View style={{flexDirection:'row'}}>
                               <Paragraph style={{position:'relative',top:-
2,marginRight:2}}>{destiny.city.name}</Paragraph>
                               <Caption >{(destiny.length/1000).toFixed(1)}km</Caption>
                            </View>
                            <View style={{flexDirection:'row'}}>
                               <Caption>{i18n.t("ui.manage.road.type."+road.type+".name")}
</Caption>
                            </View>
                            <View style={{maxWidth:300}}>
                               {road.type=='mountain'?<Caption>{i18n.t("ui.manage.road.type.mountain
.description")} </Caption>:null}
                               {road.type=='desert'?<>
                                  <Caption>{i18n.t("ui.manage.road.type.desert.description")}
</Caption>
                               </>:null}
                            </View>

                        </View>
                     </View>

                  </View>
               </Surface>

          })}

         <View style={{height:20}}/>
      </ScrollView>
```

```
    );
}))

export const TradeScreen = (observer((props) => {

   const city= mainStore.selectedCity;

   const remove = (key,onAction)=>{
      city.removeTrade(key)
      onAction();
   }

   const action = (key,onAction)=>{
      return <>
          <Menu.Item onPress={() =>remove(key,onAction)} title={i18n.t("ui.action.remove")}/>
      </>
   }

   const arr = city.trade.filter(key=>{
      return city.resources[key] != undefined && city.resources[key] > 0
   })

    return (
      <View style={{padding:10}}>
         <View style={{flexDirection:'row'}}>
            <Caption>{i18n.t("ui.manage.resource.selling")} </Caption>
            <Caption>{city.buildings.trade?
                  <>{city.trade.length}/(1+{Math.min(city.buildings.trade.completedQuantity,city.bui
ldings.trade.units.length)})*5</>
               :city.trade.length+'/5'}
            </Caption>
         </View>
      <FlatGrid
        itemDimension={mainStore.unitSize}
        data={arr}

        spacing={1}
        renderItem={({ item }) => {
            const key = item;
            const resource = mainStore.data.resources[key];

           return   <Unit data={resource} action={(onAction)=>action(item,onAction)}

           />
        }}
       />

      </View>
    );
}))

export const UnitIcon = (props =>{
```

```
    const data = props.data

  if(data.type=='group'){
     const units = data.units;
     if(units){
        if(units.length>0){
           const ret = []
           for(var i = 0;i<Math.min(units.length,4);i++){
              const unit = units[i];
              ret.push(<Icon icon={unit.icon}  color={unit.color} style={{alignItems:
'center',justifyContent: 'center',marginTop:2,minWidth:18,width:18,height:16}}
contentStyle={{marginLeft:14}}/>)
           }
           return <View style={{flexDirection:'row'}}>
              {ret}
           </View>
        }
     }
  }
  return   <Icon icon={data.icon}  color={data.color}/>

})

export const InfoContent =  (observer((props) => {

    const city= mainStore.selectedCity;

    const change = ()=>{
       props.navigation.navigate('HeroList', {city:city});
    }

    const changeChancellor = ()=>{
       props.navigation.navigate('SelectChancellor', {city:city});
    }

  const setCapital = ()=>{
       const factionData =city.factionData
       factionData.capital = city
       city.setFactionData({})
       city.setFactionData(factionData)
  }

  return (
    <ScrollView style={{padding:10}}>
       {city?.factionData?.capital?.id == city.id?
          <View style={{flexDirection:'row'}}>
             <Icon icon="star"  />
          <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.capital")}
</Paragraph>
          </View>
       :null}
```

```
      <View style={{flexDirection:'row'}}>
         <Icon icon="bank"  />
        <Paragraph  style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.governor")}
</Paragraph>
        <Caption>{city.governor==undefined?i18n.t("ui.common.none"):city.governor.name}</Ca
ption>
        {props.type!='info'?<Button icon="playlist-edit" onPress={()=>change()} />:null}
      </View>
      {city.buildings.palace?.completedQuantity>0?
      <View style={{flexDirection:'row'}}>
         <Icon icon="bank"  />
        <Paragraph  style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.chancellor")}
</Paragraph>
        <Caption>{city.chancellor==undefined?i18n.t("ui.common.none"):city.chancellor.name}</
Caption>
        {props.type!='info'?<Button icon="playlist-edit" onPress={()=>changeChancellor()}
/>:null}
      </View>
      :null}

      {city.civilization?
        <View style={{flexDirection:'row'}}>
          <Icon icon="pillar"  />
          <Paragraph>{i18n.t("ui.manage.info.civilization")} </Paragraph>
          <Caption >{city.civilization.name} </Caption>
        </View>
      :null}

      <View style={{flexDirection:'row'}}>
        <Icon icon="account-multiple"  />
        <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.population")}
</Paragraph>
        <Caption>{Util.number(city?.population)}</Caption>
      </View>
      {props.type!='info'? <>
        <View style={{flexDirection:'row'}}>
          <Icon icon="barley"  />
          <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.food
consumption")} </Paragraph>
          <Caption> {Util.number(city?.getFoodConsumption())}/{i18n.t("ui.manage.info.day")}
{Util.number(city?.getFoodConsumption()*365)}/{i18n.t("ui.manage.info.year")}</Caption>
        </View>


        <View style={{flexDirection:'row'}}>
          <Icon icon="heart"  />
          <Paragraph style={{position:'relative',top:-2}}>{i18n.t("resource.happiness.name")}
</Paragraph>
          <Caption>{city.happiness.toFixed(2)}/{city.maxHappiness}</Caption>
        </View>
        <View style={{flexDirection:'row'}}>
```

```
            <Icon icon="hospital-box"  />
            <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.hygiene")}
</Paragraph>
            <Caption>{city.getHygiene()}/65</Caption>
          </View>

          <View style={{flexDirection:'row'}}>
            <Icon icon="account-multiple-plus"  />
            <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.population
growth rate")} </Paragraph>
            <Caption>{city.getGrowthRate()}%/{i18n.t("ui.manage.info.year")}</Caption>
          </View>
        </>:null}
        <Divider/>
        <View style={{flexDirection:'row'}}>
          <Icon icon="knife-military"  />
          <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.military units")}
</Paragraph>
          <Caption>{Math.floor(city.getMilitaryUnits())} </Caption>
          <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.armed")}
</Paragraph>
          <Caption>{Math.floor(city.getMilitaryUnits('armed'))} </Caption>
          <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.unarmed")}
</Paragraph>
          <Caption>{Math.floor(city.getMilitaryUnits('unarmed'))} </Caption>

        </View>

        <View style={{flexDirection:'row'}}>
          <Icon icon="chess-rook"  />
          <Paragraph style={{position:'relative',top:-2}}>{i18n.t("ui.manage.info.city defense")}
</Paragraph>
          <Caption> {Math.floor(city.getDefense())}/{city.getDefenseMax()}</Caption>
        </View>
        {city.snapshotSub?.resource?
        <>
          <Divider/>
          <Paragraph>{i18n.t("ui.manage.info.natural resources")} </Paragraph>
          {
            Object.keys(city.snapshotSub.resource).map(key=>{
              const resource = city.snapshotSub.resource[key]
              return <ResourceRow key={key} resource={key}   lv={resource}/>
            })
          }
        </>:null}
        {city?.factionData?.capital?.id == city.id || props.type=='info'?null
        :<Button onPress={()=>setCapital()}>{i18n.t("ui.manage.info.set as capital")}</Button>
        }
        <View style={{height:20}}/>
      </ScrollView>
    );
}))
```

```
export const UnitScreen  = (observer((props) => {

    const unit = props.unit;
    const city= props.type=='group'?unit.city:mainStore.selectedCity;
    const units = props.type=='group'?unit.units.slice():city.units.slice();
    let heroes = props.type=='group'?unit.heroes:city.heroes;
    const editable = props.editable

    if(heroes){
        heroes = heroes.filter(hero => hero.assigned==undefined)
    }else{
        heroes = []
    }

    const build = (unit,onAction)=>{
        props.navigation.navigate('Build', {unit:unit});
        onAction();
    }

    const disband= (unit,onAction)=>{
        unit.disband()
        onAction();
    }

    const assign = (unit,onAction)=>{
        if(unit.type=='farmer'){
            if(city.buildings.farm){
                city.removeUnit(unit);
                city.buildings.farm.addUnit(unit);
            }
        }else{
            props.navigation.navigate('Assign', {unit:unit});
        }

        onAction();
    }

    const assignHero = (hero)=>{
        props.navigation.navigate('Assign', {unit:{type:'hero',data:hero}});

    }

    const equip = (unit,onAction)=>{
        mainStore.pause();
        mainStore.selectedUnit = unit

        props.navigation.navigate('Equip',{editable:editable});

        onAction();
    }
```

```
  const move = (unit,onAction)=>{
    mainStore.move(unit);
    props.navigation.navigate('Home', {unit:unit});
    mainStore.scene.current.rendered = false
    onAction();
  }

  const group = (unit,onAction)=>{
    mainStore.selectedUnit = unit

     props.navigation.navigate('Group',{unit:unit});

     onAction();
  }

  const leave = (u,onAction)=>{
    unit.removeUnit(u);
    u.inGroup= false
    city.addUnit(u);
    unit.checkCapacity()
    onAction();
  }

  const action = (unit,onAction)=>{

    return <>
      {unit.state=='deploy'?
      null
      :<>
        {unit.data.action.manage?<Menu.Item onPress={() => group(unit,onAction)}
title={i18n.t("ui.action.manage")}/>:null}
        {unit.data.action.build&&props.type!='group'?<Menu.Item
onPress={()=>build(unit,onAction)} title={i18n.t("ui.action.build")}/>:null}
        {unit.data.action.assign&&props.type!='group'?<Menu.Item onPress={() =>
assign(unit,onAction)} title={i18n.t("ui.action.assign")}/>:null}
        {unit.data.action.equip?<Menu.Item onPress={() => equip(unit,onAction)}
title={i18n.t("ui.action.inventory")}/>:null}
        {unit.data.action.deploy&&props.type!='group'?
          unit.type!='group'||unit.units.length>0?
            city.happiness<=0&&unit.hasArmy()?
            null
            :<Menu.Item onPress={() => move(unit,onAction)}
title={i18n.t("ui.action.move")}/>
          :null
        :null
        }
        {props.type=='group'&&props.unit.state==''?<Menu.Item onPress={() =>
leave(unit,onAction)} title={i18n.t("ui.action.leave group")}/>:null}

        {props.type!='group'?<Menu.Item onPress={() =>disband(unit,onAction)}
title={i18n.t("ui.action.disband")}/>:null}
      </>
```

```
        }
      </>
    }

    const heroInfo = (hero)=>{

    }

  return (
    <ScrollView style={{paddingTop:10}}>
      <SafeAreaView style={{padding:10}}>
        <View style={{flexDirection:'row',justifyContent:'space-between'}}>
        {props.type=='group'?
          <>
            <View style={{flexDirection:'row'}}>
            </View>
            <View style={{flexDirection:'row'}}>
              {props.unit.state==''?
              <Button dark={true}
style={{marginRight:2}}   onPress={()=>props.navigation.navigate('SelectUnit',
{city:city,unit:unit})} icon="account-plus" contentStyle={{marginLeft:16}} />
              :null}
            </View>
          </>:<>
            <View style={{flexDirection:'row'}}>
              <Icon icon="account" />
              <Paragraph>{i18n.t("ui.manage.unit.available manpower")}
{Util.number(Math.floor(city?.manpower))} /
{Util.number(city?.getMaxManpower())}</Paragraph>
            </View>
            <View style={{flexDirection:'row'}}>
              <Button  style={{marginRight:2}}
onPress={()=>props.navigation.navigate('Employ', {city:city})} icon="account-plus"
mode="contained"  labelStyle={{color:'white'}} contentStyle={{marginLeft:16}} />
            </View>
          </>
        }
        </View>
          <View style={{padding:5}}>
            {props.type=='group'?<>
              <Caption>{i18n.t("ui.manage.unit.units")} {unit.units.length}/10</Caption>
            </>:<Caption>{i18n.t("ui.manage.unit.idle units")}</Caption>
            }
            <FlatGrid
              itemDimension={mainStore.unitSize}
              data={units}

              spacing={1}
              renderItem={({ item }) => (
                <Unit data={item}  action={(onAction)=>action(item,onAction)} />
              )}
            />
```

```
            </View>

            <View style={{padding:5}}>
               {heroes&&heroes.length>0?<>
                  <Caption>{i18n.t("ui.manage.unit.idle heroes")}</Caption>
               </>:null
               }
                <FlatGrid
                    itemDimension={mainStore.unitSize}
                    data={heroes}

                    spacing={1}
                    renderItem={({ item }) => (
                      <Hero data={item} assign={assignHero} navigation={props.navigation}/>
                    )}
                  />

            </View>

      </SafeAreaView>
   </ScrollView>
  );
}))

@observer
export class Resources extends Component {

   constructor(){
      super();

      this.state={
         visible :false,
         amount:1
      }
   }

    changeAmount = ()=>{
         if(this.state.amount ==1){
            this.setState({amount:10})
         }else if(this.state.amount ==10){
            this.setState({amount:100})
         }else{
            this.setState({amount:1})
         }
      }

      transferToUnit = (key)=>{
         const unit = this.props.unit;
          let city = unit.currentLocation;

          if(this.props.target != undefined){
             city = this.props.target;
```

```
        }

        const quantity = Math.min(this.state.amount,        unit.capacity - unit.carrying)
        let consume = city.consumeResource(key,quantity)
        unit.addResource(key, consume);
    }

    transferToCity = (key)=>{
        const unit = this.props.unit;
        let city = unit.currentLocation;

        let condition = city.factionData.id==mainStore.selectedFaction.id;

        const target = this.props.target;
        if(target != undefined){
            city = target;
            condition = true;
        }

        if(condition){

            let quantity = this.state.amount
            if(target != undefined){
                quantity = Math.min(this.state.amount,        target.capacity - target.carrying)
            }
            const consume = unit.consumeResource(key, quantity);
            city.addResource(key, consume);
        }
    }

  render(){

    const unit = this.props.unit;
    let city = unit.currentLocation;
    const deploy = unit.data.action.deploy;

    let condition = city.factionData.id==mainStore.selectedFaction.id && unit.currentRoad ==
undefined;

    if(this.props.target != undefined){
        city = this.props.target;
        condition = true;
    }

    const arr = Object.keys(city.resources).filter(key=> key!='undefined')

    return <>
        {deploy==true?<>
            <View style={{flexDirection:'row',display:'flex',justifyContent:'space-between'}}>
              <Caption>{i18n.t("ui.common.resources")}</Caption>
              {condition?<View style={{flexDirection:'row'}}>
```

```
                <Caption style={{    position: 'relative',top: 5}}>
{i18n.t("ui.common.transfer")}</Caption>
                <Button onPress={()=>this.changeAmount()}>{this.state.amount}</Button>
                <Caption style={{    position: 'relative',top:
5}}>{i18n.t("ui.common.unit")}{this.state.amount>1?i18n.t("ui.common.s"):null}
{i18n.t("ui.common.per click")}</Caption>
              </View>:null}
            </View>
             <View style={{flexDirection:'row',display:'flex',justifyContent:'space-between'}}>
              <Caption></Caption>
              <View style={{flexDirection:'row'}}>
                <Caption > {i18n.t("ui.common.can survive")} {unit.survivableDays}
{i18n.t("ui.common.days")} </Caption>
              </View>
            </View>

            <FlatGrid
              itemDimension={mainStore.unitSize}
              data={Object.keys(unit.resources)}
               scrollEnabled={false}
              spacing={1}
              renderItem={({ item }) => {
                  const key = item;
                  const quantity = unit.resources[key];
                  const resource = mainStore.data.resources[key];

                 return   <Unit data={resource} onPress={()=>this.transferToCity(key)}
                     quantity={
                       ()=>(<Caption>{Util.number(Math.floor(quantity))}</Caption>)
                     }
                  />
               }}
             />
             {condition?<>
               <View style={{flexDirection:'row',display:'flex',justifyContent:'center'}}>
                 <Icon icon="arrow-up"/>
                 <Icon icon="arrow-down"/>
               </View>
               {this.props.target == undefined?
               <Caption>{i18n.t("ui.common.city")} {i18n.t("ui.common.resources")}</Caption>
               :<Caption>{i18n.t("ui.common.target")} {i18n.t("ui.common.resources")}</Caption>
               }
               <FlatGrid
                 itemDimension={mainStore.unitSize}
                 data={arr}

                 spacing={1}
                 renderItem={({ item }) => {
                     const key = item;
                     const quantity = city.resources[key];
                     const resource = mainStore.data.resources[key];
```

```
                return   <Unit data={resource} onPress={()=>this.transferToUnit(key)}
                    quantity={
                      ()=>(<Caption>{Util.number(Math.floor(quantity))}</Caption>)
                    }
                />
              }}

            />
          </>:null}
        </>:null}
      </>
    }
}

@observer
export class Moral extends Component {
   render (){
      const unit = this.props.unit;

      return <View style={{flexDirection:'row'}}>
          <Icon icon="account"  />
          <Caption>{Math.floor(unit.manpower)} </Caption>
          {unit.state!=''?
           <>
          <Icon icon="heart"  />
          <Caption>{unit.moral.toFixed(2)}/100</Caption>

          <Paragraph> {i18n.t("ui.deployed.origin")} </Paragraph>
          <Caption>{unit.city.name}</Caption>
          </>
          :null}
        </View>

    }
}

export class SubUnits {

   @observable units = [];

   getUnit(type,aiType){
      let ret;
      this.units.forEach(unit=>{
         if(unit.type==type){
            if(aiType!=undefined){
               if(unit.aiType == aiType){
                  ret = unit
               }
            }else{
               ret= unit;
            }
         }
```

```
        })
        return ret;
    }

    getMilitaryUnits(armed){

        let ret = 0;
        this.units.forEach(unit=>{
            if(unit.type=='group'){
                ret+=unit.getMilitaryUnits(armed)
            }else{
                if(unit.category=='army'){
                    const damage = unit.getEffect('piercingDamage') + unit.getEffect('slashDamage')
+unit.getEffect('bluntDamage')
                    if(armed=='armed'){
                        if(damage>0)  ret+=unit.men
                    }else if(armed=='unarmed'){
                        if(damage==0)  ret+=unit.men
                    }else{
                        ret+=unit.men
                    }
                }
            }
        })

        return ret;
    }

     @action
    refreshUnits(){
        this.units = this.units.splice(0);
    }
}

export const NearByUnit = (props) =>{

    const unit = props.unit;

    let currentPosition = unit.currentLocation?.name;
    if(unit.currentRoad!=undefined){
        const destinies = unit.currentRoad.road.destinies
        currentPosition = destinies[0].city.name +" - "+destinies[1].city.name
    }

    return <Surface >
            <View style={{flexDirection:'row',justifyContent:'space-between',height:28}}
key={props.index}>
                <View style={{flexDirection:'row',height:50,display:'flex',justifyContent:'center'}}>
                    <Icon  icon={unit.icon}
contentStyle={{position:'relative',top:8,marginLeft:20,height:40}}/>
                    <Paragraph style={{position:'relative',top:7}}>{unit.name}</Paragraph>
                    {unit.resources.food==undefined||unit.resources.food<=0?
```

```
                <Icon  icon="barley" color="#fc8b8b"
contentStyle={{position:'relative',top:8,marginLeft:20,height:40}}/>
                :null}
                {unit.moral<=0?
                <Icon  icon="heart" color="#fc8b8b"
contentStyle={{position:'relative',top:8,marginLeft:20,height:40}}/>
                :null}
                {unit.state=='fighting'?
                <Icon  icon="sword-cross" color="#fc8b8b"
contentStyle={{position:'relative',top:8,marginLeft:20,height:40}}/>
                :null}
            </View>
            <View style={{flexDirection:'row'}}>
                <Button  style={{position:'relative',top:8,minWidth:40,width:40,marginRight:2,paddin
gTop:3}} icon="swap-horizontal" color="grey" onPress={()=>props.onExchange(unit)}/>
            </View>

        </View>
        <View style={{flexDirection:'row',justifyContent:'space-between',height:28,margin:9}} >
            <View style={{flexDirection:'row'}}>
                <Paragraph>{i18n.t("ui.deployed.location")} </Paragraph>
                <Caption >{currentPosition}</Caption>
            </View>
            <View style={{flexDirection:'row'}}>
                <Paragraph>{i18n.t("ui.deployed.origin")} </Paragraph>
                <Caption >{unit.city.name}</Caption>
            </View>
        </View>
    </Surface>
}
```

Додаток Г – Ілюстративна частина

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська дипломна робота на тему:
«Навчальне програмне забезпечення ігрового типу для занурення користувача в історичний контекст через інтерактивні сценарії та симуляції»

Автор: ст. гр. ЗПІ-21б
Кім Т. О.
Науковий керівник: к.т.н.,
доцент ПЗ Чехместрюк Р. Ю.

Вінниця - 2025

Рисунок Г.1 – Титульний слайд

Мета, предмет та об’єкт дослідження

- ✓ **Метою роботи є** підвищення ефективності вивчення історії за рахунок використання спеціалізованого навчального програмного забезпечення ігрового типу, яке дозволяє оптимізувати процес засвоєння історичних знань через інтерактивні сценарії та симуляції.
- ✓ **Об'єкт дослідження є** процес розробки навчального програмного забезпечення ігрового типу для вивчення історії.
- ✓ **Предмет дослідження є** методи та засоби розробки навчального програмного забезпечення ігрового типу з використанням мови JavaScript/TypeScript та фреймворку Phaser.js.

Рисунок Г.2 – Мета, об’єкт та предмет дослідження

Актуальність розробки

✓ **Гейміфікація освітнього процесу:** Сучасні освітні технології потребують інтерактивних методів навчання, що забезпечують активну взаємодію користувача з навчальним матеріалом. Ігрові технології сприяють ефективнішому засвоєнню історичних знань через занурення в контекст та "проживання" історичних подій.

✓ **Використання сучасних веб-технологій:** Застосування JavaScript/TypeScript та фреймворку Phaser.js дозволяє створити адаптивне, історично точне програмне забезпечення з високим рівнем інтерактивності, що забезпечує персоналізований навчальний досвід і розвиток критичного мислення користувачів.

✓ **Актуальність та освітня цінність:** Розробка спеціалізованого навчального програмного забезпечення ігрового типу є необхідною для підвищення ефективності історичної освіти. Впровадження інноваційних ігрових механізмів дозволить формувати історичну компетентність, забезпечувати високу мотивацію до навчання та universal застосування як у формальній освіті, так і для самостійного вивчення історії.

Рисунок Г.3 – Актуальність розробки

Новизна роботи

❖ Розроблено метод модульної архітектури для навчального програмного забезпечення ігрового типу, що поєднує систему відображення історичного контенту, механізми зміни стану об'єктів та управління підрозділами для створення цілісного інтерактивного середовища занурення користувача в історичний контекст.

❖ Запропоновано нову структуру інтеграції програмних модулів освітньої гри історичного спрямування, що характеризується незалежністю компонентів відображення інформації, управління об'єктами та графічного інтерфейсу, забезпечуючи високу адаптивність системи до різних історичних сценаріїв та навчальних завдань.

❖ Створено комплексний підхід до розробки графічного інтерфейсу навчальних ігор історичної тематики, який інтегрує модулі управління підрозділами, відображення обладнання та зміни стану об'єктів в єдину систему, що максимізує ефективність взаємодії користувача з історичним контентом через інтерактивні симуляції.

Рисунок Г.4 – Новизна роботи

Порівняльний аналіз аналогів

Критерій	Civilization VI	Assassin's Creed Discovery Tour	Historia: Game of Claims	Mission US	Власна платформа
Історична достовірність	Середня	Висока	Висока	Висока	Висока
Імерсивність	Висока	Висока	Низька	Середня	Висока
Адаптивність контенту	Низька	Відсутня	Висока	Середня	Висока
Системні вимоги	Високі	Дуже високі	Низькі	Низькі	Низькі
Можливість модифікації	Висока	Відсутня	Відсутня	Відсутня	Висока
Крос-платформність	Windows, macOS	Windows, PlayStation, Xbox	Web	Web	Web, Windows, macOS
Інтеграція з освітніми стандартами	Відсутня	Часткова	Повна	Повна	Повна
Розвиток критичного мислення	Середній	Низький	Високий	Середній	Високий
Інструменти для викладачів	Відсутні	Базові	Розширені	Розширені	Розширені
Аналітика навчальних результатів	Відсутня	Базова	Розширена	Розширена	Розширена
Альтернативні історичні сценарії	Обмежені	Відсутні	Обмежені	Обмежені	Розширені

Рисунок Г.5 – Порівняльний аналіз аналогів

Вибір засобів для реалізації застосунку

Критерій	JavaScript	TypeScript	WebAssembly (C++/Rust)	Dart (Flutter Web)
Продуктивність	★★★★	★★★★	★★★★★	★★★★
Простота розробки	★★★★★	★★★★	★★	★★★
Екосистема для ігор	★★★★★	★★★★★	★★	★★
Розмір bundle	★★★★	★★★	★★★	★★
Час завантаження	★★★★★	★★★★	★★★	★★
Підтримка браузерів	★★★★★	★★★★★	★★★★	★★★
Інструменти розробки	★★★★★	★★★★★	★★★	★★★
Спільнота та документація	★★★★★	★★★★	★★★	★★★
SEO оптимізація	★★★★	★★★★	★★★	★★
Мобільна адаптація	★★★★	★★★★	★★★	★★★★★

Рисунок Г.6 – Вибір засобів для реалізації застосунку

Загальний алгоритм роботи системи

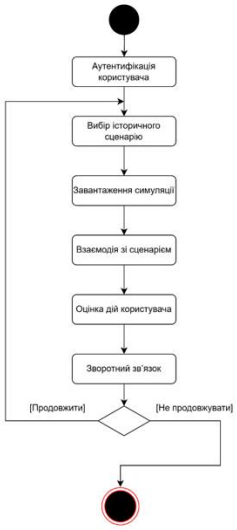

Рисунок Г.7 – Загальний алгоритм системи

Алгоритм управління локацією

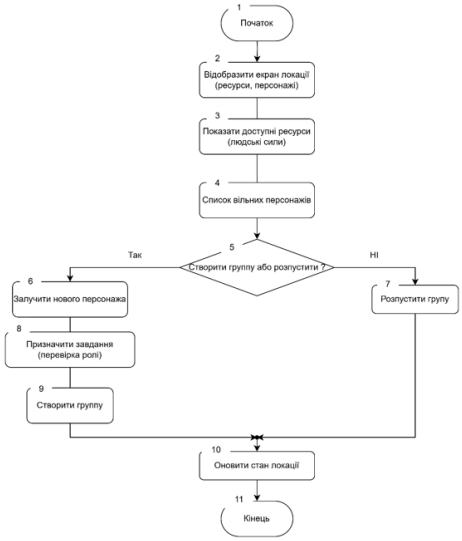

Рисунок Г.8 – Алгоритм управління локацією

Алгоритм управління командою

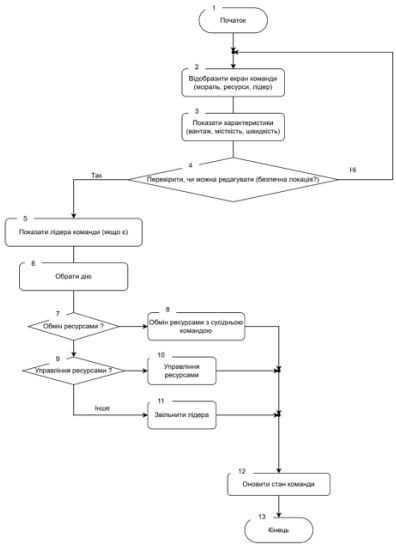

Рисунок Г.9 – Алгоритм управління командою

Використані технології

JavaScript – мова розробки

Node js – програмна платформа

React – бібліотека Java Script

TypeScript – мова розробки

TypeScript – ігровий фреймворк

Рисунок Г.10 – Використані технології

Демонстрації інтерфейсу застосунку
«Початковий екран користувача»

Рисунок Г.11 – Демонстрації інтерфейсу застосунку «Початковий екран користувача»

Демонстрації інтерфейсу застосунку
«Елементи стратегічного ігрового процесу»

Рисунок Г.12 – Демонстрації інтерфейсу застосунку «Елементи стратегічного ігрового процесу»

Демонстрації інтерфейсу застосунку «Наративний інтерфейс»

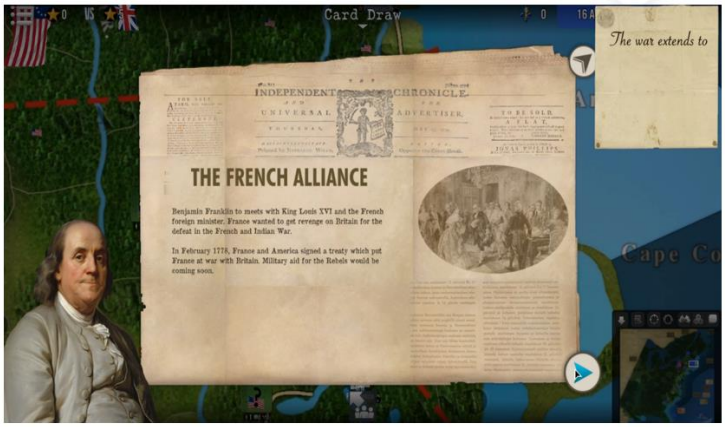

Рисунок Г.13 – Демонстрації інтерфейсу застосунку «Наративний інтерфейс»

Результат розробки

- розроблено модуль історичного контенту для представлення достовірної інформації про різні історичні періоди та події;
- розроблено модуль інтерактивних сценаріїв для реалізації альтернативних сюжетних ліній та симуляції історичних ситуацій;
- розроблено модуль користувацького інтерфейсу для забезпечення інтуїтивного керування та візуальної привабливості програми;
- розроблено модуль оцінювання знань для відстеження навчального прогресу та адаптації контенту;
- розроблено модуль аудіовізуального супроводу для посилення занурення в історичний контекст;
- інтеграція різних модулів в єдину програмну систему.

Рисунок Г.14 – Результати розробки

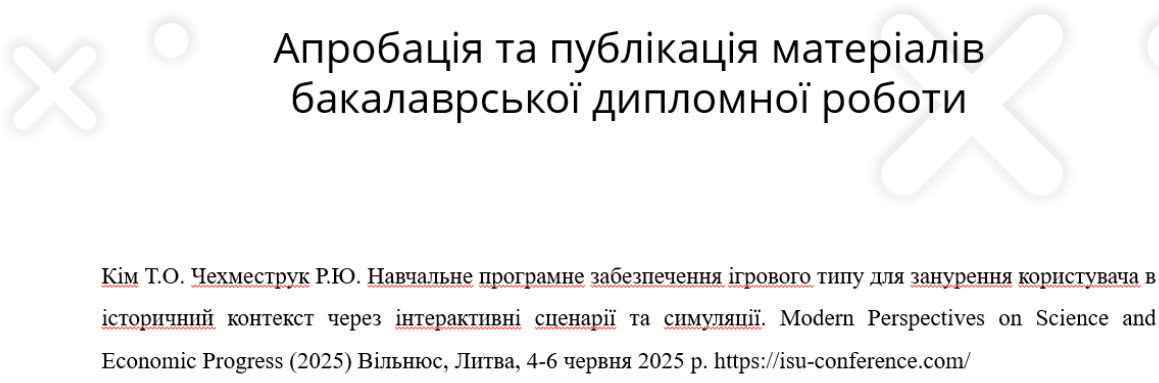

Рисунок Г.15 –Апробація та публікація матеріалів бакалаврської кваліфікаційної роботи

Рисунок Г.16 – Фінальний слайд