Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Ігровий програмний застосунок в жанрі колекційної карткової гри
(тема)

Виконав:
студент 4 курсу, групи ПЗПІп-22-2

Ницик А. М.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст. викл. кафедри ПІ Олійник О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ницику Андрію Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Ігровий програмний застосунок в жанрі колекційної карткової гри

Затверджена наказом по університету від 17.06. 2024р. № 588 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.07.2024

3. Вихідні дані до роботи Розробити ігровий програмний застосунок в жанрі колекційної карткової гри, що є цифровою демонстраційною версією колекційної карткової гри «Світ Заару», використовуючи ігровий рушій Unreal Engine.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.02.2024	*виконано*
2	Створення специфікації ПЗ	22.03.2024	*виконано*
3	Проектування ПЗ	24.04.2024	*виконано*
4	Розробка ПЗ	28.05.2024	*виконано*
5	Тестування ПЗ	30.06.2024	*виконано*
6	Оформлення пояснювальної записки	05.07.2024	*виконано*
7	Підготовка презентації та доповіді	06.07.2024	*виконано*
8	Попередній захист	16.07.2024	*виконано*
9	Нормоконтроль, рецензування	17.07.2024	*виконано*
10	Здача роботи у електронний архів	18.07.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.07.2024	*виконано*

Дата видачі завдання 18 червня 2024р.

Студент (ка)_____ Ницик А. М.
(підпис)

Керівник роботи _____ ст. викл. кафедри ПІ Олійник О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Звіт з передатестаційної практики: 96 с., 53 рис., 4 таблиці, 12 джерел.

АЛГОРИТМ, КОЛЕКЦІЙНА КАРТКОВА ГРА, BLUEPRINT, UNREAL ENGINE, UML, CCG TOOLKIT

Об’єктом роботи є проєктування та розробка демонстраційної версії цифрової колекційної карткової гри «Світ Заару».

Метою розробки є формалізація правил колекційної карткової гри «Світ Заару», створення архітектури застосунку та відповідної документації, реалізації на їх основі цифрової версії гри.

Методи рішення спираються на кодову базу ігрового рушію Unreal Engine 5 та набір інструментів для створенні цифрових колекційних карткових ігор «CCG Toolkit».

В результаті роботи була створена демонстраційна версія цифрової колекційної карткової гри «Світ Заару». Вона реалізовує базові механіки та правила оригінальної гри, дає можливість двокористувацької гри на одному ПК. Включає в себе колоду з 30 унікальних карт, 20 з яких мають здібності. Гравець грає гральною колодою, що випадковим чином формується із зазначених раніше 30 карт.

ALGORITHM, COLLECTIBLE CARD GANE, BLUEPRINT, UNREAL ENGINE, UML, CCG TOOLKIT

The object of the work is the design and the development of a demo version of the digital collectible card game “World of Zaar”.

The method of development is the formalization of the rules of the collectible card game “World of Zaar”, the creation of the architecture of the game and the related documentation, implementation of the digital version of the game on their basis.

The solution methods are based on the code base of the game engine Unreal Engine 5 and a set of tools for creating digital collectible card games “CCG Toolkit”.

The demo version of the digital collectible card game “World of Zaar” was created as the result of the work. It implements the basic mechanics and rules of the original game, giving the ability to play the game on single PC for two players. It includes a deck of 30 unique cards, 20 of which have special abilities. Player plays with a deck, which is formed randomly from mentioned 30 cards.

Я, Ницик Андрій Михайлович, студент гр. ПЗПІп-22-2, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок в жанрі колекційної карткової гри», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ККГ – колекційна карткова гра (ігри)

ЦККГ – цифрова колекційна карткова гра (ігри)

ПК – персональний комп’ютер

ARPG – action role-playing game, піджанр рольових відеоігор, в якому важливу частину займають елементи жанру бойовик

ОЗ – очки здоров’я

ОЕ – очки енергії

ОЕА – очки енергії для активації

ОБ – очки броні

ОНВ – очки негативного впливу

ОПС – очки потаємної сили

ОП – очки пошкоджень

ОНП – очки нанесених пошкоджень

GUI – graphical user interface, графічний інтерфейс користувача

ВСТУП

Темою кваліфікаційної роботи є створення ігрового програмного застосунку в жанрі колекційні карткові ігри. У якості основи були вибрані правила існуючої настільної колекційної карткової гри «Світ Заару».

Колекційні карткові ігри (Collectible card game) [1, 2, 3] – це настільні ігри, які передбачають використання спеціально розроблених гральних карт. Кожна картка містить унікальні атрибути, здібності та зображення. Мета ККГ – створити колоду карт, яка зможе перемогти супротивників. У ККГ зазвичай беруть участь двоє або більше гравців, які по черзі грають картами та стратегічно використовують свої ресурси, щоб отримати перевагу.

Однією з важливих характеристик ККГ є те, що самі по собі карти мають цінність, а не лише ігровий процес. Деякі карти вважаються рідкісними або цінними через обмежену доступність, унікальні здібності або як культові твори мистецтва.

Гра «Світ Заару» є представником ККГ, в той же час вона містить додаткові цікаві механіки та правила, що можуть зацікавити як новачків так і затятих поціновувачів жанру.

Метою роботи є розробка програмної системи, що реалізує демонстраційну версію настільної колекційної карткової гри «Світ Заару».

Цільова платформа реалізації – MS Windows 10, 11. Двокористувацька гра відбувається на одному ПК. Для розробки продукту використовується ігровий рушій Unreal Engine 5 [4], який пропонує однойменне середовище розробки та мову візуального програмування Blueprint [5].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

ККГ – це різновид карткової гри, що поєднує стратегічні елементи побудови колоди з функціями колекційних карток. Першим представником жанру є гра «Magic: The Gathering», представлена у 1993 році.

Як правило, гравець починає грати в ККГ із попередньо створеною стартовою (гральною) колодою та облаштовує її за допомогою карт, які він збирає зі звичайних наборів, пакетів підсилення або в результаті обміну з іншими гравцями. Що більше карт – то різноманітніша гральна колода може вийти. Гравці вибирають, які карти додати до своїх колод на основі певної стратегії, залишаючись при цьому в межах набору правил. В ККГ зазвичай грають удвох, хоча формати для кількох гравців також поширені. Ігровий процес у ККГ зазвичай покроковий, коли кожен гравець починає з перетасованої колоди, а потім у свій хід бере та розігрує карти, щоб атакувати іншого гравця та зменшити очки його здоров’я до нуля, перш ніж противник зробить те саме з ним. Гральні кубики, фішки ігрові килимки використовуються для доповнення ігрового процесу. Існують турніри ККГ, де гравці змагаються за призи.

ККГ зазвичай охоплюють жанри фентезі чи наукової фантастики, а також включають теми жахів, мультфільмів і спорту, а також можуть містити ліцензовану інтелектуальну власність. Карти в ККГ – це спеціально розроблені набори гральних карт. Дизайн та властивості кожної карти в ККГ розробляються таким чином, щоб максимально відповідати тематиці самої гри та сприяти ігроладові. Кожна карта може належати до таких категорій, як істоти, вдосконалення, події, ресурси, локації і т. п. Усі картки в ККГ зазвичай мають одне й те саме зображення зворотного боку, тоді як лицьова сторона має комбінацію власних ілюстрацій, щоб прикрасити картку, а також інструкцій її використання. Набори розширень використовуються для розширення ККГ, представлення нових стратегій ігроладу, тощо. Успішні ККГ зазвичай мають тисячі унікальних карт через кілька розширень. «Magic: The

Gathering» спочатку було запущено з 300 унікальними картками, а станом на березень 2020 року їх налічувалося вже понад 22000.

Останнім часом завдяки успіху онлайн-версій ККГ, таких як «Magic: The Gathering Online», і повністю цифрових ККГ, таких як «Hearthstone», набули популярності ЦККГ.

Кожна ККГ має фундаментальний набір правил, який описує цілі гравців, категорії карт, які використовуються в грі, і основні правила, за якими карти взаємодіють. Кожна карта матиме додатковий текст, який пояснює вплив цієї конкретної карти на гру. У багатьох іграх використовуються набір ключових слів для спрощення опису карти, при цьому ключові слова стосуються загальних правил гри. Кожна карта також зазвичай представляє певний елемент, похідний від жанру гри, обстановки або вихідного матеріалу. Карти зображені та названі відповідно до цих вихідних елементів, і їхня ігрова функція може бути пов’язана з цими елементами. Наприклад, «Magic: The Gathering» засновано на жанрі фентезі, тому багато карт представляють істот та магічні заклинання цього жанру.

Більшість ККГ розроблено навколо системи ресурсів, за допомогою якої контролюється темп кожної гри. Часто карти, які складають гральну колоду гравця, вважаються ресурсом, і перехід карти з колоди в ігрову зону або руку суворо контролюється правилами. Відносна сила карти не рідко врівноважується кількістю або типом ресурсів, необхідних для того, щоб зіграти цією картою. Ресурсами можуть бути представлені або самими картами, або іншими засобами.

На відміну від традиційних карткових ігор, таких як покер, де вміст колоди обмежений і визначений заздалегідь, гравці ККГ вибирають, які карти складатимуть їхню ігрову колоду з будь-яких доступних карт, надрукованих для гри. Це дозволяє гравцям стратегічно облаштовувати свою колоду, щоб задля перемоги зосередитися на певних аспектах та механіках гри.

1.2 Виявлення та вирішення проблем

Для більш детального аналізу предметної області варто окремо розглянути яскравих представників ЦККГ та виокремити їхні спільні та відмінні риси.

Для аналізу було вибрано наступні популярні ігри:

- «Gwent: The Witcher Card Game» [6];
- «Magic: The Gathering Online» [7];
- «Genius Invokation TCG» [8];

В таблиці 1.1 вкажемо деякі важливі порівняльні характеристики ігор, після чого коротко розглянемо їхні особливості.

Таблиця 1.1 – Порівняльні характеристик ігор (Таблиця виконана самостійно)

Показник	«Gwent: The Witcher Card Game»	«Magic: The Gathering Online»	«Genius Invokation TCG»
Кількість фаз	2	5	3
Містить платний контент	+	+	+
Кількість гравців у партії	2	2-4	2
Кількість карт на руках в гравця для проведення партії	>25	>60	33
Кількість числових характеристик	1	2	2
Кількість типів карт	3	8	2

Гра «Gwent: The Witcher Card Game». На рисунку 1.1 представлений її користувацький інтерфейс.

Рисунок 1.1 – Користувацький інтерфейс гри «Gwent: The Witcher Card Game» ([6])

Гра «Gwent: The Witcher Card Game» зроблена на основі серії популярних ARPG ігор «The Witcher». Вона відображає зіткнення армій на полях битв у всесвіті «Відьмака», де гравці є командирами, а карти – їхніми загонами.

Гравці грають попередньо зібраними колодами однієї з шести фракцій, які охоплюють не менше 25 карт. Кожна фракція має різних «лідерів», кожен з яких володіє індивідуальними здібностями. Фракційний лідер карткою не є: він представлений на ігровому полі у вигляді тривимірної анімованої фігурки.

Гравці ходять по черзі. Черговість першого ходу визначається жеребом. Протягом свого ходу гравець може викласти на ігрове поле карту загону або артефакт, віддати накази загонам або лідеру, а також спасувати. Мета гри: комбінуючи наявні карти в потрібній послідовності, посилюючи свої загони та послаблюючи загони суперника, домогтися переваги за очками й перемогти у двох раундах. Якщо гра не завершується за два раунди, додається третій – фінальний. Пас з боку обох гравців – це кінець раунду або матчу. У матчі виграє той гравець, чия сила загонів на ігровому полі до кінця раунду перевищує силу загонів противника.

Карти поділяються на персонажів, істот або події. Карти відрізняються один від одного на підставі різних властивостей:

– колір і рідкісність – впливають на силу карти, на кількість споживаної нею провізії;

– фракційна приналежність – гравець може виставляти колоди, які стосуються шести фракцій, що мають ряд унікальних механік, які безпосередньо впливають на ігролад: «Королівства Півночі», «Нільфґаард», «Скоя'таелі», «Скелліґе», «Монстри» і «Синдикат»;

– тип – загони, заклинання, особливі й погодні карти, а також артефакти. [6, 9]

Гра «Magic: The Gathering Online». На рисунку 1.2 представлений її користувацький інтерфейс.

Рисунок 1.2 – Користувацький інтерфейс гри «Magic: The Gathering Online» ([7])

Гра «Magic: The Gathering Online» це цифрова реалізація популярної настільної ККГ «Magic: The Gathering». Партія гра має вигляд боротьби між чаклунами, що звуться «Світоходці». Вони використовують закляття, артефакти, істот з різних карток колоди для перемоги супротивників. Оригінальна ідея гри

походить від традиції фентезійних рольових ігор на кшталт «Dungeons & Dragons», як наслідок гра має в арсеналі набагато більше карток і складніші правила, ніж решта настільних ККГ.

Одночасно в партії гри можуть брати участь від 2 до 4 гравців. Гра має 5 фаз, що циклічно повторюються. На кожній із фаз гравець може виконувати певні регламентовані правилам дії. На початку партії кожен гравець має по 20 очок життя та 60 карт у гральній колоді. Партія завершується, коли у грі залишається один гравець з кількістю очок життя більше нуля.

Карти поділяються на наступні категорії:

– землі – виробляють магічну енергію певного типу для використання чар;

– чари – вимагають вказаної кількості магічної енергії та типу на те, щоб гравець міг їх застосувати.

У свою чергу чари поділяються на наступні типи:

– істоти – створіння, що можуть атакувати та захищатися; деякі наділені додатковими характеристиками, що роблять їх сильнішими чи вразливішими за певних умов;

– закляття та миттєвості – перші впливають на інші карти у відповідну фазу, другі мають ефект у мить розкриття, потім скидаються;

– зачарування та артефакти – залишаються у грі на вказаний термін та мають ефект на одну чи кілька карток;

– світоходці – позначають могутніх героїв з визначеною кількістю спеціальних здібностей, що оцінюються в жетонах відданості своєму гравцю. Якщо світоходець зазнає атак, відданість зменшується, але це не усуває його з гри, а змінює доступні можливості.

Магічна енергія має 5 типів зі своїми особливими властивостями. [7, 10]

Гра «Genius Invokation TCG». На рисунку 1.3 представлений її користувацький інтерфейс.

Рисунок 1.3 – Користувацький інтерфейс гри «Genius Invokation TCG» ([8])

Гра «Genius Invokation TCG» є мінігрою всередині ARPG «Genshin Impact». Максимальна кількість гравців – два. Кожен гравець починає партію з колодою в 33 карт, де 3 карти представляють собою персонажів з оригінальної гри, а решта – карти дій. Перемагає той гравець, в якого на полі залишився хоча б один персонаж із кількістю очок здоров'я більше за 0.

Гра включає 3 фази, які циклічно повторюється. Лише одна з них вимагає від гравця активних дій.

Як раніше було згадано, карти поділяються на 2 групи: персонажів та дій.

Група персонажів можуть мати наступні властивості:

– очки здоров'я; якщо число опускається до нуля, то карта скидається з поля;

– нормальна атака;

– стихійне вміння;

– стихійний вибух;

– пасивне вміння;

Нормальна атака, стихійне вміння та стихійний вибух для використання вимагають очок стихій, які гравець отримує на першій фазі випадковим чином. Всього гра налічує 7 стихій, атака з використанням певної стихії може залишати на атакованому персонажі певний ефект, послідовний напад на того ж персонажа з використанням іншої стихії може посилити, послабити чи перетворити ефект.

Карти дій діляться на 3 типи:

– екіпірування – використовуються на картах персонажів для підсилення;

– подія – специфічний ефект розігрується миттєво після активації;

– підтримка – розміщується в спеціальній зоні для постійного застосування допоміжного ефекту. [8]

Підсумовуючи огляд існуючих ЦККГ варто зауважити, що всі вони є комерційними проектами, які або мають платний контент, або популяризують базову гру, на основі якої були зроблені.

Гра «Magic: The Gathering Online» має найдовшу історію розвитку серед перелічених ігор. Як видно з кількості типів карт, кількості фаз, кількості карт на руках гравця, ця гра є найскладнішою та одночасно найрізноманітнішою з перелічених. Решта ігор простіші, відповідно поріг входження в них нижчий. З цього випливає, що для новачків простіше освоїти «Gwent: The Witcher Card Game» та «Genius Invokation TCG». У той же час більш просунуті гравці з більшою ймовірністю за решти однакових умов виберуть саме «Magic: The Gathering Online» через більшу варіативність гри.

Необхідно зауважити також, що ігри «Gwent: The Witcher Card Game», «Genius Invokation TCG» є «побічними» іграми, що завдячують своєю популярністю проектам, на базі яких були зроблені. З одного боку це перевага, оскільки вони вже мають певну базову аудиторію без додаткових витрат на рекламу. З іншого боку через такий підхід подібні ігри зазвичай простіші за ті, які розробляються з нуля з метою створити саме конкретну гру. Тут варто наголосити, що аби пограти в гру «Genius Invokation TCG» необхідно грати в «Genshin Impact» і дійти до певного рівня розвитку персонажа.

У гру «Magic: The Gathering Online» можна грати компанією від 2 до 4 гравців, у той час, як в решті ігор максимум удвох. Це явна перевага «Magic: The Gathering».

Тепер розглянемо оригінальну настільну версію ККГ «Світ Заару». Її правила та короткий опис наведені в Додатку Г.

«Світ Заару» – фентезійна ККГ, де гравці є воєводами, що командують своїми арміями та накладають різноманітні впливи за допомогою спеціальних заклять. Ціль партії – залишитися єдиним воєводою у грі з ОЗ більшими за 0.

«Світ Заару» дозволяє грати 2 – 4 гравцям одночасно. Кожен раунд гри включає 3 фази, у 2-х з них гравці беруть активну участь. Гра має 5 числових характеристик, які впливають на різні аспекти гри, а відповідно на силу гравців та тактику їхніх дій. Залежно від кількості гравців у партії, кожен з них отримує по 15, 30, 50 карт у гральну колоду.

Карти дій діляться на 2 типи:

– оточення – обов’язково мають здібності, що активуються одразу ж після введення карти у гру, не мають ОЗ, але обов’язково мають ОЕА, автоматично скидаються після завершення раунду;

– воїни – можуть мати або не мати додаткових здібностей, основними їхніми характеристиками є ОЗ, ОЕА, ОП, ОБ, здатні атакувати як суперників, так і їхні армії на відповідній фазі гри;

«Світ Заару» знаходиться посередині між «Magic: The Gathering», «Gwent: The Witcher Card Game», «Genius Invokation TCG» за кількома аспектами. За кількістю правил вона простіша за «Magic: The Gathering», але складніша за «Gwent: The Witcher Card Game», «Genius Invokation TCG». Вона дає можливість грати більше ніж двом гравцям. Є безкоштовною, не має платного контенту та особливих умов, за яких можна в неї пограти (наприклад, як «Genius Invokation TCG»). Єдина потенційна проблема гри – кількість числових характеристик, що більша за кількість числових характеристик в інших розглянутих ККГ. Це явно ускладнює гру, адже зазвичай гравець має особливо слідкувати за власними ОЗ або

ОЕ (чи аналогічними характеристиками в інших іграх). Але з іншого боку при вдалому ігроладі ця особливість зробить гру цікавішою та варіативнішою.

Підсумовуючи все раніше сказане, варто зазначити, що ККГ «Світу Заару» є конкурентоздатною грою, що може зацікавити як новачків, так і поціновувачів жанру як через гарну варіативність гри, так і через відносно невисокий поріг входження.

1.3 Постановка задачі

У ході виконання роботи необхідно реалізувати цифрову демонстраційну версію ККГ «Світу Заару». Вона має програмно реалізовувати правила та механіки гри (див Додаток Г). До них відносяться:

– вигляд та функціонування ігрових полів гравця та його суперників;

– підготовчий етап гри з формуванням ігрових колод та визначенням послідовності ходу;

– циклічне повторення 3-х фаз гри до моменту її завершення;

– розрахунок та порядок використання числових характеристик гравців та карт;

– механізм функціонування унікальних здібностей карт;

– забезпечення виконання дій гравця відповідно до його можливостей та фази гри.

Особливу увагу необхідно приділити архітектурі застосунку та формалізації текстового опису правил.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Метою виконання роботи є створення програмного застосунку, що є демонстраційною версією ЦККГ «Світу Заару», якого достатньо для розуміння базових принципів та механік гри загалом. Таке рішення було прийняте з огляду на значний обсяг оригінальної гри, що включає в себе:

- можливість одночасної гри більш ніж 2-м гравцям;
- більш ніж 200 унікальних карт;
- більш ніж 70 унікальних здібностей;
- можливість отримувати, обмінювати, збирати карти;
- можливість збирати власну ігрову колоду.

Демонстраційна версія обмежується наступною функціональністю:

- відсутність можливості колекціонування, збирання карт;
- базові механіки та правила (див. 1.3 Постановка задачі);
- двокористувацька гру на одному ПК;
- 30 унікальних карт;
- 20 унікальних здібностей;
- гра випадковою гральною колодою.

У якості середовища розробки та ігрового рушія була вибрана Unreal Engine 5. Ігровий застосунок повинен працювати на ОС MS Windows 10/ MS Windows 11.

2.2 Загальний опис

Програмну реалізацію можна розділити на візуальну та функціональну частини. Візуальна частина повинна максимально відповідати вигляду ігрового поля та карток ККГ «Світу Заару» (див. Додаток Г).

Відповідно обмеженню демонстраційної версії до можливості винятково двокористувацької гри, зменшимо ігрове поле до необхідних розміру та кількості

елементів. На рисунку 2.1 схематично представимо загальний вигляд ігрового поля користувача для цифрової версії гри.

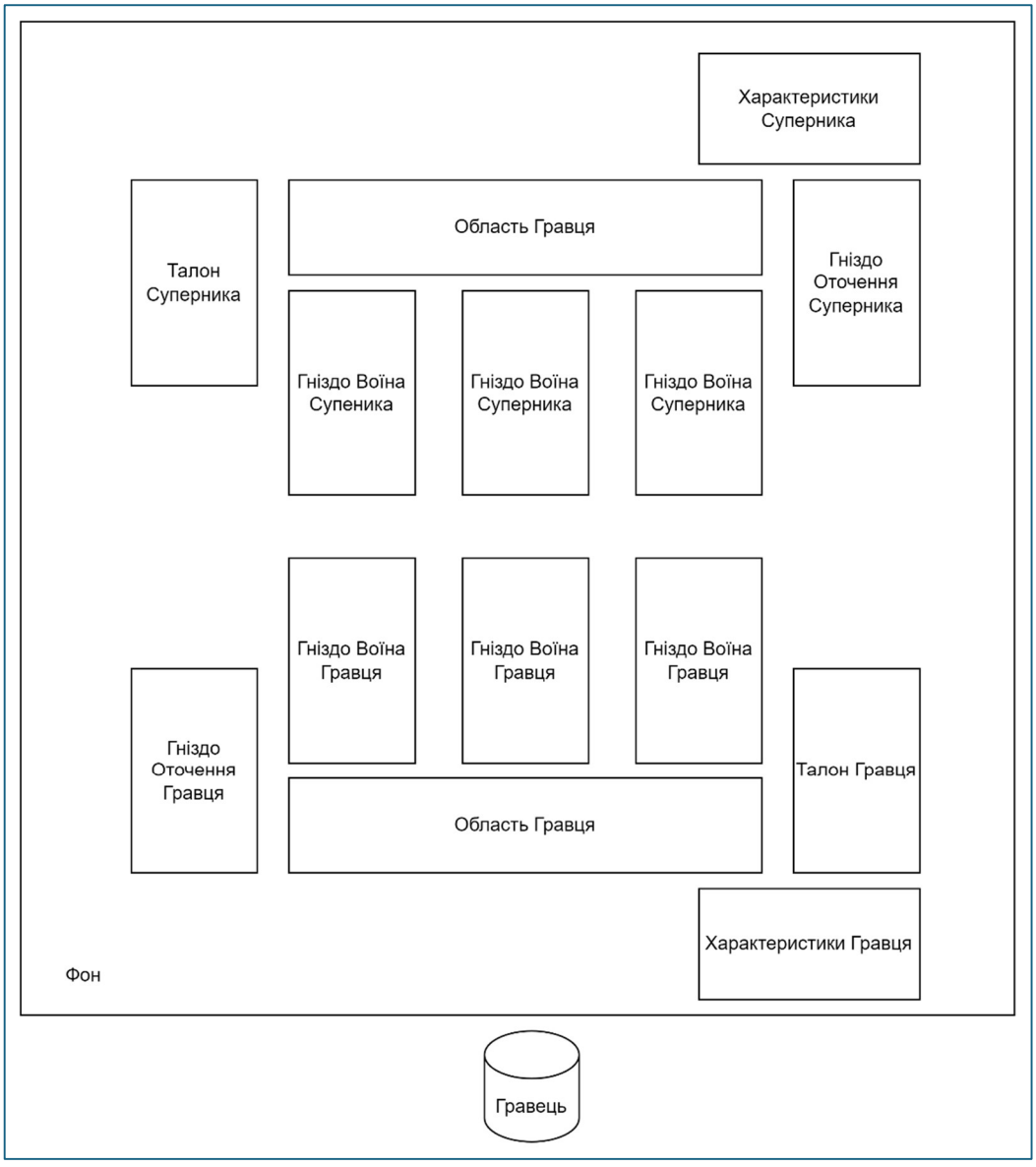


Рисунок 2.1 – Загальний вигляд ігрового поля користувача для цифрової версії гри (Рисунок виконаний самостійно)

На рисунках 2.2, 2.3 схематично представимо загальні вигляди лицьового боку карт «Оточення» та «Воїна».

Рисунок 2.2 – Загальні вигляди лицьового боку карт «Оточення» для цифрової версії гри (Рисунок виконаний самостійно)

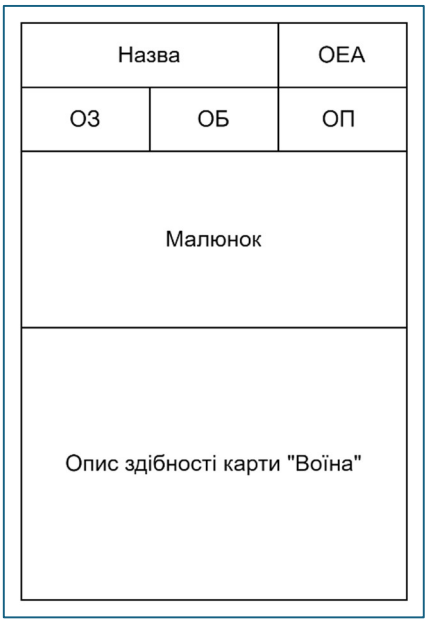


Рисунок 2.3 – Загальні вигляди лицьового боку карт «Воїна» для цифрової версії гри (Рисунок виконаний самостійно)

Анімація візуальних компонент гри повинна бути схожою на поведінку реальних об’єктів – елементів ККГ «Світу Заару».

Функціональна частина гри повинна реалізовувати об’єм правил та механік, визначений у попередньому підрозділі.

2.3 Обмеження, припущення та залежності

З огляду на використання рушія Unreal Engine 5:

- додаток може не запускатися або працювати не коректно на версіях Unreal Engine відмінних від 5-ї;
- у якості системи збереження та захисту даних був вибраний вбудований в Unreal Engine 5 тип DataTable. Тож саме Unreal Engine 5 відповідає за безпеку та цілісність даних;
- у якості малюнків дна лицьових сторонах карт будуть вибрані узагальнені однотипні зображення, не захищені авторським правом.

Також необхідно вказати на можливість використання готових додаткових інструментальних бібліотек Unreal Engine 5, що допоможуть в реалізації ЦККГ.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проєктування графічної частини застосунку

Загалом ЦККГ повинна мати мінімум 2 GUI – окремо для кожного з двох аспектів гри:

- колекціонування та взаємодія з картами;
- проведення змагання.

Оскільки демонстраційна версія ЦККГ «Світ Заару» обмежується винятково проведенням змагання, то все подальше проектування та розробка не поширюватимуться на зазначений перший аспекти ККГ.

Схематичний вигляд ігрового поля користувача, представлений на рисунку 2.1, береться за основу для проектування GUI. Ігрове поле не включає в себе додаткові елементи керування, які необхідні гравцеві для взаємодії з грою згідно правилам у цифровій версії. Такими елементами є:

- кнопка завершення ходу;
- «рука» з картами для вибору та активації;
- кількість карт, які залишилося в колоді гравці.

Карти з ігрової колоди гравця не повинні обов’язково відображатися на GUI оскільки вони випадковим чином додаються в руку гравця у фазі «Завершення Кола».

Також важливою фактором GUI є активність елементів, з якими може або не може взаємодіяти гравець. Цей фактор окрім іншого залежить від фази гри. Відповідно для кожної з фаз гри карта активних елементів відрізнятиметься. На фазі «Завершення Кола» винятково проводяться розрахунки, тож гравець не взаємодіє з жодним елементом GUI. Відповідно, необхідно спроектувати вигляд GUI з активними елементами для фаз «Активація Карт» та «Битва Армій» окремо. На основі правил гри та проведеного аналізу зобразимо GUI з активними елементами для відповідних фаз гри на рисунках 3.1 та 3.2.

Необхідно звернути увагу, що зеленим кольором виділені ті елементи GUI, з якими може взаємодіяти гравець.

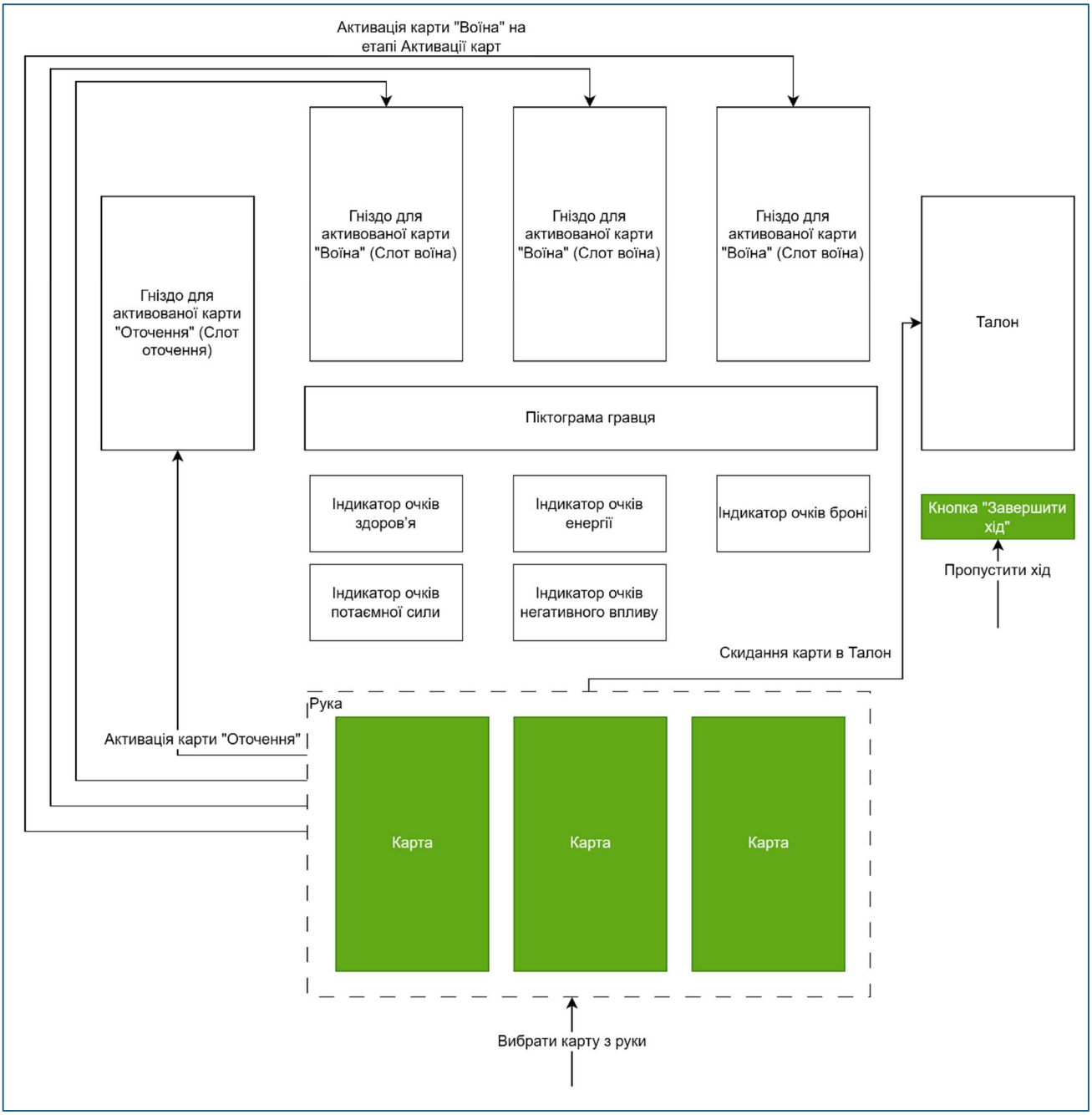


Рисунок 3.1 – Схематичний вигляд GUI з активними елементами для фази «Активація Карт» (Рисунок виконаний самостійно)

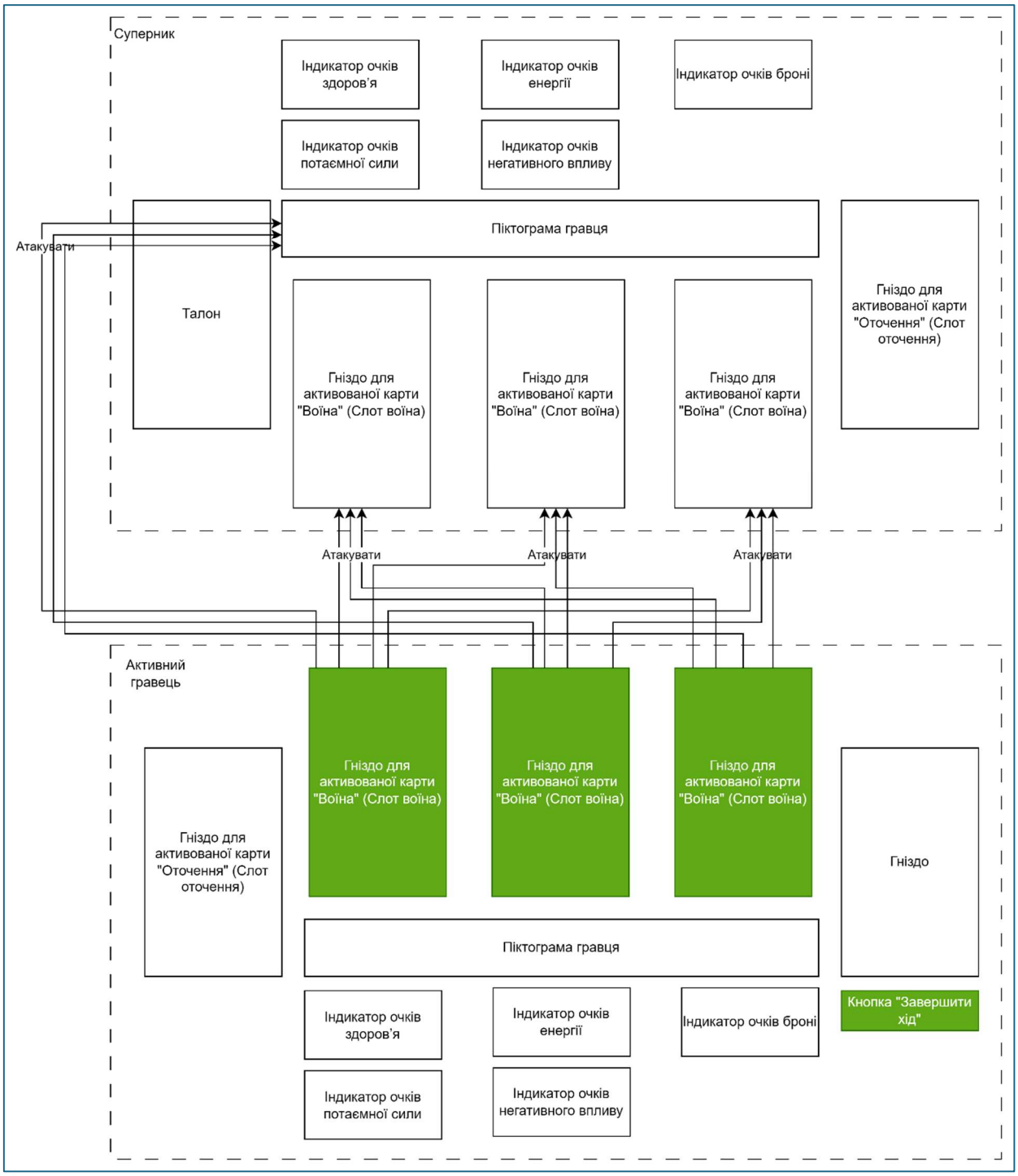


Рисунок 3.2 – Схематичний вигляд GUI з активними елементами для фази «Битва Армій» (Рисунок виконаний самостійно)

Вигляд лицьової частини карт, представлений на рисунках 3.2, 3.3 підходить для GUI та не вимагає якоїсь переробки, оскільки він несе лише інформаційне і не має якихось активних елементів.

3.2 Проектування логічної частини застосунку

На рисунку 3.3 зобразимо UML діаграму використання [11] та визначимо, як гравці використовують застосунок згідно правилам гри та активним елементам GUI, що було описано в попередньому розділі.

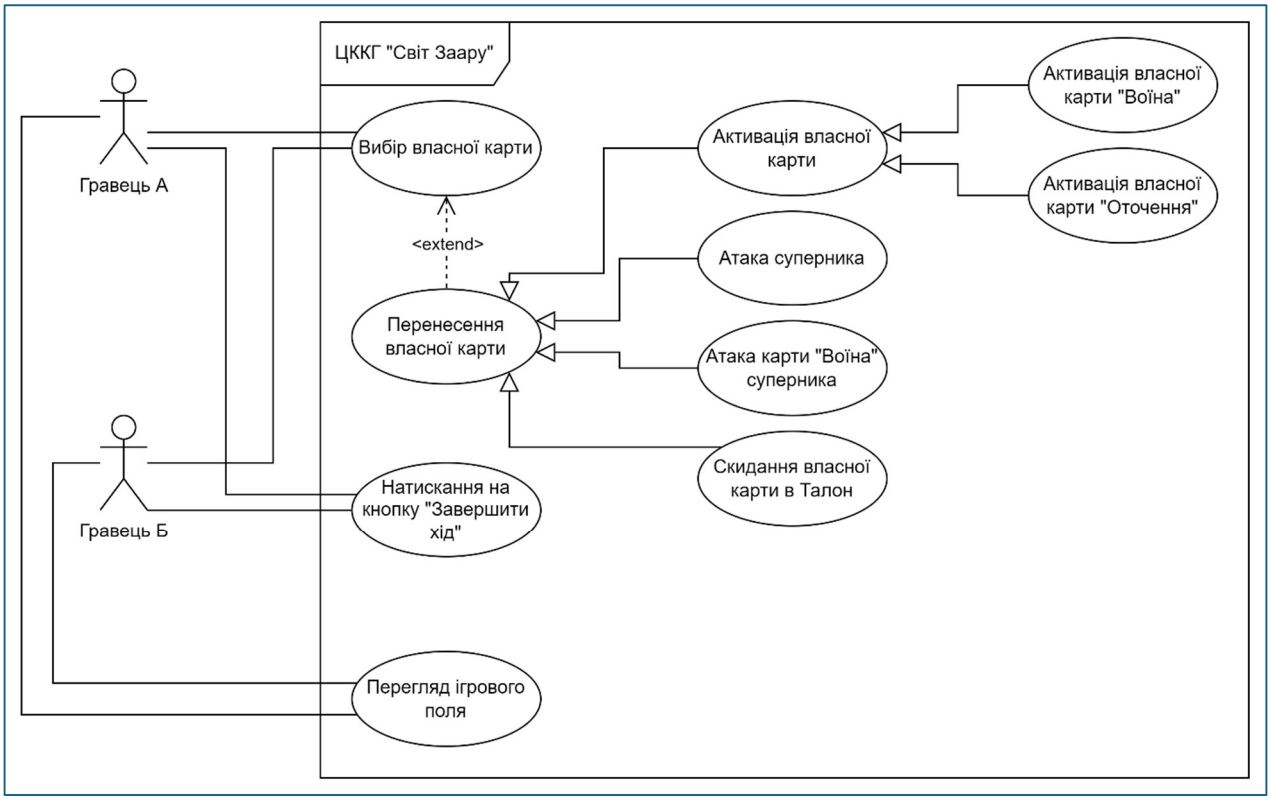


Рисунок 3.3 – UML діаграма використання (Рисунок виконаний самостійно)

Як видно з рисунку 3.3 логічна частин застосунку ЦККГ «Світ Заару» не розрізняє поняття «гравець» та «суперник». Ці поняття важливі винятково для графічної частини, яка розставляє акценти на протиборчих сторонах. Логічна частина уніфікована до «власних» та «інших» сутностей, що циклічно змінюються в процесі гри.

Гравець своїми активними діями може:

– вибирати власні карти з можливістю їх перенесення;

– натискати на кнопку «Завершити хід».

У якості зворотного зв’язку логічна частина застосунку оновлює ігрове поле.

Представимо правила гри у формі UML діаграм станів окремо для кожної фази та в згрупованому вигляді разом, користуючись введеними раніше сутностями та поняттями.

Діаграм станів була вибрана через те, що така нотація детально описує скінченний автомат, який добре підходить для використання в системах зі складною логікою.

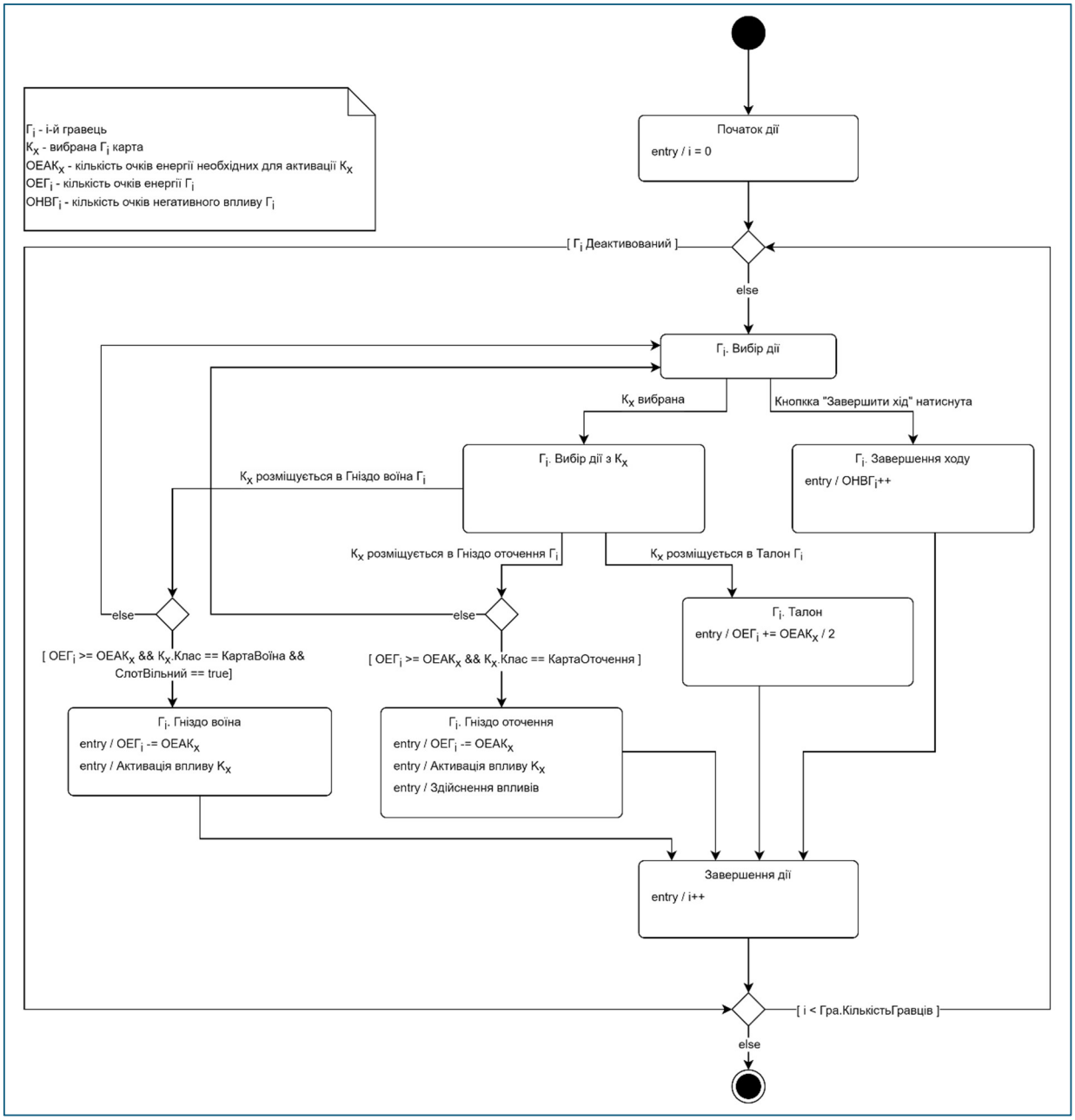


Рисунок 3.4 – UML діаграм станів для фази «Активація Карт» (Рисунок виконаний самостійно)

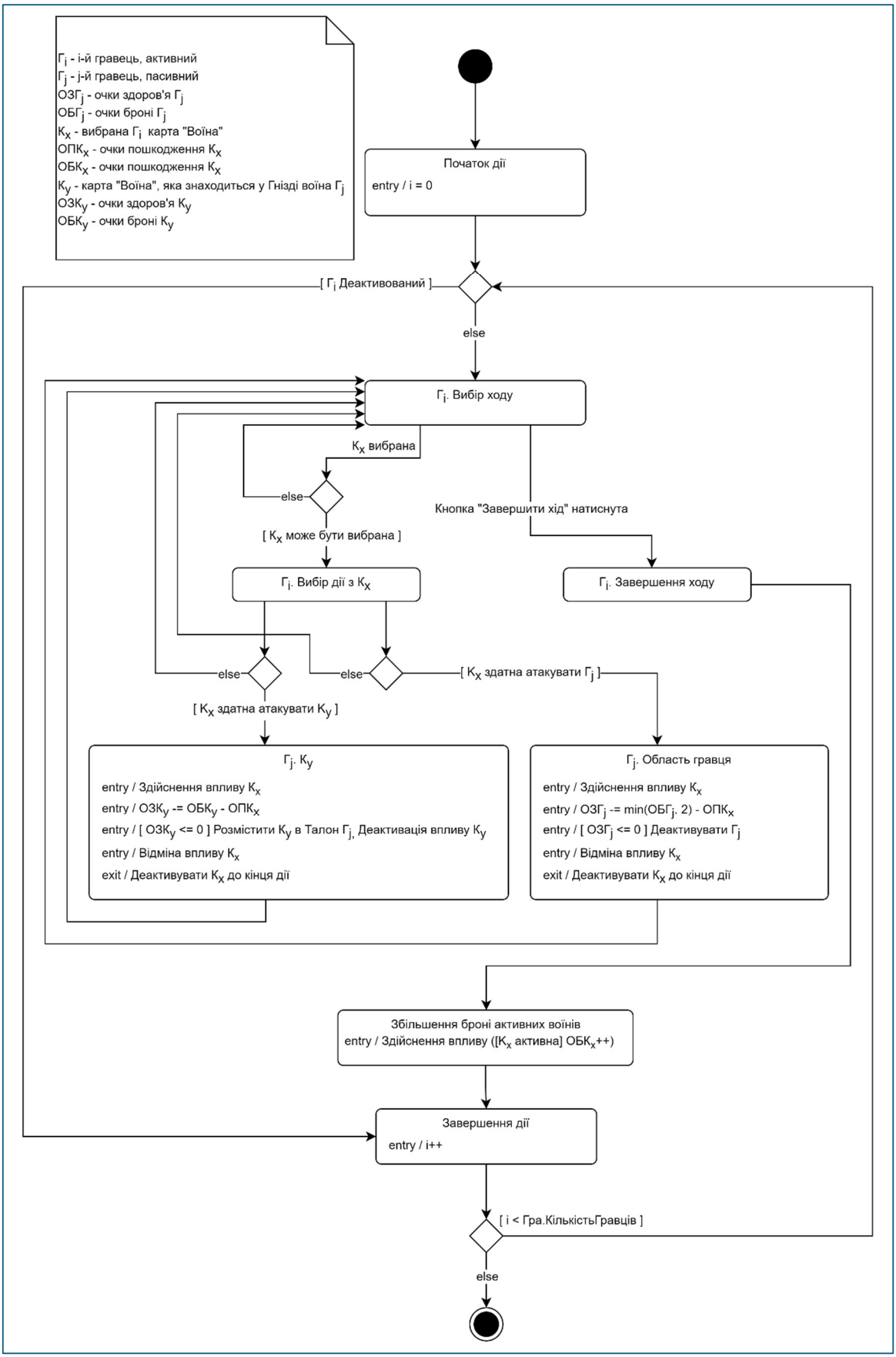


Рисунок 3.5 – UML діаграм станів для фази «Битва Армій» (Рисунок виконаний самостійно)

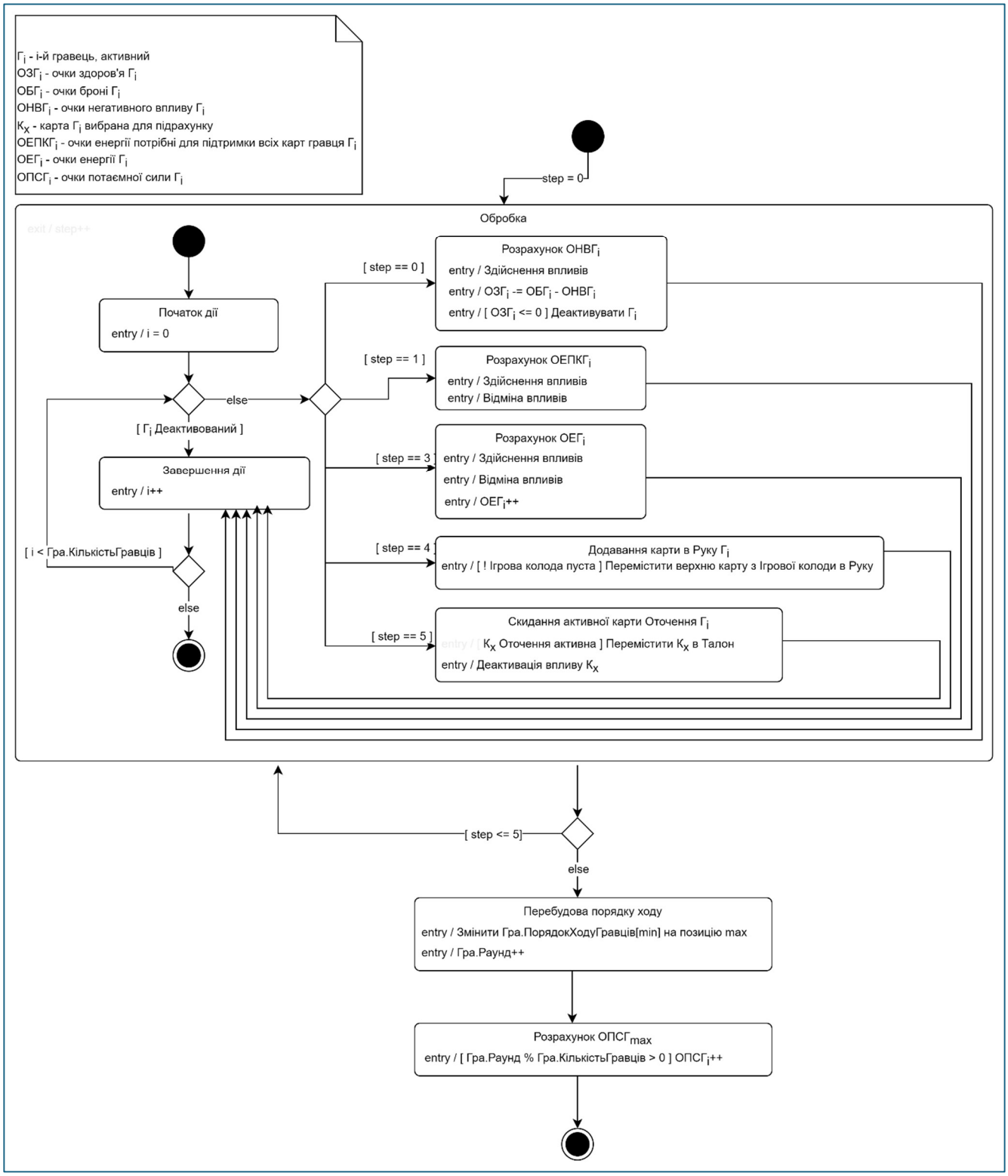


Рисунок 3.6 – UML діаграм станів для фази «Завершення Кола» (Рисунок виконаний самостійно)

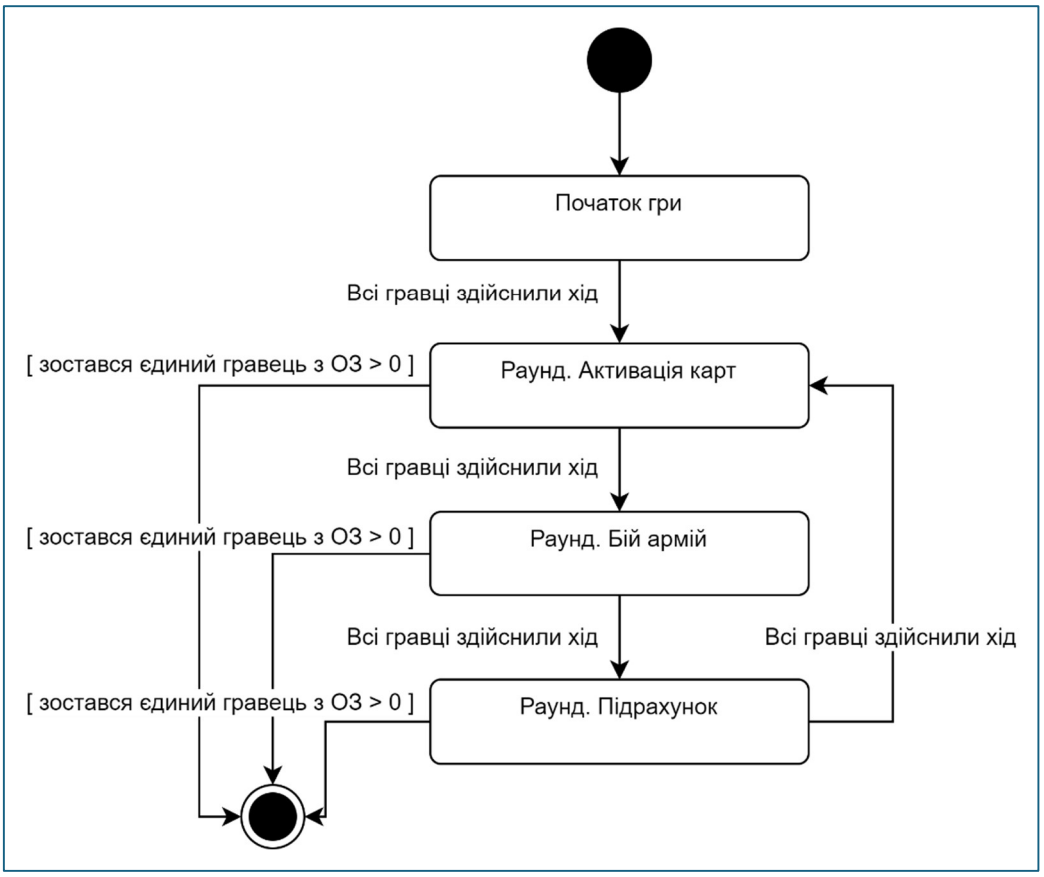


Рисунок 3.7 – Загальна UML діаграм станів (Рисунок виконаний самостійно)

UML діаграми станів формалізовано описують правила гри та дозволяють відносно легко перевести їх в програмний код. Наступним етапом необхідно виокремити сутності, що зустрічаються в правилах та в діаграмах станів, щоб сформувати структуру даних, яка використовується в грі.

До таких сутностей та їхніх властивостей відносяться:

– «Гра»;
– «Гравець» (ідентифікатор, ОЗ, ОЕ, ОБ, ОНВ, ОПС);
– «Карта» (назва, тип, ОЗ, ОЕА, ОБ, ОП);
– «Здібність» (назва, опис);
– «Рука»;
– «Колода»;
– «Гральна колода»;
– «Гніздо»;

– «Талон»;

– «Фаза»;

– «Стан».

Перед подальшою побудовою структури даних варто окремо зупинитися на здібностях. З описів здібностей, наведених в таблиці Г.1 та правилах гри, можна виявити кілька закономірностей:

– Ефект здібності обумовлюється типом карти;

– ефект здібності активується в момент, коли карта з’являється на ігровому полі;

– ефект здібності здійснює вплив на якомусь певному етапі гри;

– ефект здібності діє або поки карта присутня на полі, або постійно з моменту активації, або до певного етапу гри;

– ефект здібності торкається або якоїсь числової характеристики, або характеристики, яку можна привести до числового виду;

– ефект здібності має об’єкт застосування;

– ефект здібності буває оборотним або необоротним;

– здібності мають повторювані та унікальні елементи, які можна звести до двох сутностей: типу та екземпляру. Тип описує як, за яких умов, коли спрацьовує ефект здібності, а екземпляр задає конкретне значення зміни.

Звідси, матимемо дві сутності (тип здібності, екземпляр здібності) замість однієї (здібність):

– «Тип здібності» (ідентифікатор, назва, опис, тип карти, оборотність, стан здійснення впливу, стан відміни впливу, об’єкт, характеристика, формула, об’єкт підсилюючої характеристики, підсилююча характеристика);

– «Екземпляр здібності» (числові характеристики).

Варто вказати, що сутності фази та стану не мають параметрів, натомість вони є характеристиками сутності гра, що вказують на поточний хід гри.

Також необхідно зауважити, що сутність колода виходить за межі визначені у другому розділі, оскільки гральні колоди гравців формуватимуться випадковим

чином з попередньо визначених карт, а не обиратимуться самим гравцем з його власних карт.

Зобразимо на рисунку 3.8 ієрархію та взаємодію даних у формі UML діаграми класів.

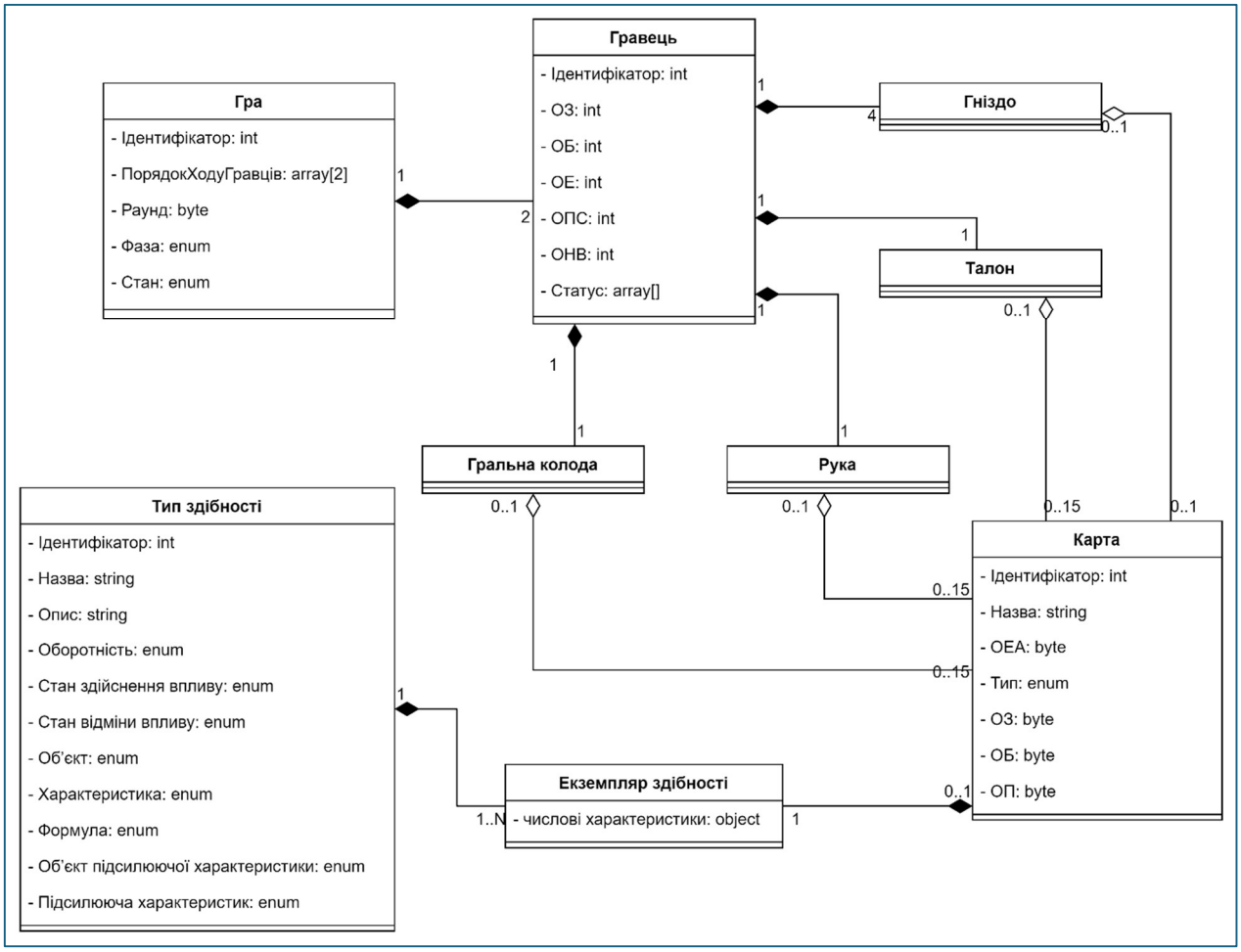


Рисунок 3.8 – Загальна UML діаграм класів (Рисунок виконаний самостійно)

Як видно з рисунку 3.8 структура даних застосунку в загальних рисах має ієрархічний характер з основою в сутності «Гра». Трохи осторонь знаходиться сутність «Тип здібності», що поєднана з рештою сутностей через «Екземпляр здібності». Як було вказано раніше, «Тип здібності» описує як, за яких умов та коли спрацьовує ефект здібності. Відповідно, в застосунку має бути функціональність, яка здійснюватиме ефект здібності (вплив), якщо передумови її виконані. В правилах гри ця процедура зустрічається на деяких етапах, як «ефект настає миттєво», «за винятком випадків, коли на них впливають здібності карт» і т.

п. Звідси, в діаграмах станів назви цієї процедури були зведені до однієї форми: «Здійснення впливів» та розмішені в місцях, де цей вплив може бути здійснений. Також необхідна процедура, яка відміняє вплив – «Відміна впливів» – у випадках: якщо карта була скинута в талон, вплив повторюється циклічно, вплив є оборотним.

Для того, щоб кожного разу при виконанні «Здійснення впливів» не перевіряти всі активні карти на ігровому полі, потрібно створити додаткову таблицю, де знаходитимуться записи активних впливів, що можуть бути здійснені на різних етапах гри. Ці записи можуть дублювати інформацію, розміщену в сутностях «Карта», «Тип здібності», «Екземпляр здібності» для більш зручної роботи з даними, але в той же час вони повинні зберігати посилання на «Карти», яким вони належать. На рисунку 3.4 запис в таблицю позначений як «Активація впливу». Згадана процедура відбувається в момент активації карти в станах «$Г_i$. Гніздо Воїна», «$Г_i$. Гніздо Оточення». З іншого боку повинна існувати симетрична функція «Деактивація впливу», що видалятиме запис з таблиці. Ця функція має викликатися, коли карта скидається з поля в талон.

Також варто звернути увагу на таку характеристику впливів, як оборотність. Звідси, вплив може бути необоротним: якщо після його відміни змінена ним характеристика не повертається до попереднього значення (або такого, яке відповідає новим умовам, якби вплив не було здійснено попередньо); та оборотним – змінена характеристика повертається до попереднього значення. Відповідно, враховуючи можливість відміни впливу на певному етапі гри, необхідно відслідковувати:

- чи вплив не був здійсненим;
- чи вплив був здійсненим і може бути здійсненим повторно;
- чи вплив був здійсненим і не може бути здійсненим повторно.

Тому варто додати до таблиці активованих здібностей зворотне значення, яке відмінятиме здійснений вплив та поле статусу з наступними можливими значеннями:

– очікує – вплив не був здійсненим;

– працює – вплив був здійсненим і може бути здійсненим повторно;

– відпрацював – вплив був здійсненим і не може бути здійсненим повторно.

Тож, таблиця активованих здібностей повинна мати наступні поля:

– ідентифікатор карти;

– ідентифікатор гравця;

– ідентифікатор типу здібності;

– числові характеристики екземпляру здібності;

– статус;

– зворотне значення.

Оскільки алгоритми роботи впливу здібностей можуть трохи відрізнятися одні від одного, то на рисунку 3.9 представимо узагальнений алгоритм роботи функції «Здійснення впливів».

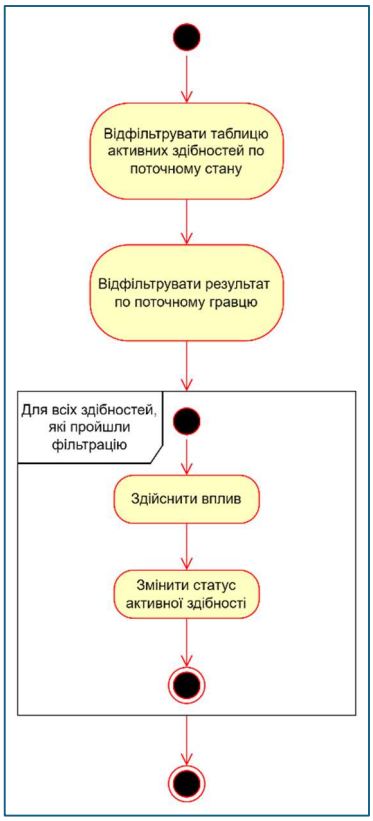


Рисунок 3.9 – Узагальнений алгоритм роботи функції «Здійснення впливів»

(Рисунок виконаний самостійно)

Алгоритм роботи функції «Відміна впливу» працює аналогічно до «Здійснення впливу», але використовує зворотну операцію та зворотне значення

Алгоритми роботи функцій «Активація впливу» та «Деактивація впливу» не включають в себе фільтрацію. «Активація впливу» – тільки додавання записів у таблицю, «Деактивація впливу» – видалення записів з таблиці.

Також варто представити алгоритм роботи для наступних здібностей (див. таблиця Г.1): «Споживає очки енергії гравця за хід» (Рисунок 3.10) та «Може завдавати шкоди безпосередньо супернику, ігноруючи його воїнів» (Рисунок 3.11).

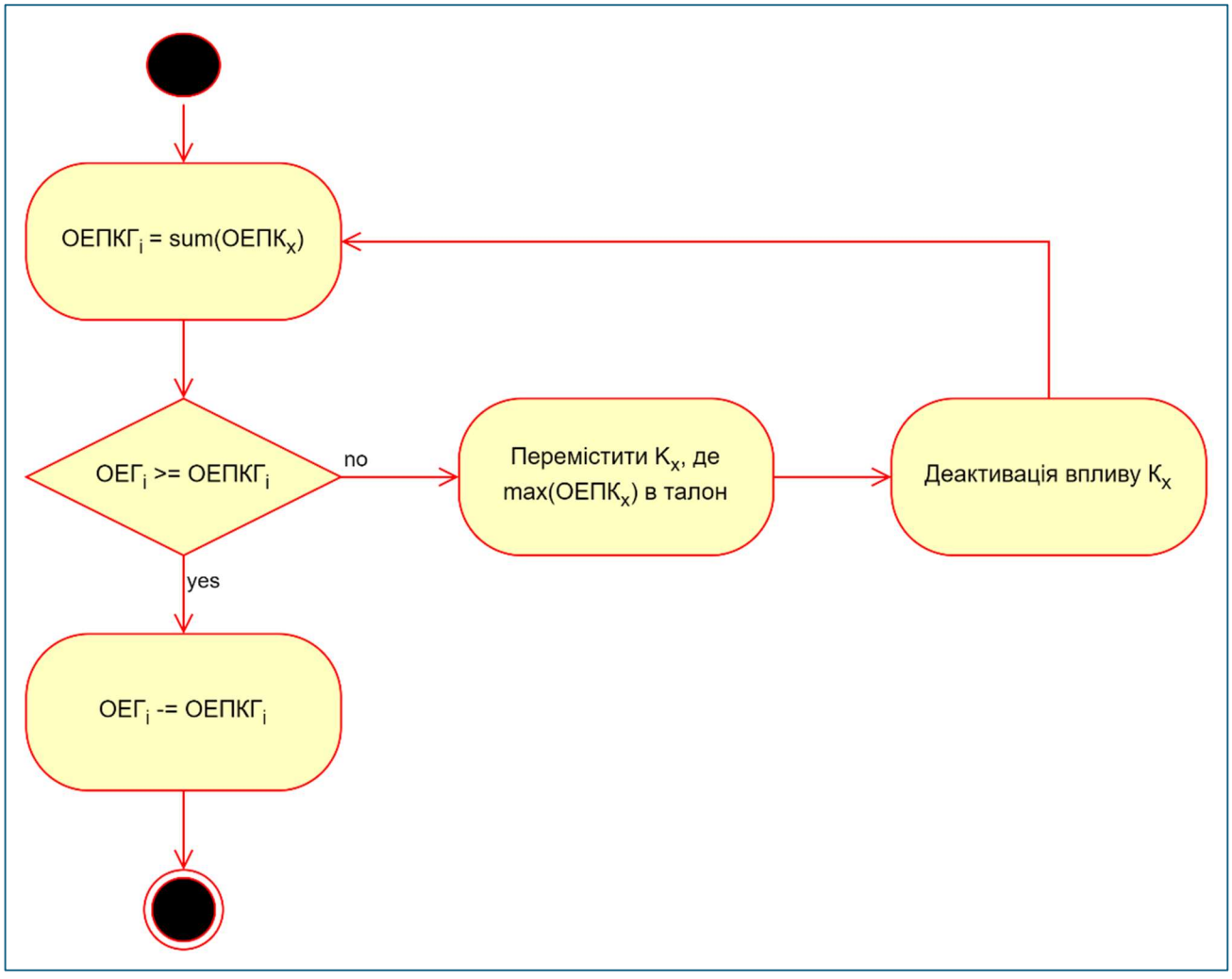


Рисунок 3.10 – Алгоритми роботи здійснення здібності «Споживає очки енергії гравця за хід» (Рисунок виконаний самостійно)

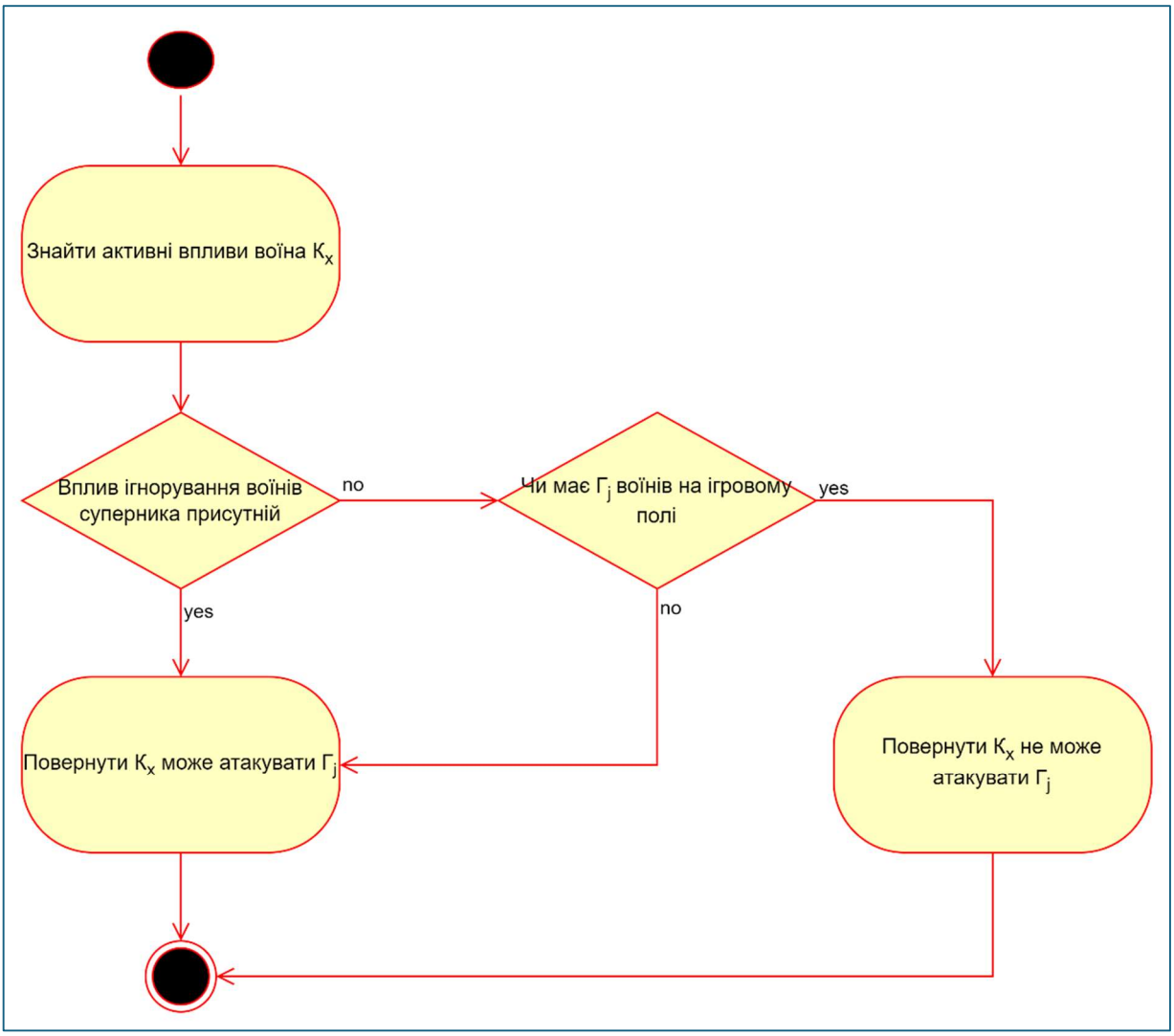


Рисунок 3.11 – Алгоритми роботи здійснення здібності «Може завдавати шкоди безпосередньо супернику, ігноруючи його воїнів» (Рисунок виконаний самостійно)

Як видно з рисунку 3.11 перевірка на наявність здібності включена в перевірку загального правила гри: «Атакувати вказаного гравця, якщо супротивник не має розгорнутих армій».

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Опис набору інструментів для створенні ЦККГ «CCG Toolkit»

«CCG Toolkit» [12] – це гнучкий та потужний фреймворк для розробки ЦККГ. Фреймворк включає в себе широкий набір інструментів для опису та реалізації різноманітних аспектів та правил ККГ. З поміж іншого він дозволяє створювати власні карти, ігрові поля, здібності. Фреймворк написаний на мові візуального програмування Blueprint.

Фреймворк включає в себе наступні готові для використання та модифікації елементи:

– ігрові поля;

– карти;

– здібності;

– елементарні механіки гри.

На рисунках 4.1 та 4.2 представимо варіанти оригінального GUI.

Рисунок 4.1 – Варіант вигляду оригінального GUI «CCG Toolkit» ([12])

Рисунок 4.2 – Варіант вигляду оригінального GUI «CCG Toolkit» ([12])

Як видно з рисунків 4.1 та 4.2 фреймворк включає в себе майже всю (відсутні деякі елементи, наприклад, піктограма для позначення очок броні) необхідну графічну частину для формування GUI ЦККГ.

З боку логічної частини, «CCG Toolkit» посеред іншого включає в себе наступний набір класів:

– Card Game Mode (сервер) – містить правила та налаштування гри. Обробляє: появу гравця, налаштування ігрового поля та гравця, передумови гри, початкові параметри;

– Card Game State (реплікується) – використовується для зберігання поточного та реплікованих станів гри. Обробляє: таймери, чергу ходу гравців, повідомлення гравців про завершення ходу;

– Card Game Player State (реплікується) – використовується для зберігання даних та обміну між клієнтами. Обробляє: ідентифікатор та цифрові

характеристики гравця, карти в руці гравця, карти в колоді гравця, активні карти гравця та інше;

– Card Game Player Controller (сервер і клієнти) – займається функціями ігроладу, що стосується гравця, карт та їхньою взаємодією. Обробляє: ігрову колоду, GUI, створює та породжує карти на ігровому полі, взаємодію гравця з картами й навпаки, тощо;

– 3D Card (реплікується) – тривимірний графічний контейнер, який відображає карту та описує її поведінку. Коли заповнюється даними, то використовується для взаємодії з іншими елементами на ігровому полі. Обробляє: спеціальні дані ігроладу, що стосуються карт, події переміщення карт на полі, тощо;

– Board Player (реплікується) – об’єкт що використовується для завдання шкоди супротивнику, графічно відображає супротивника на ігровому полі;

– Card Placement (реплікується) – елемент, який керує як, хто, за яких умов може розміщати карти на ігровому полі;

– Graveyard (реплікується) – елемент, який зберігає карти, що вже зіграли.

«CCG Toolkit» також включає документацію та інструкції для більш ефективної розробки ЦККГ на основі нього.

4.2 Опис реалізації графічної частини застосунку

Варіанти стандартного ігрового поля «CCG Toolkit», що представлені на рисунках 4.1 та 4.2 відрізняються від вигляду цільового схематичного ігрового поля, представленого на рисунку 2.1. Основними відмінностями є:

– різна кількість місць для розміщення карт;

– відмінне позиціонування карт;

Також необхідно звернути увагу, що частина GUI, яка описує числові характеристики гравців в оригінальному GUI містить меншу кількість елементів,

аніж потрібна. Таким чином до ОЗ (Health), ОЕ (Energy або Mana) необхідно додати:

– ОБ (Armor);
– ОНВ (Negatory);
– ОПС (Arcane).

Відповідно, необхідно створити новий рівень Arena_ZaarWorld (нащадок класу Map), який описуватиме ігрове поле необхідного формату. Представимо на його графічний вигляд на рисунку 4.3.

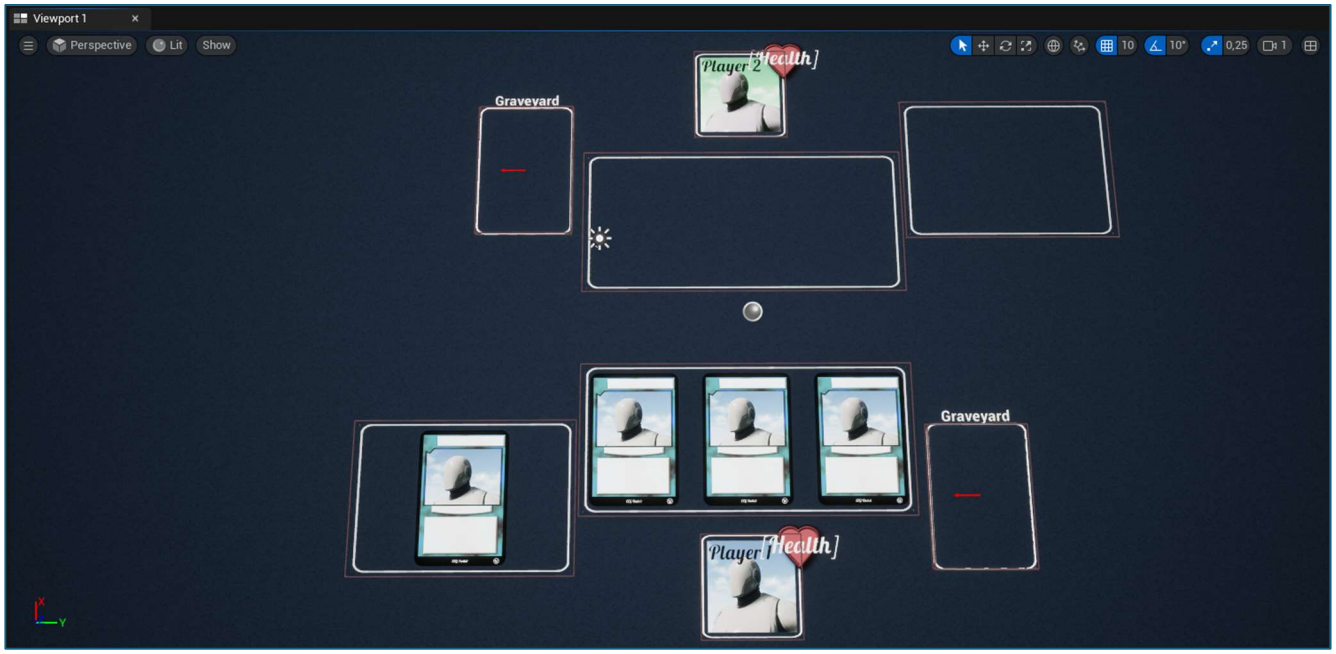


Рисунок 4.3 – Графічний вигляд рівня Arena_ZaarWorld в редакторі Unreal Editor (Рисунок виконаний самостійно)

Для виправлення елементів GUI, що відповідають за вивід характеристик гравця та його суперника необхідно зробити зміни в класах PlayerProfileUI (нащадок класу User Widget) та OpponentProfileUI (нащадок класу User Widget). На рисунках 4.4 та 4.5 представимо фінальний вигляд.

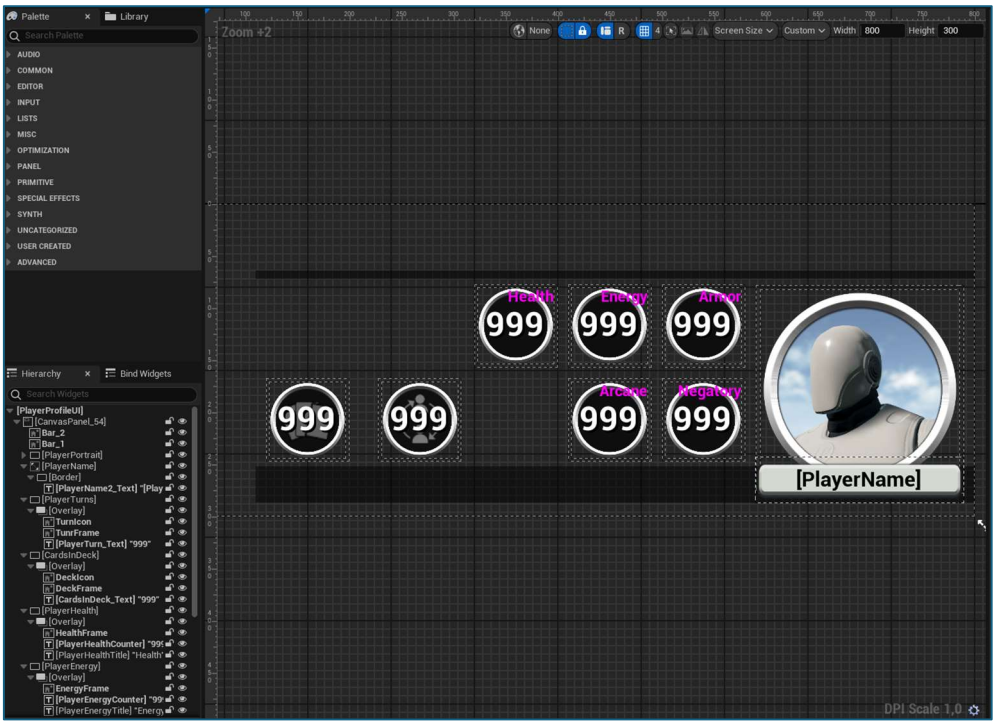


Рисунок 4.4 – Графічний вигляд віджету PlayerProfileUI в редакторі Unreal Editor (Рисунок виконаний самостійно)

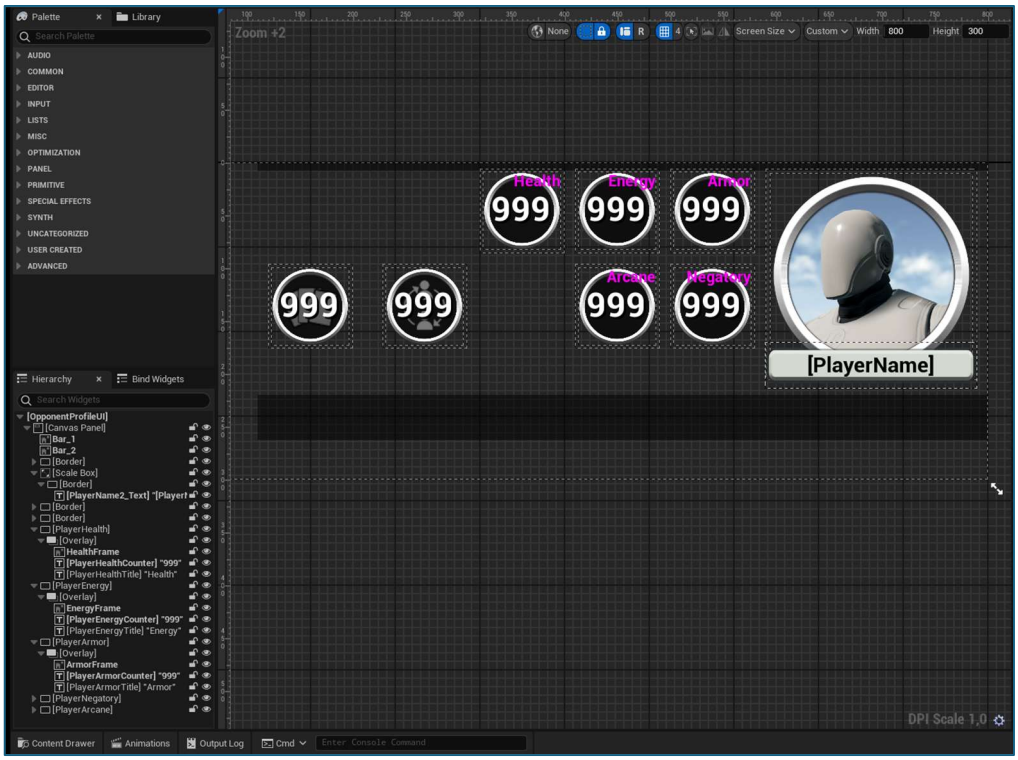


Рисунок 4.5 – Графічний вигляд віджету OpponentProfileUI в редакторі Unreal Editor (Рисунок виконаний самостійно)

Результуючий вигляд GUI для обох гравців представимо на рисунку 4.6.

Рисунок 4.6 – GUI ЦККГ «Світ Заару» (Рисунок виконаний самостійно)

Також необхідно зазначити що в оригінальному описі карти в класі 3DCard (нащадок класу Actor) відсутня характеристика ОБ. Відповідно 3DCard необхідно відредагувати та додати характеристику, якої бракує.

Варто зауважити, що додавання нових характеристик включає в себе додавання нових змінних, нових функцій обробки, нових візуальних елементів відображення.

На рисунку 4.7 представимо оновлений вигляд 3DCard.

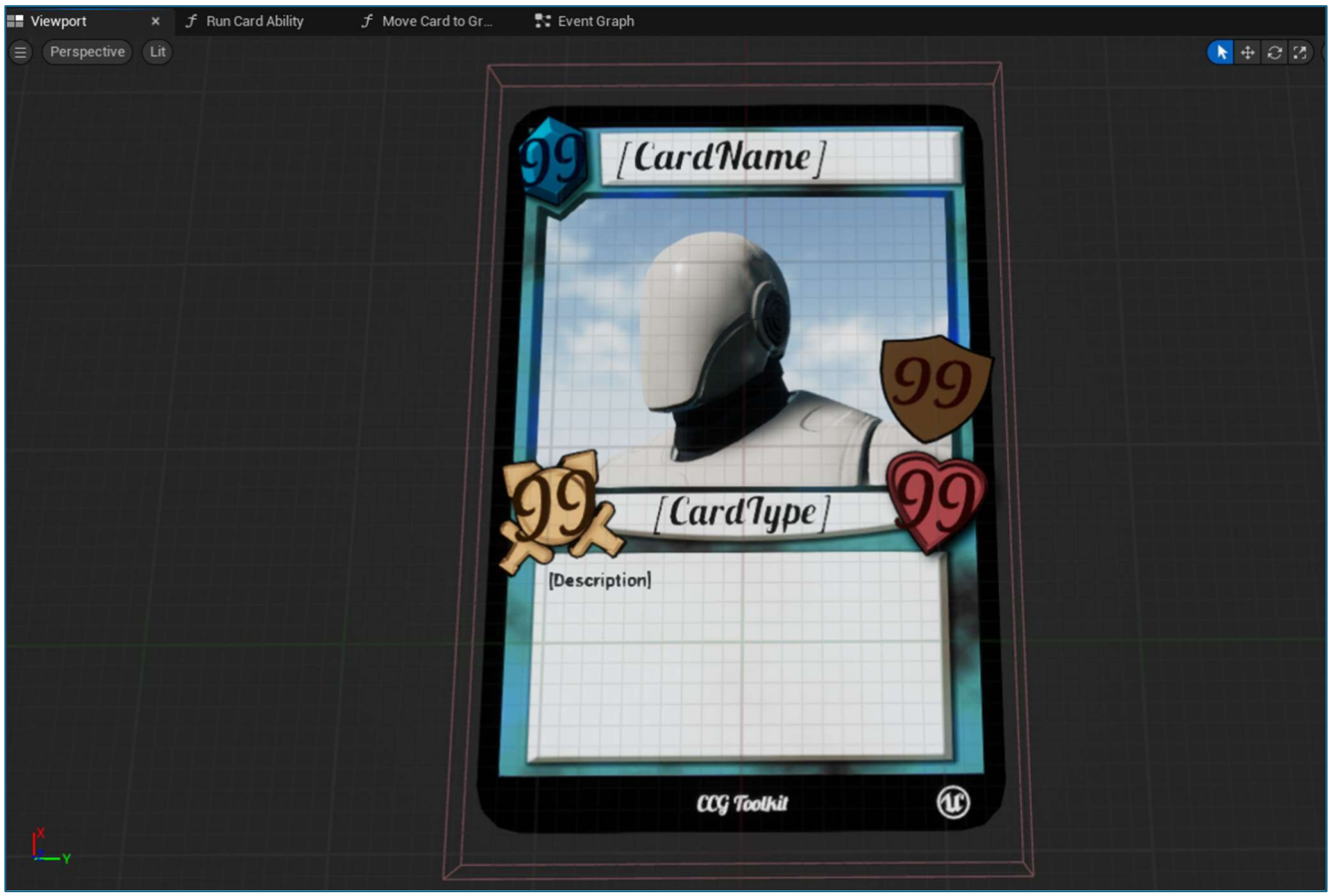


Рисунок 4.7 – Графічний вигляд 3DCard (Рисунок виконаний самостійно)

Варто зазначити, що приведені на рисунках зміни є лише візуальною частиною змін, які були внесені в «CCG Toolkit» задля того, щоб відображення та його опис відповідали вимогам, визначеним у попередніх розділах.

4.3 Опис реалізації логічної частини застосунку

«CCG Toolkit» вже має власну програмну архітектуру, що забезпечує можливість двокористувацької гри без якихось додаткових модифікацій. Особливостями базової програмної архітектури фреймворку є:

– подіє-орієнтована архітектура;

– інкапсуляція даних щодо гравця в об’єкті класу CardGamePlayerState (нащадок класу GameState);

– інкапсуляція функцій та подій, що стосуються гравця в об’єкті класу CardGamePlayerController (нащадок класу PlayerController);

– інкапсуляція опису ігрової партії та її загальної функціональності в об’єкті класу CardGameState (нащадок класу GameState);

– інкапсуляція опису карти та її загальної функціональності в об’єкті класу 3DCard (нащадок класу Actor);

– наявність інтерфейсів та функціональних бібліотек для спрощення розробки;

– готова реалізація можливості двокористувацької гри через локальну мережу;

– наявність інструментів для збирання ігрової колоди гравця та її збереження.

Відповідно, алгоритми роботи ЦККГ «Світ Заару», описані у попередньому розділі мають бути модифіковані для того щоб використовувати існуючу функціональність фреймворку і не ламати існуючої робочої архітектури.

Розглянемо алгоритми роботи фаз гри, представлені у вигляді UML діаграм станів на рисунках 3.5, 3.6, 3.7, 3.8. Кожен з алгоритмів описує як загальний стан гри, так і стан гри для кожного з гравців окремо. Враховуючи архітектуру фреймворку, необхідно винести загальну частину в клас CardGameState. До неї відносяться: фаза гри (ZW_SMGroups CurrentGroup), поточний хід в межах фази (integer CurrentTurnWithinGroup), загальна кількість ходів у фазі (integer TotalTurnsWithinGroup). Решта функціональності, яка стосується стану конкретного гравця переноситься в клас CardGamePlayerController. Таким чином, враховуючи інструменти фреймворку, діаграма станів для фази «Активація Карт» матиме вигляд, представлений на рисунку 4.8, а діаграма станів для фази «Битва Армій» матиме вигляд, представлений на рисунку 4.9

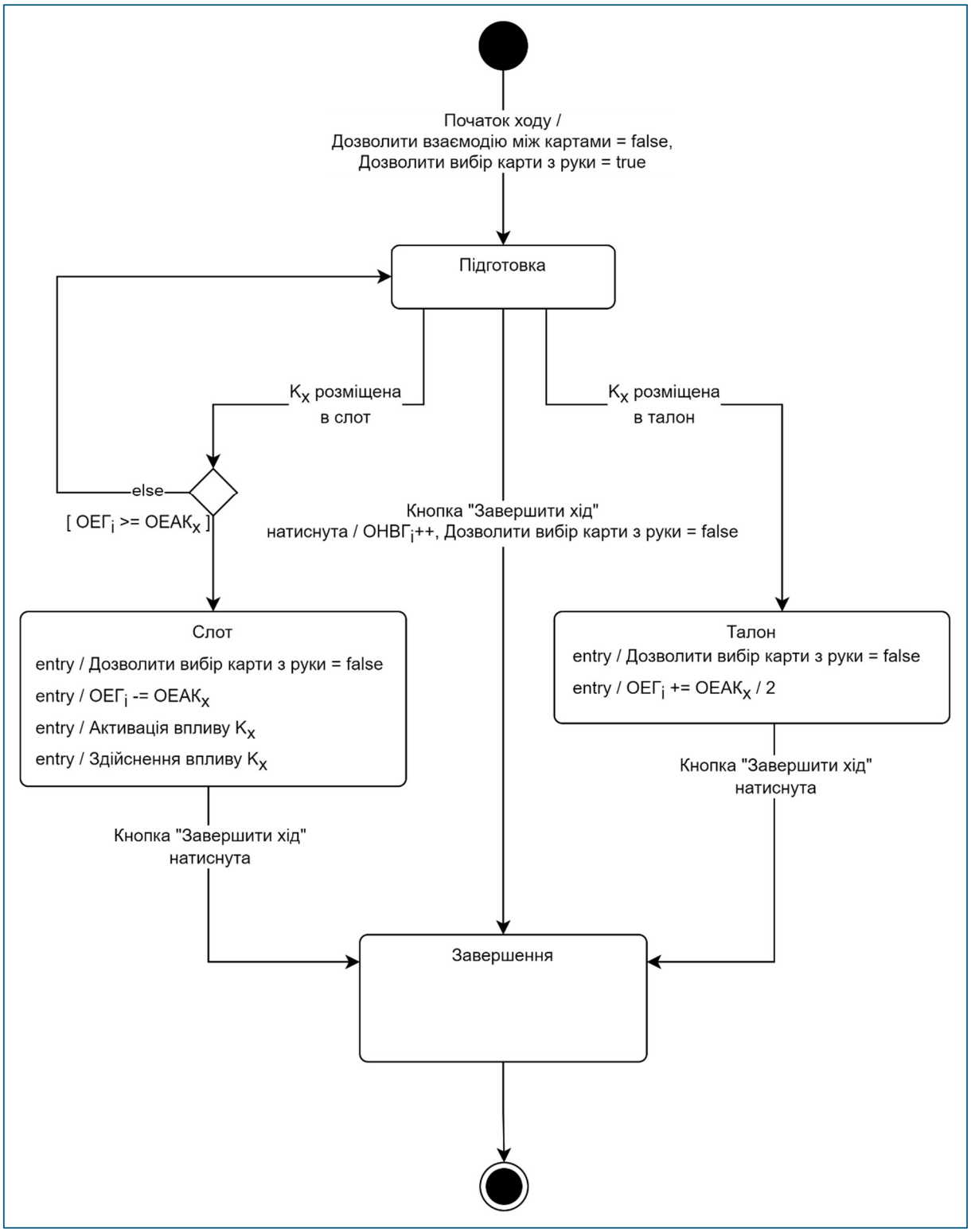


Рисунок 4.8 – Модифікована UML діаграм станів для фази «Активація Карт» (Рисунок виконаний самостійно)

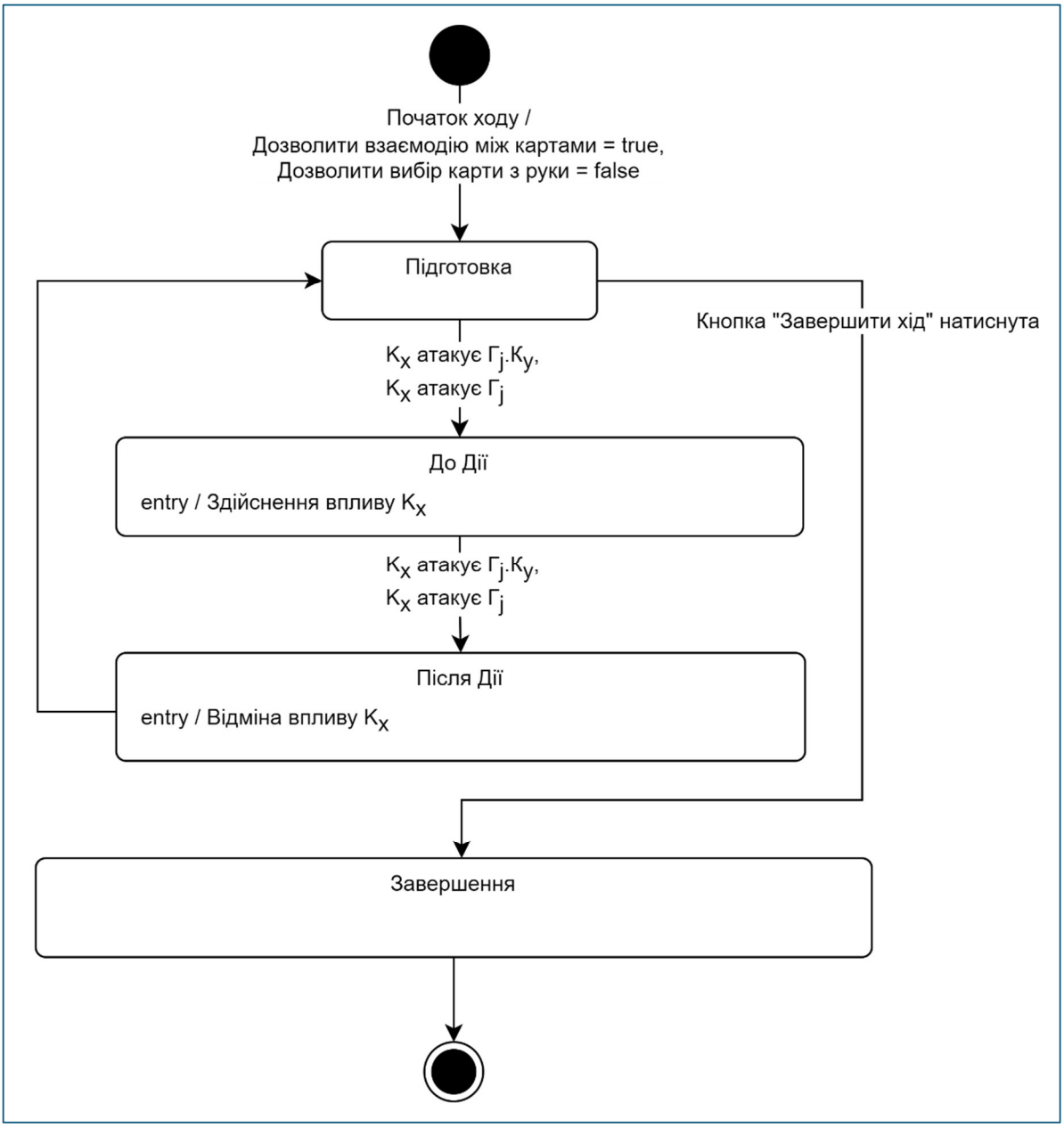


Рисунок 4.9 – UML діаграм станів для фази «Битва Армій» (Рисунок виконаний самостійно)

Оскільки алгоритм роботи фази «Завершення Кола» стосується розрахунків і не потребує втручання гравця, то він залишається без змін.

На рисунках 4.10, 4.11, 4.12 наведемо приклади зроблених змін в існуючому програмному коді для реалізації необхідної логіки на мові візуального програмування Blueprint. Зроблені зміни позначені як «ZW_StateMachine injection».

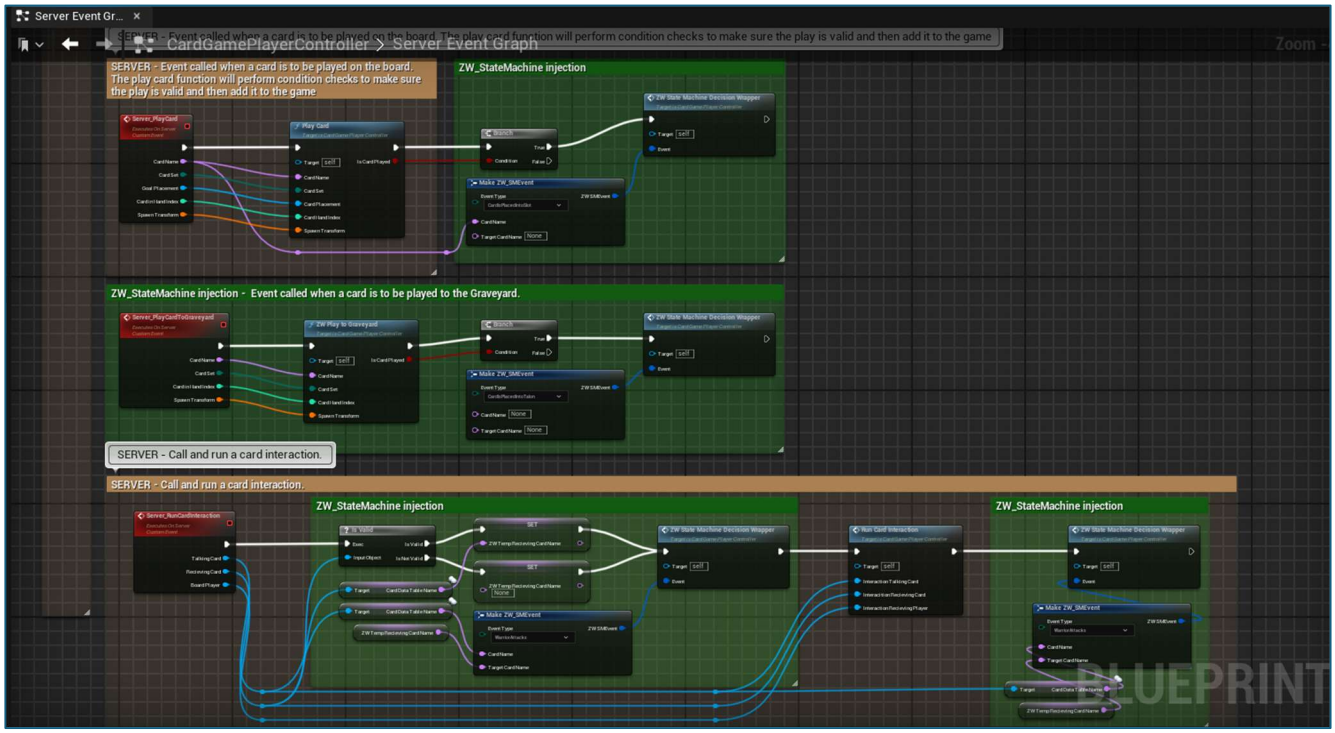


Рисунок 4.10 – Приклад зроблених змін в діаграмі Server Event Graph класу CardGamePlayerController (Рисунок виконаний самостійно)

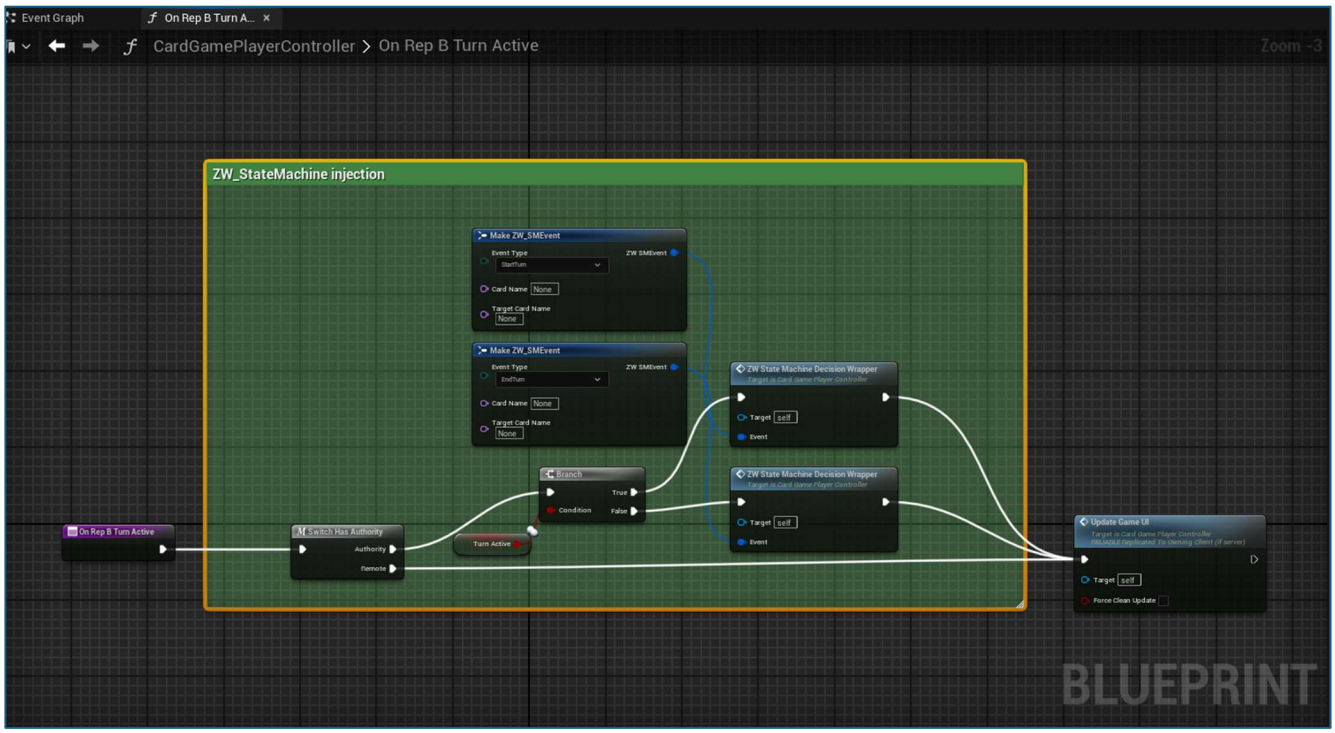


Рисунок 4.11 – Приклад зроблених змін в функції OnRep_bTurnActive класу CardGamePlayerController (Рисунок виконаний самостійно)

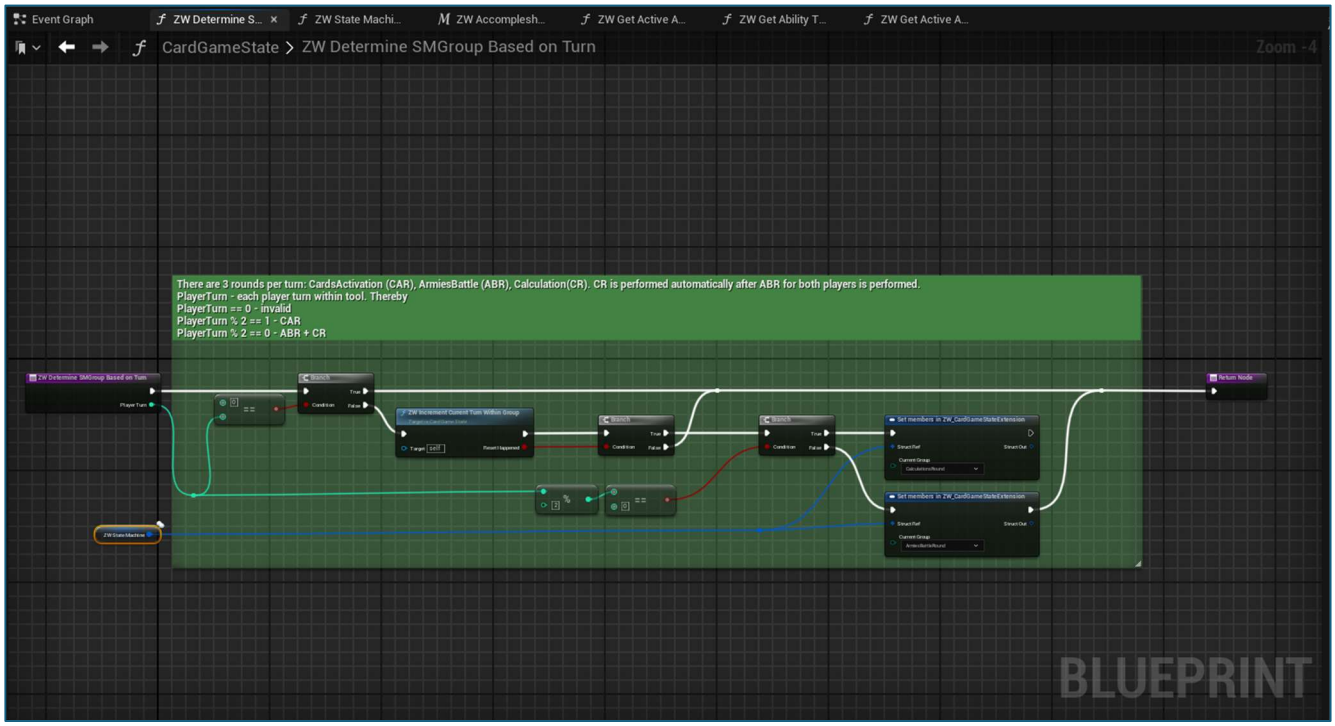


Рисунок 4.12 – Приклад зроблених змін в діаграмі Server Event Graph класу CardGamePlayerController (Рисунок виконаний самостійно)

Здібності карт були формалізовані та розміщені в контейнері даних типу DataTable. На рисунку 4.13 представимо вигляд таблиці типів здібностей карт відповідно до полів, які були описані в попередньому розділі.

Data Table | Data Table Details

Search

	Ro	ID	Title	Description	Reversibility	StartPerformingState	StopPerformingStat	Object	Feature	Formula	AmplificationObject	AmplificationFeature
1	Er	Environment_1	The opponent	When activated,	Irreversible	CAR_Slot	CAR_None	Opponent	Health	SubtractionWithAmplification	Self	Arcane
2	Er	Environment_2	Increase playe	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Armor	Addition	Self	None
3	Er	Environment_3	Removes all th	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Negatory	Assignment	Self	None
4	Er	Environment_4	Increases play	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Arcane	Multiplication	Self	None
5	Er	Environment_5	Restores playe	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Health	Addition	Self	None
6	Er	Environment_6	Increases play	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Energy	Multiplication	Self	None
7	Er	Environment_7	The player get	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Energy	AdditionWithAmplification	Self	Arcane
8	Er	Environment_8	Deprives the o	When activated,	Irreversible	CAR_Slot	CAR_None	Opponent	Armor	Assignment	Self	None
9	Er	Environment_9	Increases all p	When activated,	Reversible	CAR_Slot	CAR_None	SelfWarriors	Damage	Addition	Self	None
10	Er	Environment_10	Decreases all (	When activated,	Reversible	CAR_Slot	CAR_None	OpponentWarriors	Armor	Subtraction	Self	None
11	W	Warrior_1	Can damage t	This warrior can	Irreversible	ABR_PreAction	CAR_None	SelfWarrior	None	Extra	Self	None
12	W	Warrior_2	Additional dar	This warrior uses	Reversible	ABR_PreAction	ABR_PostAction	SelfWarrior	Damage	AdditionWithAmplification	Opponent	Negatory
13	W	Warrior_3	Restores X poi	This warrior rest	Irreversible	ABR_PreAction	ABR_PostAction	SelfWarriorsExclusion	Health	Addition	Self	None
14	W	Warrior_4	Reduces each	This warrior redu	Reversible	CAR_Slot	CAR_None	OpponentWarriors	Damage	Subtraction	Self	None
15	W	Warrior_5	Increases near	This warrior incr	Reversible	CAR_Slot	CAR_None	SelfWarriorsExclusion	Armor	Addition	Self	None
16	W	Warrior_6	Ignores oppon	This warrior ignc	Reversible	ABR_PreAction	ABR_PostAction	SelfWarrior	Damage	AdditionWithAmplification	OpponentWarrior	Armor
17	W	Warrior_7	Increases near	This warrior incr	Reversible	CAR_Slot	CAR_None	SelfWarriorsExclusion	Health	Addition	Self	None
18	W	Warrior_8	Consumes the	This warrior con	Irreversible	CR_EnergyConsumption	CAR_None	Self	Energy	Subtraction	Self	None
19	W	Warrior_9	Each cycle the	This warrior incr	Irreversible	CR_EnergyDonation	CR_EnergyDonation	Self	Armor	Addition	Self	None
20	W	Warrior_10	Each cycle the	This warrior incr	Irreversible	CR_EnergyDonation	CR_EnergyDonation	Self	Arcane	Addition	Self	None

Рисунок 4.13 – Зміст контейнера даних ZW_Abilities (Рисунок виконаний самостійно)

Представимо на рисунках 4.14, 4.15, 4.16, 4.17 реалізації функцій «Активація впливу», «Деактивація впливу», «Здійснення впливу», «Відміна впливу».

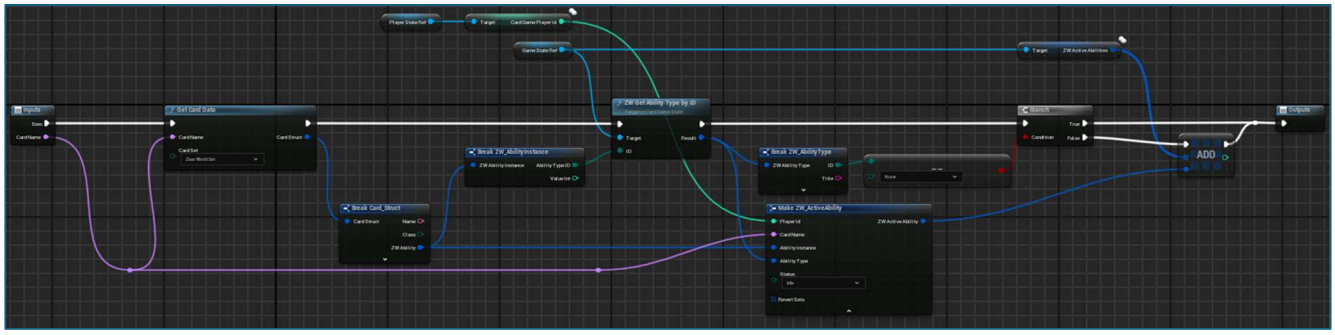


Рисунок 4.14 – Програмна реалізація функції «Активація впливу» (Рисунок виконаний самостійно)

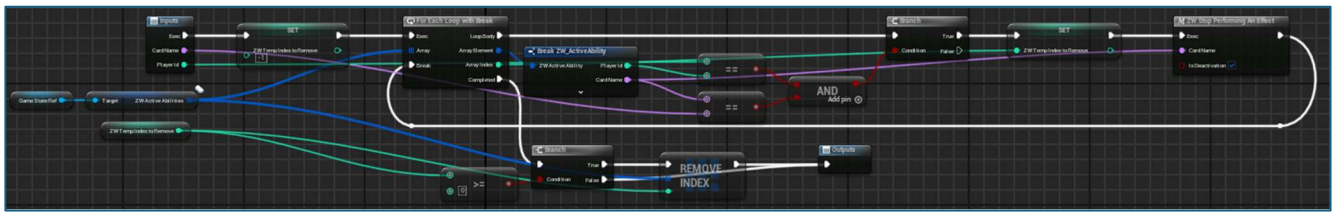


Рисунок 4.15 – Програмна реалізація функції «Деактивація впливу» (Рисунок виконаний самостійно)

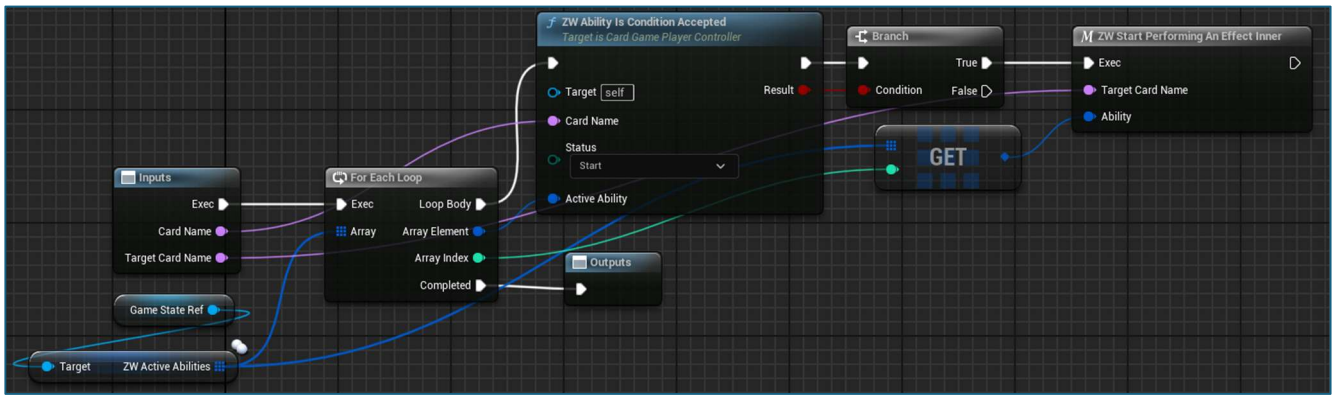


Рисунок 4.16 – Програмна реалізація функції «Здійснення впливу» (Рисунок виконаний самостійно)

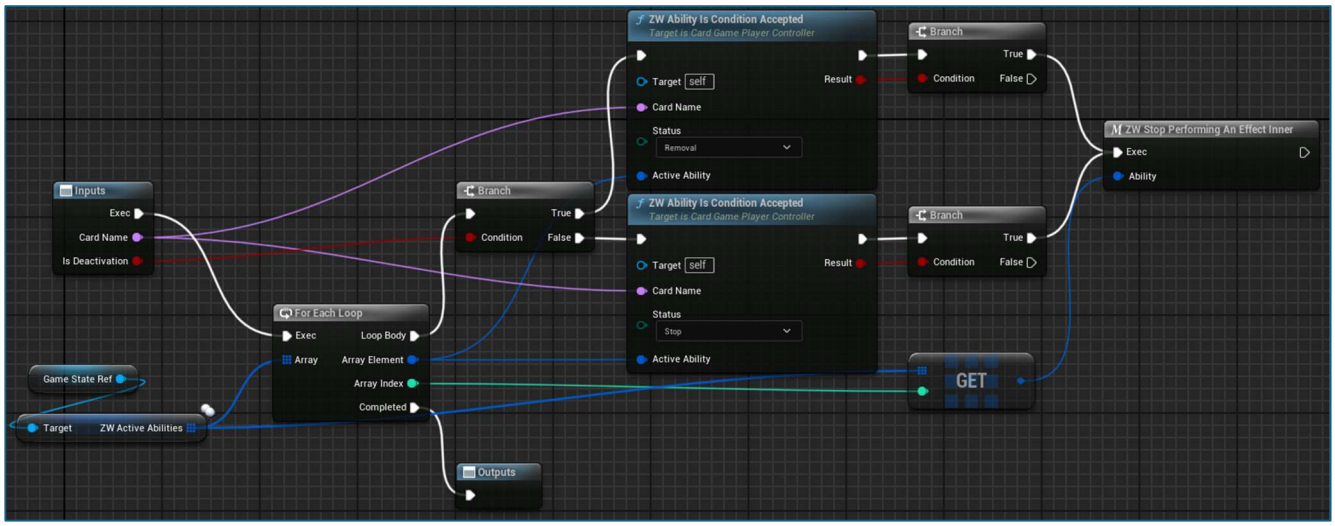


Рисунок 4.17 – Програмна реалізація функції «Відміна впливу» (Рисунок виконаний самостійно)

Як видно з рисунків 4.16 та 4.17 функції «Здійснення впливу» та «Відміна впливу» мають внутрішні частини. Їхній код займає значно більше місця, тому наведений в Додатку В.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування було проведене вручну без залучення додаткових засобів. Перевірялися як функціональні, так і не функціональні вимоги.

5.1 Тестування графічної частини застосунку

Тестування графічної частини включає в себе перевірку наступних пунктів:

– відповідність готового GUI вимогам, поставленим у другому розділі (схематичні зображення представлені на рисунках 2.1, 2.2, 2.3);

– активні елементи інтерфейсу реагують на дії гравця;

– відсутність перекривання між елементами інтерфейсу, що заважає огляду;

– зручність користування GUI при зміні розміру екрану.

На рисунку 5.1 представимо вигляд GUI гравця в розгорнутому вікні, на рисунках 5.2 та 5.3 у вікнах меншого розміру.

Рисунок 5.1 – GUI гравця в розгорнутому вікні (Рисунок виконаний самостійно)

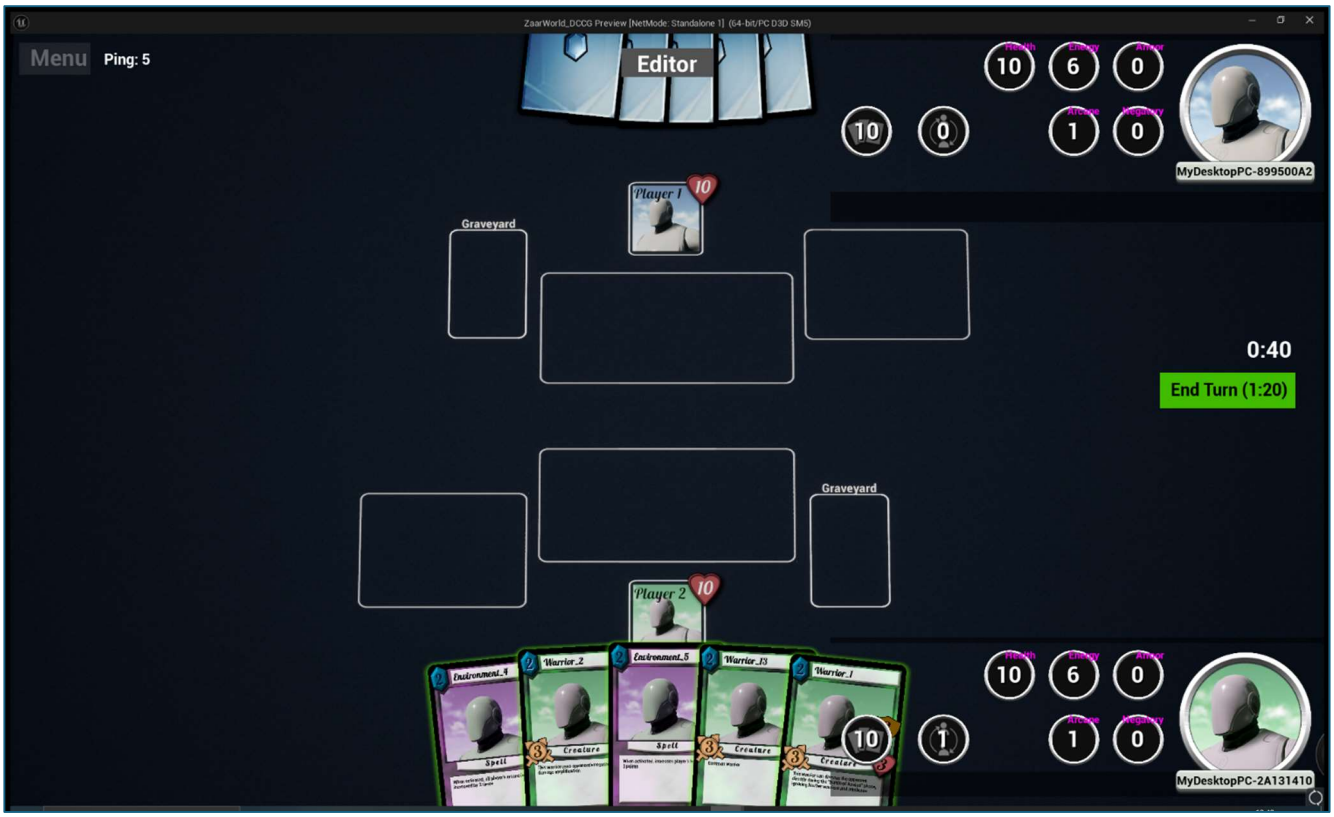


Рисунок 5.2 – GUI гравця у вікні маленького розміру (Рисунок виконаний самостійно)

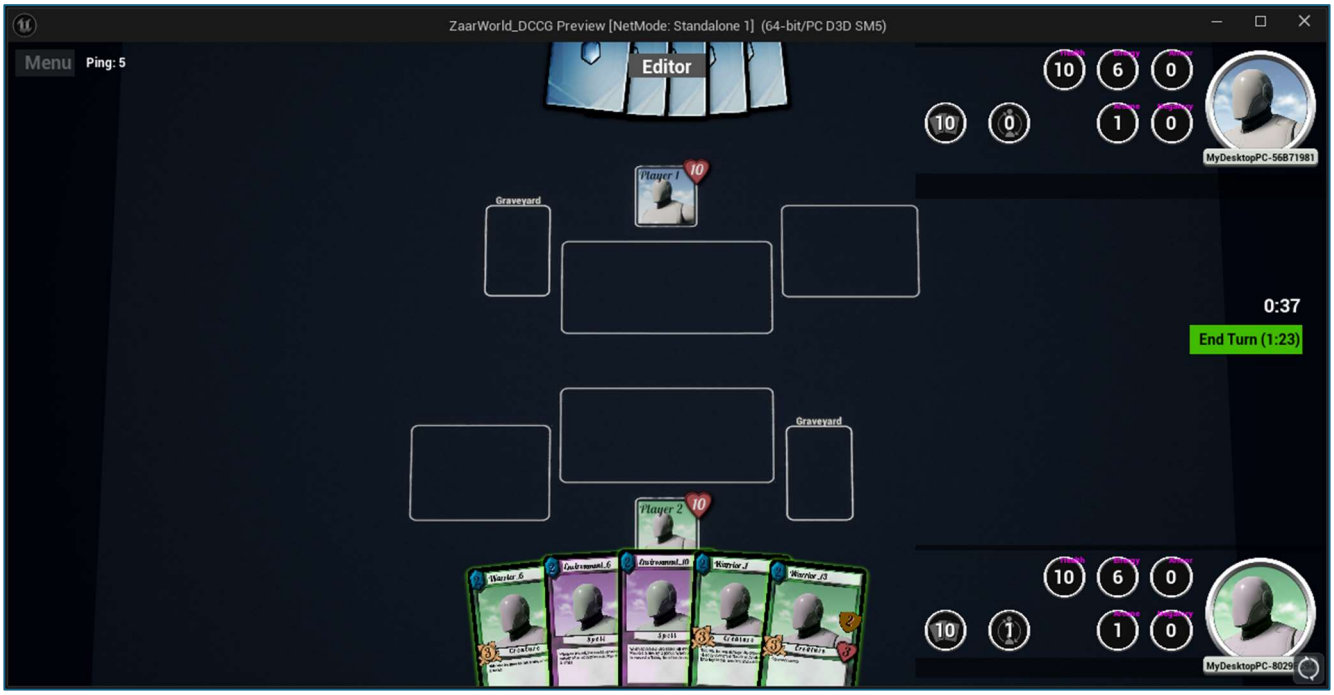


Рисунок 5.3 – GUI гравця у вікні середнього розміру (Рисунок виконаний самостійно)

Як видно з рисунків, існуюча GUI трохи відрізняється від схематичного вигляду, представленого на рисунку 2.1 за рахунок того, що характеристики гравця та суперника були зміщені вправо до меж вікна. З одного боку це покращує зручність використання, бо однакові характеристики суперників знаходяться на одній вертикальній лінії і вони швидко потрапляють на очі. Недоліком реалізації є те, що при зміні розміру ігрового вікна характеристики гравця можуть перекривати зображення його карт «у руці» (див. рисунок 5.1, 5.2). Найкращий розмір вікна без перекриття представлений на рисунку 5.3, але, на жаль, він не виставляється автоматично, тож гравцеві доводиться після початку гри змінювати розмір під себе.

На рисунках 5.4, 5.5 представимо деякі приклади анімації в GUI.

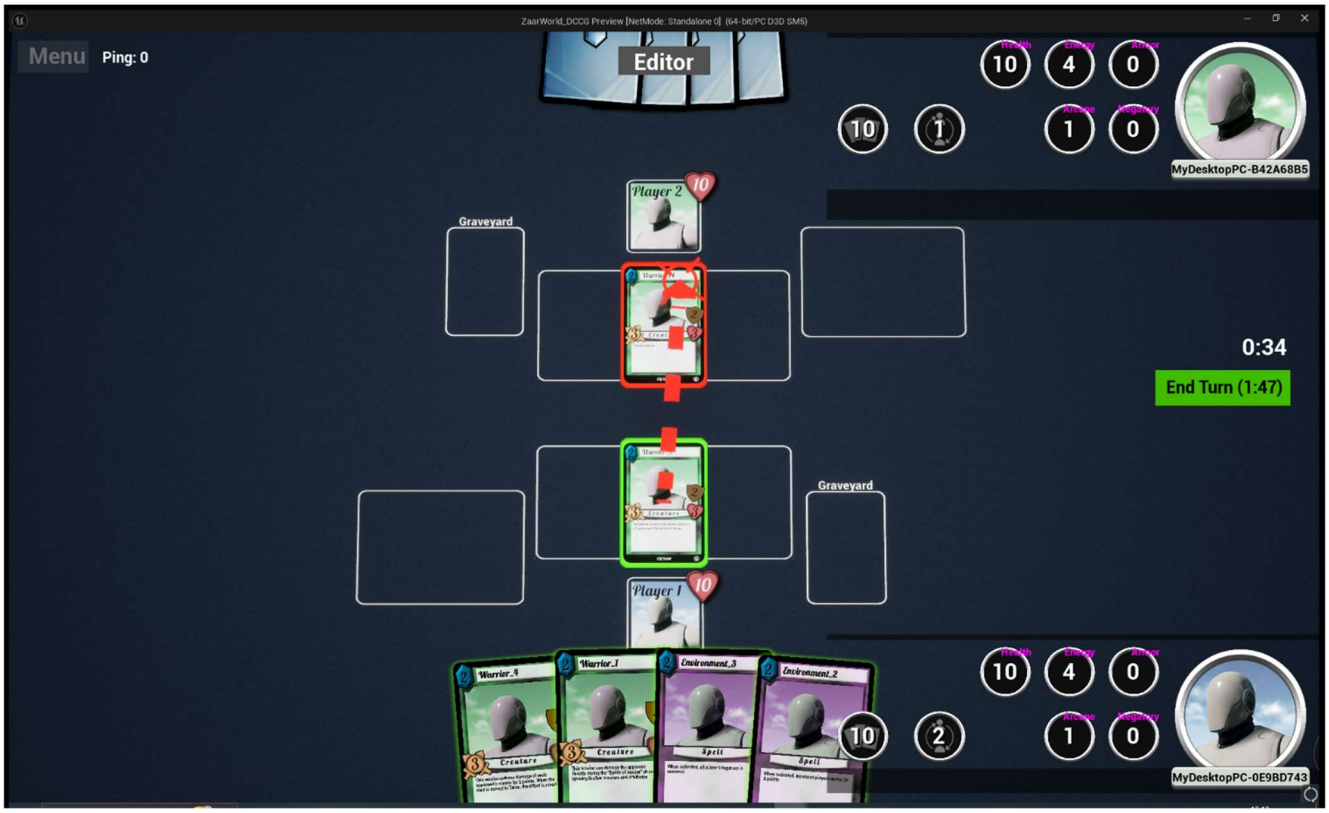


Рисунок 5.4 – Анімація атаки картою воїна суперника у вікні повного розміру

(Рисунок виконаний самостійно)

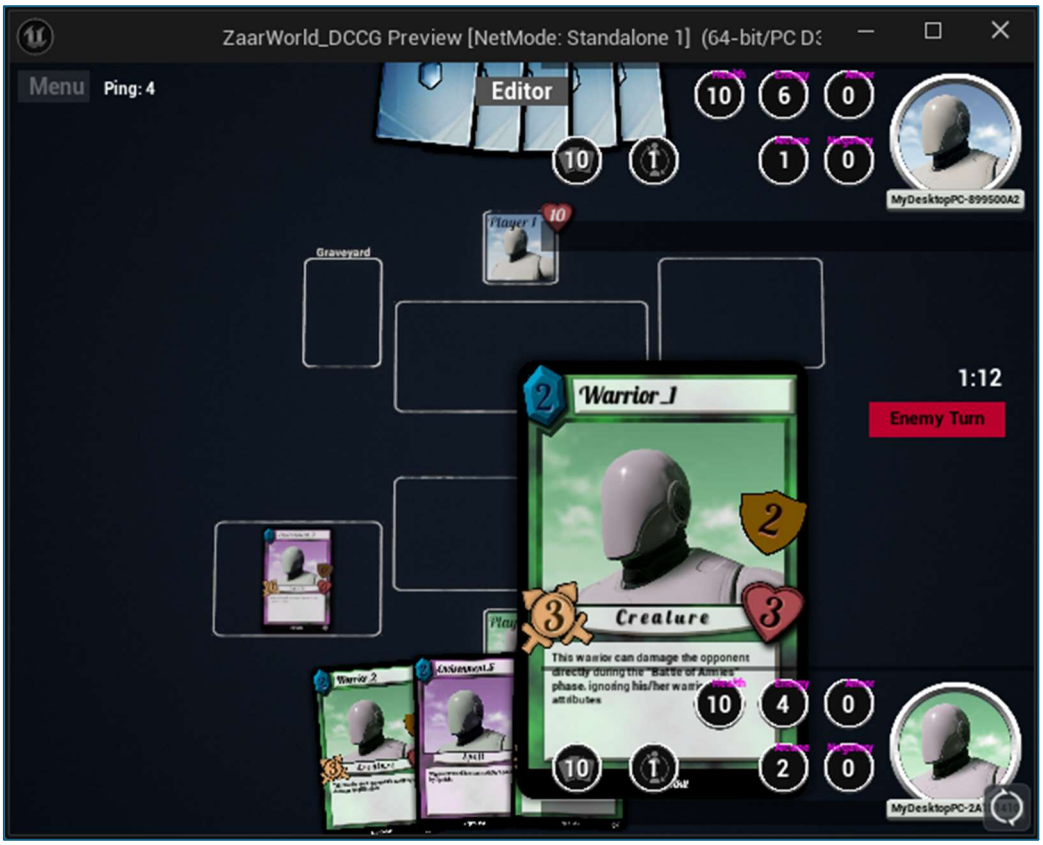


Рисунок 5.5 – Анімація перегляду карти «Воїна» «в руці» у вікні малого розміру (Рисунок виконаний самостійно)

Рисунок 5.6 – Анімація перегляду карти «Оточення» «в руці» у вікні малого розміру (Рисунок виконаний самостійно)

Як видно з рисунку 5.4 активна карта виділена, а напрямок дії вказаний піктограмою цілі, що є зручним для гравця.

На рисунках 5.5, 5.6 ми знову натрапляємо на згадану раніше проблему з перекриттям. Також вигляд карт «Воїна» та «Оточення» трохи відрізняється від схематичного вигляду з рисунків 2.2 та 2.3. Але ці відмінності зберігають всю повноту інформації.

Отже, хоч реалізований GUI дечим відрізняється від спроектованої, результат роботи однаково зберігає потрібні функції. Єдиною серйозною проблемою є перекриття елементів GUI у випадку малих розмірів вікна.

5.2 Тестування логічної частини застосунку

Тестування логічної частини застосунку можна розділити на три частини:

- тестування скінченного автомату;
- тестування загальних взаємодії;
- тестування здібностей.

Тестування скінченного автомату, який лежить в основі реалізації правил ходу гри, полягає у прогоні різних варіацій гри та співставлення тих переходів між станами скінченного автомату, які відбулися з тими, які мали би відбутися відповідно до правил гри. Наприклад, правильна послідовність переходів між станами повинна мати вигляд:

- CAR_None -> StartTurn -> CAR_Preparation;
- CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot;
- CAR_Slot -> EndTurn -> CAR_Completion;
- CAR_Completion -> StartTurn -> ABR_Preparation.

На рисунку 5.7 представлений код, який логує зміну станів скінченного автомату.

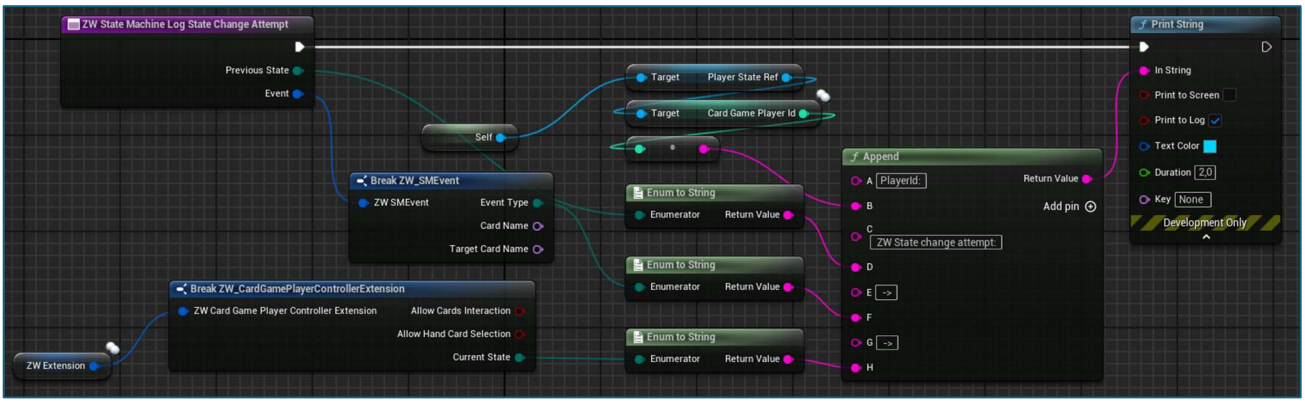


Рисунок 5.7 – Код логування зміни станів скінченного автомату (Рисунок виконаний самостійно)

На рисунку 5.8 представимо приклад отриманих логів.

```
Server: PlayerId: 1 ZW State change attempt: CAR_None -> EndTurn -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 2 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 2 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 1 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 1 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 1 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 2 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 2 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 1 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> WarriorAttacks -> ABR_PreAction
Server: PlayerId: 1 ZW State change attempt: ABR_PreAction -> WarriorAttacks -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 1 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 2 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 1 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 2 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 1 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 2 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 1 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 2 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 1 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 2 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 1 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 1 ZW State change attempt: CAR_None -> EndTurn -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 2 ZW State change attempt: CAR_None -> EndTurn -> CAR_None
Server: PlayerId: 1 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 1 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 1 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 2 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 2 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 1 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> WarriorAttacks -> ABR_PreAction
Server: PlayerId: 1 ZW State change attempt: ABR_PreAction -> WarriorAttacks -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 2 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 2 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 2 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 1 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 2 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 1 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 2 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 1 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 2 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 1 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 2 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 1 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 2 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
Server: PlayerId: 1 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
```

Рисунок 5.8 – Приклад логів зміни станів скінченного автомату (Рисунок виконаний самостійно)

В результаті аналізу логів було зроблено висновок, що скінченний автомат працює справно – відповідно правилам гри.

До тестування загальних взаємодій відноситься перевірка поведінки числових характеристик гравців та карт при взаємодії. В таблиці 5.1 представимо виключний перелік загальних впливів з описом механік розрахунку.

Таблиця 5.1 – Перелік загальних впливів

№	Опис	Операція
1	Списання ОЕ за активацію карти	Гравець.ОЕ -= Гравець.Карта.ОЕА
2	Зарахування ОЕ за відправлену карту "з руки" в талон фазі "Активація карт"	Гравець.ОЕ += Гравець.Карта.ОЕА / 2, якщо (Гравець.Карта.ОЕА % 2) == 0 => Гравець.ОЕ++
3	Зарахування ОНВ за неактивність у фазі "Активація Карт"	Гравець.ОНВ++
4	Атака суперника "Воїном" гравця	Суперник.ОЗ -= Гравець.Воїн.ОП - min(Суперник.ОБ, 2), якщо Гравець.Воїн.ОП > min(Суперник.ОБ, 2)
5	Атака "Воїна" суперника "Воїном" гравця	Суперник.Воїн.ОЗ -= Гравець.Воїн.ОП - Суперник.Воїн.ОБ, якщо Гравець.Воїн.ОП > Суперник.Воїн.ОБ
6	Зарахування ОБ "Воїнам", які не атакували в фазі "Битва Армій"	Гравець.Воїн.ОБ++
7	Списання зарахованих ОБ за неактивність "Воїнам" у фазі "Завершення кола"	Гравець.Воїн.ОБ--
8	Списання ОЗ гравця через дію ОНВ у фазі "Завершення Кола"	Гравець.ОЗ -= Гравець.ОНВ
9	Зарахування ОЕ у фазі "Завершення Кола"	Гравець.ОЕ++
10	Зарахування ОПС у фазі "Завершення Кола"	Гравець.ОПС++

В результаті мануальної перевірки вказаних загальних впливів було визначено, що всі впливи, окрім 6 та 7 були успішно реалізовані та працюють.

В таблицях 5.2 представимо здібності, аналогічно до того, як це було зроблено для загальних впливів. Відмінність полягатиме в тому, що замість опису будуть використані ідентифікатори з таблиці Г.1, також таблиця матиме кілька додаткових полів з огляду на рішення прийняті в третьому та четвертому розділах. Також значення полів рядків будуть описані в термінології реалізованої системи для полегшення тестування.

Таблиця 5.2 – Перелік реалізованих здібностей

ID	Тип карти	Оборотність	Стан здійснення впливу	Стан відміни впливу	Об'єкт	Характеристика	Формула	Об'єкт Підсилюючої Характеристики	Підсилююча Характеристика
1	Оточення	Irreversible	CAR_Slot	CAR_None	Opponent	Health	SubtractionWithAmplification	Self	Arcane
2	Оточення	Irreversible	CAR_Slot	CAR_None	Self	Armor	Addition		
3	Оточення	Irreversible	CAR_Slot	CAR_None	Self	Negatory	Assignment		
4	Оточення	Irreversible	CAR_Slot	CAR_None	Self	Arcane	Multiplication		
5	Оточення	Irreversible	CAR_Slot	CAR_None	Self	Health	Addition		
6	Оточення	Irreversible	CAR_Slot	CAR_None	Self	Energy	Multiplication		
7	Оточення	Irreversible	CAR_Slot	CAR_None	Self	Energy	AdditionWithAmplification	Self	Arcane
8	Оточення	Irreversible	CAR_Slot	CAR_None	Opponent	Armor	Assignment		
9	Оточення	Reversible	CAR_Slot	CAR_None	SelfWarriors	Damage	Addition		
10	Оточення	Reversible	CAR_Slot	CAR_None	OpponentWarriors	Armor	Subtraction		
11	Воїн	Irreversible	ABR_PreAction	CAR_None	Self		Extra		
12	Воїн	Reversible	ABR_PreAction	ABR_PostAction	SelfWarrior	Damage	AdditionWithAmplification	Opponent	Negatory
13	Воїн	Irreversible	ABR_PreAction	ABR_PostAction	SelfWarriorsExclusion	Health	Addition		
14	Воїн	Reversible	CAR_Slot	CAR_None	OpponentWarriors	Damage	Subtraction		
15	Воїн	Reversible	CAR_Slot	CAR_None	SelfWarriorsExclusion	Armor	Addition		

Продовження Таблиці 5.2

16	Воїн	Reversible	ABR_PreAction	ABR_PostAction	SelfWarrior	Damage	AdditionWithAmplification	OpponentWarrior	Armor
17	Воїн	Reversible	CAR_Slot	CAR_None	SelfWarriorsExclusion	Health	Addition		
18	Воїн	Irreversible	CR_EnergyConsumption	CR_EnergyDonation	Self	Energy	Subtraction		
19	Воїн	Irreversible	CR_EnergyDonation	CR_EnergyDonation	Self	Armor	Addition		
20	Воїн	Irreversible	CR_EnergyDonation	CR_EnergyDonation	Self	Arcane	Addition		

В результаті мануальної перевірки функціонування реалізованих здібностей усі 20 здібностей працювали відповідно до правил гри.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є комплекс заходів для реалізації та реалізація демонстраційної версії ЦККГ «Світ Заару» на основі ігрового рушія Unreal Engine 5.

На етапах аналізу предметної галузі та формування вимог були проаналізовані деякі представники сучасних ЦККГ: визначені їхні спільні та відмінні риси, недоліки та переваги. Далі були проаналізовані правила ККГ «Світ Заару» та порівняні з висновками, отриманими на попередній ітерації. Як наслідок, була сформована концепція подальшої розробки та обмежень.

На етапі проєктування програмного забезпечення правила ККГ «Світ Заару» були формалізовані та перетворені в набір UML діаграм, на основі яких був написаний застосунок. Варто вказати на зручність та ефективність підходу втілення логіки роботи правил через модель скінченного автомату. Це дозволило наглядно представити текстовий опис загального ходу гри у вигляді чотирьох UML діаграм станів.

Демонстраційна версія ЦККГ «Світ Заару» реалізовує базові механіки та правила оригінальної гри, дає можливість двокористувацької гри на одному ПК. Включає в себе колоду з 30 унікальних карт, 20 з яких мають здібності. Гравець грає гральною колодою, що випадковим чином формується із зазначених раніше 30 карт.

В процесі виконання роботи були здобуті навички проєктування програмного забезпечення, створення програмної архітектури та висвітлення її в формі набору взаємозв’язаних UML діаграм. Важливою частиною роботи стала практика у формалізації правил та механік гри, що дозволило більш ефективно сформувати програмну архітектуру.

Також в процесі виконання роботи був отриманий досвід візуального програмування на мові Blueprint рушія Unreal Engine 5 та роботи з бібліотекою інструментів для створення ЦККГ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Колекційна карткова гра [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%B5%D0%BA%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%BA%D0%B0%D1%80%D1%82%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B3%D1%80%D0%B0 (дата звернення: 30.05.2024)

2. Exploring the World of Collectible Card Games: A Beginner's Guide [Електронний ресурс] – URL: https://clarkfranklin.medium.com/exploring-the-world-of-collectible-card-games-a-beginners-guide-4ad45a9aeb75 (дата звернення: 30.05.2024)

3. Collectible card game [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/Collectible_card_game (дата звернення: 03.06.2024)

4. Unreal Engine 5.4 Documentation [Електронний ресурс] – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-4-documentation (дата звернення: 30.05.2024)

5. Introduction to Blueprints [Електронний ресурс] – URL: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/ (дата звернення: 30.05.2024)

6. Gwen [Електронний ресурс] – URL: https://www.playgwent.com/en/ (дата звернення: 30.05.2024)

7. Magic: The Gathering Online [Електронний ресурс] – URL: https://www.mtgo.com/en/mtgo (дата звернення: 30.05.2024)

8. Genius Invokation TCG [Електронний ресурс] – URL: https://genshin-impact.fandom.com/wiki/Genius_Invokation_TCG (дата звернення: 30.05.2024)

9. A BEGINNER'S GUIDE TO GWENT – [Електронний ресурс] – URL: https://web.archive.org/web/20201026174535/https://www.playgwent.com/en/news/19701/a-beginners-guide-to-gwent (дата звернення: 31.05.2024)

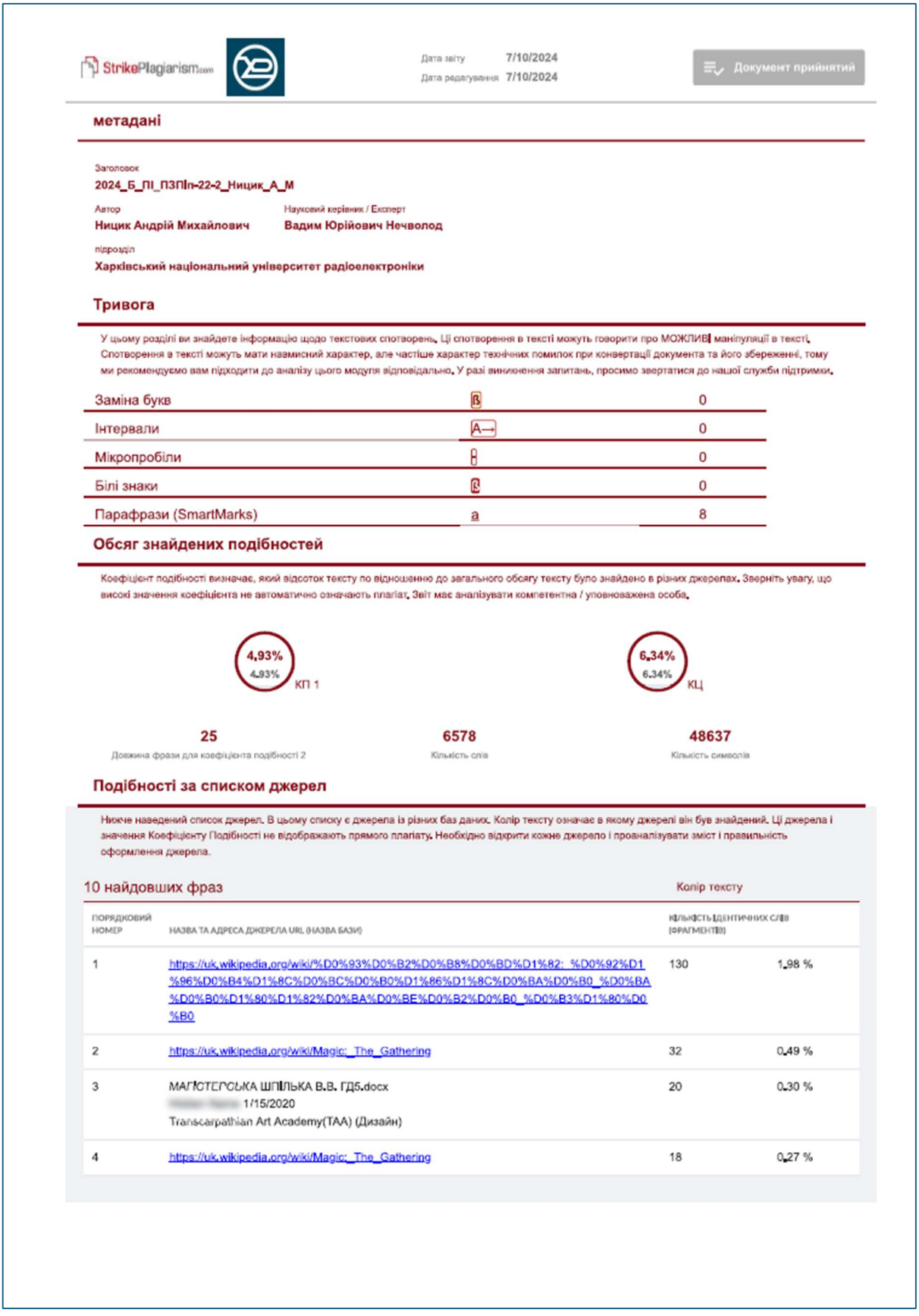
10. Magic: The Gathering – [Електронний ресурс] – URL: https://magic.wizards.com/en (дата звернення: 01.06.2024)

11. The Unified Modeling Language – [Електронний ресурс] – URL: https://www.uml-diagrams.org/ (дата звернення: 05.06.2024)

12. CCG Toolkit – [Електронний ресурс] – URL: https://www.unrealengine.com/marketplace/en-US/product/ccg-toolkit (дата звернення: 05.07.2024)

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/10/2024
Дата редагування 7/10/2024

Документ прийнятий

метадані

Заголовок
2024_Б_ПІ_ПЗПІп-22-2_Ницик_А_М

Автор
Ницик Андрій Михайлович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	8

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

4,93% 4,93% КП 1

6,34% 6,34% КЦ

25	6578	48637
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://uk.wikipedia.org/wiki/%D0%93%D0%B2%D0%B8%D0%BD%D1%82:_%D0%92%D1%86%D0%B4%D1%8C%D0%BC%D0%B0%D1%86%D1%8C%D0%BA%D0%B0_%D0%BA%D0%B0%D1%80%D1%82%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B3%D1%80%D0%B0	130	1,98 %
2	https://uk.wikipedia.org/wiki/Magic:_The_Gathering	32	0,49 %
3	МАГІСТЕРСЬКА ШПИЛЬКА В.В. ГД5.docx [redacted] 1/15/2020 Transcarpathian Art Academy(TAA) (Дизайн)	20	0,30 %
4	https://uk.wikipedia.org/wiki/Magic:_The_Gathering	18	0,27 %

5	МАГІСТЕРСЬКА ШПІЛЬКА В.В. ГД5.docx [redacted] 1/15/2020 Transcarpathian Art Academy(TAA) (Дизайн)	13	0.20 %
6	https://uk.wikipedia.org/wiki/%D0%93%D0%B2%D0%B8%D0%BD%D1%82:_%D0%92%D1%96%D0%B4%D1%8C%D0%BC%D0%B0%D1%86%D1%8C%D0%BA%D0%B0_%D0%BA%D0%B0%D1%80%D1%82%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B3%D1%80%D0%B0	12	0.18 %
7	https://uk.wikipedia.org/wiki/Magic:_The_Gathering	10	0.15 %
8	https://uk.wikipedia.org/wiki/Magic:_The_Gathering	10	0.15 %
9	https://uk.wikipedia.org/wiki/%D0%93%D0%B2%D0%B8%D0%BD%D1%82:_%D0%92%D1%96%D0%B4%D1%8C%D0%BC%D0%B0%D1%86%D1%8C%D0%BA%D0%B0_%D0%BA%D0%B0%D1%80%D1%82%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B3%D1%80%D0%B0	10	0.15 %
10	https://uk.wikipedia.org/wiki/%D0%93%D0%B2%D0%B8%D0%BD%D1%82:_%D0%92%D1%96%D0%B4%D1%8C%D0%BC%D0%B0%D1%86%D1%8C%D0%BA%D0%B0_%D0%BA%D0%B0%D1%80%D1%82%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B3%D1%80%D0%B0	9	0.14 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.50 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	МАГІСТЕРСЬКА ШПІЛЬКА В.В. ГД5.docx [redacted] 1/15/2020 Transcarpathian Art Academy(TAA) (Дизайн)	33 (2)	0.50 %

з Інтернету (4.42 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://uk.wikipedia.org/wiki/%D0%93%D0%B2%D0%B8%D0%BD%D1%82:_%D0%92%D1%96%D0%B4%D1%8C%D0%BC%D0%B0%D1%86%D1%8C%D0%BA%D0%B0_%D0%BA%D0%B0%D1%80%D1%82%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B3%D1%80%D0%B0	207 (11)	3.15 %
2	https://uk.wikipedia.org/wiki/Magic:_The_Gathering	84 (6)	1.28 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

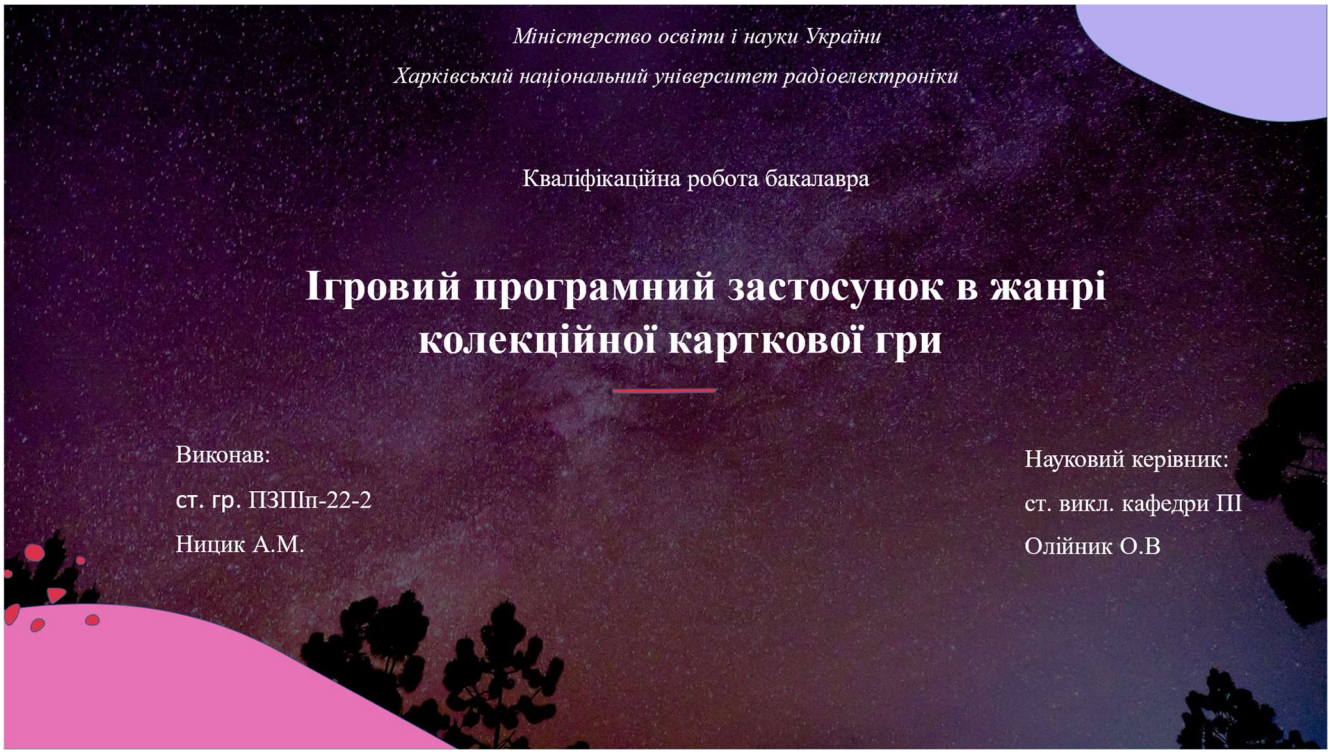
8

ВСТУП

Темою кваліфікаційної роботи є створення ігрового програмного застосунку

ДОДАТОК Б

Слайди презентації



Гра «Світ Заару»

«Світ Заару» – фентезійна ККГ, де гравці є воєводами, що командують своїми арміями та накладають різноманітні впливи за допомогою спеціальних заклять. Ціль партії – залишитися єдиним воєводою у грі з очками здоров'я більшими за 0.

«Світ Заару» дозволяє грати 2 – 4 гравцям одночасно. Кожен раунд гри включає 3 фази, у 2-х з них гравці беруть активну участь. Гра має 5 числових характеристик, які впливають на різні аспекти гри, а відповідно на силу гравців та тактику їхніх дій. Залежно від кількості гравців у партії, кожен з них отримує по 15, 30, 50 карт у гральну колоду.

Карти дій діляться на 2 типи:

- оточення
- воїни

Постановка мети

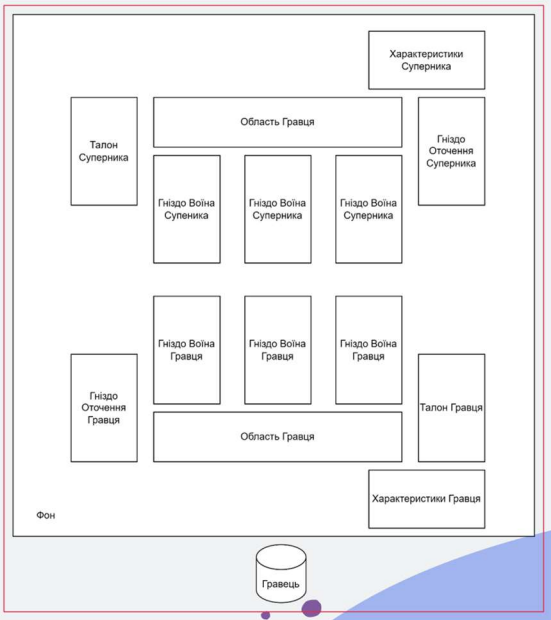
Метою виконання роботи є створення програмного застосунку, що є демонстраційною версією ЦККГ «Світу Заару», якого достатньо для розуміння базових принципів та механік гри загалом.

Демонстраційна версія обмежується наступною функціональністю:

- відсутність можливості колекціонування, збирання карт;
- базові механіки та правила;
- двокористувацька гру на одному ПК;
- 30 унікальних карт;
- 20 унікальних здібностей;
- гра випадковою гральною колодою.

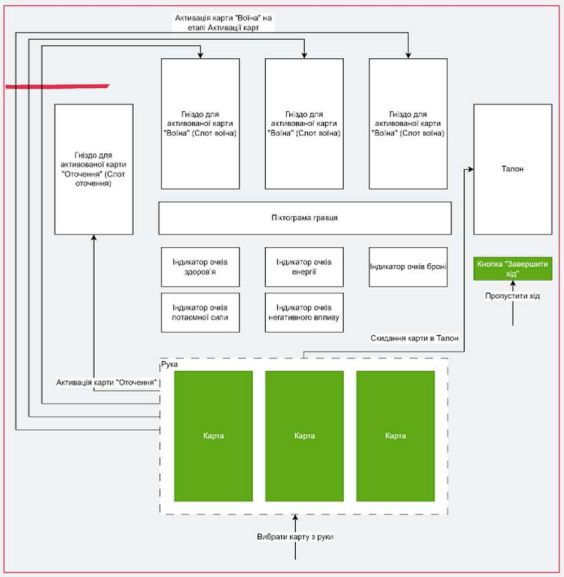
У якості середовища розробки та ігрового рушія була вибрана Unreal Engine 5. Ігровий застосунок повинен працювати на ОС MS Windows 10/ MS Windows 11.

Схематичний вигляд елементів GUI

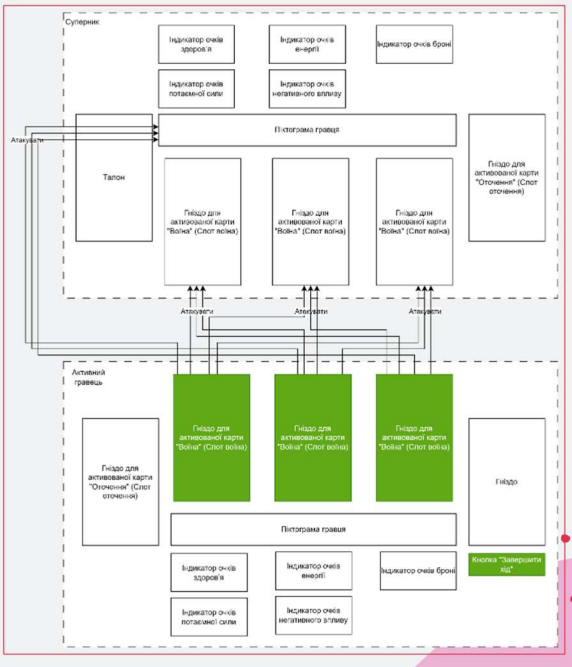


Проєктування графічної частини застосунку

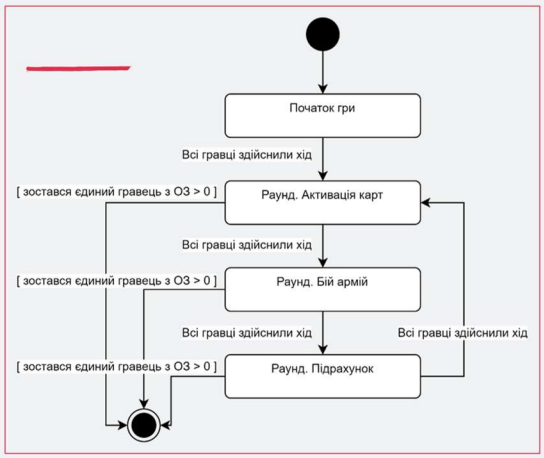
Фаза «Активація карт»



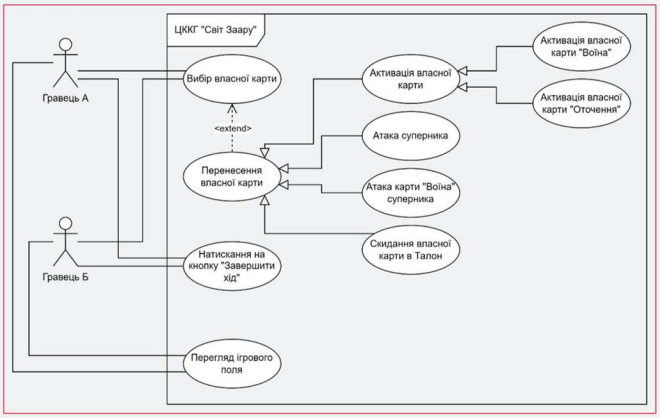
Фаза «Битва Армій»



Проєктування логічної частини застосунку



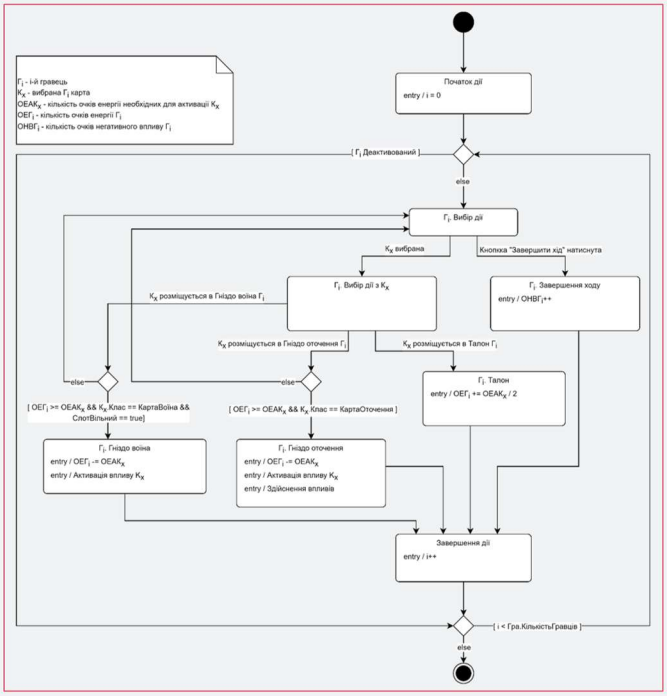
Загальна UML діаграм станів



UML діаграма використання

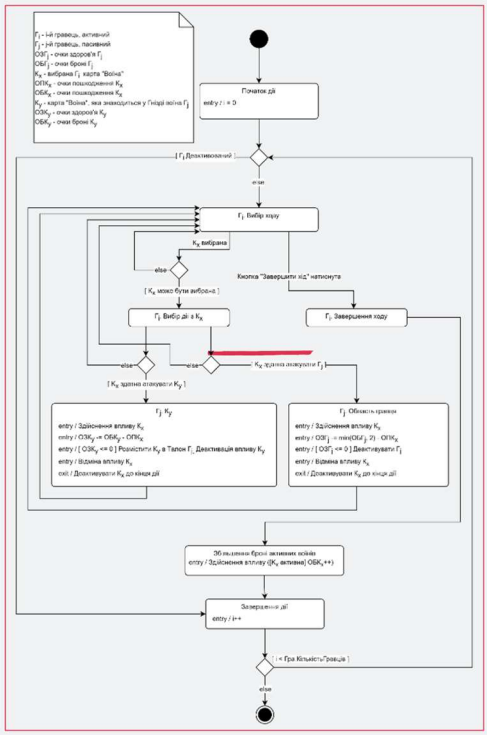
Проєктування логічної частини застосунку

UML діаграм станів для фази «Активація Карт»



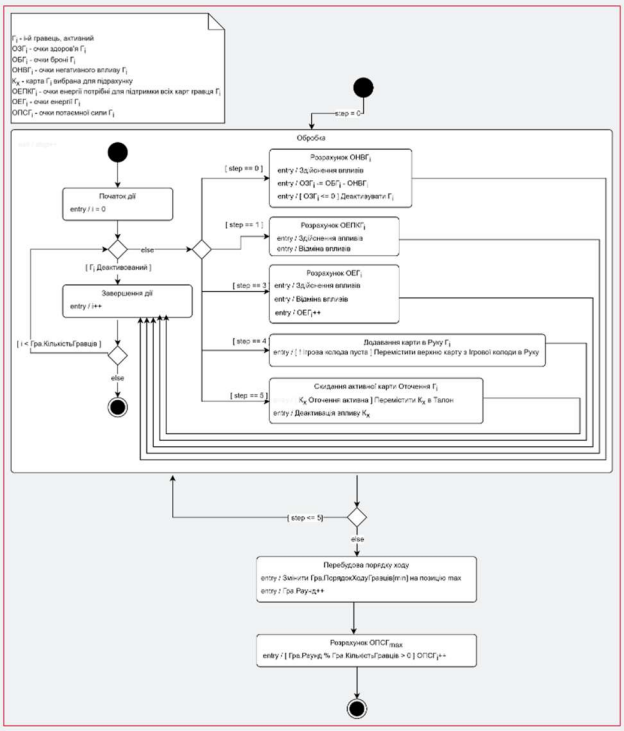
Проєктування логічної частини застосунку

UML діаграм станів для фази «Битва Армій»



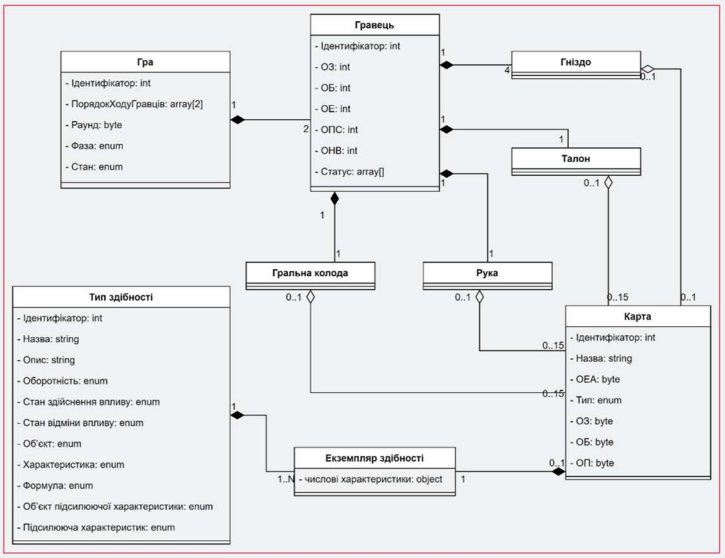
Проєктування логічної частини застосунку

UML діаграм станів для фази «Завершення Кола»



Проєктування логічної частини застосунку

Загальна UML діаграм класів



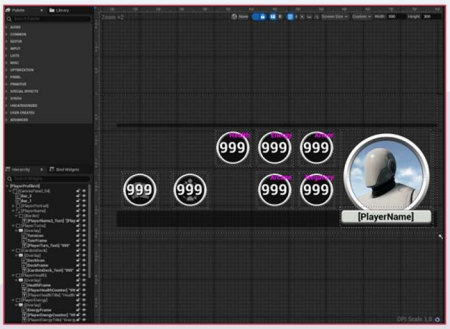
Набір інструментів для створенні цифрових колекційних карткових ігор «CCG Toolkit»

«CCG Toolkit» – це гнучкий та потужний фреймворк для розробки цифрових колекційних карткових ігор. Фреймворк включає в себе широкий набір інструментів для опису та реалізації різноманітних аспектів та правил. З поміж іншого він дозволяє створювати власні карти, ігрові поля, здібності. Фреймворк написаний на мові візуального програмування Blueprint.

Фреймворк включає в себе наступні готові для використання та модифікації елементи:

- ігрові поля;
- карти;
- здібності;
- елементарні механіки гри.

Реалізація графічної частини застосунку



Реалізація логічної частини застосунку

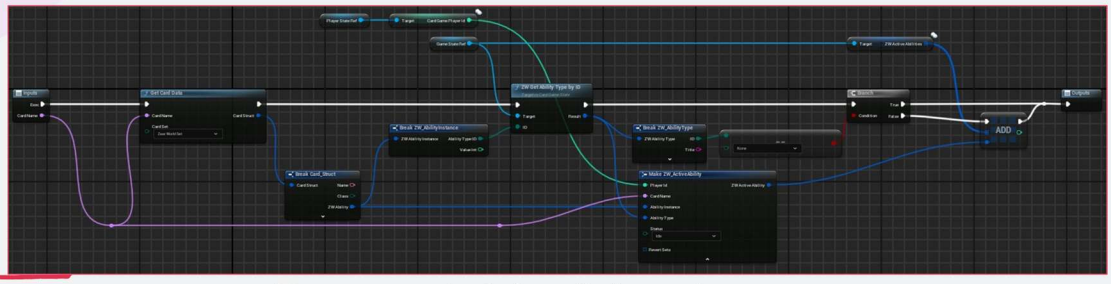
Data Table | Data Table Details

Search

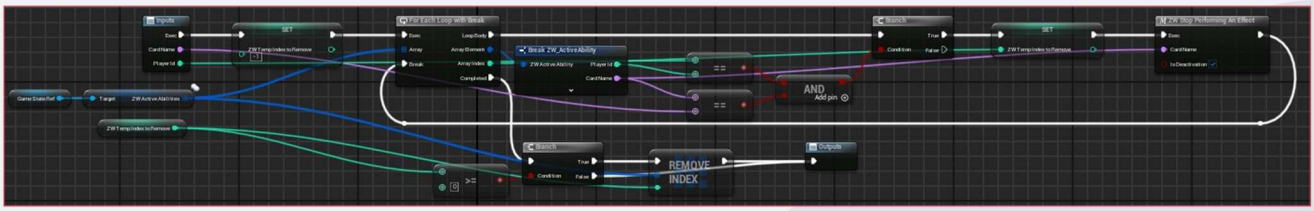
	Ro	ID	Title	Description	Reversibility	StartPerformingState	StopPerformingStat	Object	Feature	Formula	AmplificationObject	AmplificationFeature
1	Er	Environment_1	The opponent	When activated,	Irreversible	CAR_Slot	CAR_None	Opponent	Health	SubtractionWithAmplificatior	Self	Arcane
2	Er	Environment_2	Increase playe	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Armor	Addition	Self	None
3	Er	Environment_3	Removes all th	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Negatory	Assignment	Self	None
4	Er	Environment_4	Increases play	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Arcane	Multiplication	Self	None
5	Er	Environment_5	Restores playe	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Health	Addition	Self	None
6	Er	Environment_6	Increases play	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Energy	Multiplication	Self	None
7	Er	Environment_7	The player get	When activated,	Irreversible	CAR_Slot	CAR_None	Self	Energy	AdditionWithAmplification	Self	Arcane
8	Er	Environment_8	Deprives the o	When activated,	Irreversible	CAR_Slot	CAR_None	Opponent	Armor	Assignment	Self	None
9	Er	Environment_9	Increases all p	When activated,	Reversible	CAR_Slot	CAR_None	SelfWarriors	Damage	Addition	Self	None
10	Er	Environment_10	Decreases all (	When activated,	Reversible	CAR_Slot	CAR_None	OpponentWarriors	Armor	Subtraction	Self	None
11	W	Warrior_1	Can damage t	This warrior can	Irreversible	ABR_PreAction	CAR_None	SelfWarrior	None	Extra	Self	None
12	W	Warrior_2	Additional dar	This warrior uses	Reversible	ABR_PreAction	ABR_PostAction	SelfWarrior	Damage	AdditionWithAmplification	Opponent	Negatory
13	W	Warrior_3	Restores X poi	This warrior resto	Irreversible	ABR_PreAction	ABR_PostAction	SelfWarriorsExclusion	Health	Addition	Self	None
14	W	Warrior_4	Reduces each	This warrior redu	Reversible	CAR_Slot	CAR_None	OpponentWarriors	Damage	Subtraction	Self	None
15	W	Warrior_5	Increases near	This warrior incr	Reversible	CAR_Slot	CAR_None	SelfWarriorsExclusion	Armor	Addition	Self	None
16	W	Warrior_6	Ignores oppon	This warrior igno	Reversible	ABR_PreAction	ABR_PostAction	SelfWarrior	Damage	AdditionWithAmplification	OpponentWarrior	Armor
17	W	Warrior_7	Increases near	This warrior incr	Reversible	CAR_Slot	CAR_None	SelfWarriorsExclusion	Health	Addition	Self	None
18	W	Warrior_8	Consumes the	This warrior cons	Irreversible	CR_EnergyConsumption	CAR_None	Self	Energy	Subtraction	Self	None
19	W	Warrior_9	Each cycle the	This warrior incr	Irreversible	CR_EnergyDonation	CR_EnergyDonation	Self	Armor	Addition	Self	None
20	W	Warrior_10	Each cycle the	This warrior incr	Irreversible	CR_EnergyDonation	CR_EnergyDonation	Self	Arcane	Addition	Self	None

Зміст контейнера даних ZW_Abilities

Реалізація логічної частини застосунку



Програмна реалізація функції «Активація впливу»



Програмна реалізація функції «Деактивація впливу»

Вигляд робочого GUI

Тестування

```
Server: PlayerId: 1 ZW State change attempt: CAR_None -> EndTurn -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 2 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 2 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 1 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 1 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 1 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 2 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 2 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 1 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> WarriorAttacks -> ABR_PreAction
Server: PlayerId: 1 ZW State change attempt: ABR_PreAction -> WarriorAttacks -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 1 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 2 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 1 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 2 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 1 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 2 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 1 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 2 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 1 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 2 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 1 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 1 ZW State change attempt: CAR_None -> EndTurn -> CAR_None
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 2 ZW State change attempt: CAR_None -> EndTurn -> CAR_None
Server: PlayerId: 1 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 1 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 1 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 2 ZW State change attempt: CAR_None -> StartTurn -> CAR_Preparation
Server: PlayerId: 2 ZW State change attempt: CAR_Preparation -> CardIsPlacedIntoSlot -> CAR_Slot
Server: PlayerId: 2 ZW State change attempt: CAR_Slot -> EndTurn -> CAR_Completion
Server: PlayerId: 1 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> WarriorAttacks -> ABR_PreAction
Server: PlayerId: 1 ZW State change attempt: ABR_PreAction -> WarriorAttacks -> ABR_Preparation
Server: PlayerId: 1 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 2 ZW State change attempt: CAR_Completion -> StartTurn -> ABR_Preparation
Server: PlayerId: 2 ZW State change attempt: ABR_Preparation -> EndTurn -> ABR_Completion
Server: PlayerId: 2 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 1 ZW State change attempt: ABR_Completion -> PerformCalculation -> CR_Negatory
Server: PlayerId: 2 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 1 ZW State change attempt: CR_Negatory -> PerformCalculation -> CR_EnergyConsumption
Server: PlayerId: 2 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 1 ZW State change attempt: CR_EnergyConsumption -> PerformCalculation -> CR_EnergyDonation
Server: PlayerId: 2 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 1 ZW State change attempt: CR_EnergyDonation -> PerformCalculation -> CR_CardToHand
Server: PlayerId: 2 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 1 ZW State change attempt: CR_CardToHand -> PerformCalculation -> CR_EnvironmentToGraveyard
Server: PlayerId: 2 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
Server: PlayerId: 1 ZW State change attempt: CR_EnvironmentToGraveyard -> PerformCalculation -> CAR_None
```

Приклад логів зміни станів скінченного автомату

Анімація атаки картою воїна суперника у вікні повного розміру

Висновки

Результатом виконання кваліфікаційної роботи є комплекс заходів для реалізації та реалізація демонстраційної версії ЦККГ «Світ Заару» на основі ігрового рушія Unreal Engine 5.

Було зроблено:

- проаналізовані деякі представники сучасних ЦККГ: визначені їхні спільні та відмінні риси, недоліки та переваги;
- сформовані концепція розробки та обмеження реалізації ЦККГ «Світ Заару»;
- формалізовані та перетворені в набір UML діаграм правила «Світ Заару»;
- написаний застосунок.

Демонстраційна версія ЦККГ «Світ Заару» реалізовує базові механіки та правила оригінальної гри, дає можливість двокористувацької гри на одному ПК. Включає в себе колоду з 30 унікальних карт, 20 з яких мають здібності. Гравець грає гральною колодою, що випадковим чином формується із зазначених раніше 30 карт.

В процесі виконання роботи були здобуті навички проєктування програмного забезпечення, створення програмної архітектури та висвітлення її в формі набору взаємозв'язаних UML діаграм. Важливою частиною роботи стала практика у формалізації правил та механік гри, що дозволило більш ефективно сформувати програмну архітектуру.

ДОДАТОК В

Приклади програмного коду

```
// ACardGamePlayerController.h

UCLASS(Blueprintable, BlueprintType)
class ACardGamePlayerController : public APlayerController
{
	GENERATED_BODY()
public:
	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void SetCardLocation(A3DCard_C* Card, FVector HoldLocation, FRotator Rotation);

	UFUNCTION(BlueprintCallable, Category="Card Interaction Functions")
	void DetectCardInteraction();

	UFUNCTION(BlueprintCallable, Category="Card Interaction Functions")
	void ValidateInteractionState(bool& ValidInteraction);

	UFUNCTION(BlueprintCallable, Category="Card Interaction Functions")
	void DetectInteractionOnMove();

	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void ValidateCardPlacement(AActor* HitActor, bool& ValidPlacement);

	UFUNCTION(BlueprintCallable, Category="Game Setup")
	void SetupGameUI();

	UFUNCTION(BlueprintCallable, Category="Game Setup")
	void SetupDeck(FString DeckName, UPARAM(ref) TArray<FName>& PlayerDeck);

	UFUNCTION(BlueprintCallable, Category="Deck / Hand  Functions")
	void FilterWeightedCards();

	UFUNCTION(BlueprintCallable, Category="Game Setup")
	void SetTimer(int32 Time);

	UFUNCTION(BlueprintCallable, Category="Deck / Hand  Functions")
	void ShufflePlayerDeck(const TArray<FName>& TargetArray);

	UFUNCTION(BlueprintCallable, Category="Card Interaction Functions")
	void RunCardInteraction(A3DCard_C* Interaction_TalkingCard, A3DCard_C* Interacition_RecievingCard,
ABoardPlayer_C* Interaction Recieving Player, A3DCard_C* InteracitionRecievingCard, A3DCard_C* InteractionTalkingCard,
ABoardPlayer_C* InteractionRecievingPlayer);

	UFUNCTION(BlueprintCallable, Category="Game")
	void NotifyCardTurnActive();

	UFUNCTION(BlueprintCallable, Category="Game")
	void NotifyCardsEndTurn();

	UFUNCTION(BlueprintCallable, Category="Card Interaction Functions")
	void CreateDragActorVisual(bool UseActorLocation?, bool& Valid);

	UFUNCTION(BlueprintCallable, Category="Deck / Hand  Functions")
	void AddCardToHand_OLD(const FName& CardToAdd, TEnumAsByte<CardSet_Enum> FromCardSet);

	UFUNCTION(BlueprintCallable, Category="Deck / Hand  Functions")
	void RemoveCardFromHand_OLD(bool ClearAll, FName CardName, int32 IndexToRemove, bool& Success);

	UFUNCTION(BlueprintPure, Category="Deck / Hand  Functions")
	void GetCardInHand_OLD(int32 Index, bool LastIndex, FName& Item);
```

```
	UFUNCTION(BlueprintPure, Category="Deck / Hand  Functions")
	int32 CountCardsInHand_OLD();

	UFUNCTION(BlueprintPure, Category="Deck / Hand  Functions")
	int32 CountCardsInDeck_OLD();

	UFUNCTION(BlueprintPure, Category="Deck / Hand  Functions")
	void HasCardInHand_OLD(FName CardName, int32 Index, bool& HasCard);

	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void CreatePlaceableCard(Client)(FName Name, TEnumAsByte<CardSet_Enum> CardSet, FVector Spawn Transform
Location, A3DCard_C*& Card Created, FTransform TEMP_SpawnTransform);

	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void CreatePlaceableCard(Server)(FTransform SpawnTransform, A3DCard_C*& Card Created, FTransform
TEMP_SpawnTransform);

	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void ServerValidateCardPlacement(ACardPlacement_C* CardPlacement, FCard_Struct CardStruct, bool& ValidPlacement,
ACardPlacement_C*& ValidCardPlacement);

	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void SetCustomCardData(A3DCard_C* Card, bool ActiveAbility, A3DCard_C*& ReturnCard);

	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void PlayCard(FName CardName, TEnumAsByte<CardSet_Enum> CardSet, ACardPlacement_C* CardPlacement, int32
CardHandIndex, FTransform SpawnTransform, bool& IsCardPlayed);

	UFUNCTION(BlueprintCallable)
	void OnRep_bTurnActive();

	UFUNCTION(BlueprintCallable, Category="Game Setup")
	void LoadClientDeck(FString& DeckName, TArray<FName>& Deck);

	UFUNCTION(BlueprintCallable)
	void OnRep_TurnState();

	UFUNCTION(BlueprintCallable, Category="ZW State Machine")
	void ZW_StateMachineLogUnhandledEvent(TEnumAsByte<ZW_SMEventTypes> Event, TEnumAsByte<ZW_SMStates>
State);

	UFUNCTION(BlueprintCallable, Category="Card Functions")
	void ZW_PlayToGraveyard(FName CardName, TEnumAsByte<CardSet_Enum> CardSet, int32 CardHandIndex, FTransform
SpawnTransform, bool& IsCardPlayed);

	UFUNCTION(BlueprintCallable, Category="ZW_StateMachine")
	void ZW_StateMachineLogStateChangeAttempt(TEnumAsByte<ZW_SMStates> PreviousState, FZW_SMEvent Event);

	UFUNCTION(BlueprintCallable, Category="ZW_StateMachine")
	void ZW_StateMachineDecisionWrapper(FZW_SMEvent Event, TEnumAsByte<ZW_SMStates> PreviousState);

	UFUNCTION(BlueprintCallable, Category="ZW_StateMachine")
	void ZW_DetermineEntranceMessage(FString& result, FString firstPart, FString secondPart);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityPlayerGetFeatureValue(TEnumAsByte<ZW_AbilityObject> Object, TEnumAsByte<ZW_AbilityFeature>
Feature, int32& Result);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityPlayerSetFeatureValue(TEnumAsByte<ZW_AbilityObject> Object, TEnumAsByte<ZW_AbilityFeature>
Feature, int32 Value, int32& Result);

	UFUNCTION(BlueprintCallable, Category="ZW_StateMachine")
	void ZW_ApplyInactivityPenalty(int32 PlayerID, int32& Result);

	/** Please add a function description */
	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
```

```
	void ZW_AbilityCardGetFeatureValue(TEnumAsByte<ZW_AbilityObject> Object, FName CardName, FName TargetCardName, TEnumAsByte<ZW_AbilityFeature> Feature, int32& Result);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityCardSetFeatureValue(TEnumAsByte<ZW_AbilityObject> Object, FName CardName, FName TargetCardName, TEnumAsByte<ZW_AbilityFeature> Feature, int32 Value, int32& Result);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityIsConditionAccepted(FName CardName, TEnumAsByte<ZW_AbilityConditionStatus> Status, FZW_ActiveAbility ActiveAbility, bool& Result);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityHandleFormula(TEnumAsByte<ZW_AbilityFormula> Formula, int32 FeatureValue, int32 Value, int32 AmplificationValue, int32& Result, int32& VatueToRevert);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityHandleRevertFormula(TEnumAsByte<ZW_AbilityFormula> Formula, int32 FeatureValue, int32 ValueToRevert, int32& Result);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityCanWarriorAttackPlayer(FName CardName, bool& Result);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW_AbilityLogUnhandledObject(FString Who, TEnumAsByte<ZW_AbilityObject> Object);

	UFUNCTION(BlueprintCallable, Category="ZW_Ability")
	void ZW Deactivate An Effect on Moving to Graveyard(FName CardName);
public:
	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, HitCard) to GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction", Replicated)
	TObjectPtr<A3DCard_C> HitCard;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction")
	TObjectPtr<ABoardPlayer_C> HitPlayer;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction")
	TObjectPtr<AActor> HitActor;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction")
	TObjectPtr<A3DCard_C> TalkingCard ;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction")
	TObjectPtr<A3DCard_C> RecievingCard;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction")
	TObjectPtr<ABoardPlayer_C> RecievingPlayer;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, TargetDragSelectionActor) to GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction", Replicated)
	TObjectPtr<ATargetDragSelection_C> TargetDragSelectionActor;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, bIsDragging) to GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction", Replicated)
	bool bIsDragging;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, bIsCardSelected) to GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction", Replicated)
	bool bIsCardSelected;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Player")
	TEnumAsByte<PlayerState_Enum> PlayerStateEnum;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	TObjectPtr<A3DCard_C> PlayCard_Server_Ref;
```

```
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Mobile")
	bool bEnabledMobileCardPreview;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="System")
	TObjectPtr<ACardGameState_C> GameState Ref;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	TObjectPtr<A3DCard_C> PlayCard_Client_Ref;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	TObjectPtr<A3DCard_C> Card on Board Ref;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Mobile")
	TObjectPtr<A3DCard_C> EnabledMobilePreview;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Deck")
	TArray<FName> TempDeck;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, PlayerDeck) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Deck", Replicated)
	TArray<FName> PlayerDeck;

	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Deck")
	bool bShuffleDeck;

	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Deck")
	bool bEnableWeightedCards;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Deck")
	int32 WeightedFilterIndex;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	FName Temp_CreateCardName;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	TEnumAsByte<CardSet_Enum> Temp_ChosenCardSet;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	TObjectPtr<UDragDropOperation> Temp_DragDropOperation;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	bool bTemp_CardIsClone;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, Temp_Location) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables", Replicated)
	FVector Temp_Location;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	TObjectPtr<ACardPlacement_C> TargetCardPlacement;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	bool ZW_IsTargetCardPlacementGraveyard;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	TObjectPtr<ACardPlacement_C> CardPlacement_Ref;

	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Card Placement")
	bool bEnableInHandMovementRotation;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	bool bIsValidClientDrop;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, bPlayCardSuccess) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement", Replicated)
```

```
	bool bPlayCardSuccess;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	bool bIsPlayed;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game Interaction")
	bool bValidInteraction;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="System")
	TObjectPtr<ACardGamePlayerState_C> PlayerState_Ref;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, CardSet) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Player", Replicated)
	TEnumAsByte<CardSet_Enum> CardSet;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, bTurnActive) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Player", ReplicatedUsing="OnRep_bTurnActive")
	bool bTurnActive;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Game", meta=(EditInline="true"))
	TObjectPtr<UCountdown_C> CountdownTimer_Widget;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Manager")
	TArray<UCardWidget_C*> CardSelection;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, CardsInHand) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Manager", Replicated)
	TArray<FName> CardsInHand;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, CardToAdd) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Manager", Replicated)
	FName CardToAdd;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, CardSet_Ref) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Manager", Replicated)
	TEnumAsByte<CardSet_Enum> CardSet_Ref;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Manager")
	int32 NumberOfCardsToAdd;


	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Player")
	int32 MaxCardsInHand;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="System")
	TEnumAsByte<Errors_Enum> Error_Enum;

	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Card Manager")
	double CardPickupDelay;

	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Player")
	int32 CardsInFirstHand;

	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Player")
	int32 CardsToDrawPerTurn;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, BoardPlayer_Ref) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Player", Replicated)
	TObjectPtr<ABoardPlayer_C> BoardPlayer_Ref;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, PlayerGameUI_Ref) to
GetLifetimeReplicatedProps");
```

```
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Player", Replicated, meta=(EditInline="true"))
	TObjectPtr<UPlayerGameUI_C> PlayerGameUI_Ref;

	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Card Placement")
	double CardHoldDistance;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, OpponentUI_Ref) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Player", Replicated, meta=(EditInline="true"))
	TObjectPtr<UOpponentUI_C> OpponentUI_Ref;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement")
	int32 CardInHandIndex_Ref;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, bSkipManaCheck) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Developer", Replicated)
	bool bSkipManaCheck;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	int32 TempHandIndex;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, TurnState) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="System", ReplicatedUsing="OnRep_TurnState")
	TEnumAsByte<GameTurn_Enum> TurnState;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Card Placement", meta=(EditInline="true"))
	TObjectPtr<UCardWidget_C> CardWidgetRef;

	static_assert(false, "You will need to add DOREPLIFETIME(ACardGamePlayerController, ZW_Extension) to
GetLifetimeReplicatedProps");
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="ZW State Machine", Replicated)
	FZW_CardGamePlayerControllerExtension ZW_Extension;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	FName ZW_TempRecievingCardName;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	int32 ZW_TempAmplificationFeatureValue;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	FName ZW_TempExcludeCardName;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	int32 ZW_TempIndexToRemove;

	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Temp Variables")
	TArray<FZW_AbilityRevert> ZW_TempRevertSets;
};
```

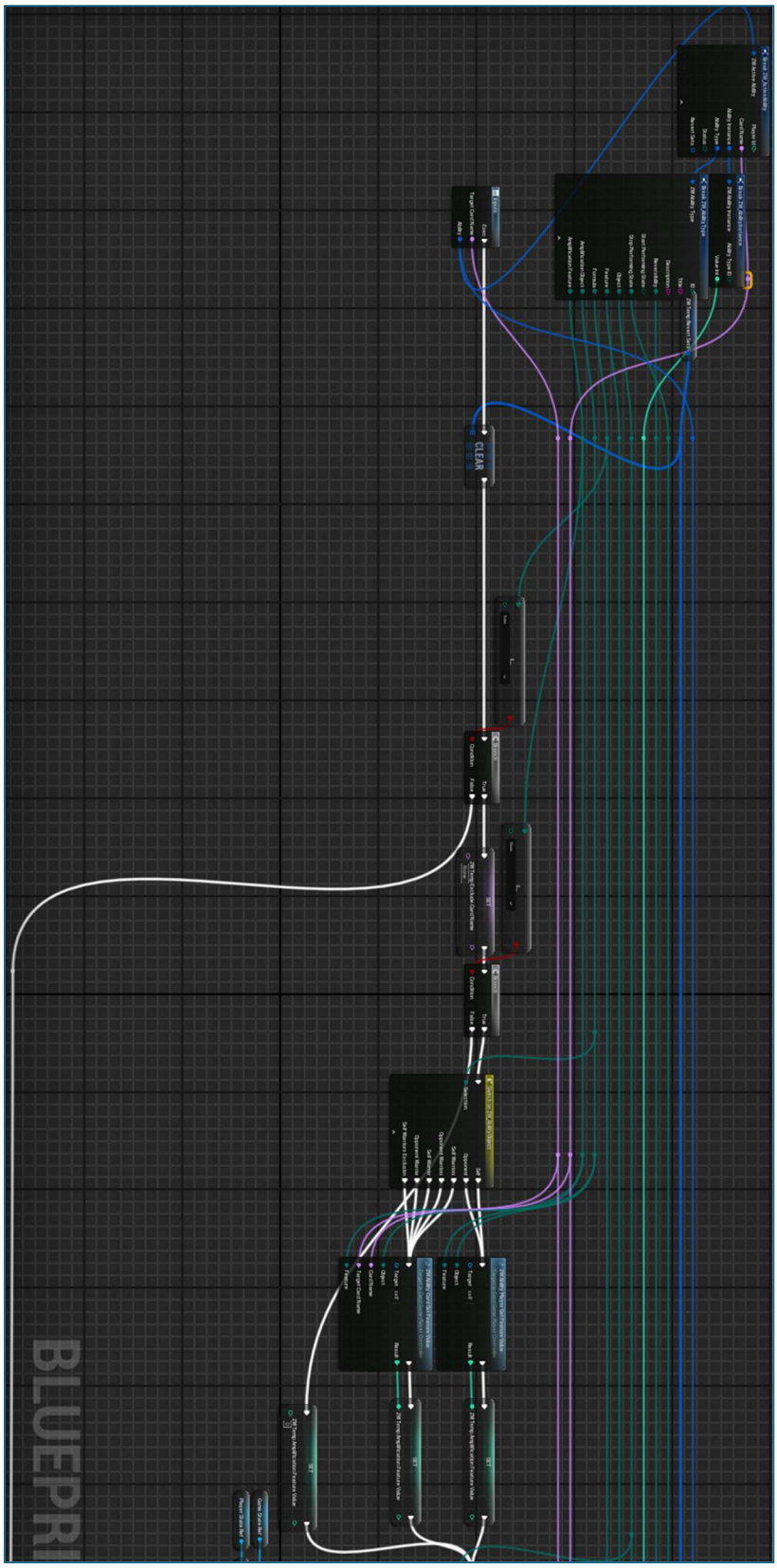


Рисунок В.1 – Реалізація функції ZW_StartPerformingAnEffectInner. Частина 1

(Рисунок виконаний самостійно)

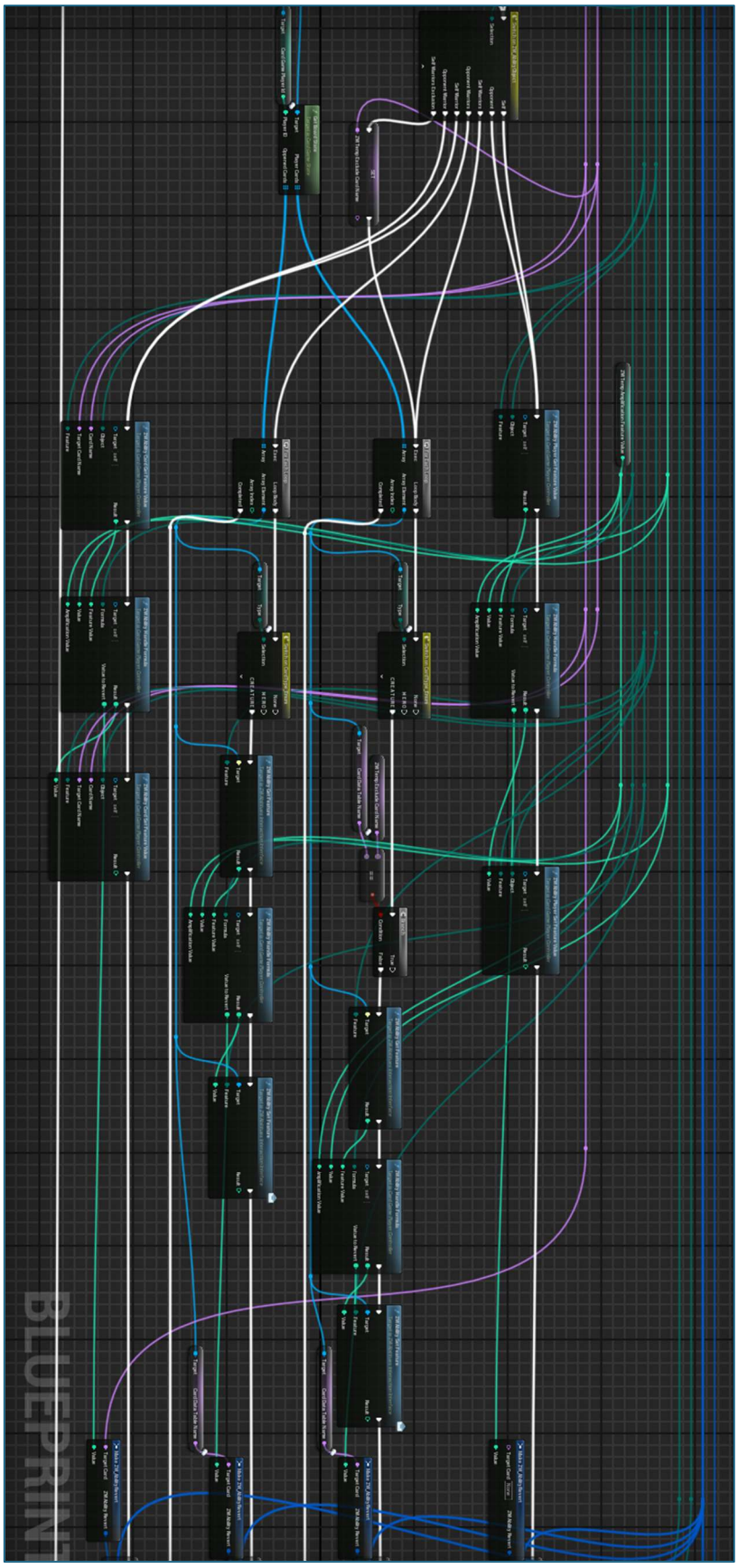


Рисунок В.2 – Реалізація функції ZW_StartPerformingAnEffectInner. Частина 2

(Рисунок виконаний самостійно)

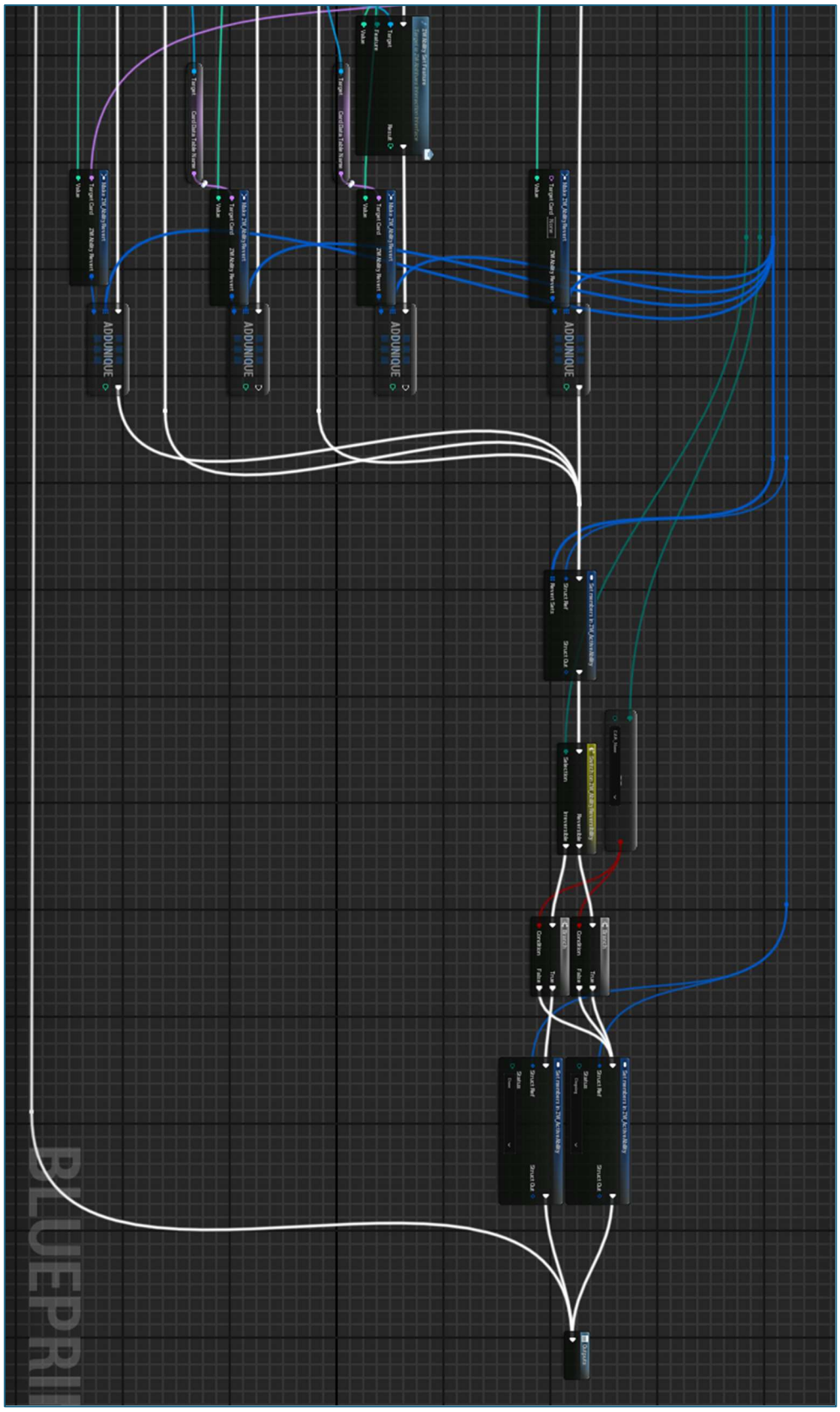


Рисунок В.3 – Реалізація функції ZW_StartPerformingAnEffectInner. Частина 3

(Рисунок виконаний самостійно)

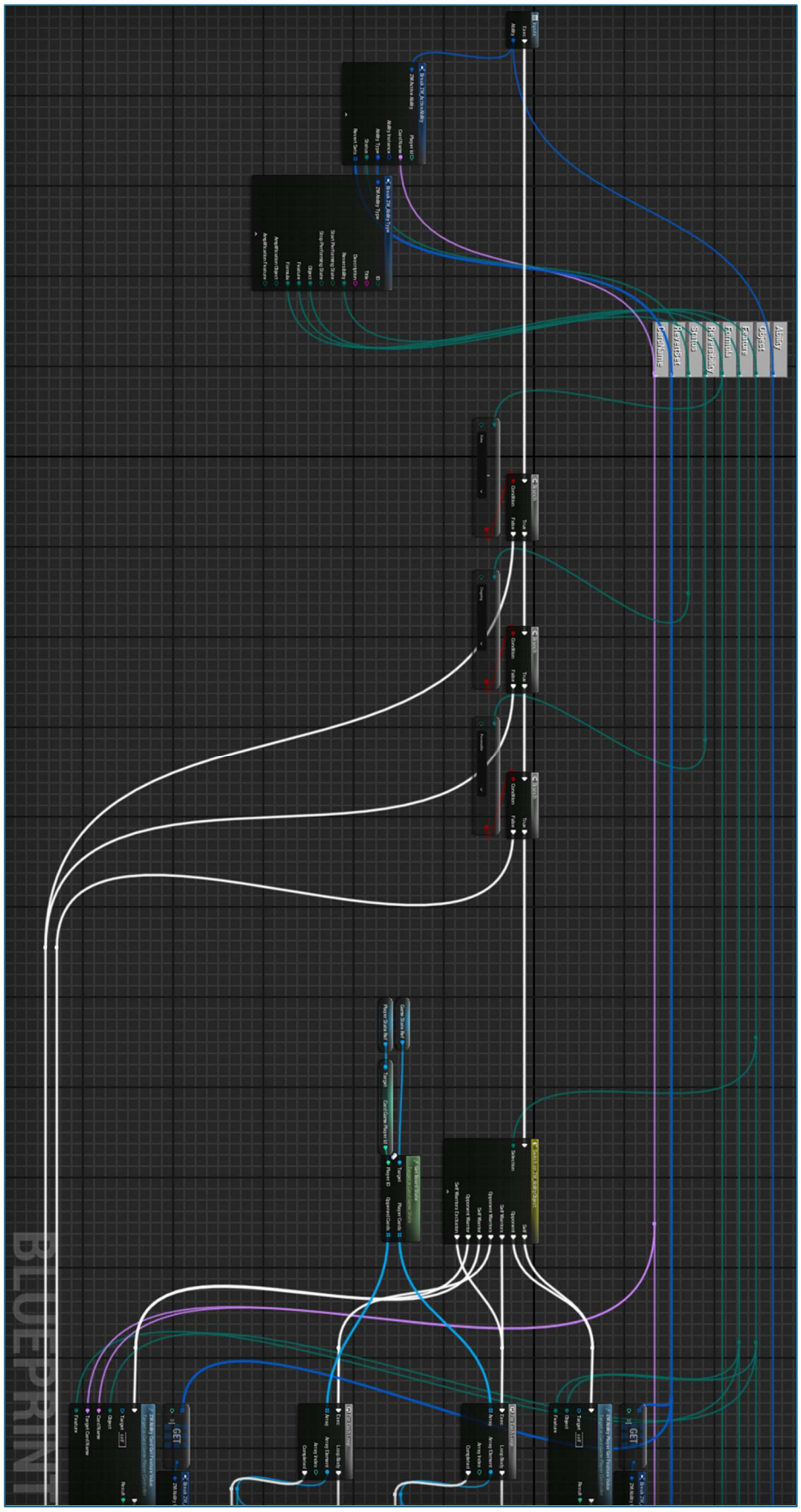


Рисунок В.4 – Реалізація функції ZW_StopPerformingAnEffectInner. Частина 1

(Рисунок виконаний самостійно)

Рисунок В.5 – Реалізація функції ZW_StopPerformingAnEffectInner. Частина 2

(Рисунок виконаний самостійно)

ДОДАТОК Г

Правила гри «Світ Заару»

«Світ Заару» – фентезійна настільної ККГ в однойменному вигаданому світі. Заар населений трьома расами: Ельфами, Велдами та Балікуру. Ельфійська раса зазнала релігійного розколу багато століть тому. Відтепер є Діти Павука Ночі та Вищі Ельфи. У Блакитних горах живуть Велди, які займаються видобутком руди, ремеслами і торгівлею. З маловивчених західних земель племена Балікуру прийшли на землі Заару.

Чотири ворогуючі сторони – чотири гравці.

Існують 3 стартових комплекти карт, фішок, кубиків. Із карт з наборів гравці збирають ігрові колоди за правилами, що залежать від кількості гравців у партії. Фішки та кубики використовуються в ігровому процесі.

На рисунку Г.1 представлене ігрове поле настільної ККГ «Світ Заару».

Рисунок Г.1 – Ігрове поле настільної ККГ «Світ Заару» (Рисунок виконаний самостійно)

Як видно, ігрове поле розділене на 4 сектори – по одному для кожного можливого гравця. Всього правила дозволяють брати участь 2 – 4 гравцям одночасно в покроковому режимі.

Звернемо увагу на позначення на рисунку Г.1:

1 – ліворуч гравець кладе свою колоду лицьовою стороною вниз;

2 – місце для викладання карти оточення (не більше однієї за хід);

3 – праворуч гравець кладе свій талон (зіграні карти) лицьовою стороною догори;

4 – очки життя;

5 – очки енергії;

6 – очки потаємної сили;

7 – очки броні;

8 – очки негативного впливу;

9 – три місця для розміщення армій гравця (не більше однієї за хід);

10 – Банк (в ньому зібрані ставки під час турніру).

Ігрова колода складається з 15, 30, 50 карт залежно від виду партії.

Кожен гравець складає свою ігрову колоду, збираючи карти зі стартових комплектів або з додаткових колод карт. Карти, обрані гравцем, можуть не прив'язуватися до конкретної раси або бути орієнтовані на характеристики однієї раси. В ігровій колоді не може бути більше 2, 3, 4 однакових карт залежно від виду партії.

Карти в грі розділені на 5 рас та 2 типи. Класифікація карт відповідно до рас представлена на рисунку Г.2. Класифікація карт відповідно до типу представлена на рисунку Г.3. Більш детально структура карт представлена на рисунках Г.4, Г.5. Зворотна сторона карт представлена на рисунку Г.6.

Рисунок Г.2 – Класифікація карт відповідно до рас (Рисунок виконаний самостійно)

Рисунок Г.3 – Класифікація карт відповідно до типу (Рисунок виконаний самостійно)

Рисунок Г.4 – Приклад вигляду карти оточення (Рисунок виконаний самостійно)

Рисунок Г.5 – Приклад вигляду карти воїна (Рисунок виконаний самостійно)

Рисунок Г.6 – Вигляд зворотного боку карти (Рисунок виконаний самостійно)

Хід гри.

а) Початок гри;

1) Розподіл стартового комплекту фішок;

– ОЗ – 30;

– ОЕ – 6;

– ОПС – 1;

2) Усі гравці беруть по 5 карт з верхньої частини своєї колоди в руку;

3) Гравці по черзі кидають 6-гранний кубик. Першим ходитиме той гравець, у кого найбільше число на грані, потім за годинниковою стрілкою. Якщо кільком гравцям випало максимальне (однакове) число, то тільки вони кидають кубики, допоки тільки в одного гравця не випаде максимальне значення.

4) Гравець, який робить хід першим, отримує биток, що вказує на першість ходу на першому колі.

а) Фаза «Активація Карт»;

1) Гравець з битком ходить першим, активуючи карту з руки. Для активації карти гравець повинен мати необхідну кількість ОЕ. Гравець може активувати лише одну карту за хід; Варіанти здійснення ходу:

– Активація карти «Воїна»;

– Активація карти «Оточення»;

– Скинути карту в талон, отримавши 50% її вартості (з округленням в більшу сторону) ОЕ;

– Пропустити хід, отримавши 1 ОНВ (якщо пропуск ходу був зроблений через вплив інших гравців, то ОНВ не нараховуються);

2) Після активації карт «Оточення», ефект настає миттєво;

3) Після першого гравця наступний у черзі гравець переходить у фазу «Активація карти»;

4) Фаза продовжується, допоки коло не замкнеться на першому гравці;

5) По тому, як усі гравці пройшли фазу «Активація карти», перший гравець переходить на фазу «Бій Армій»;

а) Фаза «Битва Армій»;

1) У фазі «Битва Армій» (беруть участь лише гравці, чиї активні армії знаходяться на ігровому полі) перший гравець атакує своїми арміями, які знаходяться в секторах розгорнутих армій. Кожна армія може здійснити лише одну атаку (винятки вказані у спеціальних властивостях карти). Армії можуть:
   – Атакувати вказаного гравця, якщо супротивник не має розгорнутих армій і не має статусу «Недосяжність»;
   – Атакувати вказаного гравця, якщо супротивник розгорнув армії, але ваша армія має особливу властивість «Ігнорувати ворожі армії»;
   – Атакувати вказаного гравця, якщо статус опонента «Недосяжність», але ваша армія має особливу здібність «Ігнорувати статус Недосяжність»;
   – Атакувати армії вказаного гравця, якщо армії супротивника не мають статусу «Недосяжність» або не можуть бути вражені вашим типом наприклад (наприклад, захист від стрілецького);
   – Не атакувати. При цьому армія отримує додатково 1 ОБ, діє винятково на цьому колі фази «Битви Армій»;
2) Розрахунок нанесеної шкоди проходить по формулі: $ОП_г$ - $ОБ_с$ = $ОНП_с$, де:
   – $ОП_г$ – очки пошкоджень армії гравця;
   – $ОБ_с$ – очки броні армії суперника або самого суперника (якщо атака здійснюється безпосередньо по супернику, то враховується тільки 2 ОБ);
   – $ОНП_с$ – очки нанесених пошкоджень супернику чи його арміям;
2) Після того, як армія кожного гравця атакувала, хід переходить до наступного гравця в черзі, поки кожен з гравців не зробить ходи з арміями (за винятком заборони контролю над арміями, нав'язаними гравцеві іншим гравцем за допомогою карти «Оточення»);

3) Після того, як всі гравці зробили свої ходи у фазі «Битва Армій», починається фаза «Завершення Кола»;

а) Фаза «Завершення Кола»;

1) У фазі «Завершення Кола» гравці, починаючи з першого, віддають ОЗ спожиті ОНВ за формулою ОНВ – ОБ = ОЗ (якщо ОБ більше ОНВ, то ОЗ не віднімаються);

2) Після того, як гравці в черговому порядку прибрали необхідну кількість ОЗ, гравці в тому ж порядку скидають ОЕ, які витрачаються на армії, у властивостях яких вказана обов'язкова підтримка додатковими очками енергії за раунд (цикл). Якщо гравець не здатен утримати картку, то вона відправляється в талон;

3) Після того, як гравці скинули необхідну кількість ОЕ, вони в тому ж порядку отримують по 1 ОЕ, за винятком випадків, коли на них впливають здібності карт (зменшення або збільшення приросту ОЕ);

4) Після того, як гравці отримали ОЕ, вони беруть по 1 новій карті зі своєї колоди, за винятком випадків впливу тих карт, які впливають на цей етап;

5) Після того, як гравці отримали нові карти, всі скидають активні карти «Оточення» у талон;

6) Після того, як гравці скинули карти «Оточення» в талон, биток переходить до наступного гравця у черзі. З нього починається новий цикл (раунд);

7) Биток передається щоразу після фази «Завершення Кола»;

8) Коли биток вдруге перейшов до першого гравця (того, хто почав гру), на етапі «Завершення кола» попереднього гравця він отримує 1 ОПС. Потім, до кінця гри, кожен гравець отримуватиме 1 ОПС наприкінці ходу попереднього гравця.

Гра триває допоки, на полі не залишиться тільки один гравець (пара – в випадку парного розряду або кваліфікаційних етапів турніру).

У грі присутні 5 видів фішок:

– ОЗ;

– ОЕ;

– ОПС;

– ОБ;

– ОНВ.

ОЗ – це очки індикації життєвої енергії гравця. На початку гри кожен гравець отримує 30 од.

ОЗ можуть бути втрачені через:

– атаки армій суперників;

– атаки карт «Оточення» суперників;

– накладені ОНВ.

Коли ОЗ падають до 0 – це означає програш гравця. По ходу гри за допомогою карт «Оточення» можна не тільки відновлювати втрачені ОЗ, а й збільшувати їх кількість.

ОЕ – це очки індикації енергії гравця. Енергія необхідна для активації карт. На початку гри кожен гравець отримує 6 ОЕ.

ОЕ можуть бути витрачені через:

– активація власних карт;

– вплив ворожих карт «Оточення».

В процесі гри можливо не тільки відновлювати втрачені ОЕ, а й збільшувати їх кількість:

– на початку кожного раунду (циклу) всі гравці отримують по 1 ОЕ;

– за допомогою карток «Оточення»;

– при скиданні картки в талон на етапі «Активація Карт», отримавши 50% від її вартості (округлено в більшу сторону) ОЕ.

ОПС – це очки індикації потаємної сили гравця, які потрібні для посилення впливу карт «Оточення». На початку гри кожен гравець отримує по 1 ОПС.

ОПС можна витратити на:

– посилення карток «Оточення», у властивостей яких вказана потреба споживання ОПС.

В процесі гри можливо не тільки відновлювати втрачені ОПС, а й збільшувати їх кількість:

– на початку ходу кожного гравця (коли гравець відкриває коло), починаючи з 2-го приходу битка, Гравець отримує 1 ОПС;

– за допомогою карток «Оточення».

ОБ – це очки індикації броні гравця, які необхідні для захисту від атак армій супротивників та карт «Оточення».

В ході гри можна збільшувати кількість ОБ:

– за допомогою карт «Оточення».

Гравець використовує ОБ:

– для захисту від атак ворожих армій (не більше 2 ОД з усіх своїх ОБ);

– для захисту від атак карт «Оточення» суперника (всі ОБ);

– для захисту від ОНВ, накладених на гравця (всі ОБ).

ОНВ – це очки індикації негативного впливу на гравця. Вони завдають шкоди гравцеві під час фази «Завершення кола». При нанесенні шкоди від ОНВ враховуються всі ОБ. ОНВ не можуть накладатися суперником на гравця, якщо той знаходиться поза зоною досяжності (статус «Недосяжність»), але раніше додані ОНВ продовжують завдавати шкоди таким гравцям.

По ходу гри кількість ОНВ збільшується від:

– атак ворожих армій (особливі здібності яких полягають у додаванні ОНВ суперникові);

– атаки карт «Оточення» суперників (особливі здібності яких полягають у додаванні ОНВ суперникові);

– пропуск ходу в фазі «Активація Карт».

Гравець використовує ОНВ для:

– зняття ОЗ у супротивників;

– підсилення карт «Оточення», здібності яких пов’язані з ОНВ.

В таблиці Г.1 представлені деякі унікальні здібності карт.

Таблиця Г.1 – Приклад унікальних здібностей карт (Таблиця виконана самостійно)

№	Короткий опис	Пояснення
1	Завдає пошкоджень опонентові, рівних (Х + кількість поточних очок потаємної сили)	На момент активації карти гравцем опонент отримує (Х + ГА.ОПС) очок пошкоджень
2	Збільшення очок броні гравця	При активації збільшує кількість ОБ гравця на Х одиниць
3	Знімає з гравця всі очки негативного впливу	На момент активації карти гравець втрачає всі очки НВ
4	Збільшує в Х разів потаємну силу гравця	На момент активації карти гравець збільшує в Х разів свою ПС.
5	Відновлює Х одиниць очок здоров'я гравцеві	На момент активації карти миттєво дає Х ОЗ гравцеві
6	Збільшує в Х разів енергію гравця, що лишилася після активації	На момент активації збільшує в Х разів енергію гравця, що лишилася після активації карти
7	Гравець отримує (Х + кількість поточних очок потаємної сили) очок енергії	На момент активації карти гравець отримує (Х + ГА.ОПС) очок енергії
8	Позбавляє суперника всіх очок броні	На момент активації позбавляє суперника всіх ОБ
9	Збільшує пошкодження, які наносять усі воїни гравця на Х одиниць	На момент активації карти збільшує на Х одиниць пошкодження, які наносять усі воїни гравця у фазу «Бою Армій». По завершенні ходу (карта пішла в талон) очки пошкоджень повертається в норму.
10	Зменшення броні всіх воїнів суперника на Х одиниць	На момент активації карти зменшує кількість очок броні всіх воїнів суперника на Х одиниць. По завершенні ходу (карта пішла в талон) очки броні повертається в норму.

Продовження таблиці Г.1

11	Може завдавати шкоди безпосередньо супернику, ігноруючи його воїнів	Воїн з цією здібністю під час фази «Бою Армій» може завдавати шкоди безпосередньо супернику, ігноруючи його армії та їх властивості
12	Додаткові пошкодження від негативного впливу	Воїн з цією здібністю має додає до власних ОП ОНВ суперника
13	Відновлює за кожен хід по Х одиниць здоров'я сусіднім воїнам	Воїн з цією здібністю під час свого ходу у фазу «Бою Армій» відновлює по Х одиниць здоров'я воїнам гравця
14	Зменшує на Х одиниць очки пошкоджень воїнів суперника	Воїн з цією здібністю зменшує на Х одиниць ОП воїнів суперника. Коли карта відправляється в Талон ефект анулюється
15	Збільшує на Х одиниць очки захисту сусідніх воїнів гравця	Воїн з цією здібністю збільшує на Х одиниць ОБ воїнам гравця. Коли карта відправляється в Талон ефект анулюється
16	Ігнорує броню воїнів суперника	Воїн з цією здібністю ігнорує броню воїнів суперника
17	Збільшує на Х одиниць очки здоров'я сусіднім воїнам гравця	Воїн з цією здібністю додає Х одиниць здоров'я сусіднім воїнам гравця. Коли карта відправляється в Талон ефект анулюється
18	Споживає очки енергії гравця за хід	Цей воїн у фазі «Завершення Кола» споживає Х ОЕ гравця. Якщо ОЕ не вистачає, то карта йде в талон
19	Кожен цикл збільшує очки броні гравця на Х одиниць	Воїн з цією здібністю збільшує ОБ гравця кожного разу на фазі «Завершення Кола» на Х одиниць.
20	Кожен цикл збільшує очки потаємної сили гравця на Х одиниць	Воїн з цією здібністю збільшує прихід ОПС гравця кожного разу на фазі «Завершення Кола» на Х одиниць, коли до нього приходить биток, починаючи з 2-го кола.