**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка бібліотеки, що реалізує комплексну магію у фентезі іграх у віртуальній реальності з використанням рушія Unreal Engine 5»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Богдан КАВАЦЮК
*(підпис)*

Виконав: здобувач вищої освіти групи ПД-42

Богдан КАВАЦЮК

Керівник: Оксана ЗОЛОТУХІНА
*к.т.н., доцент*

Рецензент: _____

**Київ 2024**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кавацюку Богдану Олеговичу

1. Тема кваліфікаційної роботи: «Розробка бібліотеки, що реалізує комплексну магію у фентезі іграх у віртуальній реальності з використанням рушія Unreal Engine 5»
керівник кваліфікаційної роботи к.т.н., доц., доцент кафедри ІПЗ Оксана ЗОЛОТУХІНА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: відеоогляди ігор віртуальної реальності, які мають вбудовані магічні системи, технічна документація з описом бібліотек для створення плагінів з допомогою рушія Unreal Engine 5.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз існуючих ігор із вбудованою системою магії для шоломів віртуальної реальності.
2. Проектування плагіну для рушія Unreal Engine 5, який спрощує створення системи магії.

3. Програмна реалізація та опис функціонування плагіну для рушія Unreal Engine 5, який спрощує створення системи магії.
4. Тестування плагіну.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів
2. Демонстрація аналогів.
3. Вимоги до програмного забезпечення.
4. Програмні засоби реалізації.
5. Діаграма варіантів використання бібліотеки
6. Діаграма варіантів використання гри, розробленої з допомогою бібліотеки.
7. Діаграма класів.
8. Екранні форми.
9. Демонстрація розпізнавання символу
10. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір та аналіз науково-технічної літератури | 28.02.-06.03.2024 | |
| 2 | Аналіз існуючих систем магії на ринку ігор віртуальної реальності | 01.04-16.04.2024 | |
| 3 | Проектування системи розпізнавання символів та жестів | 16.04-19.04.2024 | |
| 4 | Проектування інтерфейсу користувача | 19.04-24.04.2024 | |
| 5 | Проектування системи додавання символів | 24.04-26.04.2024 | |
| 6 | Проектування системи додавання комбінацій символів та призначення заклинань | 28.04-28.04.2024 | |
| 7 | Реалізація системи додавання символів | 28.04-30.04.2024 | |
| 8 | Реалізація системи додавання комбінацій символів та призначення заклинань | 30.04-04.05.2024 | |
| 9 | Тестування бібліотеки | 04.05-08.05.2024 | |
| 10 | Оформлення роботи: вступ, висновки, реферат | 08.05-10.05.2024 | |
| 11 | Розробка демонстраційних матеріалів | 10.05-12.05.2024 | |
| 12 | Попередній захист роботи | 13.05-31.05.2024 | |

Здобувач вищої освіти _____ Богдан КАВАЦЮК
*(підпис)*

Керівник
кваліфікаційної роботи _____ Оксана ЗОЛОТУХІНА
*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 41 стор., 2 табл., 29 рис., 10 джерел.

*Мета роботи* – Спрощення процесу розробки ігор, пов’язаних з використанням магії у віртуальній реальності.

*Об’єкт дослідження* – Процес створення механіки гри у віртуальній реальності.

*Предмет дослідження* – Засоби розробки механіки гри, пов’язаної з магією у віртуальній реальності.

*Короткий зміст роботи*: В роботі проаналізовано магічні системи у різних іграх віртуальної реальності. Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: можливість додати символ, можливість додати заклинання, можливість вказати базовий клас заклинання, розпізнавання жестів та заклинань. Проведено функціональне та модульне тестування додатку. В роботі використано Figma для прототипування інтерфейсів, Unreal Engine Blueprints для створення алгоритмів, які взаємодіють з ігровим світом, C++ для написання логіки плагіну і інтерфейсів.

КЛЮЧОВІ СЛОВА: Розробка ігор, віртуальна реальність, Unreal Engine, C++.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

**ВР** - Віртуальна реальність

**MMORPG** (з англ. Massively Multiplayer Online Role-Playing Game) - Масова багатокористувацька онлайн рольова гра

**ААА гра** - гра, створена середніми і великими видавництвами, що вкладають багато коштів на розробку й рекламу.

**Інді гра** - гра, створена незалежно від фінансової підтримки великих видавців, зазвичай одним, або невеликою групою розробників.

# ВСТУП

За останні 10 років шоломи ВР значно розвинулася, що привернуло увагу як користувачів так і розробників. Багато великих компаній приєднались до розробки шоломів ВР та програмних продуктів для вирішення проблем бізнес сфери. З іншого боку завдяки інноваціям у галузі сенсорів та графіки створення захоплюючих іммерсивних ігор стає все доступнішим для інді розробників.

Сучасний ринок ігор для шоломів ВР переповнений інді іграми, і нараховує всього декілька ААА ігор. На цьому ринку існує ніша ігор у жанрі фентезі, яка зайнята малою кількістю ігор, які різко втрачають популярність. Тільки одна з цих ігор має систему магії, закладену розробниками з самого початку. Інші - не мають встроєної системи магії, тому гравці розробили модифіковані сервери із власними системами магії. (приклад - A Township Tale) Інді-розробники, зазвичай, не мають часу на створення власних систем магії, а фокусуються на інших аспектах гри. Саме тому створення інструментів для системи магії, зручної для гравця та для модифікації розробниками гри є актуальною проблемою.

Об’єктом дослідження є Процес створення механіки гри у віртуальній реальності.  Предмет дослідження - засоби розробки механіки гри, пов’язаної з магією у віртуальній реальності. А мета даної роботи полягає в спрощенні процесу розробки ігор, пов’язаних з використанням магії у віртуальній реальності.

Завдання дослідження:

1. Дослідження потреб ринку ігор ВР
2. Аналіз магічних систем у існуючих іграх ВР та їх вплив на ігровий процес
3. Проектування архітектури бібліотеки
4. Розробка інструмента для розпізнавання символів
5. Розробка інструмента для додавання символів активації
6. Розробка інструмента для створення заклинань

7. Розробка інструмента для призначення виклику заклинань на комбінацію символів або жест
8. Інтеграція розробленої бібліотеки у демонстраційну гру, розроблену для оцінки її ефективності та функціональності.
9. Проведення мануального тестування системи
10. Провести апробацію роботи

Робота пройшла апробацію на Всеукраїнській науково-технічній конференції “Застосування програмного забезпечення в ІКТ” (24.04.2024, ДУІКТ, м. Київ) та Міжнародній науково-практичній інтернет-конференції “Молодь в науці: дослідження, проблеми, перспективи” (20.05.2024, ВНТУ, м. Вінниця).

За результатами участі опубліковано тези доповідей [1, 2].

# 1 АНАЛІЗ АНАЛОГІВ

## 1.1 Аналіз концепції та видів магічних систем ВР

Поняття магії дуже обширне, і відрізняється у всіх художніх творах та комп'ютерних іграх, у яких воно згадане або описане. Якщо у художніх творах автор не обмежений у фантазії, і може написати все, що захоче, не задумуючись про ігровий баланс та можливості ігрового рушія, то у сфері комп'ютерних ігор магія - досить обмежений інструмент досягнення дуже конкретних цілей. Зазвичай, магія у комп'ютерних іграх - це частина бойової системи з атакуючими та захисними заклинаннями, які викликаються на натискання однієї, або комбінації клавіш на клавіатурі. Дуже яскравим винятком із правила є гра Arx Fatalis, яка для виклику заклинання змушує гравця малювати символи курсором на екрані. У додачу до незвичайної системи магії у цій грі вона має цікавий сюжет та деталізовану історію світу, що поринає гравця у створений розробниками світ. Система магії Arx Fatalis нараховує більше 40 заклинань - комбінацій рун, які включають дуже корисні небойові заклинання, на приклад заклинання Ignite, яке запалює горючі речовини і предмети у невеликому радіусі навколо гравця.

Процес застосування гравцем магії у віртуальній реальності складається з декількох етапів, на кожному з яких є можливість покращення ігрового досвіду гравця.

Перший етап - це вибір гравцем заклинання, яке він хоче застосувати. На цьому етапі вступає у силу різновидність магічних заклинань. На приклад, багато ситуацій у магічних дуелях і боях, які займають більшу частину часу, проведеного у грі, подібні, і зводяться до декількох основних дій:

- Захист.
- Атака.
- Відновлення.

Другий етап - це виклик вибраного заклинання. Заклинання може бути визвано чотирьма способами:

1. Комбінацією клавіш
2. Жестами
3. Малюванням символів у повітрі
4. Голосовою активацією

Для контролерів віртуальної реальності перший метод не підходить через обмежену кількість кнопок на контролерах, що зображено на рис 1.1. Такий спосіб виклику заклинань існує у декількох іграх на одного гравця, і жертвує швидким вибором заклинань, тому що перед активацією потрібно вибрати заклинання зі списку доступних, а це займає набагато більше часу, ніж вибір і виклик заклинань у інших методах.

Рис. 1.1 Контролер Oculus Quest 2

Рис. 1.2 Контролер Valve Index

Рис. 1.3 Контролер HTC VIVE

На рисунках 1.1 - 1.3 зображені контролери різних ВР систем. На них виділені жовтим кольором усі кнопки, придатні до використання у грі як вхідних сигналів:

1. Рисунок 1.1 - контролер Ocukus Quest 2, 5 кнопок, і джойстик
2. Рисунок 1.2 - контролер Valve Index, 5 кнопок, і джойстик

3. Рисунок 1.3 - контролер HTC VIVE, 5 кнопок, і сенсорна панель, яка виконує роль джойстика

При активації заклинання жестами гравець дуже виграє у швидкості у порівнянні з іншими методами, тому що жести виклику дуже короткі. Єдиний мінус цієї системи виклику заклинань - це неможливість додати велику кількість заклинань у неї. При виконанні одноманітних або схожих жестів заклинання можуть плутатись, і буде викликане не те заклинання, яке хотів викликати гравець.

Система виклику заклинань методом малювання символів у повітрі оптимальна для їх великої кількості. При чергуванні простих символів кількість комбінацій буде достатньою, щоб описати всі заклинання.

Щодо голосової активації - її недолік полягає у розпізнаванні слів, які говорить гравець. Ця система дозволяє відносно швидко викликати заклинання, і має потенціал для зберігання багатьох заклинань, оскільки тут, аналогічно з системою викликів символами, можна створювати фрази із простих слів, яких вистачить для опису всіх заклинань

Дипломна робота концентрується на системі активації символами та жестами, оскільки це найпопулярніші і найбільш протестовані системи, які прижилися серед магічних систем ВР через надійність спрацювання та інтерактивність.

Розпізнавання слів-заклинань імплементовано у декількох іграх як модифікації до цих ігор, але виклик заклинань часто не спрацьовує, при спробі його активації, або спрацьовує із значною затримкою, що негативно впливає на ігровий досвід.

## 1.2 Аналіз магічних систем у існуючих іграх ВР

Для аналізу існуючих магічних систем у іграх ВР розглянемо найпопулярніші ігри у цій категорії:

1. Orbus VR

2. War of Wizards VR
3. Rumble VR
4. Magitek VR

Orbus VR - VR MMO RPG гра. Як і в усіх MMO RPG іграх, у ній є різні класи. Один з цих класів - маг, який може використовувати магічні заклинання з допомогою чарівної палички, якою у повітрі потрібно намалювати певний символ, що зображено на рисунку 1.2. У більшості заклинань є 3 рівні потужності, де рівень потужності визначається складністю символу, що зображено на рисунку 1.3. Заклинання в цій грі обмежені бойовими - тобто там існують тільки атакуючі і захисні заклинання: на приклад, вогняний шар, або щит мани. Ця система магії дає гравцю можливість протистояти монстрам та іншим гравцям у бою, з відносно швидким часом створення заклинань. Також, більшість заклинань в цій системі - односимвольні - тобто для виклику заклинання потрібно намалювати один унікальний для цього заклинання символ. У цій системі символи заклинання першого, другого і третього рівня сили значно різняться між собою, що додає ще більше складності та незрозумілості цій системі. Маг вважається одним з найскладніших класів у цій грі, і через це дуже мало гравців вибирають саме цей клас для свого персонажа.

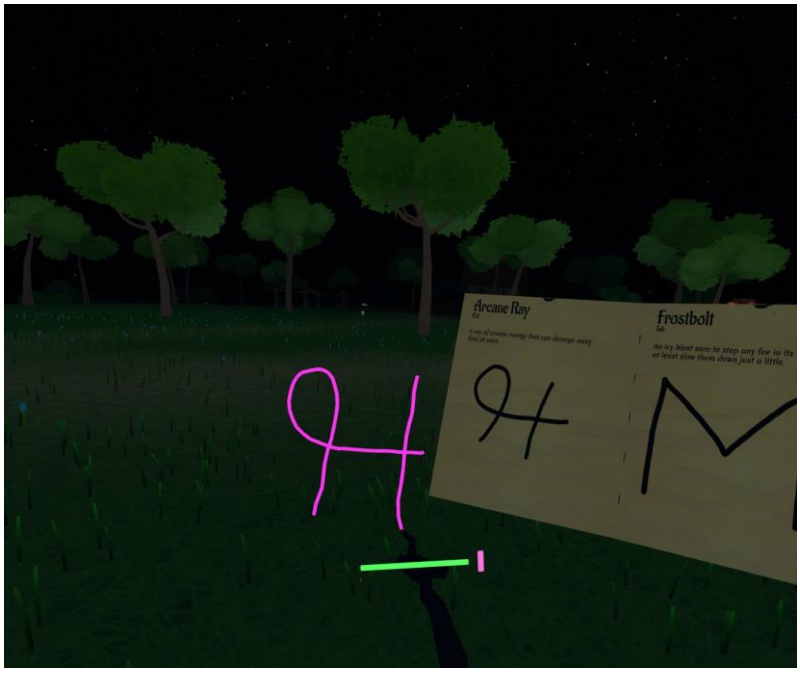



Рис. 1.4 Символ для виклику заклинання у Orbus VR

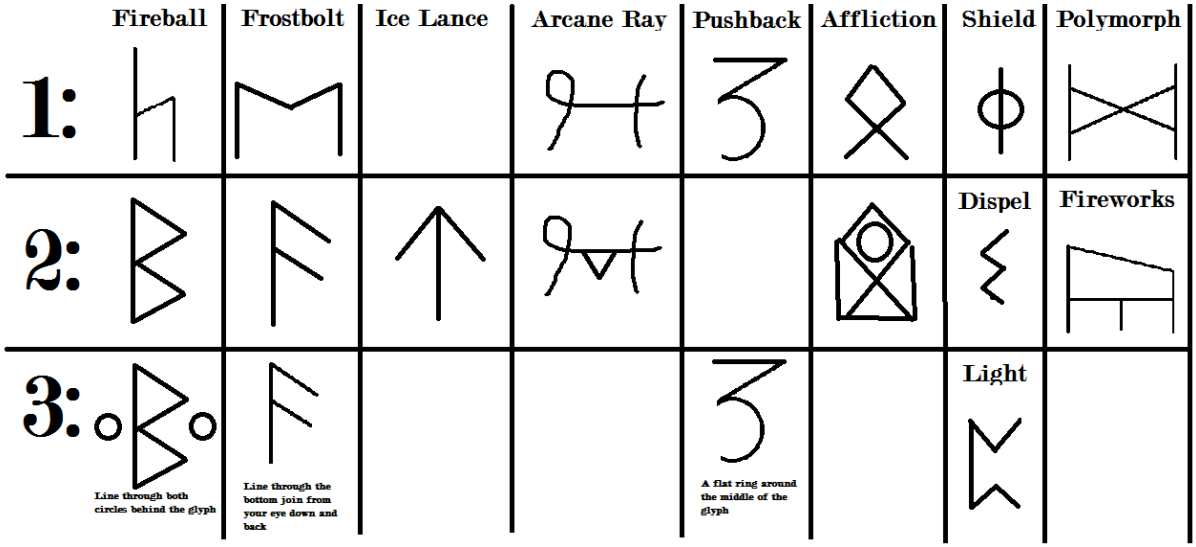



Рис. 1.5 Ступені потужності заклинань у Orbus VR

War of Wizards VR - VR MOBA гра. У цій грі також імплементована система виклику заклинань символами. Різноманіття заклинань у цій грі дуже велике, і можна підібрати собі набір заклинань, який буде співпадати зі стилем гри будь-якого гравця. Кожен знак заклинання складається з одного символу, хоч він і дуже складний, що зображено на рисунку 1.4. Чим складніший знак - тим потужніше заклинання. Також, гравець не має безпосереднього доступу до всіх заклинань під час бою: він повинен обрати 4 заклинання, які він буде використовувати під час бою, а всі інші заклинання будуть йому недоступні.


Рис. 1.6 Символи для виклику заклинання у War of Wizards VR

Переходячи до систем магії, де заклинання викликаються жестами - неможливо не розглянути Rumble VR - VR грау - дуель двох гравців. Вона стала найпопулярнішою грою - дуеллю через надзвичайну динаміку та інтенсивність гри. Вона вимагає від гравця швидкої оцінки ситуації і блискавичного прийому рішень. У цій грі магія (бойова система) реалізована 16-ма жестами, де кожен жест викликає певне заклинання. Для жестів задіюються обидві руки з розсинхронаними жестами - тобто якщо для виклику заклинання одна рука повинна рухатись від гравця вліво, то інша - вверх. У грі використовуються тільки заклинання стихії землі (земляна стіна, земляний диск, земляний куб, ітд), що є нерізноманітним, але цього вистачає для створення найрізноманітніших тактик ведення бою. Особливістю цієї гри є те, що уже створені заклинання можуть використовуватись повторно: на приклад, гравець визвав земляну стіну, щоб захиститись від атаки суперника, а наступним жестом запустив цю стіну у свого опонента, перетворивши заклинання захисне у заклинання атакуюче.

Magitek VR - VR MMO RPG гра. У цій грі заклинання теж викликаються жестами. Для деяких жестів задіюється одна рука, а для деяких - обидві руки. Перед заклинанням гравець обирає стихію для обох рук. Можна комбінувати дві стихії (одна стихія в одній руці, інша в іншій), а можна взяти однакову стихію в обидві руки. Ця система дає набагато більше різноманіття заклинань, ніж Rumble VR, тому що на одні й ті ж рухи викликаються різні заклинання, залежно від вибраних стихій.

Система магії у демонстраційній грі буде складатися з декількох символів, які гравець зможе комбінувати у довільному порядку. Таким чином буде досягнуто простоти і логічності виклику заклинань. На приклад символ “Water” і символ ”Area of effect” в комбінації створять заклинання дощу, який буде йти у вибраній гравцем зоні. Дане заклинання не призначене для бойових сценаріїв, а призначене для поливання рослин, або наповнення ємностей питною водою. Також у демонстраційній грі буде метод швидкого визову заклинань жестами, і можливість гравцю визначити, які заклинання будуть викликатись якими рухами.

Далі наведена таблиця порівняння існуючих систем магії, та систему магії, яку розробник може створити з допомогою розробленої бібліотеки.

Таблиця 1.1.

Порівняння аналогів

| Критерій порівняння | Orbus VR | War of Wizards VR | Rumble VR | Magitek VR | Розроблена бібліотека |
|---|---|---|---|---|---|
| Система активайії символами | + | + | - | - | + |
| Система активації жестами | - | - | + | + | + |
| Кількість заклинань, доступних одномоментно | 22 | 4* | 16 | 4** | 30*** |
| Наявність небойових заклинань | - | - | - | - | + |
| Можливість призначити заклинання на жест | відсутня система жестів | відсутня система жестів | - | - | + |

*Загальна кількість заклинань - 71, але гравець обмежений до 4 заклинань, які він вибрав перед боєм.

**Загальна кількість заклинань - 30, але гравець обмежений до 4 заклинань, які доступні йому одномоментно.

***5 символів у комбінації дають 30 неповторних комбінацій

Як видно з таблиці - основний недолік існуючих систем магії ВР полягає у відсутності комбінації системи виклику жестами та системи виклику символами. Розроблена бібліотека дозволяє впровадити обидві системи у єдину гру, а також надати гравцю можливість перепризначати заклинання на жести для більшої зручності.

# 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА БІБЛІОТЕКИ

## 2.1 Огляд програмних засобів реалізації

Перш за все використовується рушій Unreal Engine 5, оскільки бібліотека розробляється саме для цього рушія.

Unreal Engine 5 - це один з найпопулярніших ігрових рушіїв на сьогоднішній день. Ігрові рушії використовують інді розробники, і невеликі компанії, у яких немає ресурсів на створення свого рушія. Unreal Engine має безліч встроєних функцій і плагінів, які вирішують багато проблем. Цей рушій дуже часто оновлюється, і надає дуже зручні інструменти для розробки програмних продуктів, а також є лідером у сфері створення візуальних ефектів.

### 2.1.1 Blueprints

В Unreal Engine є власна візуальна мова програмування - Blueprints, яка дозволяє створювати складні логічні конструкції без необхідності написання коду. Ця візуальна скриптова система програмування підпадає під класифікацію No Code, що є набагато доступнішим для новачків, а також пришвидшує процес створення прототипів. Blueprints дозволяє розробникам інтерактивно налаштовувати поведінку об'єктів і події у грі, що є незмінним для розробки систем, які безпосередньо інтерактують з ігровим світом.

Для розробки алгоритму в системі Blueprints використовуються Events та Functions. Усі Events класу доступні для редагування у Event graph, а функції доступні для редагування тільки окремо.

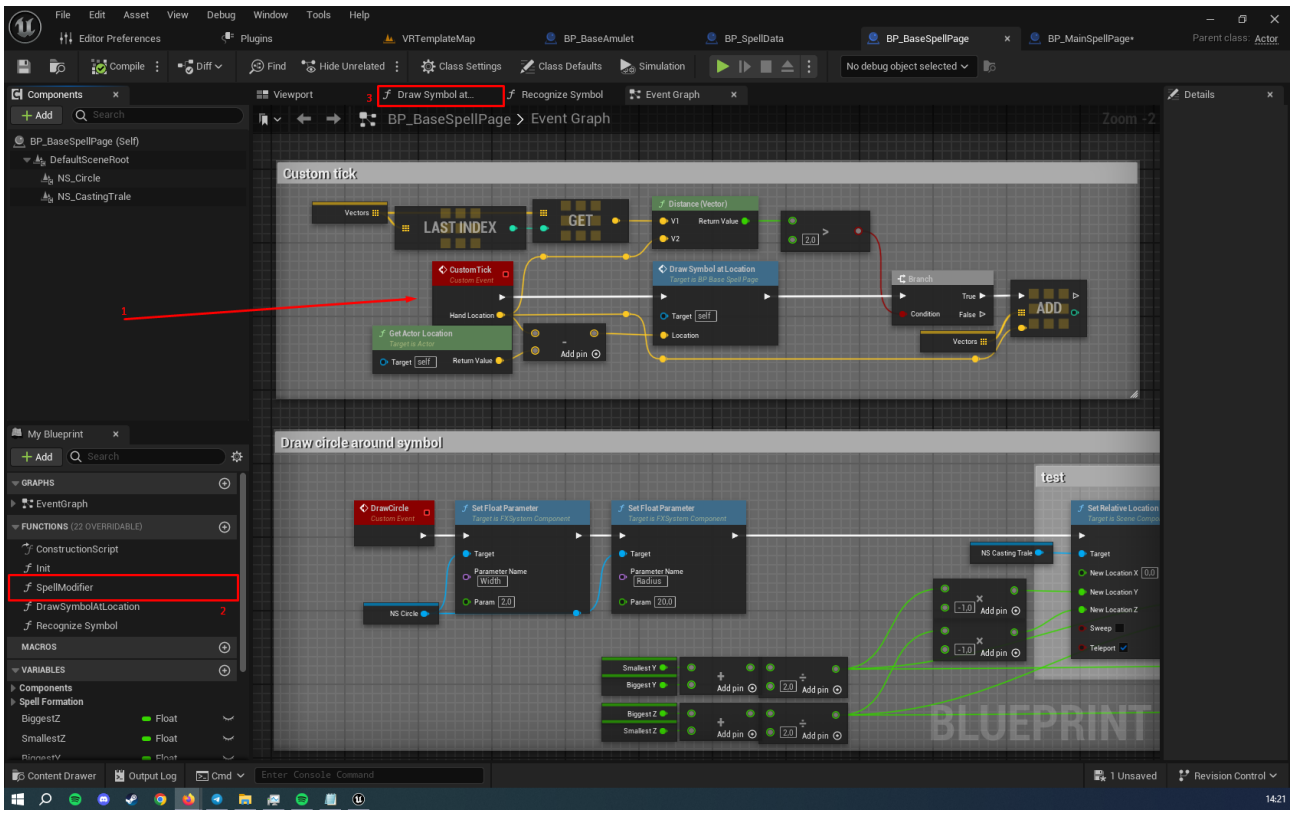



Рис. 2.1 Система No Code програмування Blueprints. Цифрою 1 позначений Event “CustomTick”, який приймає поле FVector “Hand Location”, цифрою 2 позначена Function, цифрою 3 позначена кнопка переходу до редагування функції “Draw Symbol At Location”

### 2.1.2 C++

Окрім Blueprints, Unreal Engine 5 підтримує мову програмування C++ з набором інструментів, доступних тільки через C++. Саме в бібліотеках C++ є класи, які дозволяють створити плагін для середовища розробки рушія. Ці класи і бібліотеки не доступні у системі Blueprints, оскільки вимагають більшого контролю для користування ними, який надає C++, але не надає Blueprints.

### 2.1.3 Visual Studio IDE і Unreal Build Tool

Unreal Engine 5 має нативну інтеграцію з Visual Studio IDE, що дозволяє зручно користуватись всіма функціями C++ розробки у цьому середовищі. Це забезпечує розробникам можливість використовувати потужні інструменти для

написання, налагодження та оптимізації коду. Visual Studio підтримує широкий спектр функцій, включаючи потужні інструменти для налагодження, автодоповнення коду та інтеграцію з системами контролю версій, що значно підвищує ефективність розробки.

Окрім того, Unreal Engine 5 пропонує вбудовані інструменти для управління компіляцією проектів і автоматизації робочих процесів, що ще більше покращує взаємодію з Visual Studio. Наприклад, Unreal Build Tool (UBT) автоматично управляє залежностями і компіляцією C++ коду, забезпечуючи швидке і ефективне збірку проектів. Це дозволяє розробникам зосередитися на написанні коду, не турбуючись про технічні деталі процесу компіляції.

### 2.1.4 HotReload

Unreal Engine 5 також підтримує функцію HotReload, яка є важливим інструментом для підвищення ефективності розробки. HotReload дозволяє розробникам вносити зміни в код C++ і швидко бачити результати цих змін без необхідності повного перезапуску рушія або проекту. Ця функція значно скорочує час на тестування і налагодження, оскільки дозволяє миттєво оновлювати код і перевіряти його роботу в реальному часі. Розробники можуть змінювати логіку гри, виправляти помилки і додавати нові функції, залишаючись у робочому середовищі, що забезпечує безперервність робочого процесу.

HotReload інтегрований з Visual Studio, що робить процес внесення змін ще більш зручним. Після внесення змін до коду в Visual Studio, розробник може просто зберегти файл, і HotReload автоматично оновить код в Unreal Engine. Це усуває потребу в тривалому перезапуску середовища розробки Unreal Engine і дозволяє швидко ітеративно працювати над проектом. Функція HotReload є особливо корисною для великих проектів з великою кількістю коду, де повний перезапуск може займати багато часу.

### 2.1.5 DataTables

Unreal Engine 5 підтримує використання DataTables для організації та управління даними в ігрових проектах. DataTables - це інструмент для зберігання великої кількості структурованих даних у зручному для читання і редагування форматі. Оскільки інтеграція БД в Unreal Engine 5 вимагає сторонніх бібліотек, і дуже часозатратного налаштування, то DataTables - це основний метод зберігання даних в Unreal Engine 5. Вони зберігають дані в табличному вигляді, де кожен рядок представляє окремий запис, а кожен стовпчик – атрибут структури, на основі якої створений даний DataTable. DataTables використовують для зберігання налаштувань персонажів, параметрів предметів, діалогів і інших типів даних, які потребують модифікації та доступу.

Створення та використання DataTables в Unreal Engine 5 досягається завдяки його інтеграції з програмним середовищем. DataTables можуть бути створені на основі структур (Structs) C++, що дозволяє розробникам визначати типи даних і організувати їх відповідно до потреб проекту. Наприклад, структура може містити атрибути заклинання, такі як кількість мани, яка використовується при виклику даного заклинання, радіус ураження та візуальний ефект ураження. Після створення структури можна створити DataTable, яка буде зберігати дані для багатьох заклинань, де кожен рядок представляє окреме заклинання з відповідними значеннями атрибутів.

Управління DataTables здійснюється за допомогою інтерфейсу Unreal Engine, що дозволяє легко переглядати і редагувати дані безпосередньо у редакторі. Це дає можливість не тільки програмістам, але й дизайнерам ігрового процесу працювати з даними без необхідності занурюватися в код. Всі зміни, внесені у DataTables, автоматично застосовуються до гри, що дозволяє швидко тестувати і налаштовувати параметри ігрових елементів. Це значно пришвидшує процес розробки, оскільки дозволяє миттєво бачити результати змін без необхідності перезапуску гри або перекомпіляції коду.

### 2.1.6 Slate

Основним інструментом розробки інтерфейсу користувача, доступним у C++ в Unreal Engine 5 є фреймворк Slate. Slate є потужною та гнучкою системою для створення користувацьких інтерфейсів, яка надає розробникам можливість створювати як прості, так і дуже складні інтерфейси. Slate використовує декларативний стиль програмування на C++, що дозволяє чітко визначати структуру і логіку UI. Розробники можуть створювати власні віджети, налаштовувати їх зовнішній вигляд і поведінку, а також комбінувати їх у більш складні компоненти. Slate віджети можуть взаємодіяти з іншими елементами гри, що дозволяє створювати інтерактивні і динамічні інтерфейси.

Крім створення інтерфейсів для гри, Slate також може бути використаний для створення користувацьких інтерфейсів для плагінів в середовищі розробки Unreal Engine. Це дає можливість розробки зручного та інтуїтивно зрозумілого інтерфейсу для своїх плагінів, які можуть використовуватися у редакторі Unreal Engine під час розробки проектів. Використання Slate дозволяє створювати розширення, які інтегруються з іншими інструментами розробки та інтерфейсом користувача Unreal Engine, забезпечуючи ідентичні по стилю інтерфейси. Це значно спрощує процес взаємодії з плагінами та підвищує ефективність розробки у середовищі Unreal Engine.

### 2.1.7 Figma

Figma — це потужний інструмент для проектування інтерфейсу користувача, який широко використовується для створення високоякісних макетів, прототипів та колаборативних проектів. Завдяки своїй роботі в хмарі, Figma дозволяє зберігати всі ваші проекти в одному місці, доступному з будь-якого пристрою. Це означає, що ви можете працювати над своїми проектами де завгодно і коли завгодно, без необхідності турбуватися про синхронізацію файлів чи версійність. Інструмент також надає можливості для інтерактивного прототипування, що дозволяє легко демонструвати та тестувати інтерфейси, додаючи переходи та анімації безпосередньо в Figma.

Figma підтримує зручне створення високоякісних макетів та прототипів, що дозволяє легко візуалізувати та тестувати інтерфейси. Інструмент надає широкий спектр функцій для компонування, стилізації та управління ресурсами, що дозволяє створювати уніфіковані та консистентні інтерфейси, які відповідають сучасним стандартам UX/UI дизайну. Завдяки своїм потужним інструментам, Figma робить процес проектування ефективним і менш схильним до помилок.

## 2.2 Створення вимог та проектування архітектури програмного забезпечення

Після детального аналізу кожної системи магії та можливостей інструментів та бібліотек Unreal Engine було сформовано функціональні вимоги до інструменту:

1. Розробити систему розпізнавання символів
2. Розробити систему розпізнавання жестів
3. Розробити інтерфейс для створення символів
4. Розробити інтерфейс для формування символів у комбінації, і призначення заклинань на комбінації
5. Розробити інтерфейс для формування жестів, і призначення заклинань на жести
6. Додати символи, жести та заклинання через розроблені інструменти

Та нефункціональні вимоги:

1. Користувацький інтерфейс не повинен вибиватися із загальної стилістики рушія
2. Введення символу повинно відбуватись швидко та просто
3. Введення заклинань та їх призначення повинно відбуватись швидко та просто
4. Швидкість активації заклинання не повинна перевищувати 0,5 секунди

Важливим є надання користувачу можливості створювати заклинання на базі його власного батьківського класу, і опису ним поведінки заклинання. Також всі створені дані повинні автоматично зберігатися у відповідних DataTables.

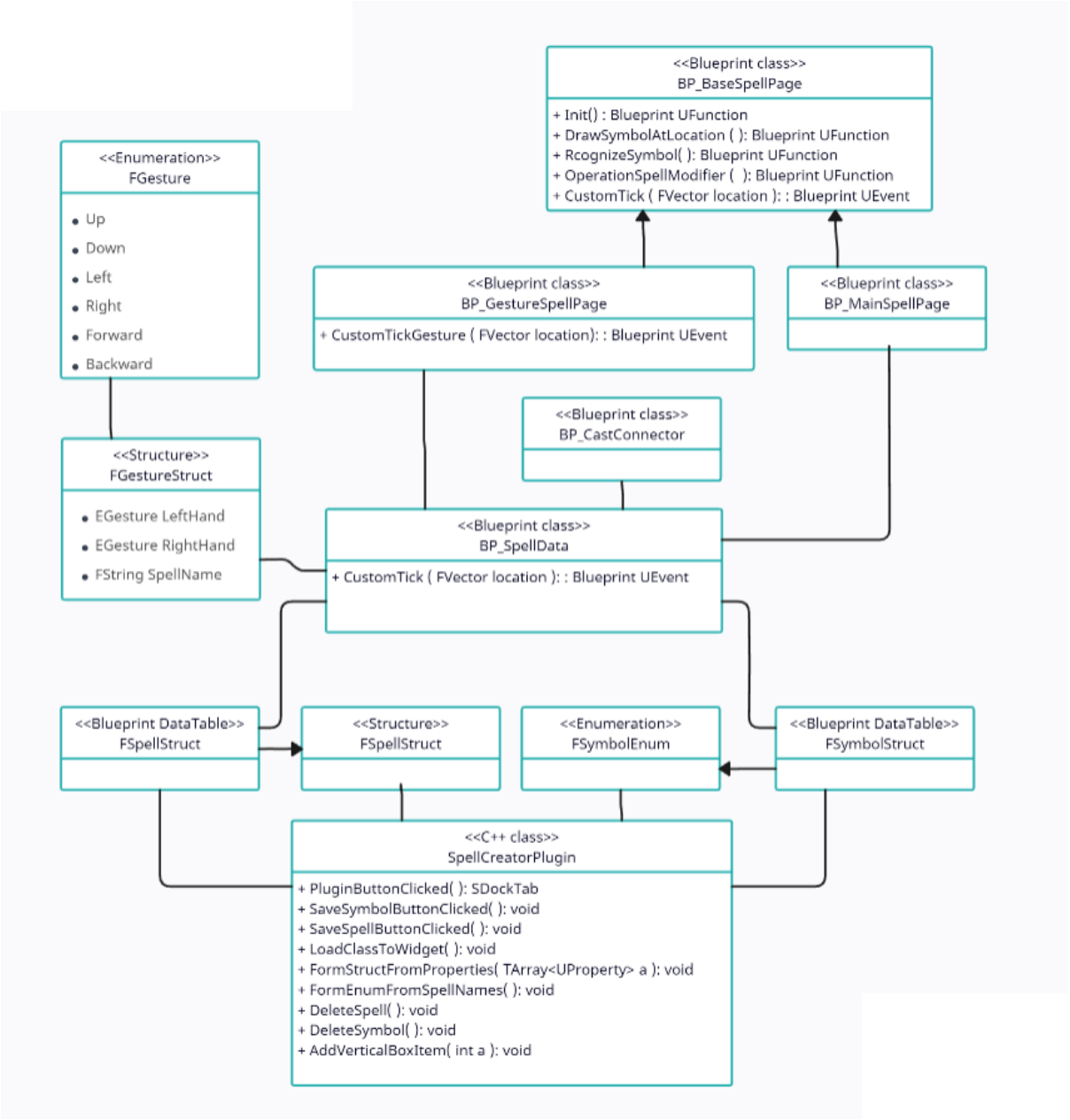



Рис. 2.2 Діаграма класів проекту

На діаграмі класів на рисунку 2.2 вказані всі класи бібліотеки, і як вони взаємодіють між собою. структури і Enum, які не мають атрибутів - динамічно заповнюються атрибутами під час вибору розробником базового класу заклинання.

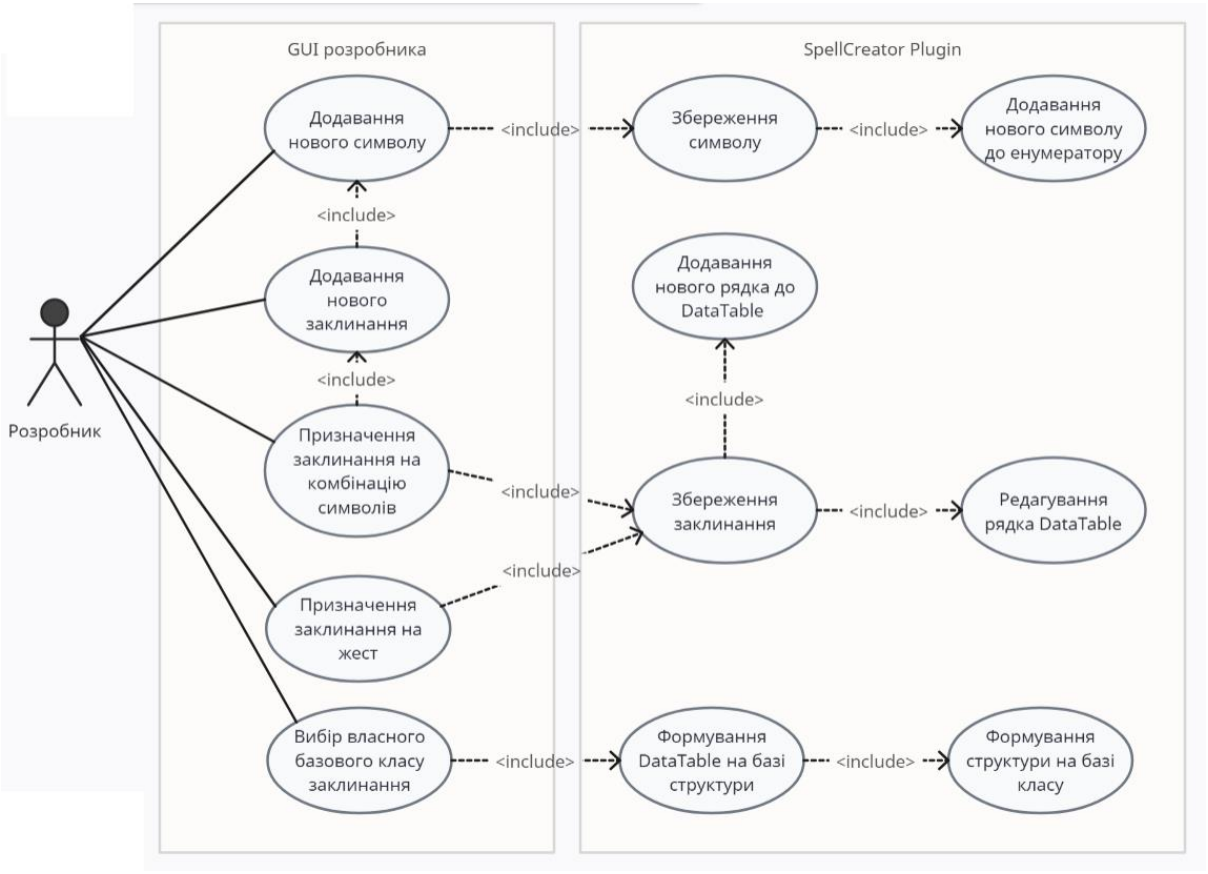



Рис. 2.3 Діаграма прикладів використання даної бібліотеки

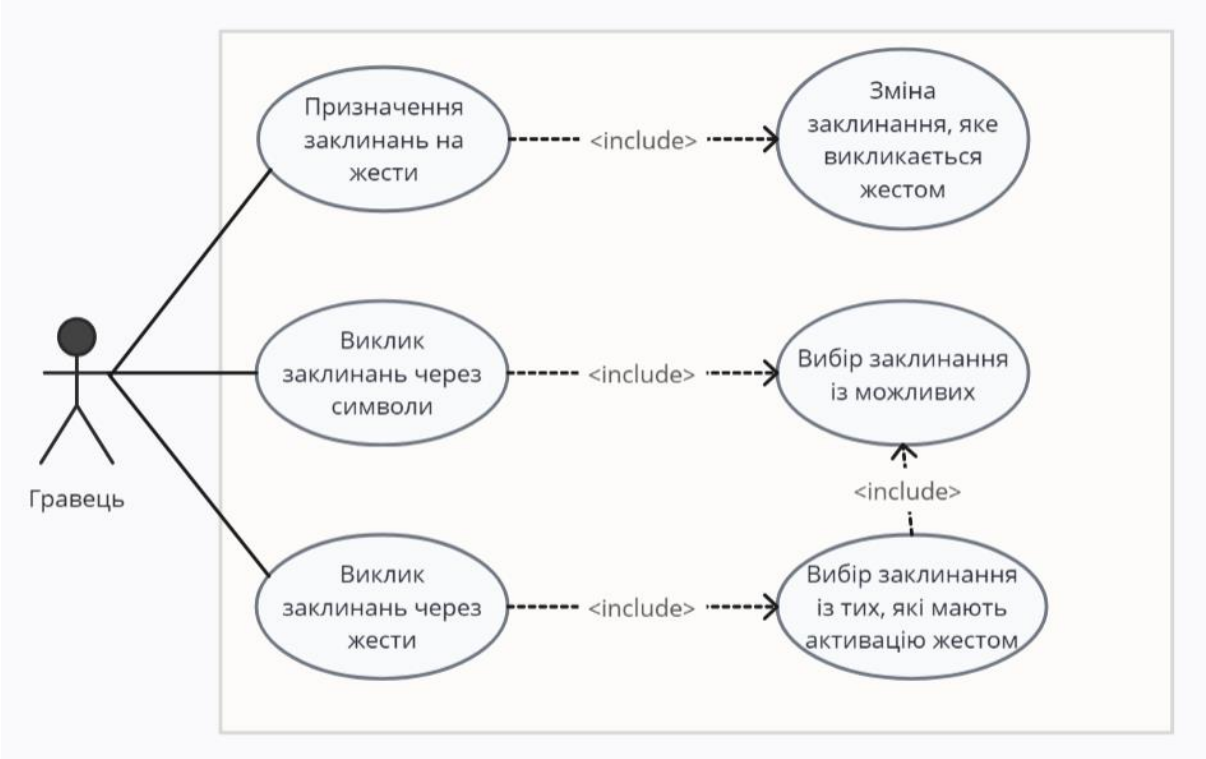



Рис.. 2.4 Діаграма прикладів використання гри, розробленої з використанням даної бібліотеки

На діаграмі класів на рисунку 2.3 зображена діаграма прикладів використання даної бібліотеки: тут показані всі функції, доступні розробнику, та на рисунку 2.4 зображена діаграма прикладів використання гри, розробленої з використанням даної бібліотеки - гравець може викликати заклинання і комбінацією символів, і жестом, а також динамічно назначити заклинання на жест, що доступно завдяки динамічній зміні DataTable під час роботи гри.

### 2.3 Розробка системи для розпізнавання символів

Для розпізнавання символів, намальованих гравцем було обрано використовувати Blueprints, тому що ця система має прямий доступ до гравця та ігрового світу, що є обов'язковими аспектами для розробки цього алгоритму.

Сам алгоритм працює наступним чином: коли гравець затиснув кнопку на контролері - кожен визначений проміжок часу зберігаються точки у просторі, де знаходиться ігровий контролер, і коли гравець відпустив кнопку - зібрані точки порівнюються з символами.

Також для візуалізації створено 3 анімації: перша - це яскравий слід за контролером під час введення символу, друга - це візуалізація успішного розпізнавання символу, яка виглядає, як коло, яке обводиться навколо намальованого символу, а третя - це візуалізація того, що введений гравцем символ не співпадає з жодним існуючим, і він виглядає, як вибух синіх частинок.

За розміщення ключових точок символів у ігровому світі відповідає клас BP_MainSpellPage, батьківським класом якого є BP_BaseSpellPage.

У батьківського класу BP_BaseSpellPage є ключові функції. та атрибути, які допомагають з налаштуванням BP_MainSpellPage, а саме Functions:

1. Function Init виконує налаштування візуальних ефектів та додає у список для порівнянь усі існуючі символи.
2. Function SpellModifier (virtual) повертає структуру заклинання, з доданим намальованим символом

3. Function DrawSymbolAtLocation створює яскравий слід за контролером.
4. Function Recognize Symbol порівнює точки у просторі, які пройшов контролер з ключовими точками символу (рисунок 2.2). Вона передає кожну збережену точку у Function Follow Spline, яка в свою чергу визначає декілька ключових моментів:
    a. Чи відстань між вхідною точкою та найближчою точкою на лінії символу не перевищує вказану відстань?
    b. Чи вхідна точка ближче до наступної контрольної точки?
    c. Чи вхідна точка досягла кінця символу?

Якщо відповідь на всі питання так, то по завершенню перевірки викликається CallBack “Success”, який сповіщає інші класи, що символ розпізнано. Якщо жоден символ не розпізнався - то викликається CallBack “Fail”, який сповіщає інші класи, що символ не розпізнано.

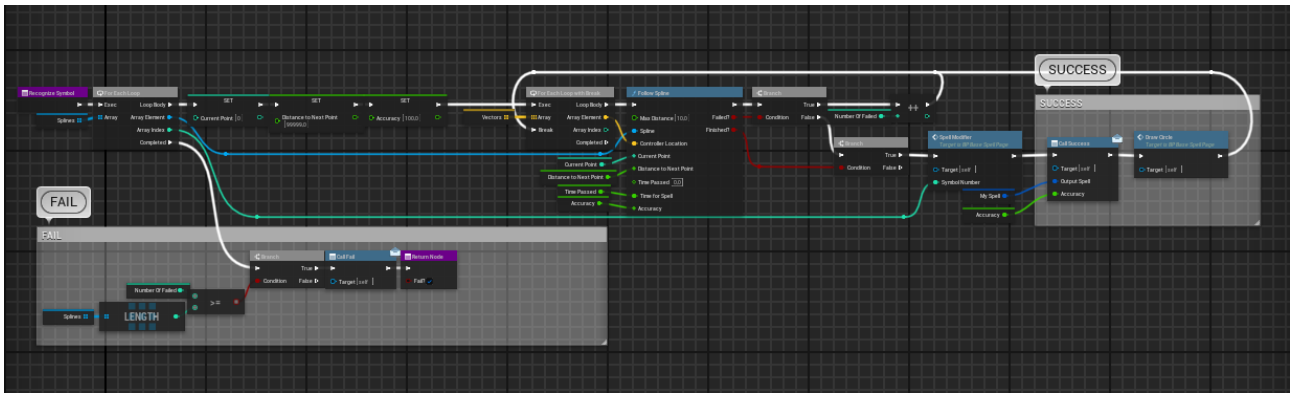



Рис. 2.5 Function “Recognize Symbol”

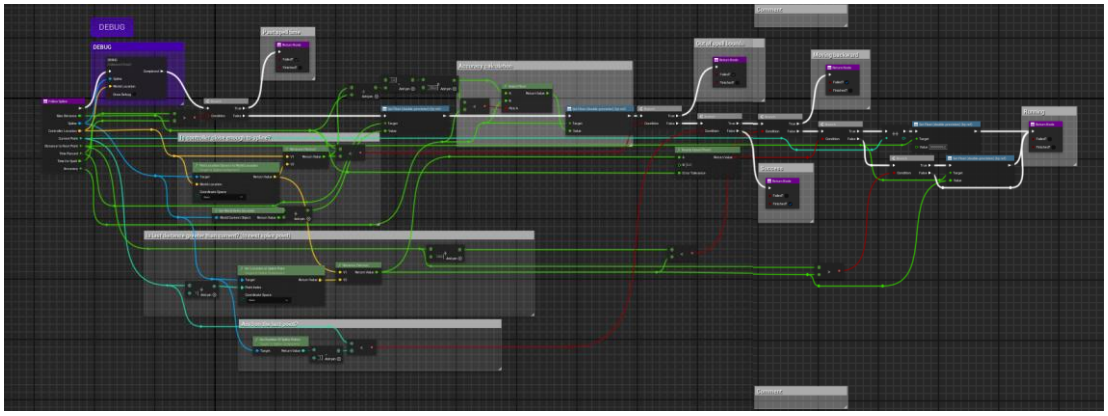



Рис. 2.6 Function “Follow Spline”

Клас BP_MainSpellPage створює всі символи з ключових точок, і викликає функцію Init, куди передає ці символи. Також у цьому класі перевизначена Function

SpellModifier для модифікації вхідного заклинання відповідно до вибраного символу.

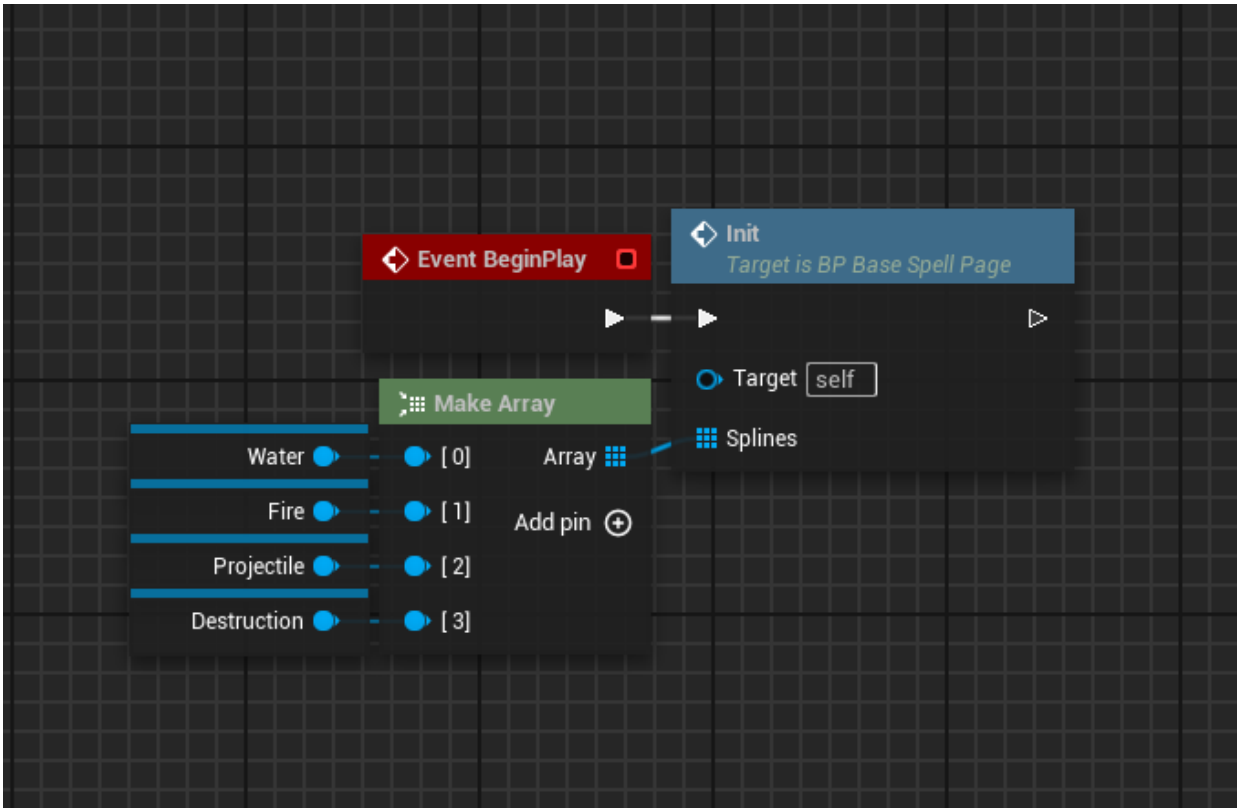



Рис. 2.7 Ініціалізація символів через Function Init у BP_MainSpellPage

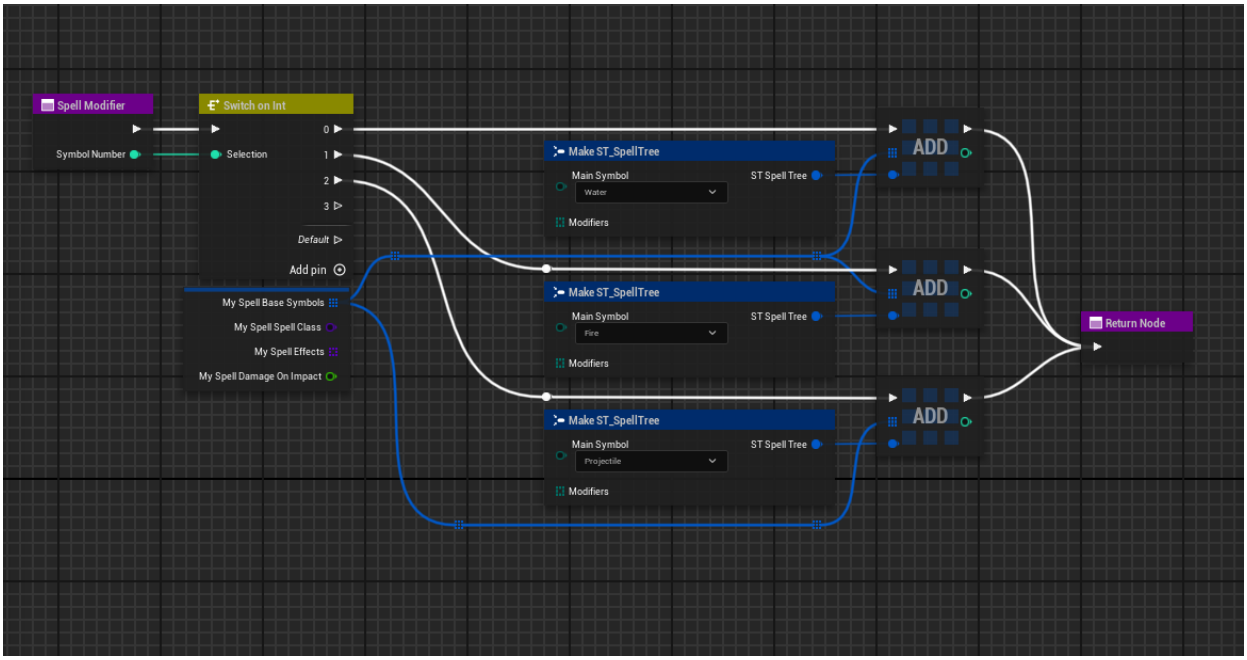



Рис. 2.8 Перевизначена Function SpellModifier у BP_MainSpellPage

Для можливості комбінації символів створений клас BP_SpellData та BP_CastConnector. Ці класи дозволяють гравцю вводити декілька символів підряд, і зберігають послідовність символів.

Алгоритм створення комбінації складається з декількох кроків:

1. Створення BP_MainSpellPage, і прив’язка CallBacks для отримання результату.
2. Після закінчення роботи BP_MainSpellPage збереження резутьтату, створення BP_CastConnector для візуального з’єднання попереднього та наступного символів, і видалення BP_MainSpellPage.
3. Повторення першого і другого пунктів, поки гравець не активує заклинання.

Цей алгоритм виконується через Event “CustomTick”, який знаходиться у BP_SpellData.

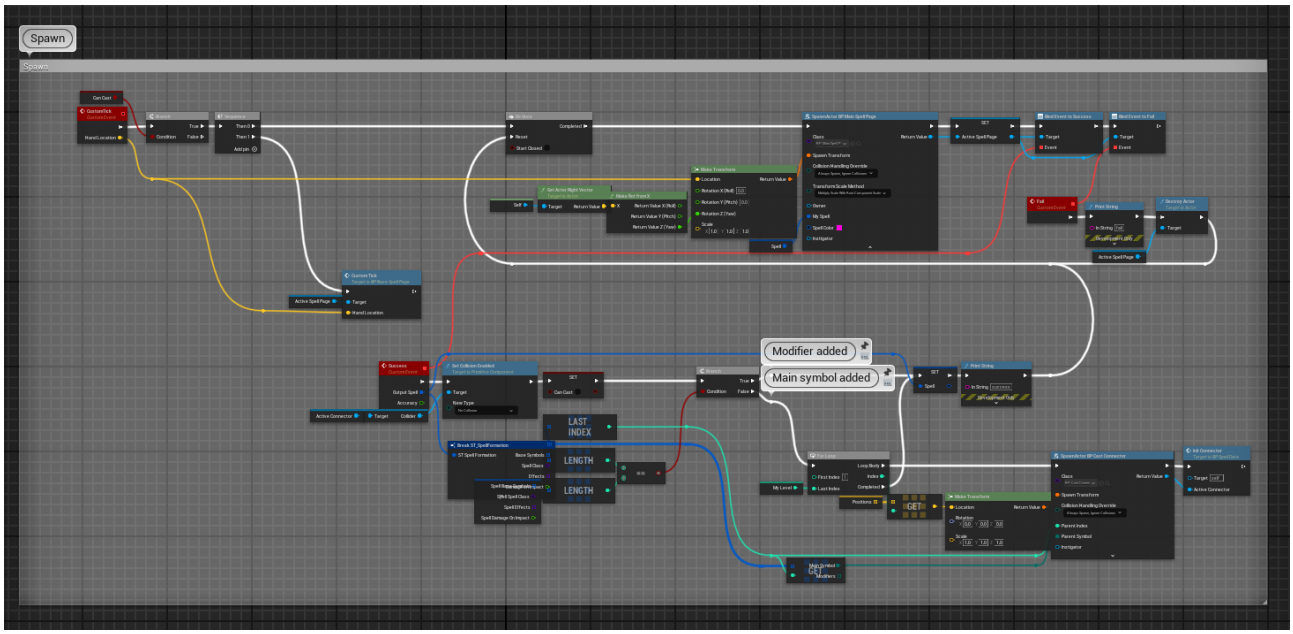



Рис. 2.9 Event CustomTick

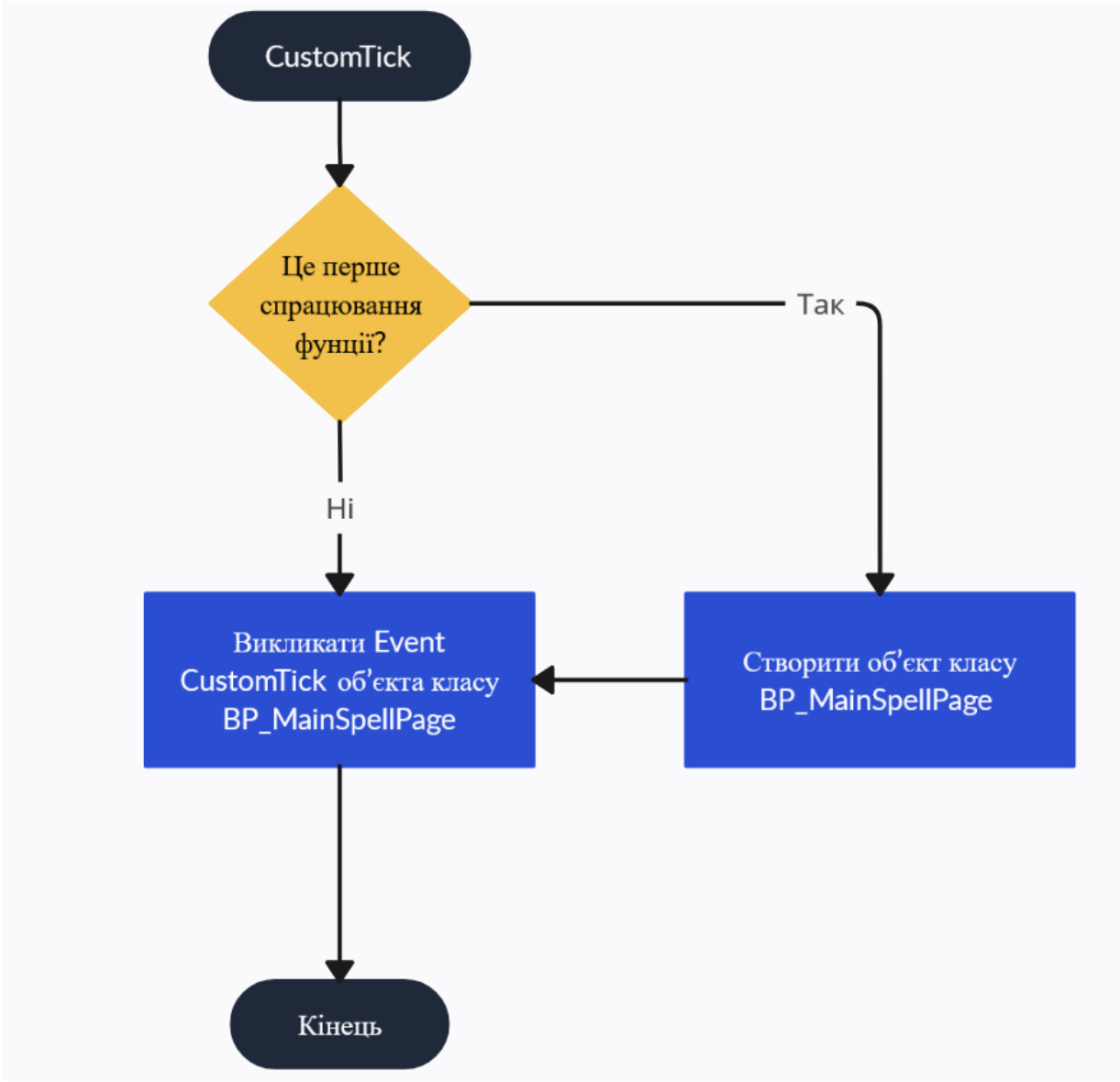



Рис. 2.10 Алгоритм роботи Event CustomTick у BP_SpellData

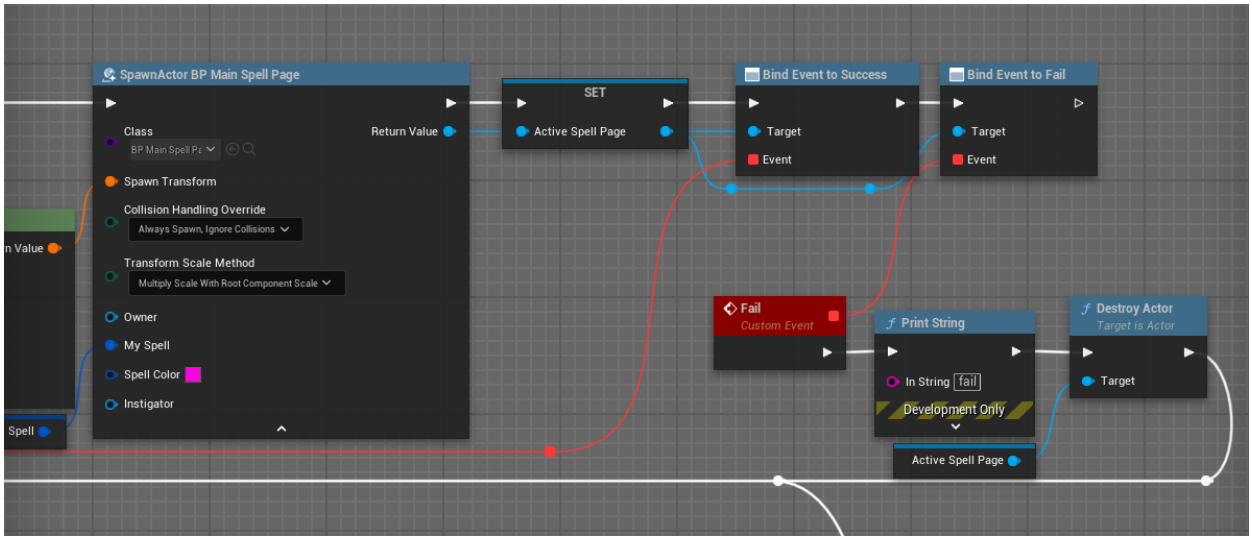



Рис. 2.11 Прив'язка Event "Fail", як CallBack Fail у BP_MainSpellPage

## 2.4 Розробка системи для розпізнавання жестів

У реалізованій системі жести представляють собою прості рухи контролера вверх, вниз вліво, вправо, вперед, або назад. Розпізнавання жестів відбувається за таким самим принципом, як і розпізнавання жестів, тільки програма не зберігає пройдені точки контролера, а зразу порівнює їх із шляхом, який потрібно пройти контролеру для спрацювання жеста.

BP_GestureSpellPage - це клас, теж дочірній для BP_BaseSpellPage, як і BP_MainSpellPage, тому наслідує всі основні методи та поля для виконання основних обчислень по розпізнаванню символів, і, в даному випадку, жестів.

BP_GestureSpellPage , на відміну від BP_MainSpellPage створюється для обох ігрових контролерів, і розпізнає обидва жести зразу, не зберігаючи точки, що значно зменшує затримку, і позитивно впливає на швидкість виклику заклинання. Як тільки гравець натиснув клавішу активації заклинання та виконав жест кожною рукою - система викликає заклинання, яке призначене на цей жест.

Enumerator EGesture містить шість елементів:

1. Up - позначає жест контролера знизу вгору
2. Down - позначає жест контролера згори вниз
3. Left - позначає жест контролера справа наліво
4. Right - позначає жест контролера зліва направо
5. Forward - позначає жест контролера ззаду вперед
6. Backward - позначає жест контролера спереду назад

Розпізнавання відбувається за рухом контролера по трьом осям координат, які фіксують положення контролера під час початку жесту, і слідкують за вектором його руху. Дані записуються у змінну структури GestureStruct, яка має поля EGesture, LeftHand, EGesture RightHand, FString SpellName. Порівнюючи дані цієї змінної з DataTable, який містить у собі всі заклинання, і методи їх виклику - можна знайти і викликати заклинання, викликане цим жестом.

У класі BP_SpellData, який має Event CustomTickGesture з двома вхідними полями типу FVector, куди передається інформація про положення контролера у

просторі кожен фіксований проміжок часу, викликається два Event, які є у BP_GestureSpellPage, і безпосередньо відповідають за розпізнавання жестів, як показано на рисунку 2.12.

При створенні об’єктів класу BP_GestureSpellPage - відбувається прив’язка Event “SuccessGestureLeft”, як CallBack SuccessGesture у BP_GestureSpellPage. Тобто при успішному жесті лівою рукою виконується Event “SuccessGestureLeft”. У ньому виконаний жест додається в структуру. Аналогічно відбувається прив’язка Event “SuccessGestureRight” для правої руки.

Але Event “GestureFail”, який прив’язується на CallBack Fail - видаляє обидва об’єкти BP_GestureSpellPage, оскільки якщо одна рука не виконала свій жест, товесь жест не спрацьовує.

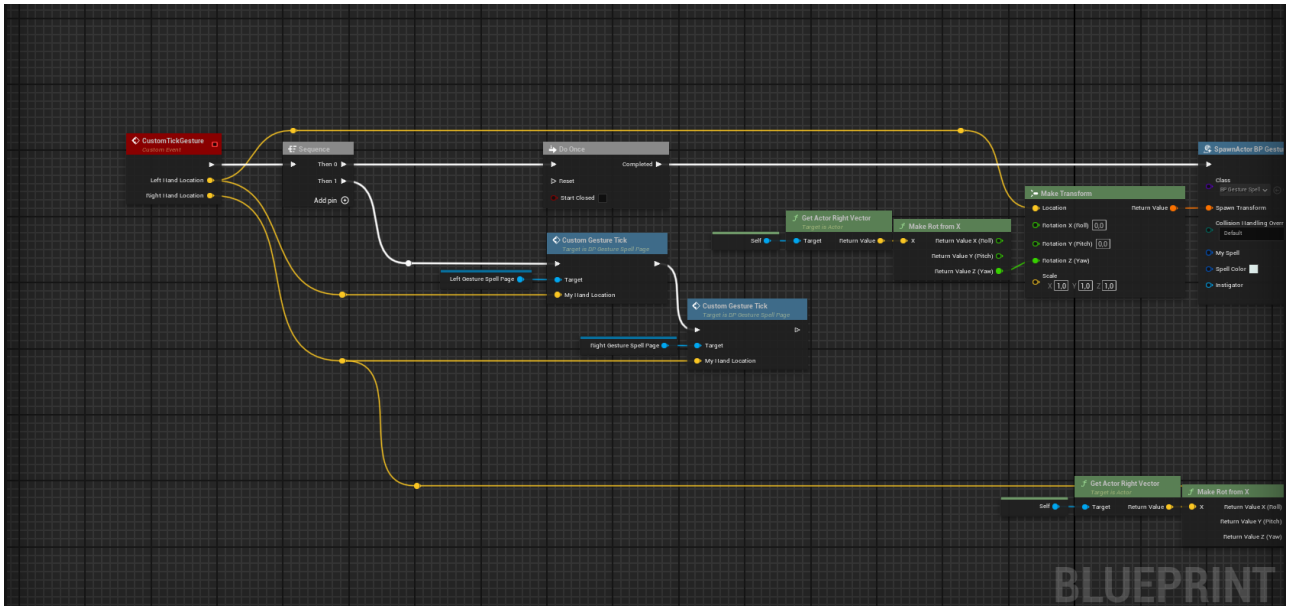



Рис. 2.12 Перша частина Event CustomTickGesture

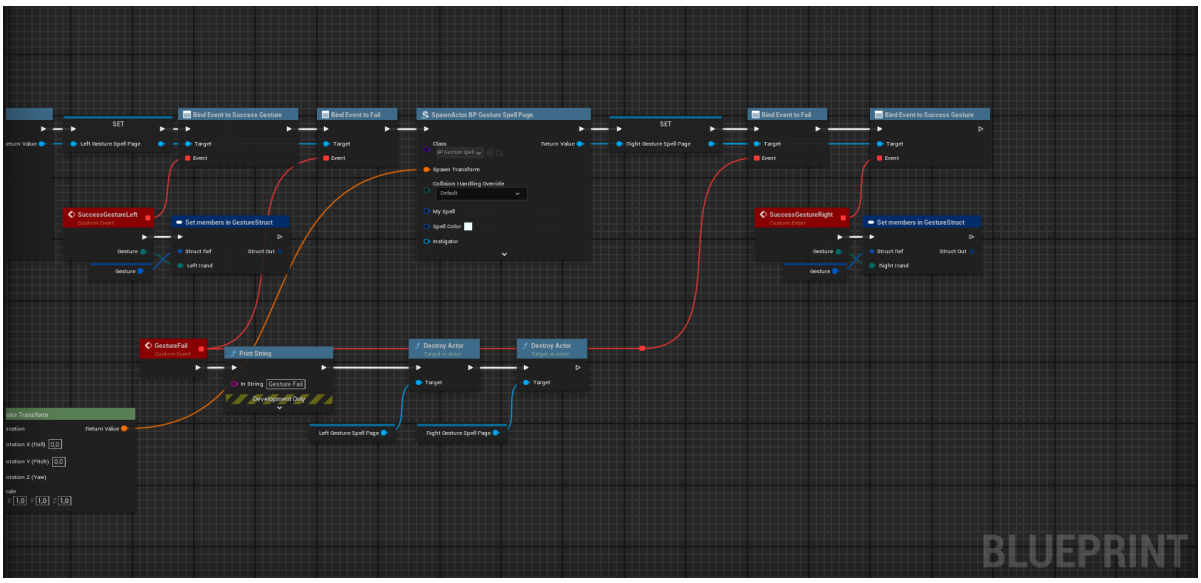



Рис. 2.13 Друга частина Event CustomTickGesture

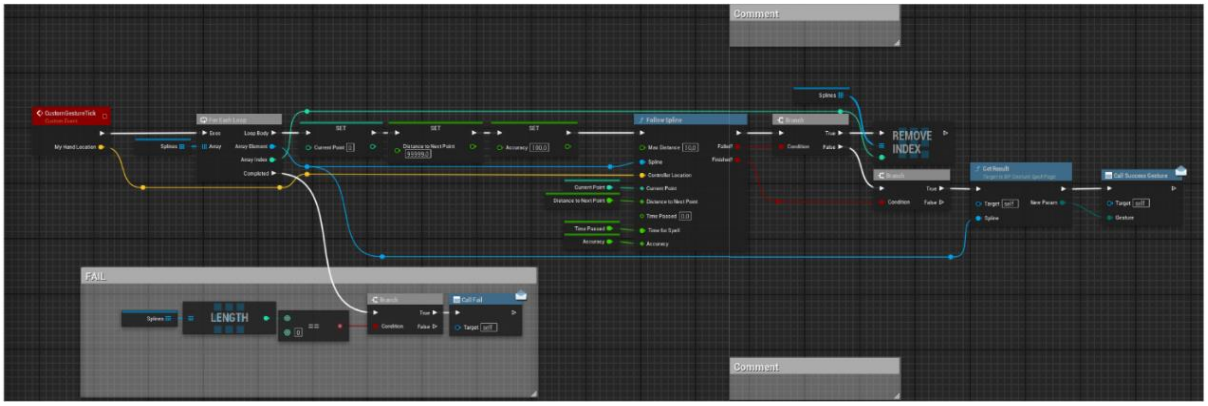



Рис. 2.14 Розпізнавання символу у BP_GestureSpellPage

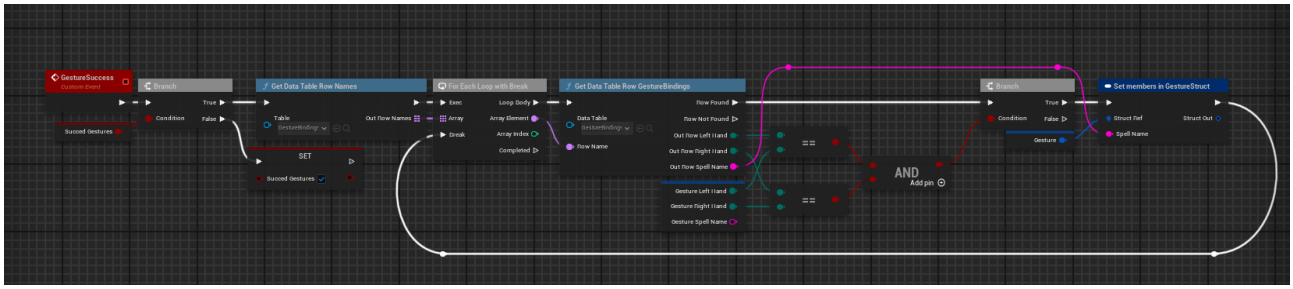



Рис. 2.15 Event GestureSuccess

Для того, щоб заклинання викликалось зразу після закінчення жеста - було створено Event GestureSuccess, який виконує дві функції:

1. Перевіряє, чи обидва жести виконані коректно
2. Здійснює пошук заклинання, відповідно до виконаних жестів у DataTable, який зберігає всі комбінації жестів

Алгоритм роботи Event GestureSuccess відображений на рисунку 2.15

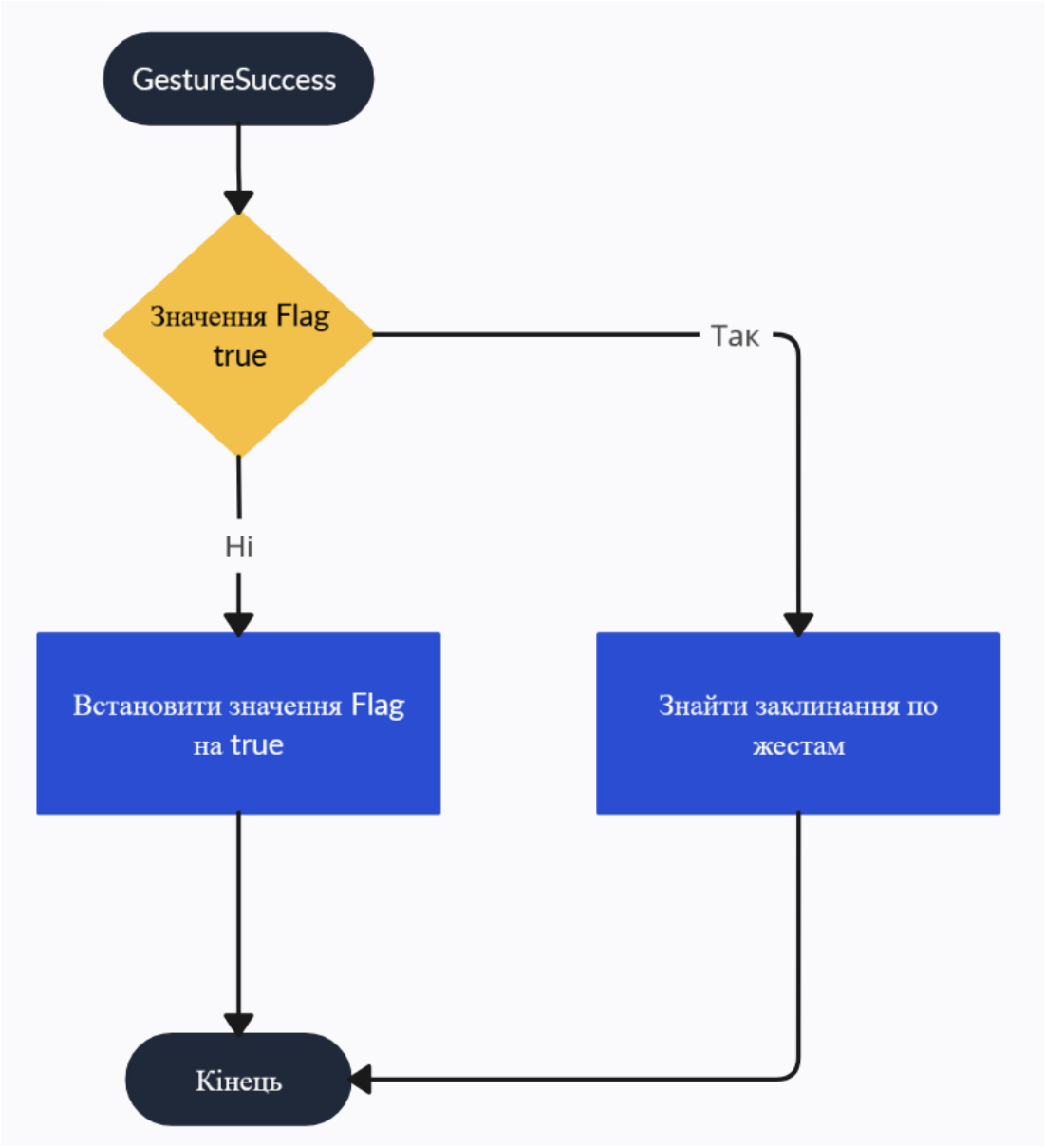



Рис. 2.16 Алгорити роботи Event GestureSuccess

### 2.5 Розробка інтерфейсу для системи введення символів

При проектуванні інтерфейсу для системи введення символів було враховано декілька ключових факторів, які впливають на досвід користувача, а саме:

1. Розташування всіх органів керування для інтуїтивного розуміння інтерфейсу
2. Співпадіння стилю інтерфейсу з загальним стилем середовища розробки Unreal Engine для того, щоб користувачу, при переключенням між вкладками

середовища розробки Unreal Engine не прийшлось відволікатись від робочого процесу, і перелаштовуватись під інший інтерфейс.

3. Система Drag-and-drop для максимально зручного налаштування символа
4. Візуальне відображення символа для створення контрольних точок поверх картинки, що спрощує додавання символа
5. Візуальне розділення розділів для логічності та послідовності заповнення інформації користувачем

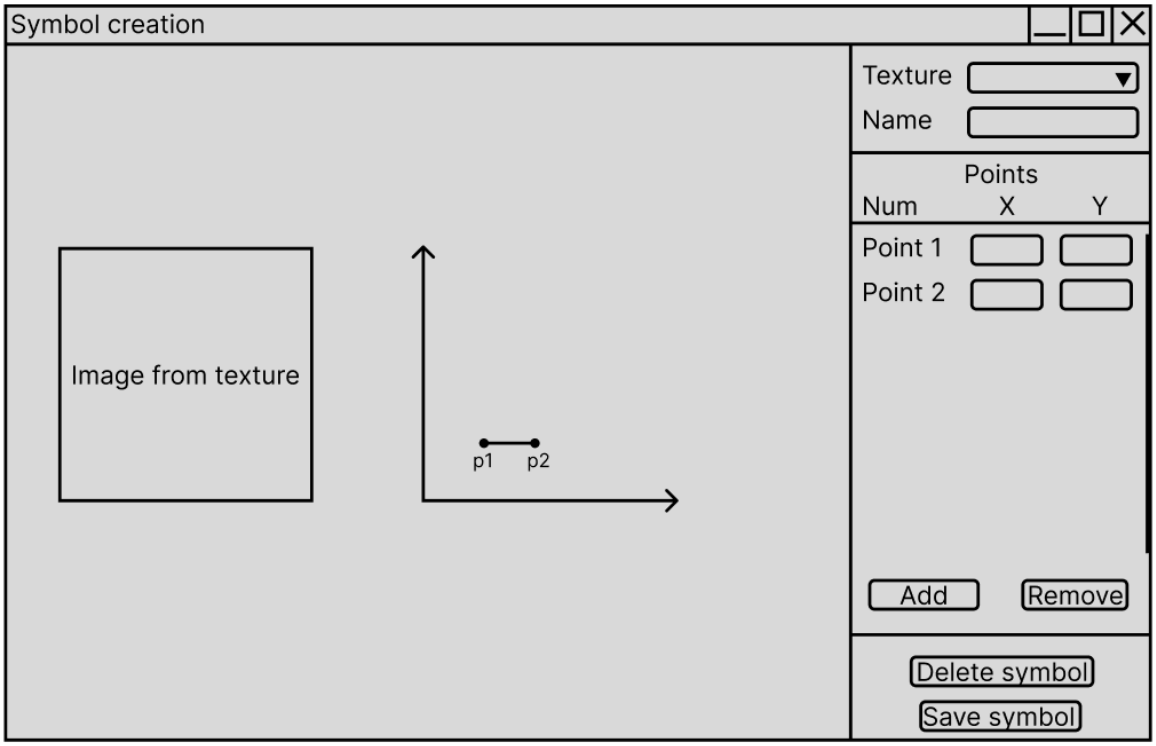



Рис. 2.17 Проектування дизайну інтерфейсу для системи введення символів

При додаванні символа користувач повинен вибрати зображення символу, назву символу, та додати контрольні точки. Контрольні точки додаються або через вказання їх координат напряму, або через систему Drag-and-drop, де користувач курсором вказує місця точок для символів.

Зона редактора символу займає більшу частину вікна, щоб користувач міг створювати складні символи, і йому вистачило місця для будь якого символу. Також відмітки на координатній прямій відображають сантиметри - основну систему вимірювання відстані в Unreal Engine. Це зроблено для того, щоб користувач розумів, наскільки великим, або малим символ буде після збереження.

Зона налаштування символу також має зону для зображення, яку вибирає користувач. При виборі зображення у полі “Texture” це зображення відображається з лівого боку від координатної сітки. Це надає користувачу можливість обвести цей символ точками зразу над зображенням цього символу. Також це зображення використовується у відображенні саме цього символу на подальших вікнах та безпосередньо у грі.

У поле “Name” приймається назва символу, під яким він зберігається у Enum. Ця назва в подальшому буде використовуватись для позначення цього конкретного символу.

Зона додавання точок “Points” - це зона із прокруткою, для того, щоб користувач міг додавати бажану кількість точок, і мати доступ до їх редагування. При пересуванні точок курсором - значення позиції змінюється динамічно. При зміні координат через вписування чисел вручну - ці зміни також динамічно відображаються на редакторі.

Кнопка “Add point” додає поле з номером точки та полями X та Y до зони з прокруткою, а також додає нову точку на поле для редагування в позицію (0, 0) відносно сітки координат. Також остання додана точка з’єднується лінією з новою доданою точкою.

Кнопка “Delete point” видаляє поле з номером точки та полями X та Y із зони з прокруткою, і видаляє останню додану точку з зони редактора та лінію між останньою та передостанньою точкою.

Кнопка “Save symbol” зберігає символ до власного DataTable, створеного на основі незмінної структури. Також вона додає назву символа до Enum, і запускає перекомпліляцію проекту та HotReload для того, щоб користувачу не потрібно було перезапускати середовище, і що забезпечує безперервну роботу в Unreal Engine.

Кнопка “Delete symbol” Викликає вікно - форму видалення символу, макет якого зображений на рисунку 2.18. Через дану форму є можливість видалити символ із DataTable. Також символ видаляється з Enum, і видаляються всі комбінації та заклинання, які асоціюються з цим символом, або містять його як свій компонент.

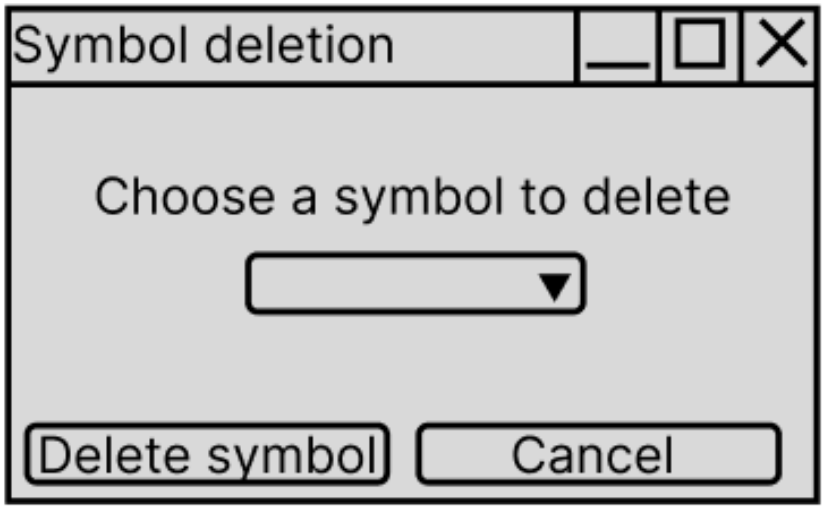



Рис. 2.18 Дизайн інтерфейсу для видалення символів

У даному вікні є поле вибору символу та дві кнопки: одна підтверджує видалення символу, а інша скасовує видалення, та повертає користувача до попереднього вікна - редактора символу.

У випадаючому списку для вибору символу для видалення присутні назви усіх символів, і при натисканні кнопки “Delete symbol” відбувається алгоритм видалення, описаний вище.

### 2.6 Розробка інтерфейсу для системи введення комбінацій і призначення на них заклинання

Для створення асоціації з комбінацією символів, жестом і заклинанням потрібні наступні дані:

1. Масив символів, при комбінації яких створиться заклинання
2. Жест правої і лівої руки, які викличуть це ж заклинання
3. Опис самого заклинання - що воно робить, і як впливає на навколишнє середовище.
4. Базовий клас, на основі якого формується заклинання

При дизайні користувацького інтерфейсу було вирішено створити логічний ланцюжок дій для заповнення всієї необхідної інформації, і тому весь інтерфейс

було розділено на 3 логічні розділи, які заповнюються зліва направо, і кожен розділ заповнюється зверху вниз. На даній формі існують наступні логічні розділи :

1. Вибір базового класу
2. Засоби виклику заклинання
3. Характеристики заклинання

На рисунку 2.19 зображений макет форми введення всієї необхідної інформації та керування уже існуючими заклинаннями.

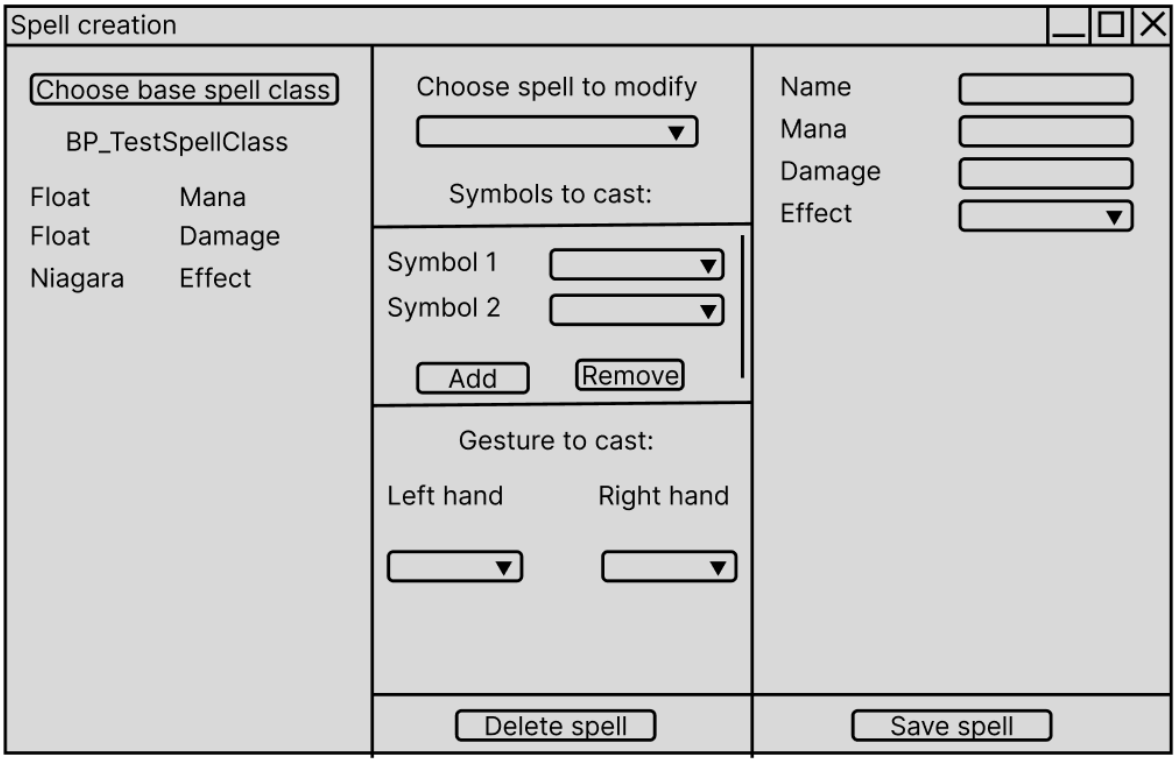



Рис. 2.19 Дизайн інтерфейсу для системи введення комбінацій і призначення на них заклинання

Перш за все, користувачу потрібно вибрати базовий клас заклинання. Це робиться першим кроком, тому поле для заповнення знаходиться з лівого боку у першому розділі форми. При натисканні на кнопку “Choose base spell class” повинен відкриватися стандартна, вбудована в рушій, форма для вибору класу, дочірнього AActor.

Після того, як користувач підтвердив вибір класу у формі, що з’явилася - на лівій частині інтерфейсу з’явиться назва вибраного класу, та у стовець додадуться

всі змінні, виділені користувачем. Кожна змінна виводиться в рядок разом зі своїм типом даних.

Після цих змін бібліотека запустить компіляцію проекту, оскільки ключові C++ файли зі структурами були оновлені, і запропонує перезапустити рушій. Перезапуск рушія не є обов'язковим кроком, і користувач може відмовитись, але для правильного сприйняття рушієм змін, зазвичай, HotReload не достатньо.

Наступним кроком додавання заклинання є введення способів його виклику. Для цього користувачу потрібно перейти до наступного логічного розділу на інтерфейсі форми, який знаходиться по центру.

Користувач може вибрати вже існуюче заклинання для його модифікації із випадаючого списку. Якщо користувач натисне на елемент випадаючого списка, в якому знаходяться назви заклинань - то всі дані цього заклинання завантажуться у форму, і будуть доступні для редагування. Для зміни уже існуючого заклинання достатньо завантажити його, змінити необхідні дані, і натиснути кнопку "Save spell". Також можливо створити нове заклинання з нуля, заповнюючи відповідні поля форми.

Під час створення заклинання користувач повинен визначити, скільки символів будуть активувати це заклинання. Тоді він може натиснути кнопку "Add", яка додає символ до масиву необхідних для активації символів, та додає у зону для прокрутки на формі елемент, який містить номер символа та випадаючий список. У випадаючому списку напроти номера символа знаходяться всі елементи Enum EGesture. Користувач повинен вказати у всі додані поля символи, потрібні для виклику заклинання.

Кнопка "Remove" видаляє останнє додане поле з зони для прокрутки елемент із символом, і елемент з масиву, у якому зберігаються дані.

Наступним кроком користувач повинен визначити жести для кожної руки, які активують заклинання. Оскільки комбінацій жестів набагато менше, ніж комбінацій символів, то ці два поля - випадаючий список під Left hand і випадаючий список під Right hand можна залишити пустими.

Після цього користувач повинен визначити характеристики заклинання, яке він призначає на дану комбінацію. Обов’язкове поле для заповнення - яке існує завжди, не залежно від користувацьких полів - це назва заклинання. Під цією назвою воно зберігається у DataTable. Усі поля узяті з базового класу, визначеного користувачем. Також всю логіку самого заклинання визначає користувач у функціях базового класу. Користувач може додавати нові поля для специфічних параметрів заклинання, що надає йому можливість створювати унікальні та складні заклинання.

При виклику заклинання - об’єкт базового класу створюється у світі, і йому передаються параметри, які вказані користувачем на дане викликане заклинання.

Для збільшення зручності користувача додані інтуїтивно зрозумілі підказки, які допоможуть орієнтуватися в процесі створення та редагування заклинань. Такі елементи значно спростять процес налаштування нових заклинань.

### 2.7 Реалізація системи введення символів

Система введення символів дозволяє користувачам легко додавати нові символи до проекту через спеціальний інтерфейс. Користувач починає процес з вибору зображення, яке буде використовуватись для візуалізації символу. Потім вводиться назва символу, яка зберігається в Enum. Наступний крок полягає у додаванні контрольних точок, які можуть бути додані вручну через введення координат або шляхом використання системи Drag-and-drop для більш інтуїтивного розміщення точок на зображенні.

Основними функціями, що використовуються для реалізації системи введення символів, є функції додавання та видалення символів, а також збереження введених даних. Функція додавання символу створює новий елемент у графічному інтерфейсі, який містить зображення символу, його назву та координати контрольних точок. Функція видалення символу дозволяє користувачеві вибрати

символ зі списку існуючих і видалити його, що включає видалення символу з Enum та з DataTable.

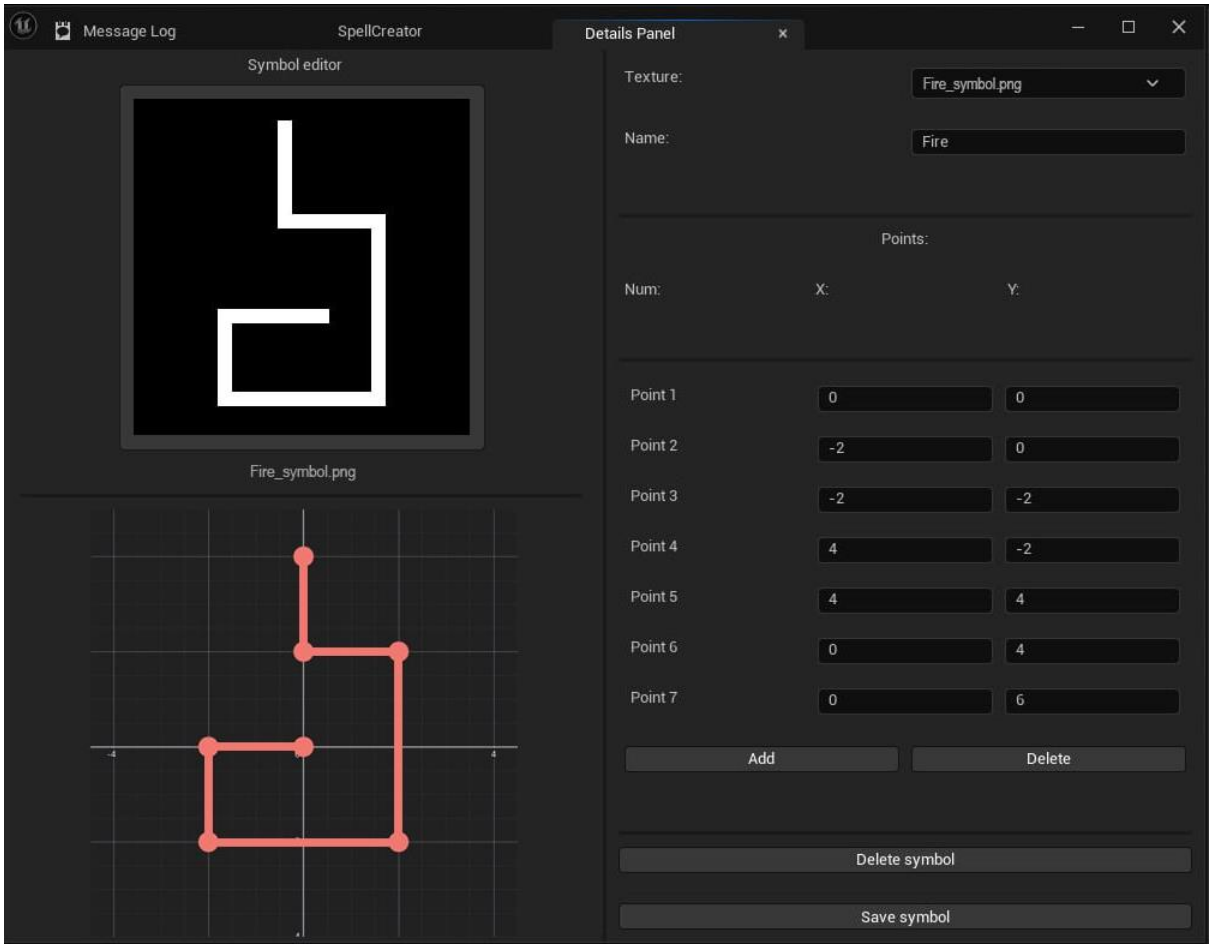



Рис. 2.20 Дизайн інтерфейсу для системи введення комбінацій і призначення на них заклинання

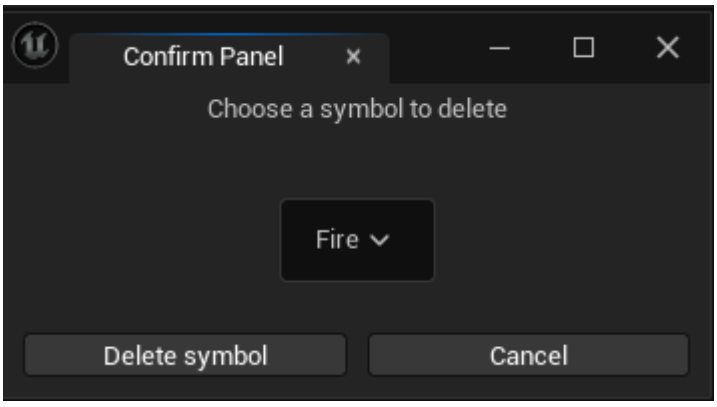



Рис. 2.21 Форма видалення символу

Форма для видалення містить випадаючий список, який заповнюється динамічно, при виклику форми. Елементи випадаючого списку - це назви символів, які збережені у Enum FSymbolEnum.

Всі дані, введені користувачем, зберігаються в спеціальній структурі DataTable, яка містить інформацію про кожен символ, включаючи його зображення, назву та координати контрольних точок. Додатково, назви символів зберігаються у Enum, що забезпечує їх швидкий доступ і використання в інших частинах проекту. Після збереження символу система автоматично запускає перекомпіляцію проекту та HotReload, що дозволяє користувачам продовжувати роботу без необхідності перезапуску середовища розробки Unreal Engine.

### 2.8 Реалізація системи введення комбінацій і призначення на них заклинання

При введенні комбінацій та призначенні на них заклинань найголовніше - сформувати структуру з правильними атрибутами. Це потрібно для того, щоб на основі цієї структури створити DataTable для збереження у нього усіх потрібних даних, які потрібні для формування заклинання.

Сама структура повинна формуватися з визначених користувачем полів, які він вказав у базовому класі заклинання. Для ідентифікації використовується категорія змінних. Сигнатура тих змінних, які закріплені користувачем за категорією “KeyParam” копіюється, і додається до структури.

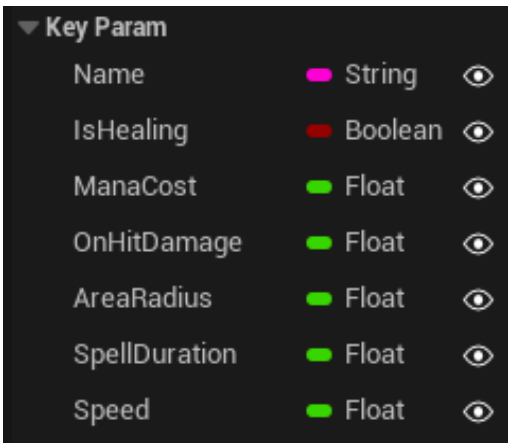



Рис. 2.22 Змінні в категорії “KeyParam”

Структура створюється і змінюється напряму у Header файлі бібліотеки. Це дає максимальний рівень доступу та можливість міняти або додавати будь яку змінну або тип даних.

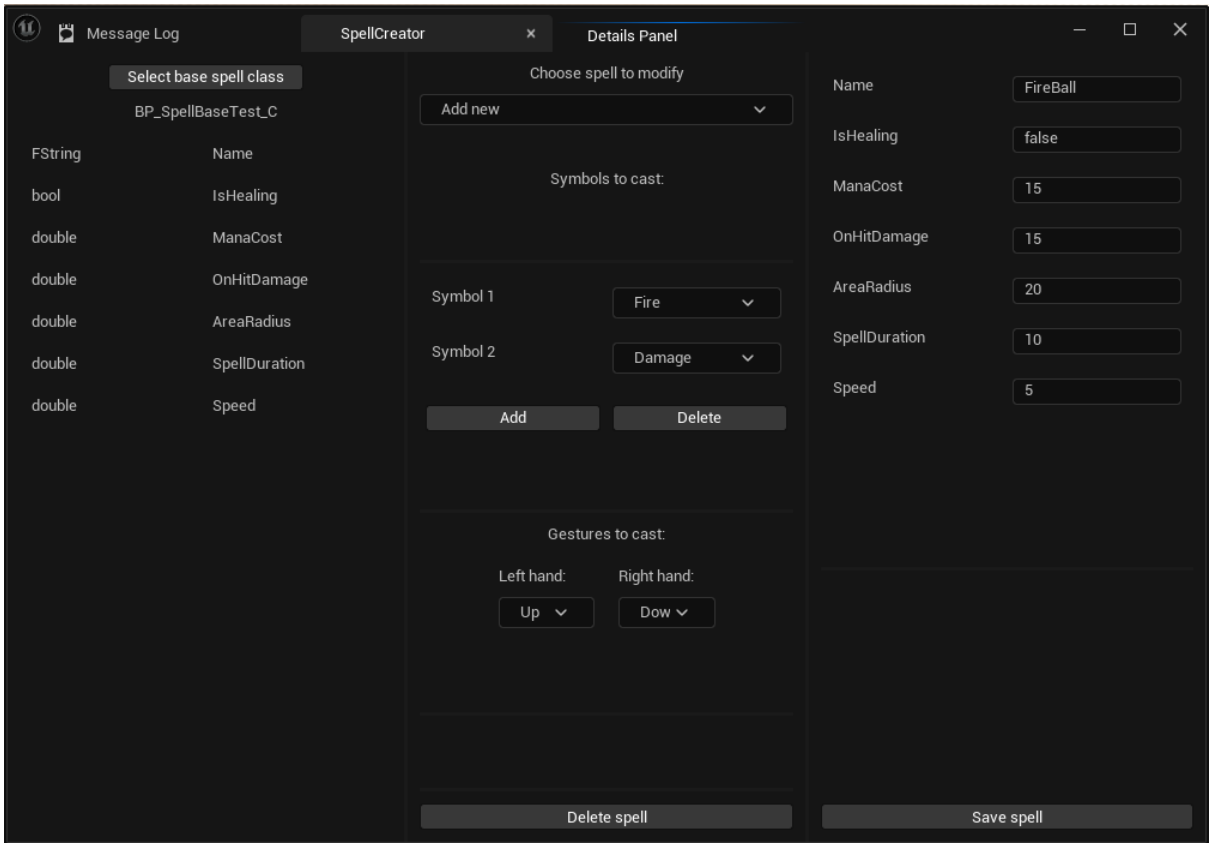



Рис. 2.23 Інтерфейс для введення комбінацій і призначення на них заклинання

Також назва і шлях до класу зберігається у Config-файлі. Config-файли не очищуються після перезапуску середовища розробки, на відміну від звичайних локальних змінних, які не зберігають дані між сесіями розробки.

При завантаженні середовища розробки запускається розроблений плагін, і виконує завантаження усіх даних до віджету. Також він порівнює поля базового класу попереднього завантаження і ці ж поля зараз. Якщо змінилась їх кількість - він коригує дані в DataTable, і запускає HotReload, який, у свою чергу, дозволяє користувачу не перезапускаючи середовище розробки продовжувати роботу з уже оновленими DataTable та усіма структурами.

У випадаючі списки завантажується збережена інформація таким чином, що кожен елемент меню випадаючого списку - це назва елемента Enumerator. Ці дані зберігаються у декількох місцях, і програма використовує найпростіший метод

доступу до них - через DataTable SymbolPoints. У цьому DataTable зберігається назва символу, його зображення, і масив FVector2D - координат у площині, які через функцію перетворюються у координати у тривимірному прсторі для безпосередньої роботи з гравцем.

### 2.9 Реалізація системи активації викликаного заклинання

Активація заклинання відбувається після того, як гравець виконав жест, потрібний для активації заклинання, або виконує потрібну комбінацію символів контролером у повітрі.

Після розпізнавання символів або жестів, виконаних гравцем - BP_SpellData викликає функцію CompareSpells, яка знаходиться у Function_library - спеціальний клас у Unreal Engine, створені функції у якому є глобальними, і доступні всім об’єктам, які існують на сцені.

# 3 ТЕСТУВАННЯ

Головною метою тестування програмного забезпечення є перевірка того, чи відповідає фактична поведінка програми очікуваній. Це досягається шляхом виконання тестів із попередньо визначеним набором даних та параметрів. Тестування є одним з методів забезпечення якості програмного забезпечення. У цьому розділі розробляються тест-кейси для перевірки функціональності бібліотеки та здійснюється тестування відповідно до плану.

## 3.1 Розробка тест кейсів

Тестові випадки (тест-кейси) були створені для перевірки відповідності функцій системи встановленим вимогам. Залежно від очікуваного результату, тест-кейси можуть бути позитивними або негативними:

- Позитивні тест-кейси використовують тільки коректні дані та перевіряють, чи виконує додаток функцію, для якої він призначений.
- Негативні тест-кейси включають як коректні, так і некоректні дані (з принаймні одним некоректним параметром) для перевірки виняткових ситуацій (спрацьовування валідаторів) і переконання, що функція не виконується при спрацьовуванні валідатора.

Структура тест-кейсу включає розділи: "Action" (дія або послідовність дій, що виконуються під час тестування), "Expected Result" (очікуваний результат) та "Test Result" (фактичні результати роботи функції, часто позначені як "passed/failed/blocked").

Наступніі тест-кейси мають на меті протестувати основні функції розробленої бібліотеки. Результати тестування наведені у таблиці нижче.

Таблиця 3.1

| № | ОК/ NOK | Дії | Очікуваний результат |
|---|---|---|---|
| 1 | ОК | 1. Відкрити інтерфейс додавання символу.<br>2. Вибрати зображення символу.<br>3. Ввести назву символу.<br>4. Додати контрольні точки.<br>5. Натиснути кнопку "Save symbol". | Новий символ зберігається в DataTable та Enum. |
| 2 | ОК | 1. Відкрити інтерфейс видалення символу.<br>2. Вибрати символ зі списку.<br>3. Натиснути кнопку "Delete symbol". | Символ видаляється з DataTable та Enum. |
| 3 | ОК | 1. Відкрити інтерфейс додавання комбінації.<br>2. Вибрати символи для комбінації.<br>3. Задати параметри заклинання.<br>4. Натиснути кнопку "Save spell". | Нове заклинання зберігається в DataTable. |
| 4 | ОК | 1. Відкрити інтерфейс додавання комбінацій.<br>2. Вибрати заклинання зі списку.<br>3. Натиснути кнопку "Delete spell". | Заклинання видаляється з DataTable. |
| 5 | ОК | 1. Вибрати комбінацію для редагування.<br>2. Змінити комбінацію або параметри заклинання.<br>3. Натиснути кнопку "Save spell". | Зміни зберігаються в DataTable. |
| 6 | ОК | 1. Відкрити інтерфейс додавання комбінації.<br>2. Змінити базовий клас заклинання.<br>3. Перезапустити вередовище розробки. | Новий базовий клас заклинання зберігся, і структура змінилась, відповідно до його атрибутів. |

# ВИСНОВКИ

1. Проведено детальний аналіз предметної області та існуючих рішень у сфері ВР ігор та магічних систем. Це дозволило визначити основні переваги та недоліки бібліотеки

2. Визначено функціональні та нефункціональні вимоги до системи. До функціональних вимог віднесено

1. Користувач може створювати комбінації символів з допомогою UI методом Drag-and-drop вибираючи символи зі списку існуючих
2. Користувач може призначати заклинання на комбінації символів з допомогою UI, де вибирає бажані ефекти від заклинання
3. Інструмент, з допомогою якого можна створювати комбінації символів повинен мати можливість редагувати уже створені комбінації

Нефункціональні вимоги включають

1. Введення нового символу не повинно займати більше 3 хвилин
2. Створення комбінації символів та призначення на неї заклинання не повинно перевищувати 5 хвилин

3. Виконано проектування програмного забезпечення, включаючи проектування користувацьких інтерфейсів та проектування класів та взаємодії класів бібліотеки.

4. Вибрано технічні засоби для реалізації системи, включаючи Figma для проектування інтерфейсів, Unreal Engine як основний рушій для гри, та C++ для реалізації функціоналу бібліотеки.

5. Розроблено інструмент для додавання символів активації, інструмент для створення заклинання і інструмент для призначення жесту і комбінації символів на заклинання.

6. Проведено мануальне тестування користувацького інтерфейсу бібліотеки, що дозволило перевірити функціональність та стабільність роботи компонентів системи.

Робота пройшла апробацію на наукових конференціях, за результатами апробації опубліковано тези доповідей:

Робота пройшла апробацію на наукових конференціях, за результатами апробації опубліковано тези доповідей:

1. Кавацюк Б.О., Золотухіна О.А. Огляд систем магії для фентезі ігор у віртуальній реальності: Матеріали Всеукраїнської науково-технічної конференції. “Застосування програмного забезпечення в ІКТ” Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С. 367
2. Кавацюк Б.О., Золотухіна О.А. Розробка бібліотеки, що реалізує комплексну магію у фентезі іграх у віртуальній реальності з використанням рушія Unreal Engine 5: Матеріали Міжнародної науково-практичної інтернет-конференції “Молодь в науці: дослідження, проблеми, перспективи”. Збірник тез. 20.05.2024, ВНТУ, м. Вінниця. К.: ВНТУ, 2024. [URL]: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/21793/18036

# ПЕРЕЛІК ПОСИЛАНЬ


1. Кавацюк Б.О., Золотухіна О.А. Огляд систем магії для фентезі ігор у віртуальній реальності: Матеріали Всеукраїнської науково-технічної конференції “Застосування програмного забезпечення в ІКТ”. Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С.367.
2. Кавацюк Б.О., Золотухіна О.А. Розробка бібліотеки, що реалізує комплексну магію у фентезі іграх у віртуальній реальності з використанням рушія Unreal Engine 5: Матеріали Міжнародної науково-практичної інтернет-конференції “Молодь в науці: дослідження, проблеми, перспективи”. Збірник тез. 20.05.2024, ВНТУ, м. Вінниця. К.: ВНТУ, 2024. [URL]: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/21793/18036
3. Офіційна документація рушія Unreal Engine 5 [Електронний ресурс] : Режим доступу до ресурсу: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-3-documentation
4. Офіційна документація Figma [Електронний ресурс] : Режим доступу до ресурсу: https://help.figma.com/hc/en-us
5. Офіційна документація C++ [Електронний ресурс] : Режим доступу до ресурсу: https://learn.microsoft.com/en-us/cpp/?view=msvc-170
6. Spellcasting in Virtual Reality: DPVR and War of Wizards Create an Immersive Gaming Landscape [URL] : https://www.dpvr.com/en/spellcasting-in-virtual-reality-dpvr-and-war-of-wizards-create-an-immersive-gaming-landscape/
7. Відгуки до гри War of Wizards [URL] : https://store.steampowered.com/app/2009460/War_of_Wizards/#app_reviews_hash
8. Відгуки до гри RUMBLE [URL] : https://store.steampowered.com/app/890550/RUMBLE/#app_reviews_hash
9. Відгуки до гри OrbusVR: Reborn [URL] : https://store.steampowered.com/app/746930/OrbusVR_Reborn/#app_reviews_hash
10. Відгуки до гри Magitek VR [URL] : https://store.steampowered.com/app/1745280/Magitek_VR/#app_reviews_hash

# ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ


ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ


## Розробка бібліотеки, що реалізує комплексну магію у фентезі іграх у віртуальній реальності з використанням рушія Unreal Engine 5

Виконав студент 4 курсу
групи ПД-42
Кавацюк Богдан Олегович
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Золотухіна Оксана Анатолівна

Київ – 2024

## МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** Спрощення процесу розробки ігор, пов'язаних з використанням магії у віртуальній реальності
- **Об'єкт дослідження** Процес створення механіки гри у віртуальній реальності
- **Предмет дослідження** Засоби розробки механіки гри, пов'язаної з магією у віртуальній реальності

2

# ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Дослідження потреб ринку ігор ВР
2. Аналіз магічних систем у існуючих іграх ВР та їх вплив на ігровий процес
3. Проектування архітектури бібліотеки
4. Розробка інструмента для розпізнавання символів
5. Розробка інструмента для додавання символів активації
6. Розробка інструмента для створення заклинань
7. Розробка інструмента для призначення виклику заклинань на комбінацію символів або жест
8. Інтеграція розробленої бібліотеки у демонстраційну гру, розроблену для оцінки її ефективності та функціональності.
9. Проведення мануального тестування системи

3

# АНАЛІЗ АНАЛОГІВ

| Критерій порівняння | Orbus VR | War of Wizards VR | Rumble VR | Magitek VR | Розроблена бібліотека |
|---|---|---|---|---|---|
| Система активайії символами | + | + | - | - | + |
| Система активайії жестами | - | - | + | + | + |
| Кількість заклинань, доступних одномоментно | 22 | 4 (загально - 71) | 16 | 4 (загально - 30) | 30 |
| Наявність небойових заклинань | - | - | - | - | + |
| Можливість призначити заклинання на жест | відсутня система жестів | відсутня система жестів | - | - | + |

4

# Демонстрація існуючих ігор з магією та систем виклику заклинань (активація символами)

Orbus VR

5

# Демонстрація існуючих ігор з магією та систем виклику заклинань (активація символами)


War of wizards VR

6

# Демонстрація існуючих ігор з магією та систем виклику заклинань (активація жестами)

Rumble VR

7

# Демонстрація існуючих ігор з магією та систем виклику заклинань (активація жестами)



Magitek VR

8

# ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Користувач може перевести зображення символу у символ з допомогою UI
2. Користувач може створювати комбінації символів з допомогою UI методом Drag-and-drop вибираючи символи зі списку існуючих
3. Користувач може призначати заклинання на комбінації символів з допомогою UI, де вибирає бажані ефекти від заклинання
4. Інструмент, з допомогою якого можна створювати комбінації символів повинен мати можливість редагувати уже створені комбінації

Нефункціональні вимоги:

1. Введення нового символу не повинно займати більше 3 хвилин
2. Створення комбінації символів та призначення на неї заклинання не повинно перевищувати 5 хвилин

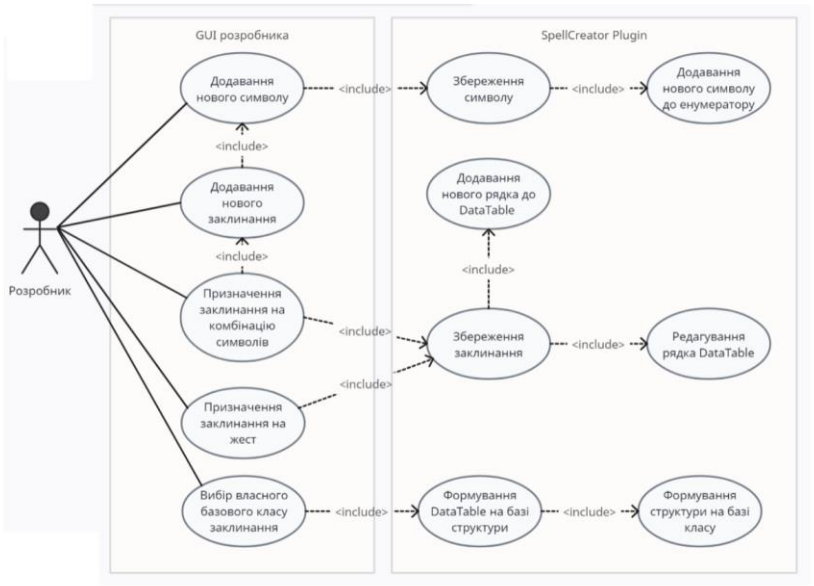
9

# ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



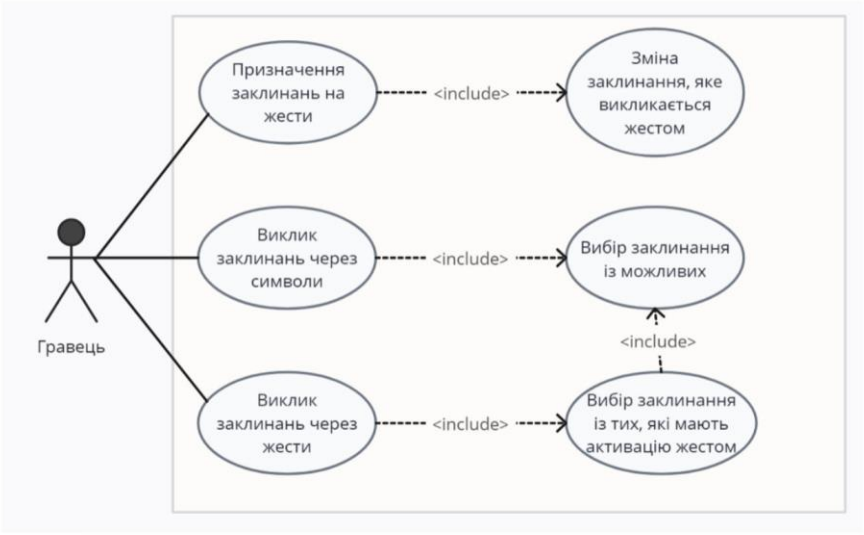
10

# Діаграма варіантів використання бібліотеки



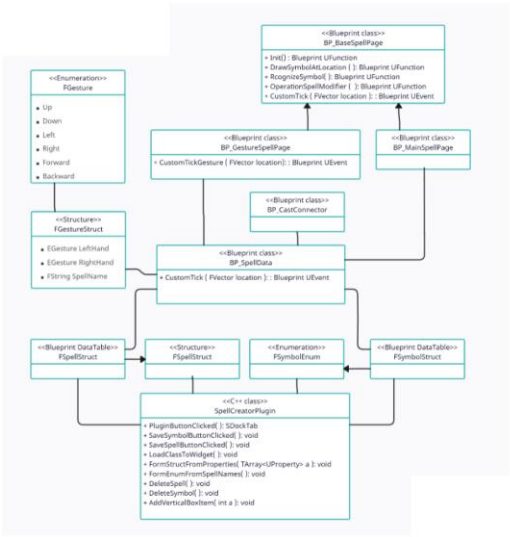

11

# Діаграма варіантів використання гри, розробленої з допомогою бібліотеки



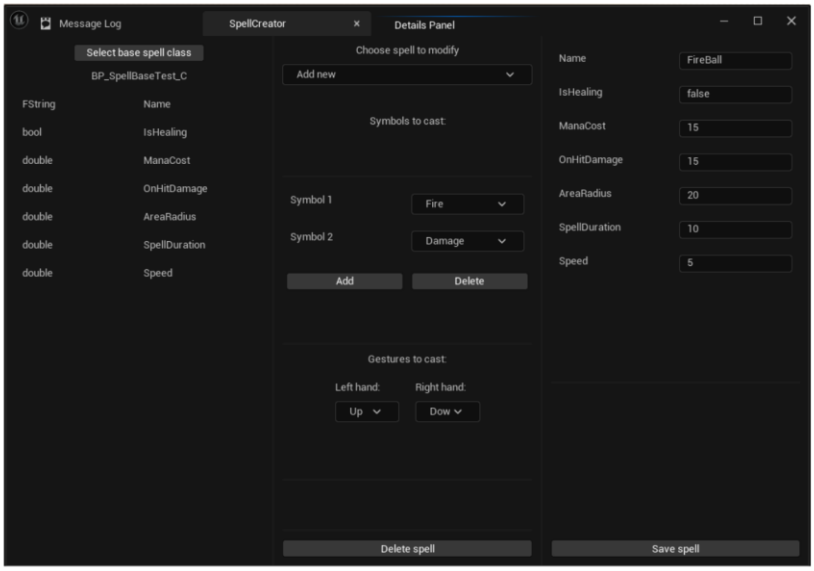

12

# Діаграма класів



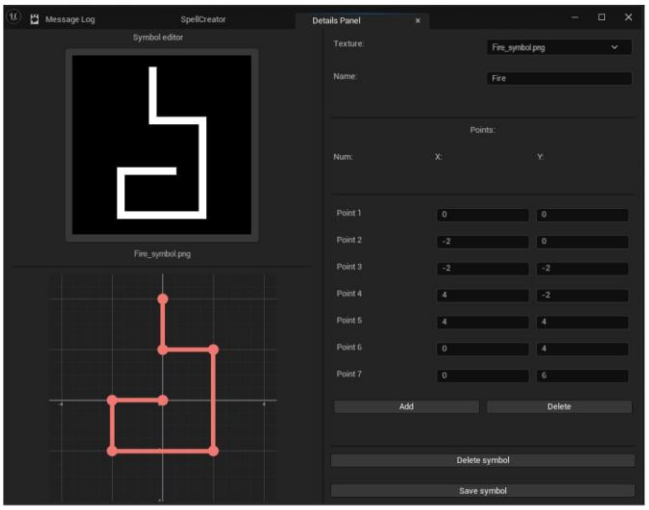

13

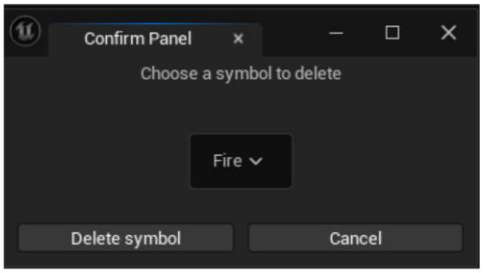
# Екранні форми




14

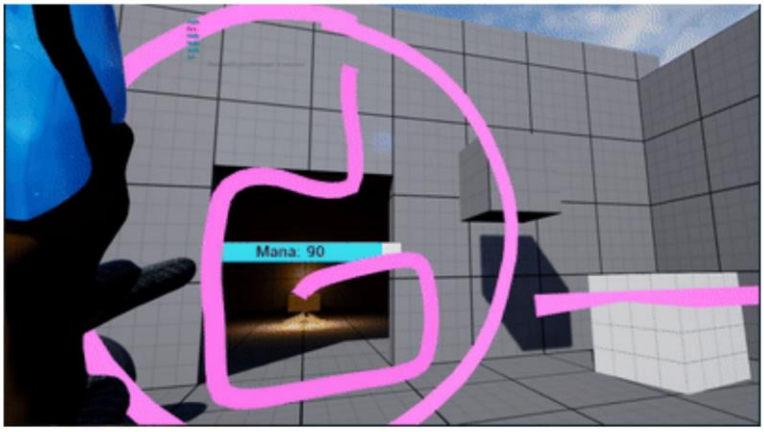
# Екранні форми




15

## Демонстрація розпізнавання символу




16

# АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Кавацюк Б.О., Золотухіна О.А. Огляд систем магії для фентезі ігор у віртуальній реальності: Матеріали Всеукраїнської науково-технічної конференції. “Застосування програмного забезпечення в ІКТ” Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С. 367

2. Кавацюк Б.О., Золотухіна О.А. Розробка бібліотеки, що реалізує комплексну магію у фентезі іграх у віртуальній реальності з використанням рушія Unreal Engine 5: Матеріали Міжнародної науково-практичної інтернет-конференції “Молодь в науці: дослідження, проблеми, перспективи”. Збірник тез. 20.05.2024, ВНТУ, м. Вінниця. К.: ВНТУ, 2024. [URL]: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/21793/18036



# ВИСНОВКИ

1. Досліджено потреби ринку ігор ВР
2. Проведено аналіз магічних систем у існуючих іграх ВР та їх вплив на ігровий процес
3. Спроектовано архітектуру бібліотеки
4. Розроблено інструмент для розпізнавання символів
5. Розроблено інструмент для додавання символів активації
6. Розроблено інструмент для створення заклинань
7. Розроблено інструмент для призначення виклику заклинань на комбінацію символів або жест
8. Проведено мануальне тестування системи