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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка детективної гри "Freedom of the Ghost"»

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

_____ Денис МАНКІВСЬКИЙ
(підпис)

Виконав: Здобувач вищої освіти групи ПД-43

Денис МАНКІВСЬКИЙ

Керівник: Олександр ЯРОШЕВСЬКИЙ
асистент кафедри

Рецензент: _____

Київ 2025

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
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____Манківському Денису Сергійовичу_____

1. Тема кваліфікаційної роботи: «Розробка детективної гри "Freedom of the Ghost" у форматі карт з вибором подій»
керівник кваліфікаційної роботи асистент кафедри ІПЗ Олександр ЯРОШЕВСЬКИЙ, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: Опис процесу створення детективної покрокової карткової гри на рушії Unity. Проведено аналіз жанрових аналогів (Cluedo, Detective: A Modern Crime Board Game, Obduction) та офіційної документації до Unity 2020.3 LTS, а також вивчено можливості побудови внутрішньоігрової логіки без залучення серверної частини. Сформульовано функціональні та нефункціональні вимоги до гри, виконано проєктування архітектури на основі об'єктно-орієнтованого підходу, реалізовано діаграми Use Case та класів, визначено основні сценарії взаємодії користувача з ігровим середовищем. Розробка та тестування здійснювались із використанням мови програмування C# та середовища Unity, з фокусом на реалізацію системи вибору дій, показу підказок, внутрішнього блокнота та фіксації історії гравців.

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

1. Огляд та аналіз наявних детективних та карткових ігор із елементами розслідування (Cluedo, Her Story, Obduction), визначення їхніх переваг та недоліків, актуальність обраної теми.
2. Визначення функціональних та нефункціональних вимог до мультиплеєрної кооперативної гри.
3. Розробка підказки від привида, вибір дій, класи карток, сценарії взаємодії з гравцем. Побудова Use Case та діаграм класів. Вибір рушія Unity, мови C#, використання Canvas UI та ScriptableObject.
4. Опис реалізації ключових компонентів: відображення карток, обробка вибору гравця, система підказок, фази гри. Застосування патернів MVP, подієвої моделі, кастомних скриптів у Unity.
5. Проведення модульного та інтеграційного тестування логіки гри. Перевірка циклу дій, роботи підказок, вибору дій, реакції на дії гравця. Тестування UI та плавності сценаріїв.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до додатку (функціональні та нефункціональні).
3. Концепт гри.
4. Геймплей та механіки.
5. Програмні засоби реалізації.
6. UML-діаграми (Use Case, класів).
7. Екранні форми.
8. Апробація результатів дослідження.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури з тематики детективних ігор та ігрової механіки	25.02-04.03.2025	
2	Аналіз існуючих аналогів (Cluedo, Obduction, Detective, Her Story)	05.03-13.03.2025	
3	Проєктування структури гри: сценарії дій, підка зки, карти, логіка вибору	14.03-23.03.2025	
4	Розробка Use Case та UML-діаграм, проєктування архітектури з використанням патерну MVP	24.03-03.04.2025	
5	Програмна реалізація ключових механік: картки, SpiritBoard, вибір дій, підказки	04.04-19.04.2025	
6	Інтеграція інтерфейсу та логіки, створення вікон вибору, історії дій	20.04-28.04.2025	

7	Тестування проєкту: перевірка фаз гри, стабільн ості, сценаріїв	29.04-08.05.2025	
8	Оформлення роботи: вступ, висновки, опис реалізації	09.05-15.05.2025	
9	Розробка демонстраційних матеріалів для захисту	16.05-22.05.2025	
10	Попередній захист та подання роботи	23.05-30.05.2025	

Здобувач вищої освіти _____ Денис МАНКІВСЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Олександр ЯРОШЕВСЬКИЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 44 стор., 1 табл., 16 рис., 21 джерел.

Мета роботи – посилити містично-детективну складову гри шляхом використання специфічних об’єктів, таких як карти дій, підказки від привида та панель досьє, що формують атмосферу розслідування та занурюють гравця у процес пошуку істини.

Об’єкт дослідження – процес створення інтерактивної детективної гри з використанням карткової системи вибору та побудови логіки підказок.

Предмет дослідження – технології та інструменти реалізації інтерактивної детективної гри з картковим інтерфейсом.

Короткий зміст роботи: У кваліфікаційній роботі проаналізовано особливості реалізації покрокових детективних ігор з картковим інтерфейсом. Досліджено ігри подібного жанру, зокрема Simulacra, Duskwood та Card Detective, на основі чого виокремлено ключові функціональні елементи: поступове розкриття сюжету, вибір дій через картки, система підказок, взаємодія з духом-привидом, а також запис та збереження виборів гравця.

Розроблено концепцію гри Freedom of the Ghost, у якій поєднано містичну атмосферу з логікою дедуктивного розслідування. Реалізовано ігрові механіки: генерація випадкової справи (вбивця, місце, зброя, час), взаємодія через чотири типи карт (підказка, пам’ять, правда/брехня, доля), збереження виборів у блокноті, підказки через SpiritBoard, досьє із передісторією та підсумкова панель результатів.

Для розробки було використано ігровий рушій Unity, мову програмування C#, інструменти Canvas UI для побудови інтерфейсів та ScriptableObject для зберігання ігрових даних. Основна логіка реалізована через розділення на менеджери (GameManager, ProgressManager, ChoiceManager тощо), що дозволяє підтримувати масштабованість та повторне використання компонентів.

Проведено тестування базового функціоналу, перевірено варіативність сценаріїв, працездатність підказок, стабільність циклу гри та відповідність логіки розслідування. Отримані результати підтверджують ефективність реалізованої структури та механік у контексті жанру інтерактивного детективу.

Сферою використання розробленої гри є організація дозвілля, розвиток логічного мислення, навичок аналізу, дедукції та командної взаємодії серед підлітків та молоді. Гру можна використовувати як інструмент для розвитку критичного мислення в неформальній освіті, психологічних тренінгах або в рамках навчальних ігрових сесій, орієнтованих на розвиток соціальної взаємодії.

КЛЮЧОВІ СЛОВА: UNITY, C#, CANVAS UI, ДЕТЕКТИВНА ГРА, SPIRITBOARD, ПРИВИД, ПІДКАЗКА, КАРТКОВА ГРА, ПЕРЕТЯГУВАННЯ КАРТ, GAMEMANAGER, БЛОКНОТ ГРАВЦЯ, КАРТКИ, ПОКРОКОВИЙ ГЕЙМПЛЕЙ.

ЗМІСТ

ВСТУП

Актуальність теми полягає в зростаючій ролі інтерактивних ігор як засобу занурення у складні сюжетні сценарії та розвитку аналітичного мислення. У сучасному ігровому просторі дедуктивні ігри з елементами містики набувають популярності завдяки здатності поєднувати інтелектуальні завдання з емоційним зануренням. Особливість гри Freedom of the Ghost полягає у використанні оригінальної карткової системи взаємодії та участі персонажа-привида як посередника у розслідуванні. Гравець не лише здійснює вибір дій через картки, а й отримує підказки за допомогою містичних елементів, таких як спиритична дошка та випадкові бачення. Це дозволяє створити глибший рівень залучення, варіативність проходження та атмосферу інтерактивного розслідування з елементами надприродного.

Об'єктом дослідження є процес створення інтерактивної детективної гри з використанням карткової системи вибору та побудови логіки підказок.

Предметом дослідження є технології та інструменти реалізації інтерактивної детективної гри з картковим інтерфейсом.

Метою роботи є посилення містично-детективної складової гри шляхом використання специфічних об’єктів, що формують атмосферу розслідування та занурюють гравця у процес пошуку істини.

Методи дослідження: аналіз жанрових особливостей детективних ігор з картковим інтерфейсом, вивчення аналогічних проєктів, проєктування архітектури гри за допомогою UML-діаграм, реалізація ігрової логіки в середовищі Unity з використанням мови C#, створення інтерфейсу користувача з опорою на Canvas UI, моделювання механіки підказок та варіативного вибору дій, проведення модульного тестування функціональних компонентів гри та ручного тестування ігрових сценаріїв.

Наукова новизна проекту визначається поєднанням карткової механіки з елементами детективного розслідування, використанням інтерактивного

персонажа-привида як джерела підказок, що формує атмосферу містики та спрямовує гравця. Реалізовано механіку SpiritBoard — виведення підказок у вигляді літер, а також систему запису виборів із візуальними маркерами правильності. Гра забезпечує повне проходження розслідування в одному сеансі з можливістю повтору та зміни справи.

Практична значущість полягає в тому, що розроблена детективна гра може бути використана як засіб розвитку аналітичного мислення, уваги до деталей та дедуктивних навичок. Завдяки використанню інтерактивного інтерфейсу, гра здатна виконувати роль тренажера для формування навичок послідовного аналізу ситуацій

1 АНАЛІЗ ПРОБЛЕМИ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Опис предметної галузі

У сучасному світі комп’ютерні ігри стали не лише джерелом розваг, а й потужним інструментом розвитку когнітивних здібностей користувачів. Особливе місце серед них посідають інтерактивні ігри, що поєднують елементи логіки, стратегічного мислення та соціальної взаємодії. Саме до такого жанру належить ігра "Freedom of the Ghost" —детективна карткова гра з покроковим ігровим процесом, що реалізується на рушії Unity. Вона об'єднує компоненти дослідження, збору підказок та поступового розкриття сюжетної лінії, що надає їй глибини та інтерактивності.

Предметна галузь дослідження охоплює розробку інтерактивних детективних ігор з картковим інтерфейсом, у яких гравець послідовно обирає дії, аналізує підказки та намагається розкрити злочин. Особливістю є впровадження елементів містики через персонажа-привида, який надає непрямі підказки, а також створення ігрового процесу, побудованого на поступовому відкритті інформації, збереженні рішень у блокноті та змінній структурі проходження.

1.2 Огляд програмних продуктів-конкурентів

1.2.1 Simulacra

Simulacra — це інтерактивна детективна гра, реалізована у вигляді знайденого телефону. Гравець досліджує особисті повідомлення, фотографії та додатки зниклої людини, щоб розкрити правду про її зникнення. Ігровий процес побудований на пошуку підказок у цифрових слідах та аналізі взаємозв’язків між знайденою інформацією. Гра поєднує атмосферу напруги, елементи психологічного трилера та нелінійний наратив.

Недоліки: обмежена взаємодія з ігровим середовищем, відсутність карткової механіки чи варіативного вибору дій у традиційному геймплейному форматі.

1.2.2 Duskwood

Duskwood — це мобільна детективна гра в жанрі інтерактивного трилера, де гравець отримує повідомлення від персонажів, які розслідують зникнення дівчини. Вся гра побудована навколо текстового спілкування, перегляду файлів, відео та аудіо, які поступово розкривають сюжет. Важливу роль відіграє атмосфера напруженості та ілюзія реального листування в чаті.

Недоліки: обмежене управління діями гравця, відсутність фізичної взаємодії з об’єктами, вузька механіка вибору реплік без альтернатив у вигляді карток чи активних дій.

1.2.3 Card Detective

Card Detective — це мобільна гра в жанрі детективу, побудована навколо карткової механіки. Гравець отримує нові картки, що символізують докази, персонажів або події, і використовує їх для з’ясування деталей злочину. Кожен вибір впливає на хід розслідування, а кінцівка залежить від зібраної інформації.

Недоліки: лінійний сюжет у більшості сценаріїв, обмежена взаємодія з ігровим світом поза межами карткових подій.

1.2.4 Порівняльна таблиця функціональних можливостей і недоліків

Таблиця 1.1

Аналіз конкурентів проєкта

Назва гри	**Формат**	**Соціальна взаємодія**	**Випадковість**	**Покроковість**	**Сюжетність**	**Недоліки**
Simulacra	Цифрова	Низька	Низька	Ні	Висока	Лінійний геймплей, обмежена взаємодія
Duskwood	Цифрова	Середня	Низька	Ні	Висока	Повільний темп, обмежена варіативність
Card Detective	Цифрова	Низька	Середня	Так	Середня	Обмежений вплив гравця на сюжет
Freedom of the Ghost	Цифрова	Висока	Середня	Так	Висока	Потреба у складній побудові сценаріїв

1.3 Огляд IT-засобів і технологій для реалізації

1.3.1 Ігровий рушій Unity

Unity — це сучасне кросплатформне середовище розробки, яке дозволяє створювати інтерактивні додатки, зокрема ігри, з високим рівнем візуалізації, анімації та логіки. Однією з ключових переваг рушія є його орієнтація на зручність розробника: інтерфейс Unity має інтуїтивно зрозумілу структуру, яка дозволяє зосередитися саме на функціональності проєкту, а не на деталях реалізації графічного рушія.

Для реалізації гри «Freedom of the Ghost» було обрано саме Unity з огляду на його потужні можливості в роботі з 2D-графікою. Завдяки компонентам Sprite

Renderer, Canvas, RectTransform, Image та Event System, розробнику вдалося реалізувати взаємодію з картками, підказками, інтерфейсом вибору дій та блокнотом гравця. Система Canvas Scaler дозволила реалізувати адаптивний інтерфейс — незалежно від розміру вікна чи екрану, всі елементи відображаються з однаковою точністю й логікою взаємодії.

Ще однією ключовою перевагою є SceneManager, який дає змогу гнучко керувати переходами між сценами: наприклад, запуск нової гри, повернення до головного меню, завершення сесії. Це значно спрощує керування ігровим процесом та підвищує гнучкість системи. Для реалізації ігрової логіки також широко використовувався Canvas Group — з його допомогою реалізовано ефекти затемнення, плавне з’явлення панелей, блокування взаємодії з елементами під час анімацій тощо.

Особливо важливим було впровадження механізму drag and drop (перетягування карток), що реалізовано через події OnBeginDrag, OnDrag, OnEndDrag, завдяки чому гравець інтуїтивно взаємодіє з картками, переміщаючи їх до спеціальних зон (наприклад, до зони примари або панелі вибору). Така реалізація підвищує рівень залученості користувача до ігрового процесу.

Крім того, Unity підтримує імпорт великої кількості форматів графіки та звуку, що дозволило швидко інтегрувати зображення карток, іконки та аудіоефекти. Завдяки Asset Store, розробник мав змогу використовувати додаткові ресурси (шаблони кнопок, фони, UI-елементи), що значно прискорило створення прототипу гри.

Не менш важливою перевагою Unity є можливість кросплатформеного розгортання. У майбутньому гра може бути експортована на інші платформи (Android, iOS, WebGL, macOS), що розширює її потенційну аудиторію та можливості подальшого розвитку. Наявність активної спільноти Unity також спростила вирішення нетривіальних задач через документацію та обговорення на форумах.

Таким чином, Unity став оптимальним вибором для реалізації гри «Freedom of the Ghost», завдяки своїй потужності, гнучкості, інструментальним можливостям та легкості у вивченні для розробника.

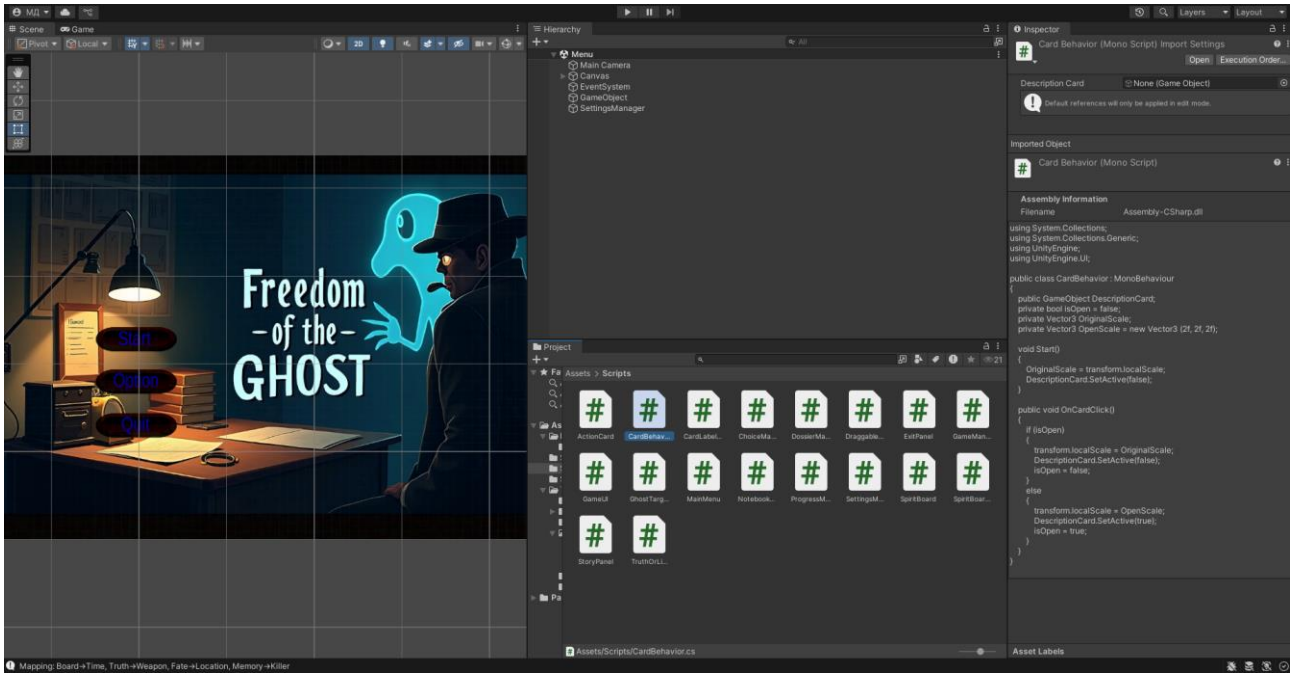


Рис. 1.1 Інтерфейс рушія Unity

1.3.2 Мова програмування C#

C# — це об'єктно-орієнтована мова програмування, створена компанією Microsoft, яка є основною мовою для скриптової розробки в Unity. У контексті створення гри «Freedom of the Ghost», ця мова надала всі необхідні інструменти для створення стабільної, структурованої та гнучкої ігрової логіки.

Завдяки суворій типізації, C# дозволяє уникнути багатьох типових помилок ще на етапі компіляції, що сприяє підвищенню надійності коду. Під час розробки гри було створено десятки класів, зокрема CardData, CardDisplay, GhostMemory, Player, GameManager, ActionPanelManager, NotebookController, кожен з яких відповідає за окрему частину логіки гри. Об'єктно-орієнтований підхід дозволив організувати код у вигляді логічно розділених блоків, які легко тестувати та підтримувати.

Однією з особливостей мови C# є підтримка подій та делегатів, що дозволило реалізувати зручну обробку подій взаємодії користувача з картками: вибір,

перетягування, активація дій, перевірка правдивості відповідей. Через систему EventTrigger Unity, поєднану з C#-скриптами, реалізовано реалістичну взаємодію з інтерфейсом — наприклад, відкриття підказок від примари чи перевірку істинності твердження гравця.

C# також дозволяє ефективно працювати з корутинами (Coroutine) — механізмом реалізації відкладених або поступових дій у часі. У грі це використовувалося, наприклад, при затримках перед відображенням відповідей привида, підсвічуванні карток, а також під час покрокового відкриття панелей вибору. Це забезпечило більш плавний та візуально привабливий ігровий досвід.

Окрім того, C# добре інтегрується з UI-компонентами Unity через UnityEngine.UI та TextMeshPro, що дозволило відображати текстові підказки, реагувати на введення користувача, оновлювати блокнот гравця відповідно до прийнятих рішень. Уся логіка оновлення інтерфейсу та збереження вибраних карток реалізована за допомогою скриптів C#, які взаємодіють з відповідними GameObject-ами на сцені.

Завдяки використанню C# розробка гри стала не лише технічно можливою, але й структурованою, керованою та масштабованою. Усі функціональні можливості гри були реалізовані виключно через цю мову, без залучення сторонніх технологій, що підвищує стабільність і надійність всього проєкту.

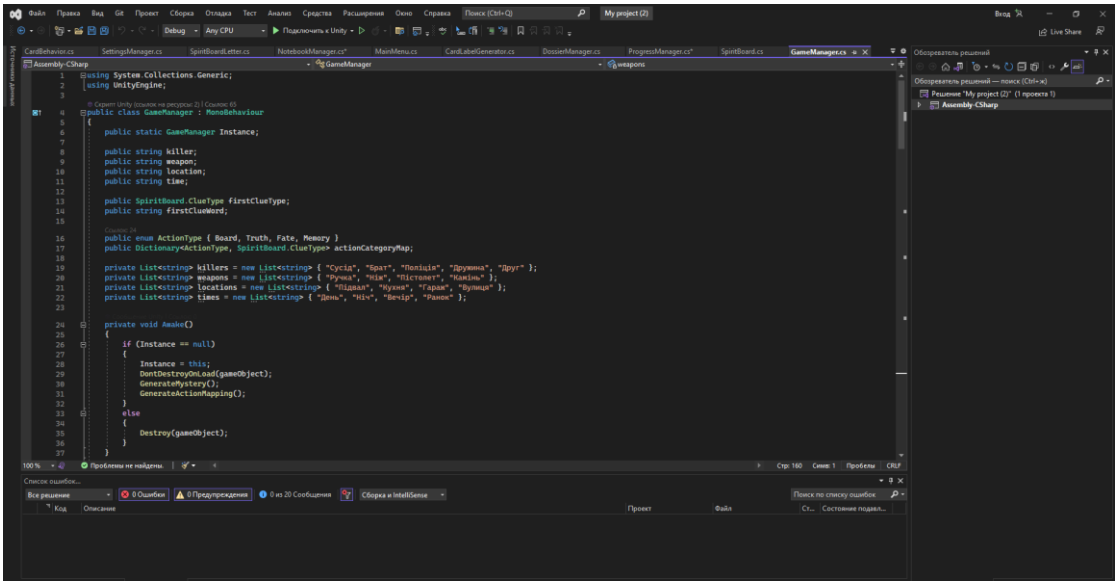


Рис. 1.2 C# розробка

1.3.3 Система UI на базі Canvas

Графічний інтерфейс гри реалізований на основі системи Canvas, що є стандартною технологією в Unity для створення UI. Вона дає змогу будувати інтерактивні панелі, кнопки, текстові поля, списки, які автоматично адаптуються під різні дозволи екранів.

У "Freedom of the Ghost" через Canvas було реалізовано:

- головне меню ігри;
- панель налаштувань;
- екран вибору дії;
- панель із досьє;
- підказки від привида;
- вікно вибору карток;
- підсвітка прогресу гравця;
- блокнот гравця з історією вибору;
- вікна завершення гри та результатів.

Компоненти Canvas поєднуються з системами Event Trigger і Graphic Raycaster, що забезпечують коректне реагування на події (кнопковий вхід, перетягування карт, натискання). За допомогою Layout Group і Content Size Fitter забезпечено адаптивність і зручне розташування елементів на екрані.

Завдяки вбудованим механізмам Unity, система Canvas забезпечила швидке створення та гнучке налаштування інтерфейсу, що відповідає стилістиці гри та її функціональності.

1.3.4 Порівняння альтернативних рішень

При виборі технологічного стеку для реалізації гри "Freedom of the Ghost" було розглянуто кілька популярних ігрових рушіїв. Основним критерієм слугувала їх придатність до розробки 2D-проєктів із покроковою логікою, широкими можливостями по роботі з інтерфейсом, підтримкою кастомної логіки дій, а також зручністю для навчальної розробки.

Unreal Engine

Unreal Engine — це один із найпотужніших і найпопулярніших рушіїв у світі геймдеву, який найчастіше використовується у високобюджетних AAA-проєктах. Його основною перевагою є унікальна система візуалізації, що забезпечує фотореалістичну графіку, завдяки чому Unreal став стандартом у створенні 3D-ігор з кінематографічною якістю. Рушій базується на мові C++, але також пропонує Blueprint — систему візуального скриптингу, яка дозволяє реалізовувати логіку без програмування, що є плюсом для дизайнерів рівнів та новачків.

Unreal має багатий набір інструментів, таких як Niagara (система частинок), MetaSounds (звук), Control Rig (анімація), які дозволяють реалізувати складні сцени, фізику, анімацію персонажів і середовища. У рушії реалізовано сучасне освітлення — Lumen, яке підтримує глобальне освітлення в реальному часі, що критично для реалістичних 3D-сцен.

Однак, незважаючи на свої сильні сторони, Unreal Engine має й ряд недоліків, які роблять його менш привабливим для створення саме 2D-проєктів, таких як «Freedom of the Ghost». Рушій орієнтований передусім на великі тривимірні проєкти, що робить реалізацію простої покрокової 2D-гри надмірно складною. Наявні інструменти для 2D хоча й існують (Paper2D), але є другорядними, менш підтримуваними і не мають розвиненої документації чи шаблонів.

Також слід зазначити, що Unreal Engine потребує потужного обладнання — як для розробки, так і для тестування гри. Високі вимоги до ОЗП, відеокарти й процесора ускладнюють швидке прототипування. Крім того, збірка ігор займає більше часу, а розмір фінальних проєктів є значно більшим.

Висновок: хоча Unreal Engine — ідеальний вибір для розробки складних 3D-ігор із високою візуальною точністю, для гри «Freedom of the Ghost», яка зосереджується на логіці, тексті, підказках і взаємодії через картки, цей рушій є надмірним та не оптимальним.

Godot Engine

Godot Engine — це відкритий рушій з відкритим кодом, що активно розвивається і має зростаючу популярність серед інді-розробників. Його головною особливістю є ієрархічна структура сцен та вузлів, що дає змогу легко компонувати логіку, UI та об’єкти гри. Такий підхід дозволяє реалізовувати проєкти без глибокого занурення в складні шаблони проектування, що робить рушій зручним для початківців.

Godot відмінно підходить для створення 2D-проєктів, маючи окрему двовимірну систему координат, камери, колайдерів і фізики. Його власна мова програмування — GDScript — базується на Python-подібному синтаксисі, що робить її доступною для нових користувачів. Окрім того, рушій підтримує інші мови: C#, C++ і навіть VisualScript (візуальне програмування).

Інтерфейс рушія компактний і зручний, особливо при роботі з UI: створення кнопок, панелей, анімацій та подій реалізується швидко. Godot також має вбудовану анімаційну систему, редактор шейдерів, систему сигналів для обробки подій без прямої залежності між об’єктами, що дозволяє легко створити подієвий геймплей.

Попри свої переваги, Godot має певні обмеження: порівняно невелика спільнота, менша кількість шаблонів і плагінів у Asset Library, складніша реалізація багатокористувацьких механік, а також обмежена підтримка зовнішніх хмарних сервісів. Крім того, деякі складні механіки потребують ручної реалізації, оскільки готові рішення часто не охоплюють всі випадки.

Висновок: Godot — чудовий рушій для створення простих 2D-ігор, але для реалізації повноцінної покрокової гри з великою кількістю UI-елементів, інтерактивного сценарію, обробки карткових дій та гнучкої логіки — можливостей Godot може бути недостатньо.

Це обмежує його придатність для розробки гри «Freedom of the Ghost».

Unity

Unity — це один із найпопулярніших рушіїв для створення як 2D, так і 3D ігор, який ідеально підходить як для інді-розробників, так і для великих студій. Його головна перевага — це баланс між потужністю, простотою та функціональністю, що робить його універсальним інструментом для реалізації найрізноманітніших ігрових ідей.

Unity має розвинену систему UI, яка включає Canvas, Layout Group, Button, TextMeshPro, EventSystem, Animation, Animator, Raycast Target, Canvas Group, що забезпечує реалізацію повного графічного інтерфейсу для карток, підказок, блокнота, меню тощо. Також реалізована підтримка drag and drop через IPointerDownHandler, IPointerDragHandler та інші інтерфейси, що дозволяє гравцеві інтуїтивно взаємодіяти з картками.

Код у Unity пишеться на мові C#, яка підтримує об'єктно-орієнтоване програмування. Це дозволило в грі «Freedom of the Ghost» реалізувати гнучку структуру: класи для обробки карток, підказок, інтерфейсу, вибору гравця, логіки перемоги та перезапуску гри. Через корутини (IEnumerator) було реалізовано затримки між фазами гри, а також плавне з’явлення підказок.

Завдяки великій спільноті та Unity Asset Store, розробник мав доступ до величезної кількості готових рішень, включаючи шаблони UI, звукові ефекти, анімації, шрифти, системи підказок. Це дозволило зосередитися саме на розробці логіки гри, не витрачаючи час на створення базових елементів з нуля.

Unity також підтримує багатоплатформенну збірку, що дає змогу в майбутньому легко експортувати гру на Android, iOS, ПК, WebGL, що особливо важливо з точки зору комерціалізації гри або її розповсюдження.

Висновок: саме Unity став найкращим вибором для реалізації гри «Freedom of the Ghost» завдяки ідеальній підтримці 2D-графіки, розвиненій інфраструктурі, простоті інтеграції UI, логіки подій та взаємодії з користувачем. Гнучкість, стабільність і велика спільнота зробили цей рушій оптимальним серед інших доступних альтернатив.

Загальний висновок

У результаті аналізу:

- Unreal Engine відзначається високим рівнем деталізації та продуктивності, але є надлишковим для нашої гри.
- Godot має хороші можливості для 2D, але обмежений у розширюваності та мережевій підтримці.
- Unity забезпечує всі необхідні інструменти для створення 2D-гри з нелінійною логікою, інтерактивним UI, зручною структурою проєкту та широкими можливостями для масштабування.

Саме тому Unity було обрано як основну платформу для розробки бакалаврського проєкту — детективної покрокової гри "Freedom of the Ghost".

2. ПРОЄКТУВАННЯ ГРИ «FREEDOM OF THE GHOST»

2.1 Архітектура гри

Архітектура гри «Freedom of the Ghost» побудована за принципами модульності та розділення відповідальностей. Основна увага приділена організації логіки гри навколо фазового геймплею, де кожна дія гравця (вибір картки, перегляд досьє, взаємодія з привидом) ініціює відповідну зміну стану системи. Гра реалізована як послідовність взаємопов'язаних етапів, що керуються центральним менеджером гри (GameManager) та підтримуються окремими модулями інтерфейсу, логіки підказок, історії рішень і візуальних ефектів.

Усі ключові підсистеми взаємодіють через події або прямі виклики, забезпечуючи цілісність ігрового процесу та контроль за послідовністю дій. Такий підхід дозволяє масштабувати гру в майбутньому, додавати нові типи карт, змінювати правила або структуру розслідування без втрати стабільності системи.

Інтерфейс побудований за допомогою Canvas-системи Unity, що дозволило швидко створити інтерактивні елементи та адаптувати гру до різних розширень екранів. Уся архітектура підтримує гнучку інтеграцію нових елементів і дозволяє реалізовувати складну логіку з мінімальною залежністю між модулями.

2.1.1 Компоненти гри та зв'язки між ними

Гра "Freedom of the Ghost" реалізована на основі модульної структури, де кожен компонент відповідає за конкретну частину ігрового процесу. Основні модулі гри:

- Game Manager — головний керуючий компонент. Відповідає за генерацію справи (випадковий вибір вбивці, місця, зброї та часу), відслідковує стан гри, ініціалізує підказки та взаємодіє з іншими системами;
- UI Manager — відповідає за відображення інтерфейсу: головного меню, екрану вибору дій, досьє, блокнота, повідомлень від духа та завершення гри;
- Scene Controller — обробляє завантаження та активацію сцен;

- SpiritBoard — спеціальний модуль взаємодії з привидом. Відповідає за виведення букв, підказок;
- NotebookManager — система запису всіх вибраних гравцем відповідей по категоріях (вбивця, місце, зброя, час). Також визначає, правильна відповідь чи ні, і відображає результат (+/–);
- ChoicePanelManager — відповідає за динамічне формування трьох варіантів відповідей, що з'являються після взаємодії з певною картою. Після вибору — передає відповідь у блокнот;
- DossierManager — генерує сюжетне досьє на початку кожної гри, базуючись на обраних правильних відповідях. Досьє надає атмосферне занурення та натяки;
- CardSystem — реалізує логіку дії кожної карти (Memory, Truth, Fate, Board). Кожна карта має унікальну поведінку та взаємодіє з SpiritBoard або ChoicePanelManager;
- ProgressTracker — веде підрахунок правильних відповідей, змінює підсвітку (кольорову індикацію) на блокноті, контролює завершення гри та її результат;
- Player Controller — відповідає за обробку вибору гравця та дій на ігровому полі.

Усі компоненти спілкуються між собою через посередника GameManager або за допомогою подій (Events) (див. рисунок 2.1).

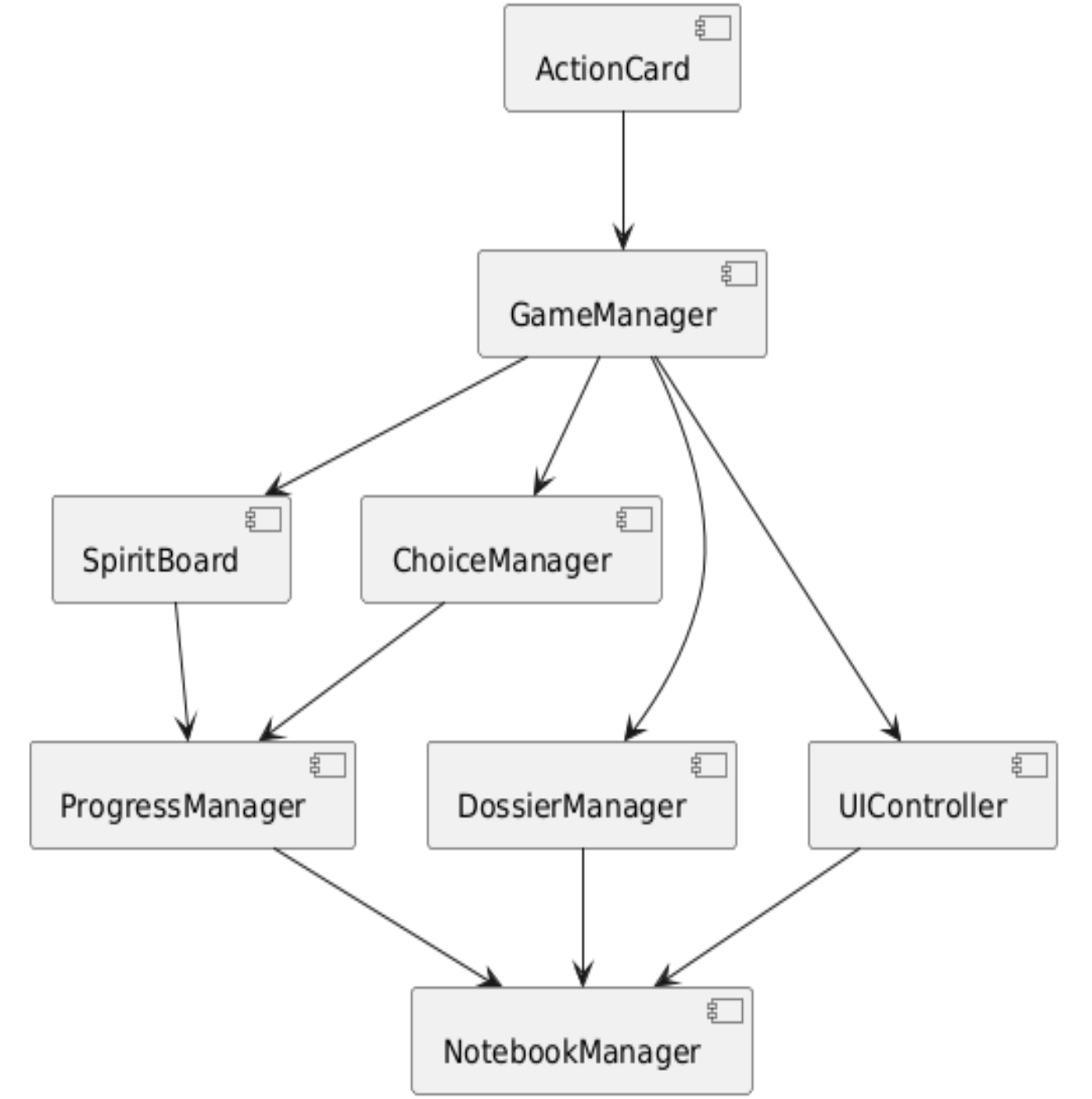


Рис. 2.1 Component Diagram

2.1.2 Сценарії взаємодії гравців

У грі "Freedom of the Ghost" гравець взаємодіє з ігровими компонентами у форматі покрокового розслідування. На початку кожної нової справи гравцю показується досьє з описом сцени злочину, що формує атмосферу та задає напрям пошуку. Далі гравець поетапно використовує чотири типи карт дій: SpiritBoard, Memory, Truth, Fate, кожна з яких пов’язана з однією з категорій: вбивця, місце, знаряддя або час.

Після кожного ходу:

- гравець отримує підказку від привида відповідного типу;
- система пропонує три варіанти вибору;
- гравець обирає відповідь і отримує результат;
- запис автоматично фіксується в блокноті.

Інтерфейс підсвічує хід розслідування, показуючи прогрес, і в кінці гри гравець дізнається, чи вдалося розкрити справу. Механіка дозволяє як повністю нову гру, так і повторне проходження поточної (див. рисунок 2.2.)

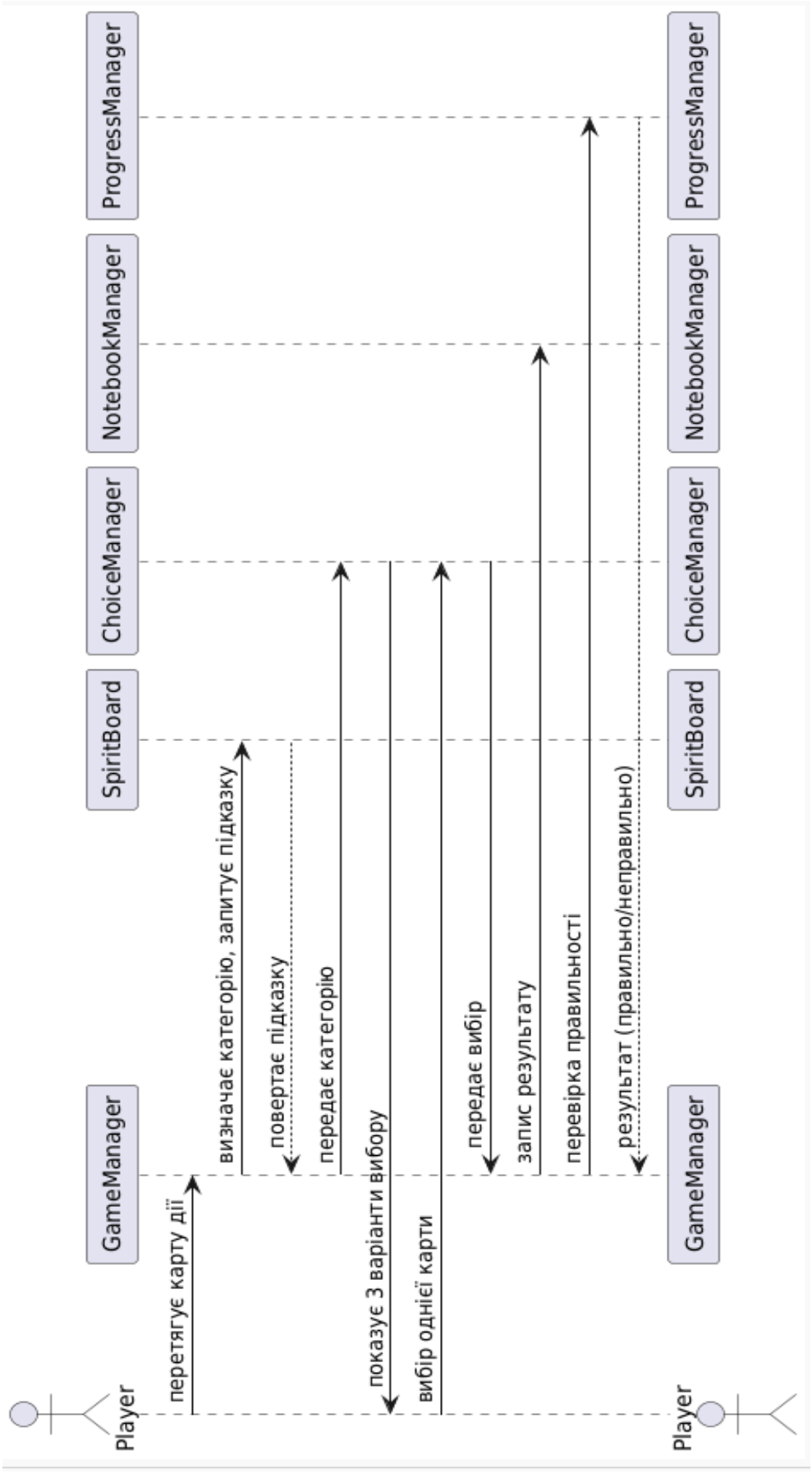


Рис. 2.2 Sequence Diagram

2.2 Патерни проектування

У процесі розробки гри "Freedom of the Ghost" було використано низку шаблонів проєктування для забезпечення гнучкої та масштабованої архітектури:

- Singleton (Одинак) — застосовано для класу GameManager, який координує загальну логіку гри, контролює стан гри, перехід між сценами та взаємодію між ключовими системами.
- Observer (Спостерігач) — реалізовано у вигляді системи подій між компонентами гри. Наприклад, NotebookManager автоматично оновлюється при новому виборі гравця без прямої залежності від джерела події.
- Factory Method (Фабричний метод) — використовується для створення об’єктів карток дій, що дозволяє гнучко створювати різні типи карт (вбивця, місце, зброя, час) на основі даних у ScriptableObject.
- Command (Команда) — застосовується для зберігання дій гравця у блокноті, що дозволяє реалізувати історію виборів, а також потенційно реалізувати функції "повернутися назад" у майбутньому.
- MVC-подібний розподіл — хоч і не формальний MVC, логіка гри розділена на:
    - модель (ClueData, PlayerChoice),
    - контролер (GameManager, ChoiceManager),
    - представлення (UIManager, NotebookUI, SpiritPanelUI).

Ці патерни забезпечили чітке розділення відповідальностей, легкість у тестуванні компонентів та зручність розширення гри новими сценаріями чи механіками.

2.2.1 Singleton

Патерн Singleton використовується для забезпечення існування лише одного екземпляра певного класу протягом усього життєвого циклу гри. У нашому проєкті цей підхід реалізовано у класі GameManager, який виступає глобальним координатором ігрового процесу.

Основні функції GameManager:

- ініціалізація ігрових параметрів;
- управління станами гри (початок, хід гравця, голосування, завершення тощо);
- передача команд до інших підсистем (UI, генератор карт, блокнот, дух тощо);
- централізована обробка подій, що впливають на весь ігровий контекст.

Використання Singleton дозволило уникнути множинних екземплярів GameManager у різних сценах, зберігаючи глобальний контекст та забезпечивши узгодженість у доступі до логіки керування. Через Instance GameManager до будь-якої частини гри можна швидко й безпечно звернутися.

2.2.2 Observer

Патерн Observer забезпечує реактивну архітектуру, в якій об'єкти (спостерігачі) автоматично повідомляються про зміну стану іншого об'єкта (суб'єкта). У грі цей механізм реалізовано через делегати та події на C#.

Приклад застосування:

- після вибору картки гравцем викликається подія OnCardSelected;
- підписані на цю подію системи (наприклад, NotebookManager, GhostHintManager, UIUpdater) миттєво реагують на зміну — оновлюють блокнот, показують підказку, активують анімацію тощо.

Цей підхід мінімізує прямі залежності між об'єктами: один компонент не має знати про конкретну реалізацію іншого. Завдяки цьому нові підсистеми можна легко підключити до наявних подій без зміни існуючого коду. Це відповідає принципам відкритості/закритості (SOLID) та підтримує гнучкість системи.

2.2.3 Factory Method

Для створення ігрових об'єктів — зокрема карток — у грі реалізовано патерн Factory Method. Ідея полягає в тому, щоб інкапсулювати процес створення об'єктів залежно від їх типу (вбивця, місце, знаряддя, час), абстрагуючи користувача від деталей конструювання.

Приклади реалізації:

- при старті гри фабрика викликається для генерації карт з різними параметрами (ім'я, тип, колір, анімація);
- для кожного типу картки використовуються різні префаби, шрифти або стилі;
- у майбутньому можна легко розширити систему новими типами карток без зміни основної логіки створення.

Такий підхід дозволяє централізовано контролювати всі параметри створення об'єктів, зменшує дублікацію коду та спрощує підтримку структури проєкту.

2.2.4 Command

Патерн Command реалізовано у механізмі блокнота гравця. Кожен вибір (наприклад, обрана картка категорії "зброя" зі значенням "Молот") представляється у вигляді окремої команди, яка зберігається у списку подій.

Особливості застосування:

- кожна команда має структуру: тип дії, параметри, час виконання;
- записи зберігаються у лог подій (можливо в майбутньому для Undo або перегляду історії);
- можна реалізувати "відмотку" ходів або підсвічування останніх дій.

Цей підхід дозволяє не лише зберігати інформацію, а й робити її доступною для подальшого аналізу, відлагодження чи модифікації механік у наступних версіях гри.

2.2.5 MVC

Хоча в грі «Freedom of the Ghost» не використано повноцінну реалізацію патерна MVC (Model-View-Controller), архітектура побудована з чітким логічним розділенням обов’язків між модулями:

Модель (Model): Класи, які містять дані гри (наприклад, ClueData, CardData, GameState, PlayerSelection). Вони не містять логіки відображення або обробки UI, лише дані.

Контролери (Controller): Скрипти, що керують логікою (наприклад, GameManager, CardController, ChoiceHandler). Вони відповідають за ініціалізацію гри, генерацію карт, зчитування взаємодії з користувачем, запуск подій.

Представлення (View): UI-компоненти, які візуалізують дані (NotebookUI, CardUI, HintPanel, CanvasManager). Вони реагують на події й оновлюють інтерфейс у відповідь на зміну стану.

Це розділення дозволяє ефективно підтримувати, оновлювати та розширювати гру. Наприклад, при зміні зовнішнього вигляду картки не потрібно змінювати логіку генерації або зберігання даних — достатньо оновити компонент CardUI.

2.3 Сценарії використання (Use Case)

2.3.1 Опис базових сценаріїв

UC1: Початок гри: гравець запускає гру, обирає нову сесію, переглядає вступне досьє з історією злочину.

UC2: Взаємодія з духом: гравець перетягує одну з карток дії на зону привида. Привид реагує, надаючи підказку, яка пов’язана зі значенням попередньої дії або типом картки. Підказка зберігається у внутрішньому стані.

UC3: Отримання вибору: після взаємодії з привидом гравець бачить 3 нові варіанти карток (по одній з категорії: вбивця, місце, знаряддя, час). Гравець обирає одну — вона зберігається в блокноті разом із відповідною позначкою правильності (умовною).

UC4: Перегляд блокнота: гравець у будь-який момент може переглянути блокнот, де зафіксовано попередні вибори, підказки від духа, а також кольорові індикатори прогресу.

UC5: Завершення гри: після завершення декількох циклів вибору, гравцю пропонується зробити фінальне припущення (хто, де, коли, чим). Якщо вибір збігається з внутрішнім набором істини — гра завершується перемогою, інакше — поразкою.

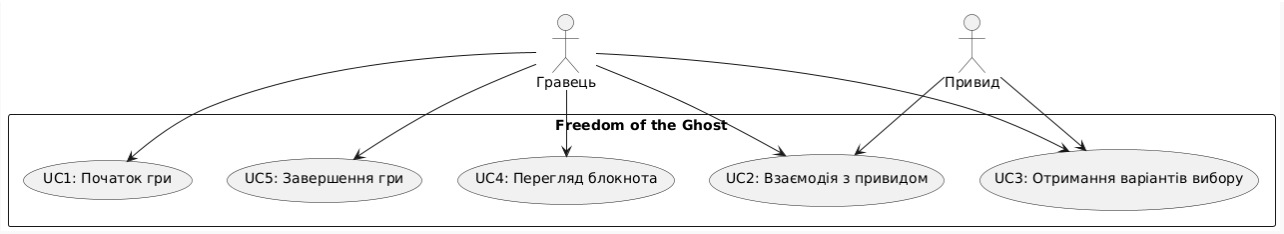


Рис. 2.3 Use Case діаграма

2.3.2 UML-діаграма класів

UML-діаграма включає основні класи: GameManager, Card, ClueSelector, SpiritBoard, UIManager, Ghost, NoteBook, TruthOrLiePanel. Показано їхні зв’язки, поля та методи.

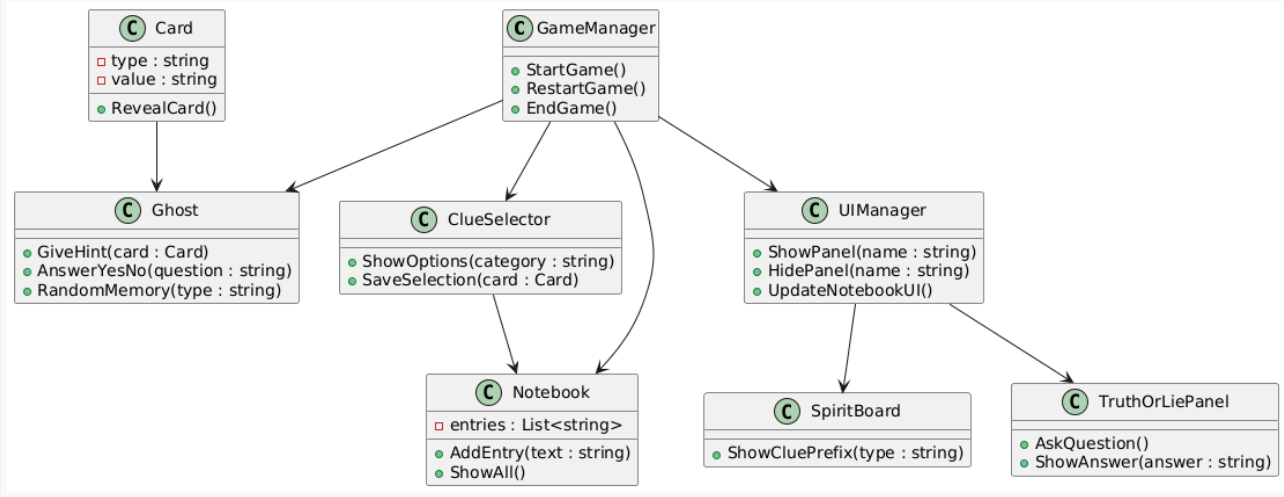


Рис. 2.4 UML-діаграма класів

2.4 Зовнішні інструменти

2.4.1 Draw.io

Draw.io — це безкоштовний онлайн-сервіс для створення UML-діаграм, діаграм процесів, архітектурних схем і логічних структур. Він був використаний для побудови наступних схем:

- Use Case diagram — описує основні сценарії взаємодії гравця з грою, включаючи перетягування картки, вибір дії, голосування тощо.
- Component diagram — відображає архітектурну структуру гри, взаємозв'язки між основними модулями (GameManager, UI, Card System,
- Class diagram — показує зв’язки між об’єктами: картками, підказками, блокнотом, гравцем, менеджерами тощо.

Draw.io дозволяє створювати інтуїтивно зрозумілі схеми, зручні для друку, вставки в диплом або демонстрації під час захисту. Всі створені діаграми було збережено безпосередньо на Google Drive, що дозволило:

- мати постійний доступ з будь-якого пристрою;
- швидко редагувати схеми при оновленні логіки гри;
- спільно працювати над діаграмами, не втрачаючи історії змін.

Завдяки простоті інтерфейсу та великій бібліотеці фігур, Draw.io став незамінним інструментом для документації технічної частини гри.

2.4.2 Git + GitHub

Git — це система контролю версій, що дозволяє ефективно керувати процесом розробки гри, особливо в умовах постійного оновлення логіки, графіки та інтерфейсу. У поєднанні з GitHub як хмарним репозиторієм було організовано зручне сховище коду, де зберігались усі етапи реалізації. Особливості використання:

- створено окремі гілки (branches) для різних задач: базова логіка, UI, тестування, вибір дій, блокнот;

- кожне оновлення супроводжувалося коментарем до коміту, що дозволяло відслідковувати, яка саме функціональність була змінена;
- у разі появи багів — було можливо відкотитися до стабільної версії;

GitHub забезпечив резервне копіювання, щоб уникнути втрати даних.

Крім того, використання Git у проєкті сприяло організованому підходу до програмування, що особливо важливо під час командної роботи або подальшого супроводу гри.

2.4.3 TextMeshPro

TextMeshPro — це розширення системи тексту в Unity, яке забезпечує високоякісний рендеринг тексту, широкий набір стилів і велику гнучкість у налаштуванні.

Використання TextMeshPro у грі дозволило досягти наступних результатів:

- висока чіткість і контрастність тексту навіть при масштабуванні інтерфейсу;
- уніфікований стиль оформлення карток, підказок від привида, діалогів, меню та інших елементів;
- можливість задавати різні кольори, тіні, межі, що підвищує візуальну привабливість;
- підтримка багатомовності, що є перспективною функцією для локалізації гри.

Зокрема, TextMeshPro було використано в наступних елементах:

- SpiritBoard — для відображення підказки від привида;
- панель дій — для чіткого відображення назв карток;
- блокнот гравця — для зберігання та перегляду вибраних доказів;
- всі текстові кнопки та заголовки UI.

Завдяки використанню TextMeshPro, інтерфейс гри виглядає професійно, легко читається на будь-якому екрані та дозволяє швидко сприймати інформацію, що особливо важливо в детективному жанрі, де кожна підказка має значення.

Рис. 2.5 Реалізація TextMeshPro

2.4.4 Unity Profiler

Unity Profiler — це потужний вбудований інструмент, який надає детальний аналіз продуктивності гри під час виконання. У процесі розробки гри «Freedom of the Ghost» Unity Profiler відіграв ключову роль у перевірці стабільності та оптимізації ресурсоємних елементів, таких як анімації, UI-компоненти, логіка перетягування карток і ефекти підказок.

Основні функції та використання:

- Вимірювання FPS (Frames Per Second): дозволяло відслідковувати, чи зберігається стабільна частота кадрів (60 FPS і вище) навіть при одночасному завантаженні кількох об'єктів на сцені, таких як картки, ефекти SpiritBoard, блокнот, панелі дій тощо.
- Моніторинг використання пам’яті: було перевірено, чи всі об'єкти коректно звільняються після завершення гри або переходу між сценами. Це дало змогу виявити та усунути можливі витоки пам’яті (memory leaks).
- Перевірка навантаження на CPU і GPU: за допомогою Profiler можна було оцінити, які саме компоненти (анімовані UI-панелі, оновлення логіки вибору, підказки примари тощо) найбільше впливають на продуктивність, і за потреби оптимізувати їхню реалізацію.

Тестування на різних пристроях: Unity Profiler дозволяє запускати віддалене профілювання (Remote Profiling), що використовувалося для аналізу поведінки гри на мобільних пристроях середнього рівня. Це дало змогу забезпечити кросплатформену стабільність гри.

Оцінка часу виконання методів: профайлер показував, скільки мілісекунд витрачається на виконання кожного методу в Update, LateUpdate, FixedUpdate тощо — це дозволило виявити критичні точки, які могли викликати затримки або зависання.

Результати:

Після застосування профайлінгу було підтверджено, що:

- гра не перевантажує систему навіть при тривалих сесіях;
- пам’ять використовується ефективно;
- всі UI-елементи оптимізовані;
- анімації та ефекти не створюють фризів або лагів.

Таким чином, Unity Profiler став незамінним інструментом не тільки для виявлення потенційних проблем, але і для гарантування плавного, стабільного ігрового досвіду користувача. Його регулярне використання на етапах розробки та тестування дозволило забезпечити технічну якість і оптимізацію гри згідно з сучасними вимогами.

3. РЕАЛІЗАЦІЯ ІГРОВОЇ ЛОГІКИ

3.1 Основна структура проєкту

Проєкт гри "Freedom of the Ghost" реалізовано у середовищі Unity з використанням мови програмування C#. Структура проєкту побудована за принципом розділення відповідальностей між основними компонентами гри. Кожна функціональна частина реалізована в окремому скрипті або групі скриптів, що дозволяє підтримувати гнучкість та масштабованість коду.

3.1.1 Менеджер гри

Менеджер гри (GameManager) виконує функцію центрального контролера, що відповідає за перебіг ігрового процесу. Він ініціалізує стартову сцену, завантажує конфігурацію гри, зберігає та змінює стан гри. У нашому проєкті GameManager використовує патерн Singleton для забезпечення глобального доступу до себе. У методі Start() відбувається ініціалізація карток, запуск першої підказки через SpiritBoard та встановлення таймерів взаємодії. Крім того, GameManager відповідає за обробку перемоги та поразки, керування інтерфейсом, паузою гри та взаємодію з іншими сервісами — наприклад, журналом підказок або історією гравця. Усі дані про гру зберігаються у внутрішніх структурах типу Dictionary<string, object>, що дозволяє зручно логувати події та вести аналітику під час проходження.

3.1.2 Система карток

Уся логіка побудована на використанні ScriptableObject. Кожна картка має власний клас-нащадок базового CardSO, який містить тип картки (запит, підказка, перевірка, доля), асоційоване текстове повідомлення, візуальну іконку, можливу взаємодію. Картки обробляються за допомогою CardManager, який випадково обирає одну з трьох нових карток певного типу для гравця після кожної дії. Завдяки розділенню логіки у ScriptableObject ми легко можемо додавати нові типи карт, не

змінюючи основну структуру проєкту. Картки з типом "Підказка" пов’язані з механікою SpiritBoard, а "Запитання до долі" — із випадковим генератором правдивої чи хибної інформації, що створює ефект гри з привидом.

3.1.3 Сценарій підказок

Підказки створено за принципом наративного дерева: кожна підказка має змістовне навантаження й орієнтована на категорію (вбивця, зброя, місце, час). Вони зберігаються як текстові файли або ScriptableObject. SpiritBoard, отримуючи першу карту, викликає обробник типу "дві літери" з ключового слова справжньої відповіді. Усі підказки містять внутрішній тег, що дозволяє відстежити, скільки разів гравець бачив подібну інформацію. Це важливо для реалізації механіки повторюваності та змішування правди з вигадкою. Обробник підказок вбудовано в HelpManager, який уміє відображати підказку, зберігати її в блокноті гравця та активувати аудіо- або візуальний ефект.

3.2 Візуалізація інтерфейсу

Візуальна частина гри реалізована за допомогою системи Canvas у Unity, що дозволило ефективно структурувати всі інтерфейсні елементи та забезпечити взаємодію гравця з грою.

3.2.1 SpiritBoard

SpiritBoard — це містичний інтерфейсний елемент, з якого починається гра. При старті на екрані з’являється дошка з алфавітом, де підсвічуються лише перші дві літери правильної підказки з однієї з чотирьох категорій (місце, знаряддя, вбивця або час). Підсвічування відбувається за допомогою зміни кольору або яскравості відповідних букв.

Цей елемент допомагає гравцеві отримати першу тематичну підказку й формує атмосферу загадковості на початку гри. У реалізації використано Canvas +

TextMeshPro, а ефекти підсвічування створено через зміну кольору тексту або анімацію появи (fade-in).

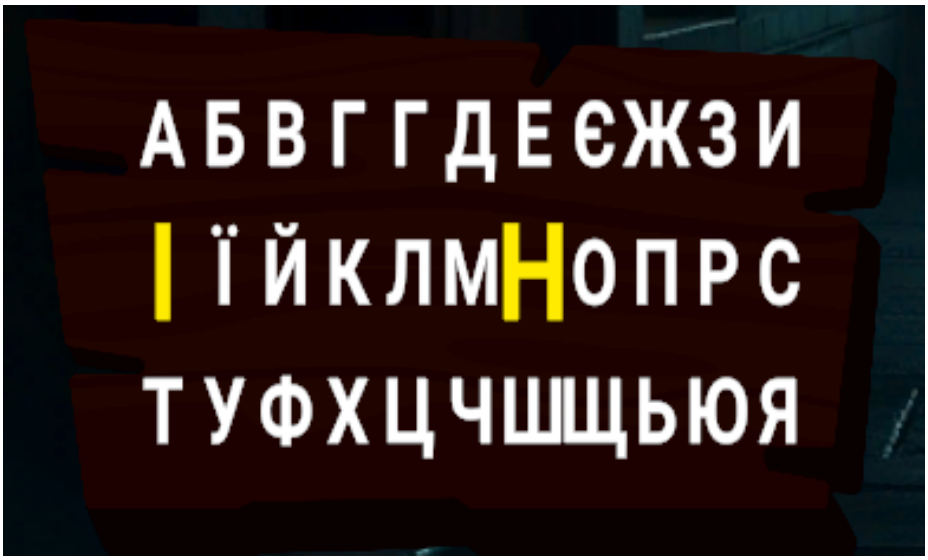


Рис. 3.1 SpiritBoard

3.2.2 Блокнот гравця

Блокнот — це окремий елемент інтерфейсу, в якому зберігаються усі отримані гравцем підказки після взаємодії з духом. Після кожного вибору однієї з трьох запропонованих карток, її текст автоматично додається до відповідної категорії в блокноті: вбивця, знаряддя, місце або час.

Інтерфейс реалізовано за допомогою TextMeshPro для чіткого відображення тексту та VerticalLayoutGroup чи GridLayoutGroup для впорядкованого розміщення записів у формі списку. Гравець має можливість переглядати весь журнал підказок, але не може змінювати попередні вибори.

Цей компонент виконує роль "історії розслідування", яка допомагає гравцеві логічно аналізувати вже отриману інформацію та наближатися до розв’язання справи.

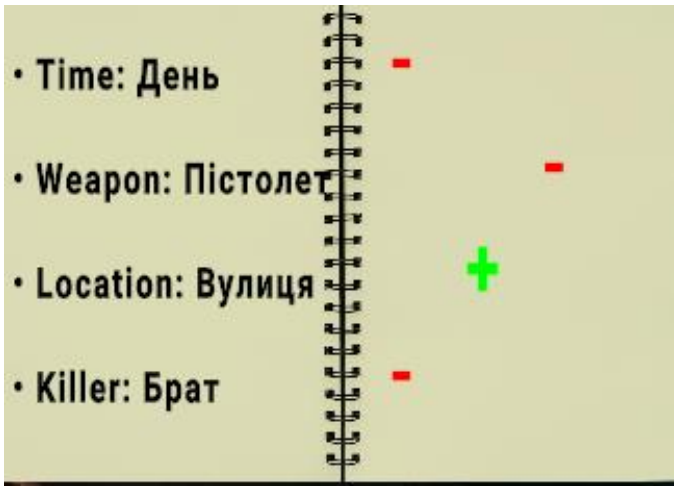


Рис. 3.2 Блокнот

3.2.3 Вікно "Правда чи Брехня"

Окремий UI-екран, що з’являється після активації картки відповідного типу. Містить поле для введення питання та кнопку підтвердження. Після натискання на кнопку привид відповідає "Так" або "Ні", причому відповідь може бути або правдивою, або випадковою (залежно від карти). Реалізовано за допомогою панелі вводу TextMeshPro InputField, а відповіді виводяться через анімоване повідомлення. Також передбачено блокування подальшої взаємодії до завершення анімації.

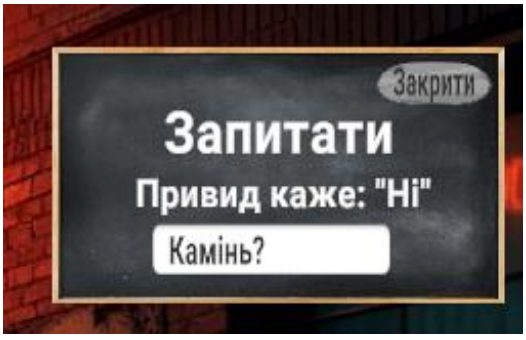


Рис. 3.3 Вікно "Правда чи Брехня"

3.2.4 Вікно результатів

Фінальний екран, що показує результат гри (перемога чи поразка). В залежності від правильності вибору гравця, виводиться відповідне повідомлення. Має кнопки перезапуску гри або повернення в меню або нової гри. Реалізовано за допомогою Canvas Group.

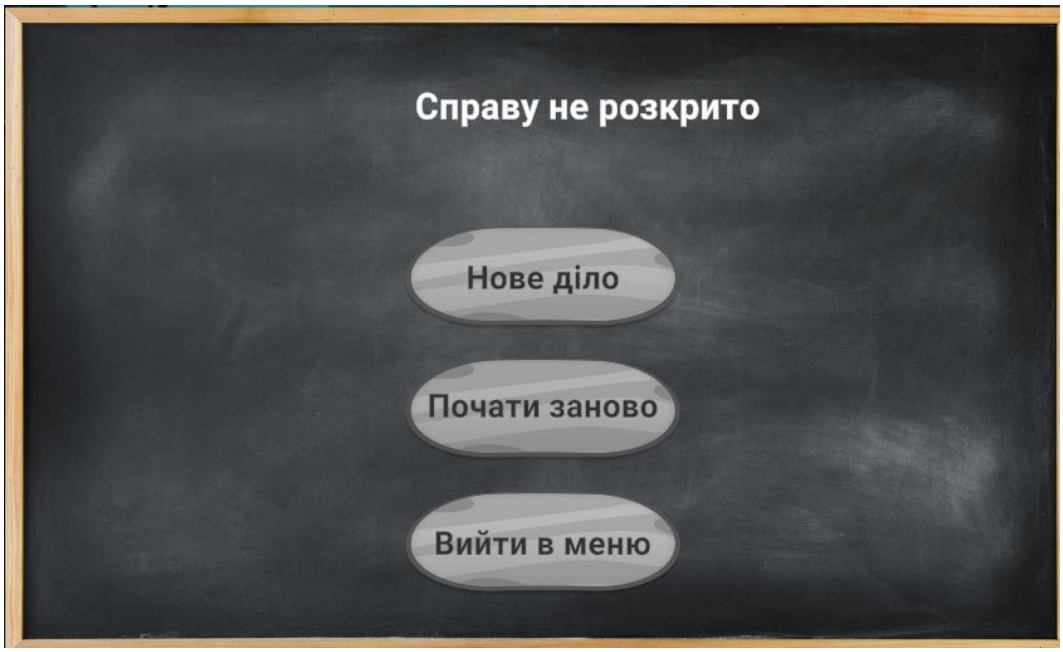


Рис. 3.4 Вікно результатів

3.2.5 Панель вибору карток

Інтерфейс, що з’являється після кожного дії. Гравець бачить 3 нові картки одного типу (місце, зброя, тощо), з яких має обрати одну. Кожна картка реалізована як кнопка з ілюстрацією, заголовком і коротким описом. При наведенні або натисканні використовується візуальний ефект підсвічування. Панель зникає після вибору, а обрана картка записується в блокнот.

Рис. 3.5 Панель вибору карток

3.3 Взаємодія користувача

У грі "Freedom of the Ghost" взаємодія користувача побудована навколо поетапного отримання підказок через серію карткових дій, а також комунікації з привидом. Весь інтерфейс реалізовано за допомогою Canvas UI у Unity з використанням кнопок, панелей, анімацій і системи перетягування.

3.3.1 Перетягування карт

Однією з ключових механік у грі "Freedom of the Ghost" є система перетягування карток (Drag-and-Drop), яка забезпечує інтуїтивну взаємодію між гравцем і ігровим середовищем. Саме через цей механізм реалізується основна форма комунікації з ігровим світом — гравець фізично "переносить" свою дію, що створює ефект занурення і прямої участі в розслідуванні.

Технічна реалізація:

Механізм перетягування побудований на основі інтерфейсів подій Unity:

- IPointerDownHandler — обробляє подію натискання на картку (реєструє, що гравець почав перетягування).
- IPointerDragHandler — активується під час руху миші або пальця і оновлює позицію об’єкта у реальному часі.
- IPointerUpHandler — викликається, коли гравець відпускає картку (завершення перетягування), після чого виконується перевірка на досягнення цільової області.

У якості цільових областей можуть виступати:

- зона привида (GhostArea) — використовується для активації підказки;
- спеціальні UI-елементи — для ініціації дій на основі типу карти (наприклад, перевірка, підказка, вибір долі тощо).

Поведінка під час перетягування:

- Картка візуально відокремлюється від колоди: її шар (sorting layer) змінюється, щоб вона завжди була поверх інших елементів інтерфейсу.

- Відтворюється звук перетягування або короткий аудіоефект (наприклад, шурхіт паперу або ехо, якщо перетягування йде до духа), що підсилює атмосферу.
- У момент наведення на допустиму зону активації підсвічується сама область або об’єкт (Outline, Glow, або інші ефекти).

Обробка результату:

Після того, як картка скинута на допустимий об’єкт:

- Система виконує перевірку колізії (наприклад, через RectTransformUtility.RectangleContainsScreenPoint або кастомну логіку визначення зони).
- Якщо все виконано правильно, викликається відповідна подія: наприклад, з’являється підказка, відкривається вікно вибору, або активується анімація духа.
- Якщо гравець відпустив картку в недопустимій області — вона повертається на місце з анімацією (через DOTween, Lerp, або Coroutine для плавного повернення).

Переваги такого підходу:

- Інтуїтивність — перетягування зрозуміле навіть без навчання, що робить гру доступною для новачків.
- Візуальна взаємодія — гравець бачить результат своїх дій у реальному часі.
- Гнучкість розробки — drag-and-drop можна масштабувати на інші взаємодії: переміщення предметів, комбінації карт, тощо.
- Атмосферність — візуальні й звукові ефекти роблять дії гравця емоційно насиченими.

Таким чином, система перетягування карт у грі є не просто способом введення даних, а невіддільною частиною ігрового досвіду, яка поєднує логіку, атмосферу та геймдизайн у єдине ціле. Вона підкреслює ключовий принцип гри — "торкнись підказки, відчуй історію".

3.3.2 Обробка вибору

У грі "Freedom of the Ghost" після виконання кожної дії гравець отримує можливість зробити вибір серед трьох випадкових карток, що належать до однієї з чотирьох категорій: вбивця, місце, знаряддя або час. Така механіка реалізована для поетапного збору доказів та забезпечення прогресивного розвитку сюжетної лінії, при цьому кожен вибір має значення і наближає гравця до розв'язання загадки.

Механіка появи вибору: Після завершення активної дії (наприклад, отримання підказки від привида або активації спеціальної картки) в інтерфейсі гри з’являється панель вибору, яка містить три унікальні картки з варіантами лише однієї категорії. Вибрана категорія визначається випадковим чином на початку кожного циклу, що гарантує різноманітність ігрового процесу.

Ці картки відображаються за допомогою UI-кнопок, кожна з яких містить текстовий опис варіанту (наприклад, "Ніж", "Ванна", "Томас" або "3:00 ночі").

Кожна така кнопка — це інтерактивний елемент, який реагує на натискання та обробляється спеціальним скриптом (наприклад, CardSelectionHandler або част иною логіки GameManager).

Поведінка після вибору:

Після натискання на одну з трьох карток:

1. Вибране значення записується до блокнота гравця — це окрема панель, у якій фіксуються всі обрані версії доказів.
2. Інші дві картки зникають з екрана, забезпечуючи чистоту інтерфейсу та акцент на наступному кроці гри.
3. Гра переходить до наступного етапу — наприклад, запускається новий хід або відкривається інтерфейс для перетягування чергової картки дії.

Уся логіка цього процесу реалізована через C#-скрипти в Unity, які централізовано обробляють події: виявлення натискання (OnClick), оновлення інтерфейсу, запис у структуру даних блокнота (NotebookManager.AddEntry()), а також виклик функцій переходу до нового стану гри.

Важливість цього етапу:

Обробка вибору — це ключова механіка, яка поєднує гравця зі стратегічним елементом гри: кожен зроблений вибір несе певний інформаційний слід, який гравець має аналізувати у подальшому.

Це стимулює логічне мислення, вміння порівнювати підказки та виключати неправильні варіанти.

Крім того, така форма вибору дозволяє легко розширювати гру в майбутньому, додаючи нові типи карт, нові категорії або спеціальні ефекти карт без зміни основної архітектури.

Таким чином, система обробки вибору в грі реалізована як інтуїтивно зрозумілий, але стратегічно важливий етап, що дає гравцеві змогу формувати власну логіку розслідування.

Завдяки використанню Unity UI, подій кнопок і зберігання вибору в спеціальну структуру, ця механіка працює стабільно, логічно та зручно для гравця.

4. ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Технічне середовище

Для тестування гри використовувалося наступне апаратно-програмне середовище:

- Операційна система: Windows 10 Pro x64, версія 22H2
- Процесор: Intel Core i5-10400F (6 ядер, 2.90 GHz)
- Оперативна пам’ять: 16 GB DDR4
- Відеокарта: NVIDIA GeForce GTX 1660 Super
- Ігровий рушій: Unity 2021.3.34f1 LTS
- Розширення: TextMesh Pro, Universal Render Pipeline (URP)
- Відеороздільність для тестів: 1920x1080 (Full HD)
- Моніторинг продуктивності: Unity Profiler, Windows Performance Monitor

Також гра перевірялась на ноутбуці з нижчими характеристиками (Intel i3, 8 GB RAM, інтегрована графіка), щоб перевірити мінімальні вимоги до запуску.

4.3.2 Продуктивність клієнта

Інструменти: Unity Profiler, Task Manager, Memory Diagnostic

Вимірювання:

- Середній FPS: 60 FPS на FullHD роздільності.
- Навантаження на GPU: 20-35%
- Пам’ять: ~350 MB у піку, при активних сценах

Результати:

- Гра стабільно працює на пристроях середнього рівня.
- Відсутні memory leaks.
- Завантаження нових сцен не призводить до зависань (до 0.5 сек max).
- Оптимізація спрайтів та використання URP покращує продуктивність без втрати якості.

4.2 Юзер-кейси для перевірки

4.2.1 Вибір карти дії

Мета: Перевірити, чи гравець може вибрати одну з трьох випадкових карт, які з’являються після кожної дії.

Кроки:

1. Завершити взаємодію з поточною картою.
2. Дочекатися появи вибору з трьох нових карт.
3. Натиснути на одну з них.
4. Переконатися, що вона зникає з вибору, а її значення зберігається в блокноті гравця.

Результат: Усі три карти відображаються коректно, одна з них зберігається, дві інші зникають. Усі дії коректно логуються.

Рис. 4.1 Момент вибору однієї з трьох карт

4.2.2 Отримання підказки від привида

Мета: Перевірити механіку генерації підказки при взаємодії з духом.

Кроки:

1. Перетягнути карту типу "Підказка" або "Запитання" на зону духа.
2. Очікувати на виведення тексту підказки.
3. Звірити категорію підказки з типом вхідної карти.

Результат: Після перетягування у 100% випадків з’являється підказка, текст відображається у відповідному вікні з коректною категорією (вбивця, зброя, місце, час).

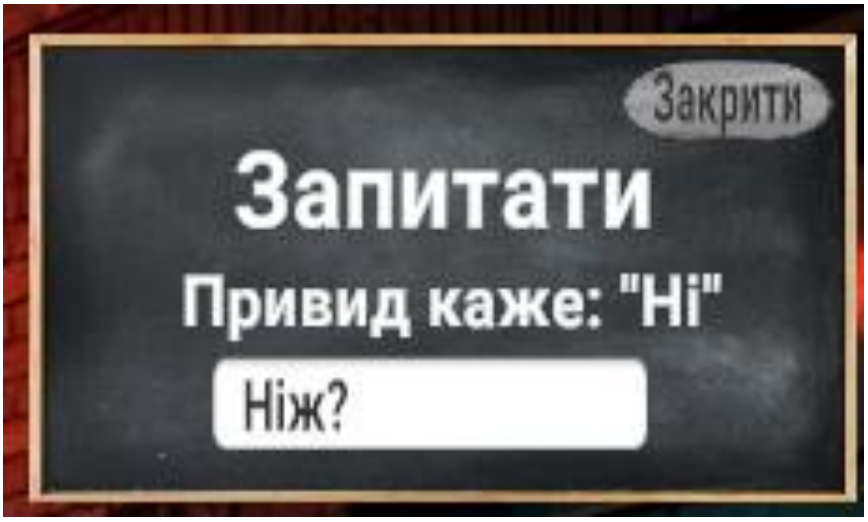


Рис. 4.2 Вікно з підказкою

4.2.3 Збереження підказки до блокнота

Мета: Перевірити, чи кожна підказка записується до блокнота гравця.

Кроки:

1. Отримати підказку через SpiritBoard або карту.
2. Подивитися блокнот гравця.
3. Знайти нову підказку серед записів за категорією.

Результат: Підказка одразу з’являється у відповідній вкладці блокнота. Інтерфейс не зависає, категорія підсвічується кольором.

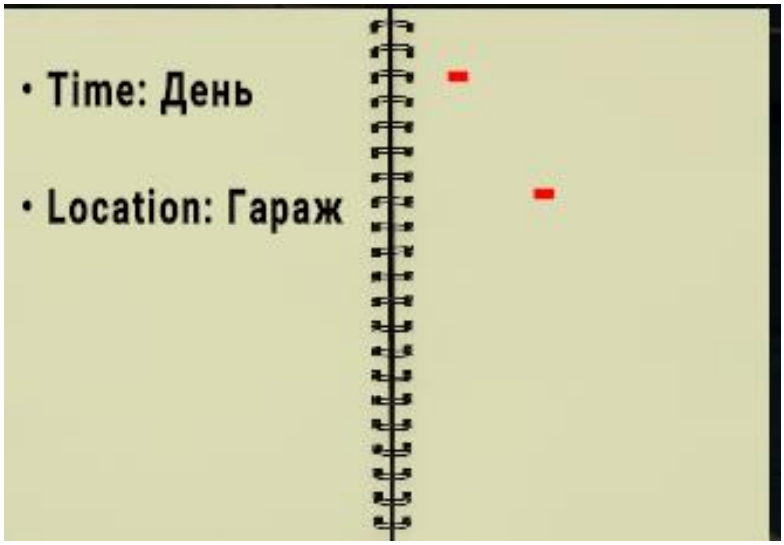


Рис. 4.3 Блокнот гравця з новою підказкою

4.2.4 Повернення до головного меню, або перезапуск

Мета: Перевірити, чи правильно працює кнопка повернення до меню з обнуленням гри, та перезапуск гри.

Кроки:

1. Пройти всі етапи й отримати повний набір підказок.
2. Почекати, поки відкриється головне меню.
3. Запустити нову гру, або вийти у головне меню

Результат: Всі тимчасові змінні обнуляються, сцена завантажується з початку. Перезапускається гра з тими ж самим параметрами.

Рис. 4.4 Головне меню після натискання “Вийти в меню”

4.3 Аналіз стабільності роботи

4.3.1 Поведінка при втраті підключення

Оскільки гра не використовує онлайн-мультиплеєр, перевірка стосувалась випадків втрати доступу до ресурсів — наприклад, недоступність локальних asset-ів, видалення файлів тощо.

Тести:

1. Примусове видалення спрайтів карт.
2. Відключення мережі (перевірка на предмет виклику помилок).
3. Завантаження пошкоджених сцен.

Результат: Усі помилки обробляються через try-catch блоки. Гра не аварійно завершується. Виводиться повідомлення про відсутність ресурсу. Система GameManager має резервне завантаження за замовчуванням.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було реалізовано ігровий прототип цифрової детективної гри «Freedom of the Ghost» у форматі покрокової карткової взаємодії з елементами містики. Гра створена за допомогою ігрового рушія Unity та мови програмування C#.

1. Для розробки було обрано рушій Unity через зручність роботи з UI, підтримку C# та широкі можливості роботи з анімацією і подіями. Використано Canvas, EventSystem, TextMeshPro, а також систему перетягування (Drag-and-Drop) і анімацій (DoTween).

2. Було спроєктовано модульну структуру, що включає GameManager, UIManager, CardManager, SpiritBoard, NoteBook, GhostController. Для взаємодії між модулями застосовано патерни Singleton, Dependency Injection, а також ScriptableObject для карток.

3. Реалізовано чотири основні типи карт (підказка, перевірка, доля, дошка), SpiritBoard для початкової підказки, блокнот для збереження вибору, та вибір дій із трьох варіантів. Уся логіка поділена на окремі скрипти з чіткою відповідальністю.

4. Інтерфейс створено на основі Canvas. У грі реалізовано: головне меню, панель вибору дій, SpiritBoard з підсвічуванням літер, вікно підказок, блокнот гравця, вікно результатів. Анімації додають плавності, а текст оформлений через TextMeshPro.

5. Проведено тестування базових сценаріїв: вибір картки, взаємодія з духом, поява підказки, завершення гри, повернення до меню. Усі механіки працюють стабільно, без помилок і зависань. Логіка підказок і збереження вибору працює коректно.

6. У майбутньому доцільно розширити гру за рахунок:

- збереження прогресу між сесіями;
- більшої кількості карток та підказок;

- складніших сценаріїв підказок;
- розширення логіки перевірки та реакцій духа;
- підтримки локалізації.

Результат відповідає поставленим цілям: створено прототип гри, що поєднує детективний сюжет, інтуїтивний інтерфейс і послідовну логіку взаємодії з гравцем.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Манківський Д.С., Ярошевський О.В. Визначення вимог до детективної гри у форматі карт та вибору варіантів дій "Freedom Of The Ghost". Матеріали: VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях 24.04.2025, ДУІКТ, Київ, Україна, С.243-245.
2. Манківський Д.С., Ярошевський О.В., Визначення вимог до ігрового рушія UNITY для розробки детективної гри у форматі карт та вибору варіантів дій. Метеріали: VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24.04.2025, ДУІКТ, Київ, Україна, С.246-247.

ПЕРЕЛІК ПОСИЛАНЬ

1. Gregory J. *Game Engine Architecture*. 3rd ed. A K Peters/CRC Press, 2022. 1248 с.
2. Troelsen A., Japikse P. *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming*. 11th ed. New York: Apress, 2022. 1215 с.
3. *Unity User Manual*. Unity Technologies. URL: https://docs.unity3d.com/Manual/index.html (дата звернення: 20.05.2024).
4. *Creating and Using Scripts*. Unity Technologies. URL: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html (дата звернення: 20.05.2024).
5. *TextMeshPro User Manual*. Unity Technologies. URL: https://docs.unity3d.com/Packages/com.unity.textmeshpro@4.0/manual/index.html (дата звернення: 20.05.2024).
6. *Introduction to ScriptableObjects*. Unity Technologies. URL: https://learn.unity.com/tutorial/introduction-to-scriptableobjects (дата звернення: 20.05.2024).
7. *DOTween: The right choice*. Demigiant. URL: http://dotween.demigiant.com/ (дата звернення: 20.05.2024).
8. *UI Canvas*. Unity Technologies. URL: https://docs.unity3d.com/Manual/UICanvas.html (дата звернення: 20.05.2024).
9. *Diagrams.net*. JGraph. URL: https://app.diagrams.net/ (дата звернення: 20.05.2024).
10. *GitHub*. GitHub, Inc. URL: https://github.com/ (дата звернення: 20.05.2024).
11. *Unity Learn*. Unity Technologies. URL: https://learn.unity.com/ (дата звернення: 20.05.2024).
12. *Unity Asset Store*. Unity Technologies. URL: https://assetstore.unity.com/ (дата звернення: 20.05.2024).

13. *Profiler overview*. Unity Technologies.
URL: https://docs.unity3d.com/Manual/Profiler.html (дата звернення: 20.05.2024).
14. *Unity Events*. Unity Technologies.
URL: https://docs.unity3d.com/Manual/UnityEvents.html (дата звернення: 20.05.2024).
15. *Newest 'unity3d' Questions*. Stack Overflow.
URL: https://stackoverflow.com/questions/tagged/unity3d (дата звернення: 20.05.2024).
16. *Unity Community Forums*. Unity Technologies.
URL: https://forum.unity.com/ (дата звернення: 20.05.2024).
17. *UI Drag and Drop*. Unity Technologies. URL: https://learn.unity.com/tutorial/ui-drag-and-drop (дата звернення: 20.05.2024).
18. *Input System*. Unity Technologies.
URL: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/index.html (дата звернення: 20.05.2024).
19. *UI Button*. Unity Technologies. URL: https://docs.unity3d.com/Manual/script-Button.html (дата звернення: 20.05.2024).
20. *Animator Component*. Unity Technologies.
URL: https://docs.unity3d.com/Manual/Animator.html (дата звернення: 20.05.2024).
21. *Audio*. Unity Technologies.
URL: https://docs.unity3d.com/Manual/Audio.html (дата звернення: 20.05.2024).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка детективної гри "Freedom of the Ghost"

Виконав студент 4 курсу
Групи ПД-43
Манківський Денис Сергійович
Керівник роботи
Ассистент кафедри ІПЗ Ярошевський Олександр Вікторович

Київ – 2025

1

Мета, об’єкт та предмет дослідження

- **Мета дослідження** – посилити містично-детективну складову гри шляхом використання специфічних об’єктів, що формують атмосферу розслідування та занурюють гравця у процес пошуку істини.
- **Об’єкт дослідження** - процес створення інтерактивної детективної гри з використанням карткової системи вибору та побудови логіки підказок.
- **Предмет дослідження** - технології та інструменти реалізації інтерактивної детективної гри з картковим інтерфейсом.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі детективні ігри з елементами інтерактивного розслідування та варіативного сюжету.
2. Визначити переваги та недоліки ігор з картковим інтерфейсом та механікою вибору дій.
3. Сформулювати функціональні та нефункціональні вимоги до детективної гри "Freedom of the Ghost".
4. Спроєктувати архітектуру гри, визначити основні класи, компоненти та взаємодію між ними.
5. Розробити інтерфейс користувача з урахуванням карткової механіки та вибору дій.
6. Реалізувати ключову ігрову логіку: механізм збору доказів, вибір карток, підказки від привида.
7. Провести модульне та ручне тестування гри, перевірити логіку сценаріїв і варіативність розвитку подій.

3

АНАЛІЗ АНАЛОГІВ

Функції	**Simulacra**	**Duskwood**	**Card Detective**	Freedom of the Ghost
Детективний сюжет	+	+	+	+
Використання підказок/вибору	+	+	-	+
Карткова механіка	-	-	+	+
Наявність привида	-	-	-	+
Рандомне формування справи на початку	-	-	-	+
Вибір із декількох відповідей	+	+	+	+
Ведення записів/блокнота	-	-	+	+
Можливість повторного проходження	+	+	+	+

4

КОНЦЕПТ ГРИ

Freedom of the Ghost — це атмосферна покрокова детективна гра з картковим інтерфейсом, у якій гравець розслідує таємниче вбивство за допомогою підказок від привида жертви. Кожна нова справа — це унікальне поєднання підозрюваного, місця, знаряддя та часу злочину, які гравець має встановити, досліджуючи підказки, аналізуючи ситуацію та роблячи вибір дій за допомогою спеціальних карт.

Сеттинг гри:
Зловісне місто, де ніч приховує більше, ніж просто темряву. Гравець — невидимий детектив, який через містичний зв'язок із духом жертви повинен розкрити правду, яку не бачить ніхто інший.

Технічні параметри:
• Платформи: Windows, Android
• Керування: мишка, клавіатура (ПК) або сенсорний екран (мобільні пристрої)
• Графіка**:** 2D-інтерфейс із текстовими та візуальними підказками

Цільова аудиторія: любителі детективів, психологічних ігор та загадок; гравці віком від 12 років, що шукають нестандартний сюжет та атмосферний досвід; прихильники ігор із варіативністю дій та елементами містики

ГЕЙМПЛЕЙ ТА МЕХАНІКИ

Геймплей:
Гравець досліджує справу, використовуючи картки дій для взаємодії з привидом жертви. Отримуючи підказки, гравець обирає варіанти відповідей і занотовує свої здогадки у блокнот, поступово наближаючись до розкриття істини.

Ігрові механіки:
- Покроковий вибір карток дій — Memory, Fate, Truth, Board, кожна з яких розкриває частину справи по-своєму.
- Привид вказує перші літери відповіді через містичний інтерфейс.
- Непрямі фрази, асоціації та спогади привида без прямої відповіді.
- Кожна справа починається з унікального тексту, який створює атмосферу і містить натяки.
- Блокнот розслідування фіксує всі дії гравця з позначенням правильності (+/–).
- Система прогресу — чотири етапи (вбивця, місце, знаряддя, час) з візуальними індикаторами правильності.
- Щоразу нове досьє та комбінації параметрів.

Вимоги до гри "Freedom of the Ghost"

Функціональні вимоги:

1. Генерація справи з випадковими параметрами (вбивця, зброя, місце, час).
2. Надання непрямих підказок від привида для спрямування гравця.
3. Відображення досьє з описом злочину на початку гри.
4. Взаємодія з грою шляхом вибору дій за допомогою карток.
5. Збереження виборів гравця у блокноті
6. Візуальне позначення правильності відповідей на шкалі прогресу.
7. Можливість почати нову справу або переграти поточну.
8. Виведення фінальної панелі з результатами та подальшими опціями.

Нефункціональні вимоги:

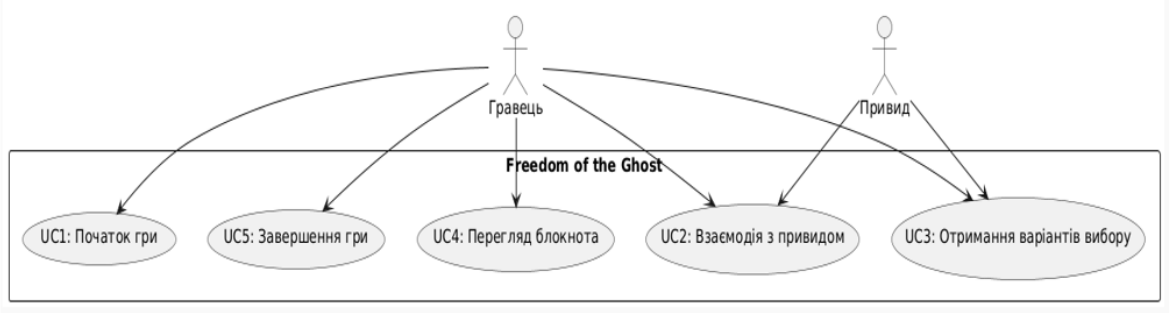
1. Завантаження карток (не більше 3 сек).
2. Можливість додавання нових сценаріїв, карток і механік.
3. Можливість керувати гучністю, повноекранним режимом та поверненням до головного меню.
4. Стабільна робота гри без збоїв.
5. Підтримка основних платформ (Windows, Android).

7

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

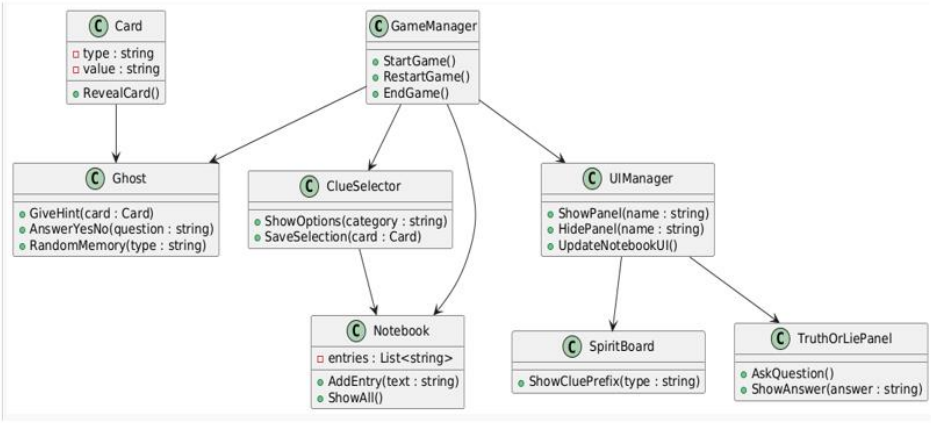
8

Use case діаграма



9

Діаграма класів

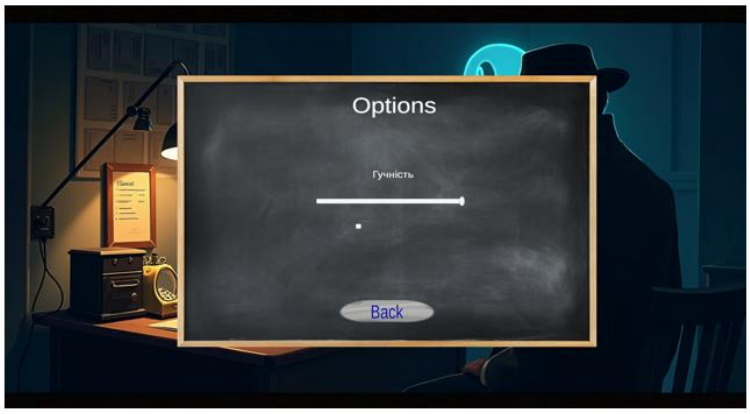


10

Головне меню

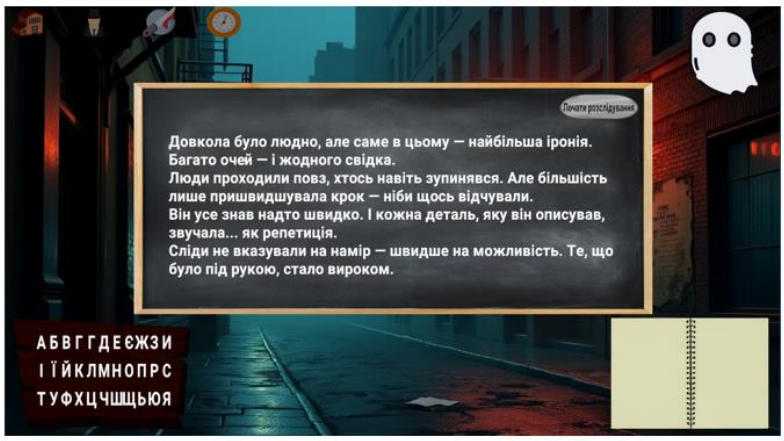
11

Меню налаштування



12

Екран ігрового процесу



13

Зображення інтерфейсу

14

Меню після завершення гри

15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Манківський Д.С., Ярошевський О.В. Визначення вимог до детективної гри у форматі карт та вибору варіантів дій "Freedom Of The Ghost". Матеріали: VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях 24.04.2025, ДУІКТ, Київ, Україна, с.243

2. *Манківський Д.С.*, *Ярошевський О.В.*, Визначення вимог до ігрового рушія UNITY для розробки детективної гри у форматі карт та вибору варіантів дій. Метеріали: VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24.04.2025, ДУІКТ, Київ, Україна, с.246

16

Висновки

1. Проведено аналіз детективних ігор з інтерактивними розслідуваннями, виявлено ключові підходи до реалізації варіативного сюжету та взаємодії з користувачем.
2. Оцінено переваги й недоліки карткових ігор з вибором дій. Зроблено висновок, що картковий інтерфейс сприяє кращому залученню гравця та зручній візуалізації процесу розслідування.
3. Визначено функціональні та нефункціональні вимоги до гри "Freedom of the Ghost", які забезпечують чітку структуру розробки.
4. Спроєктовано архітектуру гри, визначено основні компоненти (GameManager, SpiritBoard, GhostTarget, ChoiceManager тощо) та їх взаємозв’язки.
5. Створено користувацький інтерфейс, що реалізує карткову взаємодію, підказки, досьє, блокнот та панель завершення гри.
6. Реалізовано ключову логіку гри: генерація випадкових сценаріїв, обробка підказок від привида, фіксація рішень гравця та відображення прогресу.
7. Проведено тестування функціональних елементів, перевірено коректність варіативних сценаріїв, забезпечено стабільність ігрового процесу.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System.Collections.Generic;
using System.Linq;

public class GameManager : MonoBehaviour
{
    public static GameManager Instance;

    [Header("UI")]
    public GameObject choicePanel;
    public GameObject notebookPanel;
    public Transform choiceContainer;
    public CardDisplay cardPrefab;
    public TMP_Text ghostHintText;

    [Header("Game State")]
    private List<string> killerOptions = new List<string> { "Anna",
"Boris", "Carla" };
    private List<string> weaponOptions = new List<string> {
"Knife", "Poison", "Gun" };
    private List<string> locationOptions = new List<string> {
"Kitchen", "Library", "Garden" };
    private List<string> timeOptions = new List<string> {
"Morning", "Noon", "Night" };

    private string correctKiller;
    private string correctWeapon;
    private string correctLocation;
    private string correctTime;

    private void Awake()
    {
        if (Instance == null)
            Instance = this;
        else
            Destroy(gameObject);
    }

    void Start()
    {
        GenerateSolution();
        ShowFirstClue();
    }

    void GenerateSolution()
    {
        correctKiller = killerOptions[Random.Range(0,
killerOptions.Count)];
        correctWeapon = weaponOptions[Random.Range(0,
weaponOptions.Count)];
        correctLocation = locationOptions[Random.Range(0,
locationOptions.Count)];
        correctTime = timeOptions[Random.Range(0,
timeOptions.Count)];
    }

    void ShowFirstClue()
    {
        string clue = correctLocation.Substring(0, 2); // Наприклад
"Ki"
        ghostHintText.text = "Підказка: " + clue;
    }
```

```
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class NotebookManager : MonoBehaviour
{
    public TMP_Text notebookText;

    private List<string> entries = new List<string>();


    public void AddEntryFormatted(string category, string
choice, string resultSymbol)
    {
        string formattedSymbol = resultSymbol == "+"
            ? "<color=green><size=150%>+</size></color>"
            : "<color=red><size=150%>-</size></color>";

        string entry = $"• <b>{category}</b>: {choice}<t>
{formattedSymbol}";
        entries.Add(entry);
        UpdateNotebook();
    }


    public void AddEntry(SpiritBoard.ClueType type, string
choice)
    {
        string correctAnswer = GetCorrectAnswer(type);
        bool isCorrect = choice == correctAnswer;

        string formattedSymbol = isCorrect
            ? "<color=green><size=200%>+</size></color>"
            : "<color=red><size=200%>-</size></color>";

        string category = type.ToString();
        string entry = $"• <b>{category}</b>: {choice}\t
{formattedSymbol}";
        entries.Add(entry);
        UpdateNotebook();
    }


    public void ClearNotebook()
    {
        entries.Clear();
        notebookText.text = "";
    }


    private void UpdateNotebook()
    {
        notebookText.text = string.Join("\n", entries);
    }


    private string GetCorrectAnswer(SpiritBoard.ClueType
type)
    {
        switch (type)
        {
            case SpiritBoard.ClueType.Killer: return
GameManager.Instance.killer;
```

```
    public void OnCardDraggedToSpirit(CardType type)
    {
        string memory = GenerateMemory(type);
        ghostHintText.text = memory;
        ShowChoicePanel(type);
    }

    string GenerateMemory(CardType type)
    {
        switch (type)
        {
            case CardType.Killer: return "Біль… обличчя знайоме…";
            case CardType.Weapon: return "Холодний метал…";
            case CardType.Location: return "Шепіт… плитка під
ногами…";
            case CardType.Time: return "Світло… або темрява?..";
            default: return "…";
        }
    }

    void ShowChoicePanel(CardType category)
    {
        choicePanel.SetActive(true);
        foreach (Transform child in choiceContainer)
            Destroy(child.gameObject);

        List<string> options = GetOptionsByCategory(category);
        foreach (string option in options)
        {
            CardDisplay newCard = Instantiate(cardPrefab,
choiceContainer);
            newCard.Setup(option, category);
        }
    }

    List<string> GetOptionsByCategory(CardType type)
    {
        return type switch
        {
            CardType.Killer => killerOptions,
            CardType.Weapon => weaponOptions,
            CardType.Location => locationOptions,
            CardType.Time => timeOptions,
            _ => new List<string>()
        };
    }

    public bool CheckGuess(string guess, CardType type)
    {
        return type switch
        {
            CardType.Killer => guess == correctKiller,
            CardType.Weapon => guess == correctWeapon,
            CardType.Location => guess == correctLocation,
            CardType.Time => guess == correctTime,
            _ => false
        };
    }

    public void AddToNotebook(string value, CardType type)
    {
        // Зберігає вибір гравця у блокнот
        GameObject entry = new GameObject("NotebookEntry");
        entry.transform.SetParent(notebookPanel.transform);
        TMP_Text text = entry.AddComponent<TMP_Text>();
        text.text = $"{type}: {value}";
        text.fontSize = 24;
    }
            case SpiritBoard.ClueType.Weapon: return
GameManager.Instance.weapon;
            case SpiritBoard.ClueType.Location: return
GameManager.Instance.location;
            case SpiritBoard.ClueType.Time: return
GameManager.Instance.time;
            default: return "";
        }
    }
}
 using UnityEngine;
using TMPro;
using System.Collections.Generic;

public class DossierManager : MonoBehaviour
{
    public GameObject dossierPanel;
    public TMP_Text dossierText;
    public GameObject startButton;
    public GameObject reopenButton;
    public GameObject actionsPanel;

    private void Start()
    {
        if (dossierPanel != null)
        {
            dossierText.text = GenerateDossier();
            dossierPanel.SetActive(true);
        }

        if (startButton != null)
            startButton.SetActive(true);

        if (reopenButton != null)
            reopenButton.SetActive(false

    }

    public void CloseDossier()
    {
        dossierPanel.SetActive(false);

        if (actionsPanel != null)
            actionsPanel.SetActive(true);

        if (dossierPanel != null)
            dossierPanel.SetActive(false);

        if (reopenButton != null)
            reopenButton.SetActive(true);
    }

    public void ReopenDossier()
    {
        if (dossierPanel != null)
            dossierPanel.SetActive(true);

        if (reopenButton != null)
            reopenButton.SetActive(false);
    }

    private string GenerateDossier()
    {
        string killer = GameManager.Instance.killer;
        string weapon = GameManager.Instance.weapon;
        string location = GameManager.Instance.location;
        string time = GameManager.Instance.time;

        List<string> lines = new();
```

```
}

public enum CardType
{
   Killer,
   Weapon,
   Location,
   Time
}

// CardDisplay.cs — UI-картка з вибором

using UnityEngine;
using TMPro;
using UnityEngine.UI;

public class CardDisplay : MonoBehaviour
{
   public TMP_Text label;
   private string cardValue;
   private CardType cardType;

   public void Setup(string value, CardType type)
   {
      cardValue = value;
      cardType = type;
      label.text = value;
      GetComponent<Button>().onClick.AddListener(OnSelect);
   }

   void OnSelect()
   {
      bool isCorrect =
GameManager.Instance.CheckGuess(cardValue, cardType);
      GameManager.Instance.AddToNotebook(cardValue,
cardType);
      GameManager.Instance.choicePanel.SetActive(false);
   }
}
```

```
      // Time description
      if (time == "Ніч") lines.Add("Небо здавалося
бездонним, а кожен звук тонув у тиші, яка ховалася між
будинками.");
      else if (time == "Ранок") lines.Add("Світ ще не
визначився, чи настав день — усе навколо затягнуте
розмитими контурами й надто спокійним повітрям.");
      else if (time == "День") lines.Add("Довкола було
людно, але саме в цьому — найбільша іронія. Багато очей
— і жодного свідка.");
      else if (time == "Вечір") lines.Add("Тіні ставали
довшими, а подих вітру ніс з собою неспокій. Можливо,
хтось не встиг дістатись додому.");

      // Location description
      if (location == "Підвал") lines.Add("Простір без вікон.
Жоден звук тут не зникає — він лише довго тліє в
кутках.");
      else if (location == "Кухня") lines.Add("Тут щось
готували — можливо вечерю, можливо план. Поверхня
столу розповідала більше, ніж слова.");
      else if (location == "Гараж") lines.Add("Запах мастила і
металу. Інструменти лежали не там, де мали б. Але хтось
не став їх класти назад.");
      else if (location == "Вулиця") lines.Add("Люди
проходили повз, хтось навіть зупинявся. Але більшість
лише пришвидшувала крок — ніби щось відчували.");

      // Killer hint
      if (killer == "Поліція") lines.Add("Ті, хто мав би
ставити запитання, самі уникали відповідей. Слова йшли за
протоколом, але очі — ні.");
      else if (killer == "Дружина") lines.Add("Вона мовчала.
Надто спокійно. Мовчання буває голоснішим за істерики
— особливо, якщо в ньому немає здивування.");
      else if (killer == "Сусід") lines.Add("Він усе знав надто
швидко. І кожна деталь, яку він описував, звучала... як
репетиція.");
      else if (killer == "Друг") lines.Add("Його голос зривався
на жарт, коли мова йшла про серйозне. Здавалося, він
уникає не питання, а відповідальності.");
      else if (killer == "Брат") lines.Add("У родинних
стосунках завжди є глибина. Але часом — і тіні, які
відбиваються лише за певного освітлення.");

      // Weapon hint
      if (weapon == "Пістолет") lines.Add("Шум, який усі
чули, але ніхто не визнав. Слід був занадто чітким, ніби
зроблений з досвідом.");
      else if (weapon == "Ніж") lines.Add("Не один удар, не
один імпульс. Це було щось більше — щось особисте й
емоційне.");
      else if (weapon == "Камінь") lines.Add("Сліди не
вказували на намір — швидше на можливість. Те, що було
під рукою, стало вироком.");
      else if (weapon == "Ручка") lines.Add("Предмет
звичний. Його не помічають — поки не стане останнім, що
ви побачите.");

      return string.Join("\n", lines);
   }
}
```