МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-08

Дипломний проєкт

здобувача освіти денної форми навчання
КГ.08.05.000.ДП

ГРАБОВСЬКОЇ
ДАРІЇ СЕРГІЇВНИ

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «*Комп'ютерна інженерія*»**

Освітньо-професійна програма: «**Комп'ютерна графіка і Web-дизайн»**

Група: **4КГ-08**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity

Проєктний матеріал складається з пояснювальної записки на 83 сторінках та графічного (презентаційного) матеріалу на 16 аркушах (слайдах)

Дипломник _____ (Грабовська Д.С.)

Керівник _____ (Скорняков В.С.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «24» червня 2025 р. Протокол ЕК № 4

Оцінка ЕК 5 (відмінно) / 90 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна графіка і Web-дизайн»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Грабовській Дарії Сергіївні*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity*

затверджена наказом по коледжу від "14" листопада 2024р. № 246

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту *Використання програмного рушія Unity; Використання мови програмування C# та основних бібліотек рушія Unity; Реалізація основних ігрових механік гри «Сапер»; Реалізація системи управління; Реалізація декількох рівнів для гри; Реалізація модифікацій рівнів; Реалізація ігрового інтерфейсу гри*

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

Аналіз аналогів на ігровому ринку; Програмне забезпечення для розробки гри; Концепція розробляємої гри; Проектування основних елементів гри; Проектування модифікацій рівнів; Реалізація основних елементів гри; Реалізація генерації рівнів; Реалізація модифікацій рівнів; Реалізація інтерфейсу; Тестування розробленої

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Аналоги на ігровому ринку; Програмне забезпечення для роботи; Концепція розробленої гри; Структура основних елементів гри; Структура модифікацій рівнів гри; Реалізація модуля генерації рівнів гри; Реалізація графіки для модифікацій рівнів; Тестування та скріншоти розробленої гри

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Скорняков В.С.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 02.06.2025

Керівник *Скорняков В.С.* _____
(підпис)

Завдання прийняв до виконання *Грабовська Д.С.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	*14.05.2025*	виконано
2	*Аналіз аналогів на ігровому ринку*	*16.05.2025*	виконано
3	*Вибір та налаштування засобів розробки*	*17.05.2025*	виконано
4	*Розробка концепції ігрового процесу*	*20.05.2025*	виконано
5	*Проектування основних елементів гри*	*22.05.2025*	виконано
6	*Проектування модифікацій рівнів гри*	*28.05.2025*	виконано
7	*Реалізація основних елементів та модифікацій*	*01.06.2025*	виконано
8	*Реалізація ігрового інтерфейсу*	*03.06.2025*	виконано
9	*Відлагодження елементів гри*	*06.06.2025*	виконано
10	*Тестування працездатності елементів гри*	*10.06.2025*	виконано
11	*Виправлення виявлених помилок*	*11.06.2025*	виконано
12	*Аналіз результатів, підготовка слайдів презентації*	*12.06.2025*	виконано
13	*Економічні розрахунки та питання з охорони праці*	*13.06.2025*	виконано
14	*Підготовка графічної частини проекту*	*14.06.2025*	виконано
15	*Підготовка проекту до захисту та тестування гри*	*16.06.2025*	виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 08. 05 000. 00 ДП	Дипломний проєкт		
А4			КГ 08. 05 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КГ 08. 05 000. 00 ДП		
Розробив		Грабовська Д.С.		16.06	Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity	Літ.	Аркуш 4 / Аркушів 83
Перевірив		Скорняков В.С.				Н Д П	
Н. контр.		Петрашова В.І.		16.06.25		ВСП "ОТФК ОНТУ" гр.4КГ-08	
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

Сьогодні комп’ютерні ігри перестали бути розвагою лише для дітей, але й є засобами для емоціональної та фізичної розрядки, коли людина відпочиває від роботи, або способом провести час самому чи в компанії друзів та близьких людей. Створення ігор стало справжнім сектором світового IT-ринку, який продовжує переживати свій бурхливий розвиток.

Сучасні комп’ютерні ігри включають до свого складу, як великі дорогі у фінансовому плані та складні за структурою проекти, так і досить прості ігри, сесія гри в які не займає дуже багато часу. Такі ігри досить прості у своїй реалізації та не вибагливі до візуального або наративного наповнення. Їх створення не потребує великої кількості спеціалістів, складних технологій та рекламних компаній.

Саме розробка такої гри є метою мого дипломного проекту: «Розробка гри «Сапер» з модифікацією рівнів на ігровому рушії Unity». Потрібно розробити просту та натомість реіграбельний проект, за яким можна буде провести деякий час, отримуючи різний ігровий досвід. Гра має генерувати унікальне розміщення для мін, надавати можливість створювати у ігровому процесі модифікації, які б давали змогу змінювати привичний ігровий процес гри Сапер.

Для розробки цієї гри буде використано сучасний ігровий програмний рушій Unity. В свою чергу, використання такого ігрового рушія потребує використання сучасної та потужної об’єктно-орієнтованної мови програмування C#. Також, реалізація гри потребує створення графічного матеріалу для спрайтів ігрових елементів, об’єктів, інтерфейсу гри.

Завданням розробки є створення ігрового застосунку для персонального комп’ютера з можливістю подальшого розширення ігрового процесу та ігрових механік, створення основи для майбутнього більш складного та унікального проекту. Використання сучасного ігрового програмного рушія Unity дає змогу створювати проекти, які можна портувати з однієї платформи виконання на іншу, що дає потенціал для подальшого розвитку проеку.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз аналогів на ігровому ринку

1.1.1 Загальні дані про ігровий жанр Сапер

Темою даної дипломної роботи є «Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity». Для виконання завдання дипломного проектування необхідно виконати визначений порядок дій як розробка концепції ігрового процесу, проектування елементів гри та їх реалізація, процес тестування. Для початку роботи потрібно визначитись з тим, що з себе представляють ігри жанру «Сапер», які основні правила таких ігор, зовнішній вигляд та інше.

Ігровий жанр сапер можна відвести до піджанру головоломок, та він пішов від однойменної гри. Суть ігрового процесу була дуже простою – потрібно було знайти на ігровому полі всі міни. Ігрове поле представляло собою поділений на клітинки прямокутник з різною довжиною сторін. Кількість мін залежала від розміру ігрового поля. Розміщення мін відрізнялось від однієї ігрової сесії до іншої. Гравець мав натискати на клітинки чим розкривати їх. Якщо в клітинці була міна – гравець програвав. В іншому разі, в клітинці з’являлась цифра, яка вказувала на кількість мін навколо цієї клітинки. Наприклад, якщо після розкриття клітинки, в ній відображалась цифра 4, то це значило, що навколо цієї клітинки 4 міни. Для успішного закінчення ігрової сесії потрібно було розставити на клітинках, які гравець вважає замінованими, прапорці та розкрити всі незаміновані клітинки.

Класичне виконання цієї гри було вперше введено до стандартного пакету ігор для операційної системи Microsoft Windows 3.1 та додавалась до кожного пакету операційних систем – останньою операційною системою, яка мала цю гру стала Windows 7. Дизайн був дуже простим та мінімалістичним, ігровий процес розділявся на рівні складності Початківець, Нормальний, Професіонал. Ці рівні відрізнялись розмірами та кількістю мін на полі. Життів завжди давалось тільки одна. Не дивлячись на вік, ця гра має свою фанбазу, а також турніри та чемпіонати по швидкості розмінування ігрових полів. На рисунку 1.1 зображено скріншот класичної гри Сапер.

Рисунок 1.1. Скріншот класичної гри Сапер для операційній системі Windows XP

Для визначення основних положень для концепції гри потрібно розглянути інші аналоги цієї гри. Знайти аналогічні ігри можна на профільному сайті, який присвячений грі Сапер. Проектів, які базовані на жанрі сапер дуже багато, вони виконуються на різних операційних системах та пристроях та навіть на калькуляторах! Для виконання аналізу було обрано два проекти.

1.1.2 Аналіз гри Hex Minesweeper

Гра Hex Minesweeper є переосмисленням оригінальної ідеї гри Сапер у напрямку зміни правил форм поля та ігрових клітинок. Якщо в оригіналі використовуються клітинки, то у цій грі розробник заміним їх на гексагони, що вказано у найменуванні проекту. З точки зору інших ігрових правил, все залишилось таким самим, як було. Так само потрібно обирати комірку, активувати її та дізнаватись чи скривалась там міна, або побачити значення кількості мін навколо комірки. Зміна форми комірки привела до деякого суб'єктивного полегшення ігрового процесу, оскільки кількість мін які можуть знаходитись

навколо комірки зменшилась з 8 до 6, що напряму впливає на те, як розраховувати присутність мін на ігровому полі. Наприклад, якщо в лінії комірок класичного саперу буде комірка з цифрою 3 та однією відомою міною, то доведеться обирати дві клітинки між трьома клітинками. У аналогічній ситуації гри Hex Minesweeper, такої проблеми не буде, оскільки у будь-якому випадку будуть не відкритими дві клітинки з мінами, що полегшує принцип вирішення головоломки. На рисунку 1.2 можна побачити скріншот ігрового процесу гри Hex Minesweeper.

Рисунок 1.2. Скріншот ігрового процесу гри Hex Minesweeper

Ця гра може виконуватись як на персональному комп’ютері, так і в браузері. Окрім того можна відмітити відсутність можливості проставляти знаки запитання. В іншому, окрім форми комірок, гра повністю повторює оригінал.

1.1.3 Аналіз гри Minesweeper Mania

Ця гра є власною розробкою однієї людини Рона Хеуса, який переосмислив гру Сапер повністю, незмінною залишилась лише основна ідея ігрового процесу – гравець повинен знайти всі міни на ігровому полі. З нововведень розробник додав до гри окрім мін пастки, які можуть впливати на ігровий процес негативним чином, збільшуючи кількість мін. Окрім того тепер у гравця є показник здоров’я,

який втрачається при активації міни або пастки. Натомість для гравця також з’явились різні допоміжні інструменти, наприклад інструменти для розмінування пасток, сканер, для пошуку мін у клітинках, медична аптечка для підвищення показника здоров’я, та інші бонуси, які купляються у «торговця» за монети, які заробляються під час гри. Разом з тим, кількість пасток, мін їх типи та розміри змінюються під час проходження рівнів, чим далі, тім складніше. На рисунку 1.3 можна побачити скріншот ігрового процесу гри Minesweeper Mania.

Рисунок 1.3. Скріншот ігрового процесу гри Minesweeper Mania

Додавши до гри Сапер досить прості модифікатори, як бонуси та пастки, розробник зміг перетворити просту гру-головоломку у справжній досить складний проект, який займає гравця не на одну годину. Не дивлячись на те, що проект був створений на початку 2000-х років, він досі має свою аудиторію, яка продовжує грати в неї. Даний проект можна випробувати лише на персональному комп’ютері.

1.1.4 Результати аналізу

В цілому було проаналізовано більше проектів у жанрі сапер, але наочний аналіз додано лише дві. В результаті аналізу було визначено, що більшість розробників ділять свої проекти на два типи модифікацій. Такі, які змінюють ігрове поле, та ті, що залишають ігрове поле незмінним, але працюють над додаванням бонусів, які вносять зміни у ігровий процес, модифікуючи ігрове поле шляхом відкриття клітинок, або додаванням додаткових видів активних клітинок.

З точки зору розробляємого проекту важливо саме визначитись із тим, як виконувати модифікацію рівнів гри, оскільки це є важливою частиною теми роботи. Визначено два шляхи таких змін, а саме зміна розмірів ігрового поля під час обрання рівня складності та додавання бонусів для гравця, який зможе використовувати їх на свій розсуд. Таким чином, при використанні таких бонусів, рівні не будуть змінюватись функціонально, але вони будуть змінюватись з контентної точки зору, додаючи для гравця додаткові прийоми гри та візуальні зміни у рівень.

1.2 Програмне забезпечення для розробки гри

1.2.1 Огляд ігрового програмного рушія Unity

У темі дипломного проекту вказано використання конкретного ігрового програмного рушія. Unity – це багатофункціональна платформа для розробки ігор з підтримкою багатьох платформ, створена компанією Unity Technologies. Цей інструмент дозволяє створювати як 2D, так і 3D проєкти, а також інтерактивні додатки, включаючи рішення для віртуальної та доповненої реальності. Платформа підтримує понад два десятки середовищ, серед яких персональні комп’ютери, мобільні ОС, ігрові приставки та веббраузери. Unity вирізняється легкістю у вивченні, широким спектром навчальних матеріалів, а також активною спільнотою розробників. Цей рушій активно використовують як незалежні розробники, так і великі компанії у сфері геймдеву, архітектурної візуалізації, освіти, кіновиробництва й моделювання. Серед відомих ігор, створених на Unity, варто відзначити Hollow Knight, Ori and the Blind Forest та Cuphead.

Серед основних особливостей рушія варто виділити об’єктно-компонентну структуру, де всі елементи сцени представлені у вигляді GameObject – базових сутностей, до яких можна прикріплювати різні компоненти, такі як скрипти, фізичні властивості, колізії, візуальні елементи тощо. Така архітектура дозволяє гнучко формувати функціональність через поєднання окремих частин. Рушій використовує мову програмування C# та надає розробнику доступ до великої кількості функцій і бібліотек через добре задокументований API.

Вибір Unity як основи для реалізації проєкту був обумовлений кількома факторами. По-перше, платформа має багату базу документації та навчальних ресурсів як від розробників, так і від спільноти, що значно полегшує пошук рішень під час роботи. По-друге, аналіз прикладів ігор показує, що різні реалізації ігор жанру сапер мають мінімалістичний дизайн, просте управління, через це використовувати більш складні ігрові програмні рушії немає сенсу. По-третє, існує тренд на виведення такого виду ігор у веббраузери, або на платформу Android. Завдяки використанню ігрового програмного рушія Unity, у разі потреби, виконати портування буде дуже просто.

1.2.2 Огляд іншого програмного забезпечення для розробки гри

Для реалізації ігрового проекту необхідно також підібрати низку додаткових інструментів. Оскільки створення гри неможливе без програмування, слід визначитися з відповідним середовищем для написання коду. Вибір мови програмування обумовлюється обраним рушієм – у даному випадку це C#. Отже, потрібно використовувати IDE, яке забезпечує повну сумісність з цією мовою. Згідно з рекомендаціями документації рушія, доцільно застосовувати Microsoft Visual Studio – потужне інтегроване середовище розробки, розроблене компанією Microsoft. Це середовище підтримує широкий набір мов, зокрема C#, C++, Visual Basic, Python, JavaScript та TypeScript, і тісно взаємодіє з .NET-інфраструктурою. До основних переваг Visual Studio належать підсвічування синтаксису, інтелектуальні підказки коду, а також можливість розширення функціоналу за рахунок численних плагінів.

Окрім написання коду, важливою складовою процесу є створення

графічного контенту. У цьому проєкті передбачено використання 2D-візуалізації, що потребує застосування відповідного графічного редактора. Враховуючи функціональні можливості, зручність роботи та наявний досвід, було обрано Adobe Photoshop – професійний інструмент для роботи з растровими зображеннями, створений компанією Adobe Inc. Ця програма широко використовується дизайнерами, ілюстраторами, художниками та іншими фахівцями для обробки зображень, створення графіки, оформлення інтерфейсів, а також підготовки матеріалів до друку чи анімації. У контексті розробки гри будуть задіяні такі можливості Photoshop, як редагування зображень, корекція кольорів, робота з шарами та інструменти ретуші. В цілому, розробляємий проект не має сильної графічної направленості, тому використання графічного редактору буде мінімальним.

1.3 Концепція розробляємої гри

1.3.1 Проробка загальних елементів гри

На початковому етапі створення гри розробники зазвичай починають із підготовки концепт-документу. Це попередній документ, що виконує схожу роль із дизайн-документом, однак не є остаточним – його зміст може змінюватися і адаптуватися до реальних умов виробництва. Концепт-документ фіксує базову ідею проекту, визначає основні напрямки, сюжетну основу, ключові механіки та інші важливі характеристики гри. Він виступає як стратегічний орієнтир на ранній стадії й поступово уточнюється в процесі розробки.

Зазвичай у межах такого документа описуються наступні компоненти:

- Загальна концепція: Формується опис ігрового простору, атмосфери, сюжетної лінії та цільової установки гри;
- Геймплейні механіки: Окреслюються правила взаємодії з ігровим світом, управління персонажем, фізика середовища, розвиток персонажа, бойова система та інші елементи геймплею;
- Сценарій і герої: Визначаються сюжетні події, основні й другорядні персонажі, їхні риси, мотиви та взаємовідносини;

- Візуальна й звукова складова: Надається загальне уявлення про стиль графіки, анімацію, кольорову палітру та аудіосупровід;
- Світобудова: Описуються додаткові деталі світу гри — його історія, соціальна структура, технології, традиції та культурний контекст;
- Технічна база: Уточнюються технічні параметри — цільові платформи, ігровий рушій, необхідні інструменти, мови програмування та середовища розробки.

Концепт-документ слугує для узгодження бачення майбутнього продукту всередині команди. Він допомагає структурувати процес розробки, планувати етапи роботи та своєчасно виявляти проблемні моменти, які потребують уточнення чи адаптації.

Для формування концепту розробляємого проекту було проведено аналіз інших ігор, які виконані у ігровому жанрі сапер. Це дало змогу отримати уявлення на загальні елементи таких ігор, дизайн, управління та інші підходи до реалізації цих проектів. Перед тим як проробляти концепцію модифікації рівнів, потрібно розробити концепцію загальних елементів гри:

- Загальна концепція гри повторює всі основні правила класичного саперу. Мінімалістичний дизайн, модифікації для рівнів, які вносять зміни у поле та ігровий процес;
- У геймплейному плані майже всі правила гри залишаються класичними. Гравець має виконувати розмінування ігрового поля шляхом розкриття клітинок. Якщо знайдено міну – гравець втрачає життя. Якщо міни нема, то клітинка демонструє цифру, яка вказує на кількість мін навколо. Розміщення мін випадкове;
- Гра не матиме сценарної складової, або сюжету, гравець буде запускати нові ігрові сесії, унікальність яких зумовлена випадковою генерацією ігрових полів;
- З візуальною точки зору, гра буде мати мінімалістичний дизайн, без анімацій або ефектів, що продовжує традиції класичної гри. Інтерфейс буде демонструвати кількість мін на полі, кількість життів та бонусів;

- Цільовою платформою є персональний комп’ютер. Гра реалізовуватиметься на ігровому програмному рушії Unity, що у майбутньому дасть змогу портувати гру на інші платформи.

Завдяки отриманим положенням вже можна уявити, якою має бути гра, та яким чином підходити до етапу проектування елементів гри.

1.3.2 Проробка концепції модифікації рівнів

В темі даного дипломного проекту вказано «Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity», де чітко визначено необхідність не тільки розробити гру, але й додати до неї модифікації ігрових рівнів. Вище, після аналізу інших проектів жанру сапер, було вирішено, що модифікаціями рівнів будуть бонуси, які мають впливати на стан клітинок на ігровому полі. Окрім того, потрібно реалізувати можливість грати на різних розмірах полів.

Концепція модифікацій рівнів потребує більш детально описати те, яким чином ці модифікації будуть працювати, та який їх загальний принцип та ціль. Отже, такі модифікації мають наступні особливості:

- Модифікації використовуються під час ігрових сесій та мають обмежену кількість викликів;
- Модифікації можуть розкривати горизонталі та вертикалі ігрового поля, розкриваючи клітинки та демонструючи їх складову, без впливу на життя гравця;
- Модифікації не можуть бути отримані під час сесії проходження рівня;

Окрім модифікацій рівнів, у грі також змінено механіку життів гравця. У класичній грі, при розкритті міни, гравець програвав партії та мусив почати знову. У реалізовуємому проекті, має бути додана функція «життів» для гравця, щоби він міг отримувати додаткові шанси.

Додаткові життя на модифікації рівнів мають стилізовані назви. Так життя гравця мають назву Щупи – як саперний щуп. Модифікації рівнів мають назву Система «Горизонталь», вона відкриває всі горизонтальні лінії гри, та Система «Вертикаль», вона розкриває всі вертикальні лінії.

1.4 Проектування основних елементів гри

1.4.1 Загальна будова проекту

Після формування ігрового концепту необхідно виконати етап проектування гри. Цей етап є важливим у розробці будь-якого проекту, оскільки створює структуру майбутньої гри за якою розробник може надалі створювати елементи гри. Проектування не створює обов'язкових потреб до реалізації, але надає можливість проробити всі частини гри до того моменту, коли вже буде виконуватись розробка, що дає уникнути ситуацій, коли під час створення того чи іншого модулю гри виявляються проблем із зв'язком модулів, або структурна помилка у проекті. Завдання цього етапу роботи полягає у створені загального контуру розробки, який у подальшому може бути скорегований.

Для ефективного проектування потрібно розуміти структуру розробляємих проектів та того, як ці проекті функціонують у ігровому програмному рушії Unity. Для вдалої реалізації проекту потрібно забезпечити реалізацію чотирьох основних складових гри, які можна побачити на рисунку 1.4.

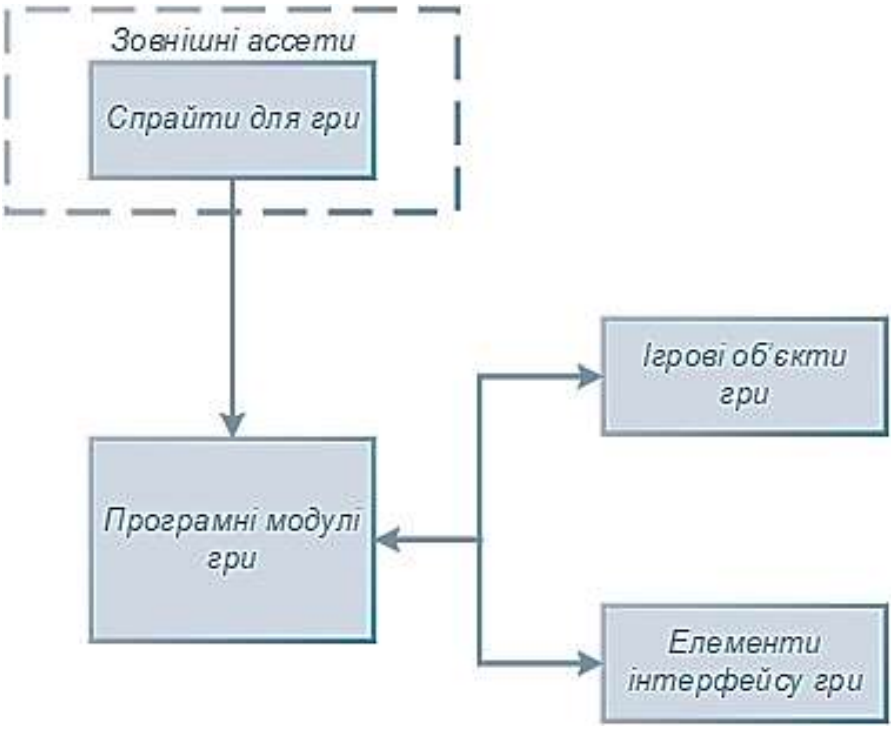


Рисунок 1.4. Структура розробляємого проекту

Для розробляємого проекту потрібно створити деякі зовнішні ассети, які будуть використовуватись грою під час свого виконання. Ассетами у ігровому

програмному рушії Unity називають файли, які не знаходяться безпосередньо на сцені та зберігаються у відповідних папках проекту. Під час побудови проекту ці файли розміщуються у спеціальній бібліотеці, а під час розробки, розробник може їх редагувати та має прямий доступ до налаштування того, яким чином виконується взаємодія з ассетами.

Ігровими об’єктами гри можна назвати все, що безпосередньо знаходиться на сцені та що виконує ту, чи іншу функцію на сцені. Такі об’єкти можуть бути як видимими для гравця та розробника, так і невидимими. Ігрові об’єкти є основними діючими елементами будь-якої гри у ігровому програмному рушії Unity. Елементи інтерфейсу це кнопки, написи, панелі, та багато іншого, що існує окремо від сцени знаходячись нібито на екрані користувача.

Програмні модулі гри є найголовнішою частиною розробляємого проекту, оскільки саме ці модулі забезпечують унікальну поведінку ігрових об’єктів на сцені, елементів інтерфейсу, а також взаємодію між деякими спрайтами гри із ассетів. Програмні модулі скриптують ігровий процес та є невід’ємною частиною розробки гри на ігровому програмному рушії Unity.

1.4.2 Детальний огляд спроектованих блоків розробляємої гри

Більш детальна проробка змісту структурних елементів розробляємого проекту дозволяє створити активні сутності гри. На рисунку 1.5 можна побачити складові блоку Ігрових об’єктів гри.

Рисунок 1.5. Перелік видів ігрових об’єктів для розробляємої гри

Як видно з рисунку, для реалізації гри потрібно створити два види ігрових об’єктів. Першим є Ігрові об’єкти клітинок ігрового поля. Ці об’єкти будуть відповідати за матеріальну репрезентацію клітинок з якими гравець буде взаємодіяти. Ці об’єкти будуть мати свої параметри клітинок, їх зміст та інші службові дані. Ігрові об’єкти написів не є активними для гравця, а слугують лише як ілюстратор змісту клітинок ігрового поля, це створює нерозривний зв’язок між цими двома об’єктами. Кожна клітинка повинна мати свій напис, з яким вона буде взаємодіяти.

На рисунку 1.6 можна побачити більше детальний зміст блоку Елементів інтерфейсу гри. Видно, що всього такі елементи діляться на три види. Елементи параметрів гравця відповідають за репрезентацію поточних параметрів гравця серед котрих є кількість життів гравця або кількість мін, які потрібно знайти для закінчення проходження рівня.

Рисунок 1.6. Перелік елементів інтерфейсу для розробляємої гри

Елементи модифікаторів-бонусів показують елементи, які дадуть змогу взаємодіяти з модифікаторами рівнів. У концепті вказано, що серед таких модифікаторів ігрових рівнів у ігровому процесі буде доступно два види, тому це будуть дві кнопки, які активують або деактивують використання бонусів.

Елементи ігрових повідомлень будуть забезпечувати виведення ігрових або службових повідомлень. Такі елементи будуть реалізовані написами. Демонструвати вони будуть повідомлення про успішне закінчення рівня, або програш.

Програмні модулі, які потрібно реалізувати зображені на рисунку 1.7. Програмні модулі повинні забезпечувати весь ігровий функціонал, тому для їх проробки потрібно розкласти ігровий процес на складові та поділити на складові, які потрібно реалізувати. В результаті такої роботи можна побачити, що блок програмних модулів складається з чотирьох елементів.

Модуль обробки натискань на клітинки відіграє роль менеджеру та обробника натискань. У ігровому програмному рушії Unity вже є свій обробник, але він реалізує лише натискання на ліву кнопку миші. Ігровий процес Саперу складається не тільки з розкриття клітинок, але й з розташування «прапорців» які дають розуміти грі, що гравець вважаю обрану клітинку замінованою. Особливістю через яку потрібно створювати окремий обробник лежить у тому, що вбудований не розрізняє натискання правою кнопки миші. Тому у розробляємому обробнику такий функціонал має бути реалізованим і він має «розуміти», коли гравець натиснув на ліву чи праву кнопку миші.

Безпосередньо реалізацією процесів при натисканні на праву кнопку миші має Модуль обробки натискання правої кнопки миші. Він не виконується сам по собі та викликається з Модулю обробки натискань на клітинки, який визначивши тип кнопки, звертається до нього.

Модуль логіки роботи клітинки змістовує в собі функціонал, який повинен реалізовуватись при натисканні різних видів кнопок миші. Окрім того цей модуль повинен буде відповідати за поведінку кнопки при результатах цих натискань.

Рисунок 1.7. Перелік програмних модулів для реалізації у розробляємій грі

Наприклад, якщо при натисканні на клітинку лівою кнопкою миші було знайдена міна, то клітинка повинна змінити свій зовнішній вигляд а також викликати зміни у параметрах гравцях та інтерфейсі рівня.

Модуль генератора ігрового поля є одним із найважливіших модулів у розробляємій грі. По суті всі ігрові механіки мають виконуватись через цей модуль, оскільки він відповідає за заповнення клітинок ігрового поля інформацією про те, чи є в них міни. Саме цей модуль має реалізовувати випадковість розміщення мін та відповідати за збереження інформації про ігрове поле та окремі його клітинки.

1.5 Проектування модифікацій рівнів

Особливу увагу потрібно приділити модифікаціям рівнів. Їх, умовно, можна поділити на два види, згідно до розробленої концепції гри. Перший вид модифікації – зміна ігрового поля під час вибору рівня. Гравець обирає розмір поля перед сесією, а гра завантажує обране поле.

Другий вид модифікацій рівня – це такі модифікації які змінюють поведінку клітинок ігрового поля, що є складовою рівнів гри. Зміна поведінки клітинок, як було розроблено у концепції, полягає у їх розкритті без впливу на гравця, проте з відміткою для самих клітинок та гри процесу розкриття клітинок ігрового поля. Варто також пам’ятати, що розробкою концепції та проектуванням гри було також встановлено, що до таких модифікаторів відноситься функціонал додаткових життів гравця. Вони стилізовані, як і модифікатори рівнів, та дають можливість гравцю отримати додатковий ігровий досвід. Окрім того, щупи дають змогу уникнути дратівливих ігрових ситуацій, коли гравець залишився з однією нерозкритою міною та двома нерозкритими клітинками та з жодними підказками. В такому випадку гравцю залишається лише здогадуватись, де знаходиться міна що створює негативний вплив на враження гравця. Якщо на ігрових полях невеликого розміру такі ситуації бувають вкрай не часто, у більш складних рівнях шанс отримати такий завершаючий етап досить високі.

Для розуміння того, як реалізовувати модифікації рівнів, потрібно виконати їх проектування. Важливо бачити їх роботу у ігровому моменті. На рисунку 1.8 можна побачити схему взаємодії модифікаторів ігрових рівнів з іншими модулями у ігровому процесі.

Як можна побачити, так чи інакше всі модифікатори ігрового рівня взаємодіють із Модулем генератора ігрового рівня. При обрані розміру ігрового поля та завантаженні рівня, розмір поля передається у Модуль генерації ігрового поля. Він відтворює ігрове поле заданого розміру, та передає інформацію про клітинку у відповідні Модулі логіки роботи клітинки, для кожного ігрового об’єкта – клітинки на полі.

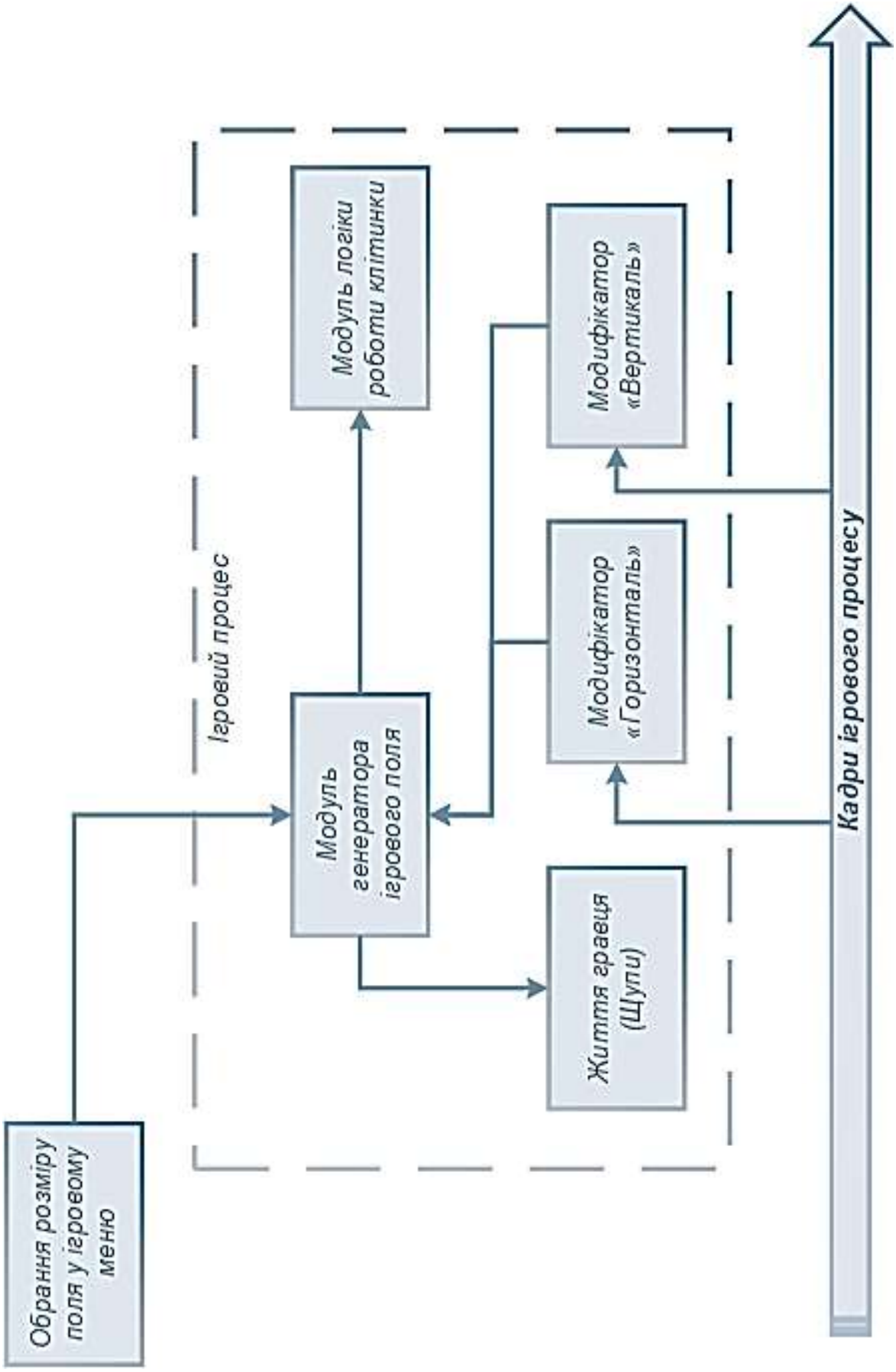


Рисунок 1.8. Схема взаємодії модифікаторів рівнів з іншими модулями у ігровому процесі розробляємої гри

Бонус «Горизонталь», при активації дає можливість обрати гравцю горизонтальну лінію клітинок ігрового поля, яку він хотів би розкрити безпечно для себе. Потрібно акцентувати увагу на тому, що цей бонус-модифікатор має змінювати статус кожної клітинки таким чином, щоби гра вважала, що клітинка вже розкрита гравцем, при цьому активація мін не повинна виконуватись. В той же час при знаходженні міни, гра повинна обов'язково відняти її від кількості мін, які потрібно розкрити. Цей модифікатор може бути активований та деактивований гравцем у моменті ігрового процесу. Те є саме стосується і бонусу «Вертикаль», який робить те ж саме та функціонує так само, як «Горизонталь», але відкриває клітинки обраної вертикалі ігрового поля. Життя гравця змінюється при виконанні деяких умов.

1.6 Реалізація основних елементів гри

1.6.1 Початкова робота з проектом та його налаштування

Першим кроком у процесі створення гри є ініціалізація нового проекту. У різних ігрових рушіях цей процес реалізований по-різному: в одних системах проекти формуються на основі готових шаблонів, в інших – робота стартує безпосередньо в середовищі рушія. У випадку з Unity для цього передбачено окремий інструмент, який називається Unity Hub, що виконує функції запуску, управління проектами, налаштування конфігурацій та параметрів, а також встановлення нових версій ігрового програмного рушія.

Можливість інсталяції декількох версій рушія є важливою особливістю, оскільки дозволяє підтримувати різні проекти, які можуть бути створені на різних ревізіях платформи розробки. Це особливо актуально для команд, що працюють над кількома іграми одночасно або здійснюють довготривалу технічну підтримку своїх продуктів. Ще одним прикладом використання декількох версій є перехід проекту гри зі старої версії ігрового рушія на нову. Unity може виконати перехід самостійно, але в процесі цього переходу можуть з'являтись помилки.

Під час створення нового проекту можна задати додаткові налаштування, зокрема параметри інтеграції з хмарними сервісами Unity та систему контролю

версій. У рамках цього проекту ці можливості залишаться деактивованими, оскільки розробка здійснюватиметься локально на одному пристрої однією людиною.

Для створення нового проекту буде використано Unity Hub, де обирається назва, розміщення на локальному диску, версія рушія та тип шаблону (2D, 3D, VR тощо). Шаблони використовуються для того щоби полегшити роботу розробника, оскільки пусті проекти не мають налаштувань управління, стартових ігрових об’єктів, налаштування сітки простору розробки, та багатьох інших функцій. Використання шаблонів дає змогу автоматизувати цей процес. Наприклад використання шаблону гри-гонки відразу створює трек, машину її фізику та початковий код для управління машиною. Додаткові шаблони для рушія можна знайти на офіційному сайті, або завантажити безпосередньо у Unity Hub. На рисунку 1.9 представлено процес створення проекту у ігровому програмному рушії Unity.

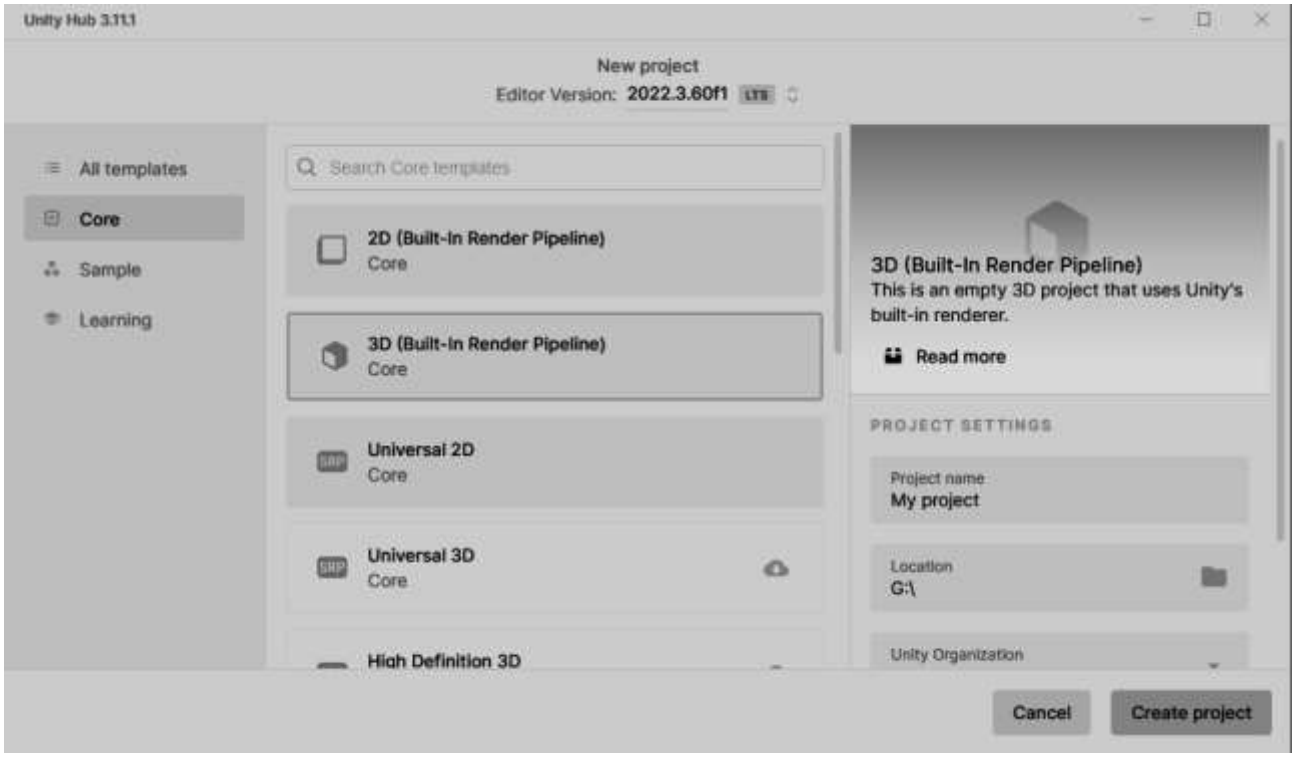


Рисунок 1.9. Вікно створення нового проекту у ігровому рушії Unity

Після заповнення даних проекту та натискання кнопки Create Project, Unity Hub автоматично ініціалізує створення проекту, згенерує необхідну структуру каталогів, завантажить відповідні пакети та додасть ресурси згідно з обраним

шаблоном.

Варто звернути увагу, що типовий проект у Unity базується на двох основних директоріях. Перша – Packages, що містить службові дані рушія: бібліотеки, технічні файли, а також тимчасові елементи, які створюються під час редагування сцени чи її компіляції. Друга директорія називається Assets і вона є основним простором роботи для розробника. Саме сюди додаються всі ресурси проекту: графічні файли, текстури, шейдери, анімації, звукові елементи, скрипти та інші матеріали, необхідні для розробки гри. Створення проекту займає деякий час в результаті якого завантажується сама середа розробки. Вона складається з багатьох вікон та панелей, в яких використовується розробка гри, редагування сцени, робота з компонентами. Робочий інтерфейс Unity, у якому здійснюється безпосереднє створення гри, показаний на рисунку 1.10.

Рисунок 1.10. Робочий інтерфейсі ігрового програмного рушія Unity

Оскільки при створені проекту було обрано шаблон для 2D-ігор, рушій самостійно встановив параметри відображення сітки, налаштування камери, світла та системи управління.

Візуальна складова відіграє значну роль у процесі створення гри, оскільки саме вона формує унікальний стиль і надає проекту візуальну особливість. Проте на стартових етапах розробки дозволяється використовувати базові 3D-примітиви

або тимчасові 2D-зображення заглушки (спрайти), які виконують функцію умовного візуального наповнення. Такий підхід частіше за все використовують для налагодження і перевірки працездатності логіки гри, оскільки він не займає багато часу. Окрім цього, попереднє створення графічного контенту, навіть у спрощеному вигляді, допомагає сформувати загальне уявлення про стилістику проекту. Це дозволяє розробникам на ранньому етапі оцінити, як виглядатиме фінальний продукт і як поєднуються окремі компоненти в єдиному ігровому середовищі.

У випадку розробляємого продукту візуальна складова не є головною частиною ігрового процесу. Розроблена концепція визначає розробляєму гру як нащадок класичної гри Сапер. Опираючись на це потрібно реалізувати мінімальну графічну складову, а саме варіанту статусу клітинок – пуста нерозкрита клітинка, клітинка з прапорцем, клітинка з міною на яку натиснули. Окрім того потрібно реалізувати спрайти для елементів інтерфейсу, що демонстрували б кількість життів, що має бути стилізовано під саперний щуп а також кількість нерозкритих мін. На рисунку 1.11 можна побачити всі спрайти які були створені для розробляємої гри.

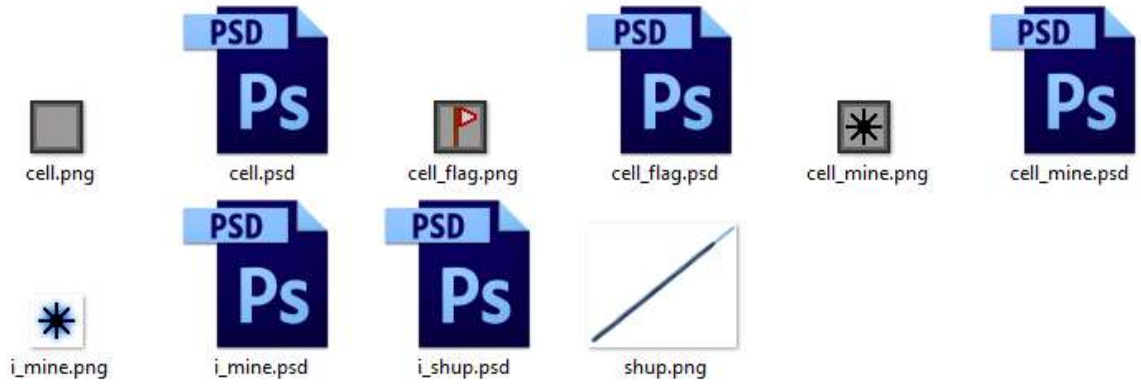


Рисунок 1.11. Скріншот спрайтів створених для розробляємої гри

На рисунку 1.11 можна також побачити файли з розширенням .psd – це файли Adobe Photoshop. Під час створення графічного матеріалу для гри дуже важливо залишати файли в яких ці матеріали створювались, оскільки бувають ситуації, коли потрібно змінити спрайт, або створити новий, який буде базуватись на старому спрайті. Це легше за все робити, коли є доступ до оригінальних слоїв

зображення та ефектів, ніж працювати з растровим варіантом матеріалу.

Після створення графічних матеріалів їх потрібно додати до проекту. Оскільки ці файли є службовими та будуть використовуватись ігровими об’єктами, потрібно додати ці матеріали у каталог Assets, де будуть розміщуватись всі ассети гри. Для спрайтів було створено відповідний каталог та імпортовано створені спрайти окрім їх файлів-проектів, як показано на рисунку 1.12. Вже в цьому каталозі можна виконати налаштування для цих файлів. Формально, у момент імпортування у проект, спрайти є зображеннями і для ігрового програмного рушія Unity вони стають спрайтами тільки після того, якщо обрати відповідний Texture Type у налаштуваннях. Імпортовані файли можуть бути як звичайними зображеннями, так і текстурами, спрайтми, картами нормалей та іншими видами текстур.

Рисунок 1.12. Каталог спрайтів у середі розробки Unity

Для подальшої реалізації гри потрібно створити ігрові об’єкти, з якими гравець буде взаємодіяти. Концепція розробляємої гри вказує на потрібність реалізувати класичний підхід до проекту, тому ігрове поле буде складатись з клітинок, на які гравець зможе натискати та викликати ти чи інші дії. В цьому моменті потрібно розкрити досить глибше одне з рішень, які вносять зміни у розроблений проект гри.

Виходячи зі спроектованих елементів гри, потрібно було створити окремо ігрові об’єкти для гри, а також окремо елементи інтерфейсу. Під час реалізації

цього підходу, в ході роботи та на етапі подальшої проробки коду, виявились проблеми використання ігрових об’єктів окремо від елементів інтерфейсу. Річ у тому, що ігрові об’єкти ігрові об’єкти знаходяться на сцені. Сцена – Scena – це уявне середовище, в якому знаходяться ігрові об’єкти зі своїми компонентами, свій інтерфейс, та скриптами, які для них реалізовані. Це середовище представлене файлом-сценою, яка складається з текстового представлення всієї інформації які міститься на сцені. Сцен у проектах Unity може бути безліч, але активних може бути лише одна.

Отже, ігрові об’єкти, які знаходяться на сцені «не вміють» з коробки обробляти натискання мишкою. Для реалізації цього функціоналу потрібно було б також написати код, який виконував би простріли променем з курсору миші на сцену. При цьому потрібно було б виконувати багато конвертацій, переводів з одної системи світових координат у систему координат локальну та багато іншого. Окрім того, потрібно було б зчитувати те, які об’єкти такий промінь перетинає, ідентифікувати їх. До того ж ігрові об’єкти у просторі сцен, складно піддаються розміщенню у ньому, оскільки не можуть бути прив’язані до горизонталей, вертикалей, або клітин простору. По цій причині потрібно буде вручну прописувати координати для кожної клітинки. З точки зору реалізації графіки, для реалізації такого підходу також потрібно було б створити окремо спрайти для всіх варіантів цифр, що збільшить роботу із кодом клітинок. Єдиною сильною стороною цього підходу є можливість генерувати особливі рівні, з довільними розмірами, які може вводити гравець у головному меню.

Натомість, є альтернативний варіант реалізації гри, набагато більш простий, а саме створення ігрового рівня через кнопки ігрового інтерфейсу. По суті потрібно розміщувати кнопки розміром з клітинку на інтерфейс гравця. Кнопки вже мають анімації натискання, код обробки натискання, компоненти кнопок а також, як нащадок, текст, який можна досить просто та швидко редагувати з коду. Окрім того, використання тексту дає змогу зменшити кількість графічного матеріалу, оскільки цифри можна відображати саме через написи на кнопках. Під час розміщення кнопок, вони створюються на інтерфейсі, тому можуть бути легко

прив’язані до країв екрану, можна легко реалізувати відступи від країв екрану та багато іншого. Також при переміщенні кнопок у інтерфейсі автоматично вмикається вирівнювання елементів по сусіднім елементам або іншим орієнтирам, що спрощує створення ігрових полів. З точки зору реалізації генерації рівнів, то через кнопки та інтерфейс також можна генерувати клітинки, але їх розміщення на екрані досить складно вирівнювати через код, оскільки зміни у роздільній здатності монітору можуть впливати на розміщення елементів на сцені.

Таким чином, після проведення експериментів з реалізацією клітинок першим та другим способом, було вирішено створювати ігрове поле через кнопки на ігровому інтерфейсі, що прискорить роботу та простоту реалізації багатьох елементів коду. Якщо звертатись до визначень буквально, то кнопки на інтерфейсі також є ігровими об’єктами, тож формально умови проектування виконуються.

Визначившись з тим, яким способом будуть розміщуватись клітинки ігрового поля потрібно їх розмітити. Оскільки ігрове поле буде виконуватись на інтерфейсі гравця, то потрібно створити відповідний ігровий об’єкт Canvas. Canvas – це базовий елемент системи інтерфейсу користувача (UI) в Unity, призначений для відображення графічних компонентів інтерфейсу, таких як кнопки, текстові поля, зображення, повзунки, меню та інші елементи взаємодії. Компонент Canvas виконує роль контейнера для всіх UI-об’єктів у сцені. Без нього графічні елементи інтерфейсу не будуть відображатися. Усі об’єкти типу Button, Image, Text, Dropdown, Slider тощо повинні бути розміщені в межах одного з Canvas-компонентів. Під час додавання будь-якого елемента інтерфейсу через меню GameObject та розділу UI, система автоматично створює Canvas, якщо такий ще відсутній у сцені, а також додає необхідні допоміжні компоненти – Canvas Scaler та Graphic Raycaster.

Для створення основних ігрових механік, модифікаторів та ігрового функціоналу, буде достатньо створити ігрове поле невеликого розміру 5 на 5 клітинок. Розмістивши всі кнопки на полотні інтерфейсу було отримано наступне ігрове поле, як показано на рисунку 1.13.

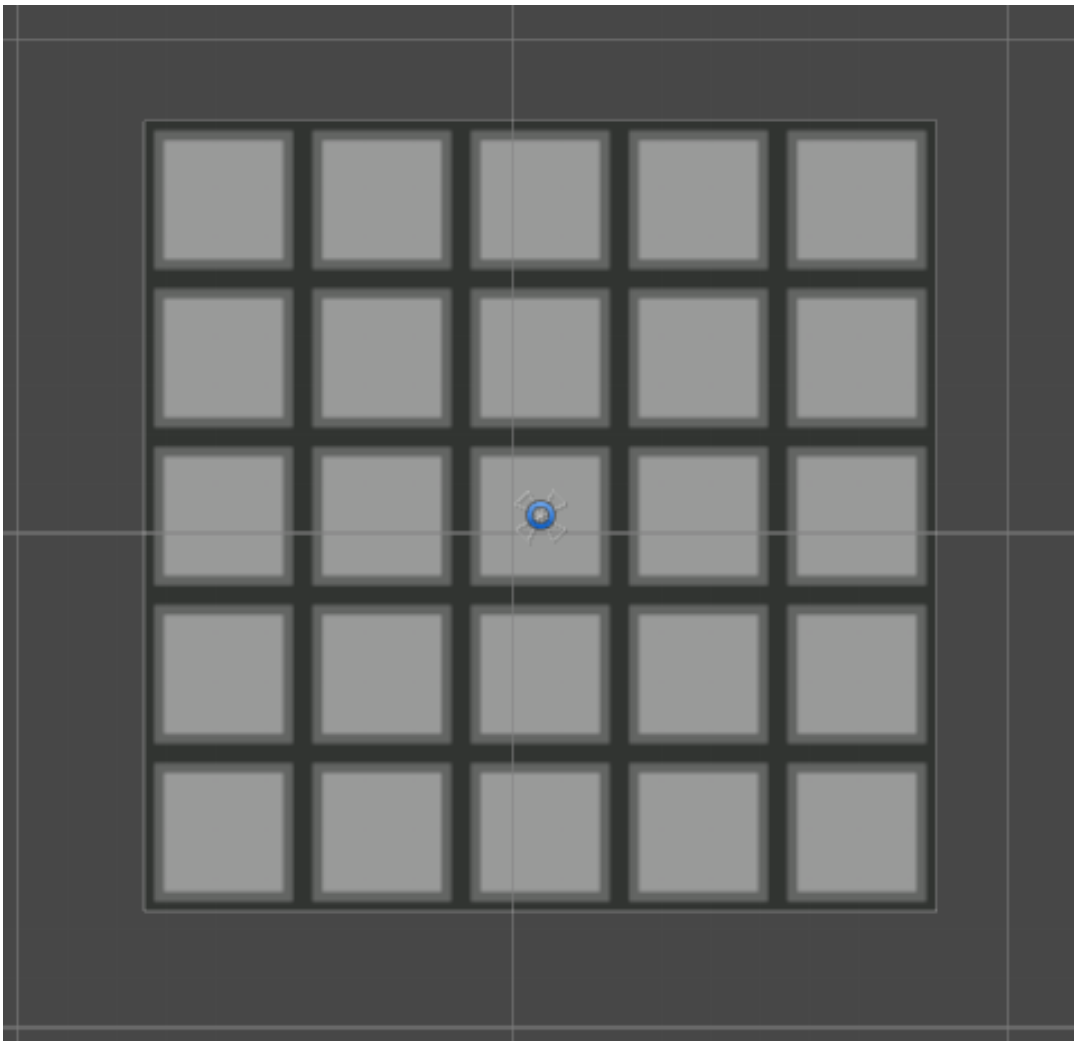


Рисунок 1.13. Реалізоване ігрове поле 5 на 5 клітинок за допомогою кнопок

Як можна побачити, до кожної кнопки використано спрайт кнопки, замість спрайту кнопки за замовченням. Також видно, що текстові елементи кнопок залишились, але їх зміст був стертий щоби у подальшому в ці текстові елементи писати інформацію про клітинку. Частину структура створеного ігрового поля зображена на рисунку 1.14. На рисунку видно, що клітинки створюються типовим шляхом: створюється кнопка, до неї автоматично додається текстовий елемент як нащадок.

Слід зазначити, що для текстових елементів використовується TextMeshPro. TextMeshPro – це потужна текстова система, яка замінює стандартні компоненти тексту (Text, TextMesh) у Unity, надаючи розширений функціонал для виведення текстової інформації з високою якістю відображення та широкими можливостями форматування.

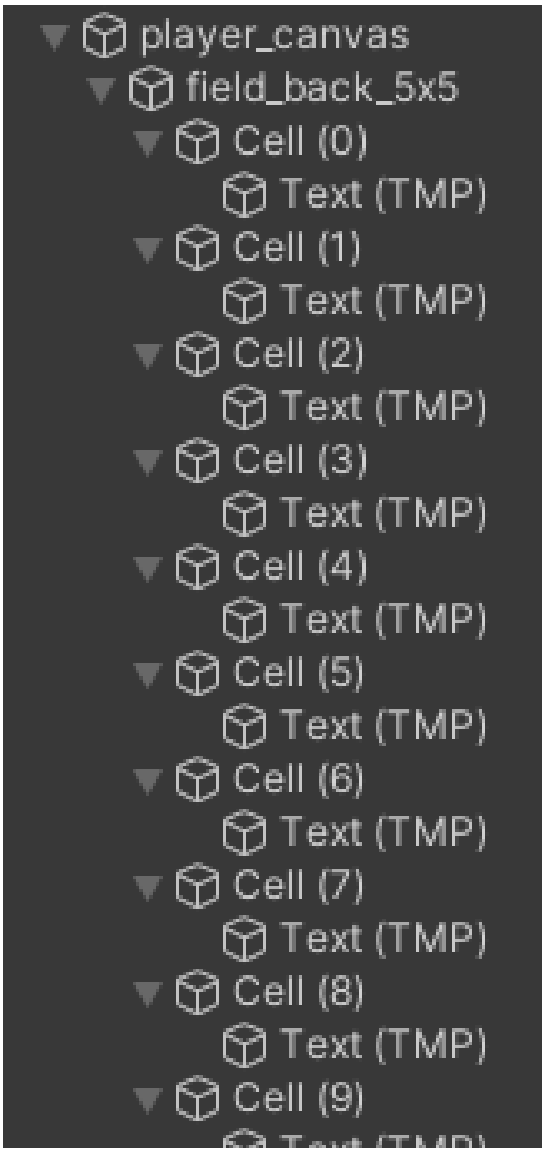


Рисунок 1.14. Часткова структура ігрового поля 5 на 5 клітинок

TextMeshPro дозволяє створювати та відображати текст із покращеною візуальною чіткістю, навіть при зміні розміру шрифту або масштабуванні UI. Це досягається за рахунок використання векторної графіки у вигляді SDF (Signed Distance Field) шрифтів, які рендеряться значно точніше, ніж звичайні растрові. Втім, для роботи цієї технології потрібно підключити до проекту додатковий каталог, який зберігає всі потрібні елементи для функціоналу TextMeshPro.

1.6.2 Реалізація механізму натискання на клітинки ігрового поля

Коли ігрове поле створене та може бути використане у тестуванні, потрібно реалізувати код який би забезпечував натискання на кнопки та обробку натискання на праву та ліву кнопку миші. На рисунку 1.15 зображено схему роботи логіки розробляємих модулів для обробки натискання на кнопки миші.

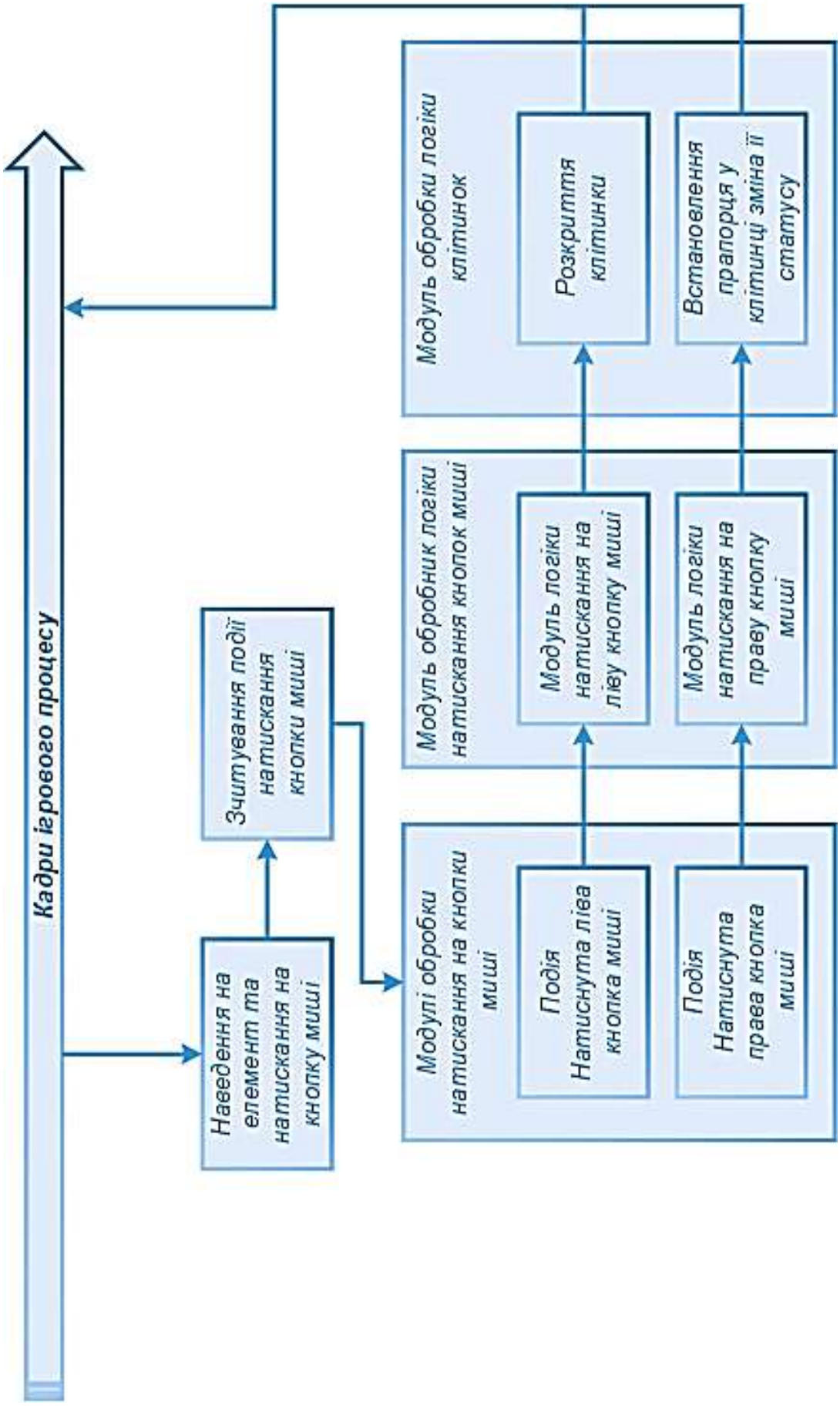


Рисунок 1.15. Схема роботи логіки натискання на кнопки миші

На рисунку 1.15 показано, що під час Кадру ігрового процесу виконується наведення на елемент (клітинку ігрового поля) та натискання на кнопку миші. Це викликає зовнішню подію, яка звертається до об’єкту на який натиснули. В свою чергу, всі скрипти, які мають у своєму складі елементи, які прослуховують події натискання на кнопки миші, починають виконувати відповідний код. Так, у Модулі обробки натискання на кнопки миші вказані блоки будуть виконуватись окремо один від одного, в залежності, яка кнопка миші була натиснута. В свою чергу скрипт обробник (Модуль обробник логіки натискання кнопок миші) викликає інші скрипти логіки виконання дій при натисканні на ту чи іншу кнопку миші. У цих скриптах в свою чергу виконується виклик кодів, які будуть виконувати логіку клітинок. Після завершення цієї логіки, управління переходить знову до Кадру ігрового процесу.

Робота цього коду виконана у декількох модулях для того, щоби можна було виконувати налаштування подій при натискання на різні кнопки миші. Кожна подія натискання на ту чи іншу кнопку реалізована окремо, що дає змогу налаштувати анімації натискання окремо одну від одної. Також, оскільки код обробник використовує сопроцеси, розділення кнопок є логічним для полегшення роботи з ними.

Створення скриптів у ігровому програмному рушії Unity виконується декількома способами. Найпростіший та швидший, створити скрипт в каталозі ассетів гри. Перед тим потрібно створити окремий каталог для скриптів та в ньому створювати модулі. Під час створення скриптів необхідно виконувати всі правила іменування класів у C#, оскільки у ігровому програмному рушії використовується саме ця мова програмування, а кожний створений скрипт є класом всередині проекту гри. Тому, наприклад, неможна писати ім’я скриптів починаючи з цифер.

Робота скрипту обробки натискання на кнопку миші можна розглянути на прикладі натискання на праву кнопку, у реалізованому проекті він має назву right_mouse_button_click.cs. Слід зазначити одну особливість у створені обробників. Стандартні скрипти у Unity виконують наслідування не від object, як у мові C#, а від аналогу object-а у Unity – MonoBehaviour. Цей клас зберігає в собі

інформацію службові команди ігрового рушія, а також синтаксис скриптів Unity. У разі реалізації обробника, потрібно додати до MonoBehaviour ще й три інтерфейси: IPointerClickHandler, IPointerUpHandler, IPointerDownHandler.

Ці інтерфейси відповідають за обробку подій та дій з курсором та мишею у Unity. Також, потрібно створити подію відповідну до типу натискання на кнопку миші. Наприклад, як показано нижче, на натискання на праву кнопку:

public UnityEvent OnRightClick;

Цей код дозволить вивести у середу розробки відповідний інтерфейс для налаштування подій. Він буде відображатись у інтерфейсі середи розробки, через що з'явиться можливість обирати об'єкти та їх компоненти на виконання при натисканні на ту чи іншу кнопку миші. Також створюється змінна із посиланням на Button – клітинку, а також змінні для кольорів для їх зміни під час натискання на кнопку. На рисунку 1.16 для більш ефективного розуміння роботи модулю, зображено будову обробника right_mouse_button_click.cs.

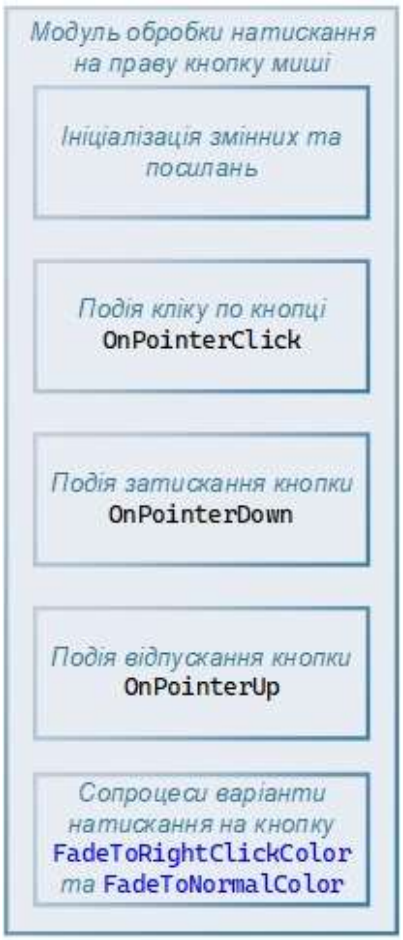


Рисунок 1.16. Структура Модулю обробки натискання на праву кнопку миші

Як можна бачити зі структури цей модуль виконує зчитування всіх видів подій ймовірних при натисканні на кнопку, просто натискання, затискання та відпускання кнопки. Це реалізовано для того, щоби можна було у майбутньому гнучким чином налаштовувати якісь інші події при різних видах натискань. Нижче приведено код цих подій:

```
public void OnPointerClick(PointerEventData eventData)
{
   if(eventData.button == PointerEventData.InputButton.Right)
   {
      OnRightClick?.Invoke();
   }
}
public void OnPointerDown(PointerEventData eventData){
   if (eventData.button == PointerEventData.InputButton.Right){
      StartCoroutine(FadeToRightClickColor());}
}
public void OnPointerUp(PointerEventData eventData){
   if (eventData.button == PointerEventData.InputButton.Right){
      StartCoroutine(FadeToNormalColor());}
}
```

Як можна побачити кожний з методів обробляє натискання через подію, після чого викликає відповідний код для виконання. У разі простого натискання на кнопку, викликається подія. У разі затискання та відпускання кнопки, завантажуються сопроцеси анімацій кнопок FadeToRightClickColor() та FadeToNormalColor(). Сопроцеси імітують паралельне виконання коду, не перехоплюючи управління програмою, а виконуючи свої команди підчас виконання основного тіла програми передаючи управління після виконання своєї команди на команду основної програми (основного потоку), на рисунку 1.17 схематично зображено цей процес.

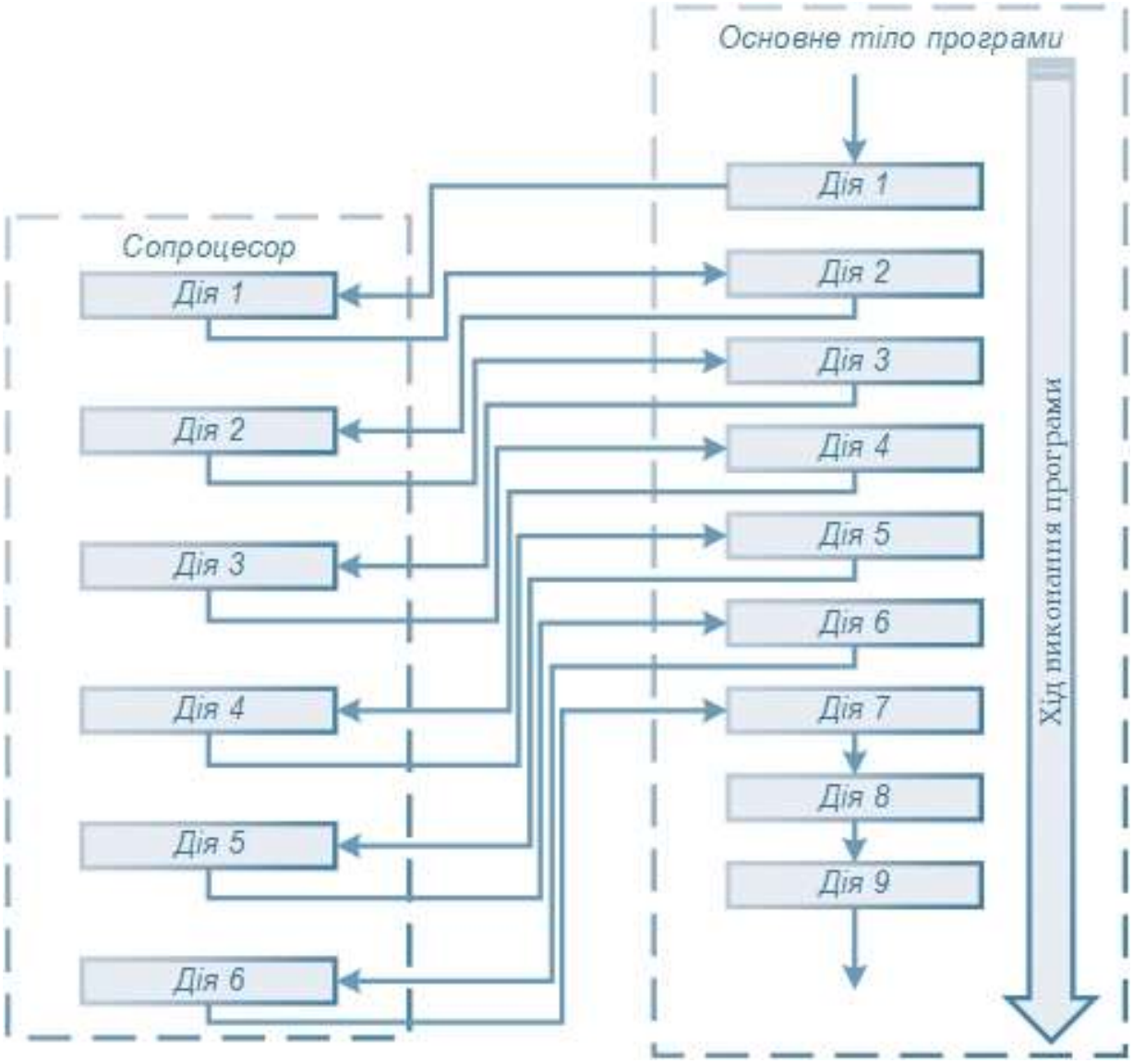


Рисунок 1.17. Схематичне зображення роботи сопроцесорів

Сопроцеси варіантів натискання на кнопки мають майже ідентичне виконання, оскільки виконують звортні одна одній дії. Спочатку отримується колір, який потрібно отримати в результаті виконання анімації, після цього створюється змінна time_elapsed яка рахує час виконання анімації. Далі сопроцесор викликає цикл, який буде виконуватись доки time_elapsed менша за час виділений на анімацію. Всередині циклу збільшується time_elapsed, створюється коефіцієнт пройденого часу між тактами циклу t, після цього сопроцесор змінює колір кнопки від поточного до цільового кольору з коефіцієнтом зсуву t. Нижче приведений приклад цього коду:

Color original_color = button.targetGraphic.color;

float time_elapsed = 0;

```
while(time_elapsed < RightClickColorDuration){
    time_elapsed += Time.deltaTime;
    float t = time_elapsed / RightClickColorDuration;
    button.targetGraphic.color = Color.Lerp(original_color, RightClickColor, t);
    yield return null;}
button.targetGraphic.color = RightClickColor;
```

Реалізований код дозволяє створити базу для основної механіки управління у грі, а саме натискання на клітинки ігрового поля.

1.7 Реалізація генерації рівнів

1.7.1 Реалізація Модуля логіки клітинки ігрового поля

Для подальшої реалізації проекту потрібно створити скрипт, який буде виконувати дії при взаємодії з клітинками на ігровому полі. Саме цей скрипт буде викликатись модулем обробки натискання на ту чи іншу кнопку миші. Згідно із результатами проектування, цей модуль повинен відтворювати логіку основних станів клітинки при натисканні. Серед таких станів є розкриття клітинки, встановлення прапорця, підрив міни та відображення тексту клітинки. Ці стани реалізовуються за допомогою відповідних методів, що викликаються ззовні. Сам скрипт не виконує розрахунків, а лише виконує виклики відповідних методів у основному коді для розрахунків. Цей скрипт має назву cell_click.cs та його схематична структура зображена на рисунку 1.18.

У частині Ініціалізації змінних та посилань, реалізовуємий модуль створює посилання на спрайти станів клітинки ігрового поля, а також на скрипт генератора рівнів. Серед зміних створюються логічні змінні, які відповідають за стан конкретної міни на полі: чи встановлено флаг is_flag_set, чи вибухнула міна is_mine_expl та чи розкрито кількість мін навколо клітинки is_number_set – всі ці змінні за замовченням встановлюються у статус false. Також цей модуль оперує текстовим полем кнопки. Всі зовнішні модулі, які вказані на рисунку 1.18 будуть розглянуті при реалізації модуля генерації рівнів.

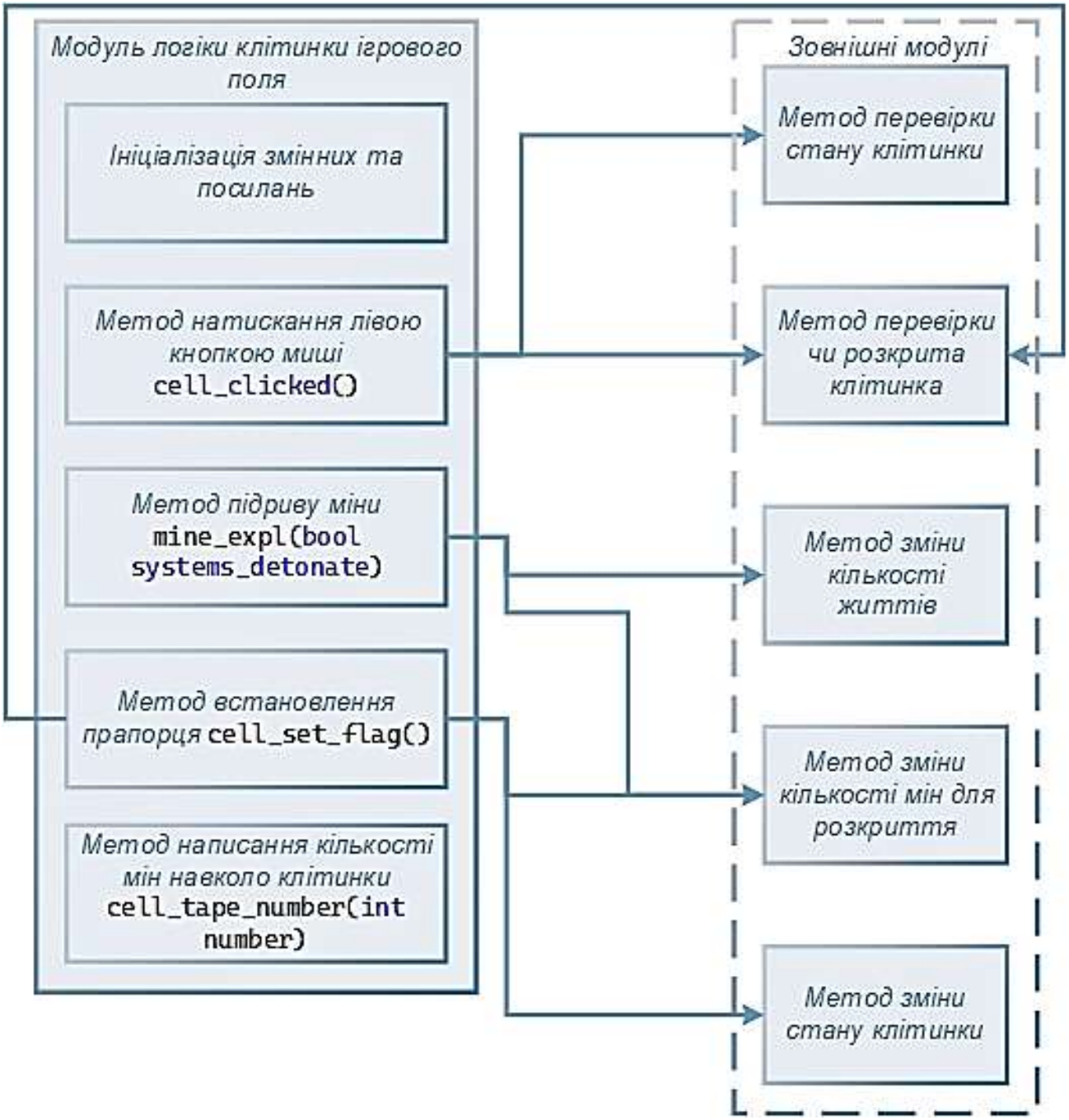


Рисунок 1.18. Схематична структура Модулю логіки клітинки ігрового поля

Метод cell_clicked() викликається ззовні, коли гравець натискає на клітинку та спрацьовують обробники події натискання на кнопку. В свою чергу, цей метод викликає зовнішній Метод перевірки стану клітинки у разі якщо логічні змінні встановлення прапорця та вибуху міни мають значення false, тобто перевірки стану клітинки виконується лише тоді, коли поточний статус клітинки не розкритий для гравця. Далі викликається зовнішній Модуль перевірки на розкриття всіх клітинок. Нижче приведено код розглядаємого методу:

```
public void cell_clicked(){
    if(!is_flag_set && !is_mine_expl)
        lvl_loader.cell_check_logic(this.name);
    lvl_loader.check_is_cell_revield();}
```

Метод mine_expl(bool systems_detonate) викликається також ззовні та приймає логічну змінну, що має вказувати міна детанує при розкритті гравцем, або при розкритті модифікатором рівня. В тілі методу виконується заміна спрайту кнопки з пустої клітинки на клітинку з міною. Далі, якщо міна не здетановує від модифікатора рівня, тоді викликається зовнішній Метод зміни кількості життів. Виконується зовнішній Метод зміни кількості мін для розкриття. В кінці роботи цього метода для клітинки встановлюється стан логічної змінної чи вибухнула міна у true. Нижче приведено код розглянутого метода:

```
public void mine_expl(bool systems_detonate){
    _cell_image_comp.sprite = _cell_image_mine;
    if(!systems_detonate)
        lvl_loader.dec_incr_lives_count(false);
    lvl_loader.dec_incr_mines_count(false);
    is_mine_expl = true;}
```

Метод cell_set_flag() виконується при зовнішньому виклику в той момент, коли потрібно встановити прапорець, тобто при натисканні на клітинку правою кнопкою миші. В першу чергу виконується перевірка станів клітинки: якщо кількість мін навколо клітинки не відображено та якщо міна не вибухнула, то виконується внутрішні перевірки. Ці перевірки зв’язані та їх суть полягає в тому, щоби встановити, чи стоїть у клітинці прапорець, чи ні. Це потрібно зробити для того, щоби реалізувати можливість знімати прапорець та ставити знову. Якщо прапорець не встановлено, то змінюється спрайт клітинки на спрайт із прапорцем, також змінюється стан змінної чи встановлено прапорец на true. Далі викликаються зовнішні методи для зміни кількості мін для розкриття у сторону зменшення, зміни статусу розкриття клітинки та виконується перевірка чи розкриті всі клітинки ігрового поля. У разі якщо прапорець встановлено, то

виконуються зворотні дії. Змінюється спрайт клітинки на пустий, перемикається стан встановлення прапорця клітинки на false та викликаються зовнішні методи для зміни кількості мін для розкриття у сторону збільшення, а також перемикач стану клітинки.

Метод cell_tape_number(int number) також викликається ззовні та виконує завдання написання цифри кількості мін навколо клітинки. Цей метод звертається до текстового поля у нащадку ігрового об’єкт клітинки та передає значення параметру цього метода. Після чого перемикає статус клітинки встановлення кількості мін навколо клітинки на true.

1.7.2 Реалізація модуля генерації рівнів

Перед розкриттям роботи Модуля генерації рівнів потрібно розкрити одну із змін у результатах проектування, яка була внесена вже під час реалізації. Початково було спроектовано систему, яка мала два повністю окремих роздільних модуля: Модуль для логіки клітинки та Модуль для генерації рівня. Початково, модуль логіки виконував всі обчислення, які мають відношення до клітинок, а модуль генерації лише створював ігрове поле. В ході реалізації цих елементів гри, стало зрозуміло, що розділити ці модулі повністю дуже складно, більше того їх розділення створювало ситуацію з переповненістю передачі даних між модулями та збільшенню даних з відкритим доступом, що ускладнювало контроль за виконанням коду.

По суті, модуль логіки клітинки не має прямого доступу до сусідніх клітинок, оскільки відповідає лише за свою клітинку. Для взаємодії із іншими клітинками ігрового поля було декілька варіантів вирішення цієї проблеми через створення у кожній клітинці списку всіх клітинок ігрового поля, або створити один масив даних по клітинкам. Під час роботи над генератором рівня закладалось створення масиву, який би зберігав ігрові об’єкти клітинок, за які він відповідає. Через це у цьому коді вже було реалізовано список по клітинкам, але іншого типу даних. Оскільки сам генератор рівня вже мав у собі створені сутності ігрових об’єктів клітинок, отже він мав і доступ до їх функціоналу. У зв’язку із тим було логічним виконувати всі операції із клітинками саме в коді генератора рівня. Це

також давало можливість більше легко орієнтуватись у виконуємих методах та уникнути великої кількості передачі даних між скриптами. В результаті розробки було створено скрипт схематичну структуру та зв’язки якого можна побачити на рисунку 1.19.

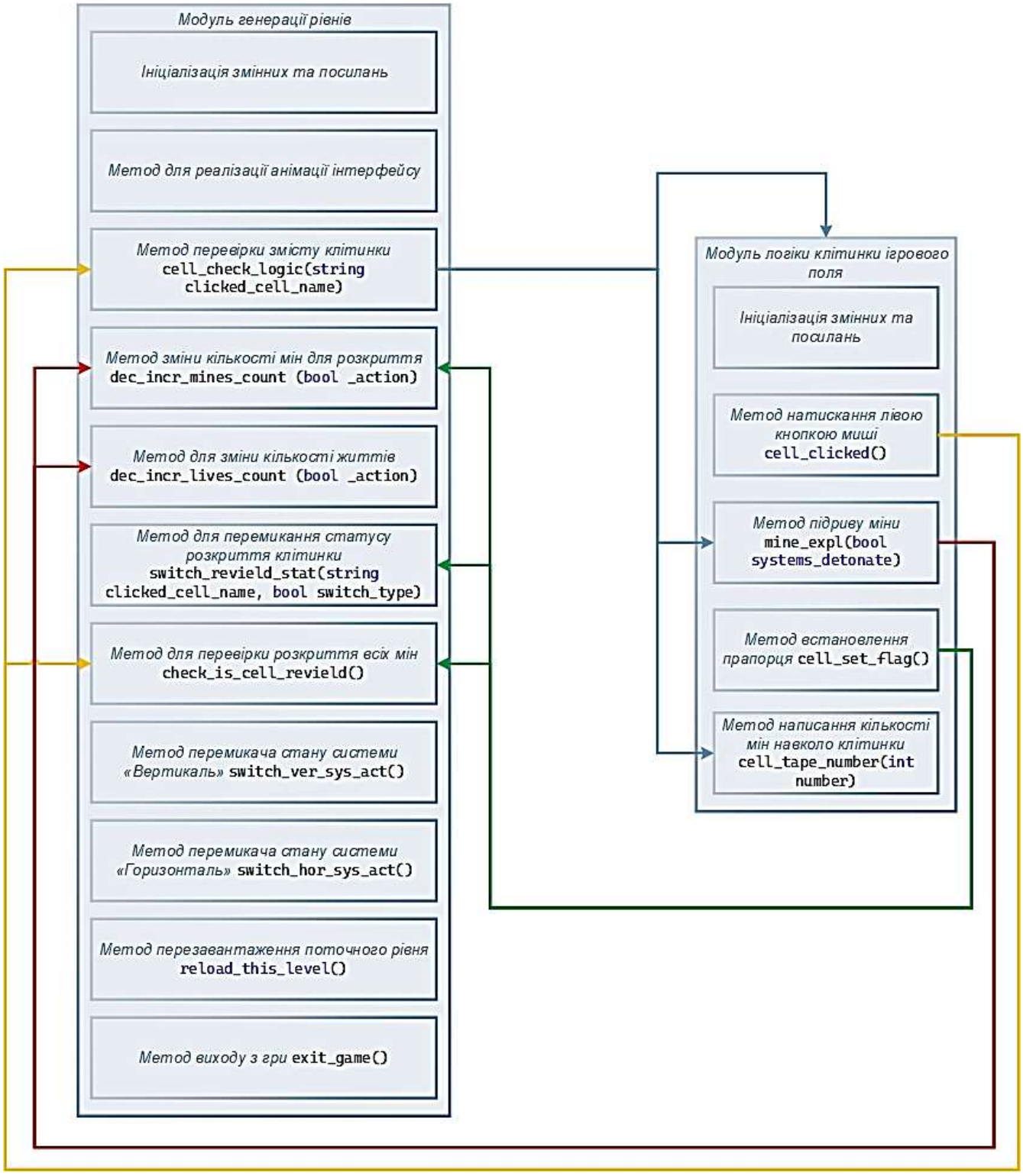


Рисунок 1.19. Схематична структура Модуля генерації рівнів та його зв’язки

У головні завдання Модулю генерації рівнів входить наповнення ігрового поля мінами та початкове налаштування ігрового процесу, встановлення кількості життів гравця, можливостей використання бонусів та інше. Також, як було вказано вище, виконує всі основні розрахунки та перевірки пов’язані із клітинками на ігровому полі оскільки саме в цьому модулі отримується доступ до ігрових об’єктів клітинок. У цьому підрозділі будуть розглянуті ті методи, які відповідають за генерацію рівня та деякі його функційні частини. Методи які відносяться до інтерфейсу та модифікаторів рівнів будуть розглянуті у відповідних розділах.

Блок ініціалізації змінних та посилань створює необхідні змінні та посилання для коректної роботи ігрового процесу. Кількість змінних велика, тому варто перерахувати лише найважливіші. Для роботи з ігровим полем, модулю потрібно створити чотири масиви для зберігання відповідної інформації. Нижче приведені типи масивів, їх імена та розміри:

bool[,] cells_stats_5x5 = new bool[5,5];

GameObject[,] cells_go_5x5 = new GameObject[5,5];

string[,] cells_names_5x5 = new string[5,5];

bool[,] is_cell_revield_5x5 = new bool[5,5];

Використання двовимірних масивів дозволяє напряму співвідносити клітинки на ігровому полі з елементами у масиві. Так, перша клітинка у другому рядку буде мати однаковий індекс у всіх масивах [0,1], тому знаючи цей індекс можна отримувати інформацію про цю клітинку, або доступ до її ігрового об’єкту. Масив cells_stats_5x5 має логічний тип та зберігає статус клітинки який вказує на те, замінована вона, чи ні. За замовченням всі клітинки на старті гри мають значення false – якщо клітинка замінована, то значення встановлюється у значення true. Масив cells_go_5x5 забезпечує доступ до ігрових об’єктів кожної клітинки на ігровому полі, що дає можливість звертатись до скрипту Модуля логіки клітинки ігрового поля. Масив cells_names_5x5 використовується для зберігання імен клітинок ігрового поля та потрібен для швидкого отримання індексу клітинки, з якою взаємодіє гравець. Масив is_cell_revield_5x5 використовується для

зберігання загального статусу розкриття клітинок ігрового поля. Він змінює значення комірок у випадку зміни статусу клітинок, при їх підриві, або розкритті чи встановленні-зняття прапорця.

Серед посилань, які використовує модуль можна виділити посилання на текстові поля, кнопки та інші елементи інтерфейси, робота з якими буде розглянута у відповідному підрозділі даної роботи.

Також потрібно окремо розглянути метод Start(), який входить до блоку Ініціалізації змінних та посилань. Цей метод виконується на момент старту життя скрипту та виконує стартову ініціалізацію значень змінних. В цьому модулі заповнюються масиви, налаштовуються бонуси та інше. Серед важливого слід відмітити, що саме тут виконується випадкове розміщення мін на ігровому полі. Цей процес виконується за допомогою вкладених циклів for, які проходять по масиву cells_stats_5x5. Кожний раз, коли у полі нерозміщених мін mines значення більше за 0, виконується отримання випадкового значення від 0 до 100. Якщо це значення більше за 50, то комірку масиву cells_stats_5x5 за індексом ітераторів циклів встановлюється значення true, що відповідає статусу замінування. Після чого пул нерозміщених мін зменшується на 1. Нижче приведено цей код:

```
for(int i=0; i<5; ++i){
   for(int i2=0; i2<5; ++i2){
      if(mines >0){
         cells_stats_5x5[i, i2] = false;
         int r = UnityEngine.Random.Range(0, 100);
         if (r > 50) {
            cells_stats_5x5[i, i2] = true;
            --mines;}
      }
      cells_go_5x5[i, i2] = GameObject.Find("field_back_5x5/Cell (" + counter
+ ")");
      cells_names_5x5[i, i2] = cells_go_5x5[i, i2].name;
      ++counter;
```

```
        is_cell_revield_5x5[i, i2] = false;}
    }
```

Також у коді вище можна побачити процес запису ігрових об’єктів у масив cells_go_5x5 через використання текстового посилання на них у Ієрархії ігрових об’єктів та особливості їх іменування, де кожна комірка нумерується у дужках від 0 до 24 для ігрового поля розміром 5 на 5. Також тут виконується отримання імен цих клітинок та їх запис у масив cells_names_5x5. Встановлюється статус розкриття клітинки у масиві is_cell_revield_5x5 у стан false.

Метод cell_check_logic(string clicked_cell_name) виконує одне з основних завдань по процесу розкриттю клітинок та їх статусу. Цей метод має дуже глибоку будову, частина якого буде розглянута у підрозділі 1.8. Спочатку цей метод створює змінні координат натиснутої кнопки та встановлює їх значення у 0. Далі виконується ідентифікація клітинки на яку було виконано натискання лівою кнопкою миші. Це виконується методом перебору вкладенними циклами масиву імен клітинок cells_names_5x5 та перевірки на співпадіння значень комірок цього масиву із рядком який отриманий як параметр цього методу clicked_cell_name. У разі співпадіння, у змінні координат натиснутої клітинки записуються значення ітераторів вкладених циклів. Нижче приведено цей код:

```
    int cell_x_coord = 0;
    int cell_y_coord = 0;
    for (int i = 0; i < 5; ++i){
       for (int i2 = 0; i2 < 5; ++i2)   {
          if(clicked_cell_name == cells_names_5x5[i,i2])      {
             cell_x_coord = i;
             cell_y_coord = i2;}
       }
    }
```

Наступним кроком є отримання скрипту Модуля логіки клітинки ігрового поля для натиснутої клітинки. Це потрібно для передачі відповідних команд до клітинок ігрового поля. Також тут створюється цілочислова змінна для

підрахунку мін навколо клітинки mines_count та встановлюється у значення 0. Спочатку виконується визначення, чи розкрита клітинка була замінованою, чи ні. У цій частині також виконується перевірка, розкриття клітинки викликано натисканням гравця, або модифікаторами рівня. Якщо це натискання гравця, то метод напряму звіряє статус натиснутої клітинки із масивом статусів клітинок cells_stats_5x5. Якщо статус true то викликається скрипт логіки клітинки ігрового поля та запускається відповідна послідовність дій при підриві.

У іншому випадку, якщо статус розкритої міни false, то починається перевірка клітинок навколо натиснутої клітинки з метою підрахувати кількість мін навколо неї. Цей процес має 16 вкладених перевірок. Така кількість перевірок викликана необхідністю передбачати ситуації, коли перевіряються клітинки у крайніх позиціях ігрового поля, що, якщо не виконувати ці перевірки, може призвести до критичної помилки виходу за межі масиву. Наприклад, якщо розглядається клітинка з координатами 0 та 0 по x та y відповідно, то неможна перевіряти клітинки деякі клітинки тому, що їх немає. На рисунку 1.20 схематично зображено за яким принципом виконуються перевірки та приклад перевірки.

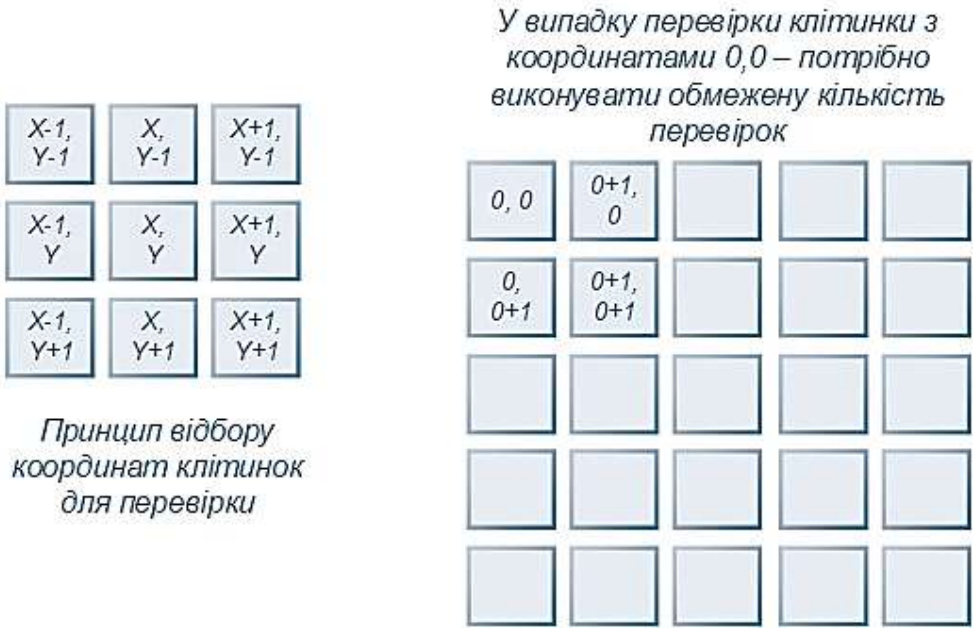


Рисунок 1.20. Приклад виконання перевірки клітинок та обмеженості кількості перевірок у деяких випадках

Як видно з рисунку 1.20 для клітинки з координатою 0 та 0 потрібно виконати лише 3 перевірки. Це правило обмеженості перевірок справедливе для

будь-яких клітинок, які розміщені на краях ігрового поля. Якщо спробувати перевірити клітинки за межами ігрового поля, то ми отримаємо значення координат, наприклад -1, що буде виходити за індексацію масивів клітинок ігрового поля та призводити до помилок. Під час перевірки кожної з судніх комірок, скрипт перевіряє по масиву cells_stats_5x5 чи заміновані вони та збільшує значення змінної mines_count. Після виконання всіх перевірок, виконується звернення до скрипту логіки клітинки ігрового поля на яку було натиснуто лівою кнопкою миші та викликається в ньому метод cell_tape_number(mines_count), куди передається кількість знайдених навколо мін.

Методи dec_incr_mines_count(bool _action) та dec_incr_lives_count(bool _action) схожі за принципом своєї реалізації та роботи. Як параметри, ці методи отримують логічне значення, яке вказує на тип дії яку потрібно виконати з кількістю мін що залишилось розкрити та кількістю життів. Якщо це значення false, то потрібно зменшити відповідний показник у методі, а якщо воно true то навпаки збільшити. Таким чином не потрібно створювати два методи на дії збільшення чи зменшення. Приклад частини коду для показника життів гравця показано нижче:

```
if (_action)
   player_lives++;
else
   player_lives--;
```

Метод switch_revield_stat(string clicked_cell_name, bool switch_type) викликається з коду методів логіки клітинок ігрового поля у момент, коли гравець встановлює прапорець. Як параметри цей метод отримує ім’я клітинки, яка змінює статус, а також тип зміни статусу true та false для випадків встановлення та зняття прапорцю відповідно.

Метод check_is_cell_revield() виконує роботу по загальній перевірці ігрового поля на відкриття всіх клітинок та повертає логічне значення результату перевірки. Мається на увазі, чи є на полі клітинки, в яких поточний статус невідомий. У разі якщо статус всіх клітинок розкрито та підтверджено, рівень

пройдено та повертається значення методу true. Перевірка виконується за допомогою вкладених циклів for та перевірки значень елементів масиву is_cell_revield_5x5. Якщо значення у комірці масиву буде false, то масив закінчує своє виконання та повертає значення false. Нижче приведено частину коду цього методу:

```
for(int i=0; i< 5; ++i){
   for(int i2=0; i2<5; ++i2){
      if (!is_cell_revield_5x5[i, i2]){
         return false;}
   }
}
```

Реалізовані функції забезпечують основний ігровий процес для розробляємого проекту.

1.8 Реалізація модифікацій рівнів

Для повного виконання завдання дипломного проектування та розробленої концепції гри потрібно реалізувати модифікатори рівнів, які визначені, як бонуси, що змінюють ігровий процес шляхом розкриття клітинок по горизонталі, або вертикалі. Як вже було зазначено вище у роботі, цей функціонал виконано у Модулі генерації рівнів, оскільки він має доступ до клітинок рівня, їх ігрових об’єктів, станів та імен. Окрім того, процес виконання модифікаторів для рівня дуже тісно пов’язано із процесом розкриття клітинок. По суті, при виконанні бонусу виконується розкриття кожної клітинки по вертикалі, або горизонталі, але з іншим типом розкриття.

Для функціонування модифікації рівнів у Модулі генерації рівнів створено деякі змінні та посилання. Серед змінних слід виділити дві логічні змінні, які зберігають статус активації бонусів, а також дві змінні які зберігають кількість разів, скільки можна використати ці бонуси. Нижче приведено методи які пов’язані з модифікаторами рівнів та реалізовані у Модулі генерації рівнів.

Метод switch_ver_sys_act() викликається при натисканні на кнопку бонусу

на інтерфейсі гри. Цей метод потрібен для перемикання активації роботи бонусу. Принцип його роботи досить простий. Якщо у гравця в запасі є невикористані бонуси для системи «Вертикаль», то значення змінної активації роботи цієї системи ver_sys_act перемикається на протилежне. Якщо це значення true то текст кнопки бонусу вказує, що система активована, а якщо значення ver_sys_act false, то текст кнопки бонусу стає за замовченням. У разі якщо під виконання цього методу вже була активована система «Горизонталь», то її значення скидається у false та змінюється текст кнопки на відповідний. Нижче приведено код цього методу:

```
public void switch_ver_sys_act(){
   if(ver_sys_count >0){
      ver_sys_act = !ver_sys_act;
      if (ver_sys_act)
         ver_sys_text.text = "Деактивувати систему \"Вертикаль\"";
      else
         ver_sys_text.text = "Використати систему \"Вертикаль\" x " +
ver_sys_count;
      if (hor_sys_act){
         hor_sys_act = false;
         hor_sys_text.text = "Використати систему \"Горизонталь\" x " +
hor_sys_count;}}
   }
```

Метод switch_hor_sys_act() аналогічний у реалізації із розглянутим модулем вище, лише змінюється типи значень, які перемикаються.

Метод cell_check_logic(string clicked_cell_name) забезпечує активне виконання бонусів. Під час розгляду роботи цього методу вище, було проілюстровано його роботу у разі, якщо бонуси не активовані, для цього виконується відповідна перевірка. У разі, якщо бонуси активовані, то виконується перевірка, який саме бонус активовано. В залежності від того, який бонус активовано, кількість активацій hor_sys_count або ver_sys_count зменшується на

1. Далі, в залежності від того, який тип бонусу активовано виконується циклічне розкриття клітинок по координаті X для горизонтального бонусу та Y для вертикального.

Код послідовно отримує доступ до скриптів логіки клітинки ігрового поля та, якщо поточна клітинка замінована, то викликає метод mine_expl(true) із аргументом true, який вказує на те, що цей підрив виконаний бонусом та не призводить до зменшення життів гравця. Інакше, якщо поточна клітинка не замінована, то виконується перевірка клітинок навколо неї та виводиться їх кількість методом cell_tape_number(mines_count) для розглядаємої клітинки. Кількість знайдених мін mines_count встановлюється у 0, цикл переходить до перевірки наступної клітинки у рядку горизонталі, чи вертикалі. Після закінчення виконання роботи бонусу, його стан активації встановлюється у false, а кнопка бонусу отримує напис за замовченням.

Реалізований код дає змогу модифікувати ігровий процес та рівень через розкриття клітинок, що дає можливість гравцю виходити із складних ситуацій а також змінювати ігровий рівень під себе.

1.9 Реалізація інтерфейсу

Ігровий процес має бути забезпечений відповідним інтерфейсом. На етапі розробки концепції було встановлено, що у гравця будуть життя у виді саперних щупів. Окрім того, гравець має знати, скільки мін йому потрібно ще розкрити. Також у гравця має бути реалізовано доступ до бонусів, які змінюють ігрові рівні та повинні виводитись службові написи на екран гравця для його розуміння поточної ситуації. Для забезпечення візуальної складової створено відповідні спрайти, для кнопок бонусів використовуються кнопки за замовченням.

Щоби реалізувати інтерфейс використовується полотно Canvas яке вже було створено раніше для реалізації ігрового поля. Елементи інтерфейсу, як зображення життів виконується за допомогою ігрових об’єктів типу Images, в яких змінюється параметр Source Image компоненту Image. Для того щоби гравець розумів кількість показників життя та кількості мін для розкриття потрібно додати

до полотна ігрові об’єкти типу Text – TextMeshPro. Створені елементи розміщуються у верхньому лівому куту екрану та закріплюються там завдяки якоря. На рисунку 1.21 зображено елементи життів гравця та кількості мін для розкриття на полотні.

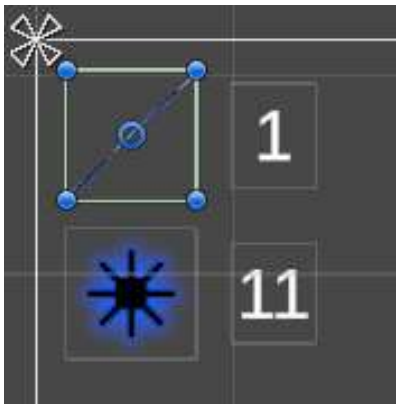


Рисунок 1.21. Розміщені елементи життів гравця та кількості мін для розкриття

Ще одним текстовим полем для розуміння гравцем статусу гри є поле для переможного або програшного повідомлення. Воно також реалізовано за допомогою ігрового об’єкту типу Text – TextMeshPro та розміщено над ігровим полем гравця.

Кнопки для бонусів, що змінюють ігрові рівні реалізовано за допомогою актуальних сучасних ігрових об’єктів Button. Є альтернативні кнопки, які знаходяться у вкладці Legacy та дають змогу використовувати старі варіанти цих ігрових об’єктів з використанням текстових полів старого стандарту. На рисунку 1.22 зображено створені кнопки для бонусів.

Рисунок 1.22. Створені кнопки для бонусів

Недостатньо тільки розмістити елементи інтерфейсу, потрібно запрограмувати їх роботу. У модулях генерації рівня та логіки клітинки ігрового поля, є рядки коду, які звертаються до кнопок, або текстових полі, у разі потреби

їх змінити. Для отримання доступу до цих елементів інтерфейсу створено відповідні посилання.

Окремо слід розглянути код який реалізує анімацію руху показників життів гравця та кількості мін для розкриття. Цей код реалізовано в Модулі генерації рівнів, оскільки в цьому модулі використовуються значення цих бонусів. Код анімацій створено у методі FixedUpdate(), який виконується 60 разів рівномірно розподілено по секунді реального часу. За творчим задумом зображення показників мають хитатись із сторони в сторону. Для додано зміну логічного типу interface_rotation_direction – її значення вказують напрям руху по годинниковій стрільці або проти. Кожний кадр метод повертає зображення бонусів трохи спочатку в одну сторону, доки кут нахилу не стане більшим за 20. Тоді значення змінної interface_rotation_direction змінюється на протилежне та починається виконуватись хитання в іншу сторону. Нижче приведено код анімації елементів інтерфейсу:

```
private void FixedUpdate(){
    if(interface_rotation_direction){
        player_lives_image.transform.Rotate(new Vector3(0,0,0.1f));
        mines_count_image.transform.Rotate(new Vector3(0, 0, 0.1f));
        if (player_lives_image.transform.rotation.eulerAngles.z > 20f)
            interface_rotation_direction = false;}
    else{
        player_lives_image.transform.Rotate(new Vector3(0, 0, -0.1f));
        mines_count_image.transform.Rotate(new Vector3(0, 0, -0.1f));
        if (player_lives_image.transform.rotation.eulerAngles.z < 1f)
            interface_rotation_direction = true;}
}
```

Після реалізації інтерфейсу, гра була приведена у виконаний стан та готова для тестування ігрового процесу. В цілому, реалізований код надає можливість у подальшому додавати ще більші рівні, а також створює базис для подальшого розвитку проекту.

1.10 Тестування розробленої гри

Для якісної реалізації будь-якої гри, її необхідно тестувати та відлагоджувати не тільки під час розробки, а ще й після закінчення кожного етапу розробки. Багато ігрових продуктів продовжують тестувати навіть тоді, коли гра у фінальній версії вже відправилась до печаті, або у магазини. Це робиться для того, щоби якомога швидше виправляти помилки, які не були знайдені під час розробки.

У випадку з реалізацією ігор на ігровому програмному рушії Unity, цей процес виконується безпосередньо в самому редакторі. Для тестування та відладки гри під час розробки, потрібно встановлювати ігрові об’єкти та сцени у такий стан, який потрібно протестувати та натиснути кнопку «Play». Таким же чином виконується тестування і відладка після закінчення розробки, достатньо виставити гру у її початковий стан и натиснути ту ж саму кнопку. Тестування виконувалось на одному розмірі ігрового поля, оскільки реалізація інших розмірів ігрового поля не має жодних елементів розробки, та несе характер шаблону, за яким можна створювати ігрові поля будь-якого розміру. Єдині помилки, які можна зробити при такій роботі несуть лише орфографічний характер та вирішуються на етапі написання коду. На рисунку 1.23, 1.24, 1.25 можна побачити процес тестування гри.

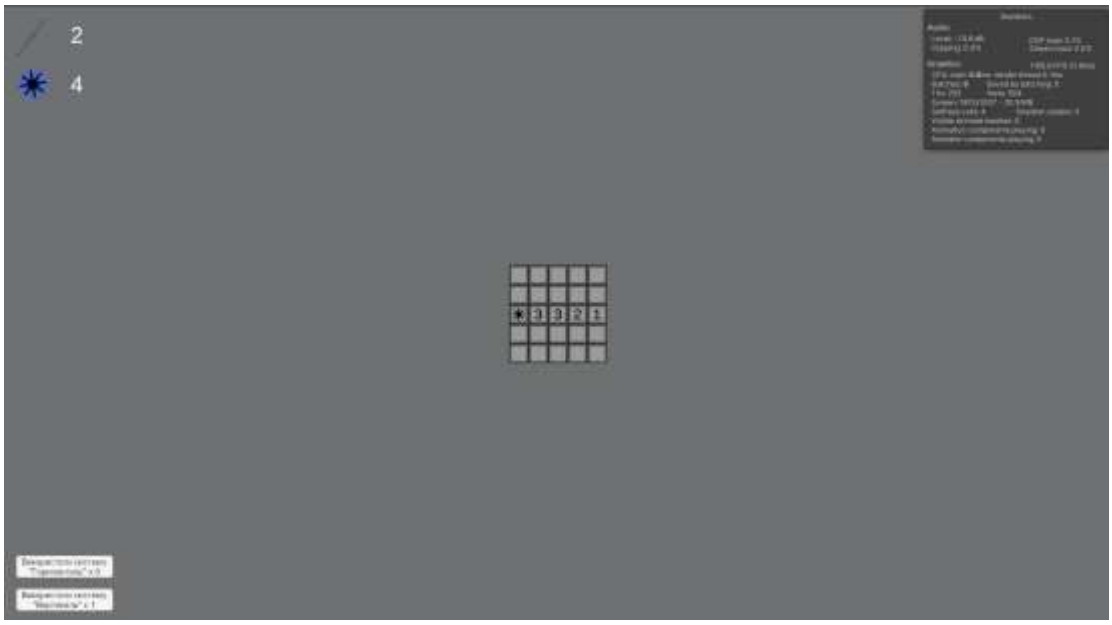


Рисунок 1.23. Процес тестування гри у ігровому програмному рушії Unity

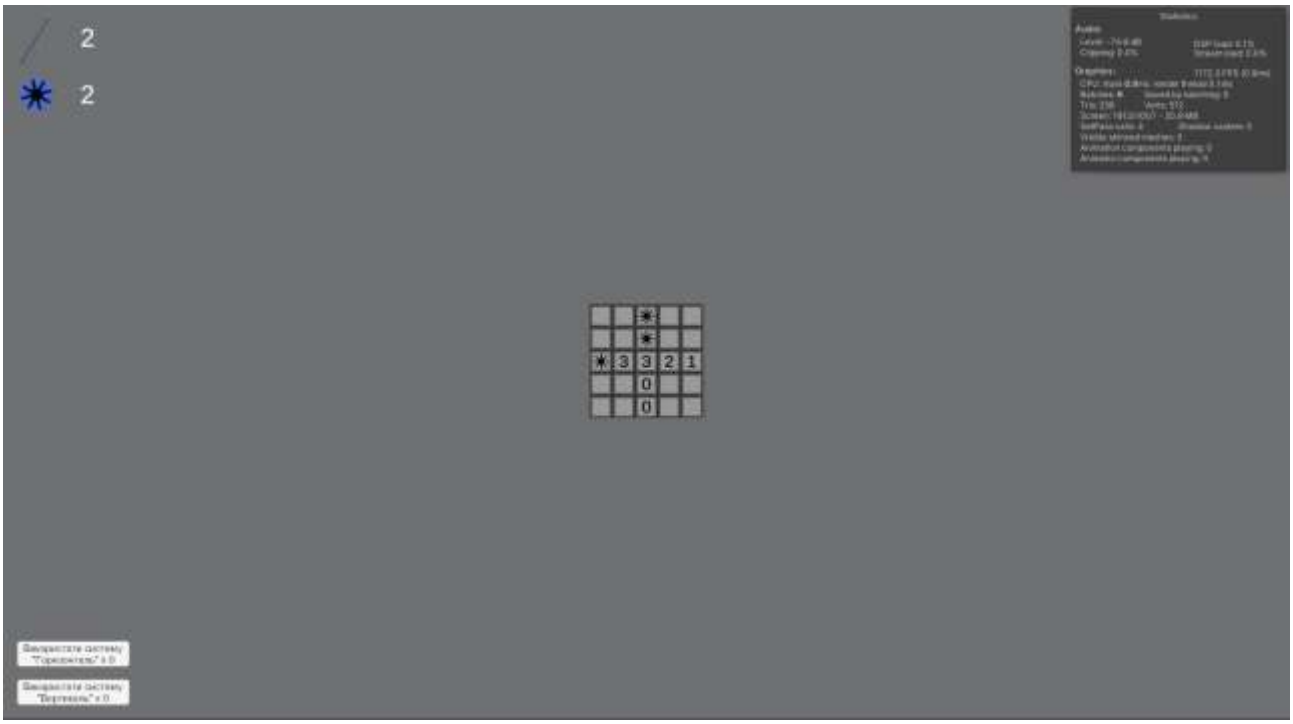


Рисунок 1.24. Процес тестування гри у ігровому програмному рушії Unity

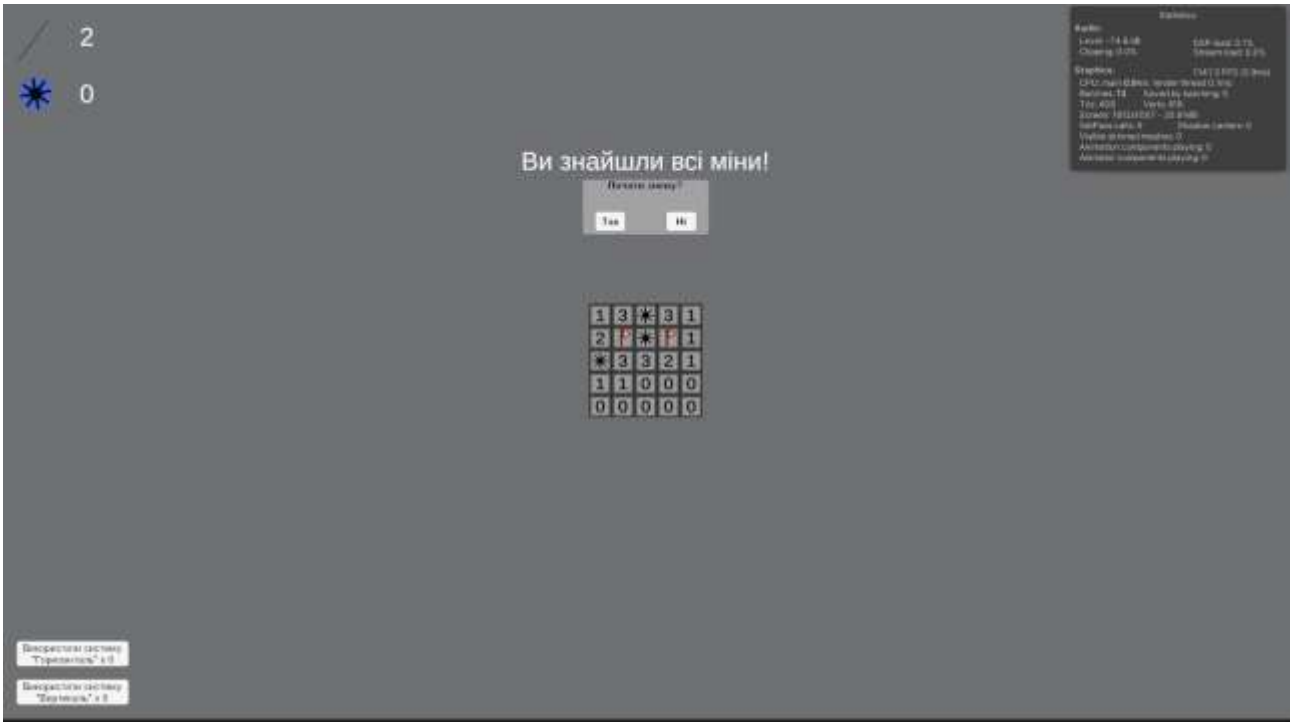


Рисунок 1.25. Процес тестування гри у ігровому програмному рушії Unity

Під час тестування гри було знайдено деякі помилки пов’язані з логікою перевірки. В основному ці помилки викликані пропущеною перевіркою статусу активованого бонусу, або взагалі його активації. Однією з помилок, які були знайдені під час реалізації проекту, були виходи за межі масивів під час перевірки, або отримання некоректного імені клітинки.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Темою мого дипломного проекту була «Розробка гри «Сапер» з модифікацією рівнів на ігровому рушії Unity». Під час гри можна обирати площу ігрової поверхні, а під час самого ігрового процесу використовувати різні додаткові модифікатори, які можуть відкривати клітинки безпечно для гравця. Для виконання цієї мети було використано сучасний ігровий програмний рушій Unity а також середу розробки Visual Studio для створення та редагування коду гри. У цьому розділі наведено розрахунок вартості розробленого програмного забезпечення.

Ефективність програмного продукту залежить як від його якісних показників, так і від результативності процесу його розробки. Якість ПЗ аналізується за кількома напрямами: з точки зору користувача, за рівнем раціонального використання ресурсів і відповідністю визначеним вимогам. Для користувача важливою складовою якості є оцінка витрат на створення продукту, зокрема трудових зусиль і фінансових витрат.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість створення програмного продукту визначається низкою чинників, серед яких – обсяг робіт, рівень трудомісткості, кваліфікація розробників, а також строки, що задаються ринковими умовами. Для оцінки обсягу програмного забезпечення використовується метод структурної аналогії, згідно з яким на основі спеціалізованих каталогів аналогів визначається кількість умовних машинних команд для подібного програмного продукту (у тисячах команд).

Таблиця 2.1. Каталог аналогів

Найменування ПП	*Обсяг функції ПП – V_o, усл. машинних командах.*
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП імітаційного моделювання	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткість визначаємо на основі табл.2.2.

Таблиця.2.2. Таблиця трудомісткості

Обсяг ПЗ, тис.умов.машинних команд	*Норма часу, люд/год*
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначаємо укрупнену норму часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПЗ, тобто в умовах комп'ютера, Кк=0,7÷0,8), для нашого варіанта виділено сірим кольором:

Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначаємо по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.).

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	*Ступінь новизни*		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	*Ступінь новизни*	*Значення $K_н$*
А	Принципово нове ПЗ	1,75 – 1,2
Б	ПЗ – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПЗ маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПЗ типовими програмами, %	*Значення $K_т$*
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором. Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$T_{тз}=Ta*L_1*K_н=183,2 *0,12*0,7= 15,38 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=Ta*L_2*K_н=183,2 *0,11*0,7 = 14,11 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$$T_{рп}=Ta*L_3*K_н*K_т=183,2 *0,61*0,7*0,6= 46,94 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=2 (стр), розробка ТП $N_{тп}$=28 (стр), розробка

робочого проекту $N_{рп}$=7 (стр), пояснювальна записка відповідно $N_{пз}$ 20 (стр). Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	*Розрахунок, годин.*		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*1=0,7	$T_{нк}=0,15*N_{тз}=0,15*1=0,15$
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}=0,7*30=21$	$T_{нк}=0,15*N_{тп}=0,15*30=4,5$
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}=0,7*N_{рп}=0,7*11=7,7$	$T_{нк}=0,15*N_{рп}=0,15*11=1,65$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$= 1,5*19=28,5	$T_{кк}=0,7*N_{тз}=0,7*19=13,3$	$T_{нк}=0,15*N_{пз}=0,15*19$ =2,85
Усього, в т.ч.:	156,78		
- на розробку	$\Sigma T_р$=104,93		
- контроль керівника		$\Sigma T_{кк}$=42,7	
-нормоконтроль			$\Sigma T_{нк}$=9,15

2.3 Розрахунок ціни програмного продукту

Для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПЗ. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2025» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2025 року – 8000 гривень; мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	*Трудомісткість робіт, години*	*Погодинна тарифна ставка, грн.*	*Розрахунок, грн.*
1.Розробка ПП	104,93	48,00	5036,64
2.Контроль керівника	42,7	105,00	4483,5
3.Нормоконтроль	9,15	110,00	1006,5
Усього	-	-	ΣЗо= 10526,64

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	*Тип, модель*	*Кількість*	*Ціна одиниці, грн.*	*Вартість, грн.*
Папір	Лист А4	61	5.0	305,00
Разом	-	-	-	$В_{м\,і}$ =285,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 0,1*305=30,5
Усього				$В_м=В_{мі}+В_{тр\ з=}$ 335,5

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	*Значення, грн.*	*Формула розрахунку*
1. Матеріали	335,5	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	10526,64	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1052,66	$Зд = 0{,}1 \times Зо$ = 10526,64*0,1
4.Відрахування до єдиного фонду соціального внеску	2547,45	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (10526,64+1052,66)
5. Накладні витрати	4210,65	$Внак. = 0{,}4 \times Зо$ =0,4 * 10526,64
6. Повна собівартість	18672,4	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 335,5+10526,64+1052,66+2547,45 +4210,65

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_{пов}*Р)/100=(18672,4*10)/100=1867,24 \text{ грн.} \quad (2.4)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о = С_{пов}+П = 18672,4+1867,24 = 20539,64 \text{ грн.} \quad (2.5)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного забезпечення становитиме:

$$Ц_р=Ц_о+ПДВ = 20539,64 + 20539,64 *0.2 = 24647,57 \text{ грн.} \quad (2.6)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Забезпечення здорового та безпечного робочого середовища є ключовим завданням керівництва підприємств, установ і організацій. Адміністрація несе відповідальність за впровадження сучасних заходів охорони праці, що мінімізують ризики виникнення травм і сприяють створенню комфортних санітарно-гігієнічних умов. Це, своєю чергою, допомагає запобігти професійним захворюванням і забезпечує сприятливі умови для продуктивної діяльності співробітників.

3.1 Аналіз шкідливих та ризикових факторів

При проведенні паяльних робіт співробітники піддаються впливу низки шкідливих та небезпечних чинників, що виникають при використанні спеціалізованих інструментів. Серед основних факторів ризику слід відзначити:

- роботу з комп’ютерною та електротехнічною апаратурою,
- недостатню освітленість робочої зони,
- психоемоційні навантаження,
- високий рівень шуму,
- недостатню вентиляцію приміщення,
- порушення правил пожежної безпеки тощо.

3.2 Гігієнічні вимоги до виробничого середовища

Для безперебійного, безпечного та якісного виконання паяльних робіт необхідно суворо дотримуватись правил техніки безпеки та організувати робоче місце оптимальним чином. Це означає, що всі інструменти та матеріали для паяння мають бути систематизовано розміщені, а роботи виконувати у заздалегідь підготовлених зонах, де мінімізовано вплив зовнішніх факторів.

Параметри мікроклімату робочої зони повинні відповідати вимогам санітарних норм мікроклімату виробничих приміщень (ДСН 3.3.6.042-99).

Рівень шуму має не перевищувати встановлених норм щодо виробничого шуму, ультразвуку та інфразвуку (ДСН 3.3.6.037-99).

Допустимі показники вібрації на робочих місцях зумовлені державними санітарними нормами загальної та локальної виробничої вібрації (ДСН 3.3.6.039-99).

Вимоги до рівнів електромагнітних полів визначені державними санітарними нормативами і правилами, затвердженими наказом МОЗ України від 18.12.2002 № 476.

3.3 Вимоги до організації робочого місця працівника

Згідно зі ст. 13 Закону України «Про охорону праці» (від 14.10.1992 р. № 2694-XII), роботодавець зобов'язаний забезпечити створення належних умов праці в кожному структурному підрозділі відповідно до чинних нормативно-правових актів та організувати лабораторні дослідження робочого середовища.

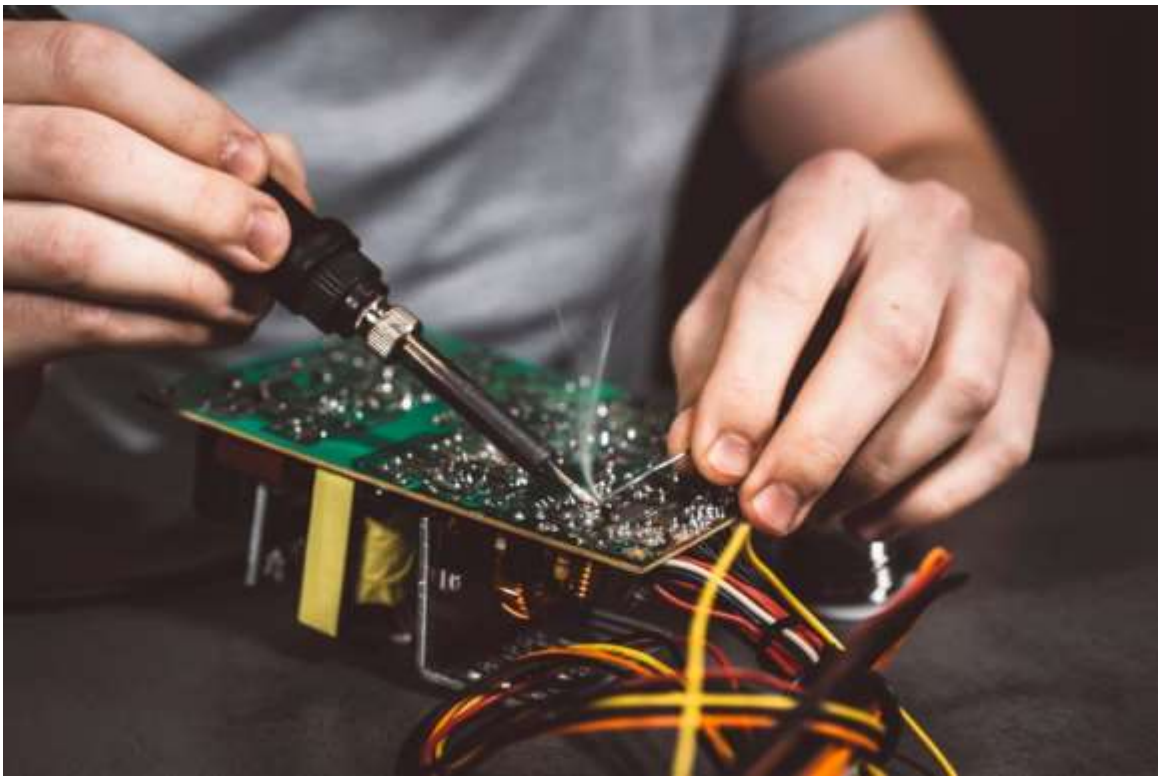


Рисунок 3.1. Процес паяння пристрою

Паяння використовується для з'єднання заготовок зі сталі, кольорових металів і їх сплавів, а також для створення з'єднань із зазначених матеріалів. Найчастіше ця технологія застосовується в електромонтажних роботах, монтажі контрольно-вимірювальних приладів, виробництві радіо- та електроприладів,

створенні теплових обмінників, а також у технологічних процесах, де використовують вироби з армованих пластин з твердих сплавів.

У виробничих приміщеннях концентрація шкідливих речовин не повинна перевищувати гранично допустимих значень, визначених відповідними стандартами (наприклад, ГОСТ 12.1.005-88 «Система стандартів безпеки праці. Загальні санітарно-гігієнічні вимоги до повітря робочої зони»).

Працівники, залучені до паяльних робіт, повинні мати забезпечення засобами індивідуального захисту, а також профілактичними засобами у вигляді захисних кремів, паст чи спеціального лікувально-профілактичного харчування.

Роботодавець повинен організувати:

Організувати проведення попередніх медичних оглядів (при прийнятті на роботу) та регулярних періодичних оглядів відповідно до затвердженого порядку МОЗ України (наказ від 21.05.2007 № 246).

Провести атестацію робочих місць за умовами праці відповідно до встановлених норм (відповідно до постанови Кабінету Міністрів України від 01.08.1992 № 442).

У разі необхідності розробити і впровадити заходи з мінімізації шкідливого впливу виробничих чинників на здоров’я співробітників.

3.4 Електробезпека

Обладнання, таке як персональні комп'ютери, периферійні пристрої, апаратура управління, контрольно-вимірювальні прилади та освітлювальні засоби, а також електропроводи і кабелі, мають відповідати класифікаційним вимогам за зоною застосування та бути обладнаними захисними елементами для запобігання коротким замиканням та іншим аварійним ситуаціям.

Лінія електропостачання для ПК і периферії повинна формувати окрему групову мережу з трьома провідниками: фазовим, робочим нульовим та захисним нульовим. При цьому нульовий захисний провід використовується виключно для заземлення апаратів, а його функціональність не може дублювати робочий нульовий провід. Він прокладається окремо від робочої лінії від групового

розподільника до електроживильних розеток, причому недопустиме підключення обох провідників до одного контактного затискача.

Основними причинами травмування електричним струмом є:

- прямий контакт з відкритими проводами,
- взаємодія з внутрішніми компонентами комп'ютера,
- використання несправного обладнання,
- відмова засобів захисту, з якими контактує користувач,
- непередбачене виникнення напруги через пошкодження ізоляції.

Для ефективного запобігання ураження струмом необхідно:

- суворо дотримуватись інструкцій з виконання робіт і правил експлуатації обладнання,
- забезпечувати недоступність частин пристроїв, що працюють під високою напругою, для оператора,
- використовувати високоякісні ізоляційні матеріали, товщина яких відповідає вимогам безпеки,
- підключати електроживлення через спеціально обладнані розетки з функцією занулення,
- розраховувати споживану потужність для запобігання перевантаженням,
- здійснювати надійне заземлення всіх металевих корпусів, доступних для оператора.

3.5 Пожежна безпека

Виробничі приміщення, технологічні установки та будівлі повинні бути обладнані першоджерельними засобами пожежогасіння, до яких належать:

- вогнегасники,
- контейнери з піском,
- негорючі покривала з теплоізоляційного матеріалу,
- високоміцні тканинні вироби тощо.

Ці засоби повинні відповідати нормативним вимогам, затвердженим документами з технологічного проектування та Правилами пожежної безпеки в Україні (НАПБ А.О1.001-2014). Вогнегасники слід встановлювати в легкодоступних, добре помітних місцях (наприклад, в коридорах, біля входів та виходів або у зонах підвищеного ризику виникнення пожежі), захищаючи їх від прямого сонячного випромінювання та впливу опалювальних приладів. Розміщення вогнегасників має забезпечувати їхнє повне відкриття, причому вони встановлюються не вище 1,5 м від підлоги та на безпечній відстані від дверей.

Рисунок 3.2. Засоби пожежогасіння

Також засоби пожежогасіння не повинні заважати евакуації персоналу. Виробничі приміщення повинні забезпечуватись запасними виходами, а двері до них мають бути позначені зрозумілими освітленими написами, наприклад, «Запасний вихід». План евакуації повинен бути розміщений у видному місці біля основного виходу.

ВИСНОВКИ

Темою дипломного проекту була «Розробка гри «Сапер» з модифікацією рівнів на ігровому рушії Unity». За мету було поставлено створення застосунку для персонального комп'ютера з можливістю подальшого розширення ігрового процесу та ігрових механік. Потрібно було створити основу для майбутнього більш складного та унікального проекту.

Для виконання цієї мети було використано сучасний ігровий програмний рушій Unity а також середу розробки Visual Studio для створення та редагування коду гри. Також у роботі використовувався графічний редактор Adobe Photoshope за допомогою якого створювались деякі графічні матеріали.

Виконання роботи було поділено на етапи проектування та реалізації. Під час проектування були розроблені модифікації ігрових рівнів, структури модулів гри, зв'язки між ними та принципи їх роботи. Реалізація складалась з роботи по відтворенню спроектованих модулів, створення та налаштування інтерфейсу. Виконання роботи над дипломним проектом було завершено на процесі тестування розробленої гри та перевірки працездатності реалізованих модифікацій рівнів.

Як результат було отримано класичну гру Сапер з привнесенням до нього модифікацій ігрових рівнів. Під час гри можна обирати площу ігрової поверхні, а під час самого ігрового процесу використовувати різні додаткові модифікатори, які можуть відкривати клітинки безпечно для гравця.

Під час реалізації було дещо змінено початковий ігровий концепт, окрім того процес реалізації показав існування проблематики оптимізації виконання коду, а саме у частині розділення частин кодів проекту. У поточній реалізації більшість ігрового функціоналу реалізовано у одному класі. На даний момент цей клас є досить великим та складним з точки зору орієнтації в ньому для майбутнього супроводу проекту. У зв'язку з тим, потрібно буде виконати перепроектування структури ігрової логіки не для оптимізації ігрового процесу, а з метою створення умов для подальшої підтримки гри.

Створений проект можна модифікувати через використання модульної системи розробки. Кожний елемент гри можна додатково відредагувати, або використати, як базу для створення нового ігрового функціоналу. Одним з напрямів модифікації є робота з розробки нових принципів модифікації ігрового процесу. Є доцільним створити справжню систему генерації рівнів відповідно до параметрів, що вказував би гравець. Також, у майбутньому можна покращити графічну складову, додати більшу кількість анімацій, створити візуальні ефекти для ігрового процесу. Додати звуковий супровід.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Ляшенко О.М., Електронний навчальний посібник «Розробка комп'ютерних ігор за допомогою Unity 3D», Херсон: видавництво ФОП Вишемирський В.С., 2018.;
2. Джозеф Хокінг Unity в дії / Джозеф Хокінг., 2018. – 335 с.;
3. Дейбук В.Г. Віртуальний електронний практикум: Навчальний посібник – Чернівці: Чернівецький нац. ун-т, 2021. – 188 с.;
4. Трофименко О.Г. С++. Алгоритмізація та програмування: підручник / О.Г. Трофименко, Ю.В. Прокоп, Н.І. Логінова, О.В. Задерейко. 2-ге вид. перероб. і доповн. – Одеса : Фенікс, 2019. – 477 с.;
5. Джон Менінг Unity для розробників. Мобільні мультиплатформені ігри / Джон Менінг, Періс Батфилд-Еддісон., 2018. – 352 с.;
6. Мосіюк О.О., Федорчук А.Л. Операційні системи та системне програмування: навчально-методичний посібник. Житомир: Вид-во ЖДУ ім. Івана Франка, 2022. – 76 с.;
7. Stroustrup B. A Tour of C++ (Second Edition). – Addison-Wesley, 2018. – 240 p. – ISBN 978-0-13-499783-4;
8. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms – Boston., McGraw-Hill, 2001. – 1082p.;
9. Robert Sedgewick, Kevin Wayne. Algorithms - Addison-Wesley, 2020. – 956 p. – ISBN 978-0321573513;
10. Procedural Generation in Game Design / Тарн Адамс, 2017. – 336 с.;
11. Бібліотека MSDN [Електронний ресурс]. – Режим доступу: URL http://msdn.microsoft.com/ru-ru/library/default.aspx (дата звернення 20.05.2025);
12. Бібліотека Unity [Електронний ресурс]. – Режим доступу: URL https://docs.unity.com/ (дата звернення 20.05.2025).

ДОДАТОК А. Лістинг основних модулів гри мовою C#

Скрипт level_loader.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
public class level_loader : MonoBehaviour
{
    bool interface_rotation_direction = true;
    [SerializeField] Image player_lives_image;
    [SerializeField] public short player_lives = 1;
    [SerializeField] TMP_Text player_lives_text;
    [SerializeField] Image mines_count_image;
    [SerializeField] public int mines_count = 1;
    [SerializeField] TMP_Text mines_count_text;
    [SerializeField] Text hor_sys_text;
    [SerializeField] public int hor_sys_count = 1;
    public bool hor_sys_act = false;
    [SerializeField] Text ver_sys_text;
    [SerializeField] public int ver_sys_count = 1;
    public bool ver_sys_act = false;
    [SerializeField] TMP_Text win_lose_message;
    [SerializeField] GameObject p_restart_go;
    bool[,] cells_stats_5x5 = new bool[5,5];
    GameObject[,] cells_go_5x5 = new GameObject[5,5];
    string[,] cells_names_5x5 = new string[5,5];
    bool[,] is_cell_revield_5x5 = new bool[5,5];
    bool[,] cells_stats_10x10 = new bool[10, 10];
    bool[,] cells_stats_15x15 = new bool[15, 15];
    void Start()
    {
        int mines = 5;
        int counter = 0;
        for(int i=0; i<5; ++i)
        {
            for(int i2=0; i2<5; ++i2)
            {
                if(mines >0)
                {
                    cells_stats_5x5[i, i2] = false;
                    int r = UnityEngine.Random.Range(0, 100);
                    if (r > 50)
                    {
                        cells_stats_5x5[i, i2] = true;
                        --mines;
                    }
                }
                cells_go_5x5[i,  i2]  =  GameObject.Find("field_back_5x5/Cell  ("  +
counter + ")");
                cells_names_5x5[i, i2] = cells_go_5x5[i, i2].name;
                ++counter;
                is_cell_revield_5x5[i, i2] = false;
            }
        }
```

```
            player_lives_text.SetText(player_lives.ToString());
            mines_count = 5;
            mines_count_text.SetText(mines_count.ToString());
            ver_sys_text.text = "Використати систему \"Вертикаль\" x " + ver_sys_count;
            hor_sys_text.text = "Використати систему \"Горизонталь\" x " +
hor_sys_count;
            win_lose_message.SetText("");
        }
        private void FixedUpdate()
        {
            if(interface_rotation_direction)
            {
                player_lives_image.transform.Rotate(new Vector3(0,0,0.1f));
                mines_count_image.transform.Rotate(new Vector3(0, 0, 0.1f));
                if (player_lives_image.transform.rotation.eulerAngles.z > 20f)
                    interface_rotation_direction = false;
            }
            else
            {
                player_lives_image.transform.Rotate(new Vector3(0, 0, -0.1f));
                mines_count_image.transform.Rotate(new Vector3(0, 0, -0.1f));
                if (player_lives_image.transform.rotation.eulerAngles.z < 1f)
                    interface_rotation_direction = true;
            }
        }
        public void cell_check_logic(string clicked_cell_name)
        {
            int cell_x_coord = 0;
            int cell_y_coord = 0;
            for (int i = 0; i < 5; ++i)
            {
                for (int i2 = 0; i2 < 5; ++i2)
                {
                    if(clicked_cell_name == cells_names_5x5[i,i2])
                    {
                        cell_x_coord = i;
                        cell_y_coord = i2;
                    }
                }
            }
            cell_click ck = cells_go_5x5[cell_x_coord, cell_y_coord].GetComponent
<cell_click>();
            int mines_count = 0;
            if(!hor_sys_act && !ver_sys_act)
            {
                if (cells_stats_5x5[cell_x_coord, cell_y_coord])
                {
                    is_cell_revield_5x5[cell_x_coord, cell_y_coord] = true;
                    ck.mine_expl(false);
                }
                else
                {
                    if (cell_x_coord > 0 && cell_y_coord > 0)
                    {
                        if (cells_stats_5x5[cell_x_coord - 1, cell_y_coord - 1])
                        {
                            ++mines_count;
                        }
                    }
```

```
                    if (cell_y_coord > 0)
                    {
                        if (cells_stats_5x5[cell_x_coord, cell_y_coord - 1])
                        {
                            ++mines_count;
                        }
                    }
                    if (cell_x_coord < 4 && cell_y_coord > 0)
                    {
                        if (cells_stats_5x5[cell_x_coord + 1, cell_y_coord - 1])
                        {
                            ++mines_count;
                        }                }
                    if (cell_y_coord < 4)
                    {
                        if (cells_stats_5x5[cell_x_coord, cell_y_coord + 1])
                        {
                            ++mines_count;
                        }                }
                    if (cell_x_coord > 0 && cell_y_coord < 4)
                    {
                        if (cells_stats_5x5[cell_x_coord - 1, cell_y_coord + 1])
                        {
                            ++mines_count;
                        }                }
                    if (cell_x_coord > 0)
                    {
                        if (cells_stats_5x5[cell_x_coord - 1, cell_y_coord])
                        {
                            ++mines_count;
                        }                }
                    is_cell_revield_5x5[cell_x_coord, cell_y_coord] = true;
                    ck.cell_tape_number(mines_count);
                }
            }
            else
            {
                if(hor_sys_act)
                {
                    hor_sys_count -= 1;
                    for (int i=0; i<5; i++)
                    {
                        cell_click      _temp_cc      =      cells_go_5x5[cell_x_coord,
i].GetComponent<cell_click>();
                        if (cells_stats_5x5[cell_x_coord, i])
                        {
                            _temp_cc.mine_expl(true);
                            is_cell_revield_5x5[cell_x_coord, i] = true;
                        }
                        else
                        {
                            if (cell_x_coord > 0 && i > 0)
                            {
                                if (cells_stats_5x5[cell_x_coord - 1, i - 1])
                                {
                                    ++mines_count;                              }
                            }
                            if (i > 0)
                            {
```

```
                            if (cells_stats_5x5[cell_x_coord, i - 1])
                            {
                                ++mines_count;                              }
                        }
                        if (cell_x_coord < 4 && i > 0)
                        {
                            if (cells_stats_5x5[cell_x_coord + 1, i - 1])
                            {
                                ++mines_count;                              }
                        }
                        if (cell_x_coord < 4)
                        {
                            if (cells_stats_5x5[cell_x_coord + 1, i])
                            {
                                ++mines_count;                              }
                        }
                        if (cell_x_coord < 4 && i < 4)
                        {
                            if (cells_stats_5x5[cell_x_coord + 1, i + 1])
                            {
                                ++mines_count;                              }
                        }
                        if (i < 4)
                        {
                            if (cells_stats_5x5[cell_x_coord, i + 1])
                            {
                                ++mines_count;                              }
                        }
                        if (cell_x_coord > 0 && i < 4)
                        {
                            if (cells_stats_5x5[cell_x_coord - 1, i + 1])
                            {
                                ++mines_count;                              }
                        }
                        if (cell_x_coord > 0)
                        {
                            if (cells_stats_5x5[cell_x_coord - 1, i])
                            {
                                ++mines_count;                              }
                        }
                        is_cell_revield_5x5[cell_x_coord, i] = true;
                        _temp_cc.cell_tape_number(mines_count);
                    }
                    mines_count = 0;
                }
                hor_sys_text.text = "Використати систему \"Горизонталь\" x " +
hor_sys_count;
                hor_sys_act = false;
            }
            if(ver_sys_act)
            {
                ver_sys_count -= 1;
                for (int i = 0; i < 5; i++)
                {
                    cell_click _temp_cc = cells_go_5x5[i, cell_y_coord].
GetComponent<cell_click>();
                    if (cells_stats_5x5[i, cell_y_coord])
                    {
                        _temp_cc.mine_expl(true);
```

```
                        is_cell_revield_5x5[i, cell_y_coord] = true;
                    }
                    else
                    {
                        if (i > 0 && cell_y_coord > 0)
                        {
                            if (cells_stats_5x5[i - 1, cell_y_coord - 1])
                            {
                                ++mines_count;                              }
                        }

                        if (i < 4)
                        {
                            if (cells_stats_5x5[i + 1, cell_y_coord])
                            {
                                ++mines_count;                              }
                        }
                        if (i < 4 && cell_y_coord < 4)
                        {
                            if (cells_stats_5x5[i + 1, cell_y_coord + 1])
                            {
                                ++mines_count;                              }
                        }
                        if (cell_y_coord < 4)
                        {
                            if (cells_stats_5x5[i, cell_y_coord + 1])
                            {
                                ++mines_count;                              }
                        }
                        if (i > 0 && cell_y_coord < 4)
                        {
                            if (cells_stats_5x5[i - 1, cell_y_coord + 1])
                            {
                                ++mines_count;                              }
                        }
                        if (i > 0)
                        {
                            if (cells_stats_5x5[i - 1, cell_y_coord])
                            {
                                ++mines_count;                              }
                        }
                        is_cell_revield_5x5[i, cell_y_coord] = true;
                        _temp_cc.cell_tape_number(mines_count);
                    }
                    mines_count = 0;
                }
                ver_sys_text.text = "Використати систему \"Вертикаль\" x " +
ver_sys_count;
                ver_sys_act = false;
            }
        }
    }
```

Скрипт cell_click.cs

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using Unity.VisualScripting;
```

```
using UnityEngine;
using UnityEngine.UI;
public class cell_click : MonoBehaviour
{
    [SerializeField] level_loader lvl_loader;
    [SerializeField] Image _cell_image_comp;
    [SerializeField] Sprite _cell_image_flag;
    [SerializeField] Sprite _cell_image_mine;
    [SerializeField] Sprite _cell_image_empty;
    bool is_flag_set = false;
    bool is_mine_expl = false;
    bool is_number_set = false;
    TMP_Text button_text;
    void Start()
    {
        button_text = GetComponentInChildren<TMP_Text>();
    }
    public void cell_clicked()
    {
        if(!is_flag_set && !is_mine_expl)
            lvl_loader.cell_check_logic(this.name);
        lvl_loader.check_is_cell_revield();
    }
    public void mine_expl(bool systems_detonate)
    {
        _cell_image_comp.sprite = _cell_image_mine;
        if(!systems_detonate)
            lvl_loader.dec_incr_lives_count(false);

        lvl_loader.dec_incr_mines_count(false);
        is_mine_expl = true;
    }
    public void cell_set_flag()
    {
        if(!is_number_set && !is_mine_expl)
        {
            if (!is_flag_set)
            {
                _cell_image_comp.sprite = _cell_image_flag;
                is_flag_set = !is_flag_set;
                lvl_loader.dec_incr_mines_count(false);
                lvl_loader.switch_revield_stat(this.name, true);
                lvl_loader.check_is_cell_revield();
            }
            else
            {
                _cell_image_comp.sprite = _cell_image_empty;
                is_flag_set = !is_flag_set;
                lvl_loader.dec_incr_mines_count(true);
                lvl_loader.switch_revield_stat(this.name, false);
            }
        }
    }
    public void cell_tape_number(int number)
    {
        button_text.SetText(number.ToString());
        is_number_set = true;
    }
}
```

Скрипт button_click_handler.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class button_click_handler : MonoBehaviour
{
    [SerializeField] cell_click _cell_click;
    public void on_left_mouse_button_click()
    {
    }

    public void on_right_mouse_button_click()
    {
        _cell_click.cell_set_flag();
    }
}
```

Скрипт right_mouse_button_click.cs

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.EventSystems;
using UnityEngine.UI;
public class right_mouse_button_click : MonoBehaviour, IPointerClickHandler,
IPointerUpHandler, IPointerDownHandler
{
    public UnityEvent OnRightClick;
    [SerializeField] private Color RightClickColor = Color.grey;
    [SerializeField] private float RightClickColorDuration = 0.1f;
    private Button button;
    private void Awake()
    {
        button = GetComponent<Button>();
    }
    public void OnPointerClick(PointerEventData eventData)
    {
        if(eventData.button == PointerEventData.InputButton.Right)
        {
            OnRightClick?.Invoke();
        }
    }
    public void OnPointerDown(PointerEventData eventData)
    {
        if (eventData.button == PointerEventData.InputButton.Right)
        {
            StartCoroutine(FadeToRightClickColor());
        }
    }
    public void OnPointerUp(PointerEventData eventData)
    {
        if (eventData.button == PointerEventData.InputButton.Right)
        {
            StartCoroutine(FadeToNormalColor());
        }
    }
```

```
        private IEnumerator FadeToRightClickColor()
        {
            Color original_color = button.targetGraphic.color;
            float time_elapsed = 0;
            while(time_elapsed < RightClickColorDuration)
            {
                time_elapsed += Time.deltaTime;
                float t = time_elapsed / RightClickColorDuration;
                button.targetGraphic.color         =         Color.Lerp(original_color,
RightClickColor, t);
                yield return null;
            }
            button.targetGraphic.color = RightClickColor;
        }
        private IEnumerator FadeToNormalColor()
        {
            Color original_color = button.targetGraphic.color;
            float time_elapsed = 0;
            while (time_elapsed < RightClickColorDuration)
            {
                time_elapsed += Time.deltaTime;
                float t = time_elapsed / RightClickColorDuration;
                button.targetGraphic.color = Color.Lerp(original_color, button.colors.
normalColor, t);
                yield return null;
            }
            button.targetGraphic.color = button.colors.normalColor;
        }
    }
```

ДОДАТОК Б. Слайди мультимедійної презентації

Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity

ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА ГРУПИ 4КГ-08: ГРАБОВСЬКОЇ ДАРІЇ СЕРГІЇВНІ

КЕРІВНИК: СКОРНЯКОВ В.С.

Аналоги на ігровому ринку

Скріншот класичної гри Сапер для операційний системі Windows XP

Скріншот ігрового процесу гри Hex Minesweeper

Скріншот ігрового процесу гри Minesweeper Mania

Програмне забезпечення для роботи

Концепція розробленої гри

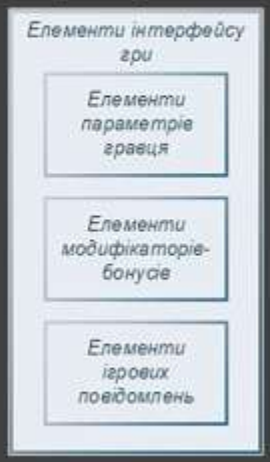
Загальна концепція гри	Загальна концепція гри повторює всі основні правила класичного саперу. Мінімалістичний дизайн, модифікації для рівнів, які вносять зміни у поле та ігровий процес
Геймплейна концепція	Гравець має виконувати розмінування ігрового поля шляхом розкриття клітинок. Якщо знайдено міну – гравець втрачає життя. Якщо міни нема, то клітинка демонструє цифру, яка вказує на кількість мін навколо. Розміщення мін випадкове
Сценарій та наратив	Гра не матиме сценарної складової, або сюжету
Візуальна складова	Класичний мінімалістичний дизайн
Цільові платформи	Цільовою платформою є персональний комп'ютер.

Структура основних елементів гри

Структура розробляємого проекту

Переліки ігрових об'єктів, елементів інтерфейсу та програмних модулів гри

Перелік видів ігрових об'єктів для розробляємої гри



Перелік елементів інтерфейсу для розробляємої гри

Перелік програмних модулів для реалізації у розробляємій грі

Структура модифікацій рівнів гри

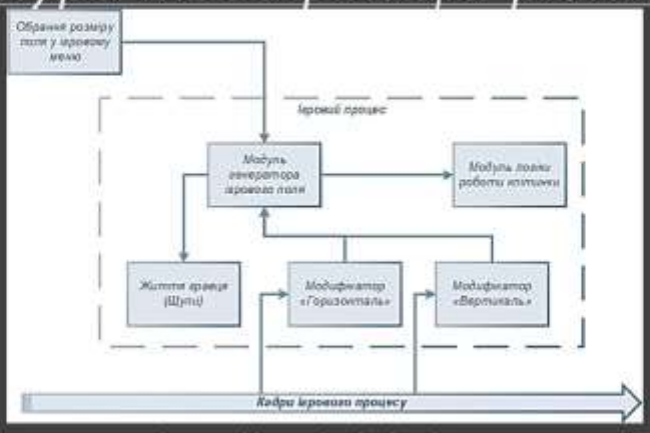


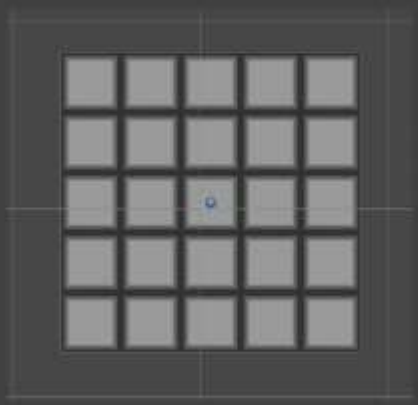
Схема взаємодії модифікаторів рівнів з іншими модулями у ігровому процесі розробляємої гри

Реалізація графічного матеріалу

Скріншот спрайтів створених для розробляємої гри

Каталог спрайтів у середі розробки Unity

Створення ігрового поля



Реалізоване ігрове поле 5 на 5 клітинок за допомогою кнопок

Часткова структура ігрового поля 5 на 5 клітинок

Реалізація натискання на кнопку

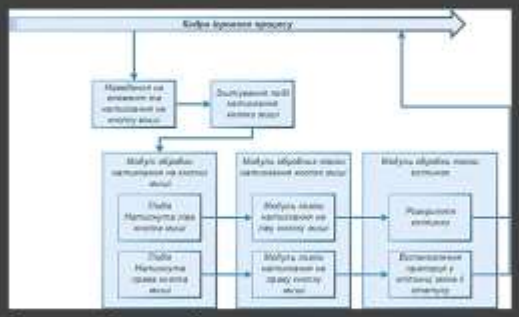
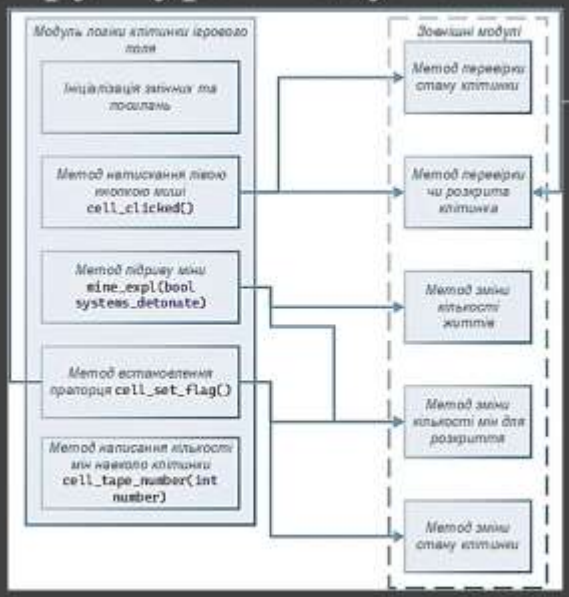


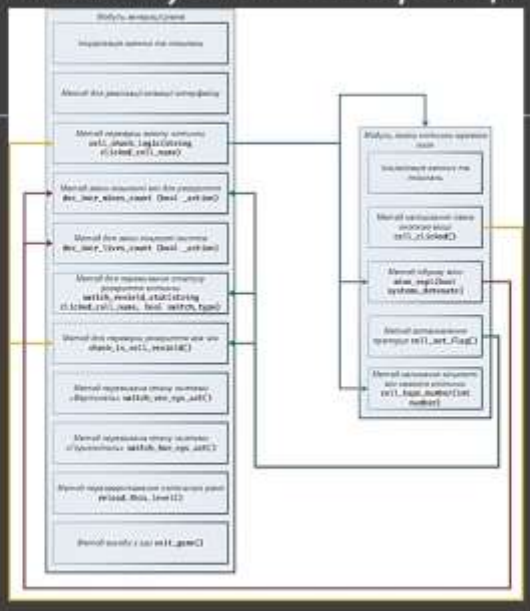
Схема роботи логіки натискання на кнопки миші

Структура Модулю обробки натискання на праву кнопку миші

Структура Модулю логіки клітинки ігрового поля



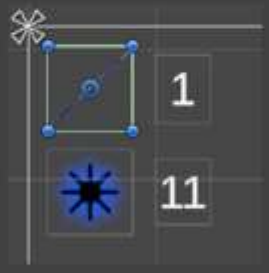
Реалізація модуля генерації рівнів гри



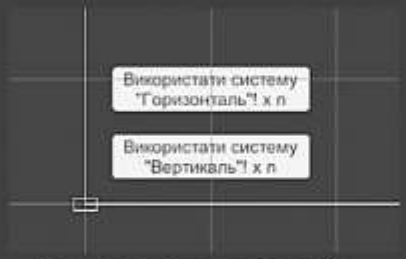
Схематична структура Модуля генерації рівнів та його зв'язки

Приклад виконання перевірки клітинок та обмеженості кількості перевірок у деяких випадках

Реалізація графіки для модифікацій рівнів

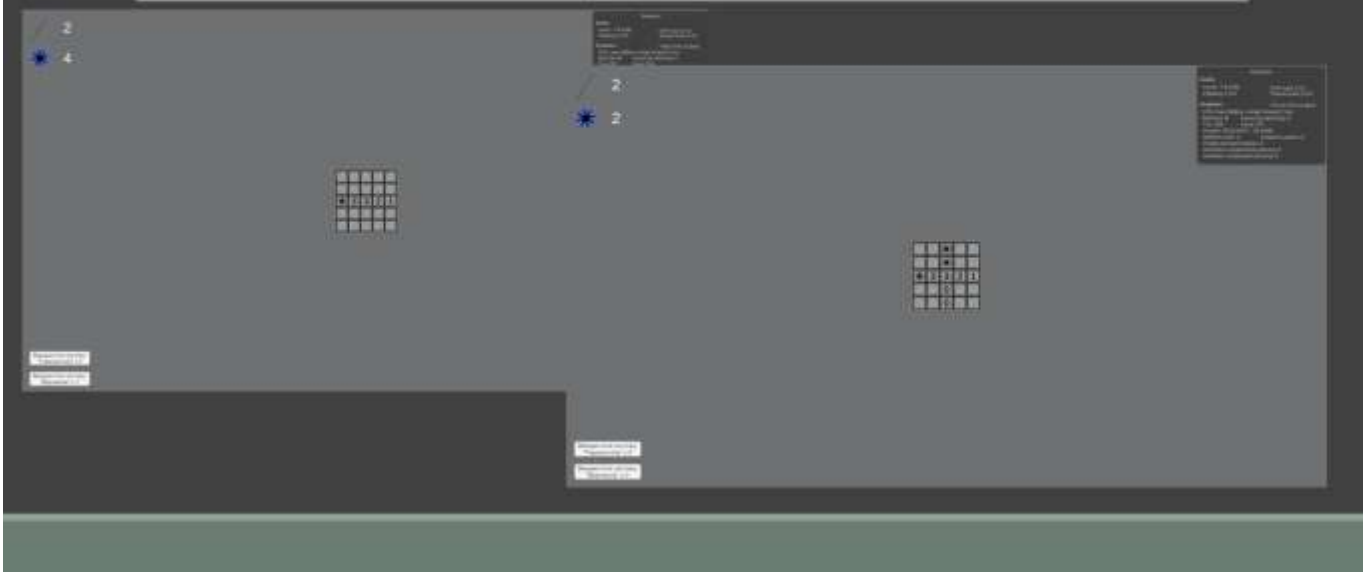


Розміщені елементи життів гравця та кількості мін для розкриття

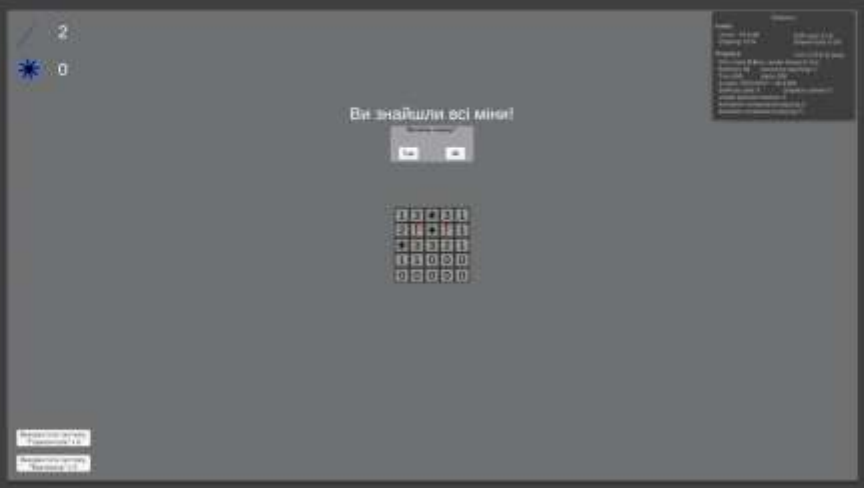


Створені кнопки для бонусів

Тестування та скріншоти розробленої гри



Тестування та скріншоти розробленої гри



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Грабовська Дарія Сергіївна

(прізвище, ім`я та по батькові)

Спеціальність *123 " Комп'ютерна інженерія "*

Освітньо-професійна програма *«Комп'ютерна графіка і Web-дизайн»*

Керівник дипломного проекту (роботи) *Скорняков Вячеслав Сергійович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity*

Обсяг розрахунково-пояснювальної записки *83* сторінок

Обсяг графічної (презентаційної) частини *16* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням створення ігрових продуктів в цілому та розробці ігор на ігровому програмному рушії Unity зокрема.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У основному розділі розглянуті питання проблематики розробки гри на ігровому програмному рушії Unity, сформовано концепцію гри згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів розробляємої гри. За допомогою відповідного програмного забезпечення реалізовані всі намічені роботи з ігровим процесом.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) *Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено*

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально описано процес виконання розробки гри;

2. Виконано проектування елементів гри із поясненнями на схемах та за допомогою коду;

3. Використано незвичайний підхід до ігрового інтерфейсу.

д) основні недоліки дипломного проекту (роботи)

Масивні блоки "if" для підрахунку мін довкола клітинки – дублювання коду, слабка масштабованість (для полів > 5×5 доведеться переписувати). Жорстко закодовані розміри поля (5×5) і пороги випадковості (r > 50) не дають гнучкості в налаштуванні й портуванні на інші розміри. Відсутність обробки винятків і перевірки коректності даних.

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

« 22 » 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проєкт здобувача (здобувачки) освіти

відділення комп'ютерних систем

Грабовська Дарія Сергіївна

(прізвище, ім`я та по батькові)

Спеціальність: *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма: *«Комп'ютерна графіка і Web-дизайн»*

Тема дипломного проєкту: *Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ

а) обсяг і якість виконання проєкту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проєкт виконано відповідно технічному завданню. Пояснювальна записка містить 83 сторінки. У пояснювальній записці виконано опис етапів розробки комп'ютерної гри для ОС Windows, виконано аналіз аналогічних продуктів, виконано концептуалізацію та проектування гри. Графічна частина складається з 16 слайдів мультимедійної презентації, які також містять графічні матеріали, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини відмінна, розробку виконано в повному обсязі.*

б) самостійність роботи над проєктом: *Протягом всього строку дипломного проєктування та переддипломної практики здобувачка освіти Грабовська Д.С. поступово та послідовно виконувала всі етапи розробки. Всі роботи здобувачка освіти виконувала самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувачка освіти Грабовська Д.С. під час роботи над дипломним проєктом вивчила достатню кількість літературних джерел та матеріалів за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і вона готова до захисту дипломного проєкту

г) вміння розв'язувати виробничі та конструкторські питання _____

Під час дипломного проєктування здобувачка освіти Грабовська Д.С. мала змогу самостійно приймати рішення з реалізації елементів і модулів гри, та показала вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проєкту

Скорняков Вячеслав Сергійович

Місце роботи і посада керівника дипломного проєкту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії

Підпис _____

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Грабовська Д.С.
здобувач освіти гр. 4КГ-08, та

Скорняков В.С.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity» (автор роботи – Грабовська Д.С., керівник роботи – Скорняков В.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Грабовська Д.С. /

Керівник _____ / Скорняков В.С. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КГ-08

Грабовської Дарії Сергіївни

на тему *Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity*

Висновок відповідальної особи за проведення нормоконтролю:

пояснювальна записка до дипломного проєкту виконана з некритичними порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 15.06.2025 р. значення коефіцієнту подібності в роботі становить 8,59%, коефіцієнт цитування – 1,12%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Грабовської Д.С.* (П.І.Б.)

проведена « *16* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/15/2025
Дата редагування 6/15/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка гри "Сапер" з модифікацією рівнів на ігровому рушії Unity

Автор
Грабовська Дарія Сергіївна
Науковий керівник / Експерт
Скорняков В'ячеслав Сергійович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

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

КП 1

КЦ

25	14511	112471
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

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

Заміна букв	ß	15
Інтервали	A⟶	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	55

Подібності за списком джерел

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

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	83 0.57 %
2	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	75 0.52 %
3	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	64 0.44 %
4	https://card-file.ontu.edu.ua/bitstreams/341a820e-d025-42f3-b7dc-27e831d6c66f/download	43 0.30 %
5	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	40 0.28 %

6	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	34 0.23 %
7	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	32 0.22 %
8	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	31 0.21 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	29 0.20 %
10	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	28 0.19 %

з домашньої бази даних (0.36 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	21 (2) 0.14 %
2	Розробка веб-застосунку інтелектуального пошуку та сумісного перегляду аніме-контенту [redacted] 6/14/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	16 (2) 0.11 %
3	Розробка web-застосунку для генерації повідомлень із використанням технологій штучного інтелекту [redacted] 6/14/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	15 (2) 0.10 %

з програми обміну базами даних (0.21 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка технологічного процесу плазмового наплавлення бандажної полки лопатки авіадвигуна [redacted] 6/9/2023 National University "Zaporizhzhia Polytechnic" (Кафедра "Інтегровані технології зварювання та моделювання конструкцій")	12 (1) 0.08 %
2	Методичні рекомендації.docx [redacted] 1/12/2023 Zhytomyr Ivan Franko State University (ZIFSU)	10 (1) 0.07 %
3	Диплом Доскач АИ-162.docx [redacted] 6/16/2020 Odessa National Polytechnic University (ІКС, кафедра інформ. систем)	8 (1) 0.06 %

з Інтернету (8.02 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**205 (7) 1.41 %**
2	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**198 (7) 1.36 %**
3	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**131 (4) 0.90 %**

4	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**96 (6) 0.66 %**
5	**https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download**	**62 (4) 0.43 %**
6	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	58 (5) 0.40 %
7	**https://card-file.ontu.edu.ua/bitstreams/341a820e-d025-42f3-b7dc-27e831d6c66f/download**	**43 (1) 0.30 %**
8	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	42 (2) 0.29 %
9	**https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download**	**31 (1) 0.21 %**
10	https://duikt.edu.ua/repozitorii/ipz/2024/%D0%9F%D0%9F%D0%97-51/%D0%9F%D0%9F%D0%97-51%20%D0%91%D1%94%D0%BB%D0%B8%D0%B9%20%D0%A0.%D0%94..pdf	30 (5) 0.21 %
11	https://stackoverflow.com/questions/57984983/detect-button-press-with-the-first-finger-but-not-if-pressed-with-more-fingers-u	27 (3) 0.19 %
12	**http://www.matmodel.puet.edu.ua/files/lic2020/rp-ossp-24.pdf**	**25 (1) 0.17 %**
13	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	22 (2) 0.15 %
14	https://ekt.elit.sumdu.edu.ua/wp-content/uploads/2023/09/Navchalnyj-posibnyk.pdf	22 (2) 0.15 %
15	https://knute.edu.ua/file/NjI4Nw==/354eadce03a1a6cfc898e26acceeea0d.pdf	18 (1) 0.12 %
16	https://card-file.ontu.edu.ua/server/api/core/bitstreams/21ac499a-a9e9-4137-810c-5f21a0318048/content	16 (1) 0.11 %
17	https://stackoverflow.com/questions/58660717/unity-jumping-code-for-player-not-working	16 (2) 0.11 %
18	https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download	15 (1) 0.10 %
19	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	14 (1) 0.10 %
20	https://dspace.znu.edu.ua/jspui/bitstream/12345/12515/1/Yatsun_O_O.pdf	14 (2) 0.10 %
21	https://card-file.ontu.edu.ua/bitstreams/9908b7a9-6b3e-46f5-a46e-84d83787cfd4/download	14 (1) 0.10 %
22	https://gist.github.com/misho-kr/be0cf68a9d1f5f0ffcf252dabbc7ca6d	14 (1) 0.10 %
23	https://ami.lnu.edu.ua/wp-content/uploads/2023/01/Mahisterska-robota-Mychka-2022.pdf	13 (1) 0.09 %
24	https://card-file.ontu.edu.ua/server/api/core/bitstreams/4bb7255e-46d4-4349-9726-9698476da02d/content	12 (1) 0.08 %
25	https://zakon.help/article/novi-vimogi-roboti-za-kompyuterami-roboti-za/	10 (1) 0.07 %
26	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	6 (1) 0.04 %
27	https://github.com/EpsilonD3lta/UnityUtilities/blob/master/Scripts/Runtime/TMProUGUIHyperlinks.cs	5 (1) 0.03 %
28	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	5 (1) 0.03 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн» Група: 4КГ- 08