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

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма: «Розробка програмного забезпечення»

Група: 4РП-08

Дипломний проєкт

здобувача освіти денної форми навчання
РП.08.16.000.ДП

ПЕТРОВЦЯ
ВІКТОРА ЛЕКСАНДРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «*Інженерія програмного забезпечення*»**

Освітньо-професійна програма: «**Розробка програмного забезпечення**»

Група: **4РП-08**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка гри-головоломки WordLines з вибіркою слів із власного словника

Проєктний матеріал складається з пояснювальної записки на 81 сторінках та графічного (презентаційного) матеріалу на 18 аркушах (слайдах)

Дипломник _____ (Петровець В.О.)

Керівник _____ (Джабраілов Д.В.)

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 «30» 06 2025 р. Протокол ЕК № 3

Оцінка ЕК _____

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітньо-професійна програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"12" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Петровцю Віктору Олександровичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка гри-головоломки WordLines з вибіркою слів із власного словника*

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 бібліотек Unity для розробки гри; Розробка основних механік для гри слова-лінії; Реалізація вибірки різних слів для ігрового процесу; Реалізація основних елементів управління; Реалізація ігрового інтерфейсу*

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

Аналіз ігрового процесу аналогічних проектів; Вибір програмного забезпечення для реалізації проекту; Розробка концепції ігрового процесу для гри; Проектування структури гри та її основних елементів; Проектування системи вибірки слів із словника; Реалізація основних ігрових механік; Реалізація вибірки слів із словника; Реалізація інтерфейсу; Тестування працездатності розробленої гри

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

Аналіз аналогічних ігор; Концепція гри WordLines; Структура розробляємого проекту; Огляд основних елементів гри; Реалізація основних елементів гри; Огляд вибірки слів із власного словника; Реалізація вибірки слів із власного словника; Реалізація інтерфейсу гри; Тестування гри; Скріншоти гри 1; Скріншоти гри 2

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Джабраілов Д.В.*		
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 *Джабраілов Д.В.* _____
(підпис)

Завдання прийняв до виконання *Петровець В.О.* _____
(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*	
2	*Аналіз аналогічних ігрових рішень*	*16.05.2025*	
3	*Розробка концепції ігрового процесу*	*17.05.2025*	
4	*Проектування основних елементів гри*	*20.05.2025*	
5	*Проектування вибірки слів з власного словника*	*22.05.2025*	
6	*Реалізація основних ігрових механік*	*28.05.2025*	
7	*Реалізація вибірки слів із власного словника*	*01.06.2025*	
8	*Реалізація інтерфейсу та графічної частини*	*03.06.2025*	
9	*Відлагодження модулів гри*	*06.06.2025*	
10	*Тестування працездатності елементів гри*	*10.06.2025*	
11	*Виправлення виявлених помилок*	*11.06.2025*	
12	*Аналіз результатів, підготовка слайдів презентації*	*12.06.2025*	
13	*Економічні розрахунки та питання з охорони праці*	*13.06.2025*	
14	*Підготовка графічної частини проекту*	*14.06.2025*	
15	*Підготовка проекту до захисту та тестування гри*	*16.06.2025*	

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 08. 16 000. 00 ДП	Дипломний проєкт		
А4			РП 08. 16 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 08. 16 000. 00 ДП		
Розробив		Петровець В.О.		16.02	Розробка гри-головоломки WordLines з вибіркою слів із власного словника	Літ.	Аркуш 4 / Аркушів 81
Перевірив		Джабраілов Д.В.		16.02		НДП	
Н. контр.		Петрашова В.І.		16.06.15		ВСП "ОТФК ОНТУ" гр.4РП-08	
Затверд.		Кривченко Ю.В.		16.06			

ЗМІСТ

Вступ 7

1 Основний розділ 8

1.1 Аналіз ігрового процесу аналогічних проектів 8

1.1.1 Огляд ігрового жанру філворди 8

1.1.2 Аналіз ігрового процесу гри Fill The Words: Themes search 8

1.1.3 Аналіз ігрового процесу гри Філворди: Гра в Слова з Букв 10

1.1.4 Аналіз ігрового процесу веб-додатку thewordsearch.com 11

1.1.5 Результати аналізу ігрового процесу аналогічних проектів 12

1.2 Вибір програмного забезпечення для реалізації проекту 12

1.2.1 Вибір ігрового програмного рушія для розробки проекту 12

1.2.2 Вибір іншого програмного забезпечення для розробки проекту 14

1.3 Розробка концепції ігрового процесу для гри 15

1.3.1 Розробка загального ігрового концепту 15

1.3.2 Розробка концепту власного словника 18

1.4 Проектування структури гри та її основних елементів 18

1.4.1 Проектування структури гри 18

1.4.2 Проектування основних елементів гри 21

1.5 Проектування системи вибірки слів із словника 25

1.6 Реалізація основних ігрових механік 27

1.6.1 Створення проекту та його налаштування 27

1.6.2 Робота з графікою 30

1.6.3 Створення ігрових об’єктів та префабів 32

1.6.4 Реалізація модулів основних ігрових механік 37

1.7 Реалізація вибірки слів із словника 45

1.8 Реалізація інтерфейсу 48

1.9 Тестування працездатності розробленої гри 49

2 Економічний розділ 53
2.1 Резюме 53
2.2 Визначення трудомісткості розробки програмного забезпечення 53
2.3 Розрахунок ціни програмного продукту 56
3 Розділ охорони праці та техніки безпеки 58
3.1 Аналіз шкідливих та ризикових факторів 58
3.2 Гігієнічні вимоги до виробничого середовища 58
3.3 Вимоги до організації робочого місця працівника 59
3.4 Електробезпека 60
3.5 Пожежна безпека 61
Висновки 63
Перелік використаних інформаційних джерел 64
Додаток А. Лістинг коду основних модулів гри мовою C# 65
Додаток Б. Слайди мультимедійної презентації 73

ВСТУП

Світова ігрова індустрія вже довгий час розвивається поширюючись між ігровими платформами, формами гри, технологіями гри та напрямом ігрового процесу. Сучасні ігри вже давно відійшли від формату іграшок для дітей та стали загально доступним видом дозвілля завдяки смартфонам та мобільним ігровим платформам, як Switch або Steam Deck.

Разом з тим робота в ігровій індустрії стає більш доступною та більш затребуваною. Для розробки великих ігрових проектів поєднуються зусилля офісів по всьому світу. Та навпаки, сучасні технології дозволяють створювати ігри невеличким командам, або навіть одній людині.

Темою цього дипломного проекту є «Розробка гри-головоломки WordLines з вибіркою слів із власного словника». Ця гра є варіацією кросвордів, які всі дуже добре знають. Інтуїтивно зрозумілий ігровий процес дозволяє іграм такого типу отримувати велику аудиторію та ставати дуже популярними.

В даному дипломному проекті передбачається створити гру в жанрі словникова головоломка для персонального комп'ютера в яку зможуть грати на системах будь-якого рівня потужності та реалізувати власний словник для слів, які будуть використовуватись у ігровому процесі. Планується реалізувати генерацію кросвордів, вибірку слів та зрозумілий візуальний інтерфейс. У візуальній реалізації мають бути використані елементи кольорової ідентифікації знайдених слів.

Розроблятись проект має на сучасному ігровому програмному рушії Unity який дозволить у майбутньому реалізувати мультиплатформну основу для проекту, що розробляється. Для створення та редагування коду використовувати середу розробки Microsoft Visual Studio. Окрім того буде використано ресурси з Asset Store ігрового рушія Unity.

Гра буде розроблятись з використанням модульної системи, що значно полегшить підтримку гри у майбутньому, а також дасть змогу відносно легко модифікувати ігровий процес, або створювати нові ігрові механіки.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз ігрового процесу аналогічних проектів

1.1.1 Огляд ігрового жанру філворди

Темою даного дипломного проекту є «Розробка гри-головоломки WordLines з вибіркою слів із власного словника». Під час будь-якої розробки, необхідно виконати аналіз питання, аналогів та зробити синтез отриманих даних для виконання поставленої мети більш ефективним чином.

Ігровий жанр філворди – це ігровий жанр, який бере свою основу з варіації кросвордів, в яких потрібно з таблиці букв створювати визначені, або невідомі гравцю слова. Такі ігри широко представлені у тематичній пресі з наборами різних кросвордів, а також у виді комп'ютерних проектів та додатків на смартфони.

Філворди є цікавими у розв'язанні та досить легкими у вивченні правил гри. Гравцю дають ігрове поле визначеного розміру. На ньому розміщуються слова записані по літерам у клітинки, запис виконується не тільки в лінію, а будь-якою формою, крім діагоналі. У філвордах не може бути зайвих букв, тому невикористаних клітинок з буквами не може бути.

Якщо казати про ігри жанру філворди на комп'ютери або смартфони, то вони бувають виконані у різних варіантах, але основною ігровою механікою залишається поєднання букв у слова. В свою чергу реалізація такого жанру на мультимедійних пристроях дають змогу додавати до ігрового процесу додаткові види дій, як підказки, використання різних кольорів замальовування та інше. Для подальшої реалізації проекту потрібно виконати аналіз ігрового процесу аналогічних проектів у ігровій індустрії.

1.1.2 Аналіз ігрового процесу гри Fill The Words: Themes search

Гра Fill The Words: Themes search розроблена для смартфонів та представляє собою класичну гру в жанрі філворд, але головною рисою цієї гри є розподілення кросвордів на темі – тобто слова у таблиці поділяються тематичні групи і якщо обрано групу їжа, то гравцю будуть потрапляти слова лише пов'язані з їжею. Ігровий процес складається з проведення пальцем по буквам, які гравець вважає

складають якесь слово. Якщо літери складені вірно, то слово зараховується та зберігається на ігровому полі. Слова відмічаються на ігровому полі різними кольорами. Важливою складовою є виділення букв у правильній послідовності. На рисунку 1.1 зображено скріншот ігрового процесу розглядаємої гри.

Рисунок 1.1. Скріншот ігрового процесу гри Fill The Words: Themes search

Аналізуємий проект представляє собою приклад творчої реалізації класичних філвордів, які не вимагають від гравця додаткових зусиль окрім тих, які продиктовані ігровим жанром.

1.1.3 Аналіз ігрового процесу гри Філворди: Гра в Слова з Букв

Проект Філворди: Гра в Слова з Букв представляє собою таку ж класичну варіацію гри у філворди, як і попередній розглядаємий проект. Тут так само потрібно обирати слова поєднуючи букви у лінії за допомогою тачскріну. Гра доступна як на смартфоні, так і на персональному комп'ютері, де букви потрібно обирати за допомогою курсора миші та натискання на ліву кнопку миші. На рисунку 1.2 зображено приклад ігрового процесу гри Філворди: Гра в Слова з Букв.

Рисунок 1.2. Скріншот ігрового процесу гри Філворди: Гра в Слова з Букв

З особливостей можна виділити прогресію рівнів складності, в залежності від кількості розв'язаних гравцем кросвордів. Чим вище рівень складності, тим більше потрібно знайти слів, а отже більше ігрове поле. Так само як і у попередньому проекті, обрані слова виділяються різними кольорами для

зручності гравця, що полегшує ігровий процес та сприйняття ігрового поля користувачем.

1.1.4 Аналіз ігрового процесу веб-додатку thewordsearch.com

Останнім проектом для розгляду є thewordsearch.com. Це веб-додаток, який доступний за вказаною адресою. Він завантажується через будь-який веб-браузер та дає змогу обрати тему кросворду та його розмір з вже заздалегіть створених кросвордів. На рисунку 1.3 можна переглянути скріншот ігрового процесу цієї гри.

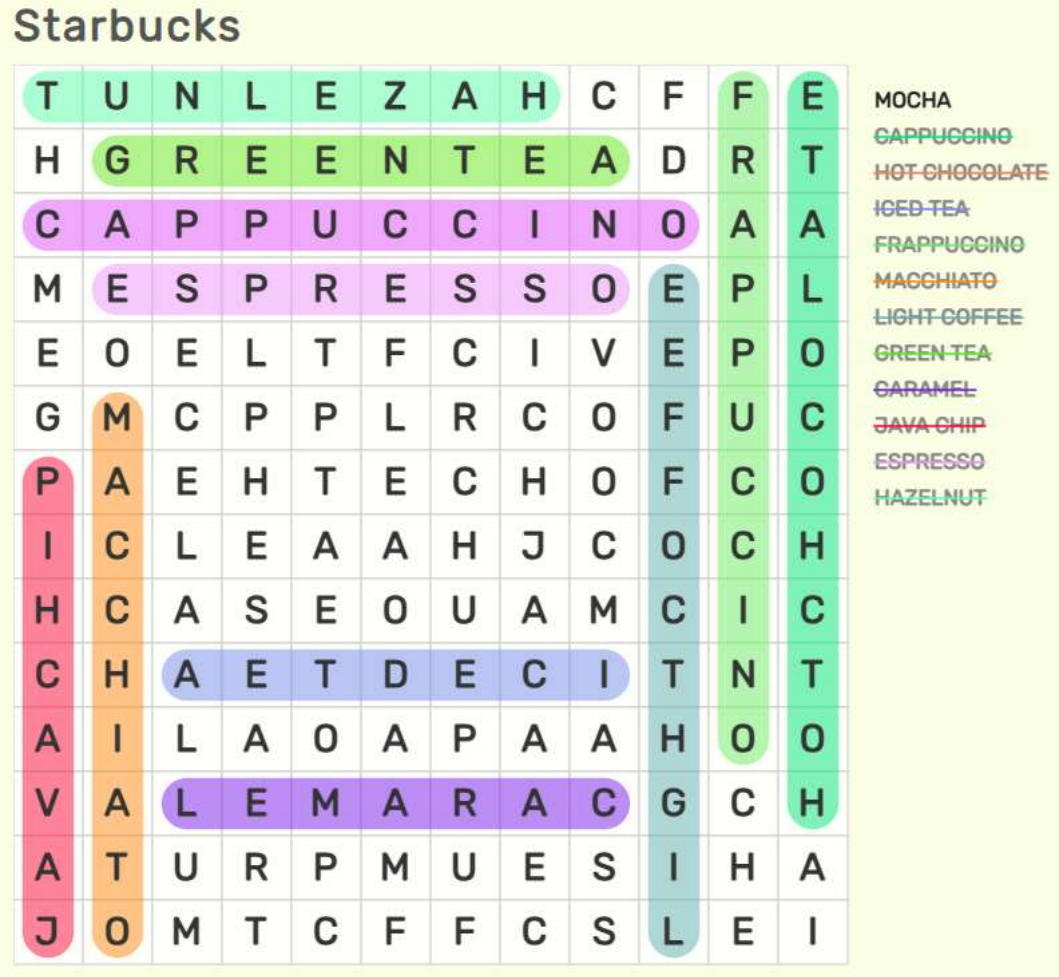


Рисунок 1.3. Скріншот ігрового процесу веб-додатку thewordsearch.com

На відміну від розглянутих раніше проектів, у цьому додатку можна обирати букві лише по горизонталі, вертикалі та діагоналі за допомогою затискання лівої кнопки миші. Окрім того на полі можуть бути букви які не утворюють слова та виконують роль наповнювача ігрового поля. Також, слова можуть утворювати перетини між собою, що ускладнює ігровий процес. На полі

можуть бути не лише слова але й словосполучення, де пробіл між словами не враховується при пошуку на ігровому полі. Тобто словосполучення ICED TEA на ігровому полі буде мати вигляд ICEDTEA. Після гри можна переглянути свій хід розв'язання кросворду, а також переглянути світові рекорди по розв'язанню саме цього кросворду.

1.1.5 Результати аналізу ігрового процесу аналогічних проектів

За результатами аналізу ігрового процесу аналогічних проектів можна побачити, що у більшості варіантів ігор в жанрі філворд правила майже однакові за деякими виключеннями, які є відмінними від класичного варіанту філвордів. Також можна побачити що всі розглянуті проекти використовують різні кольорові схеми для полегшення ігрового досвіду для користувача. З іншого боку розглянуті проекти не реалізовують генерацію слів, а демонструють заздалегідь створені розробниками рівні. Частіше за все в таких іграх намагаються поділяти кросворди на теми слів, але є варіанти де слова не поділені за темами.

1.2 Вибір програмного забезпечення для реалізації проекту

1.2.1 Вибір ігрового програмного рушія для розробки проекту

Тема цього дипломного проекту стосується створення відеогри, що потребує вибору відповідного ігрового програмного рушія. На сьогоднішній день існує кілька популярних рушіїв, які мають безкоштовні ліцензії, проте кожен з них відрізняється за функціональністю, принципами розробки та доступними інструментами. Перед тим як прийняти остаточне рішення, було проведено порівняльний аналіз наявних варіантів.

Одним із розглянутих варіантів став Unreal Engine – сучасний високопродуктивний рушій, розроблений компанією Epic Games. Він активно застосовується в ігровій індустрії, а також у сферах інтерактивної візуалізації, кіновиробництва, VR/AR. Unreal підтримує передову графіку, фізику, анімацію та має інструмент візуального програмування Blueprints, що дозволяє створювати логіку гри без глибоких знань мов програмування. Цей рушій використовують як незалежні розробники, так і великі студії. Прикладами успішних проектів на базі

Unreal Engine є Fortnite, серія Borderlands, Hellblade: Senua's Sacrifice тощо, що демонструє гнучкість рушія у плані реалізації різних стилістик.

Серед технологічних особливостей Unreal Engine варто відзначити систему Nanite – механізм роботи з віртуальною геометрією, який дозволяє використовувати детальні 3D-моделі без втрати продуктивності. Інша важлива система – Lumen, що відповідає за глобальне освітлення та реалістичні відбиття у реальному часі. Також варто відмітити World Partition – функціонал для роботи з великими відкритими світами, який забезпечує автоматичне зонування території та динамічне завантаження потрібних ділянок.

Іншим проаналізованим рушієм був Unity – кросплатформна технологія розробки інтерактивних продуктів, яка підтримує створення 2D, 3D, VR та AR проектів для широкого спектру платформ: від настільних систем і мобільних пристроїв до ігрових консолей і веббраузерів. Цей рушій розроблено компанією Unity Technologies. Його ключовими перевагами є зручність у використанні, потужна база документації та велика спільнота користувачів. Завдяки цьому рушій ефективно застосовується як у незалежній розробці, так і у великих комерційних проектах у сфері геймдеву, архітектурної візуалізації, навчальних симуляцій і навіть у кіновиробництві. Серед відомих ігор, створених у Unity, варто назвати Hollow Knight, Ori and the Blind Forest, Cuphead.

Суттєвою технічною особливістю Unity є його об'єктно-орієнтований підхід до побудови ігрової сцени: базовим структурним елементом є GameObject, до якого додаються різноманітні компоненти – скрипти, колайдери, рендери, фізичні модулі тощо. Така компонентна архітектура дозволяє гнучко комбінувати елементи для реалізації складної поведінки. Unity використовує мову C#, яка має розвинену екосистему, широку підтримку бібліотек і є безпечною з точки зору доступу до ресурсів систем.

Для реалізації дипломного проекту остаточний вибір було зроблено на користь Unity. Це рішення прийнято з огляду на доступність матеріалів для навчання та підтримки, що значно спрощує розв'язання типових проблем у процесі розробки. Окрім того, аналіз прикладів продемонстрував, що Unreal

Engine краще підходить для створення масштабних AAA-проектів, тоді як Unity ідеально підходить для реалізації інді-ігор і проектів невеликого масштабу.

1.2.2 Вибір іншого програмного забезпечення для розробки проекту

У процесі створення гри, окрім вибору рушія, необхідно визначитись і з додатковими інструментами, які забезпечують повноцінний цикл розробки. Одним з ключових елементів цього середовища є система для написання програмного коду, оскільки жоден ігровий проект не обходиться без реалізації логіки.

Мова програмування, що буде використовуватись у проекті, визначається специфікою обраного ігрового програмного рушія – у випадку Unity це C#. Відповідно, потрібно середовище розробки, яке підтримує цю мову та забезпечує стабільну інтеграцію з рушієм. Згідно з рекомендаціями документації Unity, оптимальним вибором є Visual Studio – професійна інтегрована розробницька платформа (IDE), створена компанією Microsoft.

Visual Studio дозволяє не лише писати код, а й проводити налагодження, юніт-тестування, компіляцію та розгортання проектів. Ця IDE підтримує широкий спектр мов програмування, включно з C#, C++, Visual Basic, JavaScript, Python і TypeScript, та має повну інтеграцію з .NET Framework. До переваг середовища належать розширені можливості автодоповнення, кольорове виділення синтаксису, підтримка плагінів і розширень, що значно підвищують ефективність розробки.

Окрім програмної частини, важливою складовою проекту є візуальна складова. Оскільки у розроблюваній грі планується реалізувати 2D-графіку, виникає необхідність у використанні відповідного графічного редактора. Серед великої кількості програм для роботи з растровими зображеннями, з огляду на функціональність, зручність інтерфейсу та наявний досвід роботи, було обрано Adobe Photoshop.

Adobe Photoshop – це професійне програмне забезпечення для обробки зображень, розроблене компанією Adobe Inc. Воно широко використовується у різних галузях: від фотографії та ілюстрації до дизайну інтерфейсів і підготовки

друкованих матеріалів. В ігровій індустрію ця програма використовуються у багатьох великих студіях розробки комп'ютерних ігор. У контексті розробки гри Photoshop забезпечує інструменти для створення та редагування спрайтів, фонових зображень, текстур, а також дозволяє здійснювати корекцію кольору, підготовку графіки до імпорту в ігровий програмний рушій, та навіть базову анімацію.

1.3 Розробка концепції ігрового процесу для гри

1.3.1 Розробка загального ігрового концепту

На початковій стадії створення ігрового проекту першим кроком зазвичай є підготовка концепт-документу. Це первинний документ, який виконує роль базової інструкції або орієнтиру, подібно до дизайн-документу, але не має остаточного характеру. Його зміст може змінюватися, доповнюватися або уточнюватись в залежності від вимог, що виникають у процесі розробки.

Метою концепт-документу є формулювання основної ідеї проекту, визначення його жанрової спрямованості, основ геймплею, візуального стилю та технічного підходу. Цей документ допомагає команді сформувати загальне уявлення про майбутню гру і виступає як фундамент для подальшого планування.

Зазвичай у рамках концепт-документу описуються наступні ключові елементи:

- Базова ідея – короткий опис ігрового сетингу, загального настрою, сюжетного підґрунтя та цілей, що стоять перед гравцем;
- Ігрові механіки – визначення принципів взаємодії гравця з середовищем: управління, системи бою, розвиток персонажа, фізика світу, унікальні механіки;
- Сюжетна структура та персонажі – опис основного наративу, подій, які визначають хід гри, а також характерів, ролей і мотивацій головних та другорядних героїв;
- Графіка та звук – загальний підхід до візуального стилю: тип графіки (2D/3D), колористика, стилізація, типи анімацій та загальні вимоги до

музичного й звукового супроводу;

- Конструкція ігрового світу – короткий опис світу гри: історія, структура суспільства, технологічний рівень, культурні особливості;
- Технічна реалізація – визначення технічних параметрів: обраний рушій, підтримувані платформи, мови програмування, інструменти та програмне забезпечення, необхідне для розробки.

Концепт-документ є інструментом внутрішнього узгодження бачення проекту. Його основна функція – забезпечити спільне розуміння ключових аспектів гри всередині команди, підвищити ефективність планування, уникнути суперечностей у підходах і своєчасно виявляти питання, що потребують доопрацювання або перегляду.

Для того щоби виконати проробку концепту розробляємого проекту, раніше було зроблено аналіз ігрового процесу аналогічних проектів. Це дозволило отримати актуальні дані щодо того, як наповнюють подібні ігри в жанрі філворд, їх інтерфейсу, ігрового процесу та іншим аспектам. Згідно з визначення концепції було виконано викладку щодо концепції майбутньої гри:

- Базова ідея проекту складається у реалізації гри в жанрі філворд з додаванням для гравця можливості реіграбельності, створенню умов для значної кількості ігрових сесій;
- Ігровими механіками є класичні правила ігор в жанрі філворд. Гравець має поєднувати букви на ігровому полі у лінії різних форм щоби отримати слова. Управління виконується за допомогою миші;
- Гра не буде мати сюжетної структури або персонажів на даному етапі розробки. Можливо, у майбутньому можна імплементувати цю складову для більшої зацікавленості у грі;
- Графіка представлена у простому 2D виконанні із використанням унікальних скайбоксів. Під час ігрового процесу виділені букви будуть виділятись в один колір створюючи різноманітні кольорові схеми;
- Цільовою платформою для розробляємого проекту є персональний комп’ютер, але використання ігрового програмного рушія Unity дає

змогу у майбутньому виконати портування проекту на інші ігрові платформи.

Коли було сформовано загальний концепт ігрового процесу розробляємого проекту, потрібно описати концепцію власного словника. На рисунку 1.4 можна побачити концептуально проілюстрований ігровий процес розробляємої гри. Як можна побачити з рисунку 1.4, букви будуть розміщені у клітинках ігрового поля. Гравець повинен буде відмічати їх одну за одною за допомогою лівої кнопки миші та курсора. Кожне нове слово повинно підкреслюватись новим кольором. Також можна відмітити, що букви слів можуть бути записані як у горизонталь та вертикаль, так і у різних послідовностях, створюючи змійки зі слів. Важливим є те, що слова не повинні перетинатись, або утворювати собою діагональні лінії. Також важливо відмітити, що у кожному рівні, всі букви мають значення, тому в ігровій сесії мають бути використані всі букви та знайдені всі слова.

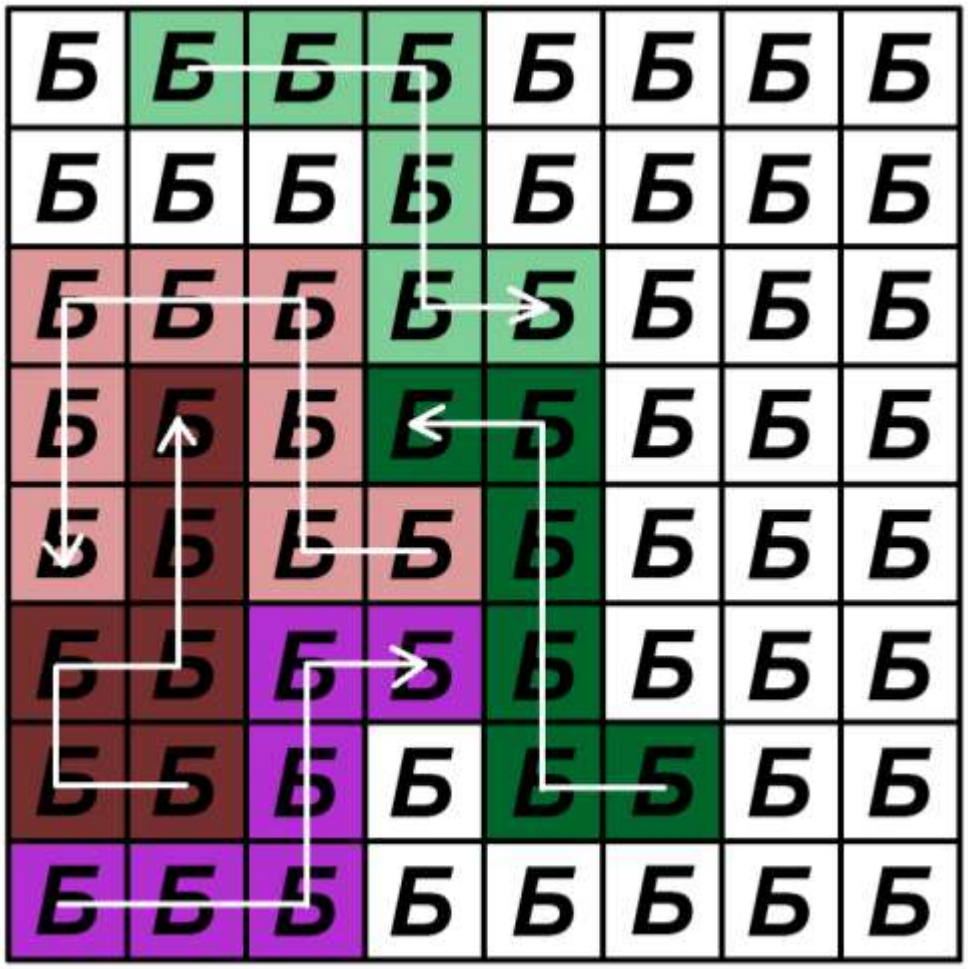


Рисунок 1.4. Концептуальне зображення ігрового процесу розробляємої гри

Останнім що потрібно відмітити, що гравець повинен мати уявлення про те, скільки слів йому потрібно знайти на ігровому полі. Для цього потрібно ввести лічильник слів. Розроблена концепція ляже в основу розробляємого проекту.

1.3.2 Розробка концепту власного словника

Одною із важливих складових розробляємого проекту є використання власного словника, що також винесено в тему дипломного проекту. Потрібно розкрити сутність цього елементу. Під власним словником розуміється Набір слів, які гра буде використовувати під час побудови рівня у кожній новій ігровій сесії. Тобто основний задум полягає у тому, щоби кожна генерація рівня створювала ігрове поле з різними словами на ньому з чого випливатиме різна послідовність букв на ігровому полі. Такий словник має бути доступний для гри під час генерації, а також він повинен бути реалізований таким чином, щоби можна було його наповнювати новими словами.

Така ігрова концепція власного словника викликає необхідність врахувати під час проектування гри можливість розробника створювати основу для генерації різних слів на ігровому полі. Це має бути реалізовано у середі розробки ігрового програмного рушія Unity. Система повинна забезпечувати розробнику простий та зрозумілий алгоритм дій для створення ігрового рівня з набором різних слів.

1.4 Проектування структури гри та її основних елементів

1.4.1 Проектування структури гри

Сформований ігровий концепт для розробляємого проекту дає змогу виконати проектування структури гри та її елементів. 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.

Одною із обов’язкових складових гри має бути модульність виконання. Це

з однієї сторони дещо ускладнює розробку, оскільки потребує розділення проекту на функційні модулі та проробки зв'язків між ними. З іншої сторони модульний підхід дозволяє полегшити модифікацію проекту у майбутньому, його супровід, а також можливість додавання нових ігрових елементів. Отже, для вдалої подальшої розробки потрібно виконати проектування структури модулів гри. На рисунку 1.5 зображено структуру розробляємого проекту.

У структурі проекту зображено основні елементи з якої складається гра та які забезпечують її функціонування, залишаючи поза дужками ігровий програмний рушій Unity. Він не розглядається як частина структури розробляємої гри, оскільки його будова є сталою і внесення змін до складових рушія не рекомендована, оскільки може призводити до критичних помилок у самій грі, або редакторі проекту. Більш досвідчені розробники, або компанії які працюють у співробітництві з Unity звісно вносять свої правки до рушія з метою посприяти на розробляєму гру, якщо її ігровий процес важко реалізуємий у поточній версії ігрового програмного рушія. У зображеній на рисунку 1.5 структурі потрібно відмітити розділення елементів на два типи.

Рисунок 1.5. Структура розробляємого проекту

Зовнішні ассети знаходяться у каталогах гри до та після її побудови. До побудови проекту, ассети знаходяться у різних каталогах, які мають тематичні імена та зберігають відповідні файли. Так, наприклад, шрифти знаходяться у каталозі TextMesh Pro, а сцени у каталозі Scenes. Після виконання побудови проекту, у кінцевій папці, ці каталоги будуть поєднанні у єдиний банк даних. Зовнішні ассети зберігаються окремо, оскільки вони не використовуються напряму, а через посилання на них. Наприклад один і той самий спрайт може бути використаний багато разів у різних місцях проекту, при цьому оригінальний файл спрайту у зовнішніх ассетах не буде змінений ігровим процесом.

Спрайти для гри будуть складатись зі спрайтів, які будуть створені для розробляємої гри та використані у інтерфейсі, або інших частинах гри. У поточній версії не передбачено варіативності зовнішнього вигляду рівнів, але у майбутньому цей каталог можна наповнити спрайтами які б цю візуальну варіативність забезпечити.

Система скайбоксів є каталогом для створення приємного заднього фону під час гри. В цьому каталозі зберігаються варіанти виконання скайбоксів приклади файлів для них.

Система шрифтів буде зберігати основні файли необхідні для використання сучасної системи шрифтів у Unity. Для її роботи потрібно окремо зберігати шрифти, ресурси матеріалів та спрайтів, шейдери та інші елементи які роблять нову систему шрифтів ефективною.

Другим типом є Елементи ігрового процесу. Ці елементи мають змінну природу, оскільки під час гри можуть змінюватись ззовні, змінювати структуру складових, змінювати один одного, або зовсім знищуватись. При цьому для кожної окремої сцени ці елементи можуть бути як заздалегідь створеними, так і згенеровані іншими елементами. В цілому елементи ігрового процесу у здебільшому забезпечені ігровими об’єктами та їх компонентами.

Програмні модулі гри складаються із коду, який описує поведінку ігрових об’єктів. Ці модулі називаються скриптами та є головною складовою будь-якого розробляємого проекту на ігровому програмному рушії Unity. По суті скрити

вказують на те як, коли та що робити ігровим об’єктам та їх компонентам. Програмні модулі можна віднести і до зовнішніх модулів, оскільки вони створюються у каталогах ассетів гри. До Елементів ігрового процесу вони віднесені через те, що скрипти не працюють із каталогів коду, а виконуються безпосередньо на ігровій сцені, тобто є елементами ігрового процесу.

Ігрові об’єкти гри – це будь-які об’єкти які знаходяться на сцені. Вони можуть бути як видимими, так і невидимими. Кожний ігровий об’єкт має своє ім’я, компоненти та параметри, які роблять їх унікальними відносно один одного. Ці об’єкти створюються та існують лише на сцені, так само, як і компоненти, що їх наповнюють. Ігровий об’єкти може стати ассетом, якщо він використовуються, як префаб, але тут принцип такий самий, як і зі скриптами, оскільки сам по собі префаб у каталогах гри нічого не робить у ігровому процесі, а працюють лише його екземпляри, які наповнюють сцену.

Елементи інтерфейсу гри також зберігаються лише на ігровій сцені, у ігровому процесі. Хоча вони по своїй формальній суті є ігровими об’єктами, але їх відрізняє база на якій вони відтворюються. Всі елементи інтерфейсу у ігровому програмному рушії Unity знаходяться у окремому ігровому об’єкті, який описує інтерфейс гри. Цей елемент можна налаштовувати відповідно за потребами розробника, але перемістити до ассетів гри неможливо.

1.4.2 Проектування основних елементів гри

Для подальшої розробки потрібно більш точніше описати будову та принцип роботи елементів гри. Це дасть змогу під час реалізації цих елементів гри вже розуміти структуру реалізовуємого модулю, його функційне наповнення, а також зв’язки з іншими модулями, які вже існують у проекті, або лише тільки будуть в нього внесені.

Основну увагу потрібно приділити елементам ігрового процесу, оскільки самі по собі ассети не виконують наповнення гри, вони лише є файловим наповненням проекту, яке може в собі вміщати або зображення, або звуки, чи програмний код. Натомість ігрові об’єкти, що знаходяться на сцені, а саме їх компоненти використовуючи ассети створюють ігровий процес. Іноді навіть самі

ассети стають компонентами ігрового процесу. Така метаморфоза виконується, коли додають скрипти до ігрових об’єктів, або використовують спрайти чи текстури у компонентах із відповідними полями. На рисунку 1.6 зображено види ігрових об’єктів та їх зв’язки з іншими елементами гри.

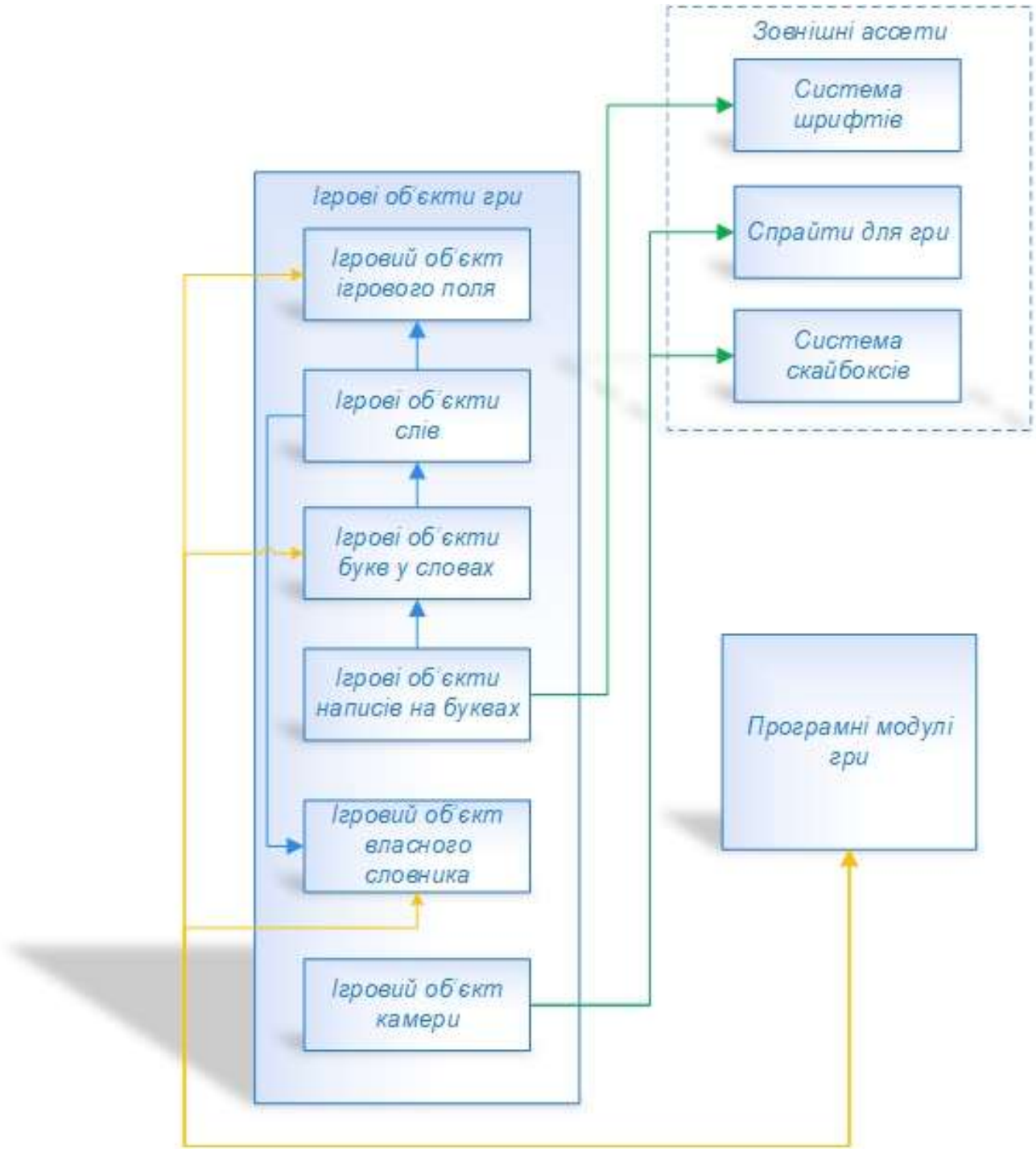


Рисунок 1.6. Види ігрових об’єктів та їх зв’язки з іншими елементами гри WordLines

Рисунок 1.6 показує яким чином пов’язані ігрові об’єкти з іншими елементами ігрового процесу. Можна побачити, що основні складові гри мають

зв’язок із програмними модулями гри, що показано жовтою стрілкою. Можна побачити, що цей зв’язок є зворотнім, тобто за допомогою цих програмних модулів виконується не тільки отримання команд та даних з скриптів гри, але й передача даних у інші ігрові об’єкти, що є дуже важливим у забезпечення ігрового процесу.

Можна також побачити зв’язок ігрових об’єктів, які відповідають за відображення контенту на екран, з елементами зовнішніх ассетів. Ігрові об’єкти букв повинні отримувати букви з Системи шрифтів, та використовувати її для відображення свого змісту. Ігровий об’єкт камери звертається до Спрайтів гри та Системи скайбоксів оскільки саме камера відповідає за відображення заднього фону для гравця.

Самі ігрові об’єкти також пов’язані одні з одними, оскільки мають складну вкладену будову. Так об’єкт ігрового поля складається із об’єктів слів, об’єкти слів складаються з об’єктів букв, а об’єкти букв у свою чергу мають у складі об’єкти написів. Така на перший погляд ієрархія ігрових об’єктів логічно обґрунтована та зрозуміла для майбутньої розробки та підтримки проекту, оскільки має схожу вкладеність із реальністю. Окремо можна відмітити, що ігрові об’єкти слів мають зв’язок з ігровим об’єктом власного словника, оскільки саме звідти об’єкти слів будуть отримувати свої вибрані слова та букви.

Програмні модулі гри є важливою складовою розробляємої гри, оскільки саме вони забезпечують ігровий процес. Окрім того програмні модулі дають змогу поділяти гру на складові частини та змінювати її функціонал у окремих частинах гри, не вносячи змін глобально. Також, створені програмні модулі можуть бути використані у інших проектах, якщо підійти до реалізації програмних модулей з точки зору ООП. Для подальшої розробки, потрібно сформувати розуміння того, які програмні модулі потрібно буде реалізувати та яким чином ці модулі будуть пов’язані один з одним та іншими елементами ігрового процесу. На рисунку 1.7 зображено програмні модулі розробляємої гри.

На представленому нижче рисунку 1.7 можна побачити, що програмні модулі всі логічним та лінгвістичним чином зрозумілі. При описані роботи гри у

жанрі філвордів потрібно реалізувати модулі, які перераховані нижче.

Модуль обробки натискань потрібен буде для визначення статусу натискання на клітинки з буквами, щоби розуміти на які букви було натиснуто та в якому слові. Даний модуль є важливою складової ідентифікації дій гравця, оскільки те, на яку клітинку натискає гравець впливає на замальовування клітинок з буквами та виконання правил ігрового процесу. Оскільки процес обробки натискань пов'язаний з ігровими об'єктами – цей зв'язок вказано зеленою стрілкою на рисунку 1.7.

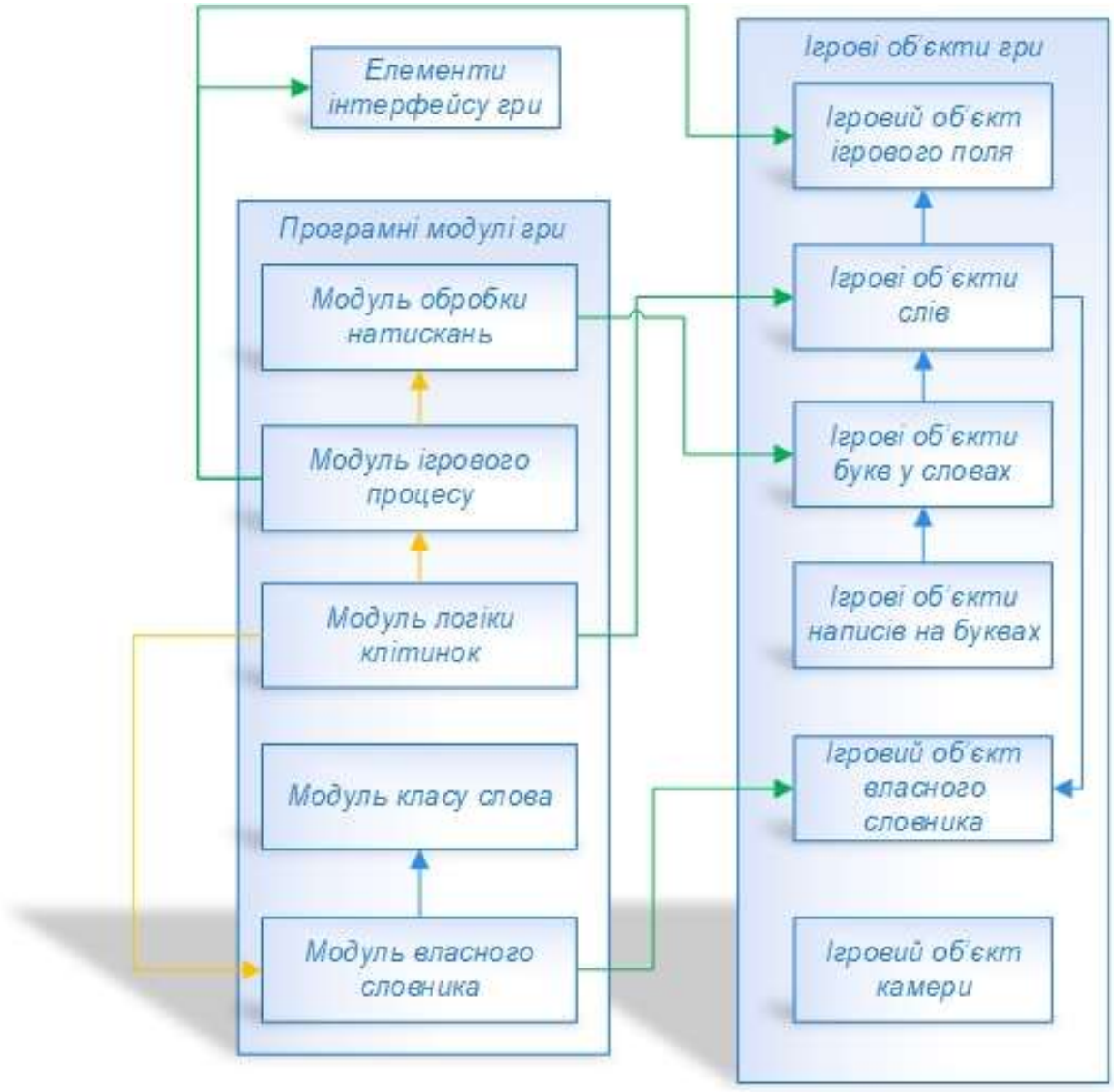


Рисунок 1.7. Програмні модулі та їх зв'язки з іншими елементами гри WordLines

Модуль ігрового процесу має відслідковувати безпосередньо як протікає гра

та чи виконує гравець правила гри. Також цей модуль повинен буде замальовувати клітинки одним кольором, а у разі якщо гравець помилився, то скидати замальовані клітинки. Також цей модуль має відслідковувати кількість слів, що потрібно розкрити. Жовтою стрілкою вказано зв’язок з модулем обробки натискань, а зеленою зв’язок з ігровими об’єктами.

Модуль логіки клітинок повинен забезпечувати формування слів у ігрових об’єктах слів. Він буде мати доступ до об’єктів букв та передавати в них складові слів. Окрім того, оскільки саме цей код формує слово у відповідному ігровому полі, цей скрипт повинен буде мати доступ до власного словника гри для отримання з неї випадкових слів.

1.5 Проектування системи вибірки слів із словника

Для повної реалізації теми дипломного проектування, а також виконання концепції ігрового процесу, потрібно реалізувати систему власного словника. Ця система має вносити важливу складову у ігровий процес, забезпечуючи випадкову вибірку слів із словника тим самим створюючи високу реіграбельність під час проходження ігрових рівнів. Система повинна мати модульну систему, яка буде давати можливість її модифікувати, а також зрозумілий процес додавання слів до словника.

Перед початком проектування стояло питання шляхів реалізації цієї системи, оскільки до вирішення задачі можна підійти різними методами. Одним з таких методів є створення зовнішньої бази даних, яку можна оновлювати ззовні ігрового програмного рушія, додаючи до гри нові слова за допомогою таблиць. Втім, такий підхід має обґрунтування якщо проект реалізується на веб-основі, або має функціонал додавання слів до словника користувачем. Також таке рішення потребує імплементації додаткового програмного комплексу, реалізації серверної частини для передачі оновлених словників до гравців. У випадку реалізовуємого проекту, таких функцій не передбачено, тому власний словник формується на базі скриптів, які можна буде змінювати безпосередньо у ігровому програмному рушії Unity, або за допомогою будь-якого редактора тексту. Це значно спрощує процес

розробки, але це не значить, що оновлення ігрових словників буде пов’язано із повним оновленням гри для користувача. Сучасні системи цифрових магазинів, як Steam дають можливість автоматично завантажувати оновлення для гравців, як тільки вони доступні. Окрім того, сама програма Steam відстежує, які файли потрібно оновити, а які залишились незмінними, тому процес оновлення не буде займати багато часу.

Визначившись із шляхом вирішення поставленої задачі, було виконано проектування системи вибірки слів із власного словника, що видно на рисунку 1.8. Проектування цієї частини гри дуже важлива частина, окрім того проектування словника з самого початку дає основу для бачення архітектури взаємодії модулів розробляємого проекту. Правильно побудований проект словника дає змогу коректно реалізувати інші модулі та елементи гри з урахуванням імплементації цієї системи.

Рисунок 1.8. Схема роботи вибірки слів із власного словника

Представлена на рисунку 1.8 схема має декілька складових, які важливі для розуміння роботи системи вибірки слів із словника. Основним елементом системи є Модуль класу слова, який забезпечує структуру, яка буде основою для слів у словнику. Структура слів у словнику буде вміщати в собі безпосередньо саме слово, його довжину, а також чи використовується це слово у ігровій сесії. Кожне нове слово, яке буде додаватись до словника, має створюватись саме з такими критеріями.

Заповнення словника повинне забезпечуватись Модулем власного словника. Цей модуль складається з Блоку ініціалізації слів, який викликається під

час початку ігрового процесу. Цей блок має створювати список слів, який базується на Модулі класу слова, створюючи слова за розглянутою раніше моделлю. Ці слова будуть записуватись у список, що на рисунку 1.8 зображений як Блок ініціалізованих слів. Нарешті Блок отримання слова зі словника має виконуватись в той момент, коли рівень побудовано. В цей момент ігрові об'єкти слів звертаються до цього блока, який повинен вибирати зі списку слова, що відповідають критеріям відбору. Така система дає змогу створювати власний великий словник, яку можна доповнювати новими словами за зрозумілою схемою доповнення. Також, використання такої системи дає змогу додавати додаткові критерії для слів, розподіл на теми, тощо. У майбутньому разом із додаванням слів до словника, можна створювати цілі комплекти оновлень, що будуть змінювати ігрові механіки та сам власний словник гри.

1.6 Реалізація основних ігрових механік

1.6.1 Створення проекту та його налаштування

Початковим етапом розробки гри є створення нового проекту. У різних рушіях цей процес реалізовано по-різному: деякі надають стартові шаблони, інші дозволяють одразу працювати в інтегрованому середовищі. У випадку Unity для цього використовується спеціальний інструмент Unity Hub, який виконує функції централізованого керування проектами. З його допомогою можна створювати нові проекти, змінювати конфігурацію середовища, обирати потрібну версію рушія, а також встановлювати оновлення або додаткові компоненти.

Однією з важливих функціональних можливостей Unity Hub є підтримка одночасного встановлення кількох версій рушія. Це особливо корисно у випадках, коли ведеться паралельна розробка кількох проектів, кожен із яких потребує певної версії рушія, або коли виникає необхідність в оновленні поточного проекту до новішої редакції Unity. Хоча рушій пропонує автоматичне перенесення проекту на нову версію, цей процес не завжди проходить без помилок і потребує додаткового контролю з боку розробника. Частіше за все розробники створюють паралельний проект, який будують на новій версії програмного рушія та

виконують перенос елементів та модулів гри зі старої версії у нову.

При створенні нового проекту користувач може вказати низку параметрів, як назву проекту, його розміщення на жорсткому диску, обрану версію рушія та тип початкового шаблону (наприклад, 2D, 3D або VR). Окрім того, Unity Hub пропонує налаштування інтеграції з хмарними сервісами компанії Unity та підключення системи контролю версій, однак у рамках цієї роботи ці опції не використовуватимуться, оскільки розробка здійснюється локально на одному комп'ютері. Втім використання цих технологій дозволяє сильно спростити процес розробки проекту, якщо команда розробників більша за одну людину. За допомогою системи контролю версій можна вносити зміни до елементів гри у окремих гілках проекту без впливу на поточну роботу іншого розробника, який зараз може працювати у другій гілці.

Шаблони проектів у Unity дозволяють швидко налаштувати середовище для роботи, позбавивши необхідності виконувати рутинні початкові дії, такі як створення базових об'єктів сцени, налаштування освітлення, камери, управління тощо. Наприклад, шаблон проекту для перегонів автоматично формує трек, транспортний засіб із базовою фізикою та початковий скрипт для керування. Додаткові шаблони можна завантажити через Unity Hub або знайти на офіційному ресурсі платформи. На рисунку 1.9 наведено приклад процесу створення нового проекту в середовищі 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.

У типовому проекті, створеному за допомогою Unity, структура файлової системи будується навколо двох ключових директорій: Packages та Assets. Каталог Packages містить службову інформацію, необхідну для роботи рушія. До нього входять внутрішні бібліотеки, конфігураційні файли, а також тимчасові дані, які генеруються під час редагування сцени або в процесі збірки проекту. Ці елементи зазвичай не змінюються безпосередньо розробником і використовуються

системою в автоматичному режимі, але при необхідності, досвідчений користувач може вносити зміни у ці файли.

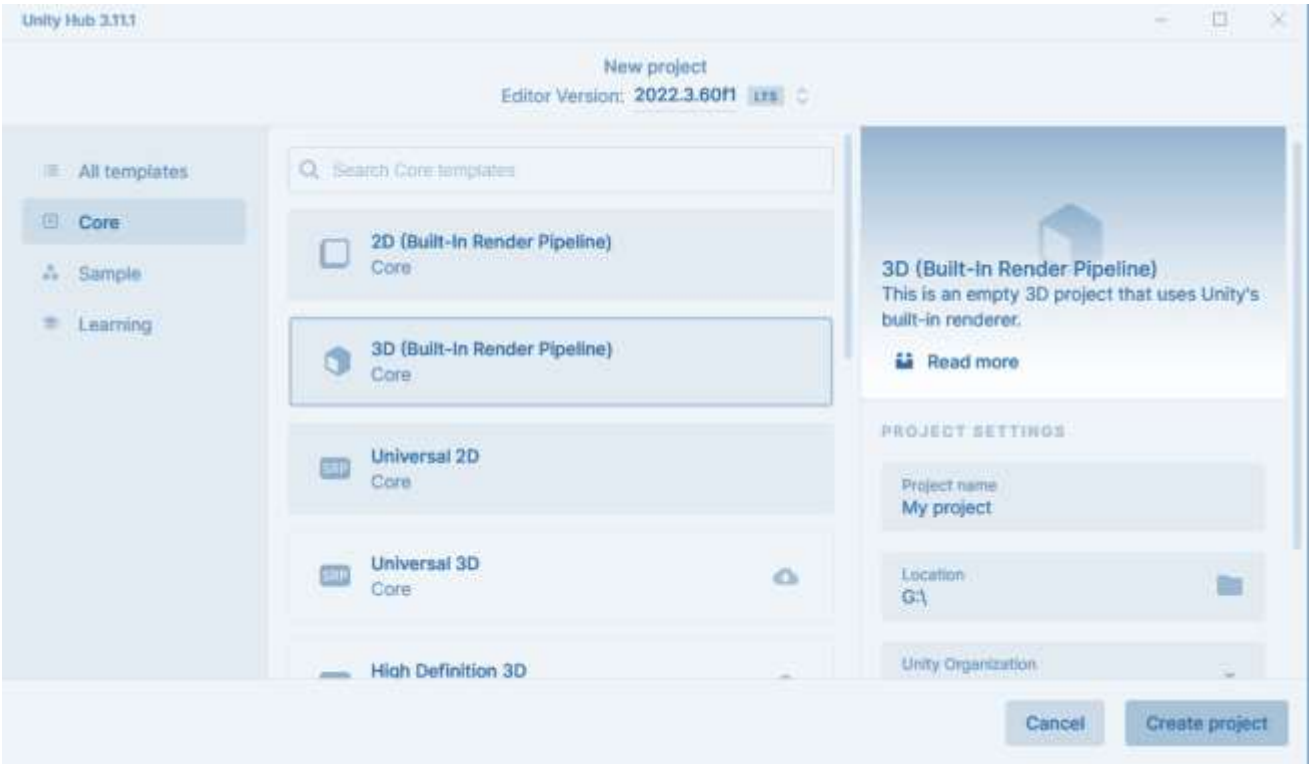


Рисунок 1.9. Створення нового проекту у ігровому програмному рушії Unity

Директорія Assets є основним робочим простором користувача. У цій папці зберігаються всі ресурси проекту, які створюються або імпортуються розробником: растрова і векторна графіка, шейдери, 3D-моделі, анімаційні кліпи, звукові ефекти, музика, скрипти на C#, матеріали та інші об’єкти, що використовуються під час створення ігрової сцени та її логіки.

Процес генерації нового проекту силами ігрового програмного рушія Unity займає певний час, після чого завантажується основне середовище розробки Unity. Інтерфейс редактора складається з множини вікон і панелей, кожна з яких виконує окрему функцію, як навігація по сцені, редагування ігрових об’єктів, робота з компонентами, перегляд ресурсів, написання коду тощо. Це модульне середовище дозволяє гнучко налаштовувати конфігурацію відповідно до потреб проекту. На рисунку 1.10 наведено загальний вигляд інтерфейсу середовища Unity, у якому безпосередньо виконується розробка гри.

Рисунок 1.10. Робочий простір ігрового програмного рушія Unity

Під час ініціалізації проекту було обрано шаблон для розробки 2D-ігор, у результаті чого ігровий рушій автоматично застосував відповідні параметри, якими є відображення сітки, конфігурації камери, освітлення та базової системи керування. Також проект відразу має всі необхідні ігрові об’єкти для першого ж запуска сцени.

1.6.2 Робота з графікою

Візуальна частина проекту відіграє важливу роль, оскільки саме вона задає художнє оформлення гри та створює її унікальний стиль. Однак на ранніх етапах розробки допустиме використання стандартних геометричних фігур у 3D або простих тимчасових 2D-зображень (спрайтів-заглушок). Такі елементи застосовуються як умовне візуальне наповнення для швидкої перевірки функціональної логіки гри та її базових механік, що дозволяє не витрачати додатковий час на підготовку остаточних ресурсів.

Цей підхід також сприяє формуванню загального уявлення про майбутню стилістику гри навіть на початковому етапі. Використання спрощених графічних елементів дозволяє оцінити цілісність ігрового середовища, протестувати сумісність візуальних компонентів і визначити напрямок подальшої роботи з художнім оформленням проекту.

Графічне наповнення будь-якого ігрового проекту є важливою складовою яка формує візуальну ідентичність, що виділяє розробляємий проект на фоні інших проектів. Є багато різних підходів до візуального стилю ігор, що може призводити до реалізації візуально стилю у мінімалістичному виді, як це роблять інді-студії, або навпаки реалізовувати проект з максимальною візуальною віддачою. Розробляємий проект є реалізацією гри жанру філворд, що саме по собі не має обґрунтування для використання великої кількості ефектів, або створення надреалістичної графіки з симуляціями поверхонь. Для філворду потрібно створити мінімальний набір графічних матеріалів.

З іншого боку гра не повинна виглядати дуже простою, або виконаною не в повному обсязі. Для цього потрібно провести аналіз та зрозуміти як можна привнести до візуальної складової розробляємої гри якусь особливість. Як вже було зазначено у розробленій концепції, клітинки букв на ігровому полі під час обрання слів будуть міняти свій колір та створювати різні цвітові переливи завдяки діям гравця.

Ще одною складовою, яку можна покращити є задній фон. В іграх задній фон на рівнях створюють різними способами – заливають одним кольором, розміщують зображення або анімацію, створюють скайбокси. Скайбокс – це технологія яка дозволяє імітувати небо з усіма його особливостями як хмари, сонце, світанок або захід сонця, зоряне небо та місяць на ньому. По суті скайбокс – це зображення неба, яке видно з будь-якої точки ігрового рівня. Його особливістю є те, що скайбокси виконуються з невеликою роздільною величиною, для зменшення навантаження на обладнання гравця. Є різні версії скайбоксів, які відрізняються один від одного якістю виконання. Перші скайбокси представляли собою 5 спрайтів неба, які розміщувались на сторонах куба скайбоксу навколо рівня. Через це та низьку якість спрайтів можна було побачити шви або кути цього куба. У сучасних ігрових програмних рушіях скайбокси імітують сферу, що виглядає набагато приємніше та якісніше для гравця.

Для створення скайбоксу потрібно імплементувати систему скайбоксів а для них створити спрайти поверхонь скайбокса. Є чотири виду розгортання

скайбоксу: 6-сторонній скайбокс, Cudemap, Панорамний скайбокс та Процедурно генеруємий кайбокс. Для розробляємого проекту буде використовуватись панорамний скайбокс для якого потрібно створити текстуру та на її основі матеріал для самого скайбоксу. На рисунку 1.11 можна переглянути створені для проекту текстури та матеріали скайбоксів.

Рисунок 1.11. Створені текстури та матеріали скайбоксів для розробляємої гри

Завдяки створеним скайбоксам можна буде створювати особливу атмосферу під час проходження рівнів. При потребі, скайбокси у майбутньому можна змінювати, або створити систему залежності скайбоксу від реального часу у гравця, або стилістики рівня.

Для того, щоби створені скайбокси були використані у ігровій сцені, достатньо обрати потрібний матеріал скайбоксу та перетягнути його на ігровий простір сцени Unity. Програмний рушій автоматично визначить цей об’єкт, як основу для скайбоксів та встановить матеріал для нового відображення скайбоксу.

1.6.3 Створення ігрових об’єктів та префабів

Для реалізації розробляєми гри є важливою складовою вибірка випадкових слів із власного словника. Через це варіант із заздалегідь створеними рівнями відпадає, оскільки неможливо та не раціонально створювати різні рівні власноруч. Більш правильним шляхом є створення умов та системи, яка сама реалізовувала рівень та наповнювала б його. Спочатку потрібно було визначитись із тим яким чином виконувати розміщення слів та букв на ігровому полі та як реалізовувати намічене. Першим етапом є проробка механізму роботи зі словами та ігровими об’єктами. Гравець взаємодіє не з словами, а з їх буквами, при цьому ця взаємодія

невідривна від самих слів. Це значить, що кожна буква має розуміти частиною якого слова вона є. З програмної точки зору, слова у власному словнику будуть реалізовані за допомогою строкового типу string. Цей тип зберігає строки, які можна розкласти на букви, або змінні типу char. Тобто, якщо це перекласти на геймдизайн, то вийде, що потрібно реалізувати окремо ігрові об'єкти букв та ігрові об'єкти їх слів. На рисунку 1.12 схематично відображено такий зв'язок.

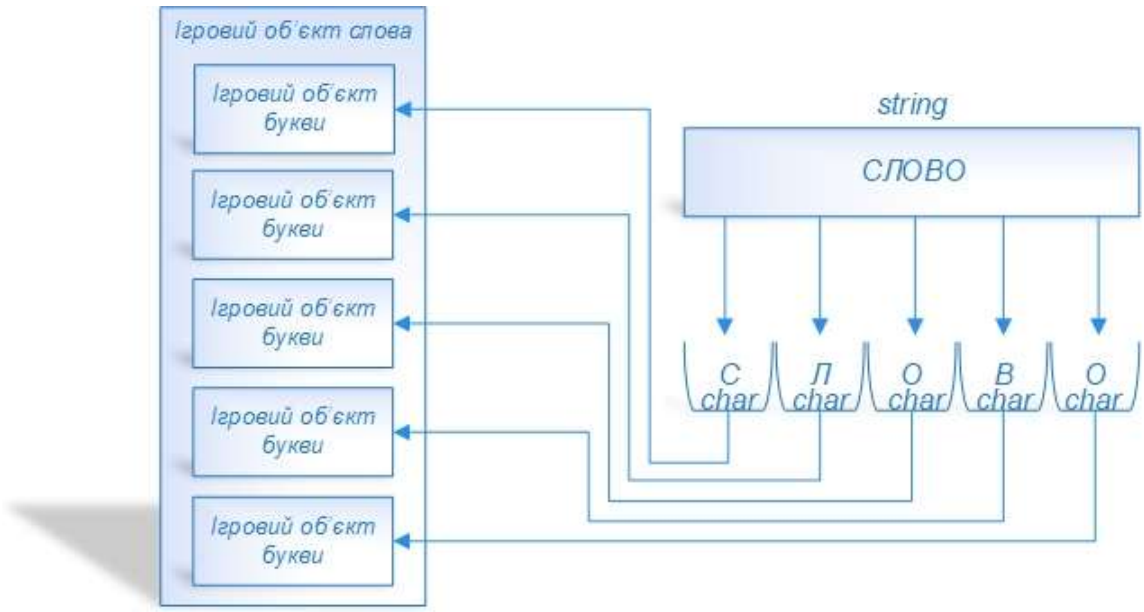


Рисунок 1.12. Зв'язок структури слів із власного словника та ігрових об'єктів на ігровому полі

Відповідно до цієї концепції зв'язку, виходячи з того, що слова мають різну довжину, потрібно створити префаби ігрових об'єктів слів із відповідною кількістю ігрових об'єктів букв у цих словах. Префаби – це такі ігрові об'єкти, які розміщуються у ассетах та можуть бути використані із коду гри. Найчастіше за все вони використовуються для клонування ігрових об'єктів одного типу. Наприклад, якщо потрібно створювати у грі ворогів, то спочатку створюється оригінал ігрового об'єкту ворога, він налаштовується та з цього ігрового об'єкту створюється префаб. Після цього цей префаб можна додавати до ігрового простору, як окремі незалежні ігрові об'єкти ворогів з усіма налаштуваннями. Ще одною сильною стороною префабів є їх незалежність один від одного. Тобто вороги можуть відрізнятись один від одного по зовнішньому вигляду, або параметрам. З іншого боку всі префаби пов'язані із префабом в ассетах і у разі

його зміни – ці зміни будуть внесені у кожну копію ігрового об’єкту з цього префабу.

Процес створення ігрових об’єктів у ігровому програмному рушії Unity виконується через відповідне контекстне меню у панелі Ієрархії об’єктів на сцені. В цій панелі знаходяться ігрові об’єкти вже створені за шаблоном. Перед тим як починати створювати об’єкти слів, потрібно визначитись також з тим, на базі яких об’єктів буде базуватись гра.

Першим варіантом є створювати ігрові об’єкти безпосередньо на сцені, як звичайні об’єкти. Для подальшої роботи з такими об’єктами потрібно буде також створити код для обробки натискань на кнопки миші, а також розміщати окремо на цих об’єктах об’єкти з написами, що будуть містити букви. Також такий підхід потребує створення додаткового ігрового об’єкту типу Grid для центрування та прив’язки створюємих об’єктів один до одного, оскільки без цього елемента буде складно створити ігрове поле без порушень у його формі.

Другий підхід – створювати ігрові об’єкти на полотні інтерфейсу, як його елементи. Такі об’єкти можна відразу створювати як кнопки, оскільки по суті вони будуть виконувати ті ж самі функції – чекати натискання на них та виконувати якусь дію. Самі об’єкти можна групувати у об’єктах словах. Також такий спосіб дає можливість відразу розміщувати об’єкти із прив’язкою один до одного на полотні інтерфейсу, що значно спрощує створення рівного за формою ігрового поля.

У межах реалізації даного проекту було обрано другий підхід до створення ігрових об’єктів слів та їх букв, який виявився більш ефективним і зручним у роботі. На початковому етапі було додано базовий UI-об’єкт – Canvas. Компонент Canvas у Unity виконує роль основної платформи для побудови елементів інтерфейсу. Він є обов’язковим контейнером для всіх UI-компонентів сцени, включно з такими елементами, як кнопки (Button), текстові поля (Text), зображення (Image), випадаючі списки (Dropdown), повзунки (Slider), меню та інші візуальні елементи взаємодії. Без Canvas ці елементи не відображатимуться в ігровому вікні, оскільки саме він забезпечує їхнє коректне позиціонування та

візуалізацію. Під час додавання будь-якого інтерфейсного об’єкта за допомогою пункту GameObject – UI, система автоматично перевіряє наявність Canvas у сцені. Якщо він відсутній – Unity створює його автоматично, разом із двома супутніми компонентами. Canvas Scaler використовується для масштабування інтерфейсу відповідно до роздільної здатності екрана. Graphic Raycaster виконує забезпечення обробки взаємодії з UI-елементами, зокрема кліків та наведеннь. Такий підхід дозволяє швидко формувати графічну складову інтерфейсу без необхідності ручного налаштування кожного компонента, що значно оптимізує процес проектування UI-середовища.

Після створення полотна користувацького інтерфейсу було створено ігрові об’єкти для слів довжиною від 5 до 8 букв. Всередині цих об’єктів було розміщено ігрові об’єкти типу Button у кількості відповідній кількості букв у словах. За замовчення Button вже мають вкладені об’єкти для написів на кнопках, тому створювати їх окремо не потрібно. Створивши всі варіанти за кількістю букв, ці ігрові об’єкти були перетворені на префаби із подальшим розміщенням на сцені та редагуванням позицій букв на ігровому полі. На рисунку 1.13 можна побачити створене ігрове поле.

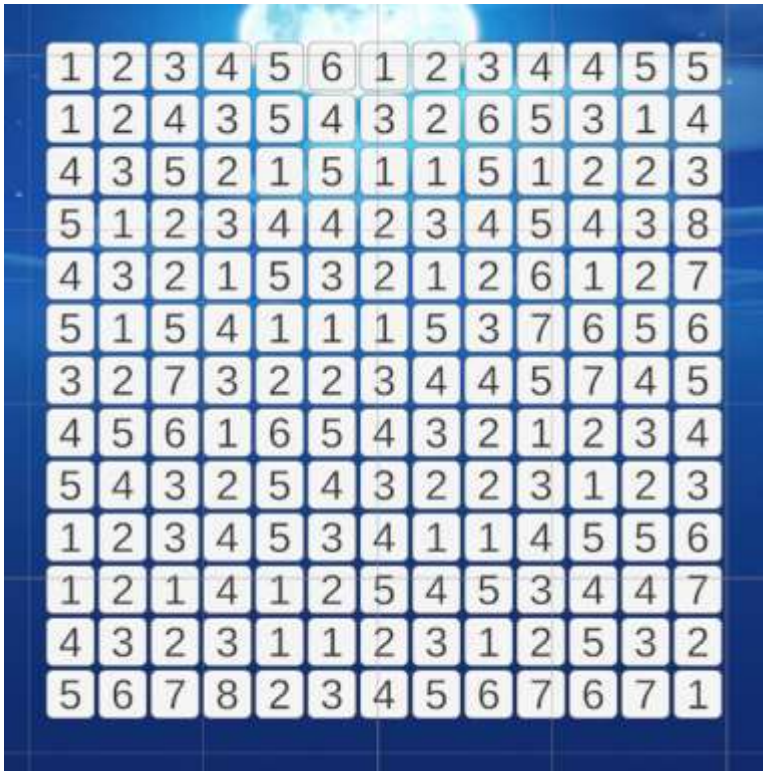


Рисунок 1.13. Реалізоване ігрове поле з розміщеними на ньому об’єктми-словами

Слід зазначити, що цифри які зображені на рисунку 1.13 грають службову роль для орієнтації розробника у черговості букв в слові. З початком ігрової сесії, ці цифри будуть перезаписані літерами слів отриманих із власного словника. Також потрібно створити ігровий об’єкт, який буде вміщати в собі власний словник розробляємої гри. Для цієї цілі також було створено пустий ігровий об’єкт, він був розміщений у середині сцени. На рисунку 1.14 можна побачити Ієрархію ігрових об’єктів після реалізації всіх необхідних об’єктів на сцені.

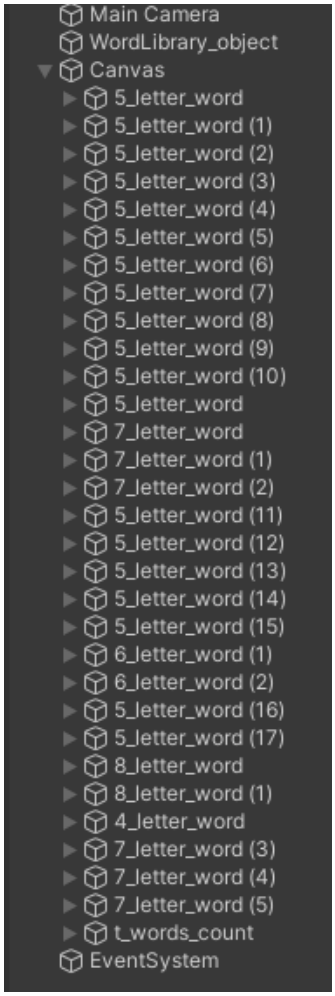


Рисунок 1.14. Ієрархія об’єктів з усіма необхідними для гри об’єктами

Серед об’єктів також можна побачити такий елемент як Camera, який відповідає за відображення гри на екран гравця. Вона показує якусь частину ігрової сцени, але для цього проекту, її ракурс не має великої роли, оскільки всі ігрові об’єкти для гри розміщені на полотні інтерфейсу, через що вони завжди будуть знаходитись перед гравцем. Коли було створено всі ігрові об’єкти потрібно реалізувати необхідний код для функціонування гри.

1.6.4 Реалізація модулів основних ігрових механік

Раніше було спроектовано перелік модулів коду, які необхідні для реалізації розробляємої гри. Щоби забезпечити основні ігрові механіки, потрібно реалізувати код обробки натискань, код ігрового процесу а також модуль логіки клітинок.

Головним модулем для роботи всієї гри є Модуль логіки клітинок Part_logic.cs. Його головною задачею є поєднання клітинок з буквами у слово, контроль їх змісту, а також виконання налаштування статусів клітинок з буквами. На рисунку 1.15 зображена зв'язки та структура цього модуля.

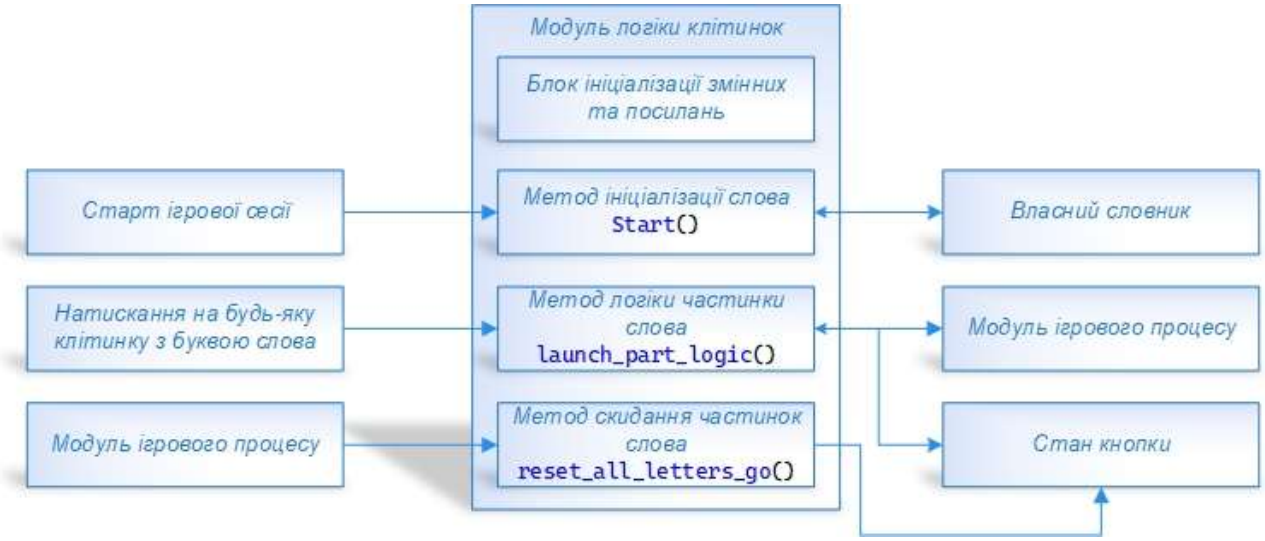


Рисунок 1.15. Структура Модуля логіки клітинок та його зв'язки з іншими елементами розробляємої гри

Цей модуль розміщується у кожному ігровому об'єкті, якій формує слово. Наприклад, структура слова з п'ятьма літерами складається з ігрового об'єкта слова, а також п'яти ігрових об'єктів букв цього слова та по одному напису на кожну букву. Завдання Модуля логіки клітинок отримати слово зі словника, передати його до ігрових об'єктів букв та на протязі всієї ігрової сесії слідкувати за статусом розкриття слова, яке записано в ньому.

Блок ініціалізації змінних та посилань відповідає за першочергове створення полів для параметрів слова. За допомогою класу SerializeField можна створювати для скрипту поля безпосередньо у редакторі Unity. Наприклад, розглядаємий скрипт має змінну word_length яка зберігає довжину слова, яка буде

записано в ігровий об’єкт слова. При додаванні класу SerializeField, ця змінна з’явиться на панелі компонентів у тілі скрипту. На рисунку 1.16 можна побачити поля, які заповнюються для коректної роботи зі словами.

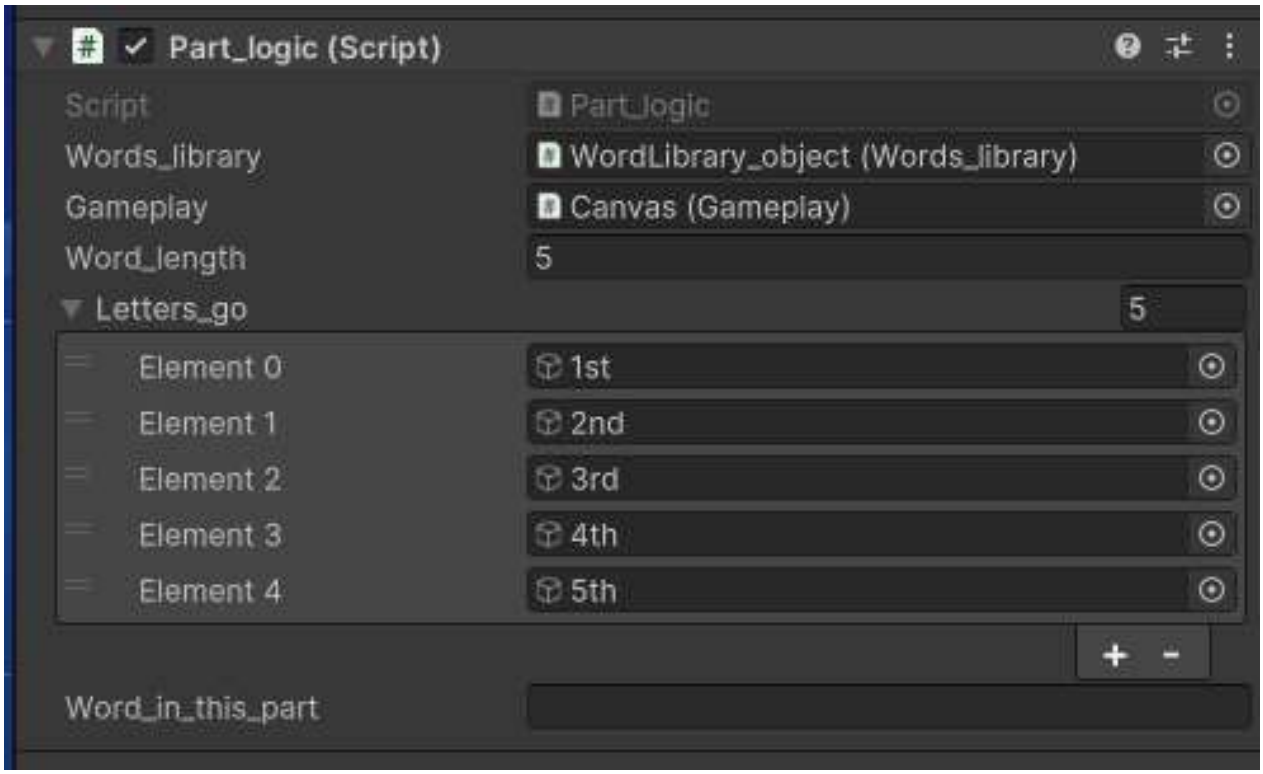


Рисунок 1.16. Всі поля Модуля логіки клітинок, які формуються у розробляємому скрипті

Як можна побачити з рисунку 1.16, до параметрів слова входять посилання на словник, ігровий об’єкт полотна ігрового поля, кількість букв у слові, а також налаштовуємий елемент List() за допомогою якого можна створювати посилання на ігрові об’єкти букв слова на прикладі рисунку 1.12. Також тут присутнє поле Word_in_this_part, яка зберігає в собі слово, яка було завантажено в цей скрипт на початку ігрової сесії – цей параметр відображається на панелі компонентів у здебільшому для зручності розробника для відладки коду. Одною із змінних які мають закритий тип є змінна булевого типу is_part_logic_activ, яка за замовченням приймає значення true. Ця змінна необхідна для визначення, чи можна взаємодіяти із цим словом в цілому, та його буквами у зокрема.

Метод ініціалізації слова Start() виконує завдання по отриманню слова із власного словника гри та запису цього слова у ігрові об’єкти-букви. Цей метод виконується на самому початку життя ігрового об’єкту слова, як тільки

завантажується ігрова сцена. Метод отримання слова буде розглянутий пізніше, у підрозділі 1.7. На даному етапі процес отримання букв реалізовується за допомогою циклу for. Кількість ітерацій цього циклу відповідає довжині слова. Цикл проходить по ігровим об’єктам букв, та отримує з них посилання на компонент TMP_Text, який відповідає за напис на клітинці букви. Отримання доступу виконується за допомогою методу GetComponentInChildren(). Раніше було зазначено, що кожна буква є ігровим об’єктом-кнопкою, в яку вкладено ігровий об’єкт напис. Таке вкладення створює ієрархіє батько-нащадок. Зазначений вище метод виконує звернення до нащадку та шукає в ньому компонент, якій вказується як аргумент цього методу. На початку ігрової сесії, TMP_Text зберігає в собі цифру із порядковим номером букви у слові. Після виконання методу ініціалізації слова, кожний такий напис отримає свою букву. Нижче продемонстровано частину коду методу Start(), яка виконує отримання літер для букв та їх запис:

```
for (int i=0; i< letters_go.Count; ++i){
    TMP_Text _button_text = letters_go[i].GetComponentInChildren
<TMP_Text>();
    _button_text.SetText(word_in_this_part[i].ToString());
}
```

Метод launch_part_logic() цей метод виконується при натисканні на кнопку-букву у слові. Як тільки це стається, ця кнопка викликає цей метод. Завдання цього методу визначити, чи активне це слово для роботи. Якщо так, то цей метод передає до Модуля ігрового процесу поточне слово, з яким виконуються дії. Наступним кроком, розглядаємий метод проходить за допомогою циклу for по буквах слова яке зараз у роботі і шукає кнопку, яка викликала цей метод за допомогою булевого значення змінної current_button_state у кожній букві. Під час гри, статус true має лише та буква, яка викликала метод launch_part_logic(), тому що на неї натиснули. Коли активна буква знайдена, отримується її компонент кольору, в нього записується поточний колір для замальовування букв. Після цього букві встановлюють статус is_letter_colored як true, що вказує на те, що з буквою вже

були виконані дії.

Далі метод launch_part_logic() перевіряє, чи всі букви слова замальовані. Для цього створюється тимчасова змінна check_if_all_colored вона відразу встановлюється у значення true. За допомогою циклю розглядаємий метод проходить по своїм буквам та виконує логічне множення змісту змінної check_if_all_colored із змінною is_letter_colored у кожній букві. Якщо хоча б одна з букв має статус false, тоді змінна перевірки замальованності всіх букв слова встановлюється у значення false.

Виконується остання перевірка. У разі якщо всі букви слова замальовані, тоді слово блокується для роботи. У Модуль ігрового процесу передається команда змінити колір для замальовування букв та змінити кількість слів для знаходження.

Останній Метод скидання частинок слова reset_all_letters_go() викликається із Модуля ігрового процесу. Його завдання скинути прогрес замальовування слова шляхом послідовної зміни кольорів букв на стандартний колір для нерозкритих слів та скинути статус замальованності is_letter_colored кожної із букв у цьому слові у значення false. Цей метод виконується у разі, якщо гравець натиснув на букву іншого слово для скидання прогресу розкриття. Нижче приведено приклад коду розглянутого методу:

```
public void reset_all_letters_go(){
   if(is_part_logic_activ)   {
      for (int i = 0; i < letters_go.Count; ++i)      {
         Image _current_go_image = letters_go[i].GetComponent<Image>();
         _current_go_image.color = _gameplay.default_color;
         Button_state _button_state = letters_go[i].GetComponent
<Button_state>();
         _button_state.is_letter_colored = false;      }
   }
}
```

Модуль обробки натискань Button_state.cs знаходиться у кожному ігровому

об’єкті кнопці, що грає роль букви. Його завдання полягає у тому, щоби зберігати поточний статус натискання на букву у слові, а також зберігати статус замальованості букви. На рисунку 1.17 зображена структура цього модуля та зв’язки з іншими модулями гри.

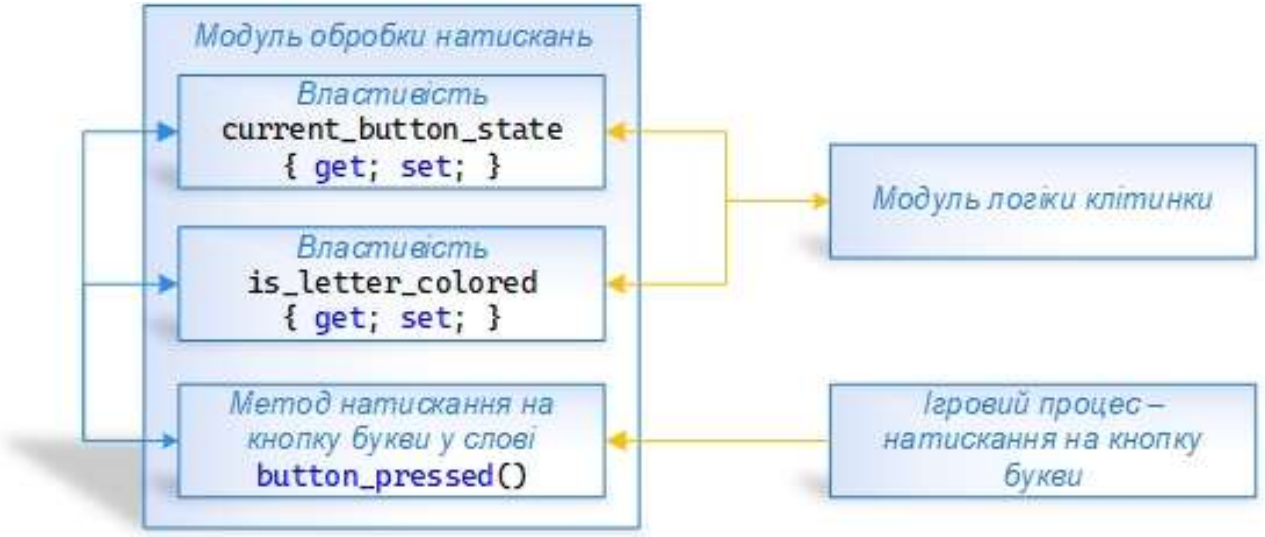


Рисунок 1.17. Структура Модуля обробки натискань та його зв’язки з іншими елементами розробляємої гри

З рисунку 1.17 видно, що Модуль обробки натискань не великий, але він відіграє ключову роль у зв’язку гравця з ігровою логікою, оскільки зберігає поточні та довгострокові статуси клітинок. Він виконується кожний раз, коли гравець натискає на букву серед ігрового поля та запускає процес ігрової логіки, який рухає геймплей.

Властивість current_button_state потрібна для ідентифікації словом букви, яка викликала Модуль логіки клітинки. Ця властивість має публічний доступ, тому для її отримання достатньо створити посилання на конкретну букву. Оскільки всі ігрові об’єкти слів вже мають посилання на свої букви, це не є проблемою.

Властивість is_letter_colored потрібна для визначення Модулем логіки клітинки, чи зафарбовані всі літери слова. Кожна буква початково має значення для цієї властивості як false. Як тільки процес активації букви закінчується, це значення властивості змінюється на true. Так само як і попередня властивість, ця має публічний доступ для спрощення роботи з ним.

Метод натискання на кнопку букви у слові button_pressed() викликається із ігрового процесу під час натискання на кнопку-букву. Як тільки це відбувається, то кнопка звертається до цього методу. Цей метод перемикає властивість current_button_state у значення true, вказуючи, що ця буква викликає Модуль логіки клітинки, після отримується посилання на Модуль логіки клітинки Part_logic.cs слова до якого відноситься буква та запускається метод launch_part_logic(). Після закінчення відпрацювання цього методу, значення властивості current_button_state знову встановлюється у false та закінчується виконання методу натискання на кнопку букви у слові button_pressed(). Нижче представлено код цього модулю:

```
public bool current_button_state { get; set; } = false;
public bool is_letter_colored { get; set; } = false;
public void button_pressed(){
    current_button_state = true;
    Part_logic _part_logic = GetComponentInParent<Part_logic>();
    _part_logic.launch_part_logic();
    current_button_state = false;
}
```

Модуль ігрового процесу Gameplay.cs знаходиться у ігровому об’єкті ігрового поля, яким виступає полотно інтерфейсу на якому знаходяться елементи гри. Таке розміщення обґрунтоване задачею створити умови для швидкого пошуку та знаходження ігрового об’єкту, який буде мати в собі компонентом цей скрипт. Оскільки кожний ігровий об’єкт слова на ігровому полі повинен мати доступ до цього модуля, його потрібно розмістити якомога вище у ієрархії. Завдання цього модулю проконтролювати те, з яким словом зараз взаємодіє гравець під час ігрової сесії, а також змінювати колір для замалювання букв на ігровому полі, колі гравець знаходить всі букви у схованих слова. На рисунку 1.18 зображено структуру Модуля ігрового процесу та його зв’язки з іншими елементами розробляємої гри.

Другий колір змінюється кожний раз, коли гравець повністю розкриває нове

слово. Також серед змінних тут оголошуються current_go_in_work та pressed_go – це поточне слово у роботі та натиснуте слово. Ці дві змінні потрібні для визначення, чи працює гравець з одним і тим самим словом під час свого наступного ходу.

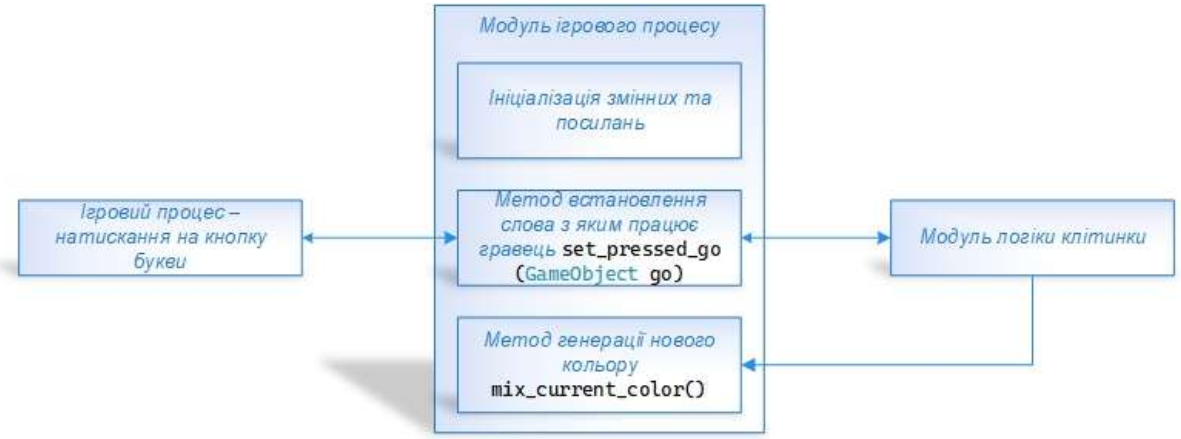


Рисунок 1.18. Структура Модуля ігрового процесу та його зв'язки з іншими елементами грозробляємої гри

Перша змінна зберігає посилання на слово, яке гравець почав відкривати, а друга змінна кожний раз отримує нове посилання на ігровий об'єкт слова. Обидві змінні ініціалізуються значенням null, що вказує на те, що вони не мають жодних посилань на ігрові об'єкти.

Метод встановлення слова з яким працює гравець set_pressed_go(GameObject go) викликається із Модуля логіки клітинки, коли гравець натискає на нову літеру на ігровому полі. Після виконання цей метод повертає логічне значення. Завдання цього методу визначити, чи гравець працює в тому ж слові, з яким працював минулий хід, або помилився та почав взаємодіяти з буквами іншого слова. Для цього цей метод отримує як параметр ігровий об'єкт слова із Модуля логіки клітинки. Цей ігровий об'єкт передає посилання на себе до змінної pressed_go. Після отримання нового посилання, метод розпочинає розгалужену перевірку.

У першому випадку, метод перевіряє, чи є у змінній current_go_in_work посилання на якійсь ігровий об'єкт, або інакше кажучи чи повертається значення null. На початку будь-якої ігрової сесії, завжди перше натискання на будь-яку букву буде викликати саме цю перевірку, яка повинна виконати першочергову

ініціалізацію значення ігрового об’єкту, з яким зараз працює гравець. Тому у зміну current_go_in_work передається посилання зі змінної pressed_go після чого метод повертає значення true.

У другому випадку, якщо значення змінної current_go_in_work не дорівнює значенню змінної pressed_go, то це значить, що гравець помилився і почав роботу з іншим словом. В тілі цієї перевірки, отримується посилання на Модуль логіки клітинки від ігрового об’єкту слова з яким раніше працював гравець, після чого викликається метод reset_all_letters_go() для скидання всіх статусів букв та кольорів букв у цьому слові. Далі посилання із змінної pressed_go записується у змінну current_go_in_work щоби фіксуватись на новому слові у роботі. Метод повертає true Нижче представлений код цієї частини перевірки:

```
else if (current_go_in_work != pressed_go){
    Part_logic _part_logic = current_go_in_work.GetComponent<Part_logic>();
    _part_logic.reset_all_letters_go();
    current_go_in_work = pressed_go;
    return true;
}
```

У третьому випадку, якщо посилання у змінних pressed_go та current_go_in_work однакові, то це значить, що гравець продовжує працювати з тим самим словом, що і раніше. Метод повертає true. У самому останньому разі перевіряються критичні помилки, у разі якщо жодна із попередній умов не виконалась.

Метод генерації нового кольору mix_current_color() викликається із Модуля логіки клітинки. Він потрібний для випадкової генерації нового кольору для ігрових об’єктів з буквами. Кольори у ігровому програмному рушії Unity зберігаються у відповідному класі, який є частиною компоненту SpriteRenderer, наприклад.

Саме значення кольору зберігається як три властивості кольору за його спектр – красний, червоний та синій. Значення насиченості цих спектрів можна вказувати вручну як число із плавоючою крапкою. Тому, кожний раз, коли гравець

успішно знаходить нове слово та замальовує його старим кольором, викликається цей метод. Далі послідовно перевизначаються величини насиченості спектрів кольору. Нижче приведено код цього методу:

```
public void mix_current_color(){
   current_word_color.r = Random.Range(0.1f, 0.70f);
   current_word_color.g = Random.Range(0.1f, 0.80f);
   current_word_color.b = Random.Range(0.1f, 0.99f);
}
```

Після реалізації цих ігрових механік, гру можна запустити, та можна буде натискати на кнопку букв, але самі слова не будуть отримуватись, оскільки потрібно реалізувати власний словник.

1.7 Реалізація вибірки слів із словника

Перед реалізацією вибірки слів із словника, потрібно виконати написання самого словника. Раніше, у концепції та на етапі проектування гри, було сформовано вимоги до власного словника та принципів його роботи. Він має бути виконаний на основі доступних технологій у ігровому програмному рушії Unity при цьому архітектура як словника, так і процесу додавання нових слів, має бути максимально простою та зрозумілою. На етапі проектування було висунуто схему роботи вибірки слів із словника. Виходячи із тієї схеми, можна констатувати, що спочатку потрібно створити базу для слів, як об’єктів у грі. Це є одним із проявлень підходу до використання ООП у розробці гри, а також кроком у напрямку створенню модульної системи кодів гри, оскільки реалізований код у подальшому можна буде використовувати у інших проектах, в яких мається на увазі використовувати словники слів, або навіть речень.

Для цього необхідно створити Клас слів Word_class.cs. Такий підхід зумовлений потребою використовувати списки, що дуже добре працюють саме з класами. Окрім того, використання класу дає змогу більш гнучким чином впливати на параметри слів у словнику. Структура Класу слів Word_class.cs дуже проста та складається лише із властивостей, які відповідають вимогам етапу

проектування. Це наступні властивості:

- public string word { get; set; } – властивість, яка описує безпосередньо саме слово, створена із типом string та може бути записана або зчитана ззовні через посилання на екземпляр класу;
- public int word_length { get; set; } - властивість, яка описує довжину слова, створена із типом цілочислених значень int, та може бути записана або зчитана ззовні через посилання на екземпляр класу;
- public bool is_word_in_use { get; set; } - властивість, яка описує чи в даний момент використовується слово на ігровому полі, створена із типом bool для отримання логічних значень, та може бути записана або зчитана ззовні через посилання на екземпляр класу;

Окремо слід зазначити, що оскільки цей клас буде використовуватись не як компонент для ігрових об’єктів, а як батьківський клас для його екземплярів, то процес його створення дещо відрізняється від створення простого скрипту для Unity. Якщо скрипти створюються як нащадки класу MonoBehaviour, що описує всі команди та методи притаманні для ігрового програмного рушія Unity, то Word_class.cs буде створюватись як самостійний клас. Це є обов’язковою умовою для створення самостійних класів у ігровому програмному рушії Unity, оскільки тільки так рушій може відрізнити скрипт, який лежить у ассетах та клас, який потребно компілювати у саму гру, як її частину.

Тепер, коли Модуль класу слів створено потрібно реалізувати Модуль власного словника. Цей модуль буде розміщуватись у ігровому об’єкті спеціально створеному для власного словника гри, та буде знаходитись поза ієрархією ігровог поля. Він буде викликатись лише на початку ігрової сесії та видавати слова за потребою. На рисунку 1.19 зображено структуру Модуля власного словника та зв’язки з іншими елементами гри.

У блоці Ініціалізації змінних та посилань буде створюватись список для слів, що будуть використовуватись у словнику _words. Початково, на етапі створення модуля під час генерації ігрового рівня, цей список не має ніяких позицій. Втім у методі Start(), який виконується під час створення ігрового об’єкту

словника, викликається Метод ініціалізації власного словника initialize_words_library().

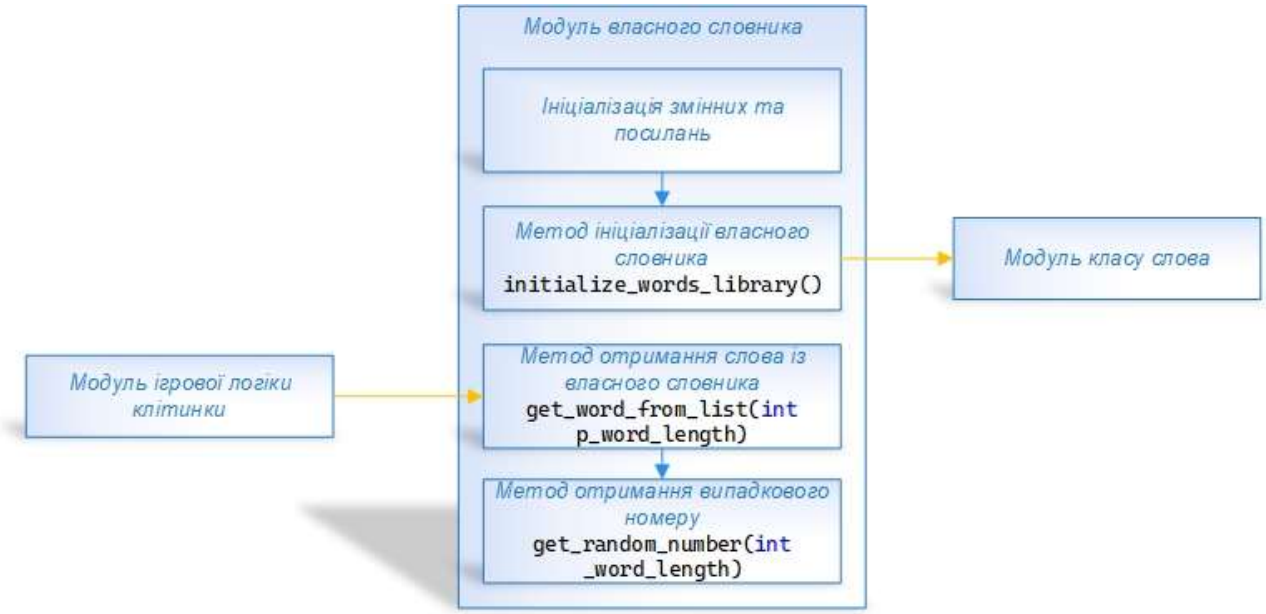


Рисунок 1.19. Структура Модуля власного словника та його зв’язки з іншими елементами розробляємої гри

Метод ініціалізації власного словника initialize_words_library() виконує заповнення списку слів словника. Для цього виконується команда Add у списку _words. При створені слова вказуються параметри слова: саме слово, його довжина та чи використовуються це слово у грі у булевому значенні – початково це значення встановлено у false. Нижче представлено приклад коду із створенням декількох слів для словника:

```
_words.Add(new Word_class() { word = "ГОРА", word_length = 4, is_word_in_use = false });
_words.Add(new Word_class() { word = "МІСТ", word_length = 4, is_word_in_use = false });
_words.Add(new Word_class() { word = "ЛІТО", word_length = 4, is_word_in_use = false });
_words.Add(new Word_class() { word = "СНІГ", word_length = 4, is_word_in_use = false });
_words.Add(new Word_class() { word = "ВОДА", word_length = 4, is_word_in_use = false });
```

Наступний Метод отримання слова із власного словника get_word_from_list(int p_word_length) викликається під час створення ігрових об’єктів слів, із Модуля ігрової логіки клітинки. Цей метод повертає значення типу string. Завдання цього методу вибрати випадкове слово із словника та передати його до ігрового об’єкта слова на ігровому полі. Для виконання пошуку слова використовується довжина слова, яке намагається отримати Модуль ігрової логіки клітинки.

За допомогою циклу do{}while, метод кожну ітерацію створює змінну номеру слова word_number та отримує випадковий номер з методу get_random_number розглядаємого модуля. Далі створюється тимчасова змінна типу класу слова із словника _word_from_library і в неї записується слово по отриманому випадковому номеру із списку власного словника. Далі виконується перевірка, чи використовується обране слово через властивість у змінній _word_from_library. Якщо використовується, тоді починається нова ітерація пошуку слова. Якщо слово не використовується, тоді до слова у списку власного словника за отриманим раніше випадковим номером встановлюється статус, що слово використовується на полі. Метод повертає отримане слово із власного словника.

Метод get_random_number(int _word_length) отримує довжину слова, випадковий номер якого потрібно отримати. Річ у тім, що слова сортовані за розміром, тому слова довжиною у 5 букв мають номера між 20 та 39. Знаючи довжину слова, викликається switch, що отримує випадковий номер у заданому діапазоні, наприклад між 20 та 39. Така система дозволяє із додаванням нових слів у список, розширювати діапазони відповідно до внесених змін.

1.8 Реалізація інтерфейсу

Розробленим концептом передбачено реалізація ігрового інтерфейсу, який би давав змогу гравцю розуміти скільки слів йому потрібно знайти на ігровому полі. Такий функціонал реалізовується шляхом додавання до полотна Canvas елементу інтерфейсу текстового типу. З початком ігрової сесії це поле повинно

отримувати значення кількості ігрових об’єктів слів, а вже в ході ігрового процесу послідовно зменшувати цю кількість разом із тим, як гравець знаходить слова на ігровому полі. На рисунку 1.20 зображено розміщений текстовий елемент.

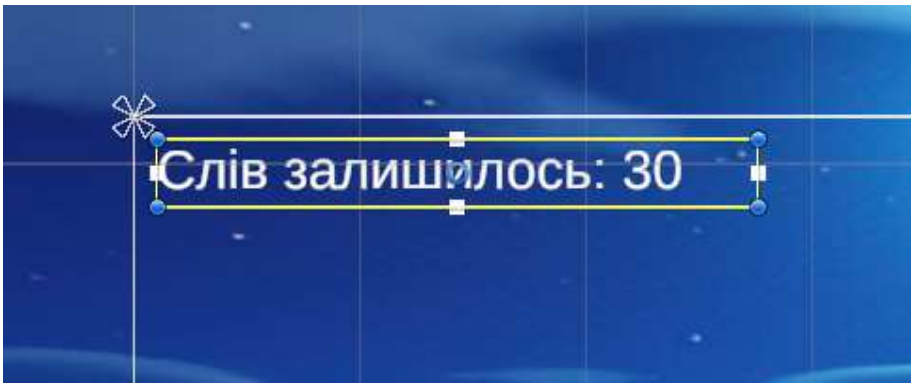


Рисунок 1.20. Текстовий елемент розміщений на інтерфейсі гравця

Початковий текст та значення встановлено під час розміщення. Для зміни напису, потрібно модифікувати код Модулю ігрового процесу, оскільки саме цей модуль знає точну кількість ігрових об’єктів слів на сцені. Для роботи з текстовим полем було створено посилання на цей елемент інтерфейсу. Для зміни значення у полі також створено ще один метод у розглядаємому модулі tape_new_words_count(). Він не отримує жодних значень та викликається із Модуля логіки клітинок, коли виконується перевірка на замальованність всіх букв у слові. Цей метод зменшує кількість слів для пошуку на одиницю та редагує компонент тексту у текстовому елементі інтерфейсу. Код що виконує ці дії зображено нижче:

```
public void tape_new_words_count(){
   --words_left;
   word_count_text.SetText("Слів залишилось: " + words_left);
}
```

1.9 Тестування працездатності розробленої гри

Щоб забезпечити належну якість ігрового продукту, процес тестування та відлагодження повинен здійснюватися не лише в ході розробки, а й після завершення кожного з її етапів. У професійній індустрії хорошою практикою є продовження тестування навіть після того, як фінальна версія гри вже відправлена

до публікації або випущена умагазини. Це дозволяє оперативно виявляти та усувати критичні помилки, які могли залишитися непоміченими на етапі внутрішнього контролю.

У випадку створення ігор за допомогою рушія Unity, процес відлагодження та перевірки функціональності реалізовано безпосередньо через інтерфейс редактора. Розробник має можливість у будь-який момент перевести проект у тестовий режим, налаштувавши сцену та об’єкти у потрібному стані, після чого достатньо натиснути кнопку «Play» для запуску симуляції. Цей функціонал використовується як під час активної фази розробки, так і після її завершення – у тому числі для фінального перегляду роботи ігрової логіки або окремих модулів.

Таким чином, Unity надає зручне та інтегроване середовище для багаторазового тестування без необхідності створення окремих збірок проекту, що істотно спрощує пошук і виправлення помилок на ранніх стадіях. Візуалізація процесу представлена на рисунку 1.21, 1.22, 1.23.

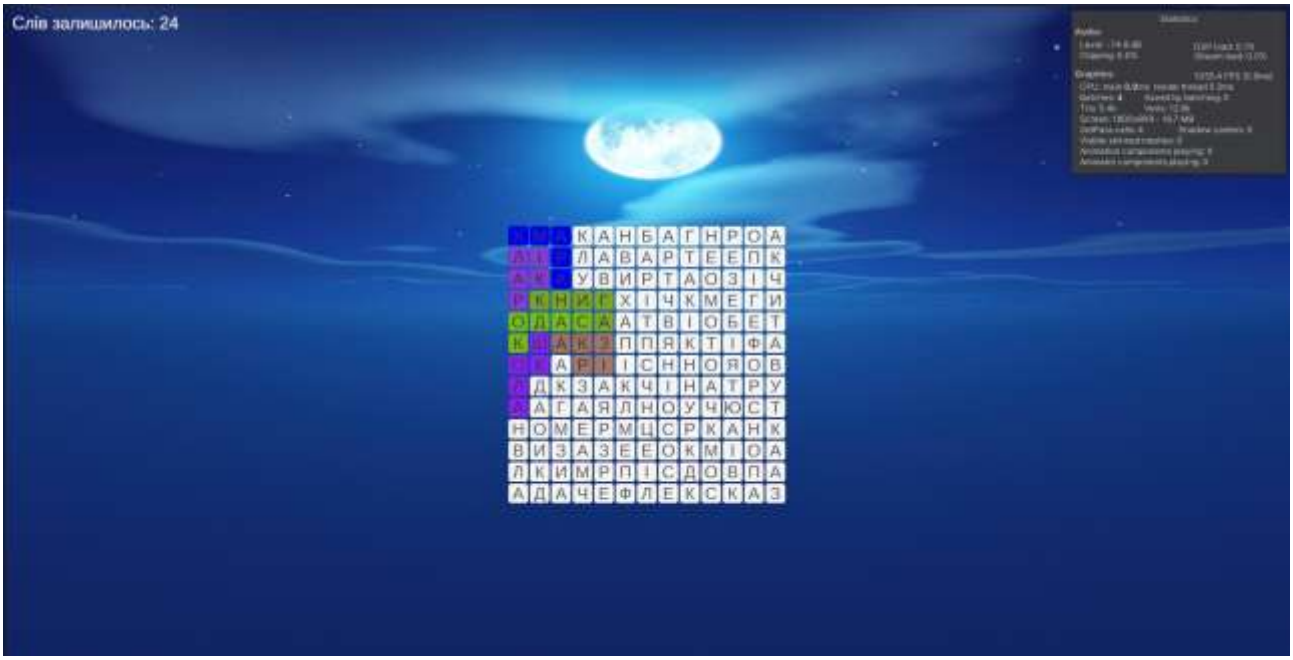


Рисунок 1.21. Процес тестування гри у ігровому програмному рушії Unity

Процес тестування був тісно пов’язаний із логікою вибірки слів із власного словника. Кожний раз доводилось тестувати ігрове поле на реальну випадкову генерацію слів. Окрім того, аналізувалась коректність вибірки нового кольору для наступного шукаємого слова. Іноді складалось враження, що система обирає

однакові кольори і таке можливо, через алгоритм випадкової вибірки. Код було змінено з метою зробити так, щоби наступний колів не був таким же, як попередній. Для реалізації пам’яті всіх кольорів знадобиться створювати окремий перелік використаних кольорів.

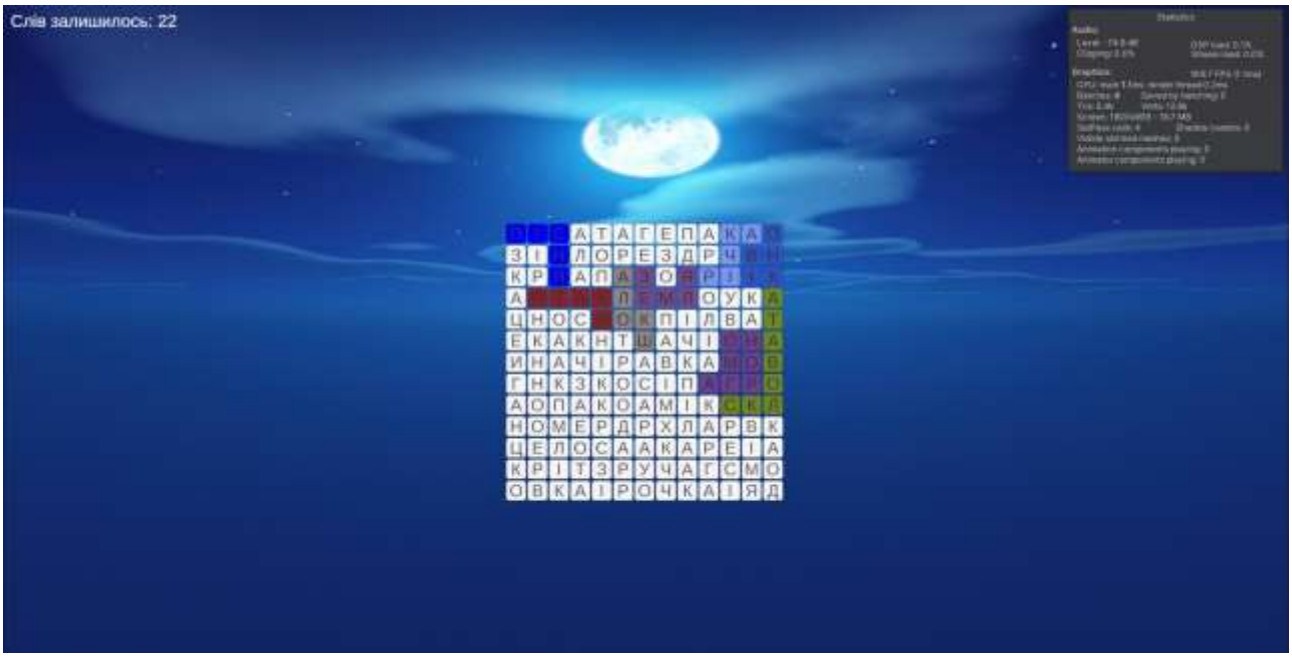


Рисунок 1.22. Процес тестування гри у ігровому програмному рушії Unity

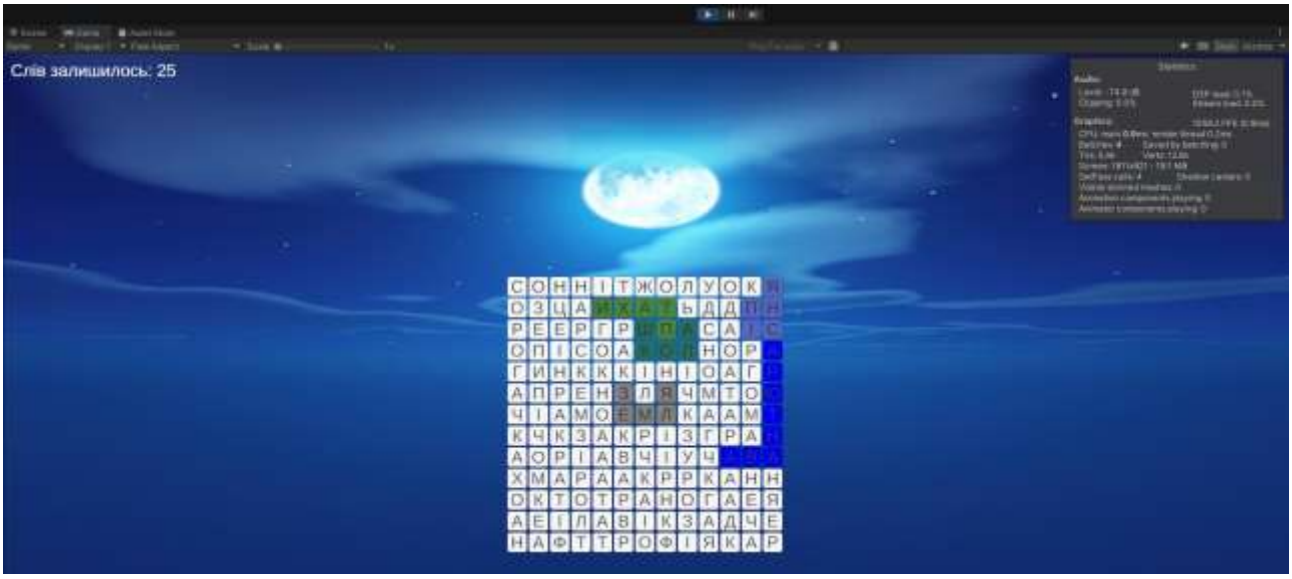


Рисунок 1.23. Процес тестування гри у ігровому програмному рушії Unity

Під час тестування в основному було виявлено помилки пов’язані із логікою взаємодії ігрових об’єктів між собою під час переходу від одного слова до іншого, або від однієї букви до іншої. Ці проблеми вирішувались аналізом посилань у компонентах Модулів логіки клітинок, оскільки саме вони зберігали всі

посилання. Інша серйозна проблема яка стала важкою у аналізі але простою у вирішені – помилка з пустим списком слів. Під час реалізації Модулю власного словника виявилась помилка, коли ігрові об’єкти слів намагались отримати свої слова, гра видавала помилку пов’язану із тим, що у списку із словами не було жодних записів. Досить довго доводилось аналізувати хід виконання ігрової логіки, доки не було знайдено, що ця помилка з’являлась через те, що всі ініціалізації виконувались у методі Start(), який виконується під час створення ігрового об’єкту с компонентом коду, що містить цей метод. І інколи виходило так, що ігрові об’єкти слів з’являлись на сцені раніше, ніж ігровий об’єкт власного словника. Виявилось, що у ігровому програмному рушії Unity є інструмент для створення черги для виконання скриптів. На рисунку 1.24 зображено панель, в якій можна розмітити черговість скриптів на виконання:

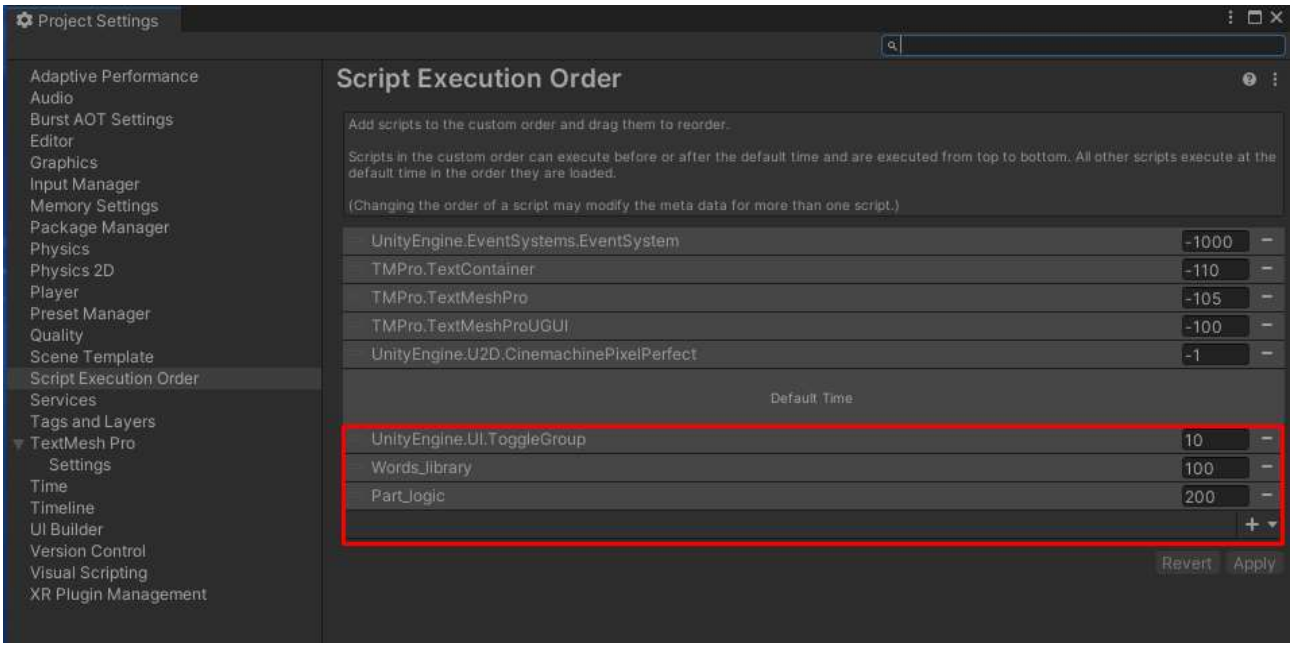


Рисунок 1.24. Панель порядку виконання скриптів

Виставивши генерацію слів до словника гри в першу чергу на виконання, вдалось позбавитись цієї помилки.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Темою мого дипломного проекту була «Розробка гри-головоломки WordLines з вибіркою слів із власного словника». Як результат було отримано гру, яка відповідає поставленій меті. В ході ігрового процесу, користувач повинен знаходити слова у кросворді, відмічати їх та складати кросворд. Кожну нову ігрову сесію, кросворд складається з різних слів, які відбираються із власного словника, який можна змінювати через редактор коду. Створений проект є базовим та має потенціал для модернізації та покращення у багатьох напрямках. Завдяки обраним технологіям розробки, гра може бути портована з ПК на смартфони.

У цьому розділі подано розрахунок вартості створеного програмного забезпечення. Рівень ефективності програмного забезпечення обумовлюється як його якісними властивостями, так і ефективністю процесу розробки. Якість ПЗ оцінюється за різними критеріями: з урахуванням потреб і вражень користувачів, раціональності використання ресурсів та відповідності заданим вимогам. З позиції користувача, визначення якості включає аналіз витрат на розробку, зокрема обсягів трудових витрат і загальної собівартості.

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*2=1,4	$T_{нк}=0,15*N_{тз}$=0,15*2=0,3
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*10=7	$T_{нк}=0,15*N_{рп}$=0,15*10=1,5
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$= 1,5*20=30	$T_{кк}=0,7*N_{тз}$=0,7*20=14	$T_{нк}=0,15*N_{пз}$=0,15*20 =3
Усього, в т.ч.:	159,13		
- на розробку	$\Sigma T_{р}$=106,43		
- контроль керівника		$\Sigma T_{кк}$=43,4	
-нормоконтроль			$\Sigma T_{нк}$=9,3

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	106,43	48,00	5108,64
2.Контроль керівника	43,4	100,00	4340,00
3.Нормоконтроль	9,3	105,00	976,5
Усього	-	-	ΣЗо= 10425,14

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	62	5.0	310,00
Разом	-	-	-	$В_{м\,і}$=310,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 0,1*310=31,0
Усього				$В_м=В_{мі}+В_{тр\ з=}$ 341,00

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	341,0	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	10425,14	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1042,51	$Зд = 0{,}1 \times Зо$ = 10425,14*0,1
4.Відрахування до єдиного фонду соціального внеску	2522,88	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (10425,14+1042,51)
5. Накладні витрати	4170,05	$Внак. = 0{,}4 \times Зо$ =0,4 * 10425,14
6. Повна собівартість	18501,58	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 341,0+10425,14+1042,51+2522,88 +4170,05

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

$$П=(С_{пов}*Р)/100=(18501,58*15)/100=2775,24 \text{ грн.} \quad (2.4)$$

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

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о = С_{пов}+П = 18501,58+2775,24 = 21276,82 \text{ грн.} \quad (2.5)$$

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

$$Ц_р=Ц_о+ПДВ = 21276,82+ 21276,82*0.2 = 25532,18 \text{ грн грн.} \quad (2.6)$$

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.

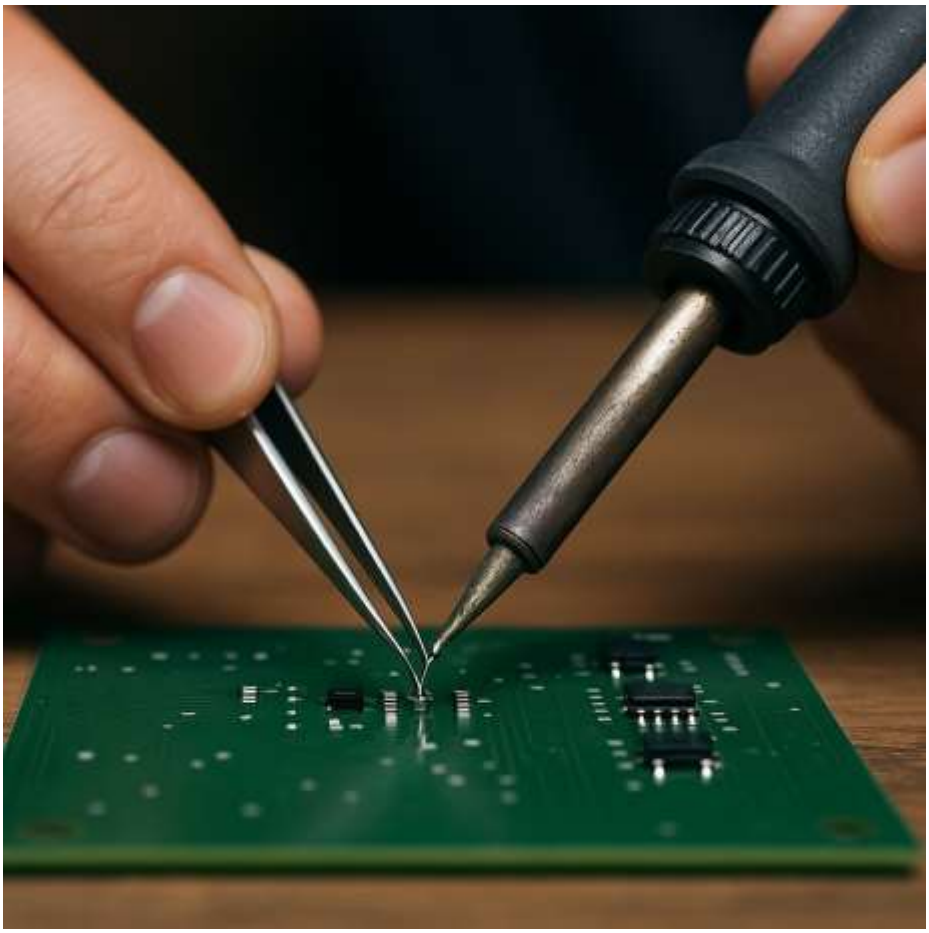


Рисунок 3.1. Процес паяння пристрою

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 зануления,
- 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 (рис.3.2) 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 мого дипломного проекту була «Розробка гри-головоломки WordLines з вибіркою слів із власного словника». Як мету було визначено створення гри для персонального комп'ютера в жанрі словникова головоломка, а основною особливістю мала стати вибірка слів із власного словника. Також було заплановано реалізацію генерації кросвордів з різною постановкою слів.

Для розробки було обрано сучасне програмне забезпечення, як ігровий програмний рушій Unity, редактор коду та середа розробки Microsoft Visual Studio, а також деякі ресурси з Asset Store який є частиною екосистеми Unity.

Процес розробки почався з аналізу ігрового процесу аналогічних ігор для пошуку більш цікавих ігрових механік. У подальшому було створено концепцію ігрового процесу та згідно із цією концепцією спроектовано структуру гри та її основних елементів. Окремо було спроектовано систему вибірки слів з власного словника. Реалізація проекту була виконана з використанням зазначених вище програмних засобів, створено інтерфейс та проведено тестування ігрового процесу на пошук несправностей та помилок.

Як результат було отримано гру, яка відповідає поставленій меті. В ході ігрового процесу, користувач повинен знаходити слова у кросворді, відмічати їх та складати кросворд. Кожну нову ігрову сесію, кросворд складається з різних слів, які відбираються із власного словника, який можна змінювати через редактор коду. Словник реалізовано як частину ігрового проекту, що зменшує кількість необхідного ПЗ для наповнення словника, а також дає змогу створювати нові ігрові режими, додаючи нові властивості до словника.

Створений проект є базовим та має потенціал для модернізації та покращення у багатьох напрямках. Завдяки обраним технологіям розробки, гра може бути портована з ПК на смартфони. Окрім того можна розробити велику кількість модифікацій ігрового процесу, які б надавали ігровим сесіям більше інтерактивності, динаміки та свіжості. Модульна система проекту дає змогу робити це досить швидко та ефективно.

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 Word_class.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Word_class
{
    public string word { get; set; }
    public int word_length { get; set; }
    public bool is_word_in_use { get; set; }
}
```

Скрипт Words_library.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Words_library : MonoBehaviour
{
    List<Word_class> _words = new List<Word_class>();
    private void Start()
    {
        initialize_words_library();
    }
    void initialize_words_library()
    {
        _words.Add(new Word_class() { word = "ГОРА", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "МІСТ", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ЛІТО", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "СНІГ", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ВОДА", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ГІРЯ", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ПТАХ", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "КІНЬ", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ДЕНЬ", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "РУКА", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ЗИМА", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ШИЛО", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ВІЗА", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "КІНО", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ЛЕЗО", word_length = 4, is_word_in_use
= false });
        _words.Add(new Word_class() { word = "ВАГИ", word_length = 4, is_word_in_use
```

```
= false });
                _words.Add(new Word_class() { word = "ТІЛО", word_length = 4, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ТЕМП", word_length = 4, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "КОРА", word_length = 4, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ЛОЗА", word_length = 4, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "КНИГА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ОЗЕРО", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ПІСНЯ", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "САДОК", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ХМАРА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ШКОЛА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ЗЕМЛЯ", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ЗІРКА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ЛІКАР", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "НІЧКА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ПТАХИ", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "РУЧКА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ВІКНО", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "РІЧКА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "СОНЦЕ", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ТРАВА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ПІСОК", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ПІЧКА", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "НОМЕР", word_length = 5, is_word_in_use
= false });
                _words.Add(new Word_class() { word = "ПАРАД", word_length = 5, is_word_in_use
= false });
                _words.Add(new  Word_class()  {  word  =  "АБЕЛІТ",  word_length  =  6,
is_word_in_use = false });
                _words.Add(new  Word_class()  {  word  =  "АБЕТКА",  word_length  =  6,
is_word_in_use = false });
                _words.Add(new  Word_class()  {  word  =  "БАБУСЯ",  word_length  =  6,
is_word_in_use = false });
                _words.Add(new  Word_class()  {  word  =  "БАГНЕТ",  word_length  =  6,
is_word_in_use = false });
                _words.Add(new  Word_class()  {  word  =  "БАКЛАН",  word_length  =  6,
is_word_in_use = false });
```

```
            _words.Add(new  Word_class()  {  word  =  "ВАЛІЗА",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ВАНТУЗ",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ВУЛКАН",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ГАРБУЗ",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ГЕПАРД",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ГРАНІТ",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЖАРГОН",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЖОЛУДЬ",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "МОЛОКО",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "МОНТЕА",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ПАЛАТА",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ПАРОЛЬ",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "СВІЧКА",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "СЛЮСАР",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "СЕСТРА",  word_length  =  6,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "АГРЕСІЯ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "АГРОНОМ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "АТРОФІЯ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "БЕГЕМОТ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "БАБАЙКА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "БАКЛАГА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ВАКУОЛІ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ВЕРТЕЛА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ГРАНЧАК",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ГРАМОТА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ДОМІВКА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ДОРІЖКА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ДЕКАНАТ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЗАГАДКА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЗАПОНКА",  word_length  =  7,
```

```
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЗІРОЧКА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "РЕФЛЕКС",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "РЕЧЕННЯ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ТАКТИКА",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ТВОРЕЦЬ",  word_length  =  7,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ВИКЛАДАЧ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ГІМНАЗІЯ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЗАВДАННЯ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "НАВЧАННЯ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "АВАНТЮРА",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "АНОМАЛІЯ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ШАБЕЛЬКА",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЩОДЕННИК",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЦЕГЕЛЬНЯ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЦЕРКОВКА",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЧАСТОКІЛ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЧЕТВІРКА",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ХОЛОДЕЦЬ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ХАРАКТЕР",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЮНІОНІЗМ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ЮСТУВАТИ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "САГАЙДАК",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "СКЛОВАТА",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "ОКЕАНАФТ",  word_length  =  8,
is_word_in_use = false });
            _words.Add(new  Word_class()  {  word  =  "НОРМАТИВ",  word_length  =  8,
is_word_in_use = false });
        }
        public string get_word_from_list(int p_word_length)
        {
            bool pass = true;
            do
            {
                int word_number;
                word_number = get_random_number(p_word_length);
```

```
            Word_class _word_from_library = _words[word_number];
            if (_words[word_number].is_word_in_use)
            {
                pass = false;
                continue;
            }
            else
            {
                _words[word_number].is_word_in_use = true;
                return _words[word_number].word;
            }
        } while (pass == false);
        return "error";
    }
    int get_random_number(int _word_length)
    {
        switch(_word_length)
        {
            case 4:
                {
                    int number = Random.Range(0, 19);
                    return number;
                }
            case 5:
                {
                    int number = Random.Range(20, 39);
                    return number;
                }
            case 6:
                {
                    int number = Random.Range(40, 59);
                    return number;
                }
            case 7:
                {
                    int number = Random.Range(60, 79);
                    return number;
                }
            case 8:
                {
                    int number = Random.Range(80, 99);
                    return number;
                }
            default:
                return 0;
        }
    }
}
```

Скрипт Part_logic.cs

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
public class Part_logic : MonoBehaviour
{
    [SerializeField] Words_library _words_library;
```

```
        [SerializeField] Gameplay _gameplay;
        [SerializeField] int word_length = 0;
        [SerializeField] List<GameObject> letters_go = new List<GameObject>();
        [SerializeField] string word_in_this_part = System.String.Empty;
        bool is_part_logic_activ = true;
        void Start()
        {
            word_in_this_part = _words_library.get_word_from_list(word_length);
            for (int i=0; i< letters_go.Count; ++i)
            {
                TMP_Text    _button_text    =    letters_go[i].GetComponentInChildren
<TMP_Text>();
                _button_text.SetText(word_in_this_part[i].ToString());
            }
        }
        public void launch_part_logic()
        {
            if(is_part_logic_activ)
            {
                _gameplay.set_pressed_go(gameObject);
                for (int i = 0; i < letters_go.Count; ++i)
                {
                    Button_state    _button_state    =    letters_go[i].GetComponent
<Button_state>();
                    if (_button_state.current_button_state)
                    {
                        Image _current_go_image = letters_go[i].GetComponent<Image>();
                        _current_go_image.color = _gameplay.current_word_color;
                        _button_state.is_letter_colored = true;
                    }
                }
                bool check_if_all_colored = true;
                for (int i = 0; i < letters_go.Count; ++i)
                {
                    Button_state    _button_state    =    letters_go[i].GetComponent
<Button_state>();
                    check_if_all_colored  =  check_if_all_colored  &&  _button_state.
is_letter_colored;
                }
                if (check_if_all_colored)
                {
                    is_part_logic_activ = false;
                    _gameplay.mix_current_color();
                    _gameplay.tape_new_words_count();
                }
            }
        }
        public void reset_all_letters_go()
        {
            if(is_part_logic_activ)
            {
                for (int i = 0; i < letters_go.Count; ++i)
                {
                    Image _current_go_image = letters_go[i].GetComponent<Image>();
                    _current_go_image.color = _gameplay.default_color;
                    Button_state    _button_state    =    letters_go[i].GetComponent
<Button_state>();
                    _button_state.is_letter_colored = false;
                }
```

```
            }
        }
    }
```

Скрипт Button_state.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Button_state : MonoBehaviour
{
    public bool current_button_state { get; set; } = false;
    public bool is_letter_colored { get; set; } = false;
    public void button_pressed()
    {
        current_button_state = true;
        Part_logic _part_logic = GetComponentInParent<Part_logic>();
        _part_logic.launch_part_logic();
        current_button_state = false;
    }
}
```

Скрипт Gameplay.cs

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
public class Gameplay : MonoBehaviour
{
    [SerializeField] TMP_Text word_count_text;
    public Color current_word_color = Color.blue;
    public Color default_color = Color.white;
    GameObject current_go_in_work = null;
    GameObject pressed_go = null;
    int words_left = 30;
    public bool set_pressed_go(GameObject go)
    {
        pressed_go = go;
        if (current_go_in_work == null)
        {
            current_go_in_work = pressed_go;
            return true;
        }
        else if (current_go_in_work != pressed_go)
        {
            //delete all colors from buttons in and color new buttons is working
part
            Part_logic _part_logic = current_go_in_work.GetComponent<Part_logic>();
            _part_logic.reset_all_letters_go();
            current_go_in_work = pressed_go;
            return true;
        }
        else if (current_go_in_work == pressed_go)
        {
            //coloring logic in button
            return true;
        }
        else
        {
```

```
            return false;
        }
    }
    public void mix_current_color()
    {
        current_word_color.r = Random.Range(0.1f, 0.70f);
        current_word_color.g = Random.Range(0.1f, 0.80f);
        current_word_color.b = Random.Range(0.1f, 0.99f);
    }
    public void tape_new_words_count()
    {
        --words_left;
        word_count_text.SetText("Ñë³â çàëèøèëîñü: " + words_left);
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Розробка гри-головоломки WordLines з вибіркою слів із власного словника

Дипломний проект студента групи 4РП-08: Петровця Віктора Олександровича

Керівник: Джабраілов Д.В.

Аналіз аналогічних ігор

Скріншот ігрового процесу гри Fill The Words: Themes search

Скріншот ігрового процесу веб-додатку thewordsearch.com

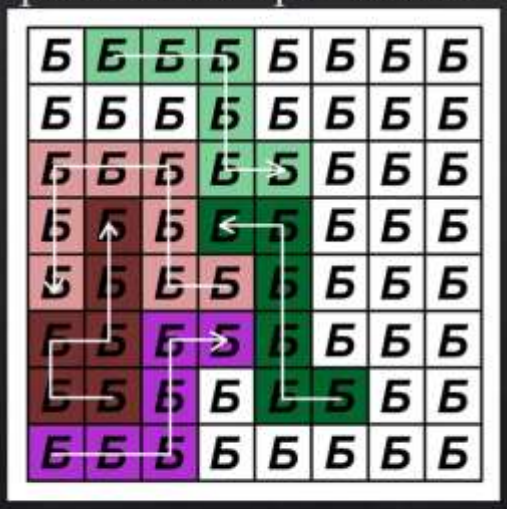
Проект-референс для формування концепту гри

Скріншот ігрового процесу гри Філворди: Гра в Слова з Букв

Концепція гри WordLines

Базова ідея гри	Базова ідея проекту складається у реалізації гри в жанрі філворд з додаванням для гравця можливості реіграбельності
Ігрові механіки	Ігровими механіками є класичні правила ігор в жанрі філворд
Сюжетна структура та персонажі	Гра не буде мати сюжетної структури або персонажів на даному етапі розробки. Можливо, у майбутньому можна імплементувати цю складову для більшої зацікавленості у грі
Графіка та звук	Графіка представлена у простому 2D виконанні із використанням унікальних скайбоксів. Під час ігрового процесу виділені букви будуть виділятись в один колір створюючи різноманітні кольорові схеми
Технічна реалізація	Цільовою платформою для розробляємого проекту є персональний комп'ютер

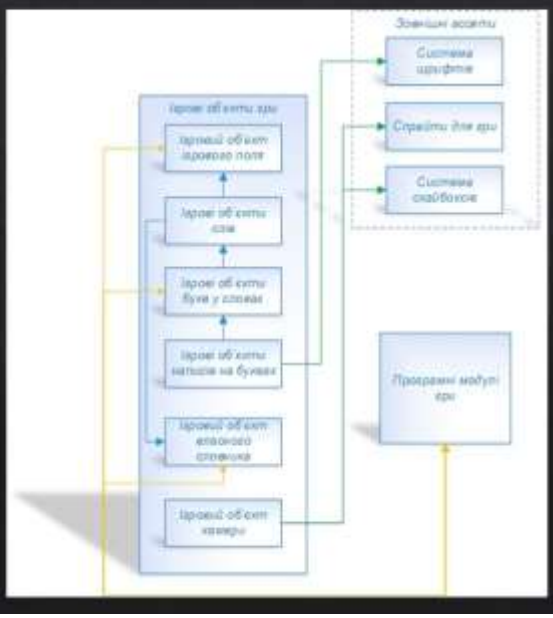
Концептуальне зображення ігрового процесу розробляємої гри WordLines



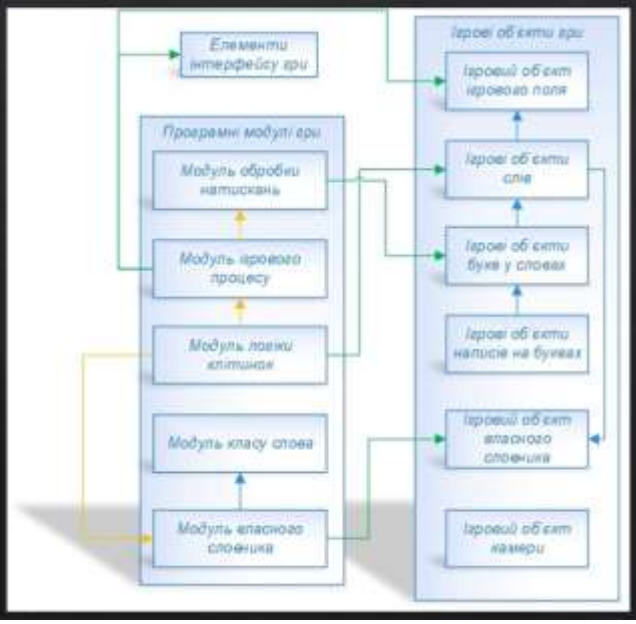
Структура розробляємого проекту

Структура розробляємого проекту

Огляд основних елементів гри



Програмні модулі та їх зв'язки з іншими елементами гри WordLines



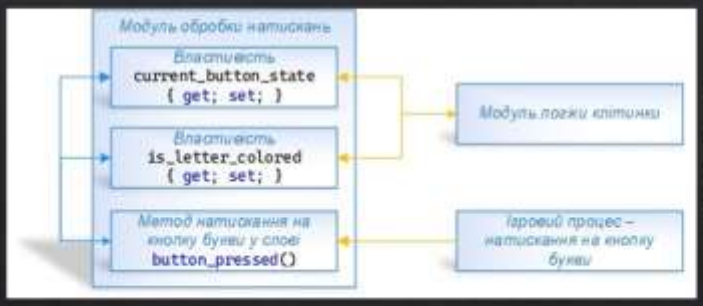
Реалізація скайбоксів та ігрового поля

Створені текстури та матеріали скайбоксів для розробляємої гри

Реалізоване ігрове поле з розміщеними на ньому об'єктми-словами

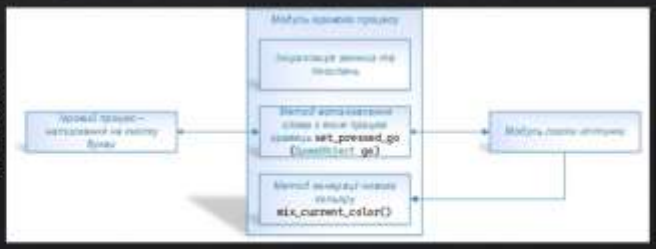
Ієрархія об'єктів з усіма необхідними для гри об'єктами

Реалізація основних елементів гри



Структура Модуля обробки натискань та його зв'язки з іншими елементами розробляємої гри

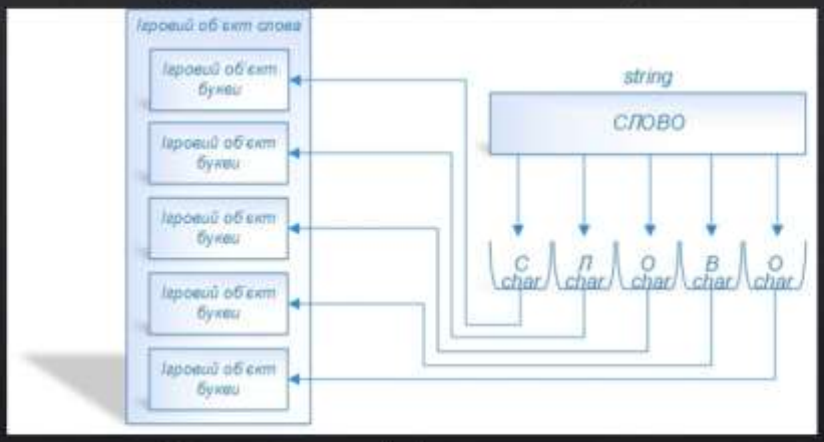
Структура Модуля ігрового процесу та його зв'язки з іншими елементами грозробляємої гри



Огляд вибірки слів із власного словника

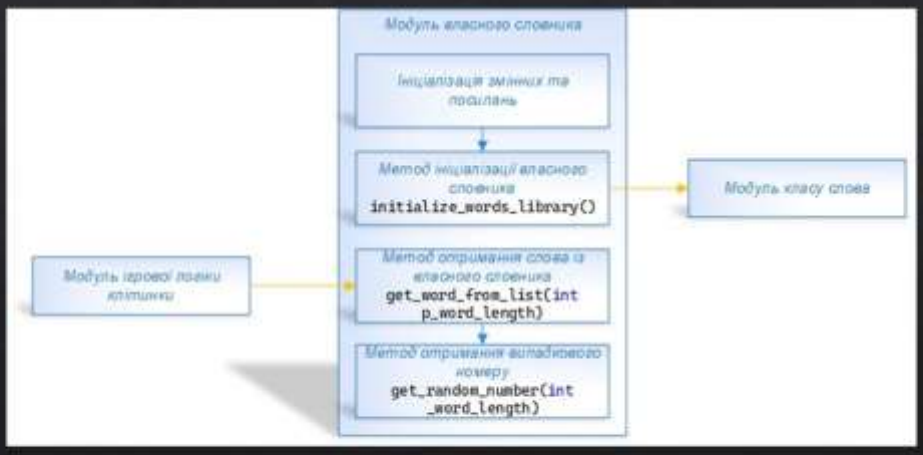
Схема роботи вибірки слів із власного словника

Зв'язок структури слів та ігрового поля

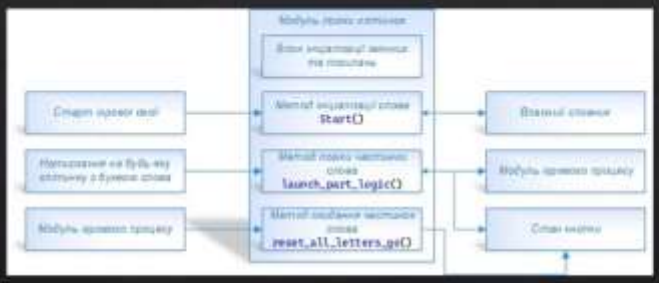


Зв'язок структури слів із власного словника та ігрових об'єктів на ігровому полі

Структура Модуля власного словника та його зв’язки з іншими елементами гри



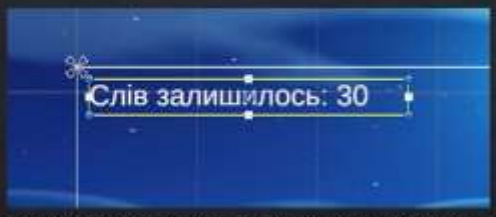
Реалізація вибірки слів із власного словника



Структура Модуля логіки клітинок та його зв’язки з іншими елементами розробляємої гри

Всі поля Модуля логіки клітинок, які формуються у відповідним скриптом ігрової логіки

Реалізація інтерфейсу гри



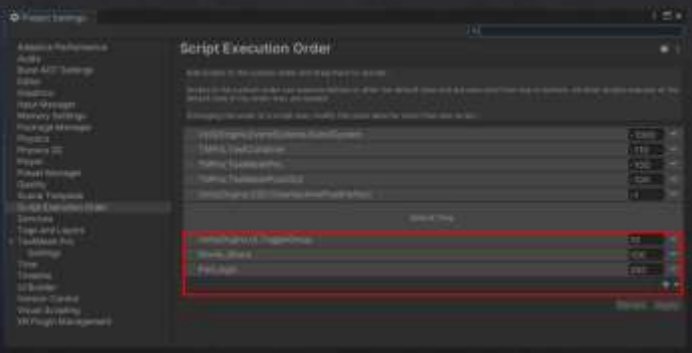
Текстовий елемент розміщений на інтерфейсі гравця

```
public class Gameplay : MonoBehaviour
{
    [SerializeField] TMP_Text word_count_text;

public void tape_new_words_count()
{
    --words_left;
    word_count_text.SetText("Слів залишилось: " + words_left);
}
```

Код оновлення значення кількості слів у інтерфейсі гравця

Тестування гри



Панель порядку виконання скриптів

Скріншоти гри 1

Скріншоти гри 2

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти
відділення комп'ютерних систем

Петровець Віктор Олександрович

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Джабраілов Дмитро Володимирович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка гри-головоломки WordLines з вибіркою слів із власного словника*

Обсяг розрахунково-пояснювальної записки *81* сторінок

Обсяг графічної (презентаційної) частини *18* 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)

Детально описано процес виконання розробки гри;
Виконано проектування елементів гри із поясненнями на схемах та за допомогою коду;
Створено свою систему для введення та збереження слів у словник гри.

д) основні недоліки дипломного проекту (роботи)

Опис реалізації окремих систем (системи атаки, роботи інтерфейсу або механіки руху) містить дублювання коду. Випадковість вибору слів на основі Random є досить передбачуваною. Необхідно більше аналізувати, як усі компоненти взаємодіють між собою, та провести глибше тестування балансу (як наприклад, співвідношення шкоди, перезарядок і рухових характеристик).

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

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

Підпис:

« *20* » *червня* 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проєкт здобувача (здобувачки) освіти

відділення комп'ютерних систем

Петровець Віктор Олександрович

(прізвище, ім`я та по батькові)

Спеціальність: *121 " Інженерія програмного забезпечення "*

Освітньо-професійна програма: *«Розробка програмного забезпечення»*

Тема дипломного проєкту: *Розробка гри-головоломки WordLines з вибіркою слів із власного словника*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ

а) обсяг і якість виконання проєкту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проєкт виконано відповідно технічному завданню. Пояснювальна записка містить 81 сторінки. У пояснювальній записці виконано опис етапів розробки комп'ютерної гри для ОС Windows, її концепції, виконано проектування елементів гри, та описана їх реалізація. Графічна частина складається з 18 слайдів мультимедійної презентації, які також містять графічні матеріали, передбачені технічним завданням. Якість виконання пояснювальної записки відмінна, графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проєктом: *Протягом всього строку дипломного проєктування та переддипломної практики здобувач освіти Петровець В.О. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Петровець В.О. під час роботи над дипломним проєктом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою. Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проєкту*

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

Ми, що нижче підписалися,

Петровець В.О.
здобувач освіти гр. 4РП-08, та

Джабраілов Д.В.,
керівник дипломного проекту,

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

«Розробка гри-головоломки WordLines з вибіркою слів із власного словника» (автор роботи – Петровець В.О., керівник роботи – Джабраілов Д.В.)

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 _____ / Петровець В.О. /

Керівник _____ / Джабраілов Д.В. /

«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 4РП-08

Петровця Віктора Олександровича

на тему *Розробка гри-головоломки WordLines з вибіркою слів із власного словника*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з несуттєвими 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 7,74%, коефіцієнт цитування – 1,36%.*

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

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Петровця В.О.* (П.І.Б.)

проведена «*16*» *червня* 20*25* 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

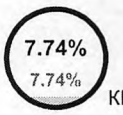
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка гри-головоломки WordLines з вибіркою слів із власного словника

Автор
Петровець Віктор Олександрович
Науковий керівник / Експерт
Джабраілов Дмитро Володимирович

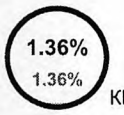
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

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
Довжина фрази для коефіцієнта подібності 2

15537
Кількість слів

114948
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	17
Інтервали	A⟶	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	50

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 0.53 %
2	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	75 0.48 %
3	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	64 0.41 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	40 0.26 %
5	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	40 0.26 %

6	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	37 0.24 %
7	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	35 0.23 %
8	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	33 0.21 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	29 0.19 %
10	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	29 0.19 %

з домашньої бази даних (0.38 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	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")	31 (4) 0.20 %
2	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")	28 (3) 0.18 %

з програми обміну базами даних (0.12 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Методичні рекомендації.docx [redacted] 1/12/2023 Zhytomyr Ivan Franko State University (ZIFSU)	10 (1) 0.06 %
2	Відновлення деталей та ремонт понтонних мостів [redacted] 5/22/2019 National University Chernihiv Politechnika (NUCP) 2 (Наукова бібліотека)	9 (1) 0.06 %

з Інтернету (7.24 %)

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**	**256 (8) 1.65 %**
2	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**151 (5) 0.97 %**
3	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**109 (3) 0.70 %**
4	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**92 (5) 0.59 %**
5	https://www.kadrovik.ua/novyny/zvertayemo-uvagu-na-umovy-praci-pid-chas-vykonannya-zvaryuvalnyh-robit	73 (6) 0.47 %
6	**https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download**	**67 (3) 0.43 %**
7	**https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download**	**62 (3) 0.40 %**
8	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	48 (5) 0.31 %
9	https://knute.edu.ua/file/NjI4Nw==/354eadce03a1a6cfc898e26acceeea0d.pdf	42 (3) 0.27 %

10	**https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download**	**35 (1) 0.23 %**
11	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	26 (2) 0.17 %
12	https://stackoverflow.com/questions/30056471/how-to-make-the-script-wait-sleep-in-a-simple-way-in-unity	25 (2) 0.16 %
13	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	21 (1) 0.14 %
14	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	21 (2) 0.14 %
15	http://uadoc.zavantag.com/text/18266/index-1.html	15 (1) 0.10 %
16	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	15 (1) 0.10 %
17	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	14 (2) 0.09 %
18	https://gist.github.com/misho-kr/be0cf68a9d1f5f0ffcf252dabbc7ca6d	14 (1) 0.09 %
19	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	9 (1) 0.06 %
20	https://stackoverflow.com/questions/70196826/how-to-load-next-scene-when-pressed-a-button-while-playing-in-unity	9 (1) 0.06 %
21	https://card-file.ontu.edu.ua/server/api/core/bitstreams/4bb7255e-46d4-4349-9726-9698476da02d/content	6 (1) 0.04 %
22	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	5 (1) 0.03 %
23	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	5 (1) 0.03 %
24	https://card-file.ontu.edu.ua/server/api/core/bitstreams/3302e08a-9549-43ba-8861-728bff7dc7ff/content	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: 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма: **«Розробка програмного забезпечення»**
Група: 4РП- 08

Дипломний проєкт
здобувача освіти денної форми навчання
РП.08.16.000.ДП

ПЕТРОВЦЯ
ВІКТОРА ЛЕКСАНДРОВИЧА

м. Одеса
2025 р.

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

Спеціальність: 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма: «Розробка програмного забезпечення»
Група: 4РП- **08**

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проєкту на тему: