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

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма: «Розробка програмного забезпечення»

Група: 4РП-07

Дипломний проект

здобувача освіти денної форми навчання
РП.07.24.000.ДП

ЩЕРБАКОВА
ДМИТРА СЕРГІЙОВИЧА

м. Одеса
2024 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-07**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка гри «Японський кросворд» на програмному рушії Unity

Проектний матеріал складається з пояснювальної записки на 69 сторінках та графічного (презентаційного) матеріалу на 12 аркушах (слайдах)

Дипломник _____ (Щербаков Д.С.)

Керівник _____ (Суліма Ю.Є.)

Консультанти:

з економічного розділу _____ (Іванченков В.С.)

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 _____ (Скорнякова О.В.)

Захист «25» 06 2024 р. Протокол ЕК № 2

Оцінка ЕК 5 (п'ять) 1955

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.*

"15" 01 2024 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Щербакову Дмитру Сергійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка гри «Японський кросворд» на програмному рушії Unity*

затверджена наказом по коледжу від "02" 11 2023 р. № 244-АД-ОА

2. Термін здачі закінченого проекту 10.06.2024

3. Вихідні данні до проекту *Використання програмного рушія Unity; Використання мови програмування C# та бібліотек Unity для розробки гри; Використання принципів модульності у проектуванні та розробці ігрових проектів; Реалізація основних ігрових механік комп'ютерної гри «Японський кросворд»; Гра повинна виконувати основні правила вирішення японських кросвордів; Гра повинна мати інтерфейс; Гравець повинен мати змогу обирати клітинки для замальовування, відмічати клітинки.*

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

Аналіз ігрового процесу японських кросвордів; Аналітичний огляд програмних рушіїв з умовно безкоштовним доступом; Вибір інструментів розробки; Формування концепту ігрового процесу гри; Проектування основних елементів ігрового процесу та принципи їх роботи; Реалізація основних елементів ігрового процесу; Тестування працездатності ігрових елементів.

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

Особливості ігрового процесу японських кросвордів; Особливості програмного рушія Unity та причини його вибору для розробки проекту; Особливості ігрових механік розроблюємого проекту; Огляд основних елементів ігрового процесу; Принцип роботи основних елементів ігрового процесу; Етапи реалізації основних елементів ігрового процесу; Хід тестування гри, Скріншоти розробленої гри.

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

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

7. Дата видачі завдання 15.01.2024

Керівник *Суліма Ю.Є.* _____
(підпис)

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

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	*20.05.2024*	виконав
2	*Аналіз ігрового процесу розв'язання японських кросвордів*	*23.05.2024*	виконав
3	*Аналітичний огляд та вибір програмного рушія*	*25.05.2024*	виконав
4	*Формування концепту ігрового процесу гри*	*28.05.2024*	виконав
5	*Проектування основних елементів ігрового процесу*	*30.05.2024*	виконав
6	*Проектування графічної частини гри*	*01.06.2024*	виконав
7	*Реалізація графічної частини гри*	*03.06.2024*	виконав
8	*Реалізація основних елементів ігрового процесу*	*05.06.2024*	виконав
9	*Імплементація та відлагодження ігровіх елементів*	*07.06.2024*	виконав
10	*Тестування працездатності елементів гри*	*09.06.2024*	виконав
11	*Виправлення виявлених помилок*	*10.06.2024*	виконав
12	*Аналіз результатів, підготовка слайдів презентації*	*11.06.2024*	виконав
13	*Економічні розрахунки та питання з охорони праці*	*12.06.2024*	виконав
14	*Підготовка графічної частини проекту*	*13.06.2024*	виконав
15	*Підготовка проекту до захисту та тестування ПП*	*14.06.2024*	виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 07. 24 000. 00 ДП	Дипломний проект		
А4			РП 07. 24 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 07. 24 000. 00 ДП		
Розробив		Щербаков Д.С.		10.06	Розробка гри «Японський кросворд» на програмному рушії Unity	Літ.	Аркуш 4 / Аркушів 69
Перевірив		Суліма Ю.Є.		10.06		Н Д П	
Н. контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4РП-07	
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

Досить давно людство отримало звичку якимось чином займати свій мозок у часи, коли потрібно щось чекати, або є вільний час. Для одних це приємне дозвілля, а для інших професійний інтерес. Серед способів займати мозок є такі ігри як судоку, кросворди різних типів та інші головоломки. Одні ігри розвивають пам'ять, а інші увагу чи здібність рахувати.

Темою мого дипломного проекту є «Розробка гри Японський кросворд на програмному рушії Unity». Японські кросворди обрані через швидкість їх вирішення – за умовою знання правил – та користь, яку вони несуть для людини. Окрім того створення такої гри допоможе створити для людини приємне дозвілля. Розробка саме гри на програмному рушії Unity я вважаю актуальною через більшу портативність такого рішення. Дуже часто, японські кросворди розміщують в великих журналах, які не дуже зручно переносити з собою. Але маючи гру на смартфоні, або портативному ігровому пристрої, можна легко грати у будь-якому місці. Створена гра має бути «легкою» для виконання на більшості пристроїв та простою у використанні, водночас я маю мету створити ігровий процес наближений до того, що людина відчуває вирішуючи японські кросворди у реальному житті.

Окрім того написання такої гри відкриває можливість у подальшому розвинути проект для створення спільноти гравців. У майбутньому, базуючись на існуючому проекті, можна розширити його для додавання соціальних елементів таких як рейтингова дошка, модифікатори складності, різні варіанти кросвордів, проведення заходів, тощо.

Отже створення такої гри є базою для подальшого розвитку великого проекту. Використання ігрового рушія Unity відкриває легку дорогу у розробці гри з таким напрямом. Також, реалізація теми дипломного проектування дає змогу розвити та закріпити навички, пов’язані із створенням ігрових додатків для будь-яких платформ.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз ігрового процесу японських кросвордів

Для виконання мети дипломного проектування, необхідно створити комп'ютерну гру Японський кросворд. Для цього необхідно визначити основні риси даного різновиду кросвордів, як вони вирішуються, як вони будуються.

Японські кросворди мають не таку велику популярність, як класичні сканворди, або навіть судоку. Це виходить через те, що для багатьох людей правила вирішення японських кросвордів здаються дуже складними. Такі кросворди вперше з'явились у 20-му сторіччі, втім точно не відомо, хто був автором першого кросворду такого типу. Спочатку японські кросворди були названі нонограмами, як результат наукової роботи Нон Ісіди, яка намагалась за допомогою таких нонограм створити засіб спілкування між людьми та тваринами. Трохи згодом, коли Нон Ісіда використавши таку нонограму прийняла участь у конкурсі малюнків вікнами, отримала перше місце та отримавши натхнення цим створила ще 3 нонограми, як головоломки.

Паралельно із цим інший автор головоломок Тецуя Нісіо вигадав головоломку «Малювання числами». Світову популярність, у своєму кінцевому вигляді, японські кросворди отримали після публікації у газеті Telegraph в 1989-1990 роках. Треба зазначити, що іменування цих головоломок «Японський кросворд» – це особливість країн СНД.

Правила розв'язування японських кросвордів прості. Є три поля. Поле з малюнком, який з'явиться при вірному вирішенні кросворду. Це поле ділиться на клітинки, утворюючи мозаїку. Також є вертикальне поле з числами, які показують скільки у даній горизонтальній лінії замальованих ліній та скільки в цих лініях клітинок. Відповідно є горизонтальне поле з числами, яке показує скільки у даній вертикальній лінії замальованих ліній та скільки в цих лініях клітинок. Окремі лінії завжди мають між собою хоча б одну пусту клітинку. Таким чином процес вирішення японського кросворду зводиться до зіставлення чисел та клітинок у вертикалях та горизонталях, що дещо нагадує вирішення головоломок «Судоку».

На рисунку 1.1 ви можете бачити приклад вирішеного японського кросворду.

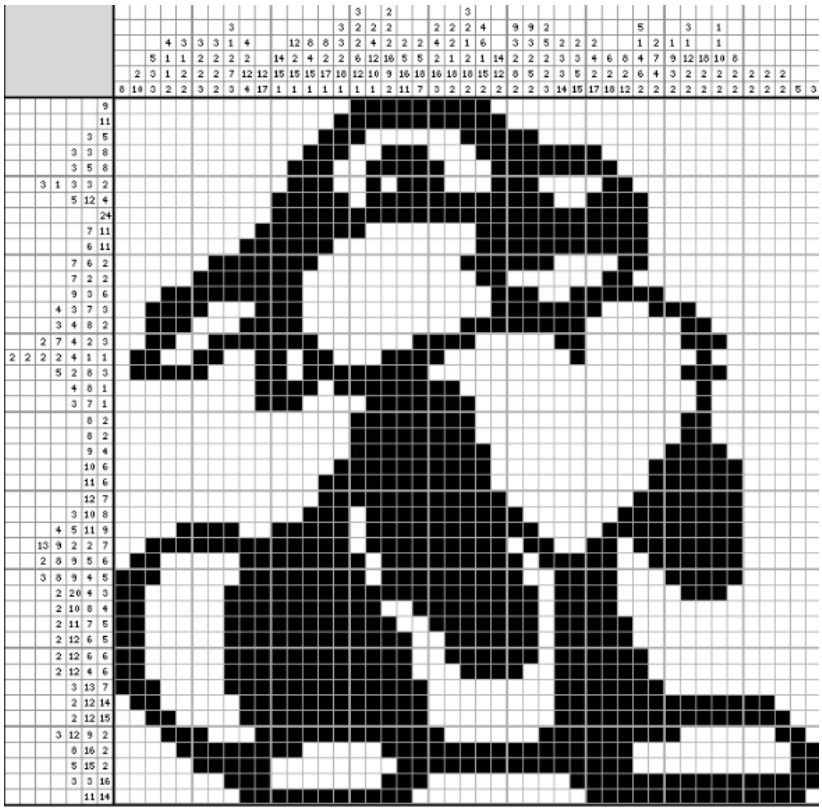


Рисунок 1.1. Приклад вирішеного японського кросворду

Окрім чорно-білих японських кросвордів, або нонограм, існують також і кольорові версії таких кросвордів. Вони не відрізняються від чорно-білих нонограм за будовою поля, але відрізняються правилами штрихування. Так, наприклад, у чорно-білих кросвордах заштриховані лінії, або комірка, не можуть бути поряд одна з одною, тому між ними завжди буде комірка з хрестиком. В той же час у кольорових японських кросвордах, лінії та комірки можуть знаходитись відразу одна за одною при умові, що кольори ліній та комірок не співпадають. Тому, суб'єктивно, такі японські кросворди розв'язувати дещо складніше, оскільки довжина штрихування ліній може бути на всю горизонталь, або вертикаль. На рисунку 1.2 можна побачити приклад розв'язаного кольорового японського кросворду.

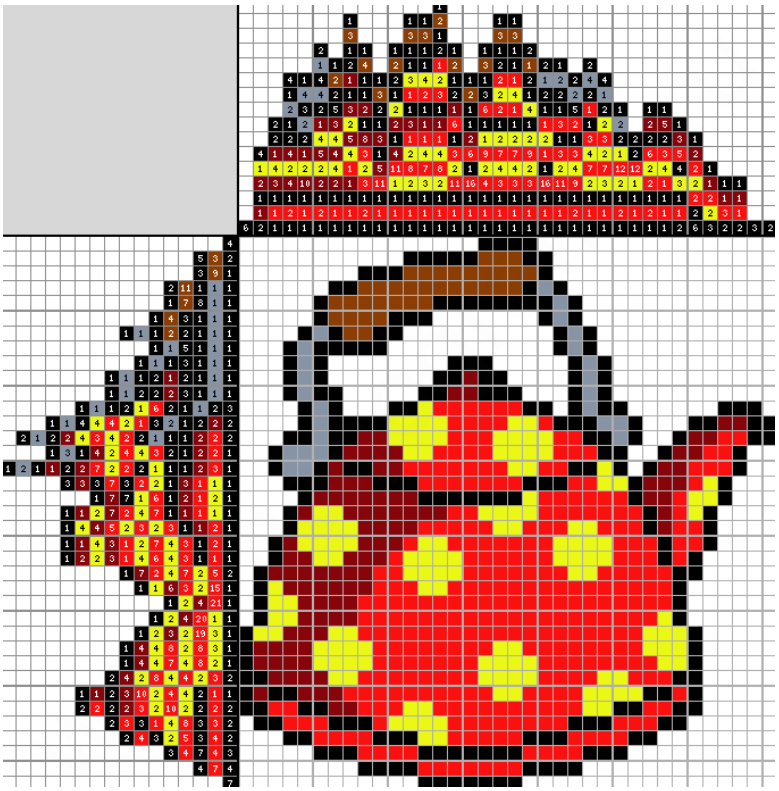


Рисунок 1.2 Приклад розв’язаного кольорового японського кросворду

Для того, щоби більш точно відтворити ігровий процес у реалізовуємому проекті, необхідно також відмітити деякі особливості вирішення цих кросвордів. Є декілька видів штрихування клітинок. Чистове штрихування – коли людина впевнена, що ця клітинка має бути заштрихованою. Чистове викреслювання – коли людина впевнена, що ця клітинка має залишатись пустою. Позначка-здогадка – коли немає повної впевненості в тому, що ця клітинка має бути заштрихована. Ці елементи ігрового процесу необхідно також додати до розробляємої гри.

1.2 Аналітичний огляд програмних рушіїв з умовно безкоштовним доступом

Для виконання цього дипломного проекту було проведено попередній аналіз програмних засобів, які дозволять розробляти комп'ютерні ігри на умовно безоплатній основі. Існує ряд таких програмних засобів, але з них можна виділити три основні, які конкурують між собою, і необхідно було вибрати один з них.

CryEngine – це потужний ігровий програмний рушій, розроблений німецькою компанією Crytek. Crytek – це німецька компанія, заснована братами

Джеваном, Авні та Фарук Йерлі у 1999 році. Сам ігровий програмний рушій відомий своїми вражаючими графікою та фізикою, а також широкими можливостями для створення ігрових світів. CryEngine був використаний у різних популярних іграх, таких як серія Crysis, Ryse: Son of Rome та Hunt: Showdown. Початково отримав відомість як ігровий програмний рушій, що відтворює неймовірно реалістичну графіку, а також правдоподібну симуляцію фізики із симуляцією руйнування довкілля. Через свої неймовірні візуальні та фізичні показники отримав відомість, як бєнчмарк для комп'ютерів того часу. На рисунку 1.3 можна побачити приклад графіки, що відтворює програмний рушій CryEngine.

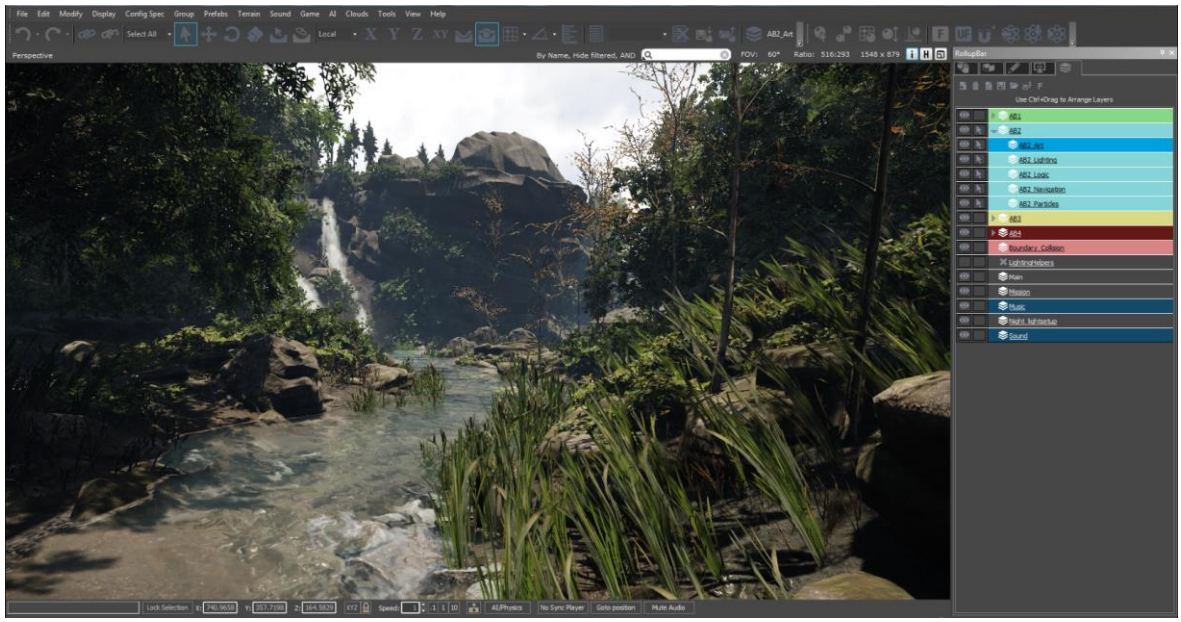


Рисунок 1.3. Приклад графіки, що відтворює програмний рушій CryEngine

З точки зору можливостей, то однією з ключових особливостей CryEngine є його підтримка технології трасування променів (ray tracing), яка дозволяє досягти фотореалістичних ефектів освітлення та відображення. Крім того, CryEngine має високий рівень масштабованості, що дозволяє створювати як невеликі інді-проекти, так і великі ААА-ігри. Програмний рушій також забезпечує розробників інструментами для створення інтерактивного та живого ігрового середовища, включаючи систему штучного інтелекту, керування анімацією персонажів, а також інструменти для створення звукової атмосфери. CryEngine також пропонує гнучку систему багатоплатформної розробки, що дозволяє випускати ігри на різних платформах, включаючи ПК, консолі та мобільні пристрої. На рисунку 1.4

можна побачити приклад роботи з вбудованим аніматором рушія CryEngine.

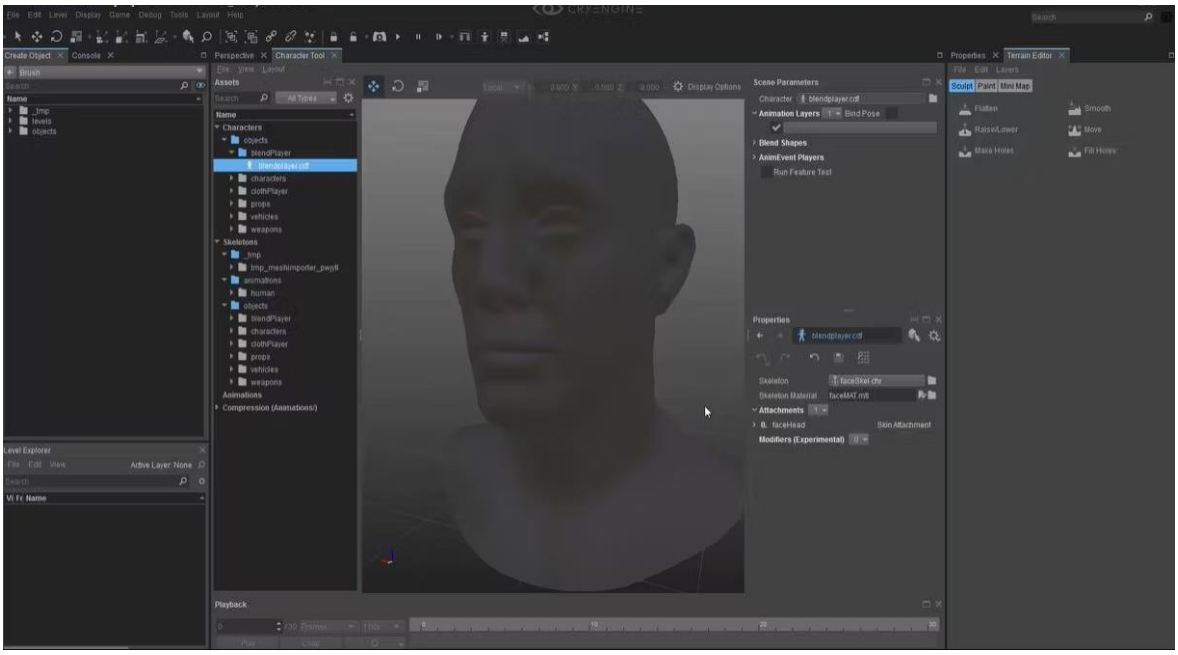


Рисунок 1.4. Приклад роботи з вбудованим аніматором рушія CryEngine

Загалом ігровий програмний рушій CryEngine є безкоштовним, втім система побудована таким чином, що використовувати індивідуальну підтримку, бібліотеку ассетів, можна лише з покупкою ліцензії. Окрім того, її все одно необхідно буде купувати, якщо кількість фінансового обороту користувача програмного рушія перевищить визначеного рівня.

Ще одним розглянутим ігровим програмним рушієм є Unreal Engine. Це потужний і широко використовуваний ігровий програмний рушій, розроблений компанією Epic Games. Він надає розробникам інструменти для створення високоякісних ігор та візуалізацій, а також розробки віртуальної реальності (VR), доповненої реальності (AR) та багатьох інших інтерактивних проектів. Unreal Engine пропонує широкий набір функцій, включаючи потужний рушій графіки, підтримку фізичного рушія, інструменти для роботи зі штучним інтелектом, анімацією персонажів, звуком та багатьом іншим. Він також включає графічний інтерфейс користувача, який спрощує процес розробки та створення контенту. На рисунку 1.5 можна побачити приклад роботи із анімацією у ігровому рушії Unreal Engine.

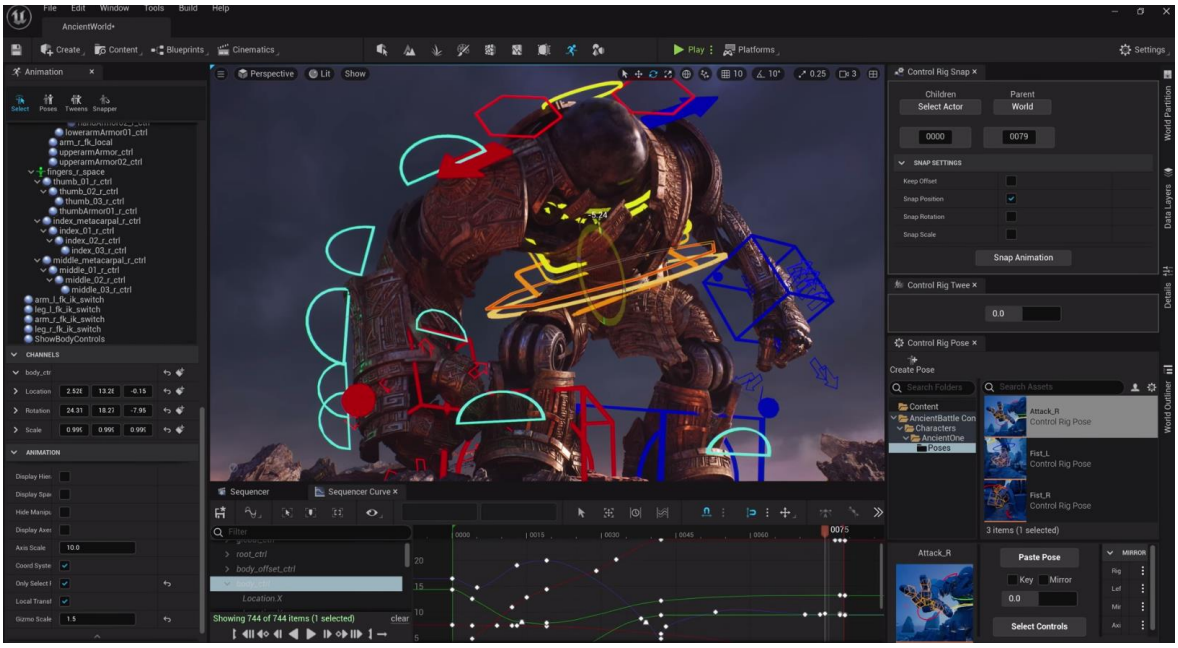


Рисунок 1.5. Приклад роботи із анімацією у ігровому рушії Unreal Engine

Однією з ключових особливостей Unreal Engine є його графічний рушій, який підтримує передові технології, такі як трасування променів (ray tracing), що дозволяє створювати неймовірно реалістичні графічні ефекти. Це робить Unreal Engine популярним вибором для розробників ААА-ігор, а також для створення вражаючих візуалізацій та симуляцій в інших галузях. На рисунку 1.6 можна побачити приклад графіки яку відтворює Unreal Engine.

Рисунок 1.6. Приклад графіки яку відтворює Unreal Engine

З точки зору ліцензування, то Unreal Engine є безкоштовним, проте не дає доступу до коду самого ігрового програмного рушія, деякі платформи розробки та можливості заблоковані, а також закритий доступ до виділеного середовища

розробників та підтримки.

Останнім проаналізованим двигуном є Unity. Це популярний багатоплатформний ігровий програмний рушій та середовище розробки, створений компанією Unity Technologies. Він надає розробникам інструменти для створення 2D та 3D ігор, додатків віртуальної та доповненої реальності, симуляцій, візуалізацій та багато іншого. Unity відрізняється простотою використання та доступністю, що робить його привабливим вибором для новачків у розробці ігор, а також для досвідчених розробників. Він надає зручний графічний інтерфейс, безліч готових ресурсів та компонентів, а також потужну мову програмування C# для створення інтерактивності та логіки ігор. На рисунку 1.7 можна побачити приклад роботи у ігровому програмному рушії Unity.

Рисунок 1.7. Приклад роботи у ігровому програмному рушії Unity

Однією з ключових особливостей Unity є багатоплатформність. Він підтримує різні платформи, включаючи ПК, консолі, мобільні пристрої, віртуальну реальність та доповнену реальність, що дозволяє розробникам досягати більшої аудиторії та випускати свої продукти на різних пристроях.

З точки зору ліцензування, то ігровий програмний рушій надає повний доступ до свого функціоналу безкоштовно, для створення ігор будь-якого рівня. Необхідність покупати ліцензію з’являється лише якщо сума зароблених грошей

на користувача дійде до рівня у двісті тисяч доларів.

Проаналізувавши ігрові програмні рушії було вирішено використовувати рушій Unity. Рішення було зумовлено в першу чергу доцільністю використання можливостей ігрових рушіїв. CryEngine та Unreal Engine створені у передусім для відтворення дуже красивих з графічної точки зору проектів. Від того йде складність використання функціоналу та роботи із простою 2D-графікою. В той же час Unity в тому числі створювався для невеликих, простих та легких проектів, що відповідає грі дипломного проектування.

1.3 Вибір інструментів розробки

Виконавши огляд програмних рушіїв, необхідно обрати інструменти для розробки гри. Основним інструментом розробки гри має бути середа розробки – редактор коду. Редактор коду – це програмне забезпечення, призначене для написання, редагування та форматування вихідного коду комп'ютерних програм. Редактори коду зазвичай надають зручний інтерфейс користувача, підсвічування синтаксису, автоматичне завершення коду, функції пошуку та заміни, а також можливість інтеграції з іншими інструментами розробки, такими як системи керування версіями, налагоджувачі та компілятори. Деякі популярні редактори коду включають Visual Studio Code, Sublime Text.

Обираючи серед великої кількості програмного забезпечення для редагування коду, я зупинився на Microsoft Visual Studio. Такий вибір був зроблений з розрахунку на те, що ігровий рушій Unity працює з мовою програмування C#, а це значить, що потрібно обрати таку середу розробки, що могла б не тільки підказувати із форматуванням коду, але й з зв'язками з іншими класами, методами, ігровими об'єктами та бібліотеками. Окрім того, використання Microsoft Visual Studio дає змогу отримати доступ до проекту гри. Також, використання цієї середи розробки – редактору коду – рекомендована розробниками та підтримкою Unity.

Для роботи з графікою необхідно знайти графічний редактор. Графічний редактор – це програмне забезпечення, призначене для створення та редагування графічних зображень. Він дозволяє користувачам створювати малюнки,

ілюстрації, фотографії та інші види графіки за допомогою різних інструментів, таких як пензлі, олівці, пензлі для розпилення, штампи, інструменти виділення та багато іншого. Графічні редактори також надають функції для редагування кольору, розміру, прозорості та інших характеристик зображення. Деякі відомі графічні редактори включають Adobe Photoshop, GIMP, CorelDRAW та Adobe Illustrator.

Для розробки даної гри я вирішив зупинитись на Adobe Photoshop, оскільки в мене вже є досвід роботи із цим редактором. Окрім того, на даному етапі розробки, гра не потребує створення високоякісної графіки. Даний проект потребує створення декількох спрайтів та редагування зображень, зміни їх співвідношення сторін та параметрів ширини та висоти.

Також, під час розробки необхідно використати систему контролю версій. Система контролю версій – це інструмент, який використовується розробниками для керування змінами у вихідному коді програмного забезпечення або в інших файлах проекту. Він дозволяє відстежувати зміни, порівнювати версії файлів, повертатися до попередніх версій та поєднувати зміни від різних учасників проекту. також забезпечує історію змін, що допомагає зрозуміти, хто і коли вносив якісь зміни, і може служити інструментом для спільної роботи над проектом. Деякі популярні системи контролю версій включають Git, Subversion та Mercurial.

У моєму випадку я використовую систему BitBucket та SourceTree. Перше – це гіт-система контролю версій, яка реалізована у вигляді веб-додатку. Вона є безкоштовною, а також дає можливість використовувати велику кількість інших додатків, які вбудовані у структуру Atlasian. SourceTree є програмою для робочою платформи, яка пов’язується із системою контролю версій. На відміну від класичного Git-a, SourceTree має графічний інтерфейс, що дуже спрощує роботу із ним. Це програмне забезпечення необхідне в перчу чергу для збереження результатів розробки у мережі Інтернет.

1.4 Формування концепту ігрового процесу

Розробка будь-якого ігрового проекту розпочинається з проробки концепту

гри та ігрового процесу. Не виконавши розробки такого концепту, буде складно створити без зайвих проблем під час розробки. Концепт-документ гри – це документ, який містить основні ідеї, концепції, механіки геймплею, сюжетні елементи та інші ключові аспекти, пов'язані з розробкою гри. Цей документ зазвичай є основою для її подальшого розвитку. У концепт-документі гри частіше за все розглядають такі питання:

- Опис гри: Загальний опис ігрового світу, сюжету, атмосфери та основної мети гри;
- Механіка геймплею: Опис ігрових механік, механіки управління, фізики ігрового світу, системи прогресії, системи бою та інших ігрових аспектів;
- Сюжет та персонажі: Опис сюжету гри, персонажів, їх характеристик, мотивацій та відносин між ними;
- Графіка та аудіо: Концепції та ідеї з візуального стилю, арт-дизайну, анімації та звукового оформлення гри;
- Світогляд і фон: Додаткові деталі про світ гри, його історію, культуру, технології та інші аспекти, які можуть впливати на ігровий досвід.
- Технічні вимоги та можливості: Обговорення технічних аспектів розробки гри, включаючи платформи, програмні рушії, інструменти та інше.

Концепт-документ гри допомагає команді розробників чітко визначити напрямок розробки, уникнути непорозуміння та уточнити ключові аспекти гри ще до початку фактичної роботи над проектом.

Оскільки метою проекту є розробка гри-головоломки, то мені необхідно було виконати формування ігрового процесу, інтерфейсу, ключових елементів гри, як меню та ігрові сцени. На рисунку 1.8 можна побачити концепт ігрового меню, яке буде мати декілька функціональних кнопок, та тематичне зображення гори Фудзі на фоні.

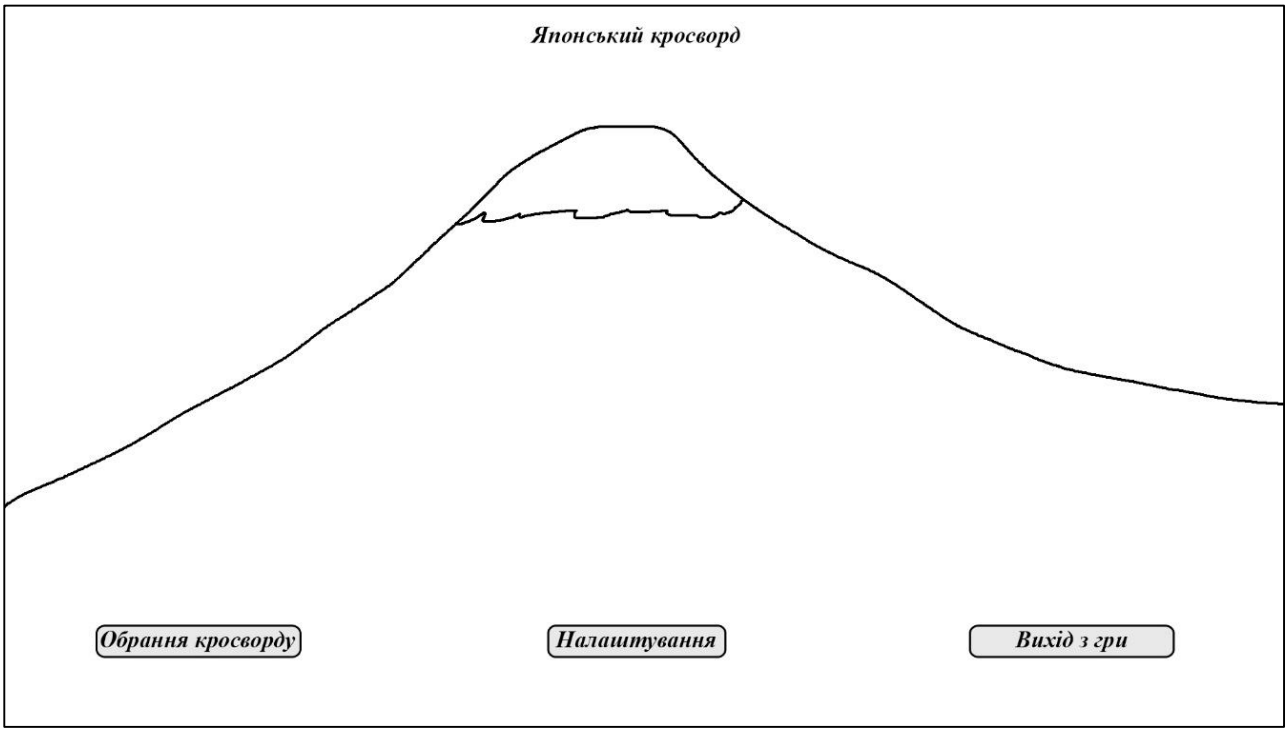


Рисунок 1.8. Концепт ігрового меню гри Японський кросворд

Маючи на увазі створити максимально простий та в той же час зрозумілий ігровий процес, було вирішено мінімізувати інтерфейс, розмістивши лише сам кросворд, відмітку про те, скільки має бути замальовано комірок у полі, а також кнопку перевірки. На рисунку 1.9 можна побачити концепцію відображення ігрового процесу.

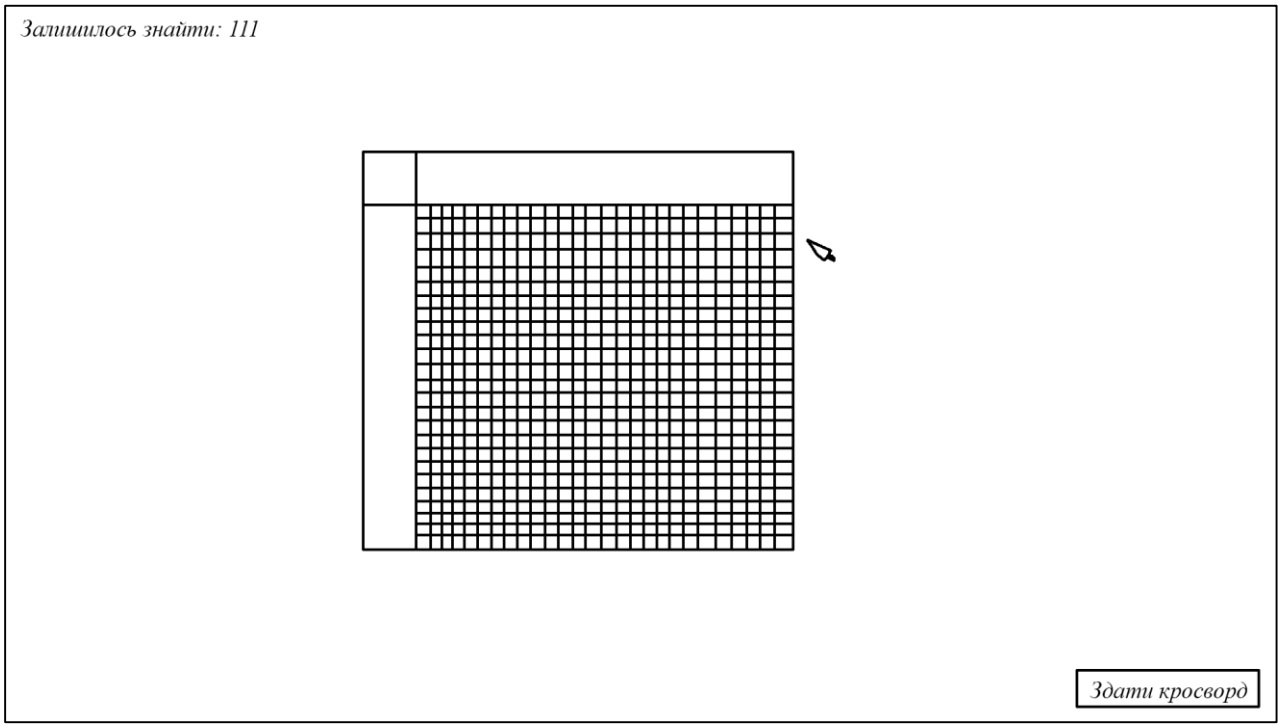


Рисунок 1.9. Концепція відображення ігрового процесу

Гра буде починатись з меню, в якому можна вибрати кросворд із списку та вийти з гри, а також мінімальними налаштуваннями. Після обрання кросворду завантажується кросворд та починається ігровий процес. Якщо була обрана опція «Навчання», то під час гри будуть виникати підказки.

Лівою кнопкою миші можна закрашувати комірки, при повторному натисканні комірка буде позначена як така, в якій нічого немає, при ще одному натисканні буде поставлена крапка, яку гравець може використовувати для маркування комірок, що викликають сумніви.

Інтерфейс буде показувати кількість комірок, які мають бути замальовані, і незалежно від того, вірно гравець відмітив комірку, чи ні, кількість відмічених комірок буде зменшуватись, або навіть уходити у від'ємні значення. Кнопка «Здати кросворд», або «Перевірити кросворд» виконає перевірку кросворду на правильність заповнення кросворду та в залежності від результату буде виведено те, чи інше повідомлення.

Кросворди має бути можливість завантажувати в папку кросвордів. Окремо потрібно передбачити інструмент обробки файлів кросвордів, та створити систему для створення файлів кросвордів.

1.5 Проектування основних елементів ігрового процесу та принципи їх роботи

Для вдалої реалізації ігрового проекту, необхідно попередньо виконати проектування основних ігрових елементів. Це дає змогу заздалегідь виявити можливі помилки під час реалізації елементів. Перед початком проектування, потрібно скласти структурну схему всього проекту, з урахуванням всього його функціоналу.

На рисунку 1.10 можна переглянути отриману структурну схему. Як можна бачити, гра поділяється на декілька станів. Стан Ігрового меню, та стану Ігрового процесу. У Ігровому меню гравець отримує доступ до функціоналу Виходу з гри, Налаштування гри, а також безпосередньо переходу до самого Ігрового процесу. Вихід з гри не має зворотнього зв'язку, але Налаштування та Обрання кросвордів

має, оскільки гравець може захотіти повернутись назад у мені, або випадково натиснути не ту кнопку.

Після переходу до Обрання кросвордів має спрацьовувати Менеджер кросвордів, який виконує пошук файлів-кросвордів та завантажує їх перелік у меню. Гравець має змогу обрати кросворд, після чого перейти до Ігрового процесу, натиснувши кнопку «Розв’язати».

Ігровий процес має складові які працюють у комплексі. Спочатку завантажується модуль Побудови кросворду. Він зчитує файл кросворду, визначає його будову, складає карту кросворду, та будує відповідний кросворд безпосередньо на екрані. Модуль системи управління виконує роль зчитування натискання на клавіші миші, та посилає відповідні сигнали до кросворду. Модуль перевірки виконує процес перевірки виконаних гравцем дій після чого дає знати гравцю щодо правильності його розв’язання.

Окремо слід відмітити модуль Ігровий менеджер. Цей модуль контролює ігровий процес та відстежує стан гри, пересування гравця по ньому, та викликає ті, чи інші модулі до роботи. Перехід від одного стану до іншого виконується тільки через Ігровий менеджер.

Оскільки Ігровий менеджер знаходиться здається над іншими модулями, його необхідно реалізувати таким чином, щоби забезпечити незалежність від станів. Це можливо реалізувати за допомогою використання такої технології ігрового програмного рушія Unity як Empty GameObject. По суті це ігровий об’єкт, який знаходиться на сцені, але він не має жодного компоненту окрім Transform. Тобто цей об’єкт не має фізичного тіла всередині гри, а також візуально ніяким чином не відображається на екрані. Це ідеальний інструмент для додавання до нього скриптів-слухачів, які виконують свій код незалежно від того, які об’єкти на сцені, які процеси на ній виконуються.

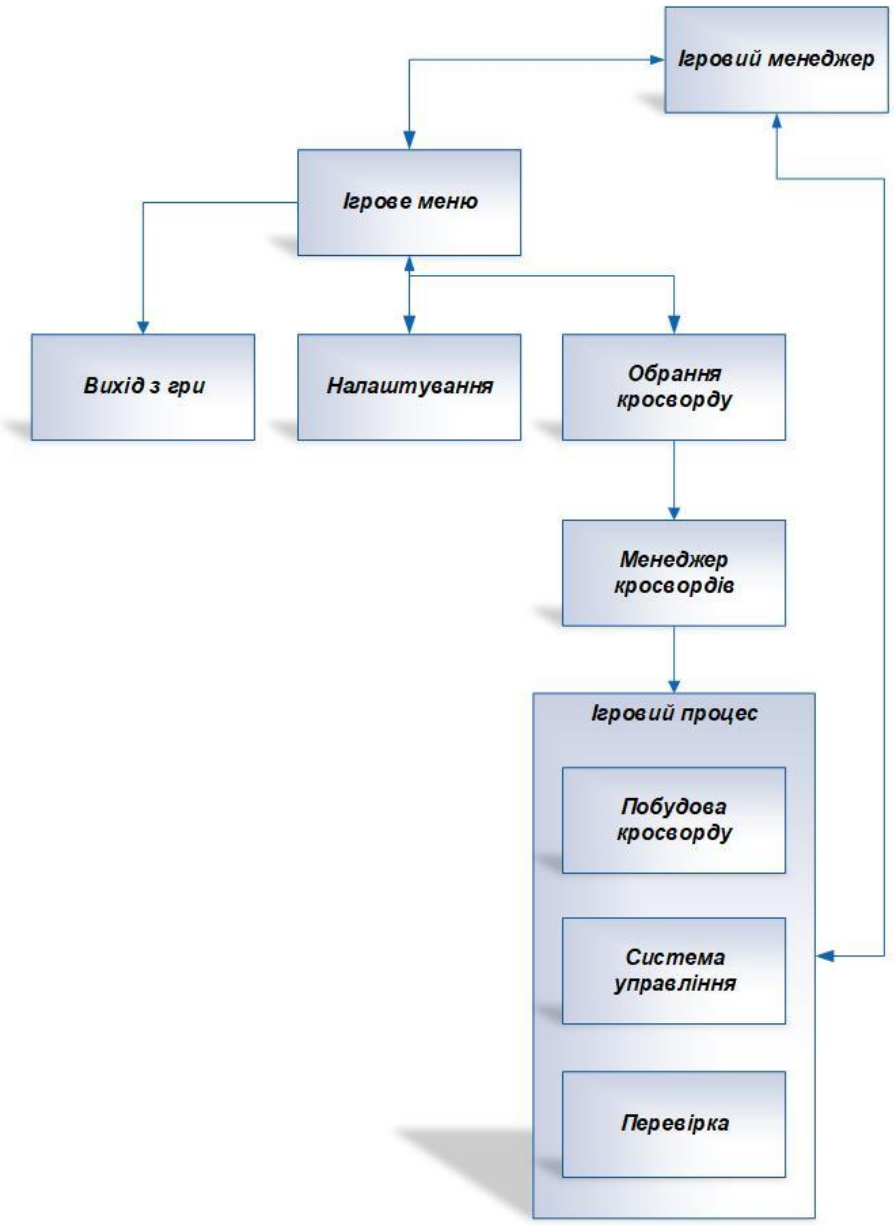


Рисунок 1.10. Структурна схема ігрового проекту Японський кросворд

Розглянемо структуру станів більш детально. На рисунку 1.11 зображено стан ігрового меню. Як можна побачити, в меню кнопки вмикання функціоналу знаходяться у порядку Обрання кросворду, Налаштування та Виходу з гри. Це дозволяє більш правильно сконцентрувати увагу гравця, викликаючи у нього перше відчуття відразу перейти до розв’язання кросвордів. Натиснувши на кнопку Обрання кросворду, ігрове меню зміниться и буде відображено Панель кросвордів з їх лістингом. Ця панель завантажується спеціальним модулем для зчитування кросвордів у спеціальній папці. Також у інтерфейсі буде знаходитись кнопка повернення гравця до Ігрового меню.

Кнопка Налаштування відразу завантажить невелику панель із

налаштуваннями роздільної здатності екрану гравця, налаштування звуку, а також кнопку повернення до головного меню.

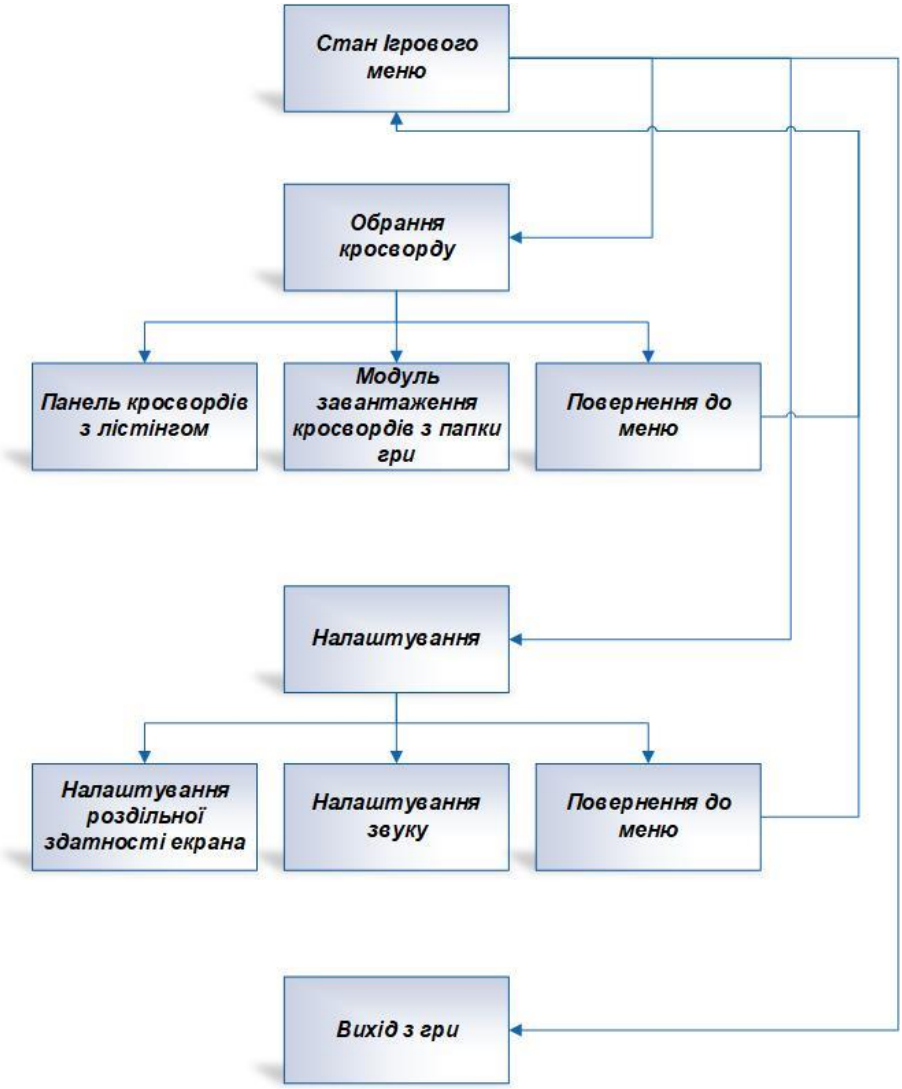


Рисунок 1.11. Структурна схема стану Ігрового меню

Наступним розглянемо структурну схему стану Ігрового процесу, яка зображена на рисунку 1.12. Зі схеми видно, що ігровий процес розпочинається з процесу побудови кросворду, який складається з шести дій.

Спочатку повинні завантажитись дані про кількість горизонтальних та вертикальних комірок. Потім завантажуються дані про ті комірки, які мають бути заштрихованими. За цими даними створюється так звана «Карта правильних комірок», яка буде використовуватись на етапі перевірки. Наступним кроком є генерація комірок на екрані та створення цифрових позначень на полях кросворду.

Після створення кросворду, гра передає гравцю систему управління і після того, як гравець буде вважати, що правильно склав кросворд, буде виконуватись

перевірка правильності заповнення кросворду. В результаті перевірки, управління може бути повернуто гравцю для подальшого вирішення кросворду, у разі якщо були помилки у штрихуванні комірок. Інакше гра піде на вихід до стану Ігрового меню.

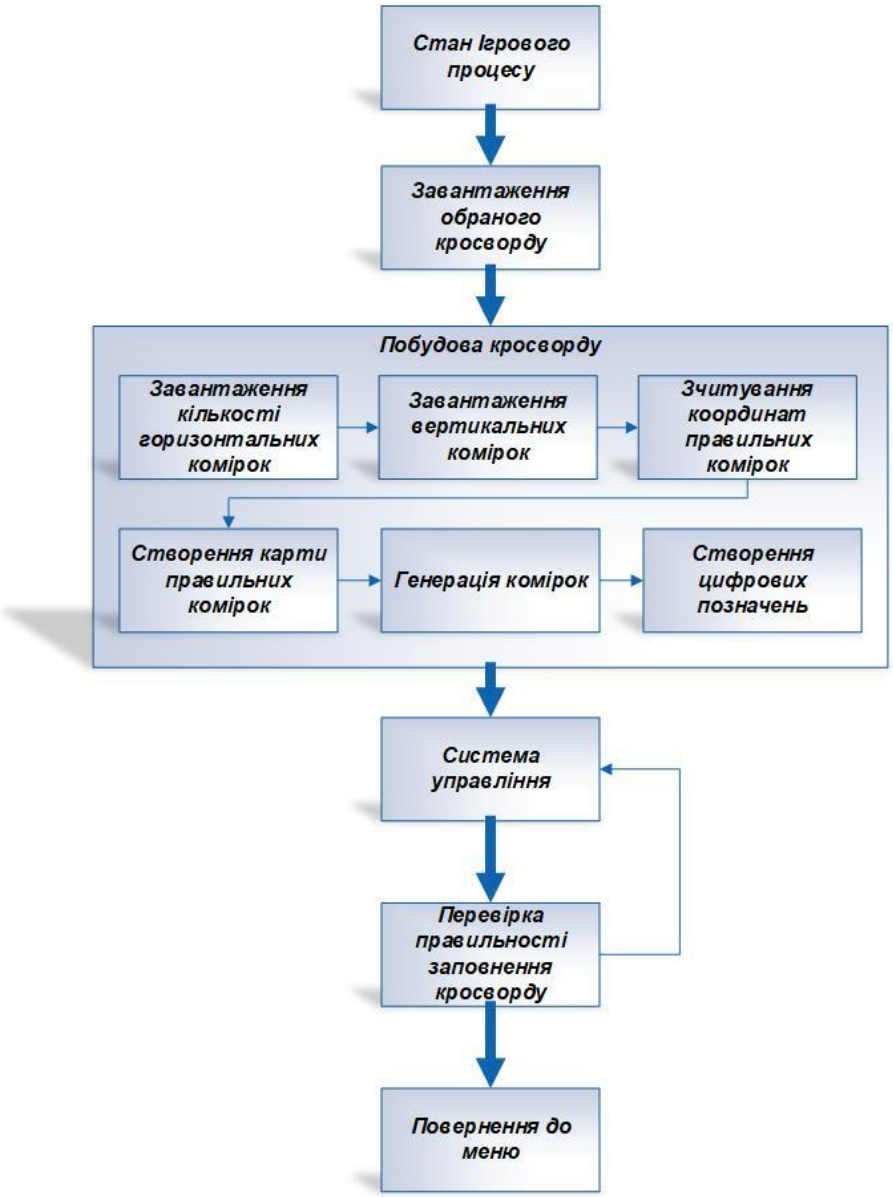


Рисунок 1.12. Структурна схема стану Ігрового процесу

Як можна побачити зі схем вище, ігровий проект можна умовно поділити на такий код, що працює з інтерфейсом, а також код, який виконує розрахунки на фоні.

1.6 Реалізація основних елементів ігрового процесу

Реалізація основних елементів ігрового процесу розпочинається зі створення графічних елементів гри. У моєму випадку, мені необхідно створити

елементи кросворду, які будуть використовуватись під час його генерації. Також потрібно створити види комірок – пусту, заштриховану, комірку з хрестиком та крапкою. Також необхідно зробити файл для заднього фону головного меню. Ці задачі виконуються за допомогою Adobe Photoshop. На рисунку 1.13 можна побачити створені спрайти та файл фону для гри Японський кросворд:

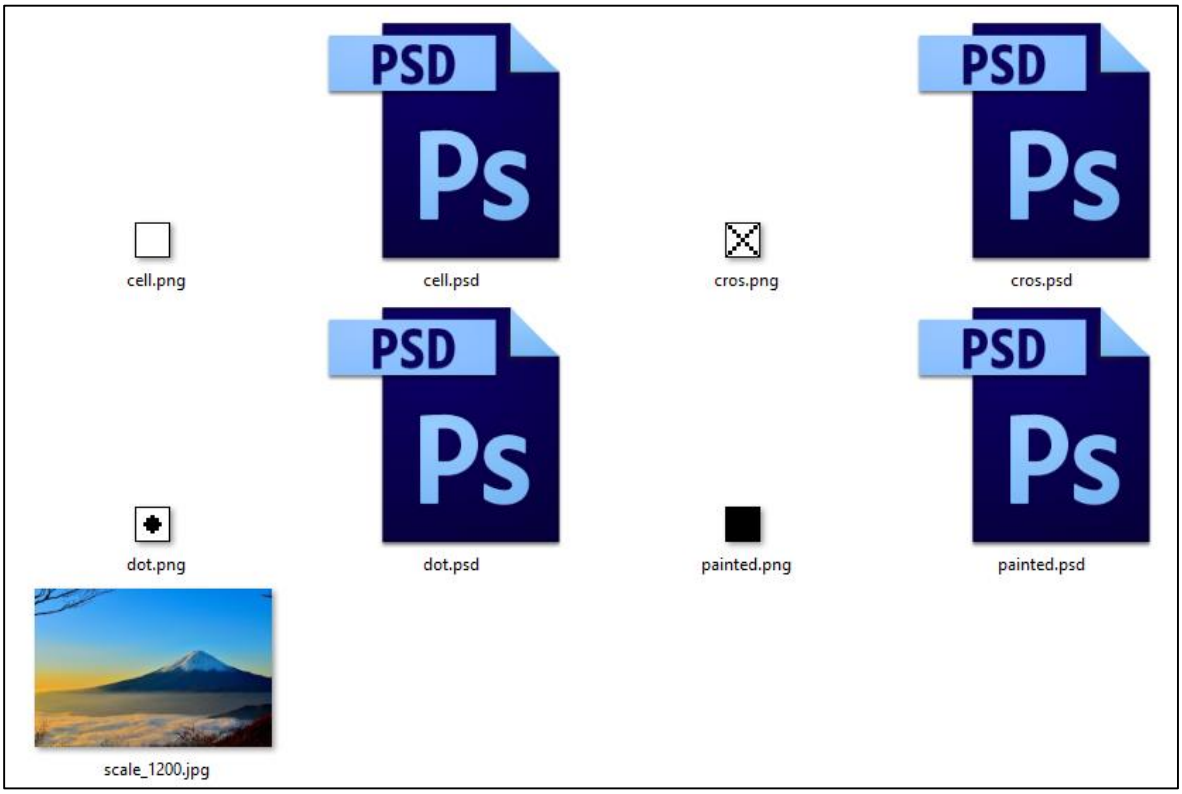


Рисунок 1.13. створені спрайти та файл фону для гри Японський кросворд

Маючи графічні файли можна перейти до створення ігрового меню. Для цих цілей потрібно створити проект у програмному рушії Unity за допомогою шаблону 2D-проекту. Це дасть змогу відразу розпочати розробки із налаштуваннями інтерфейсу для роботи з 2D-проектами. У тільки створеному проекті Unity, у ієрархії ігрових об’єктів створюємо елемент Canvas. Він дозволяє працювати з ігровим інтерфейсом, використовуючи класичні елементи інтерфейсу, як кнопки, панелі, зображення та інше. На рисунку 1.13 зображено пустий елемент Canvas у редакторі Unity.

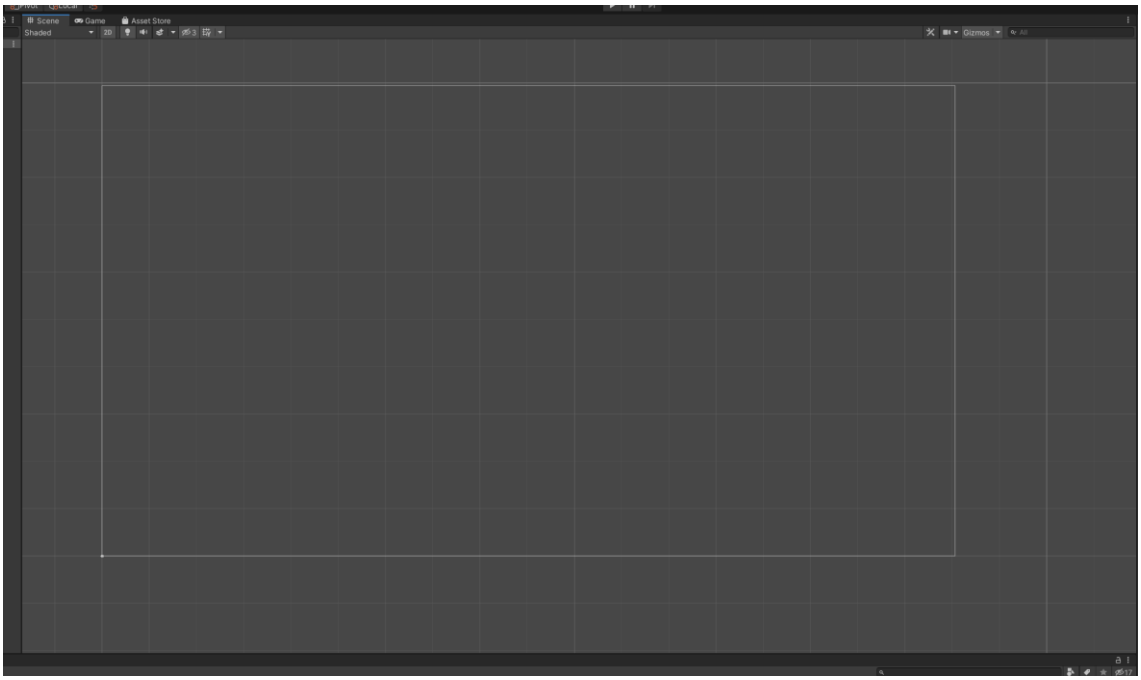


Рисунок 1.13. Пустий елемент Canvas у редакторі Unity

Згідно існуючого концепту інтерфейсу меню, створюємо всі необхідні елементи. Оскільки нам необхідні кнопки, то створюємо 3 елементи Button. Для редагування тексту кнопок, використовується елемент Text, який вкладений до шаблонів кнопок. Змінювати візуальний стиль кнопок можна за бажанням у відповідних налаштуваннях елементу кнопок. Направши текст кнопок, створюю заголовок гри «Японський кросворд» за допомогою елементу інтерфейсу користувача Text. Наступним важливим кроком для створення відповідного антуражу буде розміщення зображення гори Фудзі на фоні меню. Створивши елемент Panel, можна розтягнути його за допомогою інструменту Stretch компоненту Transform панелі. Використовуючи цей інструмент, можна обирати спосіб розтягування елементу по Canvas. В результаті виконання робіт отримуємо меню гри Японський кросворд, який можна переглянути на рисунку 1.14:

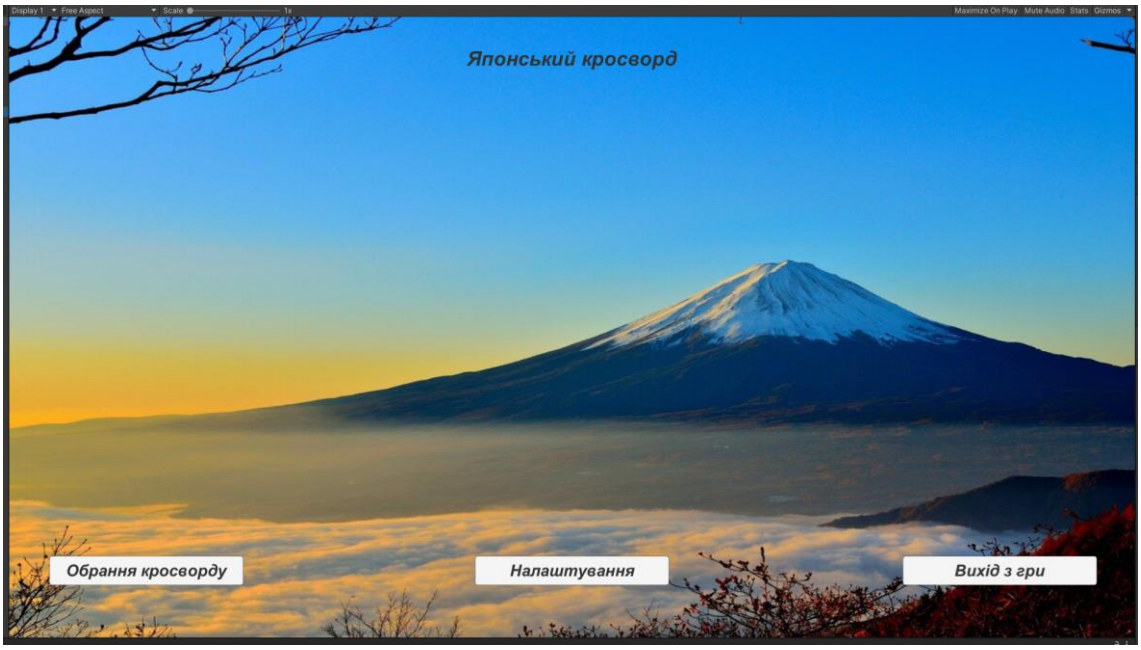


Рисунок 1.14. Меню гри Японський кросворд

Подальша робота з ігровим меню зводилась до створення відповідних панелей управління для кнопок Обрання кросворду та Налаштування, вкладених панелей та кнопок.

Після розміщення елементів інтерфейсу необхідно виконати програмування елементів ігрового меню. Якщо з програмуванням елементів меню, які знаходяться перед гравцем із запуском гри все зрозуміло, то елементи, що спочатку знаходяться у скритому стані є деякі особливості.

Ці елементи на момент розробки встановлюються як видимі, для того щоби можна було з ними працювати. При побудові білду гри, або для відладки гри, необхідно встановити їх у положення невидимих. Це необхідно у зв’язку з тим, що таким чином можна зменшити кількість коду, яку потрібно було б відводити на початковий процес хования елементів інтерфейсу. Далі, упродовж роботи ігрового рушія, кожний такт ігрового процесу, якщо гравець натискає відповідну кнопку, вона скриває інші елементи ігрового меню, саму себе та викликає потрібну панель. На рисунку 1.15 зображено схему роботи схованих елементів меню:

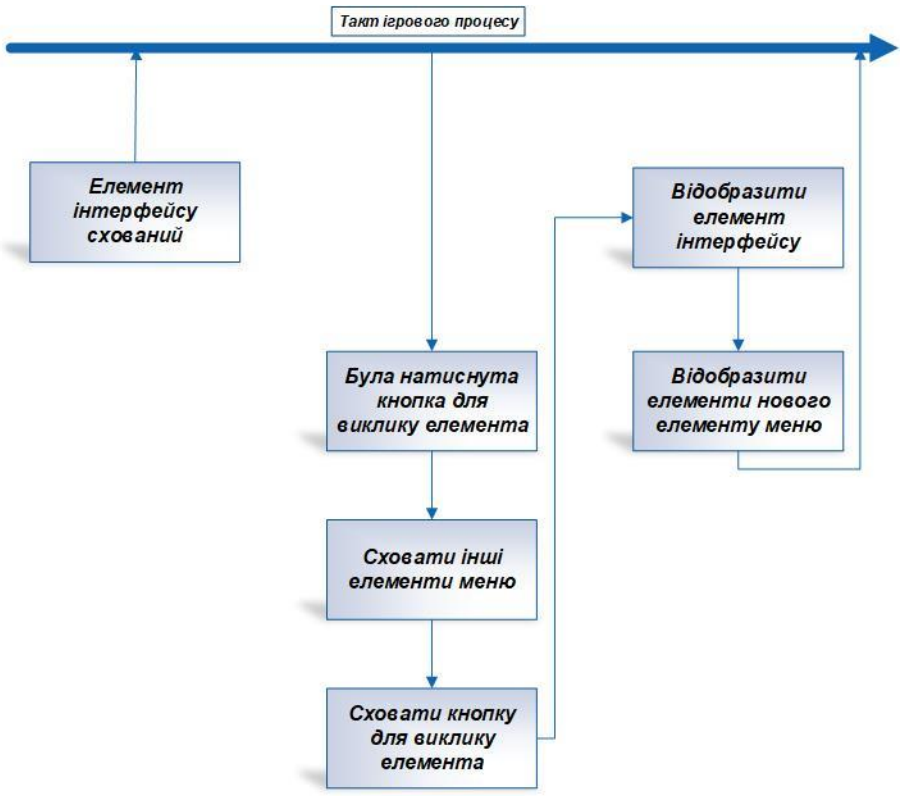


Рисунок 1.15. Схема роботи схованих елементів меню

Для реалізації на програмному рівні цього процесу, потрібно створити відповідний скрипт по роботі із ігровим інтерфейсом. Він буде відповідати за виклик елементів меню, їх активацію, наявність на екрані, або зміну елементів. За цю роботу відповідає скрипт player_ui_manager.cs. Наприклад процес відображення панелі обрання кросвордів написаний у відповідному методі цього скрипта, що можна побачити нижче у фрагменті коду скрипта player_ui_manager.cs:

```
public void show_chose_panel()
{
    chose_button.SetActive(false);//сховати кнопку обрання кросвордів
    options_button.SetActive(false);//сховати кнопку налаштувань
    exit_button.SetActive(false);//сховати кнопку виходу з гри

    chose_panel.SetActive(true);//відобразити панель обрання кровсордів
    return_button.SetActive(true);//відобразити кнопку повернення у меню
}
```

Як можна бачити, метод має публічний модифікатор доступу, що дає можливість викликати його з інших скриптів, або з інших місць цього скрипта. Всередині методу йде звернення до елементів ігрового меню, таких як кнопка

обрання кросвордів, налаштування та вихід з гри. У цих елементах викликається метод SetActive, який відповідає за режим відображення ігрового об’єкту у ігровому просторі, або інтерфейсі користувача. На рисунку 1.16 можна побачити панель обрання кросвордів:

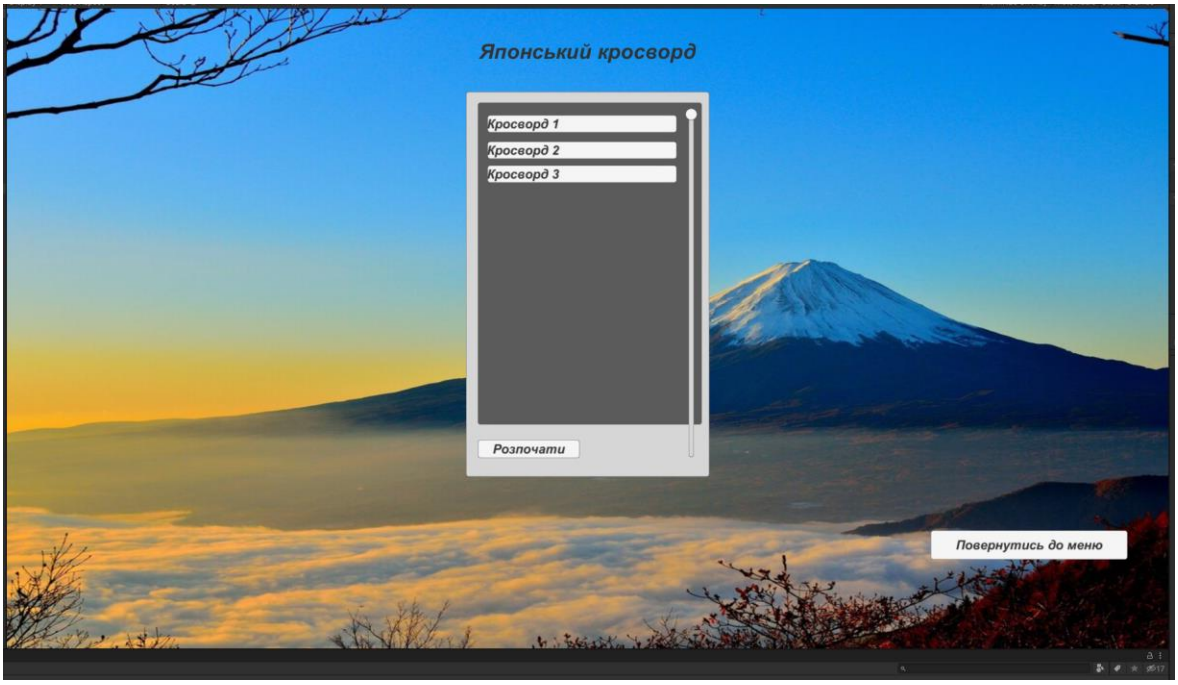


Рисунок 1.16 Панель обрання кросвордів

Кнопка Розпочати викликає вимкнення всього ігрового меню. З екрану прибираються всі кнопки, панелі та зображення на фоні. Після цього гра переходить у стан Ігрового процесу. Кнопка Повернутись до меню вимикає панель обрання кросвордів та завантажує кнопки головного меню.

Таким чином для кожної кнопки ігрового меню створюється відповідний метод, який регулює відображення того, чи іншого елемента інтерфейсу. За всі маніпуляції з інтерфейсом, як головного меню, так и інтерфейсу під час розв’язання кросворду відповідає скрипт player_ui_manager.cs.

Наступним кроком у реалізації основних ігрових елементів буде створення алгоритму, який буде малювати кросворд, в залежності від того, який було обрано у ігровому меню. Схема роботи такого алгоритму була зображена раніше. По суті необхідно створи автомат, який буде малювати клітинки та номери у залежності від вхідних даних. Перед тим потрібно реалізувати систему цих вхідних даних. Потрібно створити чітку структуру файлу, який буде зчитуватись програмою для

побудови кросворду.

Як вже було зазначено у схемі роботи Ігрового процесу, нам потрібно вказувати у файлі такі дані, як кількість горизонтальних комірок, кількість вертикальних комірок, розміщення комірок, які мають бути замальованими. Для цього створюється файл у визначеній структурі із кодовими словами, які потім алгоритм буде зчитувати та визначати які дані куди записати. На рисунку 1.17 зображено зміст одного з файлів кросворду:

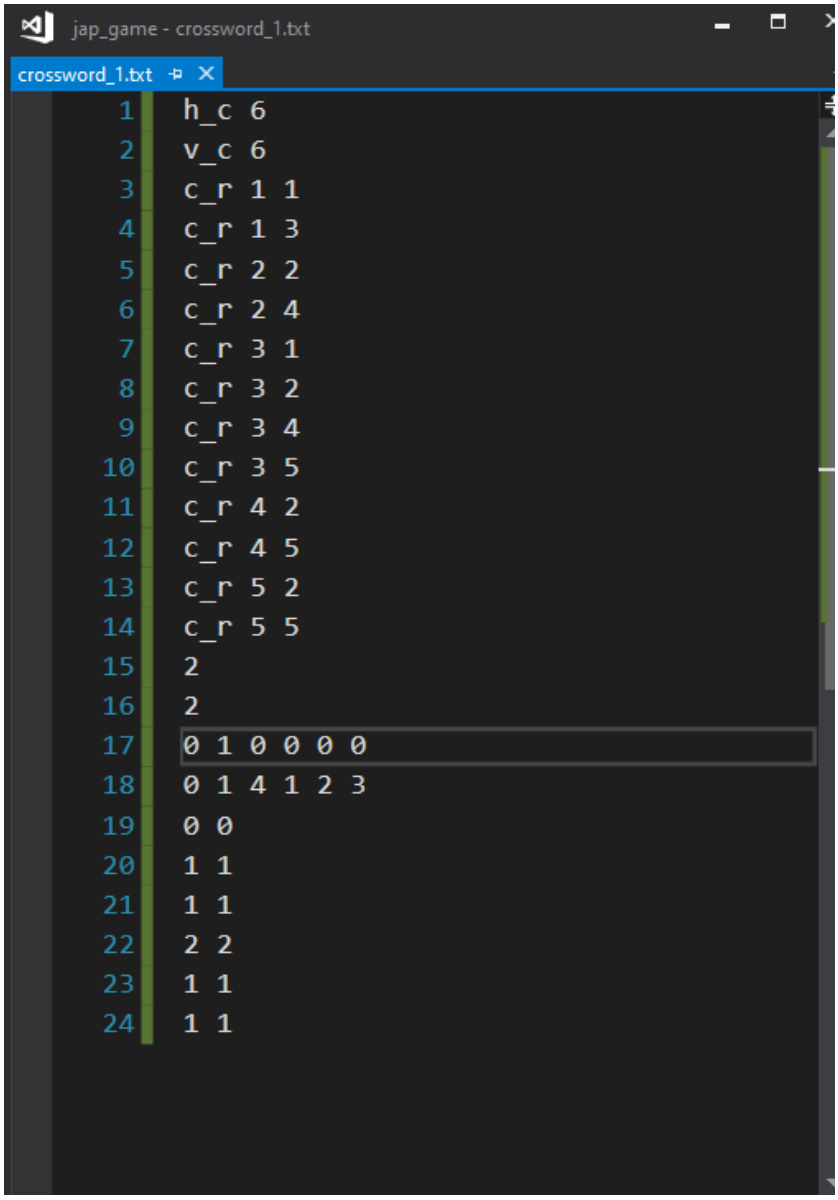


Рисунок 1.17 Зміст одного з файлів кросворду

Розглянемо зміст файлу більш детально у таблиці 1.1. В ній відображена інформація по строкам, які позначені у файлі кросворду, та розшифровано призначення кожної строки.

Таблиця 1.1. Пояснення змісту файлу кросворду гри Японський кросворд

Номер строки	Призначення змісту
1	Строка із службовим словом h_c. Повідомляє алгоритму кількість комірок по горизонталі.
2	Строка із службовим словом v_c. Повідомляє алгоритму кількість комірок по вертикалі.
3-14	Строки із службовим словом c_r вказують алгоритму в яких комірках двомірного циклу знаходяться комірки, які мають бути заштрихованими.
15	Значення кількості строк цифрових позначень по горизонталі.
16	Значення кількості стовпців цифрових позначень по вертикалі.
17-18	Цифрові позначення для першої та другої строки горизонталі цифрових позначень.
19-24	Цифрові позначення для построчного заповнення вертикальних стовпців цифрових позначень.

Як можна побачити з таблиці файл кросворду має визначену структуру, яку спеціальний алгоритм переглядає та виконує дії побудови кросворду.

Спочатку зчитуються кількість комірок по горизонталі та вертикалі у строках файлу 1-2, таким чином алгоритм дізнається якої розмірності двомірний масив йому створювати, який буде картою відповідей кросворду. Даний масив створюється у логічному типу Boolean. Крім того, ця інформація домагає алгоритму у побудові комірок кросворду.

Після розміщення комірок на екрані та визначення розмірності двовимірного масиву – карти відповідей кросворду, необхідно заповнити комірки цього масиву. За це відповідають строки файлу 3-14. В них є роздільне службове слово c_r, яке дає команду алгоритму, що наступні дві цифри будуть індексами масиву. Отримавши обидва індекси, алгоритм встановлює зміст комірки у

значення true і зчитує наступний службовий символ. Якщо це знову c_r, то починається пошук нової комірки. Одночасно із записом у карту відповідей кросворду, алгоритм збільшує лічильник кількості комірок, які потрібно заштрихувати.

Якщо при зчитуванні наступним символом після знаку кінця строки буде цифра, то алгоритм зберігає кількість горизонтальних та вертикальних строк цифрових позначень. Це потрібно для того, щоби скрипт знав скільки йому створювати ігрових об’єктів типу TextMesh, які будуть зберігати кількість комірок, які потрібно заштрихувати у відповідній горизонталі, чи вертикалі.

Виконавши 2 зчитування – оскільки цифрові позначення існують лише у горизонталі та вертикалі – алгоритм виконує визначення вмісту цифрових позначень, послідовно зчитуючи 2 строки з 6 значень для горизонталі. Саме такі значення через те, що алгоритму відомо, що у даному випадку всього 2 строки по горизонталі, а також 6 комірок по горизонталі.

Закінчивши з горизонтальними строками цифрових позначень, алгоритм переходить до вертикальних, де кількість строк дорівнює кількості комірок по вертикалі – 6 – а кількість значень, які потрібно отримати дорівнює 2, оскільки вказано 2 стовпця для цифрових позначень по вертикалі. Необхідно зазначити, що дані, які отримуються у строках файлу кросворду 1-2 та 15-24 не зберігаються алгоритмом.

Нижче продемонстровано фрагмент коду алгоритму зчитування файлу та побудови кросворду, де надаються імена файлам комірок, а також ініціалізується масив із правильними відповідями:

```
for(int i=0; i<6; ++i){//Іменування згенерованих комірок
        for (int i2 = 0; i2 < 6; ++i2){
                cells_name[i, i2] = "c_" + i + "_" + i2;
        }
}
for (int i = 0; i < 6; ++i){//Початкова ініціалізація масиву карти відповідей
        for (int i2 = 0; i2 < 6; ++i2){
                cells_right[i, i2] = false;
        }
}
```

```
for (int i = 0; i < 6; ++i){//Початкова ініціалізація масиву здогадок гравця
        for (int i2 = 0; i2 < 6; ++i2){
                player_gues[i, i2] = false;
        }
}
```

Окремо необхідно зазначити ініціалізацію масиву здогадок гравця. Даний масив необхідний для збереження варіантів відповідей гравця під час вирішення. На рисунку 1.18 можна побачити результат генерації кросворду алгоритмом, згідно файлу кросворду.

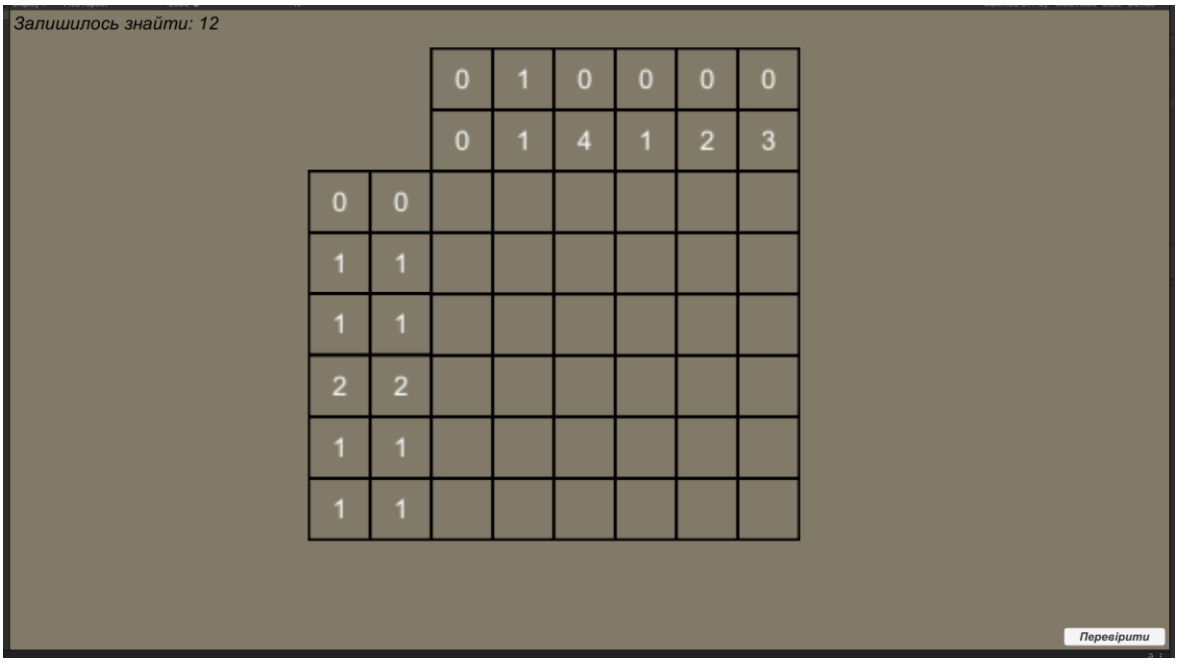


Рисунок 1.18. Результат генерації кросворду алгоритмом згідно файлу кросворду

Також, на рисунку 1.17 можна побачити й елементи інтерфейсу Ігрового процесу. Вони розміщені на Canvas, як елементи інтерфейсу гравця Text та Button. Розглянемо їх детальніше.

Елемент Text виводить на екран кількість комірок, які необхідно замалювати. Це дуже зручно для гравця, щоби той міг орієнтуватись, скільки ще комірок необхідно замалювати. Значення для цього текстового елементу береться з результату роботи алгоритму зчитування та створення кросворду, коли він заповнював лічильник кількості комірок, що мають бути замальовані. Цей текстовий елемент не є сталим і динамічно змінює своє значення у випадку, якщо гравець замальовує комірку. Слід зазначити, що значення буде зменшуватись і у від’ємні значення, що не є помилкою. Якщо кросворд було розв’язано не вірно, то

це буде підказкою для гравця. На рисунку 1.19 можна побачити, як змінюється значення текстового елемента згідно із замальованими комірками. Усього було замальовано 6 комірок, текстовий елемент відображає, що залишилось заштрихувати 6 комірок.

Також, на рисунку 1.20 можна побачити результат перебільшення кількості комірок, які потрібно замалювати. Як можна бачити, гра не видає ніякого сповіщення, як помилка, оскільки це один з вірогідних варіантів розвитку ігрового процесу.

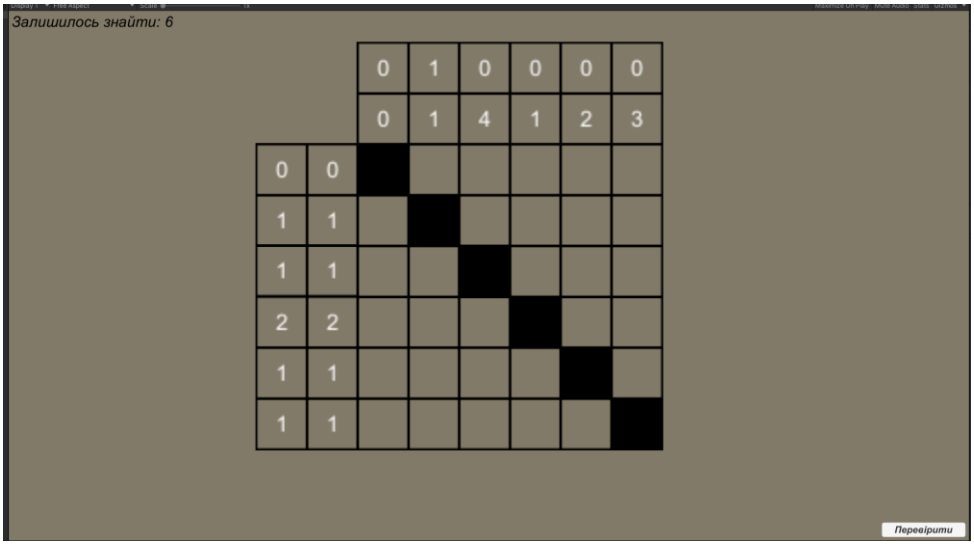


Рисунок 1.19. Зміна значення текстового елемента згідно із замальованими комірками

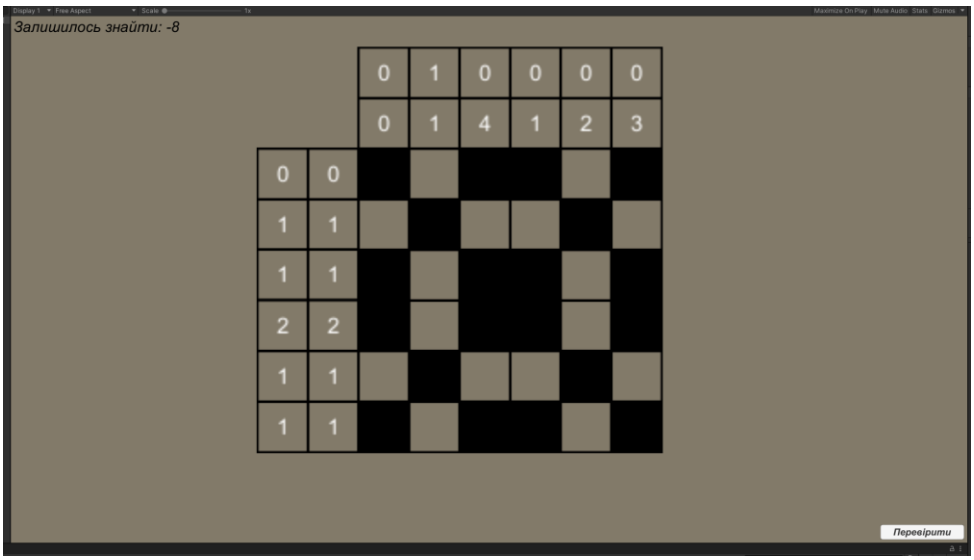


Рисунок 1.20. Результат перебільшення кількості комірок, які потрібно замалювати

Кнопка Перевірити виконує роль перевірки правильності відмічених

гравцем комірок. Цей процес дуже тісно пов’язаний з реалізацією основного ігрового елементу – відмічання комірок. На рисунку 1.21 можна побачити варіації стану комірки кросворду. Ми бачимо, що існує 4 стани комірки: невідмічений, відмічений, хрестик, крапка.

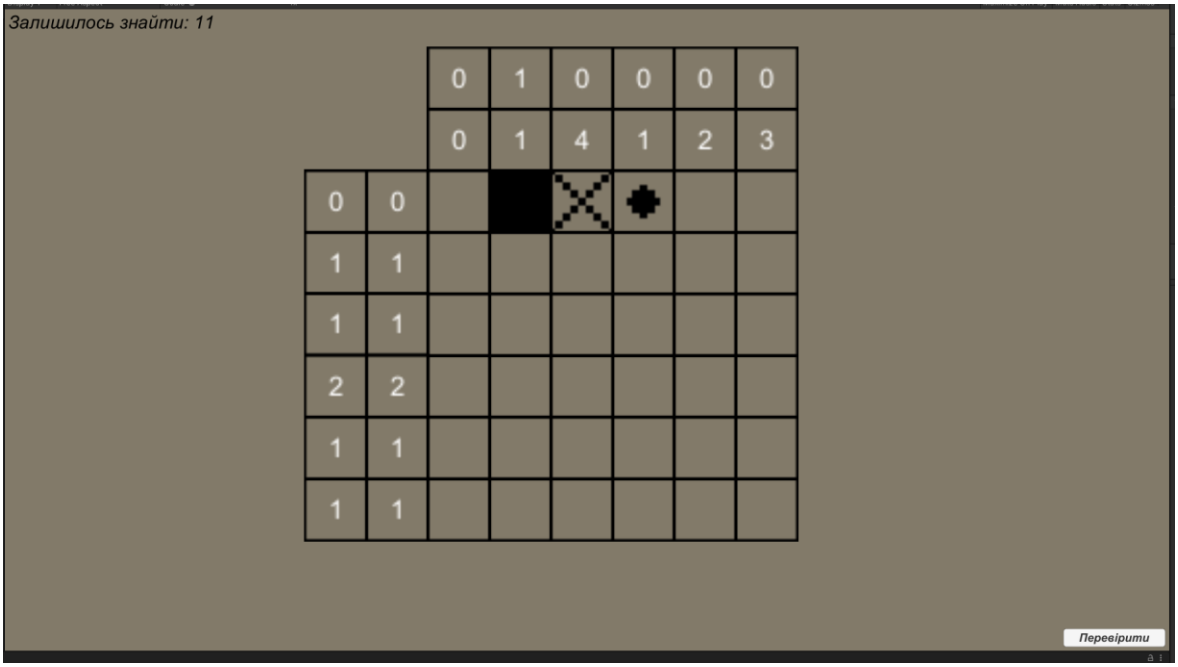


Рисунок 1.21. Варіації стану комірки кросворду

Невідмічена комірка ніяк не впливає на кількість комірок, що потрібно замалювати. Відмічена віднімає кількість комірок, що мають бути замальовані. Хрестик вказує що ця комірка не може бути замальованою, тому не впливає на кількість комірок, що мають бути замальовані. Крапка вказує на сумніви, тому так само не впливає на кількість комірок, які потрібно замалювати.

Втім, не дивлячись на те, що три стани комірки з чотирьох, здається, ніяк не впливають на кількість комірок, що необхідно замалювати, проте в коді гри це має декілька інший вигляд.

Перед тим як перейти до розгляду роботи відмічання комірок, необхідно проілюструвати те, як само виконується відмічання комірок у грі. Кожна комірка має свій файл-скрипт, який називається cell_logic.cs. Даний скрипт відповідає за те, який вигляд має комірка в залежності від того, скільки разів на неї натиснули, або, інакше кажучи, який стан вона має зараз. На рисунку 1.22 можна побачити схематичне зображення роботи структури станів комірки.

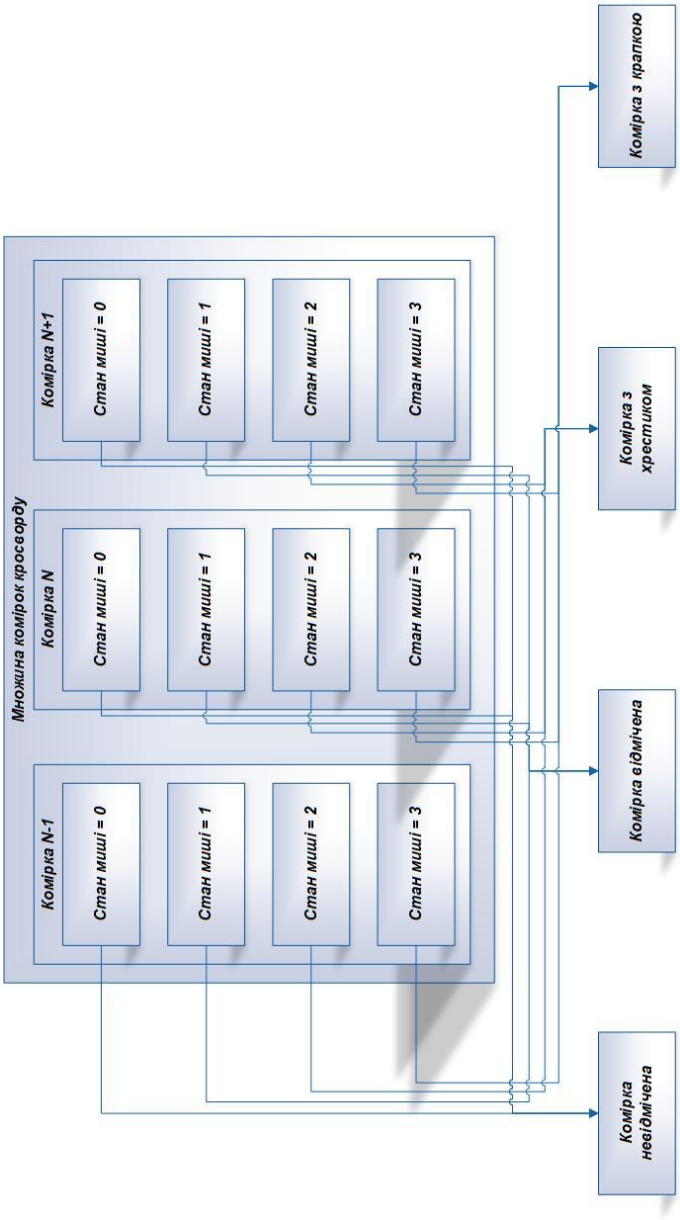


Рисунок 1.22. Схематичне зображення роботи структури станів комірки

Як можна побачити з рисунку 1.21, існує множина комірок кросворду, яка має динамічну величину, в залежності від кросворду. Кожна комірка має 4 стани, які відповідають малюнку на комірці. Кожна комірка має свій стан незалежно від інших, окрім того, кожна комірка зберігає свій стан до наступного разу зміни стану. Кожна така зміна викликається натисканням лівої кнопки миші на комірку.

Можливість натиснути на комірку реалізовується завдяки компоненту Collider, що створює фізичні кордони ігрового об’єкту. Втім, вони є умовними до тих пір, доки до ігрового об’єкту не буде додано компонент твердого тіла RigidBody. Структуру компонентів кожної комірки кросворду можна побачити на рисунку 1.23:

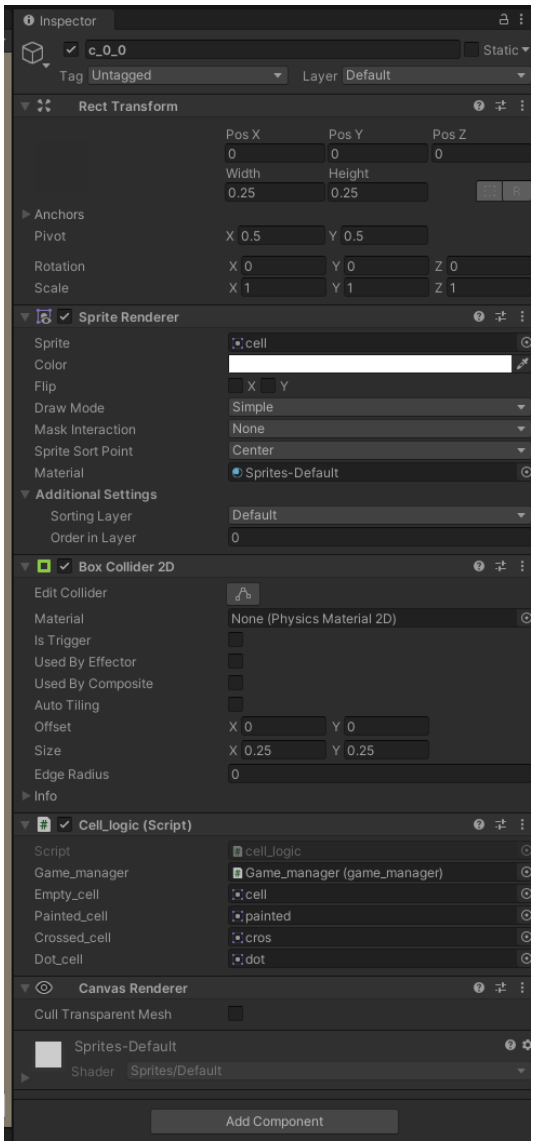


Рисунок 1.23. Структура компонентів кожної комірки кросворду

Як можна бачити з найменування ігрового об’єкту це є перша комірки, оскільки в названі зазначено c_0_0. Вона розміщується в центрі ігрового простору. Має всього три компоненти, SpriteRenderer, BoxCollider2D та Cell_logic. SpriteRenderer відповідає за відображення комірки. У даному компоненті є властивість Sprite, яка зберігає в собі спрайт, який буде відображений ігровим рушієм. BoxCollider2D потрібен для зчитування натискання лівою кнопкою миші на комірку. Оскільки комірка має форму квадрату, то використовується саме BoxCollider2D. Останній компонент є скрипт Cell_logic, який відповідає за логіку комірки. Як можна побачити, цей скрипт отримує чотири спрайти, які відповідають стану комірки. Ці спрайти, в залежності від стану комірки передаються до властивості Sprite компоненту SpriteRenderer, що змінює вигляд

комірки. Нижче можна побачити фрагмент коду, який відповідає за зчитування станів комірки при натисканні лівої кнопки миші:

```
public void OnMouseDown(){//При спрацюванні події натискання миші
    if (mouse_state == 0){//Якщо стан натискання миші дорівнював 0
        mouse_state = 1;//Перемкнути стан натискання миші
        _sprite_renderer.sprite = _painted_cell;//Змінити спрайт комірки
        _game_manager._cell_painted();//Викликати код зміни стану комірки
        _game_manager._take_cell_name(gameObject.name, true);
        //Передача до ігрового менеджера ім’я комірки до зміни
    }
    else if (mouse_state == 1){
        mouse_state = 2;
        _sprite_renderer.sprite = _crossed_cell;
        _game_manager._cell_unpainted();
        _game_manager._take_cell_name(gameObject.name, false);
    }
    else if (mouse_state == 2){
        mouse_state = 3;
        _sprite_renderer.sprite = _dot_cell;
        _game_manager._take_cell_name(gameObject.name, false);
    }
    else{
        mouse_state = 0;
        _sprite_renderer.sprite = _empty_cell;
        _game_manager._take_cell_name(gameObject.name, false);
    }
}
```

Логіка роботи даного коду доволі проста. Оскільки кожна комірка має свій сталий стан, то при створені комірки та з моменту існування скрипту комірки, їй встановлюється стан mouse_state у значення 0, що відповідає не відміченій комірці. Коли гравець натискає на кнопкою на комірці, вона зчитує свій поточний стан, збільшує його на одиницю, завантажує відповідний спрайт, викликає код зміни стану комірки та передає до ігрового менеджеру ім’я комірки, яка змінилась.

Цей код працює без змін до тих пір, поки стан комірки під час натискання лівою кнопкою миші не буде дорівнювати 3, що відповідає комірки з крапкою. В цьому разі, значення стану комірки буде встановлено в 0, комірка змінить свій вигляд на невідмічену комірку та процес чергування комірок продовжиться спочатку.

Тепер, коли реалізовано код зміни стану та вигляду комірок, потрібно реалізувати безпосередньо саму ігрову логіку перевірки – як гра зрозуміє, чи вірно розв’язано кросворд. Раніше вже було зазначено, що алгоритм побудову кросворду створює карту правильних відповідей. Такий підхід відразу підштовхує на ідею створити карту відповідей гравця. Саме тому, коли гравець відмічає комірку, виконується виклик методу передачі ім’я комірки, яку відмітили. Втім, з цим є деякі складнощі.

Оскільки комірки можуть приймати більше ніж 2 значення, гра має якимось чином розуміти, ми відмітили комірку як таку, що хочемо заштрихувати? Окрім того, це важливо і для зміни значення текстового елементу, що відображає кількість комірок, які залишилось заштрихувати.

Ця проблема вирішується за допомогою передачі до ігрового менеджеру game_manager.cs не тільки імені комірки, яку гравець натискає, а також стану комірки. Тут потрібно конкретизувати, що в даному випадку під «станом комірки» мається на увазі рахувати комірку як заштриховану, або навпаки – незаштриховану. Через це, ці значення передаються у вигляді логічних значень true або false. Схематичне зображення цього процесу можна побачити на рисунку 1.24.

На схемі видно, що окрім передачі стану комірки у логічному вигляді, в залежності від стану комірки від 0 до 3, також є така частина як «Відмітити на мапі здогадок гравця як незаштриховану комірку», або «Відмітити на мапі здогадок гравця як заштриховану комірку». Це важлива частина, яка ілюструє саме передачу інформації про відмічену комірку до мапи здогадок гравця. Цей процес також знаходиться всередині коду ігрового менеджеру game_manager.cs. Це реалізовано у вигляді двох окремих методів, кожен з яких або ставить стан комірки true (комірка заштрихована), або false (комірка незаштрихована). Варто зазначити, що кількість елементів у двомірному масиві мапи здогадок гравця дорівнює кількості комірок по горизонталі та вертикалі.

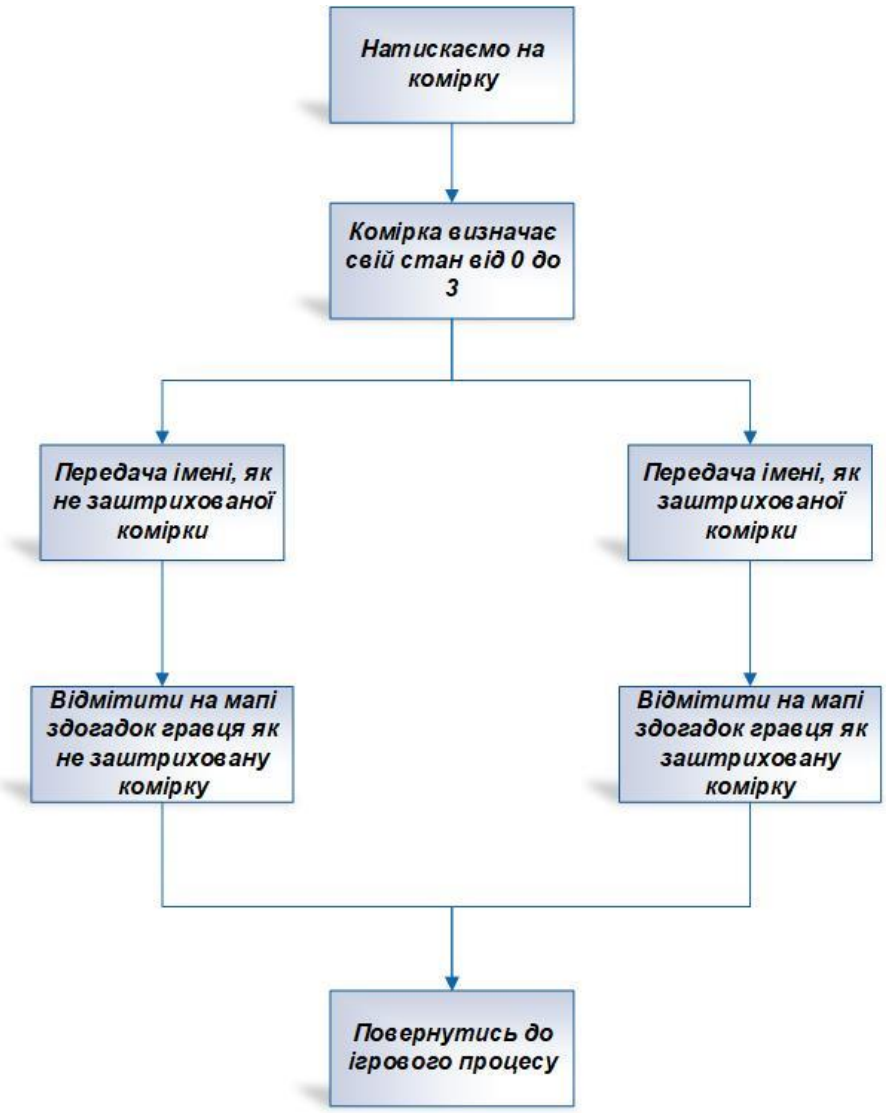


Рисунок 1.24. Схематичне зображення особливості передачі імені комірки до ігрового менеджеру

Необхідно зауважити й особливості у підрахунку комірок, які ще потрібно відмітити у текстовому елементі ігрового інтерфейсу. Для того, щоби уникнути ситуації, коли мі додатково змінили стан комірки та помилково зменшили кількість комірок, які потрібно заштрихувати, потрібно реалізувати виняткове зменшення цього значення. Також варто пам’ятати, що це значення може також і збільшуватись у випадку, якщо гравець змінив рішення і хоче відмітити комірку, як незаштриховану. Тому на рисунку 1.25 можна побачити модифіковану схему передачі імені комірки до ігрового менеджеру. Виходячи зі схеми, бачимо, що зміна значення кількості комірок виконується лише під час стану комірок 0 або 1. Це дозволяє уникнути проблеми з лишнім зменшенням значення.

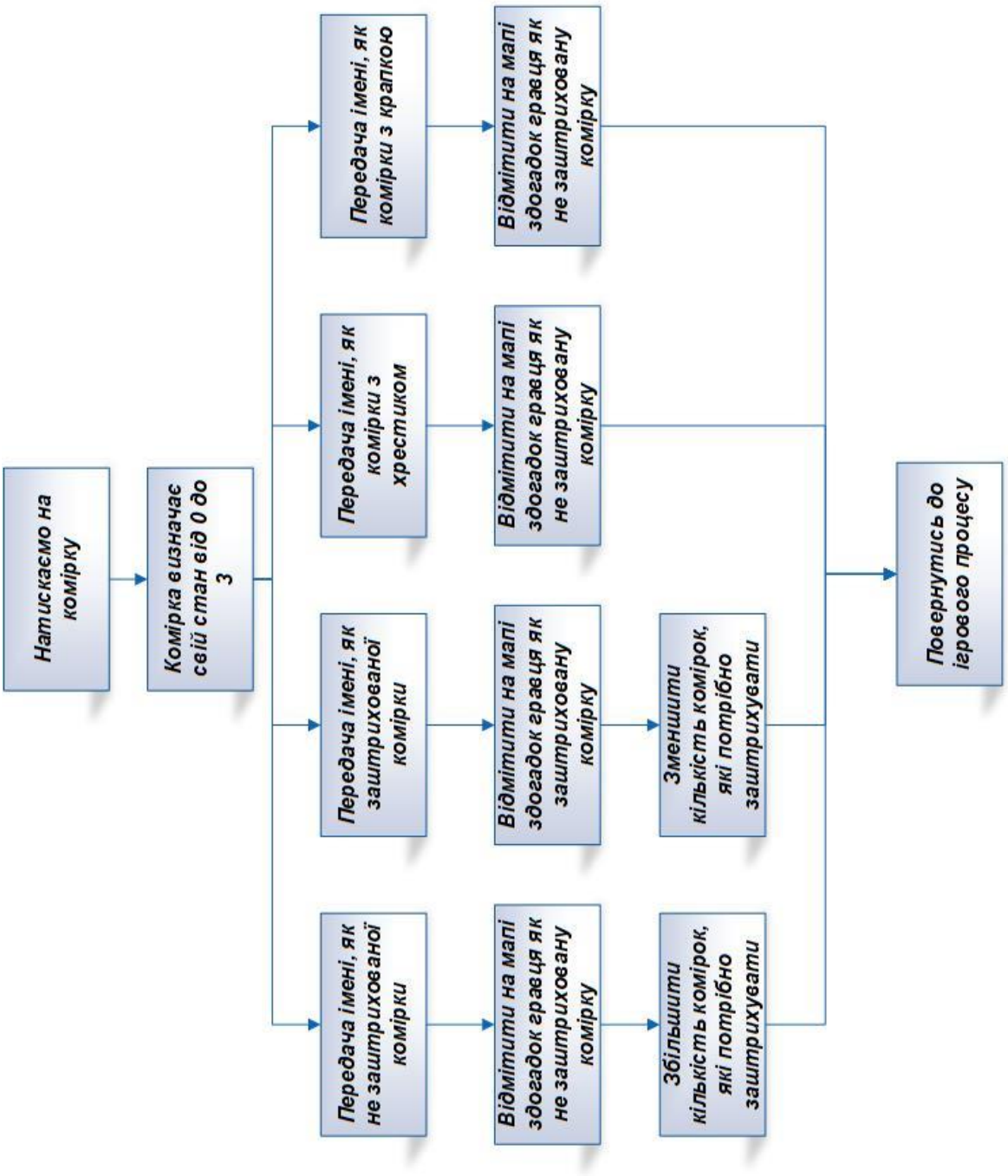


Рисунок 1.25. Модифікована схема передачі імені комірки до ігрового менеджеру

З точки зору реалізації всього цього функціоналу, нижче представлено фрагмент коду, який отримує ім’я комірки. Також тут можна побачити, що в залежності від переданого стану позначення комірки, виконується те, чи інше позначення на мапі здогадок гравця.

```
public void _take_cell_name(string painted_cell_name, bool cell_state){
        for (int i = 0; i < 6; ++i){
                for (int i2 = 0; i2 < 6; ++i2){
```

```
                    if(cells_name[i,i2] == painted_cell_name){
                        if(cell_state){
                            _mark_player_guess(i, i2);
                        }
                        else{
                            _unmark_player_guess(i, i2);
                        }
                    }
                }
            }
        }
```

Також нижче представлено код реалізації відмічання здогадок гравця на мапі, або навпаки, зняття позначки:

```
    void _mark_player_guess(int x_coord_player_guess, int y_coord_player_guess){
        player_gues[x_coord_player_guess, y_coord_player_guess] = true;
    }

    void _unmark_player_guess(int x_coord_player_guess, int y_coord_player_guess){
        player_gues[x_coord_player_guess, y_coord_player_guess] = false;
    }
```

Вище наведені два методи відрізняються один від одного у зв’язку з тим, що один проставляє логічну істину на мапі у разі відмічання здогадки, а інший проставляє логічне неправдиво у разі зняття відмітки здогадки. Також нижче приведено код, який змінює зміст текстового елементу ігрового інтерфейсу:

```
    public void _cell_painted(){
        --cells_count;
        cells_count_text.text = "Залишилось знайти: " + cells_count;
    }

    public void _cell_unpainted(){
        ++cells_count;
        cells_count_text.text = "Залишилось знайти: " + cells_count;
    }
```

Можна побачити, що ці два простих методи збільшують, або зменшують лічильник комірок, які залишилось відмітити, після чого змінює зміст текстового елемента.

Останній елемент, який потрібно реалізувати – це перевірка правильності вирішення кросворду гравцем. Як вже було позначено вище, перевірка

виконується завдяки звірки двох двомірних масивів – масиву мапи правильних відповідей та масиву мапи здогадок гравця. Цей код виконується при натисканні кнопки Перевірити. На рисунку 1.26 зображено схему перевірки правильності розв’язання кросворду:

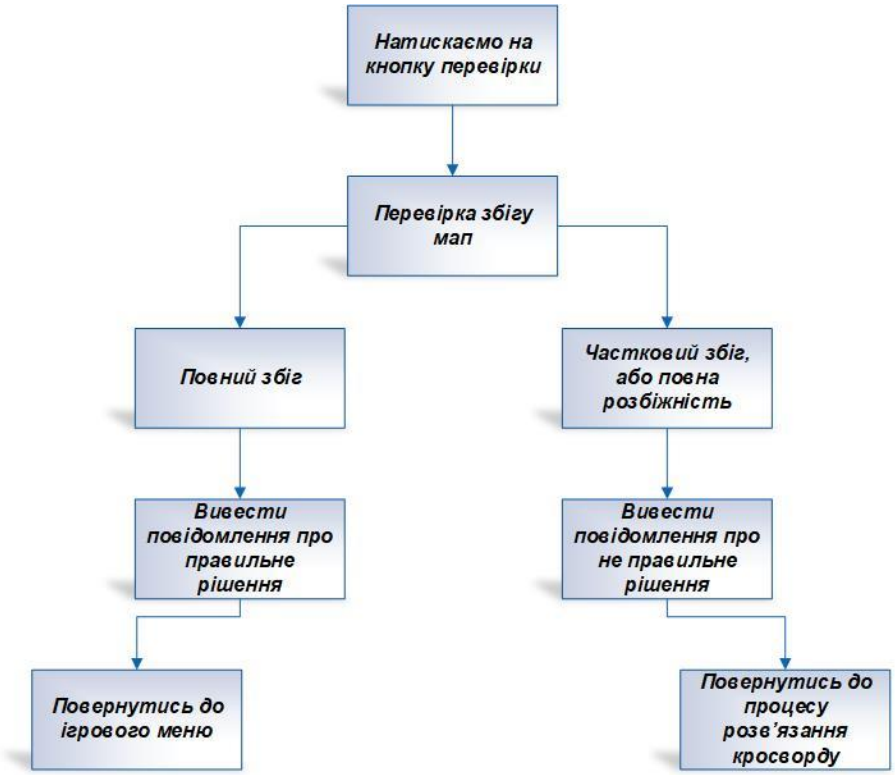


Рисунок 1.26. Схема перевірки правильності розв’язання кросворду

Як видно зі схеми, у разі збігу обох мап, гра має вивести вітання та завантажити головне меню гру. В іншому разі, навіть якщо є частковий збіг, гра повинна вивести повідомлення про помилку та повернути гравця до кросворду. Нижче приведено фрагмент коду, який виконує перевірку мап:

```
public void _check_painted_cells(){
    bool result = true;
    for(int i=0; i<6; ++i){
        for (int i2 = 0; i2 < 6; ++i2){
            if (cells_right[i, i2] != player_gues[i, i2])
                result = false;
        }
    }
    if (result){
```

```
                _player_ui_manager._show_won_panel(true);
                _player_ui_manager.show_main_menu();
        }
        else{
                _player_ui_manager._show_won_panel(false);
        }
}
```

З коду видно, що спочатку роботи методу статус правильності у змінній result виставляється як логічна істина. Тільки потім іде перевірка на помилку. Будуть знайдені розбіжності, то статус правильності result буде встановлено у логічне неправдиво. І далі, незалежно від результату подальшої перевірки, цей статус залишиться. Після перевірки масивів мап, починається перевірка результатів, яка або викликає повідомлення про перемогу та головне меню, або повідомлення про помилку, та повертає до ігрового процесу.

В кінці процесу розробки проект отримав свою кінцеву структуру, яку можна побачити на рисунку 1.27. Є папки, які належать ігровому програмному рушію Unity і працювати із ними частіше за все не доводиться, тільки у випадках внесення змін у тонкі механізми роботи рушія.

Рисунок 1.27. Кінцева структура проекту гри Японський кросворд

Основі ж файли та папки проекту знаходяться у каталозі Assets, зміст якого можна бачити на рисунку 1.28. Файли проекту було структуровано за

призначенням та вмістом, а також розміщено у відповідних підкаталогах. Так, всі сцени знаходяться в папці Scenes, а скрипти для гри в каталозі scripts, спрайти, які використовуються для гри, розміщені у каталозі Sprites.

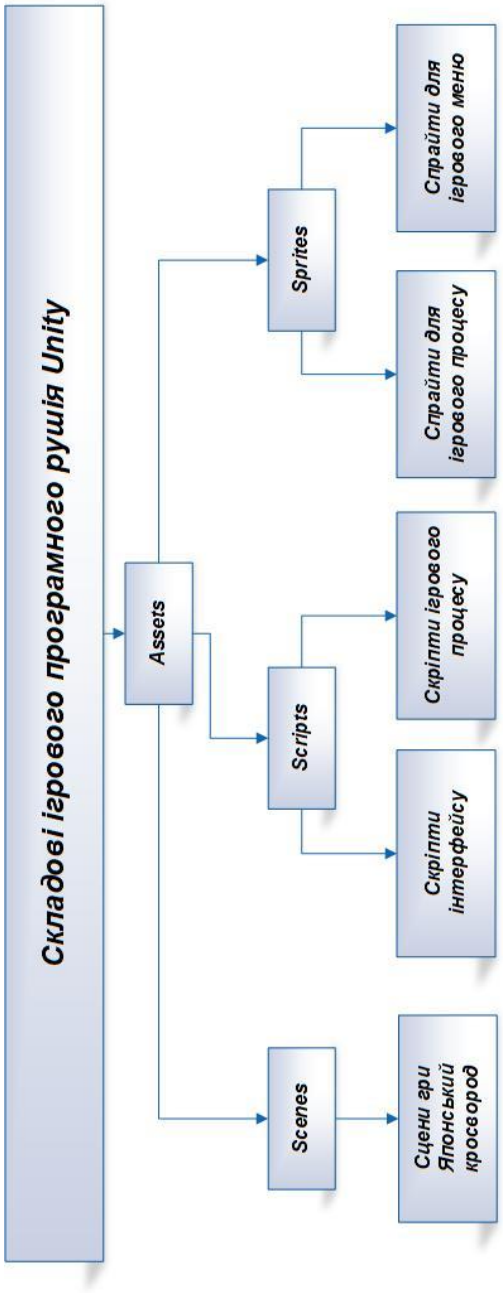


Рисунок 1.28. Зміст каталогу Assets проекту Японський кросворд

Проект залишає широкі можливості для покращення ігрового досвіду та якості гри. Так, наприклад, можна додати до гри механіку підказок, яка могла б допомогти починаючим гравцям розібратись із тим, як саме розв’язуються японські кросворди. Підказки можна зробити різних видів. Наприклад такі, що показують комірку на деякий час, або такі, що розкривають всю горизонталь, або

вертикаль.

Така ідея підказок наштовхує на необхідність запровадити якийсь ресурс, що робив би такі підказки більш ціннішими за інші. Наприклад можна ввести таке поняття, як ігровий рахунок гравця. За розв’язування кросвордів, гравець отримує деяку кількість ігрового рахунку. Цю величину можна використовувати для отримання підказок різного рівня. Кількість рахунку, який гравець отримає за розв’язання рівня залежить від складності кросворду.

Втім, необхідність вводити ігровий рахунок наштовхує на ідею створення додаткового рівня розваг для гравця. Наприклад, можна ввести ресурси, які гравець отримує за розв’язання кросворді. Після чого ці ресурси можна витрачати на будівництво, скажімо кузні, яка могла б виробляти підказки з якоюсь чистотою, в залежності від рівня підказки.

Можна створити мережу гравців, які отримали би змогу передавати ресурси, від одного користувача до іншого, створювати клани та збирати ресурси разом, для будівництва кланового міста.

Кількість ідей, які можна імплементувати у даний проект дуже велика, від таких, зо були зазначені вище, до кланових війн, рангової таблиці гравців, відкритого редактору кросвордів, та багато інших різних нововведень.

Все це відкриває двері для майбутнього портування гри на інші платформи, наприклад смартфони. Це дасть змогу значно збільшити кількість гравців, а також надає пасивний доход у вигляді переглядів реклами.

1.7 Тестування працездатності ігрових елементів

Використовуючи функціонал ігрового програмного рушія Unity, тестування та відладку гри можна проводити у процесі її створення. Для цього достатньо розмістити ігрові об’єкти на сцені, розмістити елементи інтерфейсу у необхідне положення, налаштувати відповідні параметри та натиснути кнопку «Play». Після цього гра запускається середою розробки, що показано нижче на рисунку 1.29.

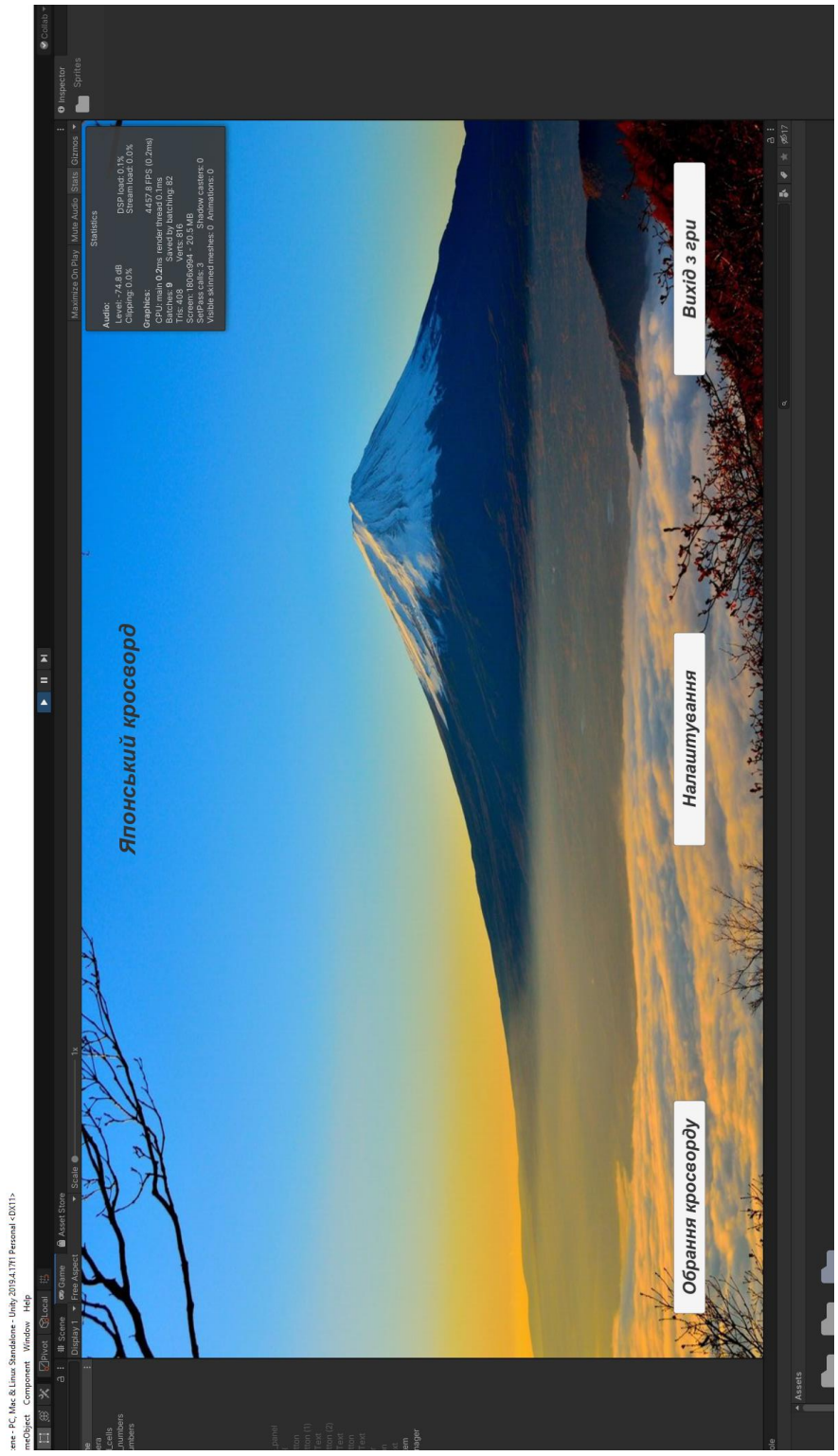


Рисунок 1.29. Процес тестування гри за допомогою режиму «Play»

Основною метою тестування є тестування працездатності ігрового меню, його елементів. Наскільки коректним є взаємодія між елементами меню. Чи не виникає помилок під час звернення до якихось кнопок, або у разі натискання кнопок, які не було передбачено налаштуваннями гри.

Окремо перевіряється ігровий процес. Для цього потрібно було перевірити

справність роботи всіх елементів гри, після чого натиснути «Обрання кросворду», обрати будь-який кросворд та спробувати вирішити його.

Під час вирішення кросворду було знайдено помилку – не було прив’язки до ігрового об’єкту у менеджері ігрового процесу. Помилка була дуже легко вирішена шляхом перетягування ігрового об’єкту до відповідної комірки у компоненті скрипта менеджеру ігрового процесу.

Таким чином, виконавши тестування, можна переконатись, що гра коректно виконує основні вимоги сформовані під час проектування проекту. Подальші модифікації модулів гри можуть розширити ігровий процес. Серед модифікацій модулів можуть бути додавання нових режимів, створення мережевої системи, або додавання до гри режимів кооперативної гри для сумісного вирішення кросвордів. Навіть створення ігрового процесу основаного на конкурсній основі по швидкості вирішення кросворду.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проекті виконано розробку та реалізацію гри Японський кросворд на ігровому програмному рушії Unity. Виготовлений програмний продукт дає змогу зіграти в гру, яку можна запустити на операційний системі Microsoft Windows. Розробка програмного продукту виконувалась за допомогою умовно безкоштовного ігрового програмного рушія Unity із використанням редактора коду Microsoft Visual Studio, систем контролю версій та графічного редактора.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

Проведемо розрахунки визначення трудомісткості розробки даного програмного продукту.

2.2 Розрахунок ціни програмного продукту нормативним методом

2.2.1 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку.

Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.

Таблиця 2.1

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	950 – 7430
3. ПП введення інформації	1060 – 5750
4. ПП оптимізації розрахунків	1300 – 4200
5. ПП автоматизації засобів по каталогу	680 – 7000
6. ПП автоматизованих розрахунків	1300 – 8600
7. ПП загальної математики і ПП імітаційного моделювання	7800 – 8800
8. ПП організації обчислювального процесу	13000 – 10200

Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця 2.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262
4.00	283
5.00	306
6.00	330
7.00	357
8.00	385
9.00	414
10.00	445

12.00	510
14.00	580
16.00	654
18.00	731
20.00	812

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,7 = 160,3 (люд/годин).

Трудомісткість програмного продукту визначається 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. 2.2);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4).

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 $К_н$
А	Принципово нове ПЗ	1,75 – 1,2
Б	ПЗ – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПЗ, що має аналог	0,7

Для нашого варіанта виділено сірим кольором.

Тому що розробка системи є ПЗ, що має аналоги програмних продуктів, то код ступеня новизни для мого ПЗ – В, а значення коефіцієнта $К_н$=0,7. По таблиці 2.3, знаючи код ступеня новизни, тепер можна визначити значення питомих коефіцієнтів трудомісткості:

L_1=0,12;

L_2=0,11;

L_3=0,61;

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

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

Для нашого варіанта виділено сірим кольором.

У розробленому програмному продукті використовується від 40 до 60 відсотків існуючих функцій, це значить, що $К_т$=0,7.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$T_{тз}=Ta*L_1*K_н=160,3 *0,12*0,8= 15,38$ (люд/годин) (2.4)

Трудомісткість розробки технічного проекту

$T_{тп}=Ta*L_2*K_н=160,3 *0,11*0,8 = 14,11$ (люд/годин) (2.5)

Трудомісткість розробки робочого проекту

$T_{рп}=Ta*L_3*K_н*K_т=160,3 *0,61*0,8*0,8= 62,58$ (люд/годин) (2.6)

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап:

- технічне завдання $N_{тз}=3$ (стр),

- розробка ТП $N_{тп}=15$(стр),

- розробка робочого проекту $N_{рп}=20$ (стр),

- пояснювальна записка відповідно $N_{пз}=30$ (стр)

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Ртз}=15,38$	$T_{кк}=0,7*N_{тз}= 0,7*3=2,1$	$T_{нк}=0,15*N_{тз}=0,15*3=0,45$
2.Розробка ТП	$T_{Ртп}=14,11$	$T_{кк}=0,7*N_{тп}=0,7*15=10,5$	$T_{нк}=0,15*N_{тп}=0,15*15=2,25$
3.Розробка РП	$T_{Ррп}= 62,58$	$T_{кк}=0,7*N_{рп}=0,7*20=14,0$	$T_{нк}=0,15*N_{рп}=0,15*20=3,0$
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}= 1,5*30 =45$	$T_{кк}=0,7*N_{тз}=0,7*30=21,0$	$T_{нк}=0,15*N_{пз}=0,15*30=4,5$
Усього, в т.ч.:	194,87		
- на розробку	$\Sigma T_р=137,07$		
- контроль керівника		$\Sigma T_{кк}=47,6$	
- нормоконтроль			$\Sigma T_{нк}=10,2$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Роботодавець або уповноважені ним органи зобов'язані дбати про умови праці працівників, полегшувати їх, оздоровляти навколишнє середовище, дбати про виконання правил безпеки і інструкцій по техніці безпеки.

Координує всю цю діяльність служба охорони праці, яка в залежності від чисельності працюючих може функціонувати як самостійний структурний підрозділ (число працюючих 50 і більше), або у вигляді групи спеціалістів чи одного спеціаліста, у тому числі за сумісництвом (число працюючих 20 і менше). Задачі службі охорони праці та її функції викладені в Типовому положенні про службу охорони праці», яке затверджено наказом Комітетом Держнаглядохоронпраці (ДНАОП 0.00-4.21-93) .

Працівники також повинні відповідально ставитись до охорони праці, знати та виконувати вимоги, визначені нормативною документацією. В сучасних умовах кожному працівнику необхідно постійно підтримувати високий фізичний, психологічний та фаховий рівень, запобігати виникненню випадків травматизму та профзахворювань.

Безпечні умови праці на підприємстві досягаються за рахунок забезпечення безпеки виробничих процесів, які обґрунтовані і прийняті в технологічній частині дипломного проекту.

3.1 Аналіз небезпечних та шкідливих чинників, що впливають на працівника

Для установлення можливого впливу на здоров'я користувачів ВДТ виробничих чинників має значення ряд якісних характеристик робочого середовища. Це середовище у приміщеннях (офісах) в основному характеризується такими фізичними параметрами, як температура, вологість та електричний опір підлоги. Фізико-хімічні показники включають інформацію про вміст у повітрі іонів та різноманітних забруднювачів, а також деякі інші якісні характеристики середовища

3.2 Розробка заходів з охорони праці

3.2.1 Виробничі приміщення

Будівлі та приміщення, де розміщені робочі місця програмістів повинні відповідати вимогам СНіП 2.09.02-85 «Производственные здания» та ДСанПіН 3.3.2.007 «Державні санітарні правила і норми роботи з ВДТ ЕОМ» Вони мають бути не нижче другого ступеня вогнестійкості. Для всіх приміщень повинно бути визначено клас зони згідно з НПАОП 40.1-1.01-97. Відповідне позначення повинно бути нанесено на вхідних дверях кожного приміщення.

Не дозволяється розташування приміщень з робочими місцями операторів ПК у підвалах і цокольних поверхах. Площа приміщення із розрахунку на одне робоче місце має бути не менше 6,0 кв.м, а об'єм – не менше 20,0 куб.м.

Для внутрішнього оздоблення приміщень з ПК слід використовувати дифузно-відбивні матеріали з коефіцієнтом відбитті для стелі 0,7 – 0,8, для стін 0,5 – 0,6. Покриття підлоги повинне бути матовим, поверхня рівною, не слизькою, з антистатичними властивостями.

Віконні прорізи приміщень для роботи з ПК мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки).

Забороняється для оздоблення інтер'єру приміщень з ПК застосовувати полімерні матеріали, що виділяють у повітря шкідливі хімічні речовини. Приміщення можуть обладнуватись шафами для зберігання документів, полицями, стелажами.

У приміщеннях слід щоденно робити вологе прибирання. Вони мають бути оснащені аптечками першої медичної допомоги.

При приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження, де слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04 – 87).

3.2.2 Мікроклімат робочої зони працівників, вентиляція

Висока температура повітря негативно позначається на функціональному стані людини. Хоч генерація теплоти дисплеєм досягає критичного рівня тільки у саму теплу пору року, необхідно створювати комфортні теплові умови постійно.

Оптимальні та допустимі мікрокліматичні параметри у приміщеннях повинні враховувати специфіку технологічного процесу при використанні комп'ютерів. Згідно з діючими у нашій країні нормативними документами (ДСанПіН 3.3.2-007-98 у холодні періоди року температура повітря, швидкість його руху та відносна вологість повітря повинні відповідно складати: 22-24^{0}С; 0,1 м/с; 40-60%. Температура повітря може коливатись у межах від 21 до 25^{0}С при збереженні інших параметрів мікроклімату.

В теплі періоди року температура повітря, його рухливість та відносна вологість повинні відповідно становити: 23-25^{0}С; 0,1-0,2 м/с; 40-60 %.

Оптимальним рівнем аероіонізації у зоні дихання користувача вважається вміст легких аерофонів обох знаків від 150 до 5000 у 1 см^3 повітря.

Нормалізуючий вплив на склад повітря робочої зони справляють примусова вентиляція, захисні екрани (оснащені заземленням) та застосування іонізаторів.

3.2.3 Освітлення робочого місця, шум, вібрація

Освітлення у приміщеннях з ВДТ має бути змішаним – природним та штучним. Природне освітлення повинно здійснюватись у вигляді бічного освітлення та відповідати нормам ДБН В.2.5-28-2006 «Природне і штучне освітлення».

При природному освітленні слід передбачити наявність сонцезахисних засобів, що знижують перепади яскравостей між природним світлом та свіченням екрана ВДТ. З цією метою можна використовувати плівки з металізованим покриттям або жалюзі з вертикальними лам елями, що регулюються.

Штучне освітлення у приміщеннях з ВДТ треба здійснювати у вигляді комбінованої системи освітлення з використовуванням люмінесцентних джерел світла у світильниках загального освітлення. На робочих місцях має бути

забезпечена рівномірна освітленість за допомогою переважно відбитого або розсіяного світлорозподілу. Світлових відблисків з клавіатури, екрана та від інших частин ВДТ у напрямку очей користувача не повинно бути.

Норма освітленості на робочих місцях складає 300-500лк.

Деякі ВДТ є потенційними джерелами цілого ряду звуків, що містять як коливання, які можна почути, так і коливання ультразвукового діапазону. Цей шум справляє негативний вплив на стан користувача, особливо при тривалому впливі.. У користувача, діяльність якого пов'язана з переробкою інформації це виражається у знижені розумової працездатності, зростає кількість помилок, розвиток зорового втомлення, зміні відчуття кольорів, появі головного болю, послаблення уваги. Нормованим параметром шуму на робочих місцях є рівень 50 дБ. Основними заходами боротьби з шумом є усунення або ослаблення причин шуму в самому його джерелі у процесі проектування, використання засобів звукопоглинання, раціональне планування виробничих приміщень.

3.2.4 Електробезпека

Причинами ураження працівника електрострумом можуть бути:

- Випадковий дотик до струмоведучих частин, у результаті ведення робіт поблизу або на цих частинах;
- Випадковий дотик до струмоведучих частин, у результаті ведення робіт поблизу або на цих частинах;
- Несправність захисних засобів, якими потерпілий доторкався до струмоведучих частин;

Помилкове прийняття устаткування, що перебуває під Електробезпека.

Значення сили струму, що проходить через організм людини, залежить від напруги, під якою перебуває людина й від опору ділянки тіла, до якого прикладена ця напруга. Джерелом живлячої напруги є мережа змінного струму з напругою 229В, на яку поширюється ГОСТ 25861-83.

Основними причинами електротравматизму є:

- напругою, як відключеного;
- Несподіване виникнення напруги через ушкодження ізоляції там, де в

нормальних умовах його бути не повинно;

- Контакт струмопровідного устаткування із проводом, що перебуває під напругою.

Для попередження поразок електричним струмом необхідно чітко й у повному обсязі виконувати правила провадження робіт і правил технічної експлуатації. Необхідно виключити можливість доступу оператора до частин устаткування, що працює під небезпечною напругою, до неізольованим частинам, призначеним для роботи при малій напрузі й не підключеним до захисного заземлення, а також підводити електроживлення до ПЕОМ від розетки за допомогою спеціальної вилки із заземлюючим контактом.

3.2.5 Організація робочого місця користувача ПК

Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування, ергономічним вимогам, з урахуванням характеру і особливостей трудової діяльності (ДСанПіН 3.3.2.-007-98).

Конструкція робочого місця й взаємне розташування всіх його елементів відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів дає можливість забезпечувати можливість індивідуального регулювання їх відповідно до потреб працівника для підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей, на відстані близько 70 см, що укладається в припустимі рамки від 60 до 90 см. Частота мерехтіння екрана дорівнює 100 Гц, що відповідає умові більше 70 Гц.

Для зниження нервово-емоційного напруження, стомлювання, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно впроваджувати виконання комплексу вправ.

ВИСНОВКИ

Для виконання дипломного проекту було обрано простий японський кросворд, який давав змогу зосередитись на розробці гри. Сам кросворд кодується у спеціальному файлі. Гра завантажує цей файл, декодує його та відображає кросворд із заданою кількістю комірок.

Візуальну складову було виконано за допомогою простих спрайтів намальованих у Adobe Photoshop. Текст гри відображається за допомогою ігрового рушія, а саме його інструментом UI та Canvas. Простий мінімалістичний інтерфейс дає змогу зосередитись на ігровому процесі. В той же час, залишає простір для майбутніх нововведень, які могли б додатково розширювати ігровий процес.

Перед реалізацією проекту, було виконано його проектування у вигляді взаємодії блок-схем мінімальних логічним функцій. Це спростило роботу із кодом і додало можливість легко створювати зв’язки між ігровими об’єктами.

Розроблявся проект на мові програмування C# у середі розробці Visual Studio. Unity легко працює із цією середою, тому під час розробки було легко орієнтуватись у проекті та його будові. Було використано основи ООП для створення відповідних класів. В кінці розробки проекту, проведено перевірку коду на помилки, та пошук несправностей у логікі роботи ігрових об’єктів та елементів.

В кінці було отримано ігровий проект, який повністю відповідає основній меті розробці – реалізувати гру Японский кросворд. А його механіка завантаження кросвордів дає змогу збільшувати їх кількість.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Ерік Фрімен, Елізабет Робсон, Берт Бейтс, Кеті Сієрра, Книга Head First. Патерни проектування, Київ, Фабула, 672 ст.
2. Коноваленко І.В., Програмування мовою C# 6.0, Тернопіль, ТНТУ, 2016 р., 227 ст.
3. Фленов М., Біблія C# 3-е видання., Київ, Фабула, 2016 р. 546 ст.
4. Алан Торн, Мистецтво створення сценаріїв в Unity, Київ, видавництво ДМК, 362 ст.
5. Дікінсон К. Оптимізація ігор у Unity 5. Поради і методики, Оптимізація додатків охоплюючи всі аспекти роботи у ігровому движку Unity 3D., Київ, Фабула, 2017 р., 306 ст.
6. Andrew R., Dave M. GAME Architecture and Design – A new Edition, Shelter Island, New York: Manning Publications. 1035 p.
7. Goldstone W. Unity Game Development Essentials, New York, PACKT, 266 p.
8. Hocking J. Unity in Action: Multiplatform Game Development in C# with Unity 5. Shelter Island, New York: Manning Publications. 352 p.
9. Unity User Manual: [Website]. URL: https://docs.unity3d.com/Manual/index.html (viewed on: 21.05.2023).
10. Unity: [Website]. URL: https://unity.com/en-us (viewed on: 21.05.2023).
11. C# docs: [Website]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (viewed on: 21.05.2023).

ДОДАТОК А. Лістинг коду основних модулів гри мовою C#

Скрипт cell_logic.cs

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

public class cell_logic : MonoBehaviour{
  int mouse_state = 0;
  [SerializeField] game_manager _game_manager;
  SpriteRenderer _sprite_renderer;
  [SerializeField] Sprite _empty_cell;
  [SerializeField] Sprite _painted_cell;
  [SerializeField] Sprite _crossed_cell;
  [SerializeField] Sprite _dot_cell;

  void Start(){
        _sprite_renderer = GetComponent<SpriteRenderer>();
  }
  public void OnMouseDown(){
        if (mouse_state == 0)
        {
              mouse_state = 1;
              _sprite_renderer.sprite = _painted_cell;
              _game_manager._cell_painted();
              _game_manager._take_cell_name(gameObject.name, true);
        }
        else if (mouse_state == 1){
              mouse_state = 2;
              _sprite_renderer.sprite = _crossed_cell;
              _game_manager._cell_unpainted();
              _game_manager._take_cell_name(gameObject.name, false);
        }
        else if (mouse_state == 2){
              mouse_state = 3;
              _sprite_renderer.sprite = _dot_cell;
              _game_manager._take_cell_name(gameObject.name, false);
        }
        else{
              mouse_state = 0;
              _sprite_renderer.sprite = _empty_cell;
              _game_manager._take_cell_name(gameObject.name, false);
        }
  }
}
```

Скрипт game_manager.cs

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

```
using UnityEngine.UI;

public class game_manager : MonoBehaviour{
  [SerializeField] player_ui_manager _player_ui_manager;
  string[,] cells_name = new string[6, 6];
  bool[,] cells_right = new bool[6, 6];
  bool[,] player_gues = new bool[6, 6];
  int cells_count = 0;
  [SerializeField] Text cells_count_text;

  void Start(){
        for(int i=0; i<6; ++i){
              for (int i2 = 0; i2 < 6; ++i2){
                    cells_name[i, i2] = "c_" + i + "_" + i2;
              }
        }
        for (int i = 0; i < 6; ++i){
              for (int i2 = 0; i2 < 6; ++i2){
                    cells_right[i, i2] = false;
              }
        }
        for (int i = 0; i < 6; ++i){
              for (int i2 = 0; i2 < 6; ++i2){
                    player_gues[i, i2] = false;
              }
        }
        cells_right[1, 1] = true;
        cells_right[1, 3] = true;
        cells_right[2, 2] = true;
        cells_right[2, 4] = true;
        cells_right[3, 1] = true;
        cells_right[3, 2] = true;
        cells_right[3, 4] = true;
        cells_right[3, 5] = true;
        cells_right[4, 2] = true;
        cells_right[4, 5] = true;
        cells_right[5, 2] = true;
        cells_right[5, 5] = true;
        for (int i = 0; i < 6; ++i){
              for (int i2 = 0; i2 < 6; ++i2){
                    if (cells_right[i, i2])
                    ++cells_count;
              }
        }
        cells_count_text.text = "Залишилось знайти: " + cells_count;
  }

  public void _cell_painted(){
        --cells_count;
        cells_count_text.text = "Залишилось знайти: " + cells_count;
```

```
    }

    public void _cell_unpainted(){
        ++cells_count;
        cells_count_text.text = "Залишилось знайти: " + cells_count;
    }

    public void _take_cell_name(string painted_cell_name, bool cell_state){
        for (int i = 0; i < 6; ++i){
            for (int i2 = 0; i2 < 6; ++i2){
                if(cells_name[i,i2] == painted_cell_name){
                    if(cell_state){
                        _mark_player_guess(i, i2);
                    }
                    else{
                        _unmark_player_guess(i, i2);
                    }
                }
            }
        }

    void _mark_player_guess(int x_coord_player_guess, int y_coord_player_guess){
        player_gues[x_coord_player_guess, y_coord_player_guess] = true;
    }

    void _unmark_player_guess(int x_coord_player_guess, int y_coord_player_guess){
        player_gues[x_coord_player_guess, y_coord_player_guess] = false;
    }

    public void _check_painted_cells(){
        bool result = true;
        for(int i=0; i<6; ++i){
            for (int i2 = 0; i2 < 6; ++i2){
                if (cells_right[i, i2] != player_gues[i, i2])
                result = false;
            }
        }
        if (result){
            _player_ui_manager._show_won_panel(true);
            _player_ui_manager.show_main_menu();
        }
        else{
            _player_ui_manager._show_won_panel(false);
        }
    }
}
```

Скрипт player_ui_manager.cs

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

```
using UnityEngine.UI;

public class player_ui_manager : MonoBehaviour
{
  [SerializeField] GameObject won_panel;
  [SerializeField] GameObject player_won_text;
  [SerializeField] GameObject player_lose_text;
  [SerializeField] GameObject chose_panel;
  [SerializeField] GameObject options_panel;
  [SerializeField] GameObject exit_button;
  [SerializeField] GameObject chose_button;
  [SerializeField] GameObject options_button;
  [SerializeField] GameObject return_button;
  [SerializeField] GameObject wallpaper;
  [SerializeField] GameObject title;

  public void _show_won_panel(bool player_won_status){
        won_panel.SetActive(true);
        if (player_won_status)
              player_won_text.SetActive(true);
        else
              player_lose_text.SetActive(true);
  }

  public void _close_won_panel(){
        won_panel.SetActive(false);
        player_won_text.SetActive(false);
        player_lose_text.SetActive(false);
  }

  public void show_chose_panel(){
        chose_button.SetActive(false);
        options_button.SetActive(false);
        exit_button.SetActive(false);
        chose_panel.SetActive(true);
        return_button.SetActive(true);
  }

  public void show_main_menu(){
        chose_button.SetActive(true);
        options_button.SetActive(true);
        exit_button.SetActive(true);
        chose_panel.SetActive(false);
        return_button.SetActive(false);
  }

  public void start_game(){
        chose_button.SetActive(false);
        options_button.SetActive(false);
        exit_button.SetActive(false);
```

```
            chose_panel.SetActive(false);
            return_button.SetActive(false);
            wallpaper.SetActive(false);
            title.SetActive(false);
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Розробка гри Японський кросворд на програмному рушії Unity

ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА ГРУПИ 4РП-07: ЩЕРБАКОВА ДМИТРА СЕРГІЙОВИЧА

КЕРІВНИК: СУЛІМА Ю.Є.

Особливості ігрового процесу японських кросвордів;

Японські красворди, або нонограми мають не таку велику популярність, як звичайні кросворди.

Втім вони відрізняються приємним процесом вирішення.

До особлививостей ігрового процесу відносять:

- Використання цифрових позначень по горизонталі та вертикалі полів кросворду;
- Визначення кількості клітинок необхідних для замальовування;
- Кожна клітинка, або лінія завжди розділяється однією пустою клітинкою;
- Можливість проставляти позначки на полях нонаграм.

Особливості програмного рушія Unity та причини його вибору для розробки проекту

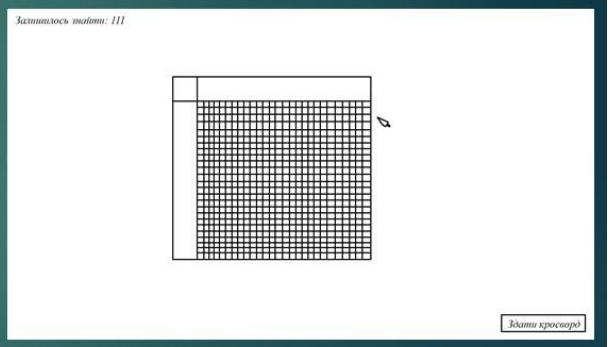
Ігровий двигун Unity є одним із популярніших ігрових двигунів у світі. За його допомогою створюють як мобільні, так і комп'ютерні ігри. Цей двигун досі активно розвивається!

- Має зручну візуальну середу розробки;
- Має можливість міжплатформенної розробки ігор;
- Підтримує модульну систему компонентів;
- Використання мови програмування C# та його можливостей;
- Велике ком'юніті розробників;
- Зрозуміла та вичерпна документація, безліч курсів у інтернеті та літературі;
- Гнучка система ліцензування ігрового двигуна;
- Безплатні та платні ассети для проектів будь-якого рівня.

Особливості ігрових механік розроблюємого проекту

Можна виділити такі особливості розроблюємого проекту:

- Гра має можливість завантажувати свої кросворди для розв'язування, або ділитись ними з іншими;
- Можливо позначати клітинки різними позначками: замалювати, хрестик, крапка, зробити клітинку пустою;
- Гра сама рахує за гравця кількість клітинок, які потрібно замалювати;
- Вбудована система перевірки правильності вирішення кросворду.



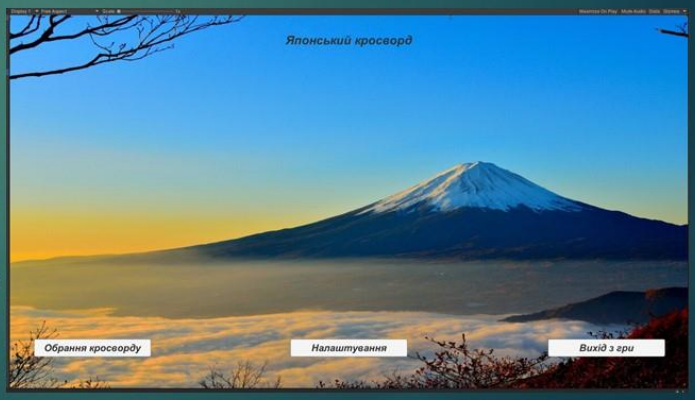
Огляд основних елементів ігрового процесу

Основні елементи ігрового процесу можна поділити на два етапи:

- Робота у головному меню гри;
- Вирішення кросворду.

Головне меню дає змогу гравцю:

- Виконати налаштування гри;
- Обрати кросворд для вирішення;
- Вийти з гри.

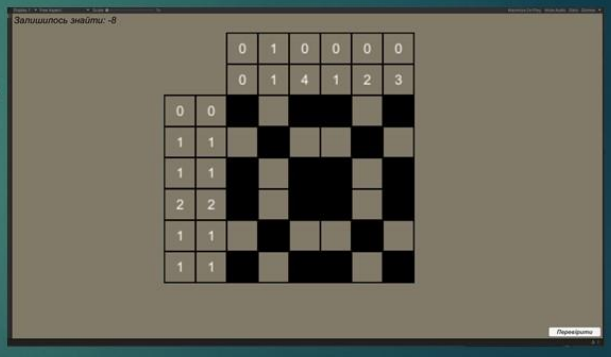


Огляд основних елементів ігрового процесу

Етап вирішення кросворду починається після обрання і завантаження кросворду з переліку кросвордів у папці з грою.

Складається з:

- Вирішення кросворду за допомогою миші;
- Позначення клітинок різних типів;
- Перевірки правильності вирішення.



Принцип роботи основних елементів ігрового процесу

Стан ігрового меню складається із двох модулей:

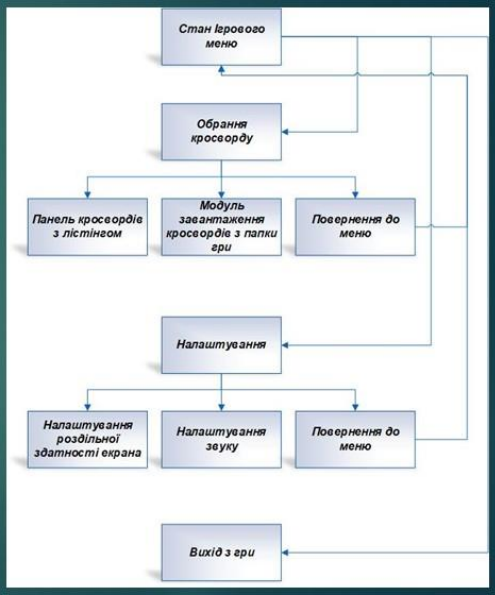
- Обрання кросворду;
- Налаштування.

- Обрання кросворду дає змогу вибрати кросворд із переліку файлів у відповідній папці гри. Додавати кросворди можна завантаженням туди нових файлів.

- Налаштування дають змогу налаштувати звук, та примітивні графічні налаштування.

- Вихід з гри закриває додаток.

Ці функції реалізовані менеджером інтерфейсу та відповідним полотном Unity

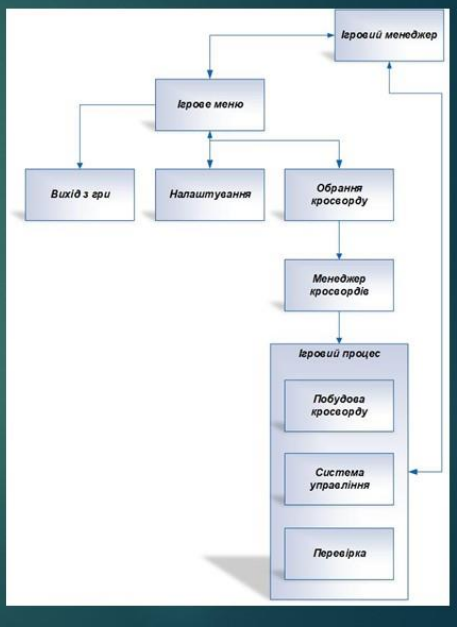


Принцип роботи основних елементів ігрового процесу

Ігровий процес побудований на взаємодії різних станів гри:

- Стан ігрового меню;
- Стан ігрового процесу;

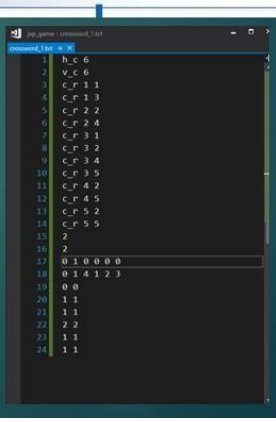
Кожний з цих станів має свій інтерфейс та код, якій відповідає за модулі в ньому



Принцип роботи основних елементів ігрового процесу

Процес побудови кросворду складається з:

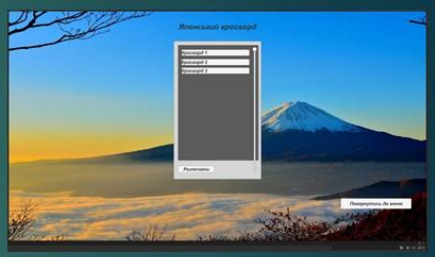
- Завантаження кількості горизонтальних комірок;
- Завантаження кількості вертикальних комірок;
- Зчитування координат правильних комірок;
- Створення карти правильних комірок;
- Генерації комірок;
- Створення цифрових позначень.

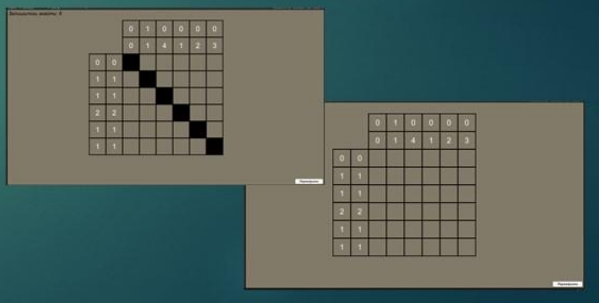


Етапи реалізації основних елементів ігрового процесу

Реалізація основних елементів гри складалась з:

- Реалізації головного меню;
- Реалізації системи завантаження кросвордів;
- Реалізації системи зчитування файлів кросвордів та їх побудови;
- Елементів управління;
- Логіки комірок;
- Перевірки правильності вирішення кросворду.



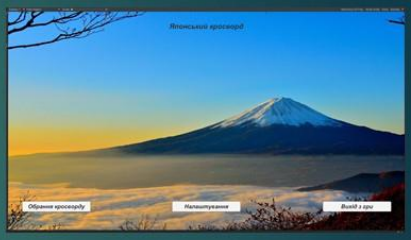


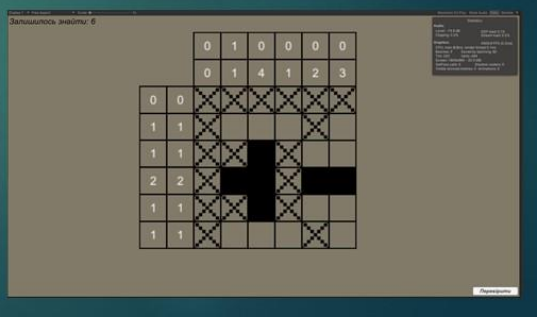
Хід тестування гри

Тестування гри проводилось з допомогою вбудованого у ігровий програмний рушій Unity інструмент.

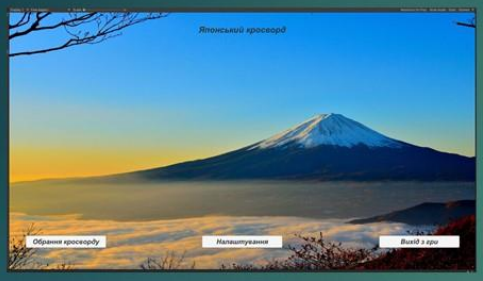
Метою тестування було:

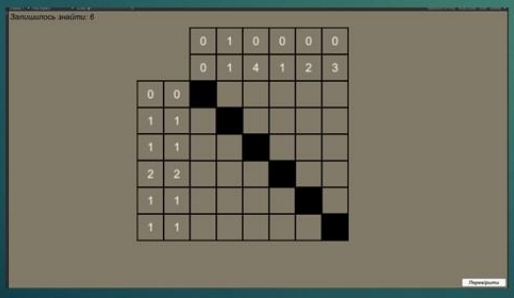
- Визначити справність роботи ігрового меню;
- Визначити справність роботи алгоритму завантаження кросвордів;
- Визначити справність роботи ігрової логіки та перевірки правильності вирішення кросворду.

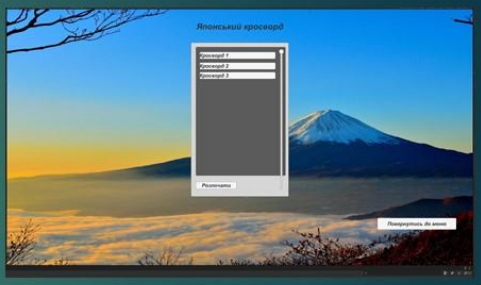


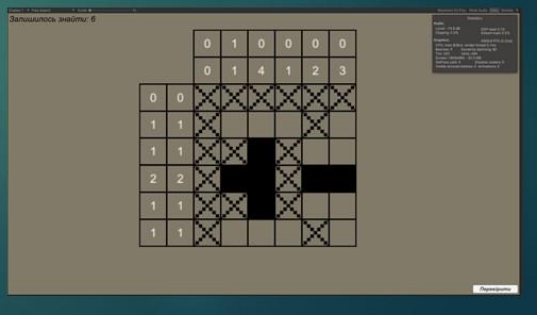


Скріншоти розробленої гри









ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Щербакова Дмитра Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність 121 " Інженерія програмного забезпечення "

Освітня програма «Розробка програмного забезпечення»

Керівник дипломного проекту (роботи) Суліма Юлія Євгенівна

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) Розробка гри «Японський кросворд» на програмному рушії Unity

Обсяг розрахунково-пояснювальної записки 68 сторінок

Обсяг графічної (презентаційної) частини 12 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 ігрового продукту на ігровому програмному рушії Unity.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі розглянуті питання проблематики розробки ігор на ігровому програмному рушії Unity, сформовано вимоги до гри згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів розробляємої гри. За допомогою відповідного програмного забезпечення реалізовані всі намічені зміни до ігрового процесу

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

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально описано процес виконання модифікації;
2. Виконано проектування елементів модифікації із поясненнями на схемах;
3. Розроблено функціонуючу гру «Японський кросворд» за допомогою відповідних інструментів розробки.

д) основні недоліки дипломного проекту (роботи)

1. Гра має невелику кількість ігрових рівнів;
2. Можливо не вистачає більш глибокого налаштування гри

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові рецензента Кривченко Юрій Вікторович

Місце роботи і посада рецензента ВСП «Одеський технічний фаховий коледж ОНТУ», голова циклової комісії комп'ютерних технологій та програмної інженерії

Підпис:

«14» серпня 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Щербакова Дмитра Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність: 121 "Інженерія програмного забезпечення"

Освітня програма: «Розробка програмного забезпечення»

Тема дипломного проекту: Розробка гри «Японський кросворд» на програмному рушії Unity

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 69 сторінок. У пояснювальній записці виконано опис предметної області, способи створення гри «Японський кросворд». Проведено проектування та реалізація головного меню та ігрового функціоналу. Графічна частина складається з 12 слайдів мультимедійної презентації, які передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Щербаков Д.С. 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 Щербаков Д.С. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Щербаков Д.С. мав змогу самостійно приймати рішення з реалізації гри «Японський кросворд» та її ігрових елементів, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Суліма Юлія Євгенівна

Місце роботи і посада керівника дипломного проекту

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

Підпис

«10» 04 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016328982

Дата перевірки:
06.06.2024 19:03:28 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 19:12:04 EEST

ID користувача:
100011688

Назва документа: **4РП-07_Щербаков**

Кількість сторінок: **42** Кількість слів: **7531** Кількість символів: **52879** Розмір файлу: **2.44 MB** ID файлу: **1016128417**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.65%
Схожість

Найбільша схожість: **4.14%** з Інтернет-джерелом (https://ela.kpi.ua/handle/123456789/49671)

6.65% Джерела з Інтернету | **135** ..... Сторінка 44

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | **3**

Підозріле форматування | **7 сторінок**

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Щербаков Дмитро Сергійович,
здобувач освіти гр. 4РП-07, та

Суліма Юлія Євгенівна,
керівник дипломного проекту,

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

«Розробка гри «Яконський кросворд» на програмному рушії Unity» (автор роботи – Щербаков Д.С., керівник роботи – Суліма Ю.Є.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 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 _____ / Щербаков Д.С. /

Керівник _____ / Суліма Ю.Є.) /

«10» 06 2024 р.