МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАНН

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Ігровий додаток "Back Memory"»

за напрямом підготовки 6.050101 «Комп'ютерні науки»

Виконавець роботи: студент групи ІТ-52 Онищенко Сергій Вікторович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «___» _____2019 р.

Науковий керівник _____ доцент, Федотова Н. А. _
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___» _____ 2019 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Онищенко Сергій Вікторович

1 Тема роботи *Ігровий додаток "Back Memory"*

керівник роботи *Федотова Наталія Анатоліївна, к.т.н., доцент*

затверджені наказом по університету від « 17 » травня 2019 р. № 084-III

2 Строк подання студентом роботи «10» червня 2019 р.

3 Вхідні дані до роботи Правила гри "BackMemory"

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити Аналіз предметної області, постановка задачі, проектування, реалізація

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Мета та задачі, аналіз предметної області, порівняння аналогів, вимоги до додатку, правила гри, вибір програмного забезпечення, контекстна діаграма процесу розробки, діаграма декомпозиції процесу розробки,д варіантів використання, реалізація, базові механічні дії, модель та анімації гравця, система часток, меню, генерація рівнів, додаткові механіки, демонстрація проекту, висновки.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області	*Федотова Н. А.*		
Постановка задачі	*Федотова Н. А.*		
Проектування	*Федотова Н. А.*		

7. Дата видачі завдання 03.12.2019

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	До 08.03.2019	
2	Оформлення технічного завдання	До 15.03.2019	
3	Проведення аналізу предметної області	До 22.03.2019	
4	Проведення структурно-функціонального моделювання	До 05.04.2019	
5	Реалізація базових механічних дій	До 19.04.2019	
6	Реалізація процедурної генерації	До 03.05.2019	
7	Візуалізація додатку	До 10.05.2019	
8	Тестування додатку	До 17.05.2019	
9	Здача пояснювальної записки та файлів проекту	До 10.06.2019	

Студент _____ Онищенко С. В.
(підпис)

Керівник роботи _____ доцент, Федотова Н. А.
(підпис)

РЕФЕРАТ

Тема роботи: «Ігровий додаток Back Memory».

Пояснювальна записка містить вступ, розділ «Аналіз предметної області», розділ «Постановка задачі», розділ «Проектування», розділ «Реалізація», висновки, список літератури, та додатки. Вона включає 85 сторінок, 6 таблиць, 64 ілюстрації, та 11 джерел.

В розділі «Аналіз предметної області» був виконаний пошук та аналіз аналогів ігрового додатку. Потім були проаналізовані існуючі засобів реалізації.

В розділі «Постановка задачі» було поставлено мету та визнчені задачі дослідження. Після чого був здійснений вибір засобів реалізації продукту.

В розділі «Проектування» були змодельовані такі діаграми як: IDEF0 діаграма реалізації додатку, IDEF3 діаграма реалізації додатку, діаграма варіантів використання, та діаграма Flowchart.

В розділі «Реалізація» описується безпосередньо розробка ігрового додатку Back Memory, яка складається з: реалізації базових механічних дій, реалізації процедурної генерації рівнів, візуалізації, та ін.

Результатом роботи над проектом є готовий ігровий додаток Back Memory.

Ключові слова: ігровий додаток, відеогра, Unity, процедурна генерація, головоломка.

ЗМІСТ

ВСТУП

Сьогодні, відеоігри є одним з найбільш перспективних напрямків на ринку медіа розваг. Вони стали важливою частиною життя багатьох користувачів персональних комп'ютерів і мобільних пристроїв. Завдяки потужному розвитку цієї індустрії, створення відеоігор все частіше сприймається авторами і користувачами як окремий вид художньої творчості.

Як і будь-яка інша культурна індустрія, відеоігри мають величезний ринок, повний видавців та розробників, які створюють високобюджетні продукти, що кожного року продаються мільйонами. Лише за 2017 рік інтерактивні розваги, що охоплюють ігри та більшість медіа навколо них, принесли більше 100 мільярдів доларів.[2]

Зараз можна сказати, що індустрія відеоігор досягла такого рівня індустріалізації, коли відеоігри випускаються масово. Рівня, де фокус-групи є основним джерелом натхнення для багатьох видавців. З року в рік це призводить до неймовірної кількості схожих один на одного продуктів.

Зростання цифрового розповсюдження відеоігор дозволила більш дрібним і незалежним розробникам мати можливість вести розробку без ризику, пов'язаного з витратами на створення фізичних копій гри. Це призвело до створення більш нішевих проектів.

Після міжнародного успіху таких ігор, як: Braid, Castle Crashers та World of Goo, "незалежні" ігри стали новим трендом в індустрії. Невеликі команди, та відсутність творчих обмежень зробили їх відомими як інноваційні, творчі та здатні на експерименти. Розробники, які були обмежені у здатності створювати дорогу та складну графіку, покладатися на інноваційність ігрового процесу.

Звісно, незалежна розробка відеоігор не є чимось новим в галузі, але за останні десять років вона зробила величезний імпульс. Ці проекти, та їх

розробники з кожним роком все більше набирають популярність, в результаті чого даний рух став ключовим в індустрії відеоігор.

Лише за 2017 рік до ігрової аудиторії сервісу Steam приєднались близько 63 млн. користувачів, це приблизно 23% від усієї аудиторії за 14 років. За цей же рік там було випущено близько 7,700 відеоігор (приблизно 21 нових проектів в день), а це 39% від усіх випущених проектів за весь час. Переважна більшість цих проектів були від незалежних розробників. І чим далі, тим більше ці цифри продовжують зростати.[3]

Але в цьому також є і негативні моменти. Продажі в Steam сильно перекошені в сторону топу з усього 100 проектів (що являє собою 0,5% від усіх проектів), на який припадає 50% сумарної виручки.[3] Так склалося через те, що більшість ігор в Steam це проекти інді-розробників з дуже низькою якістю, бо за останні роки їх розробка стала більш простішою та дешевшою. Отже незважаючи на те, що у Steam останній рік був найуспішнішим за весь час його існування, число нових користувачів, що приходять на платформу і купують товари, не може встигнути за кількістю нових проектів.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих аналогів

Cubiscape. Це тривимірна відеогра в жанрі покрокова головоломка, що розроблена Пітером Ковачом та випущена 12 лютого 2017 року на Android. Після пошуку та аналізу аналогів було зроблено висновок, що саме цей проект має більше всього схожих елементів з "Back Memory".[4]

Під час ігрового процесу гравець керує блакитною сферою, та переміщується по ігровому полю, яке складається з кліток. Один хід це перехід на одну клітину. Ціль кожного рівня – пройти від початкової точки до зеленої клітини, при цьому уникаючи пасток. В цілому описані вище механіки є базовими для кожного з представлених в цьому розділі проектів, включаючи “Back Memory”.

На мій погляд можна виділити дві сильні сторони цього проекту. Перша - це використання штучного інтелекту для реалізації противників. В зв’язку з цим, ідея гри полягає в спробах гравця перехитрити штучний інтелект заганяючи ворогів в пастки та уникаючи їх. В моєму проекті було вирішено не використовувати штучний інтелект, бо це може зробити її занадто складною для проходження.

Другою сильною стороною даного проекту є використання процедурної генерації рівнів. Завдяки неї вдалось реалізувати велику кількість рівнів (більше 180). Втім після мого тестування проекту було виявлено, що не всі з них є достатньо збалансовані. Це вказує на проблеми з алгоритмом процедурної генерації який використовується в Cubiscape.

Візуальна складова гри є дуже простою, вона складається з примітивних геометричних фігур (кубів, сфер та ін.) а також має ізометричну «точку зору»

(POV). На мій погляд це є незначним мінусом цього проекту. Слід також зазначити, що Cubiscape розроблено на Unity, як і наступний проект.

Lara Croft Go. Це тривимірна відеогра в жанрі покрокова головоломка, що розроблена компанією Square Enix Monreal та випущена 27 серпня 2015 року на iOS, Android и Windows 10 Mobile. Пізніше гра вийшла в Steam а також для приставок Playstation 4 та Playstation Vita. Гра є частиною франшизи Tomb Raider.[5]

Під час процесу гри гравець керує головною героїнею яка переміщується по ігровому полю. Можна сказати, що базові механіки та управління схожі на Cubiscape, за винятком ігрового поля яке має вертикальні елементи. Гра має близько 150 рівнів, які створені вручну.

Для того, щоб пройти кожен рівень, потрібно перемогти або уникнути ворогів, які стоять на шляху. У Lara Croft GO є три головних ворога: ящірки, змії, та павуки, всі вони мають дуже передбачуваний штучний інтелект. Іноді вони можуть бути використані в інтересах гравця, щоб утримувати перемикачі або відкривати двері. Також гравець може нейтралізувати ворогів, підкрадаючись до них ззаду або кидаючи спис.

Сильною стороною гри є її візуальна складова. Гра має стильну та приємну, хоч і дуже просту, низькополігональну тривимірну графіку, а також непогану анімацію. Це сильно виділяє її на фоні конкурентів. Проект розроблено на ігровому рушії Unity.

Nova-111. Це научно-фантастична двовимірна гра в жанрі покрокова головоломка у реальному часі, яка розроблена студією Funktronic Labs, та випущена 25 серпня 2015 року на ПК. Гравець бере на себе управління космічним кораблем, та подорожує по планетам.[6] Базовий процес гри, та управління схожі на попередні аналоги. Але на мій погляд можна виділити такі сильні сторони:

- інноваційний ігровий процес – гра є цікавим поєднанням одразу двох на перший погляд взаємовиключних жанрів, а саме покроковий режим та режим реального часу;

- приємна візуальна складова – кожна планета має свій, ні на що не схожий, дизайн;
- дуже складні головоломки та стратегічні бої;
- динамічний звуковий супровід – музика змінюється залежно від стилю гри;
- прокачка корабля головного героя.

Vandals. Це тривимірна відеогра в жанрі покрокова стелс головоломка, що розроблена компанією ARTE Experience, та випущена в 12 серпня 2018 на ПК, Android та IOS. Гравець управляє художником, який малює графіті. На кожному рівні йому потрібно дістатися до певного місця, намалювати графіті, після чого втекти, не потрапивши до рук поліції.[7]

Базовий ігровий процес та управління дуже схожі на всі попередні аналоги, а більше всього на Lara Croft Go, за винятком механіки хованок. На мій погляд найсильнішою стороною гри є візуальна складова. Головний мінус даного проекту це кількість контенту. Гра має всього 60 рівнів, і проходиться дуже швидко, це значно менше ніж у інших аналогів. Продукт розроблена на ігровому рушії Unity.

Таблиця 1.1 – Порівняння аналогів

	Cubiscape	Lara Croft Go	Nova-111	Vandals	Back Memoty
Тривимірна графіка	+	+	–	+	+
Використання процедурної генерації	+	–	+	–	+
Штучний інтелект	+	–	+	–	–

Таблиця 1.1 – Продовження таблиці

Таблиця 1.1 – Продовження

	Cubiscape	Lara Croft Go	Nova-111	Vandals	Back Memoty
Мобільна версія	+	+	–	+	+
Більше 100 рівнів	+	+	+	–	+
Підтримка контроллеру	–	+	+	–	–

1.2 Вибір програмного забезпечення

Ігровий рушій − це середовище розробки програмного забезпечення, яке призначене для створення відеоігор. Основні функції, які виконує ігровий рушій: рендеринг 2D або 3D графіки, фізичний рушій та виявлення зіткненнь, реалізація скриптів, звук, анімація, управління пам'яттю, реалізація штучного інтелекту та ін. Ігровий рушій є ключовим компонентом процесу розробки гри. Його вибір може мати вирішальне значення.

Сьогодні ігрові рушії прискорюють процес розробки ігор за допомогою існуючих шаблонів та ассетів, які можна використовувати повторно. Тим самим мінімізуючи, а в деяких випадках навіть повністю виключаючи необхідність глибоких вмінь програмування. Більш того, сучасні ігрові рушії дають можливість розробникам створити гру один раз, після чого експортувати її на різні платформи, включаючи мобільні пристрої, з мінімальними змінами оригінальної версії.

Між ігровими рушіями існує певна конкуренція. Такі великі компанії як Rockstar games, Ubisoft, або CD Projekt RED, використовують рушії власної

розробки. Однак невеликі студії зазвичай не мають можливостей та ресурсів для їх створення, тому вони використовують вже існуючі програмі рішення.

Зараз на ринку існує багато ігрових рушіїв які мають як загальні, так і відмінні риси. Можна сказати, що різні рушії мають різну філософію розробки ігор, та націлені на різні потреби. Деякі з них можуть не вимагати від розробників знань програмування, інші засновані на популярних веб-технологіях, є рушії з відкритим кодом, що можуть бути налаштованими або навіть розширеними досвідченими користувачами і т.д.

Сьогодні, серед великої кількості ігрових рушіїв, обрати оптимальний може бути не просто. Мною був проведений аналіз найвідоміших з них, для використання в моєму проекті.

Unity − багатоплатформовий ігровий рушій, який розроблений компанією Unity Technologies. Перша версія продукту була представлена та випущена на конференції Apple в 2005 році, та призначалась лише для розробки на OS X.

Сьогодні це один з найпопулярніших ігрових рушіїв, на ньому можна розробляти проекти на: Windows, OS X, Android, Windows Phone, Apple iOS, Linux, Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One а також VR. Наприклад 50% всіх мобільних ігор створюються на Unity. З моменту запуску було випущено декілька основних версій Unity, на сьогодні остання з них — Unity 2018.3.11.[8]

На мій погляд така велика популярність цього продукту зумовлена його простотою в використанні. Розробка ігор на Unity є дуже швидкою, особливо для мобільних платформ. Самі проекти та їх збірки займають небагато пам’яті, а процес їх експорту є дуже простим. Редактор Unity має Drag & Drop інтерфейс який, на мій погляд, є простим та зручним. Продукт підтримує дві мови для написання скриптів: C# та JavaScript.

Для мене, найсильніша сторона Unity це його велика спільнота. Існує безліч форумів, груп, Youtube каналів, що присвячені розробці ігор на Unity (треба зазначити, що деякі з них спонсорує сама компанія). Це дає можливість

легко та швидко знаходити відповіді на будь-які запитання по користуванню продуктом, що значно полегшує розробку проектів.

Ще одним плюсом спільноти Unity є магазин ассетів, що знаходиться в самій програмі. В ньому знаходиться більше тисячі ассетів, як платних, так і безкоштовних, які можна завантажити в свій проект за секунди. Це допомагає розробникам значно зекономити час під час розробки своїх проектів.

Основним мінусом Unity я б виділив реалізацію графіки, яка поступається конкурентам, таким як Unreal та Cry Engine. Але в моєму проекті не використовується складна графіка, що робить цей мінус не дуже значним.

Unity має декілька версій, на мій погляд для мого проекту оптимальною є версія «Personal», яка є безкоштовною за умовою, що бюджет проекту не перевищує 100 000$ на рік. Інші версії: «Plus» та «Pro» коштують 25$ та 125$ на місяць відповідно.[8]

Unreal Engine − багатоплатформовий ігровий рушій, який розроблений компанією Epic Games. Перша версія була реалізована в 1998 році в грі Unreal. Остання версія - Unreal Engine 4, була випущена в 2014 році.[9] На сьогоднішній час це провідний рушій для складної візуалізації, реалізації рослинності та створення складних ландшафтів.

Unreal підтримує розробку ігор на ті ж платформи, що і Unity, але зазвичай його використовують для більш крупних проектів. На мій погляд він погано підходить для розробки невеликих або мобільних ігор, хоча й підтримує iOS і Android.

Для мене сильною стороною Unreal Engine є система Blueprint — це система візуального програмування. Blueprints являє собою графи з блоків, які з’єднані разом. Ці з'єднання створюють певну логіку замість скриптів. Це допомагає розробникам реалізовувати код набагато швидше та наглядніше. Крім того треба зазначити що подібна система скоро з’явиться і в Unity. Втім класичне написання скриптів тут також є.

Unreal має відкритий вихідний код на C++. Мова C++ дає розробникам ігор значний контроль над всією системою, яка є досить великою. Однак на

мій погляд ця структура складна і важка для вивчення, що звісно не відміняє її потенціалу. В своєму проекті я б її не став використовувати.

Технологія рендерінгу в Unreal Engine є його значною перевагою, пост-процес ефекти є дуже швидкими та підтримують багато функцій. Крім того даний рушій має редактор для створення користувачами власних матеріалів, а також відмінні інструменти для оптимізації ігор.

Спільнота Unreal Engine 4 є не меншою ніж в Unity, але документація Unreal на мій погляд є слабшою, тому новим користувачам тут буде складніше. Сам рушій є безкоштовним, але якщо доходи гри перевищують 3000$ за квартал, 5% віддається компанії Epic Games.[9]

CryEngine − багатоплатформовий потужний ігровий рушій, який розроблений компанією Crytek. Перша версія була реалізована в 2002 році в грі Far Cry. Остання − CryEngine 5.5 випущена 20 вересня 2018 року.[10] Рушій підтримує мови програмування C++ та Lua.

Суттєвим мінусом даного рушія є те, що він не призначений для розробки мобільних ігор, а лише для Windows, Linux, PlayStation 3, PlayStation 4, Wii U, Xbox 360, та Xbox One та VR. Натомість можна виділити декілька сильних сторін даного продукту:

- реалістичне освітлення та високі можливості рендерингу;
- вдосконалена система анімацій;
- реалізація густої рослинності.

Слід зазначити, що конкретно для мого проекту дані плюси CryEngine не є суттєвими. Також, на мій погляд, даний рушій має не такий зручний та зрозумілий інтерфейс як у конкурентів. Сам рушій є безкоштовним, але якщо доходи гри перевищують $5000 за квартал, 5% віддається компанії.[10]

GameMaker Studio − багатоплатформовий ігровий рушій, який розроблений компанією YoYo Games. Перша версія була випущена 15 листопада 1999 року. Остання – версія 2.2.2 випущена 26 березня 2019 року.[11] З самого початку рушій розроблявся для того, щоб дозволити розробникам початківцям мати можливість створювати ігри без серйозних

навичок програмування. Але останні версії орієнтовані і на більш просунутих розробників.

GameMaker Studio дає можливість розробляти ігри на: Windows, macOS, Ubuntu, HTML5, Android, iOS, Amazon Fire TV, Android TV, Microsoft UWP, PlayStation 4, and Xbox One. І хоча рушій дає можливість створення 3D ігор, все таки він в першу чергу призначений для 2D ігор, та дозволяє використовувати як растрову так і векторну графіку.

Як і Unreal Engine рушій має інструмент візуального програмування − Drag and Drop. GameMaker підтримує власну мову програмування GameMaker Language, яка схожа на JavaScript.

GameMaker Studio має декілька версій: Trial, Creator, Developer, та Console, всі вони мають різну ціну та функціонал. На мій погляд оптимальною є версія Creator за 39$.[11]

2. ПОСТАНОВКА ЗАДАЧІ

2.1 Мета та задачі дослідження

Відеогра Back Memory представляє собою покрокову 3D головоломку, основний упор якої робиться на покращення пам'яті гравця. Рівні гри створюються за допомогою процедурної генерації контенту.

Мета: Розробити 3D гру Back Memory на Unity в жанрі головоломка з використанням випадкової генерації рівнів. Для досягнення поставленої мети потрібно вирішити наступні задачі:

1) Аналіз існуючих аналогів та вибір способу реалізації;
2) Розробка структури додатка;
3) Розробка ігрового додатку;
4) Тестування проекту.

2.2 Вибір засобів реалізації

Ігровий рушій Unity. Після аналізу ігрових рушіїв, було вирішено розробляти мій дипломний проект на Unity. Далі буда більш детальний аналіз його архітектури. (Рисунок 2.1)

Відеоігри, які створюються на Unity, підтримують OpenGL та DirectX. Також, даний рушій використовує Drag&Drop інтерфейс, який є простим, складається з різноманітних вікон, що дає можливість робити налагодження гри в самому редакторі.

Кожен проект Unity повинен мати хоча б одну сцену – це окремий файл, який містить в собі ігрові об’єкти, скрипти, та налаштування. В свою чергу об’єкти можуть містити в собі різні набори компонентів, з якими взаємодіють

скрипти. Крім того, у кожного об’єкти є назва, шар, на якому він відображається, і тег. Слід зазначити, що рушій допускає наявність декількох об’єктів з однаковими назвами.

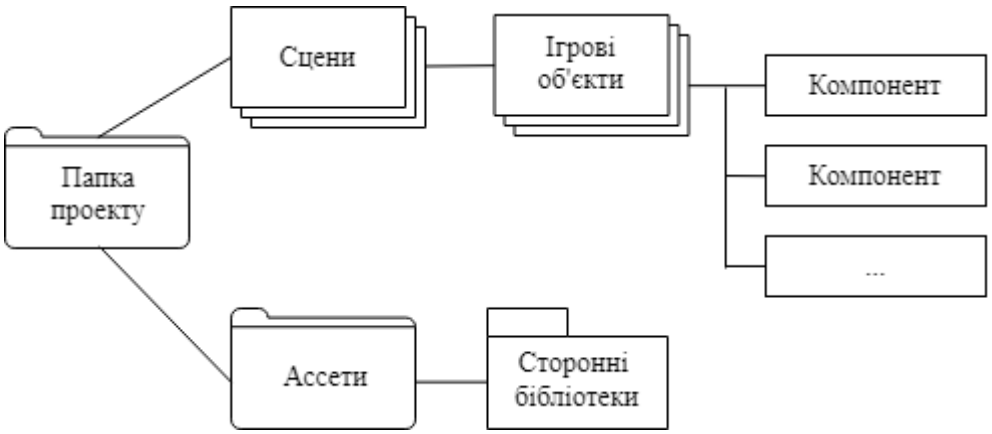


Рисунок 2.1 − Структура проекту на Unity

Важливо, що у кожного об’єкта повинен бути компонент Transform, яких зберігає інформацію про координати положення його на сцені, поворот, та розмір об’єкту по трьом осям. Ще один важливий компонент Unity − Mesh Renderer, який робить видимою модель об’єкта.

Редактор Unity підтримує написання та редагування шейдерів, а також має компонент для створення анімації. Слід зазначити, анімацію також можна попередньо створити в будь-якому 3D-редакторі, та імпортувати її разом з моделлю. Крім того, моделі, скрипти, матеріали, та звуки можуть бути запаковані в формат unityassets, після чого їх можна легко передати. Саме цей формат використовує Unity Asset Store, в який розробники викладають свої ассети, та заробляють на цьому.

Рушій Unity підтримує мови C# JavaScript для написання скриптів, а також декілька шейдерних мов зі спеціальною надбудовою Shader Lab. У Back Memory я збираюсь використовувати мову C#, про яку далі.

Мова програмування C# − об’єктно орієнтована мова програмування для загального призначення, що розроблена компанією Microsoft спільно з платформою .NET. Вона відноситься до мов з C-подібним синтаксисом, і

найбільш всього схожа на C++ та Java. За допомогою C# розробляється різноманітне програмне забезпечення: веб-сайти, веб-додатки, офісні додатки, десктопні та мобільні додатки, ігри та ін. Також слід зазначити, що ігровий рушій Unity підтримує дану мову програмування.

Можна виділити такі переваги даної мови:

- компонентно-орієнтований підхід програмування, який сприяє гнучкості, та повторному використанню деяких фрагментів програм
- мова претендує на справжню об'єктну орієнтованість, тобто будь-яка сутність претендує на те, щоб бути об'єктом;
- система типізації є уніфікованою;
- мова більше орієнтована на безпеку коду, в порівнянні з C++;
- розширене подійно-орієнтоване програмування.

Також можна виділити декілька мінусів C#:

- продуктивність є відносно невисокою;
- небагато концептуальних свіжих ідей;
- складний синтаксис.

Microsoft Visual Studio – це інтегроване середовище розробки (IDE) створене компанією Microsoft. Цей продукт використовує такі платформи для розробки програмного забезпечення, як Windows API, Windows Presentation Foundation, Windows Forms, а також Microsoft Silverlight.

Visual Studio Tools це безкоштовне розширення для Visual Studio. Воно перетворює продукт Microsoft на дуже потужний інструмент для багатоплатформенної розробки на Unity, оскільки в самому рушії немає редактору коду. Дане розширення використовує функції редагування коду, відладки, та продуктивності Microsoft Visual Studio для створення скриптів на мові C#, а також проведення їх налагодження використовуючи потужні можливості Visual Studio.

Крім того Visual Studio Tools має дуже глибоку інтеграцію з самим редактором Unity, тому розробник витрачає менше кліків для виповнення

простих задач. Це забезпечує підвищення продуктивності під час розробки гри. Ще можна виділити такі сильні сторони продукту:

- використання технології автозаповнення IntelliSense, яка пише назву функції під час введення її початкових літер;
- потужні можливості рефакторингу (зміна внутрішньої структури коду для полегшення його розуміння, але при цьому не змінюючи зовнішню поведінку самої системи);
- швидке налагодження для Unity;
- широка варіація налаштуваннь робочої середи.

Autodesk 3ds Max – це програмне забезпечення для 3D-моделювання, рендерингу та створення анімацій, що розроблене компанією Autodesk. Продукт широко використовується для розробки відеоігор, рекламних роликів, мультфільмів а також для створення спецефектів у фільмах.

Даний продукт надає багато засобів для створення різних за складністю та формою тривимірних моделей об’єктів, з використанням таких механізмів та технік як: полігональне моделювання, моделювання на основі раціональних неоднорідних B-сплайнів, моделювання з використанням спеціальних вбудованих бібліотек параметричних об’єктів, та моделювання за допомогою поверхонь Безьє.

Окрім інструментів моделювання та створення анімацій в 3ds Max також має шейдери, системи часток, динамічне моделювання, дифузне відбиття, глобальне освітлення, широкі можливості налаштування свого інтерфейсу, а також власну мову для написання скриптів. Після розробки моделей та анімацій в 3ds Max їх необхідно експортувати в Unity, зробити це можна завдяки формату .FBX або іншим універсальним форматам.

3. ПРОЕКТУВАННЯ

3.1 Структурно-функціональне моделювання реалізації ігрового додатку “Back Memory”

Функціональна модель IDEF0 відображає функції та структуру системи, а також потоки матеріальних об'єктів та інформації, які перетворюються за допомогою цих функцій. Дана методологія означає створення системи ієрархічних діаграм, тобто одиничних описань її фрагментів. Спочатку створюється контекстна діаграма, тобто повний опис системи та взаємодії її з навколишнім середовищем. Далі проводиться декомпозиція системи, тобто розбиття її на окремі підсистеми та їх окремий опис. Після чого ці системи розбиваються на більш дрібніші, до досягнення потрібного рівня деталізації. [12]

Кожна IDEF0 діаграма повинна складатися з блоків робіт та стрілок. Роботи це функції або процеси, які мають чіткі результати та відбуваються протягом певного часу. Стрілки описують взаємодію робіт між собою або з зовнішнім світом.

Для створення даної моделі використовувався програмний продукт Erwin Process Modeler, який розроблений компанією Computer Associates. На рисунку 3.1 зображена контекстна діаграма реалізації ігрового додатку Back Memory.

- Ціль – розробити ігровий додаток.
- Точка зору – розробник додатку.
- Предметна область – ігрові додатки.

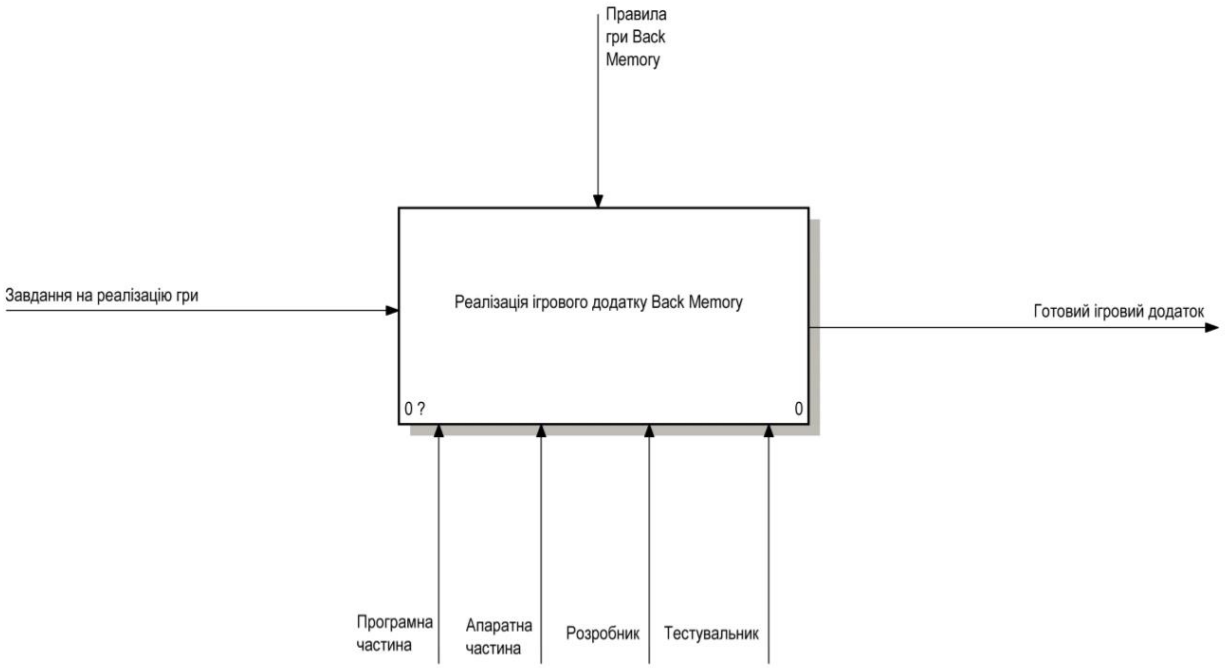


Рисунок 3.1 – Контекстна діаграма IDEF0

На рисунку 3.2 зображена декомпозиція діаграми реалізації ігрового додатку Back Memory. Інші декомпозиції наведені в Додатку В.

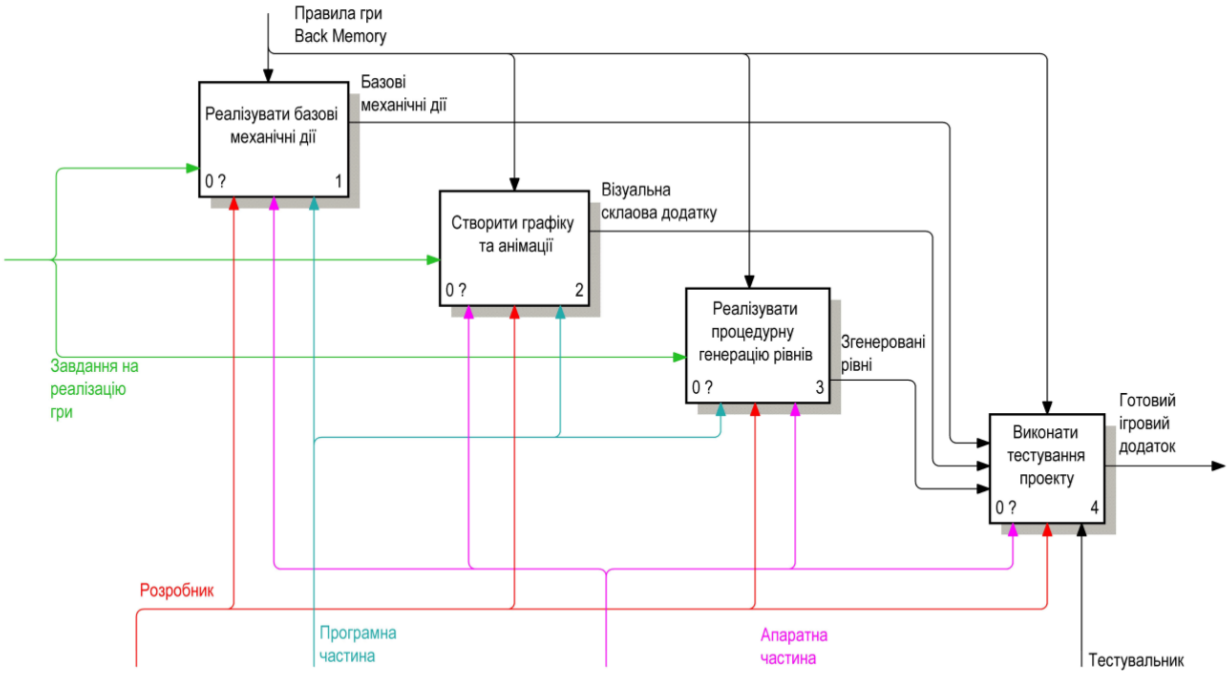


Рисунок 3.2 – Діаграма декомпозиції IDEF0

3.2 Моделювання IDEF3 діаграми реалізації ігрового додатку “Back Memory”

IDEF3 – це методологія моделювання, яка графічно описує інформаційні потоки взаємин між процесами, що обробляють інформацію, та об’єктами, які є їх частиною. Дана методологія дає можливість описати ситуацію, коли процеси виконуються послідовно, та описати об’єкти, що беруть участь в одному процесі. Кожна IDEF3 діаграма містить: роботи, зв’язки, об’єкти посилань та перехрестя.[13]

Роботи відображаються у вигляді прямокутних блоків, кожен з яких має власний ідентифікаційний номер. Вони можуть бути декомпозовані для більш детального аналізу. Зв’язки показують відносини між роботами, всі вони є односпрямованими та можуть бути спрямованими куди завгодно. IDEF3 може бути три види зв'язків: тимчасове передування, нечітке ставлення, та об'єктний потік. Перехрестя використовуються для того, щоб відобразити логіку руху потоків між роботами, вони дозволяють вказати події, що повинні або можуть відбутись для виконання наступної роботи.

IDEF3 діаграму процесу розробки ігрового додатку Back Memory та її декомпозицію, можна побачити на рисунках 3.3 – 3.4.

Рисунок 3.3 – Контекстна діаграма IDEF3

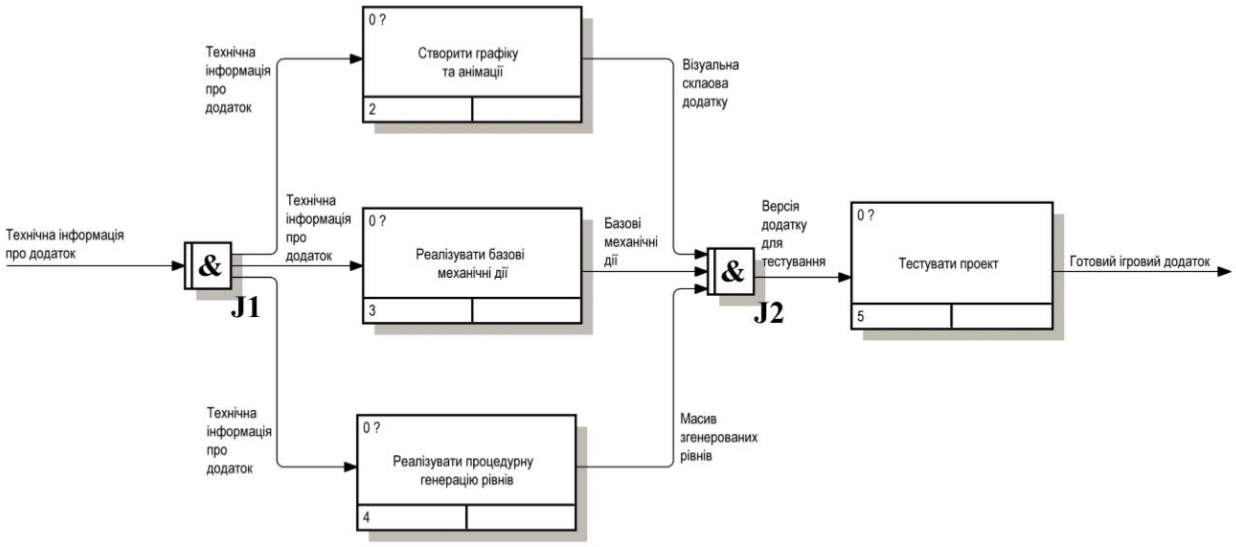


Рисунок 3.4 – Діаграма декомпозиції IDEF3

3.3 Моделювання діаграми варіантів використання

Діаграма варіантів використання це вихідне концептуальну уявлення системи в процесі її розробки. Вона складається з варіантів використання, акторів, та відносин між ними. Варіанти використання це описання послідовності дій які робить система у відповідь на діяльність користувачів, або інших програмних систем. Вони відображають функціональність системи. [14]

Актори

Гравець – людина, яка грає в гру;

Варіанти використання

1) ВВ Розпочати гру – ВВ запускає останній рівень до якого дійшов гравець.
2) ВВ обрати рівень – ВВ дозволяє гравцю обрати будь-який рівень з відкритих ним.

3) ВВ Згенерувати рівень – ВВ дозволяє гравцю згенерувати новий рівень, обравши попередньо його складність.
4) ВВ Закрити додаток – ВВ дозволяє гравцю, вийти з ігрового додатку.
5) ВВ Змінити налаштування – ВВ дозволяє гравцю змінювати налаштування додатку.

Діаграму варіантів використання ігрового додатку Back Memory можна побачити на рисунку 3.5.

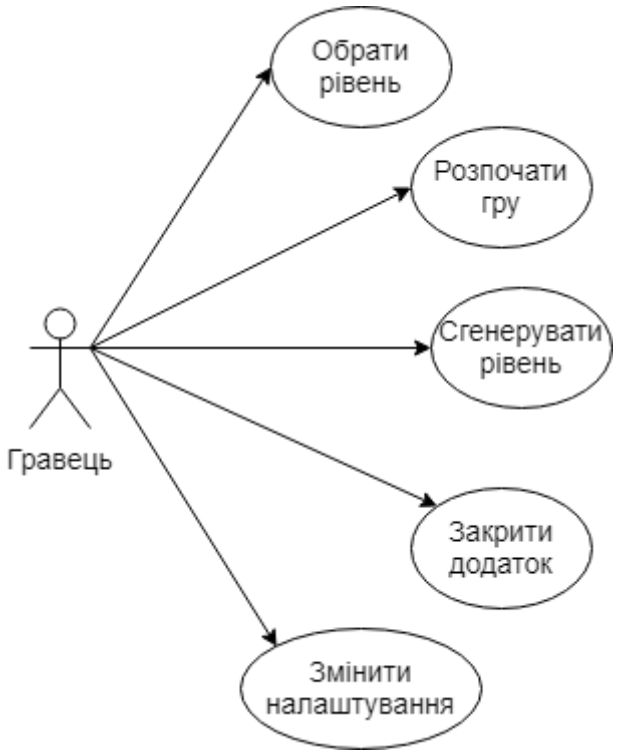


Рисунок 3.5 – Діаграма варіантів використання

3.4 Моделювання Flowchart діаграми

Діаграма Flowchart відображає систему, процес, або комп’ютерний алгоритм. Вона складається з різних функціональних блоків, таких як: початок, кінець, введення, виведення, виклик функції та інші.

Дана діаграма зазвичай використовується при документуванні та проектуванні простих програм або процесів. Вона дає змогу візуалізувати процеси, що краще допомагає їх зрозуміти, та інколи знайти їх неочевидні особливості, або слабкі місця. [15]

Діаграму Flowchart ігрового додатку Back Memory можна побачити на рисунках 3.14. – 3.16

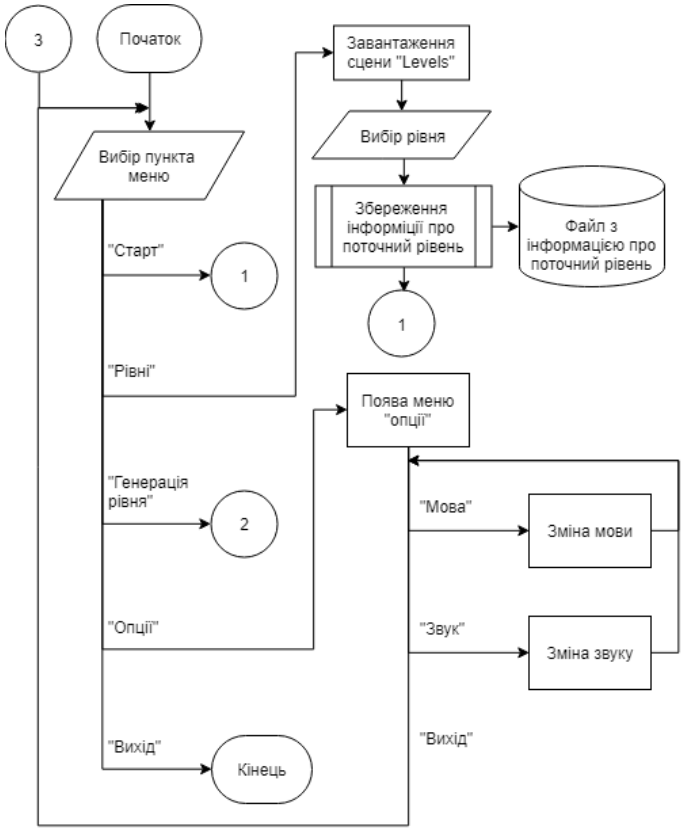


Рисунок 3.6 – Блок-схема ігрового додатку “Back Memory”

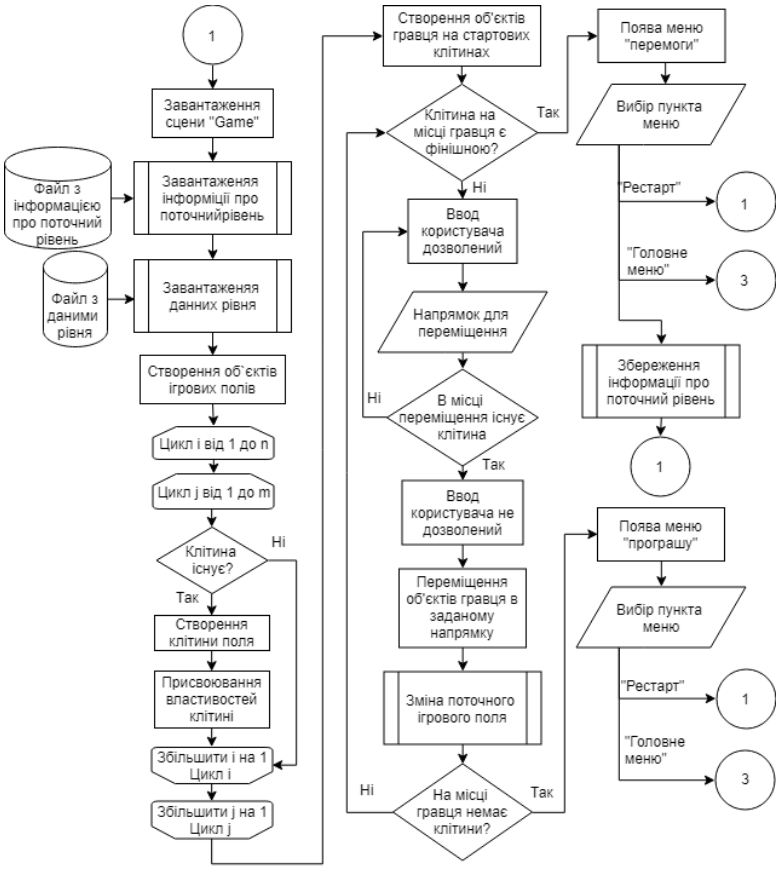


Рисунок 3.7 – Блок-схема ігрового додатку “Back Memory”

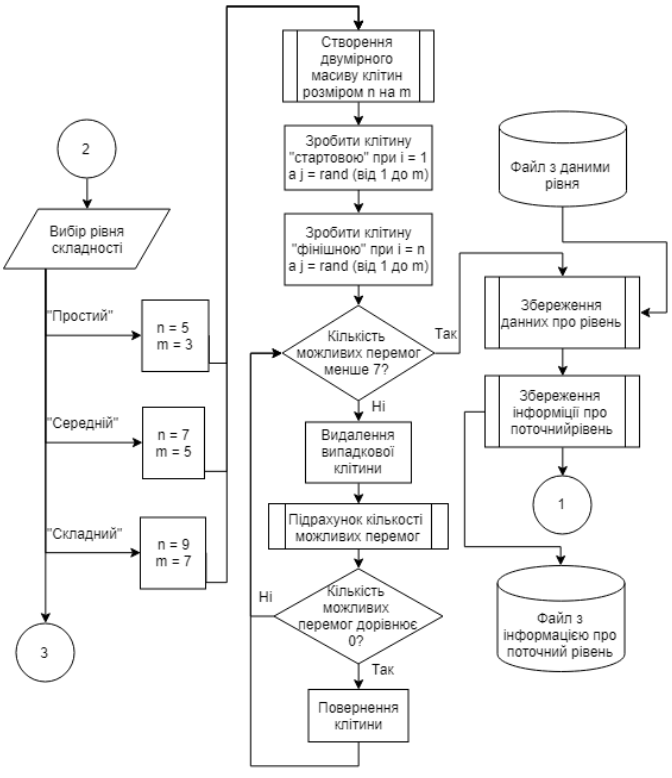


Рисунок 3.8 – Блок-схема ігрового додатку “Back Memory”

4. РЕАЛІЗАЦІЯ

4.1 Реалізація базових механічних дій

4.1.1 Ігрове поле

Компонент Node

Даний компонент описує одну клітину ігрового поля. Він присвоюється ігровому об’єкту Node. Кожен такий ігровий об’єкт може належати лише до одного стану ігрового поля, тому він має посилання на ігровий об’єкт Board.

Крім того цей компонент має змінну типу Vector2, яка містить в собі координати об’єкта Node. Змінні типу Vector2 потрібні для представлення 2D векторів та точок.

Компонент Board

Даний компонент описує ігрове поле. Він присвоюється пустому ігровому об’єкту Board. Всього на сцені існує 3 такі ігрові об’єкти кожен з яких репрезентує один з трьох станів поля.

Компонент Board має в собі список всіх ігрових об’єктів Node на сцені, які належать до того самого стану, що й його ігровий об’єкт. Для пошуку та запису об’єктів Node в список, в скрипті Board існує метод GetNodeList.

Спочатку він, використовуючи стандартний метод Unity – FindObjectsOfType, шукає усі ігрові об’єкти типу Node. Однак всі ці об'єкти належать до різних станів ігрового поля, тому далі, за допомогою циклу, вони перебираються та записуються у список.

Ще одним важливим методом компоненту Board є метод FindNodeAt. Він виконує функцію пошуку ігрового об’єкту типу Node за його координатами. Даний метод приймає змінну типу Vector3, тобто потенціальні координати клітини, та повертає ігровий об’єкт типу Node. Vector3 це

стандартна структура в Unity, що використовується для передачі 3D-позицій та напрямків.

4.1.2 Переміщення героя

Компонент PlayerInput

Даний скрипт, як зрозуміло з назви, відповідає за ввід користувача. Після його зчитування PlayerInput викликає одну з чотирьох функцій кожна з яких означає один з напрямків переміщення. Важливим елементом є змінна InputEnabled, яка відповідає за дозвіл або заборону вводу зі сторони гравця.

Очевидно, що якщо даний додаток розробляється як для ОС Windows так і для Android, типи вводу для кожної з платформ будуть різними. Об’єднує їх лише те, що зчитування вводу відбувається в функції Update(). Це стандартна функція в Unity, яка автоматично викликається заново кожен кадр.

Ввід на ПК відбувається за допомогою клавіатури. За це відповідає функція GetAxisRaw стандартного класу Input. Вона повертає значення вертикальної (кнопки “W” та “S”) або горизонтальної (кнопки “A” та “D”) осі в діапазоні від -1 до 1.

Ввід на Android виконується за допомогою свайпів по діагоналям. За це відповідає функція SwipeCheck та дві змінні типу Vector2 – startTouchPosition та endTouchPosition. Отже startTouchPosition це координати початку свайпа, а endTouchPosition – координати його кінця. Після зчитування йде порівняння їх значень x та y, та визначення напрямку переміщення.

В ігровій сцені Game, даний компонент присвоєний пустому ігровому об’єкту Level.

Компонент PlayerMover

Даний скрипт відповідає за переміщення та падіння ігрового об’єкту героя. Дані процеси відбуваються за допомогою Coroutines та iTween. Переміщення героя відбувається запуском функції Move, в яку передається

змінна типу Vector3 яка містить в собі координати місця призначення, та змінна типу float, що визначає час затримки перед початком переміщення. Компонент PlayerMover присвоєний ігровому об’єкту героя (Player).

Coroutine

Метод, який дозволяє зробити паузу і почекати до деякого стану системи, перш ніж продовжити. Він являє собою простий C# ітератор, який повертає IEnumerator, та використовують ключове слово yield. В рушії корутини виконуються та реєструються до першого yield за допомогою методу StartCoroutine.[17]

ITween

Це потужна та проста анімаційна система для Unity. В своїй суті вона являє собою систему інтерполяції. Вона приймає одне значення, та протягом певного часу анімує його в інше. iTween дозволяє переміщувати, маштабувати, обертати, трясти, штовхати об’єкти, а також управляти звуком та камерами. Також вона дає можливість повністю контролювати такі речі в анімації як: цикл, затримка, функції зворотного виклику та багато іншого.[16]

ITween є лише одним файлом C#, що можна використовувати усіма версіями Unity, а також в будь-якій мові програмування, яку підтримує Unity. Оскільки це статичний клас, то він не потребує створення його екземпляру, а дозволяє одразу викликати властивості та запускати методи.

Слід зазначити, що iTween не є стандартним ассетом в Unity. Він розроблений компанією Pixelplacement, та є безкоштовним для завантаження в офіційному магазині ассетів Unity.

Функції плавності

Більшість речей в нашому житті ніколи не рухаються з постійною швидкістю, та ніколи не починають свій рух миттєво. Наприклад при падінні м’яч постійно прискорюється, а після приземлення - трішки підстрибує. Для анімації такі речі є дужа важливими.

Функції плавності (easing) визначають швидкість течії анімації, що робить її більш реалістичною. ITween використовує рівняння плавності Роберта Пеннера, що мають відкритий вихідний код.[19]

Переміщення та падіння гравця реалізується функцією iTween - MoveTo. Переміщення обчислюється формулою easeInOutExpo, що зображена на рисунку 4.1, а падіння – формулами easeOutSine та easeInCubic, що на рисунку 4.2.

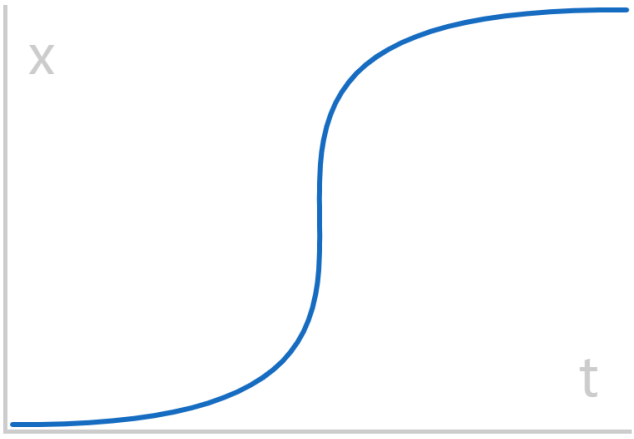


Рисунок 4.1 - Функція easeInOutExpo

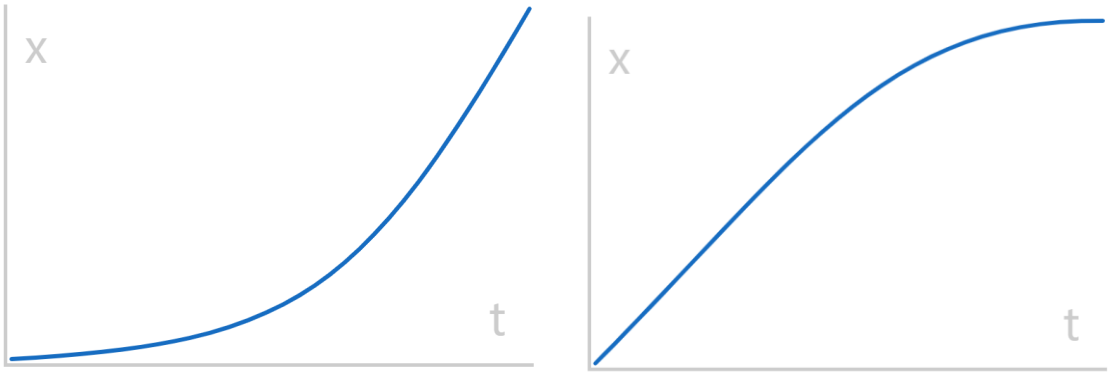


Рисунок 4.2 – Функції easeOutSine та easeInCubic

В правилах гри сказано, що ігрове поле має три стани, які змінюються по черзі. Дана механіка було реалізована за допомогою зміни позиції камери.

Камера

В Unity камера це ігровий об'єкт, який захоплює та відображає ігрову сцену гравцеві. Кількість камер на одній сцені може бути необмежена. Вони можуть бути налаштовані на рендеринг в будь-якому місці та в будь-якому порядку.

В ігрових сценах Back Memory існує всього одна камера. Вона має ізометричну точку зору та орфографічну перспективу.

На рисунку 4.3 можна побачити як повністю виглядає ігрова сцена проекту. На ній одразу знаходяться 3 ігрові поля, кожен з яких відображає окремий стан, а також три об’єкти героїв. Кожне поле знаходиться на сталій відстані від сусіднього. Червоним прямокутником зображено область яку захоплює камера.

Після кожного кроку гравця, камера змінює своє положення та направляється на інше ігрове поле. Оскільки відстань між сусідніми полями є сталою, а об’єкти героїв залишаються на тому самому місці – гравцю здається, що змінюється не камера а самі ігрові поля. За ці дії відповідає компонент TimeManager.

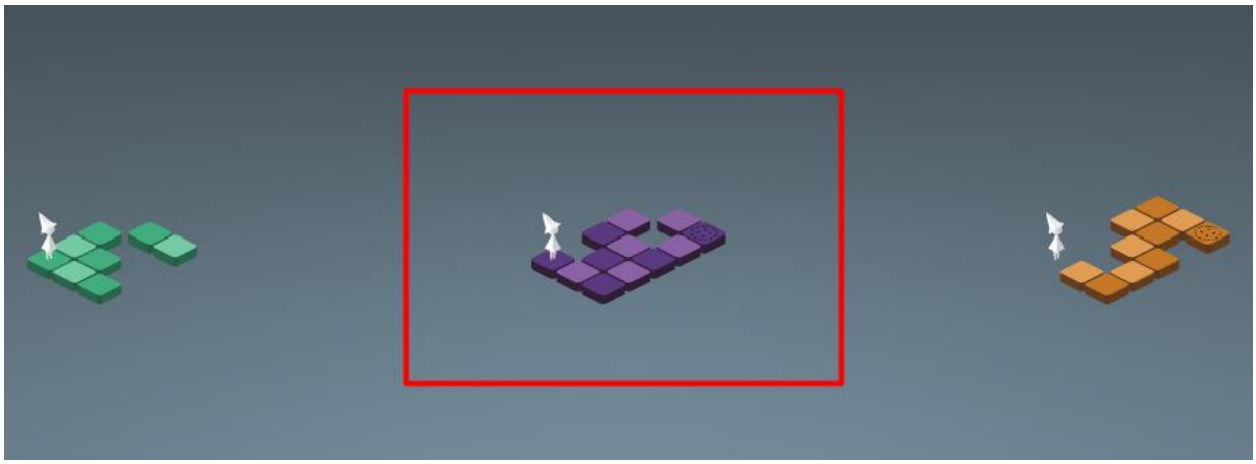


Рисунок 4.3 – Сцена Game

Компонент TimeManager

В ігровій сцені, даний компонент присвоєний пустому ігровому об’єкту Level. Він має змінну типу INT яка відповідає за те, який стан має ігрове поле в даний час (1, 2, або 3). Також даний компонент має посилання на ігровий об’єкт камери, та на три ігрові об’єкти гравця. За зміну положення камери відповідає метод ChangingCamera.

В функції Start значення стану поля одразу дорівнює 2. Метод Start є стандартним методом Unity, який викликається на початку роботи компонента лише один раз.

Оскільки об’єкт гравця переміщуються по ігровому полю, то він повинен рухатись тільки по його клітинам. Отже повинен бути метод який перевіряє чи є клітина на місці, куди хоче здійснити хід гравець. Цей метод називається MoverTest.

Він викликається з компоненту PlayerInput, всередині функцій, що визначають напрямки переміщення. MoverTest приймає одну змінну типу Vector3, яка може дорівнювати: (2, 0, 0), (-2, 0, 0), (0, 0, 2), або (0, 0, -2), залежно від напрямку який обрав гравець. Далі в MoverTest, з компонента Board, викликається метод FindNodeAt який повертає ігровий об’єкт типу Node (якщо він існує), що знаходиться на місці координат. Змінна яка передається в FindNodeAt являє собою потенційні координати шуканої клітини поля, і дорівнює сумі координати положення об’єкту героя та вектору напрямку.

Дані дії здійснюються лише для одного (з трьох) об’єкта Board та для одного об’єкта Player, залежно від того який в даний момент стан поля. Отже якщо шуканий об’єкт клітини поля існує, то здійснюється переміщення всіх трьох об’єктів гравця, за допомогою методу MoveAllPlayers. Крім того, під час переміщення гравець не може здійснювати ніякий ввід, доки крок не закінчиться.

4.1.3 Перемога та програш

Перемога

Для перемоги гравцю об’єкту гравця потрібно дійти до кінця рівня. Для позначення кінця рівня існує змінна endNode, яка знаходиться в компоненті Node. Отже якщо в ігровому об’єкті типу Node, значення змінної endNode є істиною, то це значить що дана клітинна є фінішною. На кожному з трьох ігрових об’єктів полів повинна бути максимум одна така клітина.

Для візуального відображення фінішної клітини використовується ігровий об’єкт EndNode, що представляє собою 2D-спрайт, який знаходиться над об’єктом фінальної клітини. Це можна побачити на рисунку 4.4.

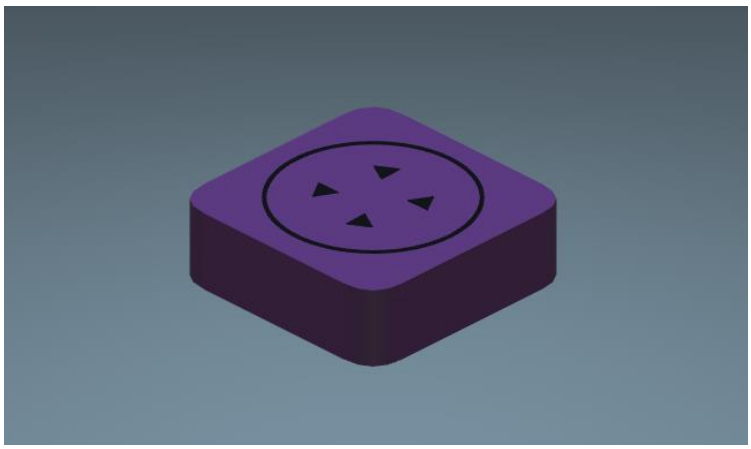


Рисунок 4.4 – Фінальна клітина

Крім того ігровий об’єкт EndNode має однойменний компонент, який відповідає за обертання спрайту навколо своєї осі. Це відбувається за допомогою вже описаного мною iTween. В методі Start викликається функція RotateBy, в якій обрано градус та швидкість обертання, а також встановлено зацикленість анімації.

Даний ефект потрібен для того, щоб гра простіше читалась, адже люди краще звертають увагу на рухомі об’єкти.

Сама механіка перемоги працює таким чином: після кожного кроку гравця потрібно перевіряти чи стоїть він на фінальній клітині. Для цього в

компоненті TimeManager існує метод WinTest, функціонування якого дуже схоже на описаний вище MoverTest.

Даний метод викликається одразу після кожного кроку гравця. Він викликає метод FindNodeAt, і передає в нього координати положення гравця в форматі Vector3. FindNodeAt повертає ігровий об’єкт клітини, на якій стоїть гравець, після чого йде перевірка чи є клітина фінальною. Якщо це так, то відкривається «Меню перемоги».

Програш

За правилами гри, програш відбувається коли: після кроку гравця, поле змінюється, а під об’єктом героя немає клітини. Після цього відбувається падіння гравця, та перезавантаження рівня. Для цього в компоненті TimeManager існує метод PlayerStandingTest.

Він працює аналогічно до методу WinTest: викликає метод FindNodeAt та перевіряє чи повертає цей метод об’єкт клітини. Якщо ні – значить під об’єктом гравця немає клітини поля, тому треба запустити корутину падіння об’єкту гравця та перезапустити рівень. Гравець не може здійснювати ніякий ввід, доки рівень не перезапуститься.

4.2 Генерація рівнів

4.3.1 Основи генерації рівнів

Процедурна генерація – це створення ігрового (або іншого медіа) контенту за допомогою алгоритмів, замість того, щоб робити це вручну. Існує дві причини використання процедурної генерації в моєму проекті.[20]

Перша з них це те, що даний метод дозволяє створити велику кількість ігрових рівнів за незначний проміжок часу. Наприклад, для створення одного

рівня вручну йшло від 40 до 60 хвилин, тоді коли процедурна генерація дозволяє створити його за декілька секунд.

Друга причина, це відсутність людської евристики при створенні рівнів. Оскільки Back Memory це головоломка, то процедурна генерація зробить її рівні більш контр-інтуітивними, а отже і більш цікавішими.

Слід зазначити, що для кожного проекту потрібен свій алгоритм залежно від бажаного результату та поставленої цілі, а кожен тип контенту вимагає свого власного підходу. Алгоритм процедурної генерації для даного проекту наведено на рисунку 4.5.

Рисунок 4.5 – Алгоритм процедурної генерації

4.3.2 Реалізація алгоритму процедурної генерації

Дані рівня

Це інформація про рівень яку обробляє генератор рівнів. Для її описання було створено два класи LevelData та Cell. LevelData має лише одне поле – двовимірний масив об’єктів класу Cell. Клас Cell описує дані про одну клітину поля, тобто до яких станів вона належить, чи є вона стартовою або фінішною, і т.д.

Компонент LevelSolver

Даний компонент відповідає за підрахунок можливих варіантів проходження рівня. В ігровій сцені GenerationLevel, він присвоєний пустому ігровому об’єкту LevelGeneration.

Головний метод компоненту це метод Solver, він приймає поле LevelData, та повертає числову змінну, яка означає кількість варіантів проходження рівня. Даний метод симулює проходження поля всіма можливими варіантами.

За репрезентацію віртуального об’єкта героя відповідає змінна типу INT класу Cell – NowCurrent. Тобто якщо NowCurrent дорівнює 0, то на клітині немає героя, якщо більше, то на клітині знаходиться один або більше об’єктів героя. Під час симуляції на ігровому полі може знаходитись багато віртуальних об’єктів героя, в тому числі і на одній клітині, а кроки прораховуються для всіх них одночасно.

Кожен крок відбувається в циклі while за допомогою метода Step, на початку цикла симулюється також зміна стану поля. В кінці циклу дані поля записуються в один з трьох списків (по одному на кожен стан поля), в залежності від поточного стану поля. Перед цим йде порівняння кожного елемента списку, з поточним полем, якщо знаходиться хоч один збіг (для кожного стану), то цикл зупиняється.

Компонент LevelGeneration

Даний компонент відповідає безпосередньо за генерацію рівня, алгоритм якої наведено на рисунку 4.5. В ігровій сцені GenerationLevel, він присвоєний пустому ігровому об’єкту LevelGeneration.

Метод Generation відповідає за ініціалізацію даних рівня. Першою дією, він створює новий об’єкт класу LevelData, з двовимірним масивом розміром m на n. Змінні m та n вводяться розробником вручну в Unity перед початком генерації. Далі визначається стартова та фінішна клітини поля, вони повинні бути розміщені в протилежних кутах матриці клітин.

Слід зазначати, що поле повинне складатися з темніших та світліших клітин які чергуються. Для цього в класі Cell існує змінна Color. Вона призначається кожній клітині в двох циклах for. Після ініціалізації ігрове поле має вигляд як на рисунку 4.6.

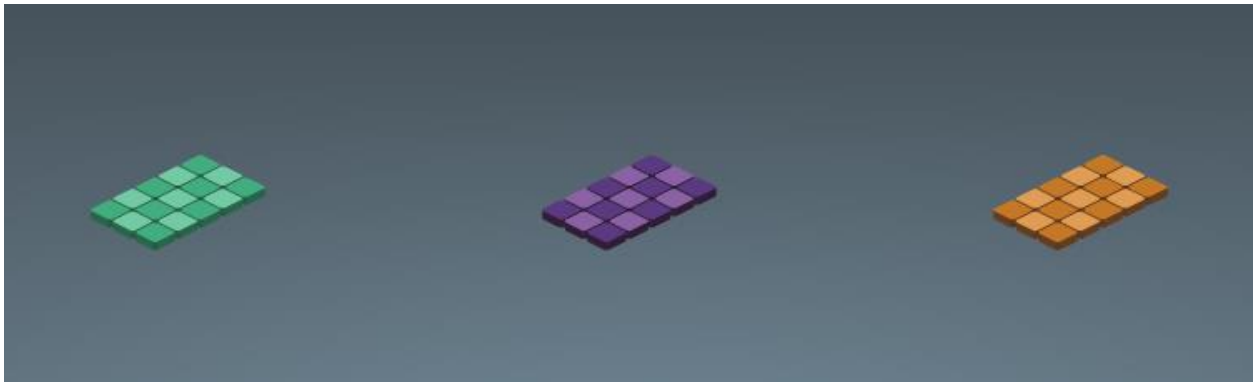


Рисунок 4.6 – Ігрове поле після ініціалізації

Наступний метод TrainNodes викликається автоматично після завершення Generation. Працює він наступним чином:

1) Відбувається зміна даних випадкової клітини;
2) За допомогою LevelSolver, прораховується кількість можливих проходжень рівня;
3) Якщо кількість можливих проходжень рівня дорівнює нулю, значить остання зміна клітина була невірною оскільки в даний момент рівень є непрохідним. Відбувається відкат зміни та повернення до пункту 1;

4) Якщо кількість можливих проходжень рівня більше 7, то цикл продовжується в пункті 1, якщо ні, то цикл зупиняється (Слід зазначити, що число 7 було обране після численних тестувань, як оптимальне);
5) Інколи після генерації на полі можуть залишитись поодинокі клітини, тобто клітини в яких немає сусідніх клітин. Вони є непотрібними, тому викликається метод DeleteOneNodes, який видаляє дані клітини.

Останнім кроком є збереження згенерованих даних.

Компонент SaveLevel

Даний компонент відповідає за збереження та завантаження даних рівня. Це відбувається в форматі бінарних файлів, для чого потрібен стандартний клас Unity - BinaryFormatter.

Для початку треба визначити місце куди будуть зберігатись файли з даними рівнів. Для цього в асстеах Unity була створена спеціальна папка під назвою Streaming Assets. Більшість ассетів Unity об’єднуються при побудові проекту, але якщо вони знаходяться в папці Streaming Assets, то поміщаються в звичайну файлову систему, що дає змогу мати до них доступ через посилання.

Алгоритм зберігання даних рівнів:

1) Визначення шляху для збереження даних. Шлях визначається у вигляді рядка string, і виглядає як: Application.dataPath + "/StreamingAssets" + "/generatedLevel" + Номер рівня +".save";
2) Створення нового об’єкту класу BinaryFormatter;
3) Створення файлу за шляхом, за допомогою директиви using;
4) Усередині using, за допомогою методу Serialize класу BinaryFormatter, дані рівня перетворюються в бінарний потік.

Алгоритм завантаження даних рівня працює аналогічно, за винятком використання методу Deserialize замість Serialize. Генерація та збереження даних рівня відбувається в сцені LevelGeneration, а завантаження даних відбувається в сцені Game. Отже компонент SaveLevel знаходиться в обох

сценах, в першій він присвоєний пустому ігровому об’єкту LevelGeneration, а в другій об’єкту Level.

4.3.3 Реалізація виводу рівня на сцену

Компонент LevelOutput

Після завантаження даних рівня в сцені Game, треба побудувати сам рівень, оскільки на даний момент сама сцена є пустою. Для цього і потрібен компонент LevelOutput. Даний процес відбувається за допомогою двох префабів та стандартної функції Instantiate.

Префаб – це спеціальний ти ассетів Unity, який дозволяє зберігати весь ігровий об’єкт зберігаючи всі його компоненти та властивості. Він виступає у ролі свого роду шаблону для створення екземплярів на сцені. Для будування поля в сцені Game використовуються такі префаби:

- префаб об’єкту ігрового поля Board;
- префаб об’єкту клітини поля Node;
- префаб об’єкту героя Player;
- префаб EndNode для позначення фінальної клітини.

Метод Instantiate є стандартним методом Unity, він приймає префаб та повертає його копію на ігровій сцені. Крім того, додатково можна також вказати положення та кут обертання створюваного об’єкта.

В компоненті LevelOutput існує всього два методи. Метод BoardSpawn за допомогою функції Instantiate створює на сцені три ігрові об’єкти Board, та надає їм властивості відповідно до одного з трьох станів.

Метод Output приймає дані рівня у вигляді матриці об’єктів класу Cell. Далі за допомогою двох циклів for, він створює ігрові об’єкти клітин та надає їм відповідні властивості, відповідно до об’єкту Cell, такі як: колір, стан поля, чи є клітина фінішною, і тд. Якщо створена клітина є фінішною, то на її координатах створюється об’єкт EndNode.

Після цього, ігрові об’єкти Board за допомогою методу GetNodeList, шукають на сцені об’єкти Node які мають відповідний стан, та записують в список.

Компонент PlayerSpawn

Даний компонент відповідає за створення об’єктів героя на сцені. В ігровій сцені Game, він присвоєний пустому ігровому об’єкту Level.

Метод SpawnPlayer працює аналогічно до попереднього методу Output, він приймає дані рівня, після чого за допомогою циклів for здійснює пошук стартової клітини. Після цього, за допомогою функції Instantiate на сцені створюються три об’єкти героя. Слід зазначити, що об’єкт героя створюється набагато вище об’єкта ігрового поля.

Перед початком проходження рівня гравцю потрібно дати шанс запам’ятати всі три стани поля. Для цього була створена анімація приземлення гравця на поле, під час якої стани поля змінюються в компоненті TimeManager. Дана анімація реалізована за допомогою корутини LandRoutine та iTween, під час її програшу гравець не може нічого вводити.

Компонент Main

Даний компонент потрібен для виклику функцій, що були наведені вище, в послідовному порядку. В ігровій сцені Game, він присвоєний пустому ігровому об’єкту Level. Main викликає функції в стандартному методі Start в такому порядку:

1) Зчитування даних про номер поточного рівня;
2) Завантаження даних рівня, метод Load;
3) Створення об’єктів ігрових полів, метод BoardSpawn;
4) Створення об’єктів клітин поля, метод Output;
5) Створення об’єктів героя, метод SpawnPlayer.

4.3.4 Генерація рівнів

Після реалізації процедурної генерації залишилось лише згенерувати самі рівні. Даний процес відбувається з безпосередньою участю розробника. Для цього була створена додаткова ігрова сцена, до якої гравець не матиме доступу.

Щоб згенерувати рівень розробнику в компоненті LevelGeneration потрібно ввести значення розмірів ігрового поля для рівня, а в компоненті SaveLevel ввести номер рівня який генерується, рисунок 4.7. Крім того були створені дві кнопки одна з яких генерує рівень та виводить його на сцену а інша зберігає дані про нього, рисунок 4.8.

Однак, у гравця також є можливість генерувати рівні для гри, для цього було створено три методи: Easy, Middle, та Hard, кожен з яких означає рівень складності рівня, що генерується. Складність рівня залежить від його розміру.

Приклади згенерованих рівнів наведені в Додатку Г.

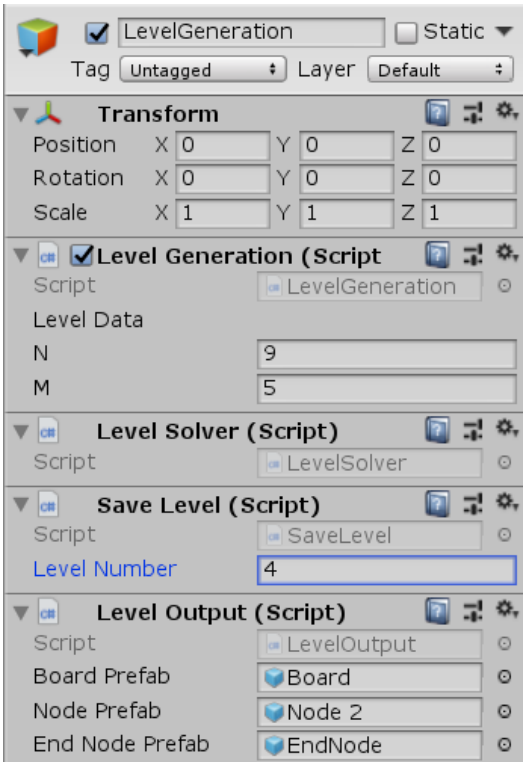


Рисунок 4.7 – Компонент LevelGeneration

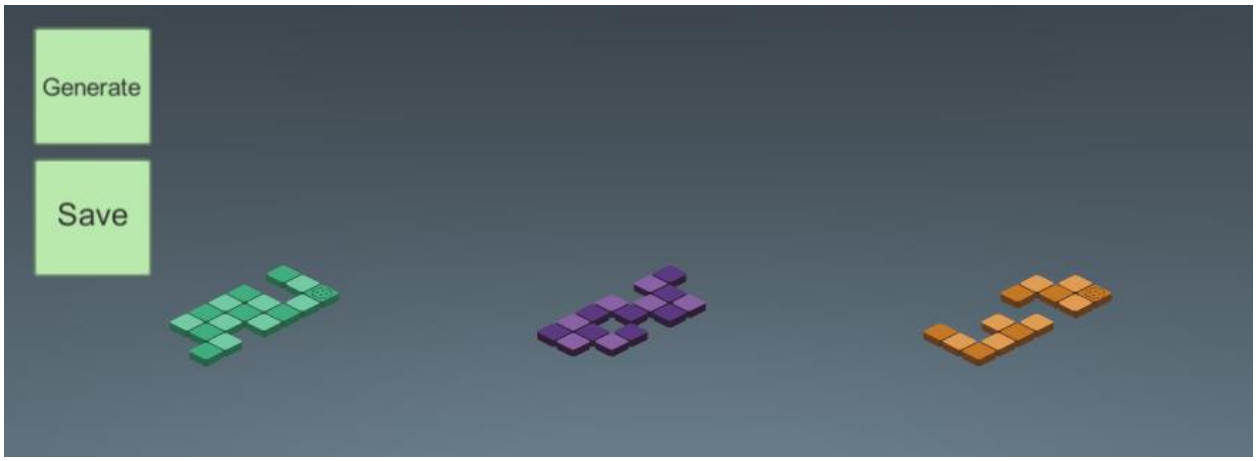


Рисунок 4.8 – Інтерфейс генерації рівнів

4.3 Візуалізація

4.2.1 3D-моделювання

3D-моделювання та анімація виконується в програмі Autodesk 3ds Max.

Модель клітини поля. Оскільки на ігровому полі може бути багато клітин, модель клітини повинна бути простою, щоб не зменшувалась продуктивність ігрового додатка (особливо мобільної версії). Отже вона не повинна мати великої кількості полігонів.

Для моделювання клітини поля було створено паралелепіпед з рівними довжиною та шириною, але з меншою в 4 рази висотою. Після чого було використано метод полігонального моделювання, для скруглення його бічних ребер. Готова модель клітини ігрового поля зображена на рисунку 4.9.

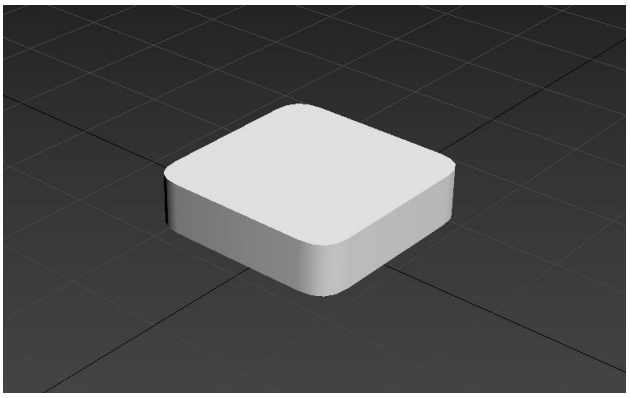


Рисунок 4.9 - Модель клітини

Модель героя

Аналогічно до попередньої моделі, модель гравця не повинна бути дуже детальною. Оскільки користувач матиме змогу бачити її лише з далекої відстані, та лише з певного ракурсу.

Модель героя складається з простих геометричних фігур: сфери, циліндра, двох пірамід, та двох конусів, деякі яких були деформовані методом полігонального моделювання. Модель героя зображена на рисунку 4.10.

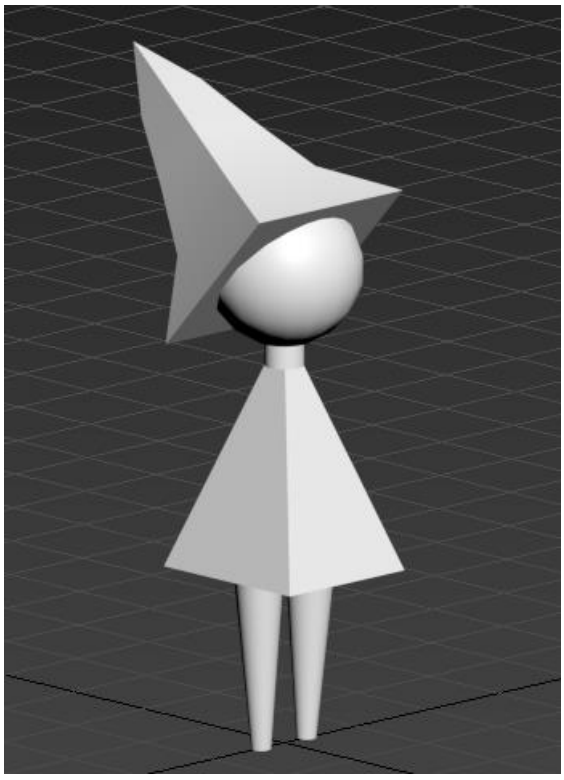


Рисунок 4.10 - Модель героя

4.2.2 Анімація

Моделі героя необхідні анімації які будуть програватись під час переміщення по полю. Для цього, в 3Ds Max, була використана скелетна анімація, за допомогою системи кісток (Bones).

Bones

Це спеціальний тип об’єктів, які ієрархічно зв’язані між собою, та використовуються, в більшості випадках, для анімації інших об’єктів та ієрархій. Bones дозволяє проводити маніпуляції для тіла героя. Дана система, як правило, поміщається всередину тіла об’єкта анімації (можна сказати, що імітує скелет тіла), після чого визначається особливості та послідовність зв’язування елементів між собою. Певна частина об’єкту персонажа - прив'язується до певної кістки.

Систему кісток для моделі героя, зображено на рисунку 4.11.

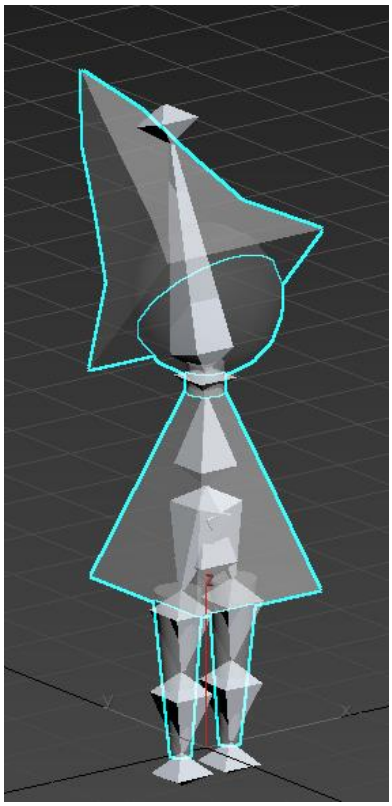


Рисунок 4.11 - Система кісток

Модифікатор Skin

Даний модифікатор потрібен для створення «шкіряної» оболонки для системи кісток. Це значить, що кожен 3D-об’єкт, до якого застосовується цей модифікатор, набуває форми капсульної оболонки. Тобто цей модифікатор прив’язує оболонку героя до скелету.

Після додавання до моделі модифікатора Skin, слід обрати систему кісток та налаштувати кожній з них ваги (weight) для вершин моделі. Вага кістки визначає наскільки сильно кістка впливає на вершину, вона може бути числом від 0 до 1.

Для анімації кісток ніг головного героя, також була використана інверсна кінематика.

Інверсна кінематика

Це метод анімації, який відрізняється від прямої кінематики протилежним принципом успадкування. При прямій кінематиці трансформація об’єктів нащадків визначається трансформацією об’єктів їхніх предків, в той час як при інверсній – об’єкти нащадки рухають батьківські об'єкти, тобто 3Ds Max розраховує анімацію батьківських об'єктів, виходячи з анімації об’єкта нащадка.[21] Інверсна кінематика для ніг героя, зображена на рисунку 4.12.

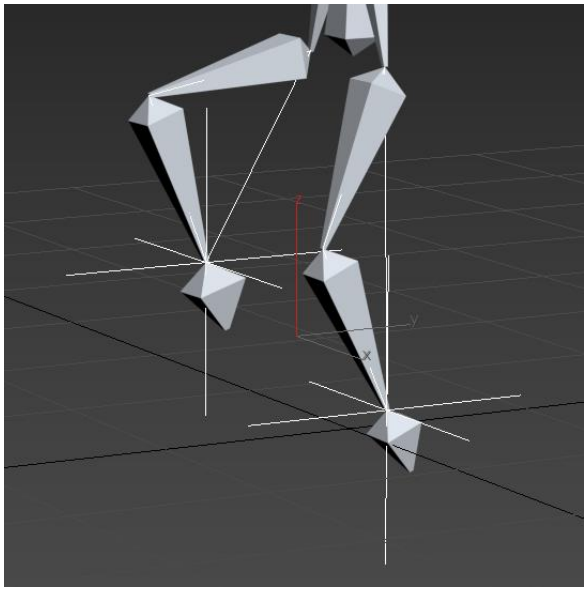


Рисунок 4.12 - Інверсна кінематика

Описані вище дії були лише підготовкою до створення анімації. Схему анімації переміщення героя, зображено на рисунку 4.13. Герой стрибком переміщується на сусідню клітину.

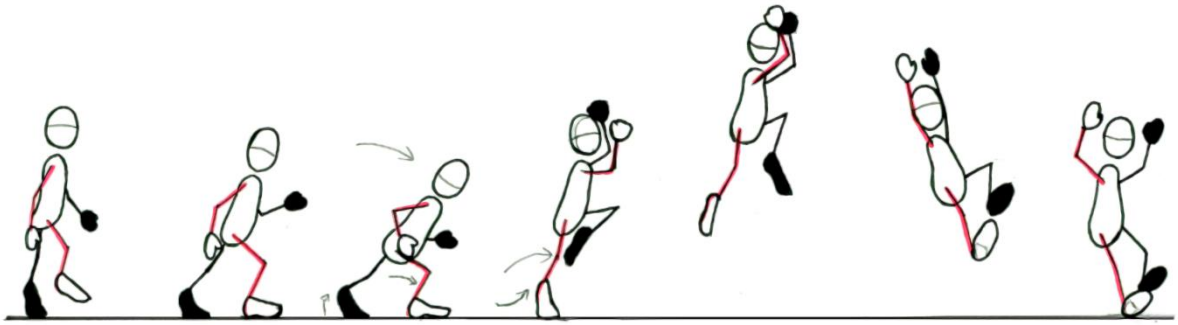


Рисунок 4.13 - Схема анімації переміщення героя

Для більшої різноманітності було зроблено дві такі анімації, які є дзеркальними одна до одної. Тобто в першій герой опирається на ліву ногу, а в другій на праву. Під час гри, вони програються по черзі. На рисунку 4.14 можна побачити таймлайн анімацій в 3Ds Max.

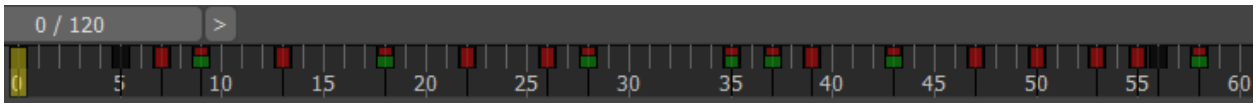


Рисунок 4.14 - Таймлайн анімацій

4.2.3 Експорт в Unity

Після створення моделей та анімацій необхідно виконати їх експорт в ігровий додаток. Для цього, в 3Ds Max існує інструмент Game Exporter та універсальний формат Autodesk .FBX.[22]

Даний інструмент призначений спеціально для розробників ігрових додатків, для збільшення ефективності експорту ігрових ассетів. Також, він дає можливість працювати з анімаційними кліпами, можна експортувати

одразу декілька кліпів одним файлом або декількома файлами. Також для кожного кліпу можна встановлювати початок та кінець відтворення, а також робити попередній перегляд кліпу.

Отже анімація до Unity експортується у вигляді анімаційних кліпів. Для управління цими кліпами потрібен Animator Controller.

Animator Controller

Це стандартний ассет Unity, який дозволяє організовувати управління анімаційними кліпами в ігровому об’єкті. Він містить в собі посилання на всі анімаційні кліпи, які в ньому використовуються, а також керує анімаційними станами та переходами. Все це робиться за допомогою блок-схеми State Machine. Дану схему, для ігрового об’єкта героя можна побачити на рисунку 4.15.

Стан Idle означає бездіяльний стан героя, в цей час не програється ніяка анімація, а герой просто стоїть. FirstMove та SecondMove це анімації переміщення гравця, які програються по черзі. Після програвання однієї з анімацій стан повертається в Idle.

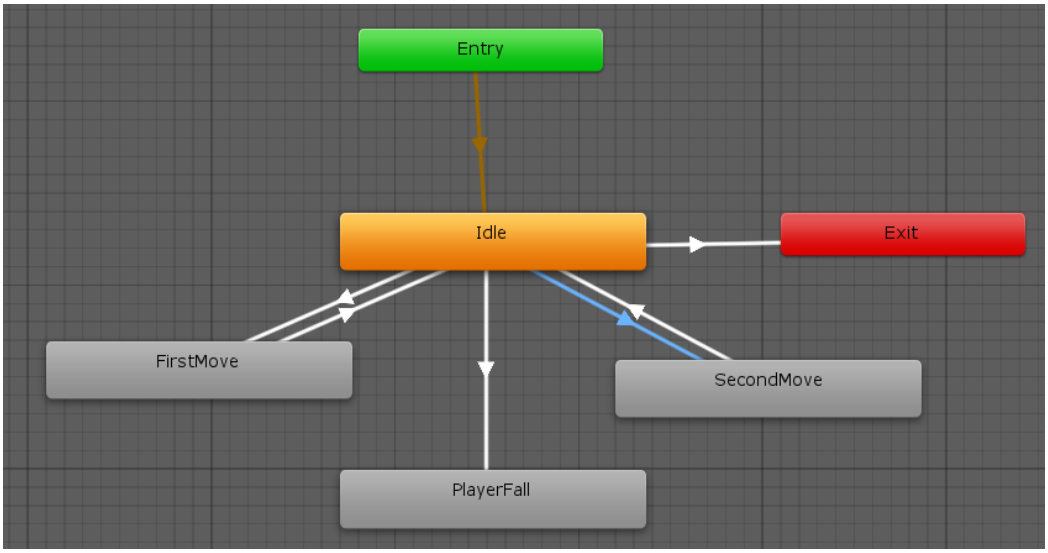


Рисунок 4.15 - Блок-схема State Machine

4.2.4 Система часток

Для надання ігровому додатку більш привабливого вигляду було вирішено використати систему часток Fireflies. Для створення системи часток в Unity існує компонент ParticleSystem. На рисунках 4.16 - 4.17 зображені налаштування для системи часток Fireflies.

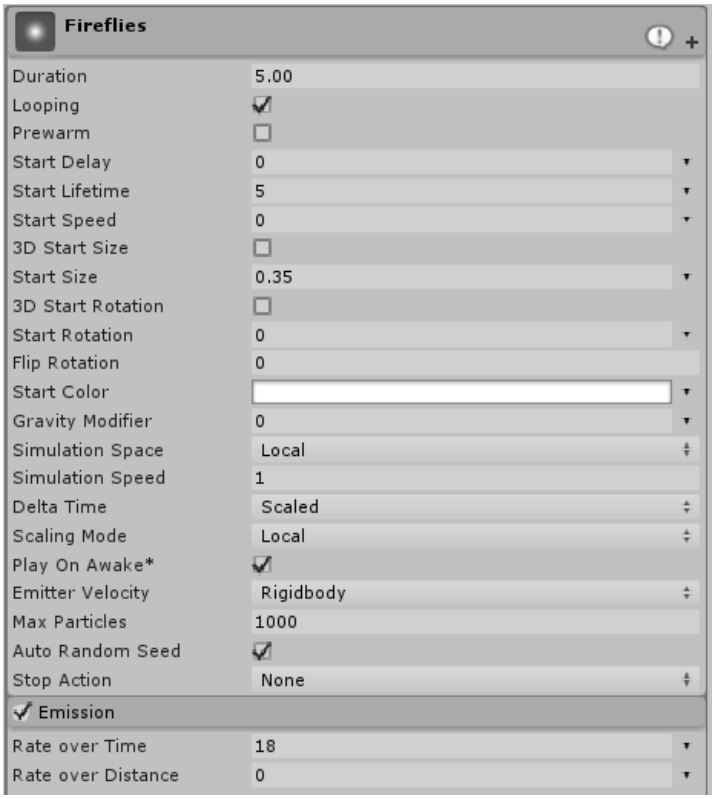


Рисунок 4.16 - Налаштування для системи часток Fireflies

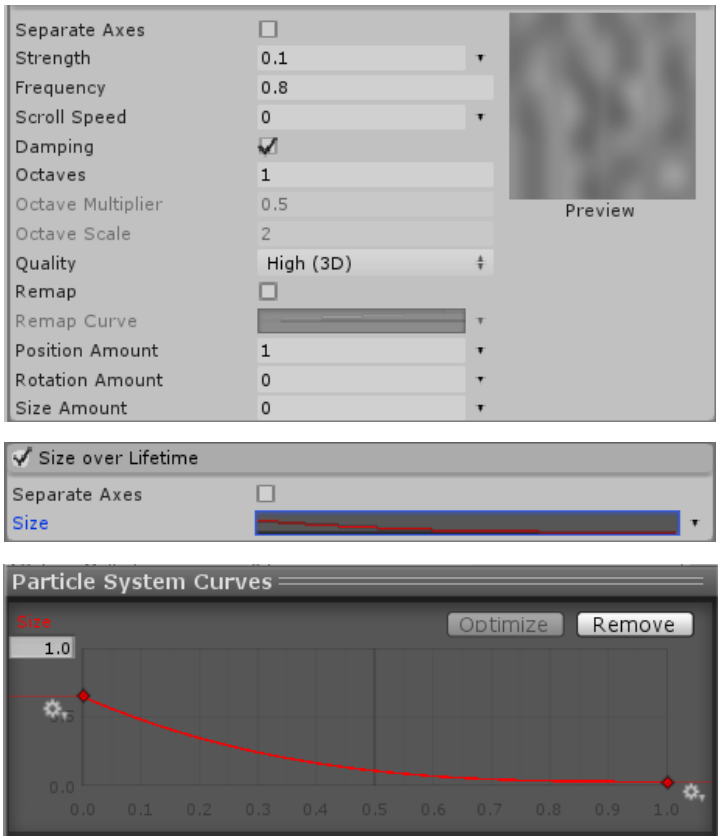


Рисунок 4.17 - Налаштування для системи часток Fireflies

4.2.5 Інтерфейс

Для розробки інтерфейсу, в Unity, були використані такі стандартні компоненти:

- Canvas. Можна сказати, що це область в якій знаходяться всі UI об’єкти Unity. Даний компонент присвоєний однойменному ігровому об’єкту, та є стандартним в Unity. Всі UI об’єкти повинні бути його дочірніми елементами;
- Text. Відображає неінтерактивний текстовий елемент
- Button. Може бути натиснута користувачем, для запуску певної події;
- Image. Відображає спрайт в системі UI;
- Slider. Надає можливість користувачу обрати числове значення в певному діапазоні. Використовується для регулювання гучності в додатку.

Головне меню

Для реалізації головного меню була створена нова ігрова сцена – MainMenu, в якій було створено об’єкт Canvas, об’єкт Text та шість об’єктів Button. Після налаштувань даних об’єктів головне меню має вигляд як на рисунку 4.18.

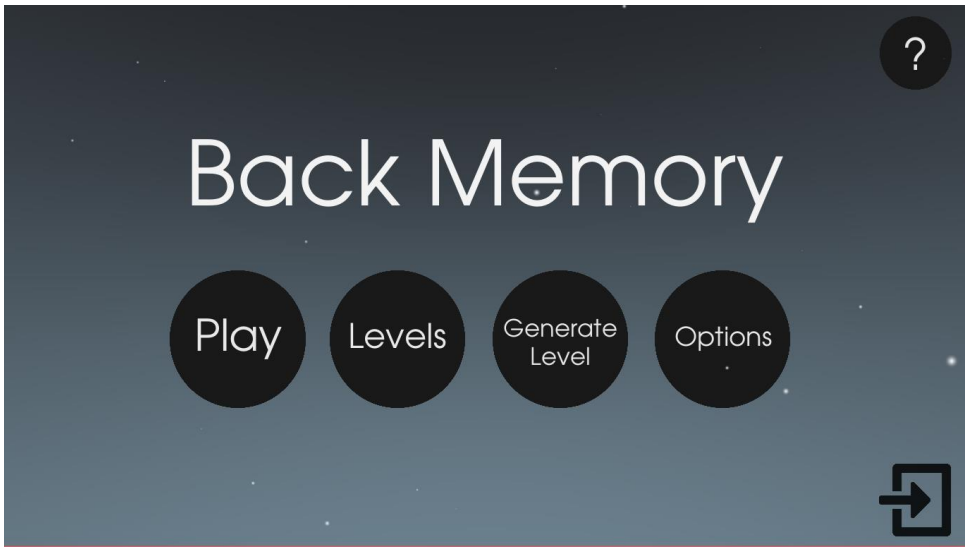


Рисунок 4.18 – Головне меню

Компонент MainButtonsScript

В даному компоненті знаходяться функції які викликається при натисканні кнопок головного меню. Він присвоєний ігровому об’єкту Canvas. Відповідно він має такі методи:

- PlayClick. Перехід на сцену Game;
- LevelClick. Перехід на сцену Levels;
- GenClick. Перехід на сцену GenerationLevel;
- OptionsClick. Поява панелі PanelOptions;
- Tutorial. Перехід на сцену Tutorial;
- Exit. Завершує роботу ігрового додатку.

Також даний компонент має метод ChangeLanguage, який змінює мову кнопок головного меню. Для зберігання невеликої даних, такої як: дані про номер поточного рівня, дані про кількість пройдених рівнів, дані про обрану

мову додатку, та дані про гучність звуку – використовується стандартний клас Unity під назвою PlayerPrefs.

PanelOptions

Дана панель знаходиться в сцені MainMenu, та складається з трьох об’єктів Button, та об'єктів Text та Slider. Після налаштувань даних об’єктів головне меню налаштувань має вигляд як на рисунку 4.19.

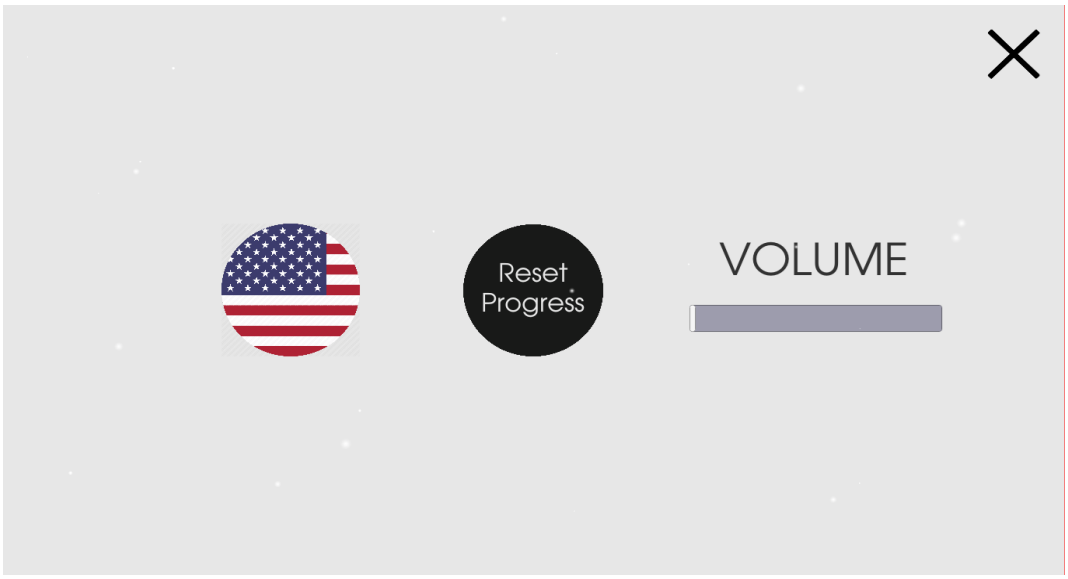


Рисунок 4.19 – Меню налаштувань

Компонент OptionsButtonScript

В даному компоненті знаходяться функції які викликається при натисканні кнопок та зміни гучності в меню налаштувань. Він присвоєний ігровому об’єкту Canvas, та відповідно має такі методи:

- ClickLanguage. Зміна даних про мову додатка;
- ValueChangeCheck. Зміна гучності звуку;
- ClickReset. Скидання даних про номер поточного рівня, та даних про кількість пройдених рівнів;
- ClickResetAttention. Поява панелі “Попередження про скидання даних”, Рисунок 4.20;
- ClickNoReset. Зникнення панелі “Попередження про скидання даних”;
- ClickExit Зникнення панелі налаштувань.

Рисунок 4.20 – Попередження про скидання даних

Меню рівнів

Для реалізації даного меню була створена нова ігрова сцена – Levels, в якій було створено об’єкт Canvas, декілька об’єктів Button, кожен з яких відповідає за вибір конкретного рівня.

Компонент ButtonScript

Даний компонент присвоєний кожному об’єкту кнопки. Він має числову змінну LevelNumber, яка означає номер рівня який запускає конкретна кнопка.

Кожна кнопка може бути активною чи неактивною, якщо змінна LevelNumber більша за кількість пройдених рівнів гравцем, то кнопка стає неактивною. Візуально "неактивність" кнопки визначається появою спрайта замка замість номеру, рисунок 4.21. За описані вище дії відповідає функція UpdateButton. Інша функція – Click, викликається при натисканні на кнопку. Вона змінює дані про номер поточного рівня та виконує перехід на сцену Game.

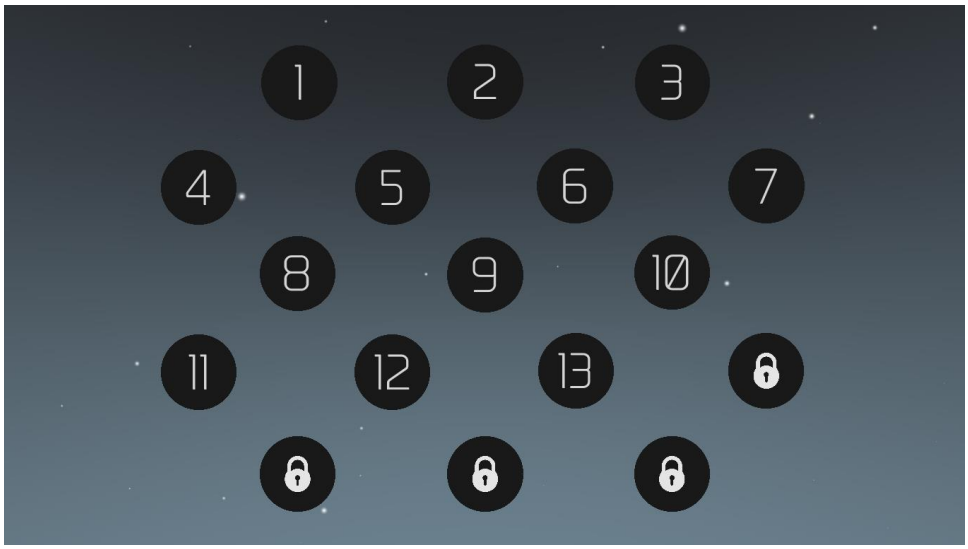


Рисунок 4.21 – Меню вибору рівнів

Компонент Levels

На екрані знаходиться лише 17 кнопок для рівнів, в той час як кількість рівнів в додатку значно більша. Даний компонент відповідає за скролінг рівнів. Для цього були створені дві додаткові кнопки для скролінгу вліво та вправо.

Компонент Levels присвоєний ігровому об’єкту Canvas, та відповідно має такі методи:

- Left. Викликається після кліку на кнопку прокрутки вліво. Додає 17 до LevelNumber в кожному об’єкті типу Button;
- Right. Викликається після кліку на кнопку прокрутки вправо. Віднімає 17 від LevelNumber в кожному об’єкті типу Button;
- ArrowsUpdate. Ховає або виводить кнопки для скролінгу;
- Back. Перехід на сцену MainMenu.

Меню генерації рівнів

Дане меню було реалізоване в ігровій сцені GenerationLevel, в якій було створено об’єкт Canvas, об’єкт Text, та три об’єктів Button. Після налаштувань даних об’єктів меню має вигляд як на рисунку 4.22. Кожна з кнопок викликає відповідний метод з компоненту LevelGeneration, після чого йде перехід до сцени Game.

Рисунок 4.22 – Меню генерації рівнів

Меню в ігровому процесі

Під час ігрового процесу можуть бути викликані три панелі меню: меню паузи, меню перемоги, та меню програшу. Всі вони реалізовані завдяки однієї панелі Panel, вміст якої залежить від того методу, який її викликає. Дані методи знаходяться в компоненті PanelScript:

- PausePanel. Викликає меню паузи, яке дозволяє перезапустити рівень, або повернутись до головного меню. Викликається після подвійного тапу по екрану Android, або після натискання Escape;
- WinPanel. Викликає меню перемоги, яке дозволяє перезапустити рівень, повернутись до головного меню, або перейти до наступного рівня. Викликається після того, як об’єкт героя став на фінішну клітину. Панель меню перемоги зображена на рисунку 4.23;
- LosePanel. Викликає меню програшу, яке дозволяє перезапустити рівень, або повернутись до головного меню. Викликається після того, як гравець програв.

Для реалізації подвійного тапу/кліку в компоненті PlayerInput був створений метод DoubleTapCheck який викликається кожен кадр в функції Update, та корутина DobleTapDetection. Корутина викликається в DoubleTapCheck після того як було зроблено два кліка, та перевіряє час між

ними за допомогою стандартного класу Time. Якщо час між двома кліками менше ніж пів секунди, викликається метод PausePanel.

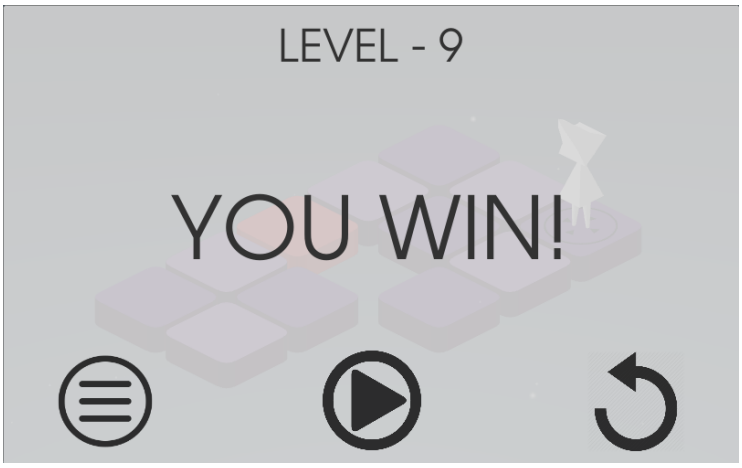


Рисунок 4.23 – Меню перемоги

Ефект затухання

Стандартний перехід між сценами в Unity є дуже різким. Щоб це згладити було вирішено створити ефект плавного затемнення екрану. Для цього було створено новий об’єкт Canvas та поміщено в нього об’єкт BlackFade типу Image.

Для BlackFade було створено два анімаційні кліпи: Fade In (Рисунок 4.24) та Fade Out (Рисунок 4.25). Після чого було створено об’єкт стандартного класу Animator Controller в який було додано анімаційні кліпи. Анімаційний кліп Fade In запускається автоматично на початку кожної сцени. Блок схема State Machine зображена на рисунку 4.26.

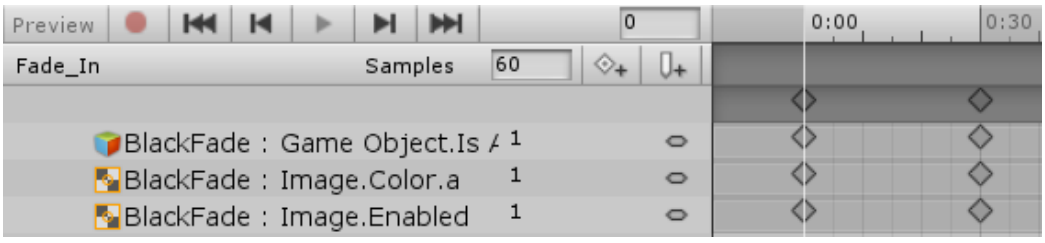


Рисунок 4.24 – Анімація Fade In

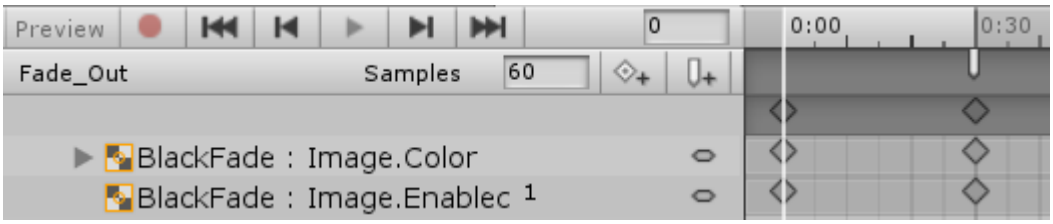


Рисунок 4.25 – Анімація Fade Out

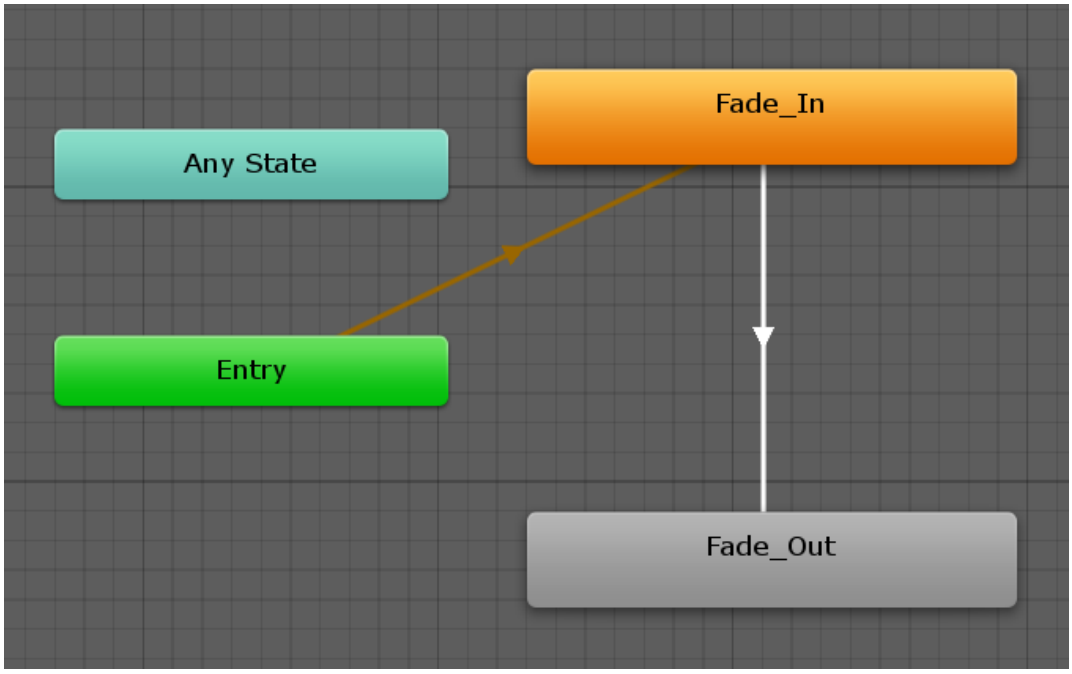


Рисунок 4.26 – Блок схема State Machine

Компонент LevelChanger

Даний компонент присвоєний об’єкту Canvas, та має посилання на об’єкт Animator Controller. Його метод FadeToLevel приймає змінну типу string з назвою рівня, та запускає анімаційний кліп FadeOut. На останньому кадрі даного кліпу, коли екран затух, розміщений тригер, який викликає інший метод даного компоненту – OnFadeComplete, що здійснює перехід на іншу сцену.

4.4 Додаткові елементи

4.4.1 Звук

Був виконаний пошук звуків для гри. В додатку використовується такі звукові файли, кожен з яких є безкоштовним:

- Три звуки кроків героя;
- Три звуки зміни стану поля;
- Звук початку гри;
- Звук перемоги;
- Звук програшу;
- Звук натискання на кнопку.

Компонент Audio Listener

Даний компонент репрезентує мікрофон, який приймає звуки в 3D просторі, та програє їх через динаміки пристрою гравця. На сцені, він присвоєний об’єкту камери.

Клас Sound

Даний клас описує звук, який використовується в ігровому додатку. Він має такі поля як: ім’я, посилання на аудіокліп, гучність, та висота звуку.

Компонент AudioManager

Даний компонент відповідає за роботу з звуками в ігровому додатку. Він має масив, в якому містяться всі об’єкти класу Sound. В методі ChangeVolume завантажуються дані про гучність та призначаються кожному об’єкту типу Sound. Наступний метод Play приймає назву об’єкту типу Sound, та запускає аудіокліп цього об’єкту.

4.4.2 Туторіал

Туторіал потрібен ігровому додатку, щоб навчити гравця базовим механічним діям у комфортному середовищі. Для цього була створена нові ігрова сцена з відповідною назвою. На ній вручну було створене ігрове поле та об’єкти гравця.

Для навчання гравця також було додано об’єкт тексту та декілька спрайтів. Текст та спрайти відповідно змінюється після кроків гравця. За це відповідає компонент Demo. Для кращого розуміння управління ігровим процесом, були створені спрайти кнопок та свайпів, відповідно до платформи. Скріншот туторіалу наведений на рисунку 4.27.

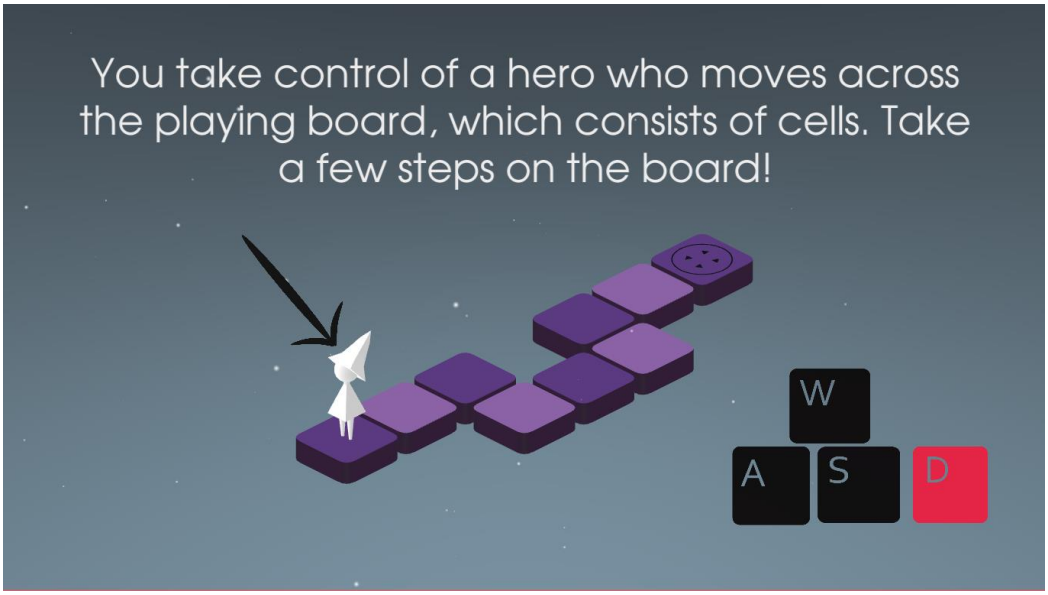


Рисунок 4.27 – Туторіал гри

Якщо гравець вперше зайшов в додаток, і одразу натиснув на кнопку початку гри, то з’являється панель яка пропонує гравцю спочатку пройти туторіал, яка зображена на рисунку 4.28.

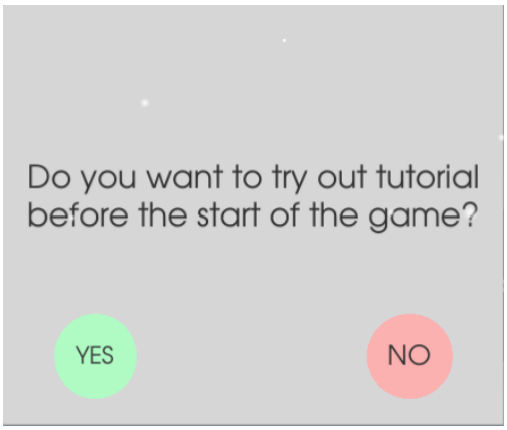


Рисунок 4.28 – Панель туторіалу

4.4.3 Додаткові механіки

Механіка червоних клітин

Для збільшення різноманітності ігрового процесу було вирішено додати в проект додаткові ігрові механіки. Першою з них є механіка червоних клітин (Рисунок 4.29). Суть полягає в тому, що якщо герой стане на таку клітину більше одного разу, то гравець програє.

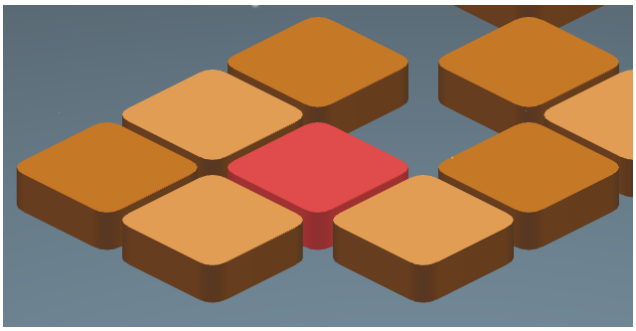


Рисунок 4.29 – Червона клітина

Для цього в компонент Node було додане поле endNode яке відповідає за стан червоної клітини, метод FallNodeStep який викликається після того як об’єкт героя став на цю клітину, та корутини FallNodeRoutine1 та FallNodeRoutine2. В компоненті LevelGeneration генерація червоних клітин відбувається в методі TrainFallNodes.

FallNodeRoutine1 викликається коли герой став на клітину перший раз, та запускає зациклену анімацію блимання кольору клітини, з інтервалом 0.2 секунди. Це дає зрозуміти гравцю, що на клітину більше не можна ставати.

FallNodeRoutine2 викликається коли герой став на клітину другий раз, та запускає анімацію падіння клітини, за допомогою iTween. Після цього гравець програє.

Механіка турелей

Друга додаткова механіка це механіка турелей. Якщо після кроку герой знаходиться на одній лінії з туреллю – турель вистрілює та гравець програє. Для початку в 3Ds max було створено модель турелі. За реалізацію вистрілу турелі відповідає компонент LaserNode.

В компоненті LevelGeneration генерація турелей відбувається в методі TrainLaserNodes. Дана механіка зображена на рисунку 4.30.

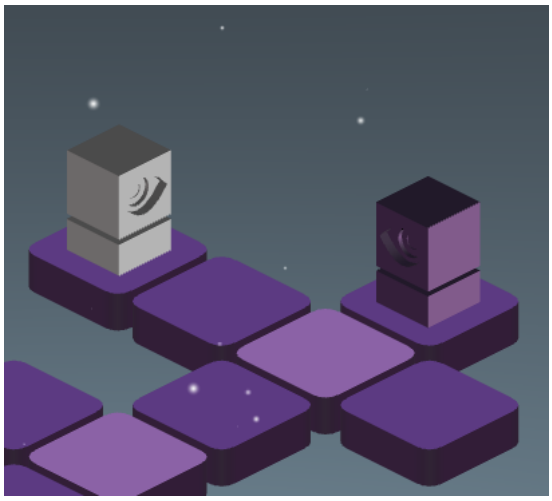


Рисунок 4.30 – Турелі

ВИСНОВКИ

На початку розробки даної роботи був проведений детальний аналіз предметної області проекту. Після чого, було прийняте рішення про розробку ігрового додатку, через високу актуальність обраної теми на сьогоднішній день.

В наступній частині були описані існуючі аналоги та проведений їх аналіз. Після чого було описане програмне забезпечення, та обґрунтовано його вибір.

Під час проектування, для ігрового додатку, були розроблені: діаграма IDEF0, діаграма IDEF3, діаграма варіантів використання, та діаграма Flowchart.

Після проектування, була розпочата безпосередньо розробка проекту. Спочатку були реалізовані базові механічні дії гри, такі як: ігрове поле та переміщення по ньому. Після цього була створена процедурна генерація рівнів. Потім була зроблена візуалізація додатку, а саме: 3D-моделі, анімації, інтерфейс, та ін.

Результатом роботи над проектом є готовий ігровий додаток "Back Memory" для операційних систем Android та Windows.

СПИСОК ЛІТЕРАТУРИ

1. Джо Хокінг Unity в дії. Мультиплатформенна розробка на C # [Текст].

2. The Global Games Market Will Reach $108.9 Billion in 2017 With Mobile Taking 42% [Електронний ресурс] – Режим доступу до ресурсу: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/

3. Итоги Steam за 2017 год [Електронний ресурс] – Режим доступу до ресурсу: http://app2top.ru/industry/itogi-steam-za-2017-god-ot-sergeya-galyonkina-118491.html

4. Cubiscape [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=cz.p3tr.escapethecube&hl=ru

5. Lara Croft Go [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Lara_Croft_Go

6. Nova-111 [Електронний ресурс] – Режим доступу до ресурсу: https://store.steampowered.com/app/325370/Nova111/

7. Vandals [Електронний ресурс] – Режим доступу до ресурсу: https://store.steampowered.com/app/809240/Vandals/

8. Unity [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Unity_(game_engine)

9. What is Unreal Engine 4 [Електронний ресурс] – Режим доступу до ресурсу: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

10. CryEngine [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/CryEngine

11. Game Maker Studio 2 [Електронний ресурс] – Режим доступу до ресурсу: https://www.yoyogames.com/gamemaker/features

12. Методологія IDEF0 [Електронний ресурс] – Режим доступу до ресурсу: http://projectimo.ru/biznes-processy/idef0.html

13. Методологія IDEF3 [Електронний ресурс] – Режим доступу до ресурсу: https://itteach.ru/bpwin/metodologiya-idef3

14. Основы UML [Електронний ресурс] – https://pro-prof.com/archives/2594

15. What is a Flowchart [Електронний ресурс] – https://www.lucidchart.com /pages/what-is-a-flowchart-tutorial

16. iTween - Asset Store [Електронний ресурс] – https://assetstore .unity.com/ packages/tools/animation/itween-84

17. Unity - Manual: Coroutines [Електронний ресурс] – https://docs.unity 3d.com /Manual/Coroutines.html

18. Грег Лукосек Learning C# by Developing Games with Unity 5.x - Second Edition [Текст].

19. Robert Penner's Easing Functions [Електронний ресурс] – http://robertpenner.com/easing/

20. Procedural generation [Електронний ресурс] – https://en.wikipedia.org/ wiki/Procedural_generation

21. IK Solvers|3ds Max 2016|Autodesk Knowledge Network [Електронний ресурс] – https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/ CloudHelp/cloudhelp/2016/ENU/3DSMax/files/GUID-AE4A0089-95F5-4199-A8 53-ABB8E0DB3439-htm.html

22. Руководство экспорта FBX [Електронний ресурс] – https://docs.unity3d.com/ru/current/Manual/HOWTO-exportFBX.html

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку гри «Back Memory»

Суми 2019

Призначення й мета створення гри

Призначення

Продукт призначений для розважально-освітніх цілей, та орієнтований на широке поле користувачів.

Мета створення

Стимулювання покращення пам’яті гравця, розвиток логічного мислення, покращення його вмінь приймати адекватні рішення та прораховувати події. Крім того, допомога користувачу відпочити, та провести час із задоволенням та користю.

Цільова аудиторія

Можна виділити такі характеристики цільовії аудиторії продукту:

- користувачі Windows та Android;
- 16 - 28 років;
- 55% чоловіків, 45% жінок.

Вимоги до програмного продукту

Вимоги до функціональних можливостей

Оскільки даний проект є відеогрою, для нього передбачається лише одна категорія користувачів – гравці. Під час гри гравець є учасником ігрового процесу і безпосередньо впливає на нього.

Продукт повинен мати наступний функціонал:

Ігровий функціонал

Програма повинна дотримуватися таких правил гри:

- гравцю представляється ігрове поле, що складається з квадратних клітин;
- переміщення по полю може здійснюватися в чотирьох напрямках: вперед, назад, вліво, та вправо, на одну клітинку за крок;
- кожен рівень складається з трьох різних полів, при цьому гравець в будь який момент часу бачить лише одне з них, а після кожного ходу гравця, одне поле змінюється на інше;
- якщо після того як було зроблено хід, поле змінюється, а під гравцем немає клітини – герой програє;
- після кожного програшу, гравець повертається на початок рівня;
- ціль кожного рівня пройти по полю, від точки А (початку) до точки В (кінця), після чого йде перехід на наступний рівень;
- Інформація про кожен пройдений рівень зберігається.

Графічний та звуковий функціонали

- вибір розширення екрану;
- вибір якості графіки;
- вибір між повноекранним та віконним режимом;
- вибір гучності звуку.

Інтерфейс

- головне меню;
- меню вибору рівнів;
- графічний інтерфейс користувача.

Вимоги до вхідних та вихідних даних

Вхідними даними програми є ігрові налаштування, які обирає користувач, а також управління аватаром під час ігрового процесу, за допомогою клавіатури або свайпів.

Вихідними даними є графічна інтерпритація процесу гри на екрані та звук, який її супроводжує. Дії гравця впливають на стан ігрової сцени.

Вимоги до апаратного забезпечення

Мінімальні:

- Процесор: Intel® Pentium 4 або AMD Athlon ™ 64 X2;
- Оперативна пам'ять: 1 GB ОЗУ;
- Відеокарта: ATI Radeon X700 (256 MB) або NVidia Equivalent (256 MB) або краща;
- Місце на диску: 2 GB;

Рекомендовані:

- Процесор: Intel Core ™ 2 Duo (3.0 GHz) або AMD Athlon 64 X2 5000+ (2.6 GHz) або кращий;
- Оперативна пам'ять: 2 GB ОЗУ;
- Відеокарта: NVIDIA GeForce 8800 GT (512 MB) або ATI Radeon HD 4870 (1Gb) або краща;
- Місце на диску: 2 GB.

Вимоги до програмного забезпечення

Програма повинна функціонувати під управлінням ОС сімейства Windows наступних версій: Windows XP, Vista, 7, 8. У додатку використовуються

бібліотеки платформи.NET Framework. Також потрібно встановлений DirectX 9.0c або більш пізньої версії.

Для функціонування програми на ПК – повинна бути операційна система: Windows 7 / Windows 8 / Windows 10. Також потрібен бути встановлений DirectX 9.0c або пізніша версія.

Для функціонування програми на мобільному пристрої – повинна бути операційна система Android 2.3.3 або новіша версія.

3 Склад і зміст робіт зі створення сайту

Докладний опис етапів створення гри наведено в табл. 1.

Таблиця 1 – Етапи створення гри

№	Склад і зміст робіт	Строк розробки (в робочих днях)
1	Реалізація базових механік	14
2	Реалізація процедурної генерації	20
3	3D-моделювання	7
4	Реалізація інтерфейсу	3
5	Створення анімацій	7

ДОДАТОК Б

Планування робіт

Таблиця Б.1 Деталізація мети методом SMART

S (Конкретність)	Розробити 3D гру в жанрі головоломка з використанням випадкової генерації рівнів.
M (Вимірюваність)	Ціль буде досягнуто коли проект буде зданий в експлуатацію та заархівований.
A (Досяжність)	Бюджет проекту складає 81 730 грн.
R (Доцільність)	Ринок відеоігор є актуальним, і приносить високі прибутки.
T (Обмеженість в часі)	Проект буде виконано вчасно, що підтверджується календарним планом проекту

WBS структура. Це інструмент, який розбиває проект на складові частини, для визначення рамок проекту. Він надає ієрархічно структурований розподіл робіт, які потрібні для реалізації проекту.

У ході побудови даної структури здійснюється послідовний поділ робіт проекту на менші компоненти, які є більш керованими, до рівня пакетів робіт. Ці пакети зазвичай належать до самого нижнього рівня деталізації, та складаються з окремих одиниць робіт. Усі елементи WBS структури одного рівня – повинні бути достатніми для реалізації відповідного елемента вищого рівня.

По суті, дана структура являє собою перелік усіх завдань, необхідних для реалізації проекту, яка може бути у вигляді простого опису або у графічному вигляді. Для даного проекту, було розроблено другий варіант. Можна сказати, що WBS структура визначає та організовує зміст проекту, а роботи які не є в неї включеними – не вважаються роботами проекту.

WBS структура повинна відображати архітектуру створюваного проекта на її верхньому рівні, та дозволяти переходити до інших структур, які

характеризують специфічні роботи на нижніх рівнях. Останні, входять до календарного плану робіт проекту, а також по ним можна оцінити витрати на проект і час його виконання. Набір цих робіт повинен бути достатнім та необхідним для виконання всього проекту.

Можна виділити такі основні етапи розробки WBS структури:

- Визначення ступеню деталізації;
- визначення кількості рівнів;
- визначення структур кожного рівня;
- короткий опис кожного елементу;
- кодування;
- проведення обчислень.

OBS структура. Це інструмент, який використовується для відображення учасників проекту та відповідальних осіб, які залучені до проекту. Елементами даної структури є як цілі служби, так і окремі робітники, при цьому взаємозв'язки можуть підтримуватись через вертикальні та горизонтальні зв'язки, і носять функціональний характер. Структура будується на основі WBS структури.

В межах цієї структури буде проходити сам управлінський процес, де будуть розподілюватись завдання та функції управління, а також відповідальність за виконання робіт.

Побудова календарного графіку. Для побудови календарного графіку розробки проекту Back Memory була використана Діаграма Ганта. Це інструмент, що використовується для ілюстрації розкладу проекту. Діаграма представляє собою відрізки, що розміщені на горизонтальній часовій шкалі, при цьому кожен відрізок являє собою певну задачу з WBS структури. Завдяки даній діаграмі можна відстежити:

- дати початку та завершення виконання проекту;
- дати виконання кожного завдання;
- тривалість виконання кожного завдання;
- способи об'єднання завдань.

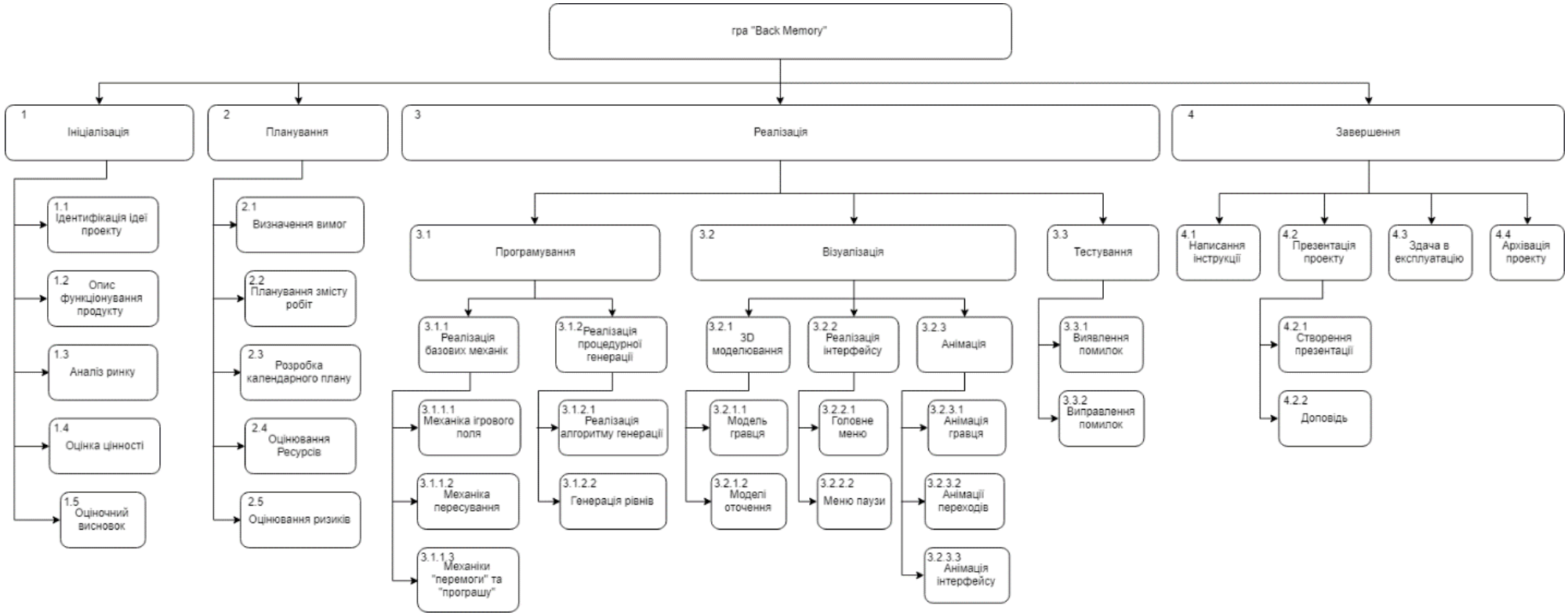


Рисунок Б.1 – WBS структура проекту

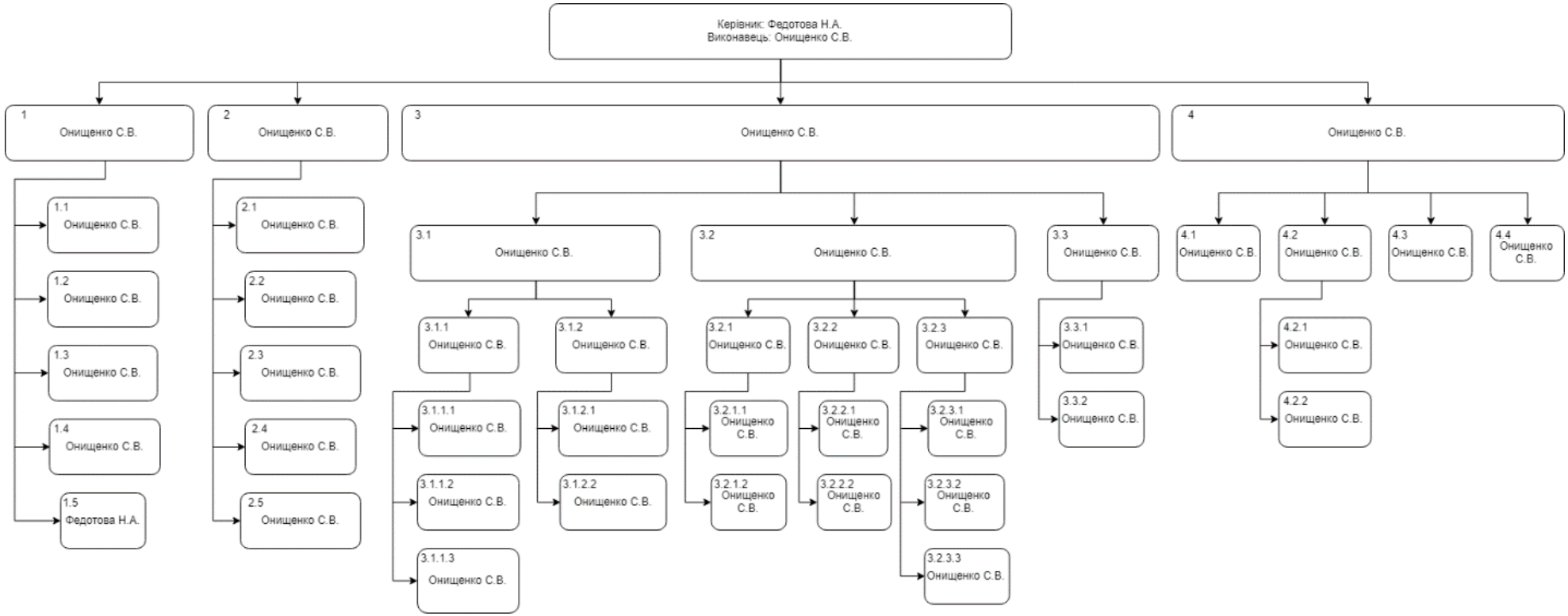


Рисунок Б.2 – OBS структура проекту

Название задачи	Длите	Начало	Окончание	Преді
Ідентифікація ідеї проекту	5 дней	Пт 01.02.19	Чт 07.02.19	
Опис функціонування продукту	5 дней	Пт 01.02.19	Чт 07.02.19	
Аналіз ринку	5 дней	Пт 01.02.19	Чт 07.02.19	
Оцінка цінності	5 дней	Пт 01.02.19	Чт 07.02.19	
Оціночний висновок	1 день	Пт 08.02.19	Пт 08.02.19	1;2;3;4
Описання вимог для проекту	2 дней	Пн 11.02.19	Вт 12.02.19	5
Описання робіт проекту	1 день	Ср 13.02.19	Ср 13.02.19	6
Планування графіку робіт	1 день	Чт 14.02.19	Чт 14.02.19	7
Оцінювання ресурсів	1 день	Пт 15.02.19	Пт 15.02.19	8
Оцінювання ризиків	1 день	Пн 18.02.19	Пн 18.02.19	9
Реалізація базових механік	14 дней	Вт 19.02.19	Пт 08.03.19	10
Реалізація процедурної генерації	20 дней	Пн 11.03.19	Пт 05.04.19	11
3D моделювання	7 дней	Пн 11.03.19	Вт 19.03.19	11
Реалізація інтерфейсу	3 дней	Ср 20.03.19	Пт 22.03.19	13
Анімація	7 дней	Пн 25.03.19	Вт 02.04.19	14
Виявлення помилок	10 дней	Ср 03.04.19	Вт 16.04.19	15
Виправлення помилок	14 дней	Ср 03.04.19	Пн 22.04.19	15
Написання інструкції	7 дней	Вт 23.04.19	Ср 01.05.19	17

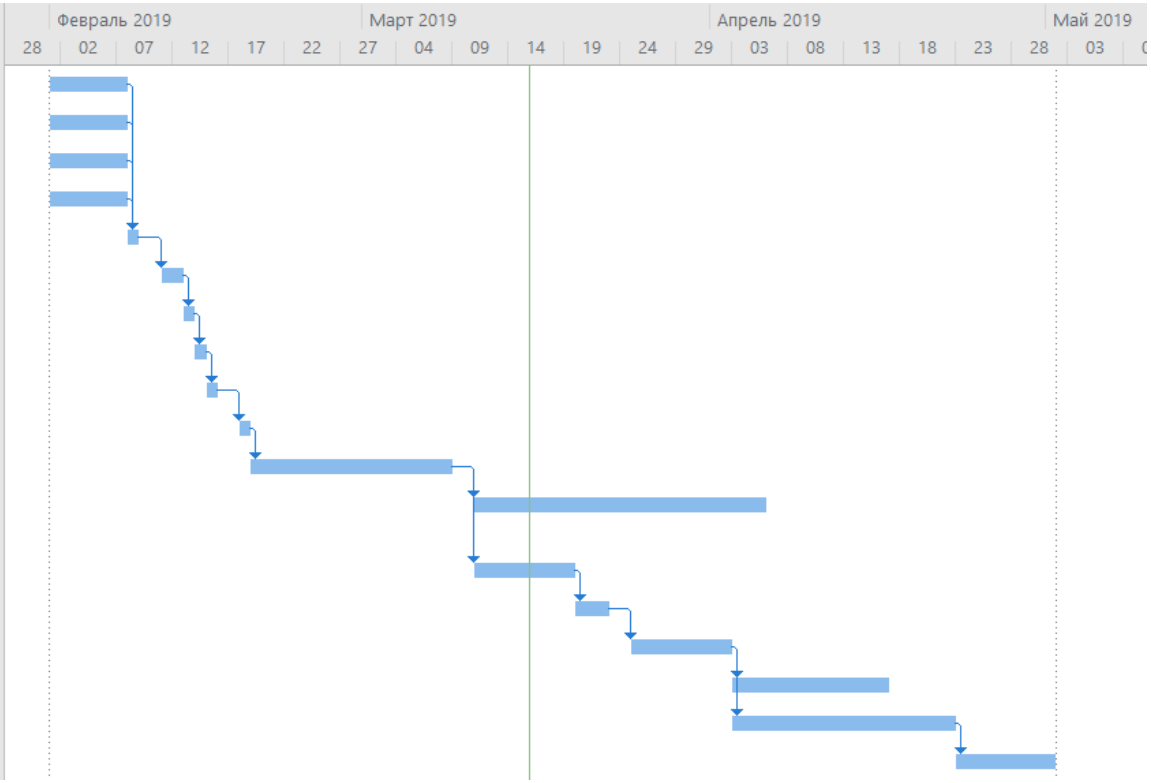


Рисунок Б.3 – Діаграма Ганта

Управління ризиками. Після аналізу, були виявлені такі ризики розробки проекту:

- слабке розуміння предметної області;
- низькі навички програмування;
- слабке планування і управління розробкою;
- відсутність досвіду роботи з Unity;
- проблеми з обладнанням;
- різке скорочення строків здачі роботи;
- людський фактор;

Для надання оцінок ймовірності ризиків було застосовано таку шкалу:

- дуже низька – 0.1;
- низька – 0.3;
- середня – 0.5;
- висока – 0.7;
- дуже висока – 0.9.

Для оцінок втрат від ризиків було застосовано наступну шкалу:

- дуже малі – 0.05;
- малі – 0.1;
- середні – 0.2;
- високі – 0.4;
- дуже високі – 0.8.

Таблиця Б.2 Ймовірність виникнення ризиків та втрати

№	Ризики	Ймовірність	Втрати
Р1	Слабке розуміння предметної області	0.5	0.2
Р2	Низькі навички програмування	0.3	0.4
Р3	Слабке планування та управління розробкою	0.5	0.4
Р4	Відсутність досвіду роботи з Unity	0.7	0.05

Таблиця Б.2 Продовження

№	Ризики	Ймовірність	Втрати
Р5	Проблеми з обладнанням	0.1	0.2
Р6	Різке скорочення строків здачі роботи	0.1	0.8
Р7	Людський фактор	0.3	0.1

Після цього було розраховано ранг ризиків, шляхом множення ймовірності на втрати. Якщо результат більше 0.15, то ризик є небезпечним, якщо від 0.06 до 0.15, то це середній ризик, а якщо менше 0.06, значить ризик є незначним.

Таблиця Б.3 Матриця ймовірності та втрат

Ймовірність	-	**0.9**	0.045	0.09	0.18	0.36	0.72
	Р4	**0.7**	0.035	0.07	0.14	0.28	0.56
	Р1 Р3	**0.5**	0.025	0.05	0.1	0.2	0.4
	Р2 Р7	**0.3**	0.015	0.03	0.06	0.12	0.24
	Р5 Р6	**0.1**	0.005	0.01	0.02	0.04	0.08
			0.05	**0.1**	**0.2**	**0.4**	**0.8**
			Р4	Р7	Р1 Р5	Р2 Р3	Р6
			Втрати				

Був виявлений один небезпечний ризик розробки проекту – слабке планування та управління розробкою. Середні ризики це низькі навички програмування та різке скорочення строків здачі роботи. А такі ризики як: слабке розуміння предметної області, відсутність досвіду роботи з Unity, проблеми з обладнанням, та людський фактор є незначними.

Таблиця Б.4 Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність	Втрати	Ранг	План А	Тип стратегії реагування	План Б
P1	Відкритий	Слабке розуміння предметної області	0.5	0.2	0.1	Більш глибокий аналіз предметної області	Пом’якшення	Пошук фахівців з предметної області
P2	Відкритий	Низькі навички програмування	0.3	0.4	0.12	Покращення навичок по ходу розробки	Пом’якшення	Більш глибоке вивчення програмування, онлайн уроки
P3	Відкритий	Слабке планування та управління розробкою	0.5	0.4	0.2	Провести аналіз процесів та робіт. Звернути увагу на розподіл часу.	Пом’якшення	Змінити порядок робіт. Оптимізація роботи із вже існуючою розстановкою.
P4	Відкритий	Відсутність досвіду роботи з Unity	0.7	0.05	0.035	Вивчення Unity по ходу розробки	Пом’якшення	Більш глибоке вивчення Unity, онлайн уроки

Таблиця Б.4 Продовження

ID	Статус ризику	Опис ризику	Ймовірність	Втрати	Ранг	План А	Тип стратегії реагування	План Б
P5	Відкритий	Проблеми з обладнанням	0.1	0.2	0.02	Залучити спеціаліста для усунення проблем	Прийняття	Замінити обладнання
P6	Відкритий	Різке скорочення строків здачі роботи	0.1	0.8	0.08	Узгодити усі питання на початкових етапах,	Попередження	Переоцінка проекту
P7	Відкритий	Людський фактор	0.3	0.1	0.03	Врахувати при побудові календарного графіку	Прийняття	-

ДОДАТОК В

Планування робіт

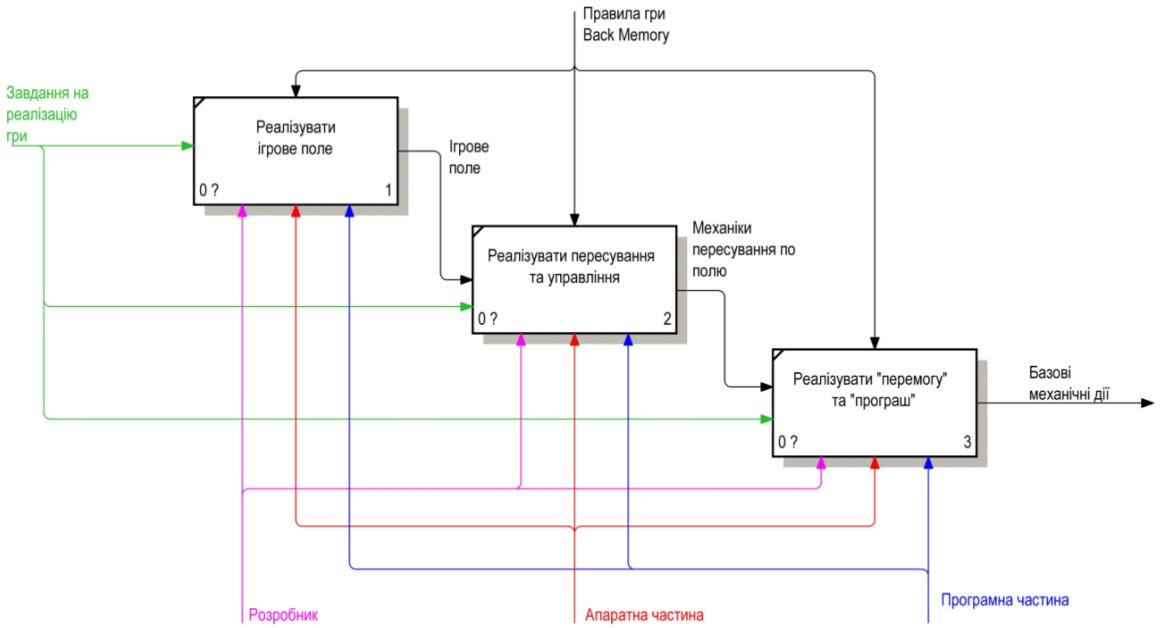


Рисунок В.1-Діаграма декомпозиції IDEF0 «Реалізувати базові механічні дії»

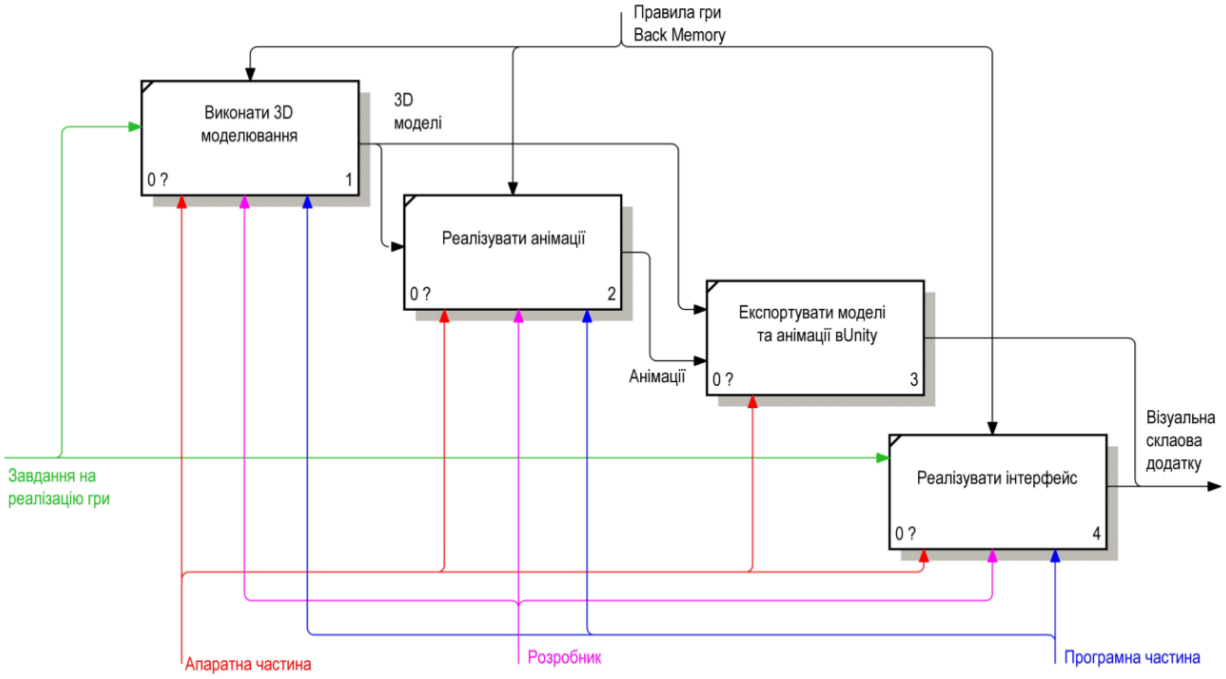


Рисунок В.2 – Діаграма декомпозиції IDEF0 «Створити графіку та анімації»

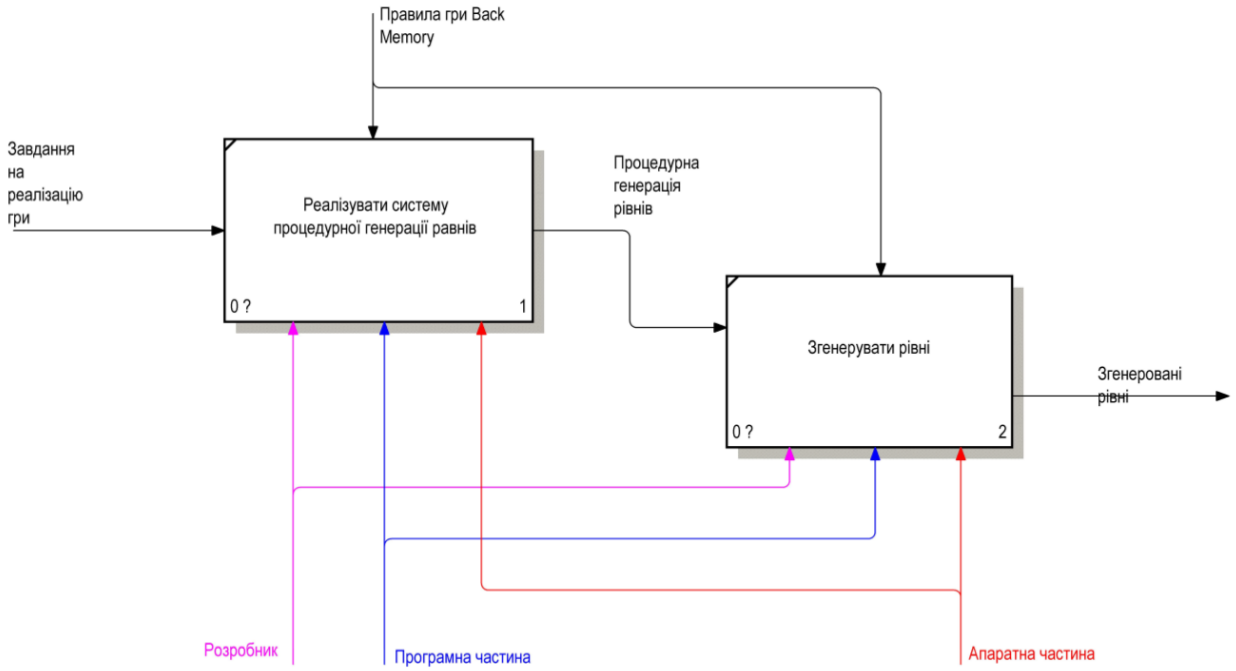


Рисунок В.3 – Діаграма декомпозиції IDEF0 «Реалізувати процедурну генерацію рівнів»

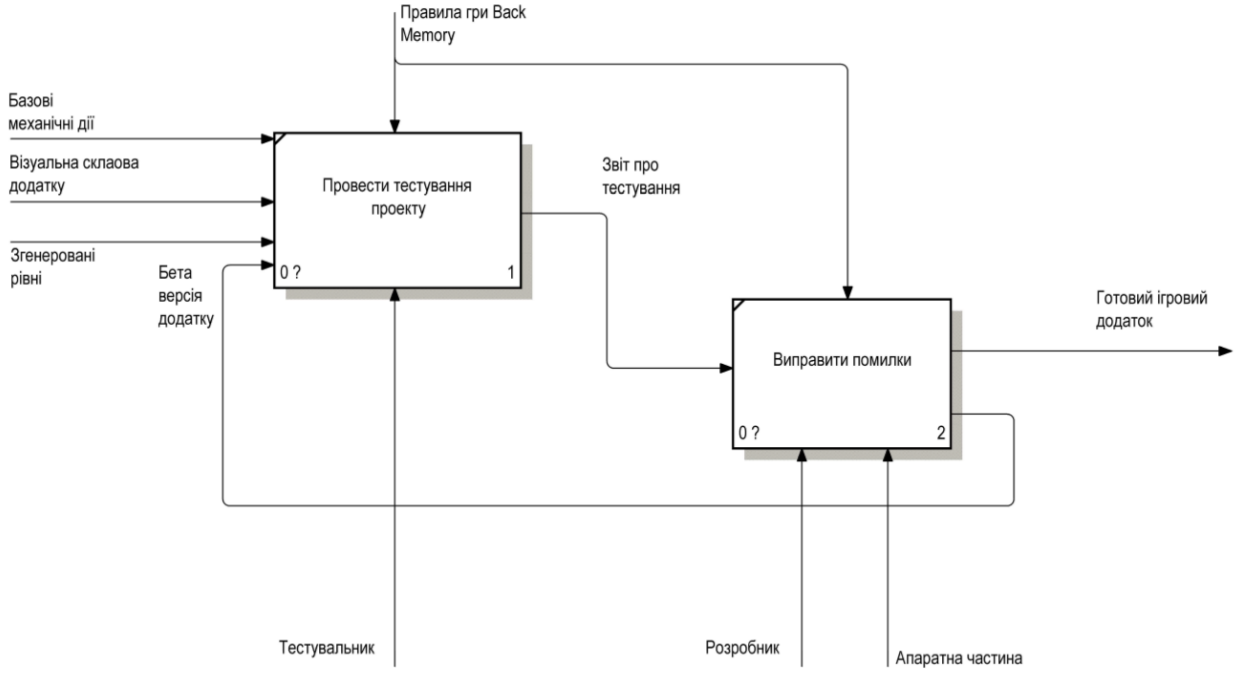


Рисунок В.4 – Діаграма декомпозиції IDEF0 «Виконати тестування проекту»

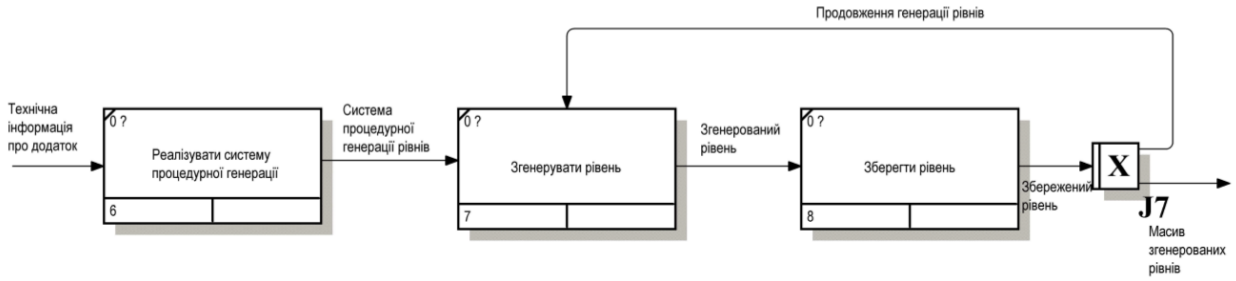


Рисунок В.5 – Діаграма декомпозиції IDEF3 «Реалізувати процедурну генерацію рівнів»

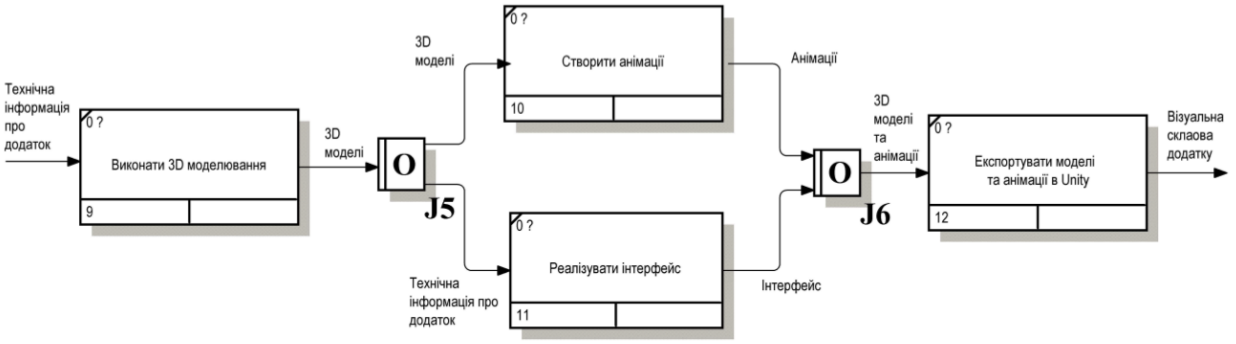


Рисунок В.6 – Діаграма декомпозиції IDEF3 «Створити графіку та анімації»

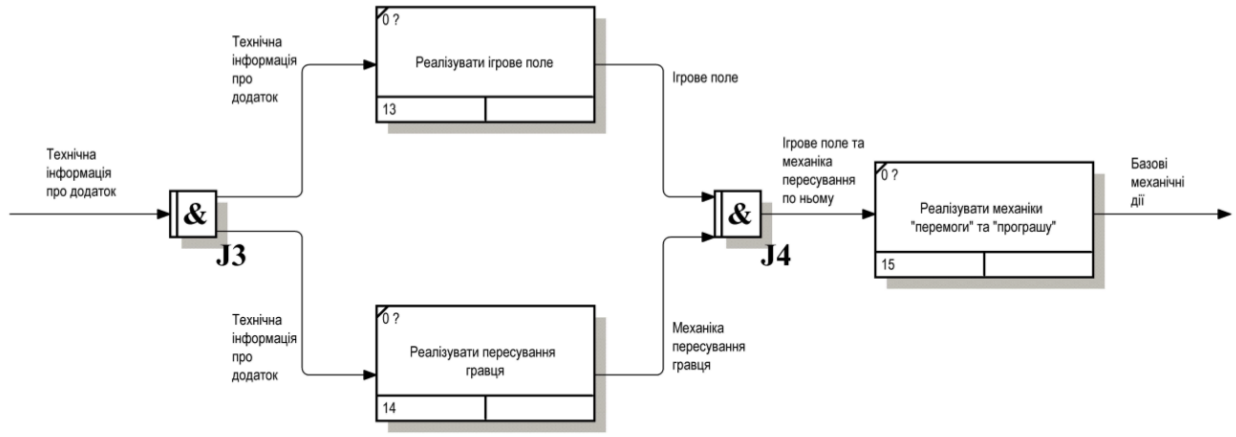


Рисунок В.7 – Діаграма декомпозиції IDEF3 «Реалізувати базові механічні дії»

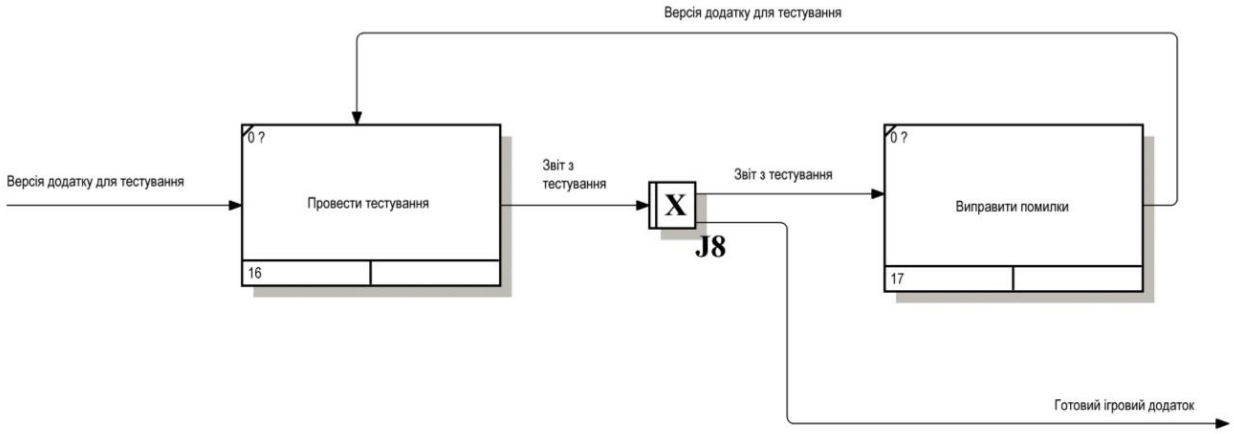


Рисунок В.8 – Діаграма декомпозиції IDEF3 «Тестувати проект»

ДОДАТОК Г

Приклади згенерованих рівнів

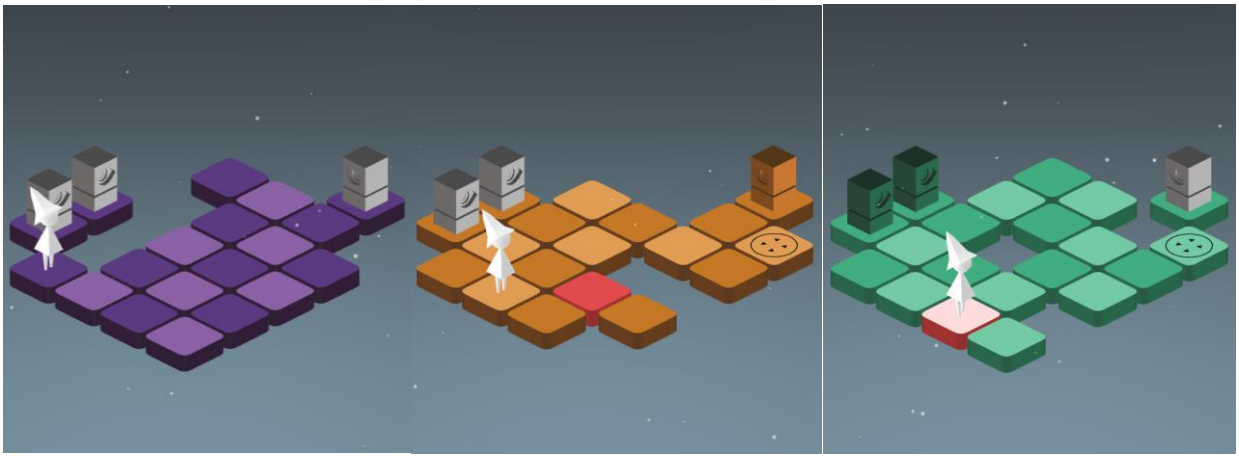


Рисунок Г.1 - Приклад згенерованого рівня

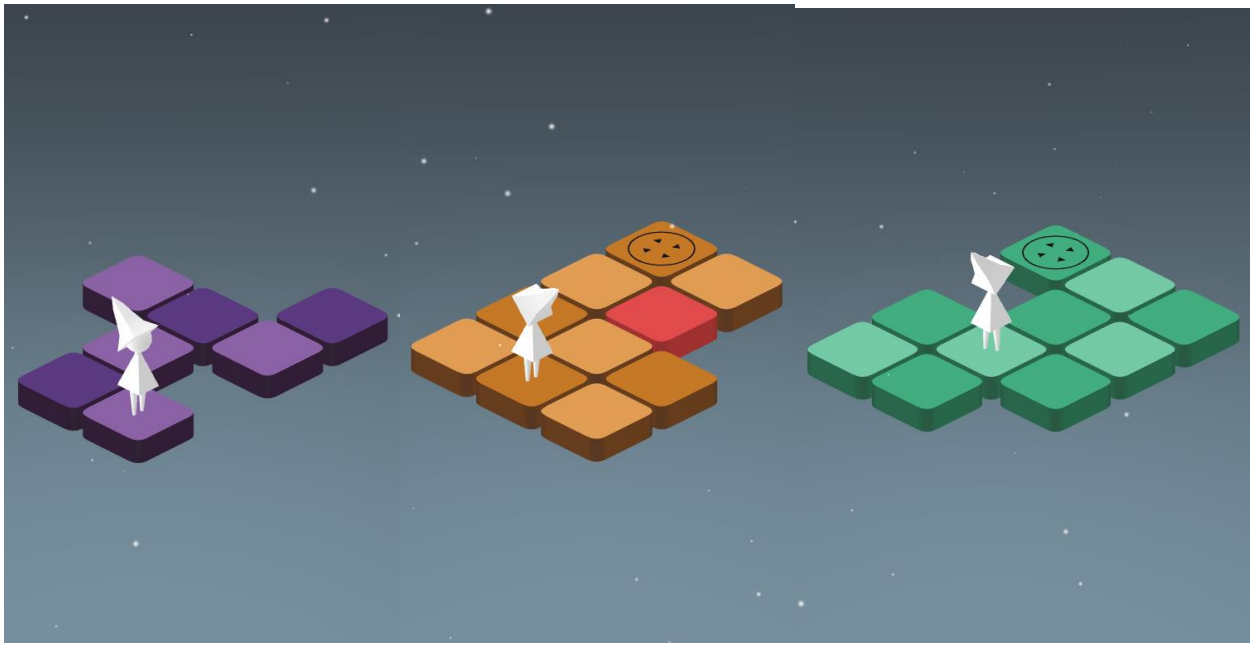


Рисунок Г.2 - Приклад згенерованого рівня

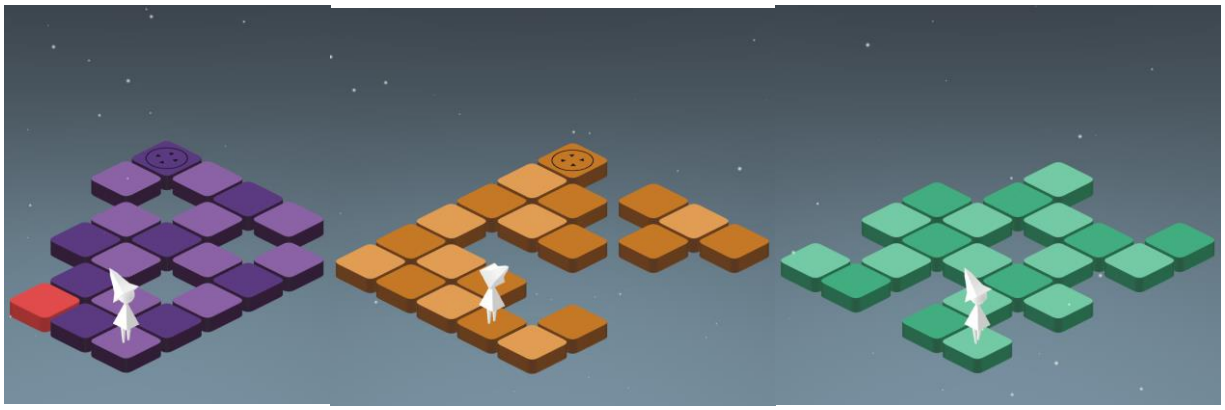


Рисунок Г.3 - Приклад згенерованого рівня

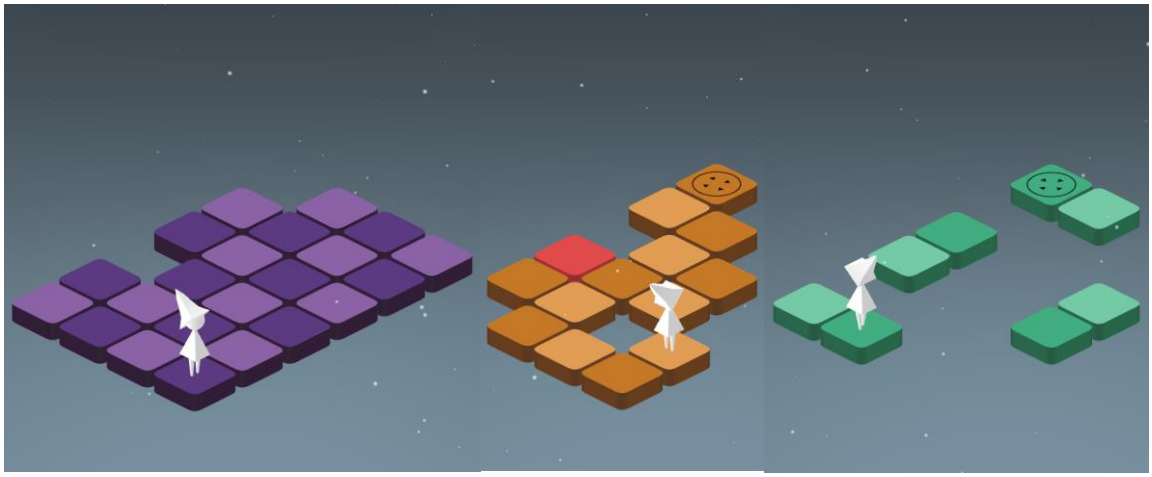


Рисунок Г.4 - Приклад згенерованого рівня

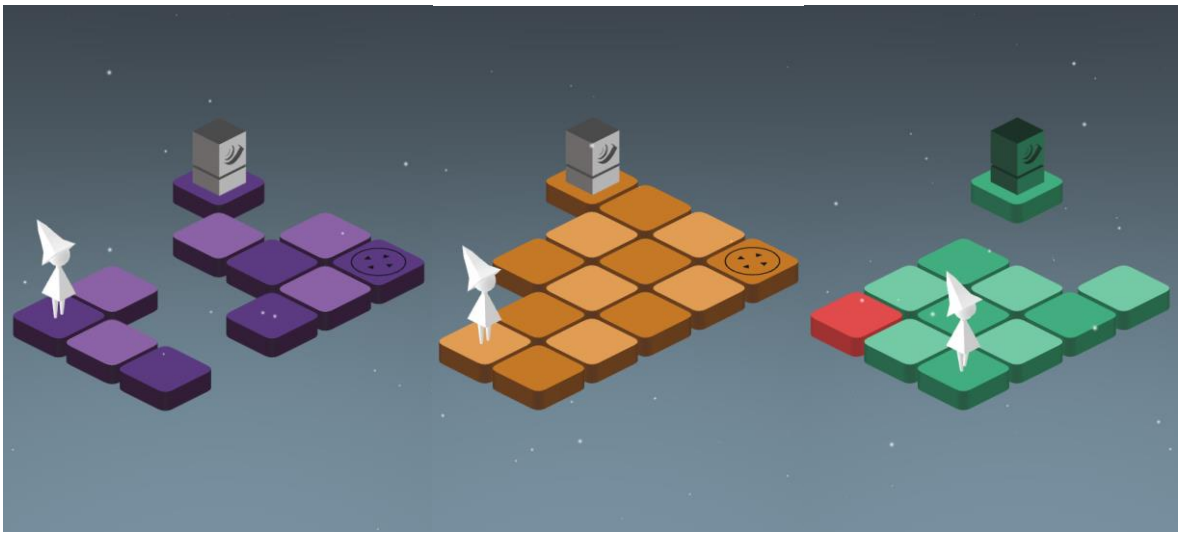


Рисунок Г.5 - Приклад згенерованого рівня

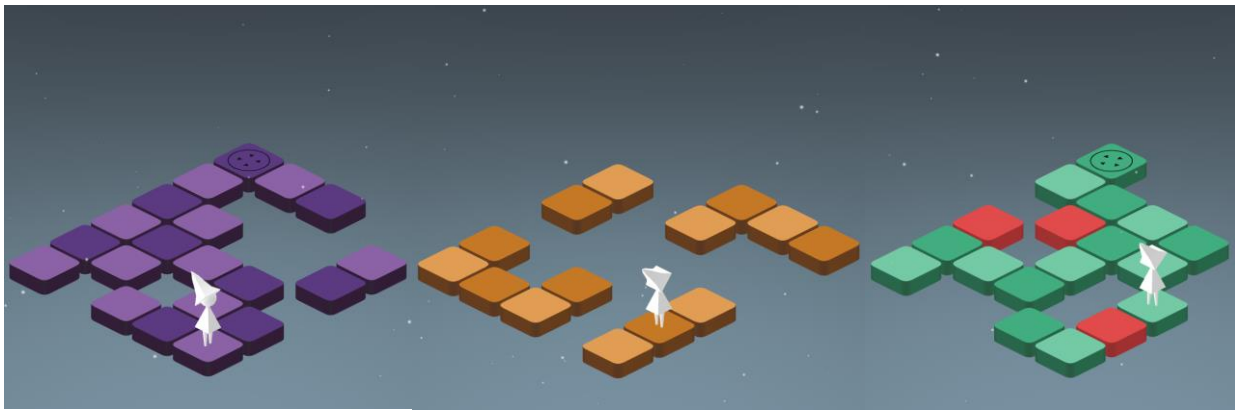


Рисунок Г.6 - Приклад згенерованого рівня

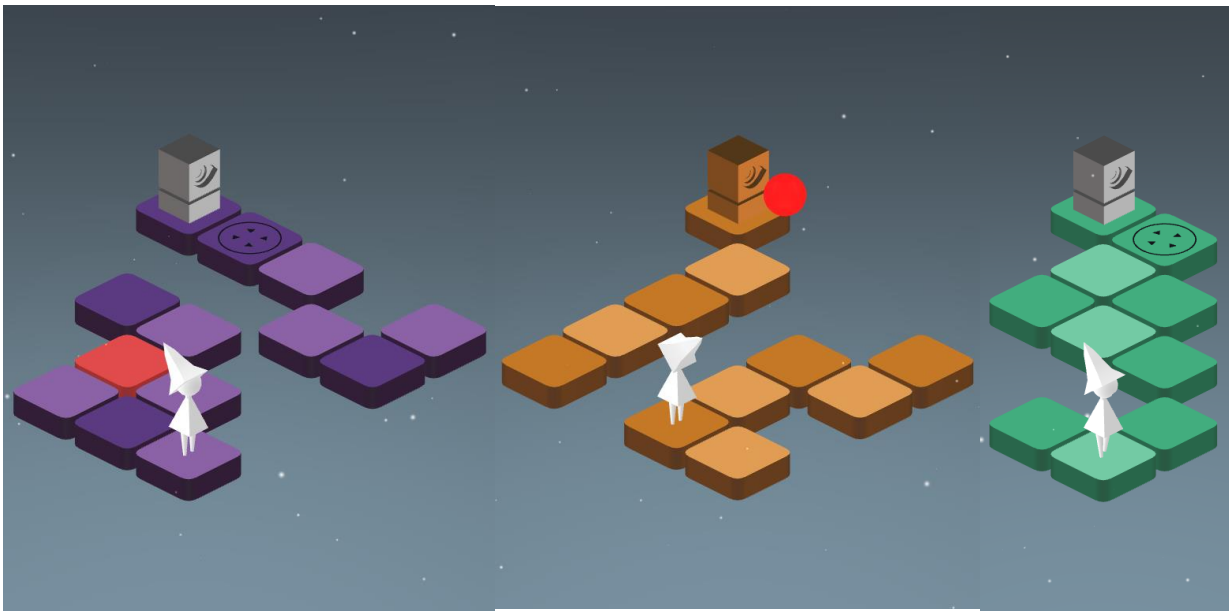


Рисунок Г.7 - Приклад згенерованого рівня

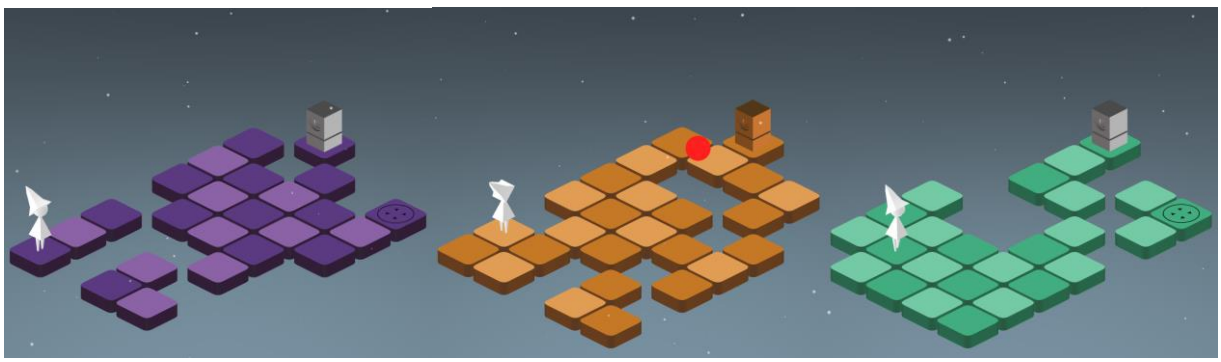


Рисунок Г.8 - Приклад згенерованого рівня

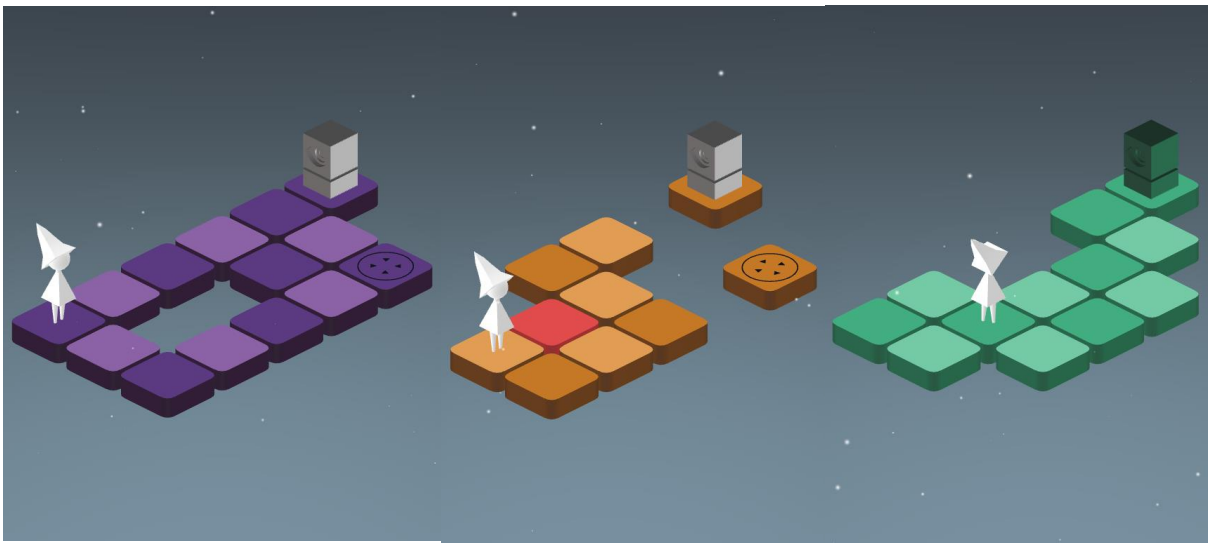


Рисунок Г.9 - Приклад згенерованого рівня

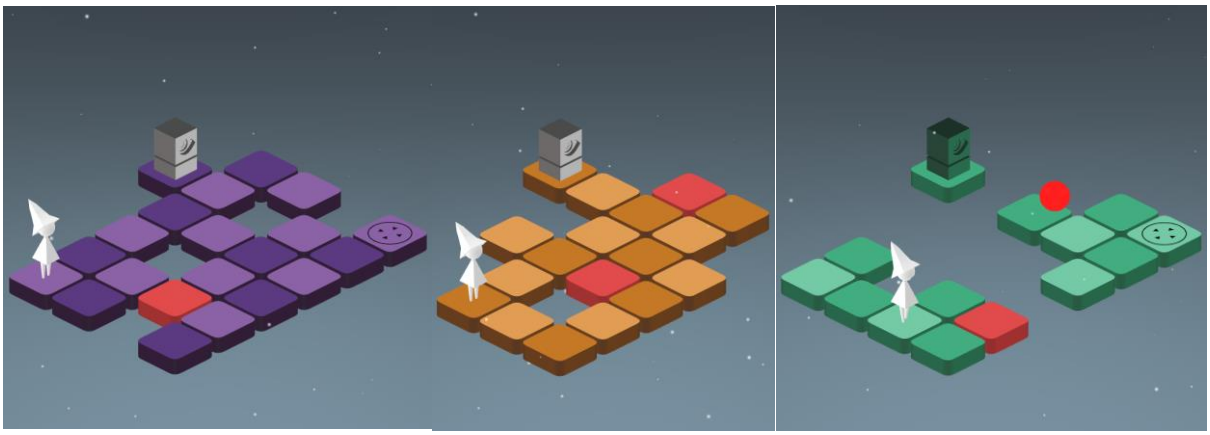


Рисунок Г.10 - Приклад згенерованого рівня

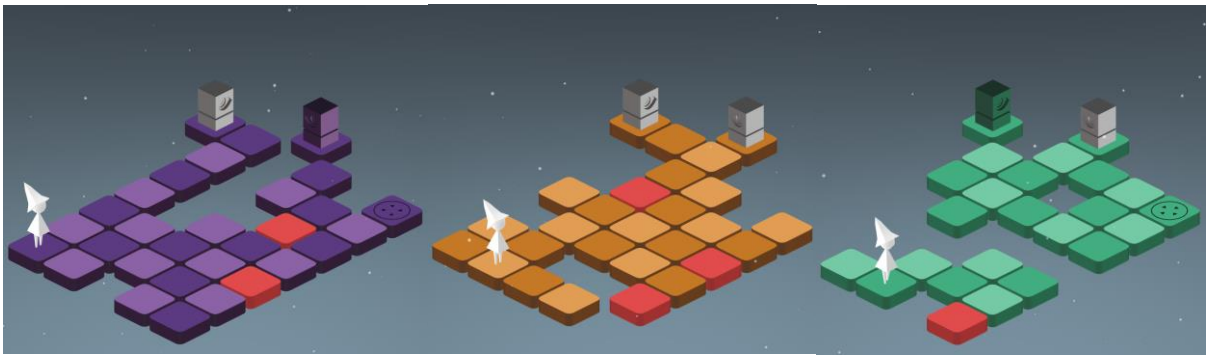


Рисунок Г.11 - Приклад згенерованого рівня

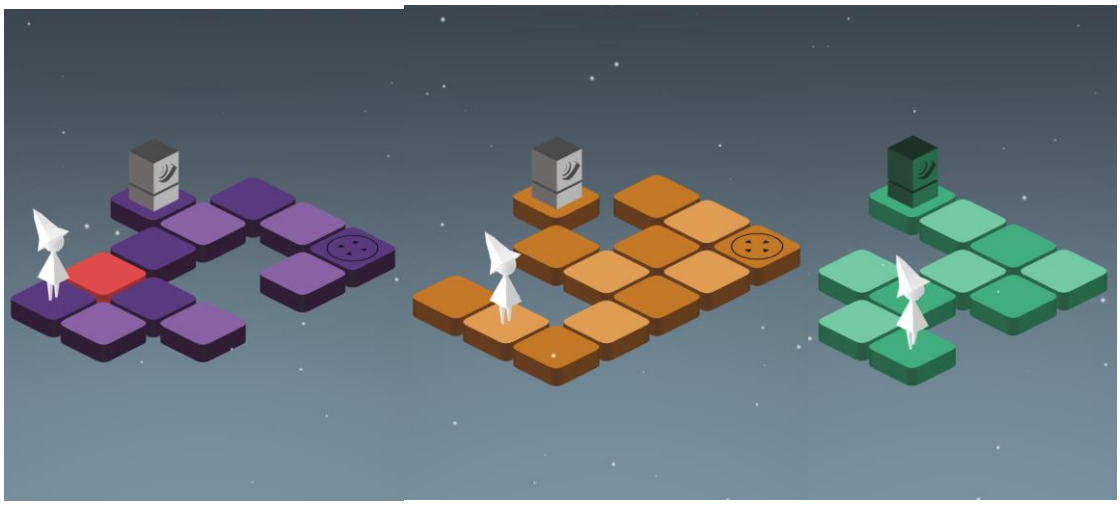


Рисунок Г.12 - Приклад згенерованого рівня

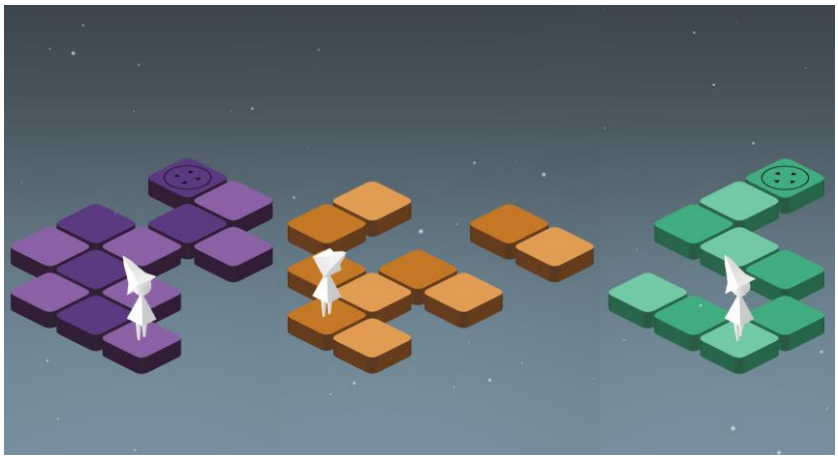
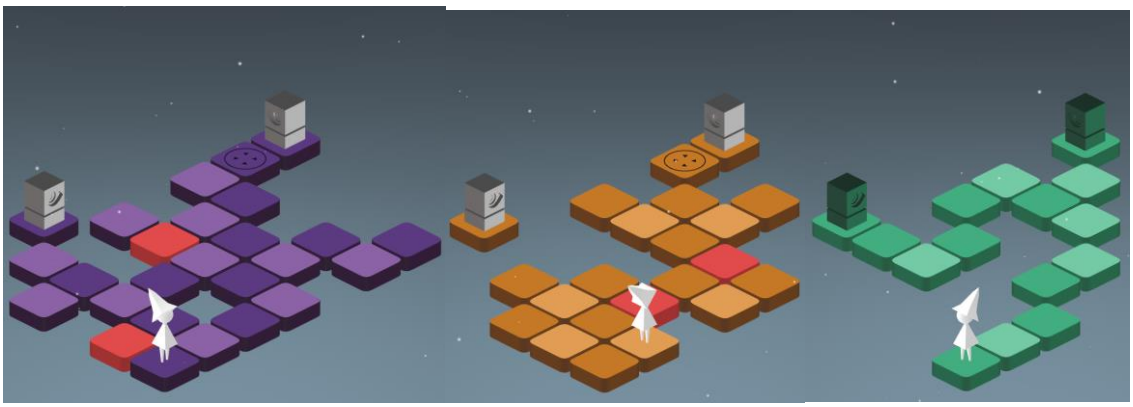


Рисунок Г.13 - Приклад згенерованого рівня



Р Рисунок Г.14 - Приклад згенерованого рівня