Міністерство освіти і науки України
Одеський національний технологічний університет
Кафедра комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему *Проектування та розробка гри жанру «Runner» з процедурною генерацією рівнів*

(назва кваліфікаційної роботи згідно наказу ОНТУ)

Здобувача *Кулакова В. А.*
(прізвище, ініціали)

4 курсу *542а* групи

Керівники: *ст. викл. Жуковецька С.Л.*
(посада, прізвище та ініціали)

к.т.н.,доц. Шестопалов С.В.
(посада, прізвище та ініціали)

Консультанти:
(посада, прізвище та ініціали)

д.е.н., проф. Басюркіна Н.Й.
(посада, прізвище та ініціали)

Кваліфікаційна робота допускається до захисту

Рішення кафедри від *10.06* 2023 р., протокол № 8

Завідувач кафедри комп. інженерії _____ Сергій АРТЕМЕНКО
(назва кафедри) (підпис) (Ім'я ПРІЗВИЩЕ)

Одеса - 2023 рік

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерної інженерії, програмування та кіберзахисту*
Кафедра *комп'ютерної інженерії*
Ступінь вищої освіти *бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*
Освітня програма *Розробка ігор та інтерактивних медіа у віртуальній реальності*

ЗАТВЕРДЖУЮ
Зав. кафедри *комп'ютерної інженерії*
Сергій АРТЕМЕНКО
« 10 » *серпня* 2022 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Кулакова Валентина Аліковича

1. Тема роботи *Проектування та розробка гри жанру «Runner» з процедурною генерацією рівнів*

Затверджена наказом університету від « *10* » *серпня* 2022 р., наказ № *440-03*

2. Термін здачі здобувачем закінченої роботи *5 червня 2023 р.*

3. Вихідні дані роботи

1. Ідея гри. 2. Редактор «Microsoft Office». 3. Середовище розробки «Visual Studio 2022», 4. Ігровий двигун «Unity 3D». 5. Ассети гри. 6. Графічний редактор «Aseprite»

4. Перелік питань, які потрібно розробити

1. Аналіз предметної області та дослідження існуючих аналогів. 2. Проектна документація. 3. Розробка гри. 4. Економічні розрахунки проекту. 5. Охорона праці та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайд 2. Мета, об'єкт, предмет, задачі та наукова новизна. Слайд 5. Існуючі аналоги. Слайд 6. Існуючі аналоги. Слайд 9. Жанр та аудиторія. Слайд 10. Основні особливості гри. Слайд 17. Алгоритм роботи процедруного генератора. Слайд 21. Системні вимоги. Слайд 23. Основна сцена гри. Слайд 28. Процедурний генератор в дії.

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Економіка	*Басюркіна Н.Й., д.е.н., проф.*		
Охорона праці	*Шестопалов С.В., к.т.н., доц.*		
Нормоконтроль	*Жуковецька С.Л., ст. викл.*		

7. Дата видачі завдання *30.09.2022*

Керівники _____ *Світлана ЖУКОВЕЦЬКА*

_____ *Сергій ШЕСТОПАЛОВ*

Завдання прийняв до виконання _____ *Валентин КУЛАКОВ*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз предметної області*	28.11.2022	
2.	*Аналіз існуючих аналогів*	28.12.2022	
3.	*Розробка концептуального документу*	16.01.2023	
4.	*Розробка дизайнерського документу*	28.02.2023	
5.	*Розробка графічного контенту*	15.03.2023	
6.	*Розробка демонстраційної версії гри*	15.05.2023	
7.	*Розробка економічної частини*	25.05.2023	
8.	*Розробка розділу охорони праці*	29.05.2023	
9.	*Оформлення пояснювальної записки*	02.06.2023	

Керівники роботи _____ *Світлана ЖУКОВЕЦЬКА*

_____ *Сергій ШЕСТОПАЛОВ*

Несу відповідальність за ідентичність електронного та друкованого варіантів кваліфікаційної роботи, даю згоду на обробку персональних даних та не заперечую проти розміщення кваліфікаційної роботи на офіційних web-ресурсах ОНТУ.

Підтверджую, що в кваліфікаційній роботі відсутні порушення норм академічної доброчесності.

Здобувач - дипломник _____ *Валентин КУЛАКОВ*

АНОТАЦІЯ

Дипломна робота присвячена розробці гри жанру «*Runner*» з процедурною генерацією рівнів.

В першому розділі було досліджено історію розвитку ігор жанру «*Runner*», описано підходи до процедурної генерації, досліджено сучасні аналоги та розроблено завдання.

В другому розділі було створено концептуальний та дизайнерські документи розроблюваної гри.

У третьому розділі описано процес розробки головного героя, основних ігрових механік та реалізацію процедурного генератора рівнів.

Четвертий розділ обґрунтовує науково-технічну ефективність гри.

В п’ятому розділі описуються правила безпеки роботи за комп’ютером та алгоритм дії при надзвичайних ситуаціях.

Результатом роботи є створена гра на платформі *Android*.

Ключові слова: процедурна генерація, *Unity, Visual Studio, Runner*

ABSTRACT

The thesis is devoted to the development of a "Runner" game with procedural level generation.

In the first chapter, the history of the development of "Runner" games was studied, approaches to procedural generation were described, modern analogs were studied, and tasks were developed.

In the second section, we created the conceptual and design documents of the game under development.

The third section describes the process of developing the main character, the main game mechanics, and the implementation of the procedural level generator.

The fourth section justifies the scientific and technical effectiveness of the game.

The fifth section describes the rules of computer safety and the algorithm of action in case of emergencies.

The result of the work is the created game on the Android platform.

***Keywords**: procedural generation, Unity, Visual Studio, Runner*

.

ЗМІСТ

Стор.

ВСТУП

На протязі всього свого існування люди вигадували різні ігри для розваги та навчання, а за останні десятиліття їх кількість та розмаїття стало дуже великим. Комп'ютерні ігри вже стали частиною нашого життя: від простих ігор як «*Pong*»[1], яка має мінімальну графіку, до ігор з високим рівнем деталізації та глибоким світом, повних різних видів діяльності.

Комп'ютерні ігри створюються на основі фільмів, книг, або ігор в реальному світі. Частина ігор створена з нуля із фантазії людини – геймдизайнера. Саме він проектує ігровий процес, загальний стиль та механіки гри. Ігри вважають видом мистецтва з 2011 року. На сьогоднішній день ігри доступні майже всюди: телефони, комп'ютери, консолі, портативні приставки. Наприклад, гра «*DOOM*»[2] запускалася майже всюди та на всьому, її запускали навіть на тесті для вагітності!

Частина людей, що любить грати в ігри, проводить багато часу за комп'ютером або на дивані, граючи на консолі, проте велика частина геймерів віддає перевагу грі на смартфоні. Причина цьому проста – смартфон портативний, що дозволяє грати в ігри майже в будь-якому місці: в автобусі, машині, на прогулянці чи в черзі, в очікуванні, тощо. Один з найпопулярніших жанрів мобільних ігор, що обирають багато людей для проведення часу є «*Runner»*. Є декілька його видів (піджанрів): кінцеві, нескінченні, музичні, двовимірні, тривимірні. Нескінченні раннери є популярніші, ніж кінцеві, так як вони дають гравцеві виклик пробігти якнайдалі та, як правило, мають процедурну генерацію рівнів. Найпопулярнішими іграми жанру «*Runner*» є: «*Subway Surfers*» від *Kiloo*»[3], «*Jetpack Joyride*» від *HalfBrick Studios*[4], «*Temple Run*» від *Imangi Studios*[5]. Зважаючи на популярність приведених ігор зокрема та жанру «*Runner»* взагалі тема кваліфікаційної роботи, яка полягає в проектуванні та розробці гри жанру «*Runner*» з процедурною генерацією рівнів є актуальною.

Зазначимо мету, об’єкт, предмет та задачі, які необхідно вирішити в ході кваліфікаційної роботи.

Метою роботи є проектування і розробка гри жанру «*Runner*» з процедурною генерацією рівня.

Об’єктом дослідження є процеси проектування і розробки гри жанру «*Runner*» з процедурною генерацією рівнів.

Предметом дослідження є методи проектування і розробки гри жанру «*Runner*» з процедурною генерацією рівнів.

Основними задачами, які необхідно вирішити в ході роботи є:

1. Аналіз предметної області та існуючих аналогів.
2. Дослідження існуючих методів процедурної генерації.
3. Розробка концептуального документа.
4. Розробка дизайнерського документа.
5. Розробка демонстраційної версії гри.

Наукова новизна кваліфікаційної роботи полягає в удосконаленні існуючих методів процедурної генерації рівнів в іграх жанру «*Runner*».

Публікації за темою кваліфікаційної роботи:

1. Кулаков В.А. Використання процедурної генерації при розробці контенту комп’ютерних ігор/ В.А. Кулаков, С.В. Шестопалов// XXIII Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса, 20-21 квітня 2023р. – Одеса, Видавництво ОНТУ, 2023 р. – С. 417-419.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ДОСЛІДЖЕННЯ ІСНУЮЧИХ АНАЛОГІВ

1.1 Аналіз історії та розвитку жанра «*Runner*»

«*Runner*» – це піджанр жанру «*Platformer*», у якому гравець без зупинки біжить через різні перешкоди на шляху до кінцевої мети. Цей жанр може мати різні варіації вигляду гри: з видом збоку, з видом зверху вниз або тривимірний, і зазвичай має спрощену фізику об'єктів, що дає більше контролю гравцю та відчуття драйву [6].

Однією з головних механік ігор жанру «*Runner*» є біг без зупинки та ухилення від різноманітних перешкод. Деякі ігри можуть мати механіку поступового збільшення швидкості, що ускладнює проходження рівнів з часом. Наприклад, у грі «*Temple Run*» персонаж поступово збільшує свою швидкість під час проходження рівнів. Однак не всі ігри жанру «*Runner*» мають такі механіки.

Історія ігор жанру «*Runner*» починається з 1980-х років. Одним з перших прикладів гри цього жанру є «*B.C.'s Quest for Tires*»[7], що вийшла у 1983 році. Гра примусово переміщує екран направо, змушуючи героя ухилятися від перешкод, проте він може пересуватися вперед та назад в межах ігрового екрану. Переміщення гравця, що відбувається безперервно вперед, стало визначальною механікою ігор жанру «*Runner*».

Після «*B.C.'s Quest for Tires*» довгий час спостерігалася стагнація жанру, поки у 2009 році Адам Сольтсмен не випустив гру під назвою «*Canabalt*»[8], що стала проривом у жанрі «*Runner*». Основна механіка гри – нескінченна генерація рівнів, тобто, рівень генерується під час гри, що дає гравцеві постійний виклик його рефлексам та вмінню протистояти складнішим викликам. Гра поступово набирає швидкість, що ускладнює проходження. Гра отримала велику кількість

позитивних відгуків від критиків та гравців, через що стала однією із найпопулярніших ігор жанру «*Runner*». Саме її успіх стимулював розвиток подібних ігор з процедурною генерацією, що стали більш поширеними у наступні роки.

Наступні ігри у жанрі «*Runner*» покращували основні механіки гри «*Canabalt*» та вносили нові особливості. Один з яскравих прикладів цього – гра «*Temple Run*», що була випущена компанією *Imangi Studios* у 2011 році. Гра взяла основні механіки з «*Canabalt*» та перенесла гравця у тривимірний світ, додавши нові можливості. Наприклад, окрім стрибка та підкату, гра додала можливість пересуватися горизонтально по доріжці, а перешкоди вимагають певної позиції на доріжці. Гравці можуть збирати ігрову валюту – монети, які можна було використовувати для покупки різних речей, наприклад, одноразових бонусів. Окрім цього, гра ввела систему мікротранзакцій[9], що дозволяло отримати ігрові речі за реальну валюту.

Після виходу гри «*Temple Run*», наступним хітом жанру стала гра від студії *HalfBrick Studios* – «*Jetpack Joyride»*, яка з'явилася на мобільних пристроях того ж року. Гра змінила спосіб уникання перешкод з горизонтального ухиляння на землі в вертикальне ухиляння у повітрі. Окрім цього, вона привнесла багато нових механік, таких як: завдання, систему рівнів, унікальні жетони, які гравець міг використати для гри у однорукого бандита для отримання додаткових нагород чи продовження гри з місця програшу. Постійні оновлення та події у грі продовжують залучати гравців і донині.

Звичайність та простота «*Jetpack Joyride»* та «*Temple Run»*, а також їхні унікальні функції та механіки, спричинили значний успіх серед гравців різних вікових груп та допомогли популяризувати жанр «*Runner*», до якого все більше приверталася увага геймерів та розробників. Відтак, «*Subway Surfers»* з'явився на світ від відомої студії *Kiloo* в 2012 році та став надзвичайно популярним, завоювавши серця мільйонів гравців. На відміну від «*Temple Run*», де необхідно нахиляти смартфон для горизонтального переміщення, «*Subway Surfers*» використовує три колії, по яким можливо пересуватися, а для зміни лінії

достатньо провести екраном палець в необхідний бік – лівий чи правий. Також, нововведенням для гри стала поява лутбоксів – віртуальних скриньок, з яких можна отримати випадковий предмет. У грі можна отримати ігрову валюту чи особливі предмети, що необхідні для розблокування персонажів, а якщо необхідна річ потрібна якнайшвидше гра має систему мікротранзакцій [9]. Як і у випадку з «*Jetpack Joyride*» гра постійно оновлюється та змінює зовнішній вигляд метрополітену, додає нових персонажів та додає нові можливості, що утримує поточних гравців та залучає нових.

Як і «*Temple Run»*, «*Subway Surfers»* також має елементи покращення, які додають гравцям додаткову мотивацію та можливість побити свій власний рекорд. Такі покращення включають збільшення множника балів, магніт, який дозволяє збирати монети, взуття, що дозволяє стрибати вище та реактивний ранець, що дозволяє літати. Також гра має різноманітних персонажів, яких можна отримати або за монети або за ключі або за особливі предмети.

На сьогодні жанр «*Runner*» продовжує залишатися популярним серед гравців і розробників. Багато ігор жанру «*Runner*» виходять на різних платформах, включаючи мобільні пристрої, персональні комп'ютери (ПК) та консолі. Ігри цього жанру можуть бути як простими тайм-кіллерами, так і складними іграми з детально розробленим ігровим процесом та сюжетом.

Однак, з часом жанр «*Runner*» почав еволюціонувати і змінюватися, і з'явилися нові напрямки в рамках цього жанру. Наприклад, ігри з елементами платформера, різноманітні режими гри, такі як змагання з іншими гравцями або створення власних рівнів та персонажів, і т.д.

Загалом, жанр «*Runner*» продовжує залишатися важливою частиною ігрової індустрії, з багатим різноманітним вибором ігор для гравців різного рівня досвіду та інтересів.

Інтерес до ігор жанру «*Runner*» суттєво зростає при використанні процедурної генерації рівні завдяки ускладненню гри та реіграбельності.

1.2 Можливості, застосування та обмеження процедурної генерації

Звичайне створення великого ігрового світу може бути складним завданням для розробників ігор. Тому, з метою полегшення цього завдання та розширення можливостей ігрового процесу, розробники використовують процедурну генерацію.

Процедурна генерація – це метод створення даних алгоритмічним способом, що здійснюється за допомогою комп'ютерних обчислювальних ресурсів [10]. Завдяки цьому методу можна створювати великі та деталізовані світи з обмеженим внеском ручної роботи.

Процедурна генерація допомагає розробникам ігор створювати складні механіки та збільшувати ігровий процес, заощаджуючи значні вкладення в ігровий світ. Наприклад, багато ігор з відкритим світом або жанру «*Survival*»[11] створюють світи процедурним методом за допомогою насіння (*seed*)[12], що дозволяє гравцям досліджувати нові території кожного разу, коли вони грають. Один з яскравих прикладів гри, що використовує даний принцип, – «*Minecraft*», розроблена студією *Mojang Studios*[13]. У «*Minecraft*» гравці можуть досліджувати нескінченний випадково згенерований світ, заселений різноманітними тваринами та істотами, різними типами блоків, об'єктами та різноманітними природними ресурсами. Використання процедурної генерації дозволяє гравцям отримувати новий, неповторний ігровий досвід кожного разу, коли вони грають. Це робить гру однією з найпопулярніших ігор в жанрі «*Survival*».

Крім того, процедурна генерація допомагає в створенні різноманітних ігрових об'єктів, таких як зброя, предмети, монстри, і т.д. Наприклад, гра «*Borderlands*»[14] базується на процедурній генерації, що дозволяє створювати понад мільйон унікальної зброї та елементів спорядження, в той час як «*Spore*»[15] за допомогою процедурної генерації створила велику кількість монстрів.

Процедурна генерація допомагає розв'язувати проблему оригінальності ігрового досвіду, що може стати фактором успіху гри. Розробники таких ігор можуть бути впевнені в тому, що гравці не зіткнуться з повторюваними рівнями чи іншими елементами гри, що забезпечує більш довгий та цікавий ігровий процес. Яскравим прикладом цього є гра «*The Binding of Isaac: Rebirth*»[16], де кожна спроба пройти гру унікальна, а досвід гри залежить від майстерності гравця.

Процедурна генерація також може допомогти зменшити витрати на розробку гри, оскільки розробники можуть створювати більш об'ємні ігри з меншими витратами на створення вмісту. Так зробили розробники гри «*Just Cause*»[17] від студії *Avalanche*, використавши процедурну генерацію для створення тропічних островів, що допомогло зекономити кошти та час працівників. Проте, важливо пам'ятати, що процедурна генерація не гарантує високої якості вмісту гри, тому розробники повинні перевіряти результат під час використання цього методу.

Додатково, процедурна генерація може також допомогти в розширенні можливостей ігрового процесу, зокрема у генерації завдань. Розробники можуть використовувати процедурну генерацію для створення випадкових завдань, які будуть виконувати гравці. Такий підхід забезпечує більшу різноманітність ігрового досвіду та високий рівень реіграбельності. Таким принципом створюються завдання у грі «*No Man's Sky*»[18], яка використовує процедурну генерацію для створення унікальних місій і цілей, які гравці повинні виконати, додаючи грі можливості повторного відтворення та відчуття відкриття.

Крім того, процедурна генерація знайшла своє застосування не тільки в ігровій індустрії, але й в інших галузях, таких як віртуальна реальність, комп'ютерна графіка, генерація текстів та інші. Наприклад, у віртуальній реальності процедурна генерація може допомогти створити реалістичні та іммерсивні світи, які можуть бути досліджені користувачами. В комп'ютерній графіці процедурна генерація може допомогти створити великі та деталізовані сцени, які можуть бути використані в кіно та інших медіа-продуктах.

Узагалі, процедурна генерація є важливим інструментом для розробників ігор та інших галузей, оскільки вона дозволяє створювати більш об'ємні та різноманітні продукти з меншими витратами на створення контенту. Також вона забезпечує більшу неповторність ігрового досвіду та високу реіграбельність, що є важливим фактором для багатьох ігор. Однак, використання процедурної генерації також має свої обмеження та ризики, тому розробники повинні ретельно планувати та контролювати процес її використання. Недостатня контрольованість може призвести до створення неякісного та нудного контенту, що в свою чергу може призвести до низької оцінки гри користувачами та негативних відгуків. Наприклад, помилками у генерації світу є в «*Minecraft*», де більш ранні версії гри могли створити літаючі острови.

Загалом, процедурна генерація є корисним та ефективним інструментом для створення ігрового контенту, але вона повинна використовуватися розумно та з обережністю. Розробники повинні забезпечувати достатній контроль над процесом генерації, щоб забезпечити якість та неповторність ігрового досвіду, а також збалансувати процедурну генерацію з ручною роботою та внесенням унікальних елементів до гри.

1.3 Порівняння існуючих методів процедурної генерації

Процедурна генерація стала дуже популярним інструментом для створення відеоігор та інших комп'ютерних додатків. Розробники використовують різні підходи до процедурної генерації, включаючи імітаційний, функціональний та планувальний. Кожен з цих підходів має свої переваги та недоліки, і може бути використаний для різних завдань.

Імітаційна процедурна генерація заснована на математичному моделюванні, що створює випадкові результати, які схожі на природні об'єкти або процеси. Цей підхід дозволяє створювати велику кількість варіативних та складних об'єктів [19]. Наприклад, гра «*Spore*» використовує імітаційну процедурну генерацію для створення різноманітних істот, які гравець може

створювати, еволюціонувати та керувати(рис. 1.1). Цей підхід також використовується для генерації ландшафту, текстур та інших елементів відкритого світу.

Рис. 1.1 – Зображення створінь, згенерованих імітаційним способом у грі «*Spore*»

Функціональна процедурна генерація заснована на використанні математичних функцій для створення об'єктів. Цей підхід дозволяє створювати варіативні та складні об'єкти, але з більшою точністю та контролем над створеними об'єктами [19]. Наприклад, гра «*Minecraft*» використовує функціональну процедурна генерацію для створення світу з блоків, які можуть бути розміщені у будь-якому порядку та можуть мати різні властивості, такі як твердість, текстура та інші. Один з прикладів генерації світу можна побачити на рисунку 1.2.

Рис. 1.2 – Приклад генерації світу у грі «*Minecraft*»

Планувальна процедурна генерація використовує алгоритми для генерації об'єктів, в яких розглядаються правила та обмеження, що дозволяють створювати більш складні та варіативні об'єкти [19]. Цей підхід зазвичай використовується для створення сценаріїв, історій, завдань та інших складних елементів відеоігор. Прикладом використання планувального підходу може бути гра «*The Sims*»[20], де генерація персонажів та їхніх поведінкових моделей відбувається на основі правил та обмежень, таких як особистість, потреби та взаємодії з іншими персонажами.

Крім того, планувальний підхід може бути використаний для генерації випадкових карт в стратегічних іграх, де карту створюють на основі обмежень, таких як розташування ресурсів та географічні особливості. Прикладом може бути гра «*Civilization*»[21], де карта генерується на основі таких факторів, як розташування річок та гірських хребтів, що впливає на стратегію гравця.

Комбінації різних підходів до процедурної генерації можуть додатково підвищити рівень варіативності та складності генерованих об'єктів. Наприклад, використання функціонального та планувального підходу в грі «*Diablo III*»[22] дозволяє створювати випадково згенеровані локації, які одночасно враховують обмеження на основі правил та забезпечують високий рівень випадковості та варіативності(рис. 1.3).

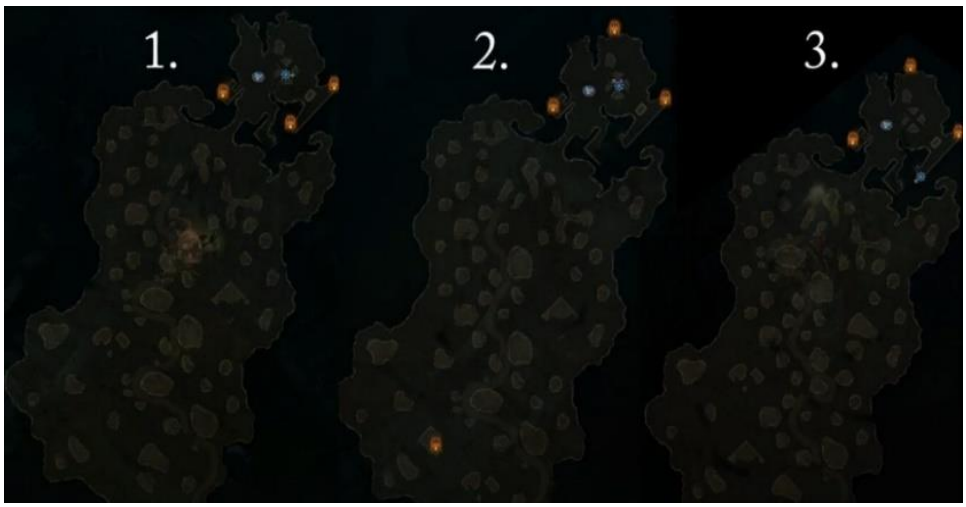


Рис. 1.3 – Приклад генерації рівня у грі «*Diablo III*»

В цілому, використання різних видів процедурної генерації відкриває широкі можливості для створення високоякісного та варіативного контенту у відеоіграх. Комбінація різних підходів може дати найкращий результат, що дозволяє створювати ще більш складні та цікаві об'єкти та світи. Наприклад, гра «*Spore*» використовує як імітаційний, так і функціональний підходи для генерації різних видів істот та елементів у світі гри. Гравці можуть створювати власних істот, виходячи з різних критеріїв та правил, які можуть бути задані функціональним підходом. Також у грі присутні готові види істот, що були згенеровані за допомогою імітаційного підходу.

Ще один приклад комбінації різних підходів до процедурної генерація може бути гра «*Minecraft*». У грі присутні різноманітні блоки та предмети, які генеруються функціональним підходом. Також у грі є різні біоми, які створюються за допомогою імітаційного підходу, враховуючи різні фактори, такі як клімат та географія. У цьому випадку планувальний підхід використовується для створення варіативності в мінеральних ресурсах та розміщення різноманітних структур у світі гри.

Унікальні можливості кожного підходу процедурної генерації дозволяють розширювати можливості ігор та забезпечувати гравців більш складним та цікавим вмістом.

Комбінування різних підходів процедурної генерації може бути особливо корисним в грі з великою кількістю різноманітного вмісту, оскільки це дозволяє створювати більш варіативний та неочікуваний геймплей.

Наприклад, у грі «*Diablo III*» використовуються як імітаційні методи, так і функціональні, щоб створювати різні варіації випадкових елементів гри, таких як монстри, локації та предмети. Крім того, планувальний підхід використовується для створення складніших сценаріїв для місій та завдань. Комбінування цих підходів може дозволити генерувати більш складні та реалістичні світи у відеоіграх, які привернуть до себе увагу гравців.

Враховуючи вищесказане, різні види процедурної генерації мають свої особливості та переваги, які можуть бути використані для створення унікального

та захоплюючого вмісту у відеоіграх. Комбінація різних підходів дозволяє створювати більш варіативний та складний вміст, який може бути цікавим та захоплюючим для гравців. Продуктивне використання процедурної генерації відкриває нові можливості для ігрової індустрії, і сприяє створенню більш реалістичних та захоплюючих ігрових світів.

1.4 Існуючі аналоги

1.4.1 *Minion Rush: Running Game*

Гра «*Minion Rush: Running Game*» – це представник жанру «*Runner*», який був випущений компанією *Gameloft* у 2013 році на мобільні пристрої. У грі головним персонажем є міньйон з мультфільму «Нікчемний я»[23]. Основна механіка гри полягає в бігу по локаціях та доланні процедурно генерованих перешкод. Для генерації рівнів використовується великий обсяг заготівок, що розташовуються на трьох доріжках, що робить їх різноманітними.

Гра має кілька режимів, включаючи нескінченний, де гравцеві потрібно зібрати якнайбільше бананів на своєму шляху. Під час цього режиму, є можливість підвищити цінність бананів, збираючи особливий предмет. Гра також має інші режими з завданнями, такими як збирання бананів, штовхання інших міньйонів або використання певного посилення протягом певного часу. Гра має систему рівнів, що додає відчуття прогресу. З його підвищенням, на 5 та 15 рівнях розблоковуються нові режими з більш складними завданнями.

Кожен костюм на міньйонів має свою рідкість та здібність, що потрібно заряджати збором бананів. Здібність – це довша версія посилення, що може дати перевагу у грі. Кожен костюм може бути розблокованим чи покращеним при умові, що у гравця є достатня кількість карточок відповідного костюму. Для отримання карточок гравці повинні відчиняти лутбокси[24], які можна отримати купивши за реальні кошти або проходячи режими з завданнями.

Гра також містить різноманітні посилення, такі як «*Fluffy Unicorn*», що запускає міньйона з єдинорогом вгору, де можна зібрати багато бананів, а при закінченні міньйон повертається до дороги. Також є посилення «*Banana Splitter*», що роздвоює банани на рівні.

Скріншот ігрового процесу показаний на рисунку 1.4

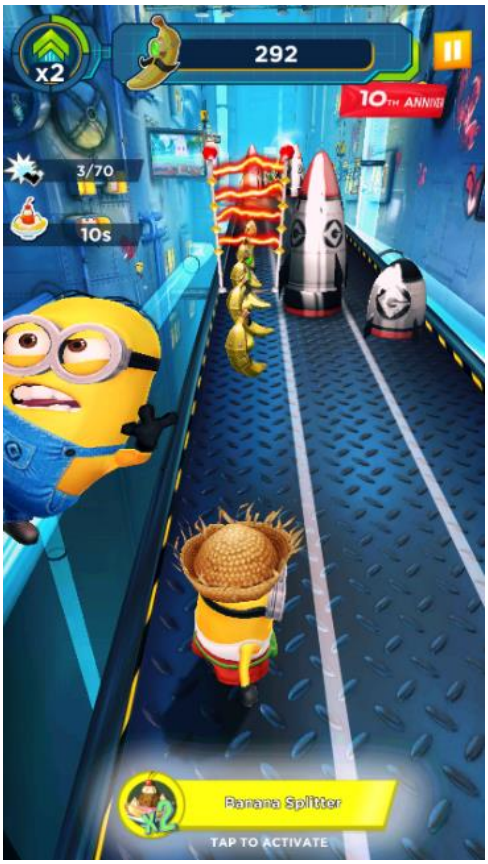


Рис. 1.4 – Геймплей гри «*Minion Rush*»

Загалом, «*Minion Rush: Running Game*» є досить простою грою, яка приваблює гравців своїм милозвучним головним героєм та кольоровою графікою. Гра має велику кількість заготівок для генерації рівнів, що робить їх різноманітними, а також різні режими гри з різними завданнями. Присутність різноманітних костюмів та посилень додає грі глибини та можливості для розвитку героїв. Однак, недоліком гри може стати необхідність витрачати багато часу на розблокування рідкісних костюмів та отримання карточок, що може зменшувати задоволення від гри. Також негативним фактором є велика кількість вікон, які з’являються час від часу, що може дратувати гравців.

1.4.2 *Cookie Run: OvenBreak*

Гра «*Cookie Run: OvenBreak*» була розроблена компанією *Devsisters* і випущена в жовтні 2016 року. Вона стала першою грою, що вийшла після того, як вони видавали «*LINE Cookie Run*» у 2014 році[25]. Незважаючи на те, що «*Cookie Run: OvenBreak*» містить більшість атрибутів попередньої гри «*Cookie Run*», таких як аркадна гра з бічною прокруткою та більшу частину графіки, вона відрізняється новою системою підрахунку балів і таблицею лідерів, а також представляє нових персонажів, домашніх тварин та скарби. Гра має нові механіки, такі як «Ліга Чемпіонів», гільдії та режим «*Breakout*». У грі гравець керує печивом, які поділяються на 5 груп за рідкістю: звичайні, рідкі, епічні, легендарні та спеціальні.

У грі наявні підсилювачі, які спрощують проходження рівнів. Вони представлені у вигляді вірних печивних друзів, які супроводжують печиво і набирають енергію, збираючи цукерки, що знаходяться на рівнях. Як тільки енергія сягає максимального значення, підсилювач автоматично активується на певний час. Після закінчення цього періоду ефект підсилювача зникає.

У грі існують різноманітні режими гри, такі як «*Trophy Race*», «*Breakout*», «*Cookie Trials*», «*Island of Memories*» та «*Guild Runs*». У режимі «*Trophy Race*» гравець обирає двох персонажів – одного, щоб почати гру, і іншого, який буде бігти після першого. Основна мета даного режиму – зібрати якнайбільше очок за обмежений час. Крім гравця, в цьому режимі беруть участь ще дев'ять гравців. Після того, як час вичерпується, гравців розташовують у локальному рейтингу в залежності від кількості очок. Гравець, який набрав найбільше очок, отримує нагороду у вигляді десяти золотих квитків, що потрібні для відкриття особливої скрині, а також трофеї – бали рейтингу. В залежності від загальної кількості трофеїв, гравець може грати на різних локаціях, де можна отримати більше трофеїв. Чим нижче гравець у локальному рейтингу після завершення гри в режимі «*Trophy Race*», тим менше квитків він отримує, а на останніх місцях загальна кількість трофеїв зменшується. У режимі «*Breakout*» гравець може

взяти до 10 печив, які почергово вирушають після того, як попереднє печиво втратило здоров'я. Учасники отримують винагороду кожного тижня в залежності від їх досягнень в цьому проміжку. Якщо кількість балів перевищує певний поріг, гравець може отримати легендарне печиво та улюбленця. Їх також можна отримати в магазині за валюту, яку можна отримати в цьому режимі.

Режим «*Cookie Trials*» дозволяє гравцям пройти рівні з певними печивами та улюбленцями. В цьому режимі вимикаються всі покращення, щоб зрівняти гравців у змаганні. Кожне випробування має 16 рангів, які можна досягти, зібравши певну кількість балів. Кожного місяця нагороджують гравців, які досягли певного рангу. Крім того, в цьому режимі є складний режим, де завдання є більш складними та вимоги для досягнення рангів вищі. Але щоб розблокувати його, потрібно досягти рангу «Діамант» в цьому режимі.

В режимі «*Island of Memories*» є три різних острови з п'ятдесяти рівнями кожен, які відносяться до різних печив, таких як «Печиво Доктор Васабі», «Гірчичне Печиво», «Печиво-вболівальник» та «Печиво-герой». Щоб грати в цьому режимі, необхідно мати спеціальний ключ. Кожен рівень у режимі «*Island of Memories*» має свої власні завдання, виконавши які, гравець отримує зірку на цьому рівні. Завдання можуть включати в себе збір певної кількості цукерок, досягнення певної швидкості або часу, або виконання певних умов на рівні. На кожному рівні зазвичай буває 3 завдання, тому максимум можна отримати три таких зірки на кожному рівні. Крім того, при досягненні певної кількості зірок, гравець отримує нагороду, яка може бути предметом одягу або іншими корисними речами для гри.

Режим «*Guild Runs*» дозволяє гравцеві брати участь у змаганні з іншими гравцями в гільдіях. Учасники гільдії роблять внесок у формі золота та діамантів, щоб розблокувати місце в змаганні. Кожна гільдія має обмеження на кількість учасників, які можуть приймати участь у змаганні. Після того, як змагання розпочинається, гравці повинні виконати низку випробувань, щоб зібрати максимальну кількість балів для своєї гільдії. Гравці, які вийшли на перше місце у змаганні, отримують нагороди, такі як печиво та улюбленець.

Гра «*Cookie Run: OvenBreak*» використовує процедурну генерацію для створення рівнів шляхом комбінування заготовлених частин рівнів. Цей метод дозволяє створювати різноманітний геймплей для гравців. Однак, у деяких режимах гри, де здійснюється особливий захід, рівні можуть бути статичними та не залежати від процедурної генерації. Всі режими гри «*Cookie Run: OvenBreak*» мають свої відмінності та особливості, що дозволяють гравцеві насолоджуватися різноманітністю геймплею та отримувати нові враження. Завдяки цьому гравці мають можливість вибирати режим гри, який найбільше відповідає їхнім уподобанням та настрою. Проте великий ухил на покупку ігрових предметів та необхідність витрачати багато часу у грі негативно впливають на загальну оцінку гри.

Скріншот ігрового процесу показаний на рисунку 1.5

Рис. 1.5 – Геймплей гри «*Cookie Run: OvenBreak*»

1.4.3 *Crash Bandicoot: On the Run!*

«*Crash Bandicoot: On the Run!*» – це гра від компанії *King*, яка розроблена для мобільних пристроїв. Гра є невеликим втіленням знаменитої серії ігор «*Crash Bandicoot*»[26], де гравці вирушають на рятування мультивсесвіту від злого Доктора Нео та його прихильників.

У грі гравці можуть грати за двох головних героїв – Креша і Коко, які бігають по рівнях у пошуках пригод і збирають фрукти Вумпа. Гра включає в

себе кілька режимів, зокрема нескінченний режим, де гравці можуть збирати фрукти і покращувати свій рекорд пройденої відстані, а також режим бойового забігу, де гравці проходять серію рівнів і зустрічаються з міні-босами.

Головна механіка гри полягає в тому, що персонажі постійно біжать вперед і прямують до фінішу на різних рівнях, які складаються з трьох доріжок. Гравці можуть керувати своїм персонажем, переміщаючись вправо та вліво, стрибаючи та ковзаючи, а також використовуючи руйнівну атаку, щоб знищувати ящики.

Окрім збирання фруктів та руйнування ящиків, гравці також можуть збирати частини зброї, яку необхідно виготовити, щоб зустріти міні-босів на кінці кожного бойового забігу. Ігрові режими «Бойовий забіг» та «Випробування на час» забезпечують багато годин геймплею, а також можливість конкурувати з іншими гравцями в таблицях лідерів. Збирання кристалів, підвищення рівня та розвиток бази Коко забезпечують стимул для продовження гри.

Крім того, гра має багато елементів, які приваблюють шанувальників оригінальної серії «*Crash Bandicoot*». Наприклад, гра включає в себе багато елементів з попередніх ігор, таких як руйнування ящиків, збирання фруктів Вумпа, маски Аку Аку та багато іншого.

Особливо вражає процедурна генерація рівнів, яка робить кожен прохід гри унікальним і відрізняє його від попередніх. Завдяки цьому гравці завжди знайдуть нові виклики та можливості в грі.

Серед недоліків гри можна виділити необхідність довго здобувати ресурси, щоб проходити бойові забіги, а сама гра не оновлювалася вже довгий час, через що впав середній онлайн гри. Також, довгі, без можливості пропуску реклами, заважають гравцям.

Процедурна генерація гри використовує функціональний підхід, що розташовує заздалегідь підготовлені частини рівня на певній відстані від попередньої частини. Таким чином, кожен рівень створюється випадковим чином, що дає гравцеві унікальний досвід. Крім того, на рівнях є місця, де з деякою ймовірністю може з'явитися коробка з посилювачем.

Отже, «*Crash Bandicoot: On the Run*!» є цікавою та захоплюючою грою для мобільних пристроїв, яка зберігає традиційні механіки серії, але водночас має свої особливості. Гра має ряд переваг, таких як можливість грати за улюблених персонажів та різноманіття режимів, але також є й недоліки, такі як проблеми з онлайн-активністю та наявність реклами.

Скріншот ігрового процесу показаний на рисунку 1.6

Рис.. 1.6 – Геймлей гри «*Crash Bandicoot: On The Run*!»

1.4.4 *Zombie Tsunami*

«*Zombie Tsunami*» – це популярна мобільна гра в жанрі нескінченного ранера, що була розроблена французькою студією *Mobigame* та випущена на платформах *Android*[27], *iOS*[28] та *Windows Phone*[29] у 2012-2015 роках. Головною метою гравця є керування зомбі, який повинен збирати монети, їсти людей, які пересуваються на вулицях та у машинах, і обходити перешкоди та пастки, що з'являються на його шляху. Ці завдання допомагають створювати нових зомбі, які долучаються до хвилі зомбі, що полегшує проходження гри.

Щоб з'їсти людину на перешкоді, необхідно мати достатню кількість зомбі, які зможуть руйнувати перешкоду. Як тільки людина була з'їдена, вона перетворюється на зомбі, що додає нових членів до хвилі. Гра закінчиться, якщо гравець втратить всіх своїх зомбі або вони не зможуть рухатися, приховавшись за екраном.

У грі присутні різноманітні бонуси, які дають гравцям різні переваги. Наприклад, гравець може отримати зомбі зі спеціальними властивостями, які зможуть допомогти йому пройти перешкоди(рис. 1.7). Крім того, гравець може використовувати монети, які він збирає на протязі гри, для покупки різноманітних покращень, декорацій та ефектів.

Рис. 1.7 – Геймплей гри «*Zombie Tsunami*»

Процес генерації рівня у грі є процедурним та використовує функціональний підхід. Не тільки перешкоди, але й поверхні, по яких зомбі можуть рухатися, створюються за допомогою алгоритмів, що дозволяють генерувати безліч варіацій рівнів та унікальних ігрових середовищ. З часом відстань між поверхнями збільшується, що ускладнює гру, а сама поверхня зменшується у довжині, що змушує гравця приймати швидкі рішення та реагувати на зміни гри.

Однією з головних переваг гри є її простота та доступність. Гра має привабливу графіку та анімацію, які відображаються в яскравих кольорах та надають грі динаміки. Крім того, динамічна музика гри доповнює ігровий процес та додає адреналіну під час гри.

Однак, не дивлячись на багато переваг, гра має свої недоліки. Наприклад, гра може викликати нудьгу для більш досвідчених гравців, оскільки вона не

надає великої кількості складних викликів та може швидко стати одноманітною. Крім того, гра містить багато реклами, що може бути надокучливим для гравців.

У підсумку, гра «*Zombie Tsunami*» – це захоплююча та проста мобільна гра з чудовою графікою та музикою, яка дозволяє гравцеві відчути себе частинкою апокаліптичного світу. Незважаючи на деякі недоліки, гра є популярною серед широкого кола гравців, які шукають легкий та захоплюючий ігровий досвід.

1.4.5 *Super Meat Boy Forever*

«Super Meat Boy Forever» – інді-гра, що була створена командою *Team Meat* для широкого спектру платформ, таких як *Windows*[30], *Nintendo Switch*[31], *PlayStation* 4[32], *PlayStation* 5[33], *Xbox One*[34], *Xbox Series X/S*[35], *Android*, *iOS* та *Linux*[36]. Вона була спочатку задумана як мобільна версія *Super Meat Boy* у вигляді гри жагру «*Runner*» та випадковими рівнями, однак створення продовження пройшло без участі оригінального творця гри, Едмунда МакМілена, який покинув *Team Meat* багато років тому. Гра була випущена у грудні 2020 року на *Nintendo Switch* та ПК, а згодом була портована на інші зазначені платформи.

Події гри відбувається через декілька років після подій оригінальної гри, де *Meat Boy* та *Bandage Girl* живуть щасливо зі своєю дочкою *Nugget*. Одного дня *Dr. Fetus* раптово з'являється, нападає на них та вони втрачають свідомість. Коли вони прокидаються, то не можуть знайти свою дитину, тому вирушають на пошуки викрадача.

«Super Meat Boy Forever» розширює складний ігровий процес та сюжетну лінію оригінальної гри «*Super Meat Boy»*. Головною механікою гри є автоматичний біг, що вимагає від гравця негайних дій. Крім того, гравець може стрибати, стрибати від стін та робити ривки. Гра розділена на 4 світи, а кожен світ поділений на 7 рівнів, з яких 6 є звичайними, а один – битва з босом. У грі є точки збереження, які дозволяють гравцеві з'являтися на цьому місці після програшу та зберегти прогрес. Рівні в грі генеруються процедурно з

підготовлених дизайнерами частин рівня, що дає можливість гравцеві проходити цікаві та різноманітні рівні. Крім того, гра пропонує гравцеві вибір персонажів. Спочатку доступні лише два персонажі: *Meat Boy* та *Bandage Girl*. Щоб розблокувати інших персонажів, гравець повинен виконати певні досягнення або зібрати дитячі со́ски, які розкинуті по рівнях.

Основна механіка гри полягає в автоматичному бігу, що вимагає від гравця негайних дій. Крім бігу, у гравця є можливість стрибати, стрибати від стін та робити ривок(рис. 1.8). Рівень боса представляє собою арену, де необхідно перемогти боса, при цьому основні механіки гри зберігаються. Кожен світ додає до минулого світу додаткові механіки, що розширює та ускладнює існуючий ігровий процес.

З головних переваг гри можна виділити її різноманітний та захопливий ігровий процес, а також якісний візуальний дизайн. Сюжет гри також є цікавим та захоплюючим, а бої з босами надають грі особливого колориту. Проте, варто зазначити, що висока складність та невелика кількість рівнів роблять гру більш нішевою та скерованою на вузьку групу фанатів першої частини

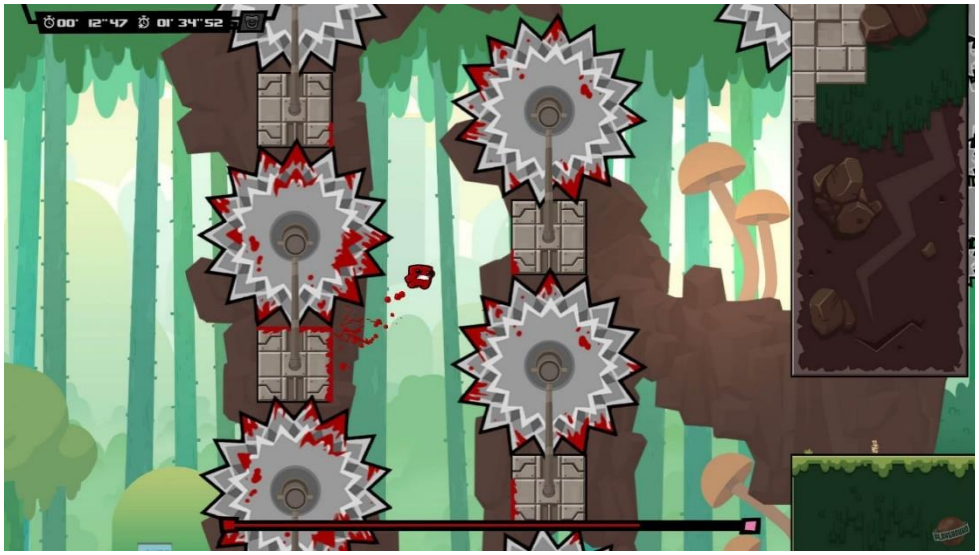


Рис. 1.8 – Геймплей гри «*Super Meat Boy Forever*»

1.4.6 *Super Mario Run*

«*Super Mario Run*» – це мобільна гра, розроблена та випущена компанією *Nintendo* для *iOS* та *Android* в грудні 2016 року. Гра була однією з найочікуваніших на мобільному ринку, оскільки це перший офіційний випуск іконічного персонажа *Mario* на мобільному пристрої. Вона стала однією з найпопулярніших мобільних ігор в 2017 році та отримала нагороду «Найкраща мобільна гра» на *The Game Awards* 2017[37].

Головний герой гри, *Mario*, знову повинен врятувати принцесу *Peach*, яка була викрадена злим ворогом, якого звати *Bowser.*У грі гравець керує *Mario*, який автоматично біжить, та повинен пройти кожен рівень, перестрибуючи перешкоди та вбиваючи ворогів(рис. 1.9). Ігровий процес гри складається з одного дотику до екрану: гравець тисне на екран, щоб змусити *Mario* стрибнути.

Однією з головних особливостей гри є режим «*Toad Rally*», в якому гравці можуть змагатися з іншими гравцями у виконанні рівнів. Гравці можуть збирати Тоадів, які є мешканцями Грибного королівства, щоб додати їх до свого королівства та підняти його рейтинг.

Рис. 1.9 – Геймплей гри «*Super Mario Run*»

Гра отримала позитивні відгуки за свій простий, але ефективний ігровий процес та красиву графіку. Однак, деякі гравці критикували гру за її високу ціну, яка становила $9.99 на момент випуску, порівняно з іншими мобільними іграми. Крім того, деякі користувачі також були незадоволені відсутністю можливості грати без підключення до Інтернету. Це може бути особливо незручним для гравців, які хочуть грати в метро або в інших місцях, де немає доступу до Інтернету.

Незважаючи на це, гра зберегла високий рівень популярності, що свідчить про успіх серії *Mario* та здатність *Nintendo* залучати інтерес гравців до своїх продуктів. Багато гравців залишались задоволеними якістю графіки та звукового супроводу, а також новими ігровими механіками, які були введені у гру.

Загалом, «*Super Mario Run*» може бути цікавою грою для шанувальників серії *Mario* та мобільних ігор загалом. Однак, вартість гри та відсутність можливості грати без підключення до Інтернету можуть бути обмеженнями для деяких гравців.

1.4.7 Порівняння обов'язкових елементів ігор жанру «*Runner*» у існуючих аналогах

Для кращого розуміння різниці між існуючими аналогами приведена таблиця 1.1, що демонструє наявність або відсутність характерних критеріїв ігор жанру «*Runner*» в досліджених аналогах.

Таблиця 1.1

Порівняння характерних критеріїв ігор жанру «*Runner*» в існуючих аналогах

Критерії	***Minion Rush: Runnning Game***	***Cookie Run:OvenBreak***	***Crash Bandicoot: On The Run!***	***Zombie Tsunami***	***Super Meat Boy Forever***	***Super Mario Run***
Система рівнів	+	+	+	–	+	+
Наявність бонусів	+	+	+	+	–	+
Наявність різноманітних персонажів	+	+	–	–	+	+
Можливість кастомізації персонажа	+	+	+	–	–	–
Наявність різноманітних локацій	+	+	+	+	+	+
Можливість грати без Інтернету	+	–	+	+	+	-
Процедурна генерація рівнів	+	+	+	+	+	+

1.5 Постановка завдання

Метою кваліфікаційної роботи є проектування та розробка гри жанру «*Runner*» з процедурною генерацією рівня.

Основний функціонал та контент, що необхідно розробити:

1. Створення проектної документації.
2. Створення унікального ігрового контенту.
3. Створення елементів процедурної генерації.
4. Розробка метода процедурної генерації рівня.
5. Створення головного персонажа.
6. Розробка фізики персонажа.
7. Розробка демонстраційної версії гри.
8. Компіляція та тестування демонстраційної версії гри.

Висновки до першого розділу

1. У результаті аналізу предметної області було виявлено основні характеристики ігор жанру «*Runner*», показано історію його становлення та розвитку. Досліджено основні методи процедурної генерації, а саме: імітаційний, функціональний та планувальний. Вказано для генерації якого контенту частіше всього використовуються перелічені методи. Зазначено, що комбінації різних методів процедурної генерації можуть додатково підвищити рівень варіативності та складності генерованого ігрового контенту.
2. Проаналізовано сучасні аналоги ігор жанру «*Runner*». Виявлено їх переваги та недоліки.
3. Здійснено постановку задачі з зазначенням бажаного результату.

РОЗДІЛ 2
ПРОЕКТНА ДОКУМЕНТАЦІЯ

2.1 Вступ

Події гри відбуваються у вигаданому світі, в якому головний герой панічно боїться курок. Під час своєї подорожі на нього нападає гігантська курка, через що головний герой тікає від неї через різноманітні локації, які з часом стає все важче пройти.

Однією з особливостей гри є посилення (бонуси), які отримуються при підбиранні відповідного предмету. У грі, вони спрощують проходження рівня на деякий час. Тривалість бонусу можна збільшити, придбавши відповідне покращення.

Структура рівнів унікальна. Кожен рівень генерується процедурно. Щоразу, коли гравець починає нову гру, рівень генерується випадковим чином, створюючи щоразу нову і непередбачувану пригоду. Рівень сповнений перешкод та ворогів, які гравець повинен подолати, використовуючи свої навички та кмітливість, щоб просуватися далі.

У міру просування гравця, рівень стає все складнішим, з новими і складнішими перешкодами для подолання, а темп гри підвищується. У дії грі представлені дві локації: ліс та пустеля, які генеруються випадковим чином. Це гарантує, що кожне проходження є унікальним, і що гравець повинен покладатися на свою адаптивність та навички вирішення проблем, щоб досягти успіху.

2.2 Жанр та аудиторія

Гра «*Chicken Chase*» виконана у жанрі «*Runner*».

Цей жанр дає виклик кмітливості та уважності, змагаючись с іншими гравцями пройденою відстанню та образами, отримуючи задоволення від динамічного геймплею.

Гра орієнтована на молодшу та підліткову аудиторію, тому не містить обмежуючого контенту, мінімальний вік гравця – 3 роки. Гра приваблює власників не найновіших моделей смартфонів, що шукають альтернативи в жанрі чи платформі. Для молодшої частини цієї категорії характерна потреба у простих іграх[39].

«*Chicken Chase*» сподобається фанатам ігор жанру «*Runner*». Ігри подібного жанру – «*Super Mario Run*», «*Subway Surfers*», «*Temple Run*». Гра є простою та динамічною, має просте керування та зрозумілі правила гри. Гра безперечно підходить для проведення вільного часу чи змагання між друзями.

Середня кількість часу, що витрачається на одну спробу становить 5-10 хвилин, що є оптимальним для цього жанру.

2.3 Основні особливості гри

Основними особливостями гри (*USP*) є:

1. Динамічний та швидкий геймплей є однією з головних особливостей гри. Гра кидає гравцю виклики, які випробовують його здатність швидко приймати рішення стежачи за ігровим оточенням.
2. Процедурна генерація є не менш важливою особливістю, що дає можливість проходити унікальні рівні кожного разу, коли гравець починає гру. Гра має різноманітні локації, які генератор випадковим чином обирає при генерації, а з часом локація змінюється.

3. У грі присутня велика кількість перешкод, що дає велику різноманітність при проходженні рівнів. При проходженні певної відстані, з’являються складніші перешкоди.
4. Протягом гри гравці можуть збирати бонуси та монети, щоб покращувати здібності свого героя та відкривати нових персонажів. Посилення можуть включати такі елементи, як прискорення, невразливість та додаткові життя.
5. Гра безкоштовна для завантаження з додатковими покупками в додатку, доступними для гравців, які хочуть прискорити свій прогрес або швидше розблокувати нових персонажів. Гра досягає хорошого балансу між тим, що пропонує гравцям безкоштовний і приємний досвід, а також надає можливості для монетизації.
6. Гра має великий потенціал для розвитку. Нові режими дають гравцям можливість проходити рівні з різноманітними завданнями чи специфічними умовами, що робить гру різноманітною. Також в майбутньому можна удосконалювати генератор та додавати нові елементи, а онлайн таблиця лідерів дасть можливість змагатися з гравцями по всьому світу.

2.4 Опис гри

2.4.1 Хід гри

Гра має звичайну механіку ігор жанру «*Runner*» з геймплейними особливостями. Основне завдання гравця – пробігти якнайдалі.

Ігровий процес гри полягає у керуванні героєм в процедурно згенерованому рівні. У грі перешкодами є вороги та статичні предмети локації.

Для керування гравця використовуються свайпи вгору та вниз. У грі гравець може: стрибати до двох разів, ковзати та швидко падати.

2.4.2 Ігрове оточення

Вся графічна складова гри виконана в піксельному стилі. У грі присутні такі локації як ліс, пустеля, засніжений ліс, рівнини, тощо . Перешкоди на рівнях генеруються, враховуючи фізику героя. Посилення на рівнях з’являються під час проходження рівня випадковим чином та обираються таким же способом. Посилення має обмежений час дії. Так як основна графіка гри є проста піксельна графіка, то в іншому обрамленні варто дотримуватись мінімалістичності та простоти виконання. Однак при цьому необхідно зробити картинку простою та запам’ятовуючою. Гра має специфічну графіку оточення: кожний елемент декору генеруються разом з рівнем, а колір заднього фону змінюється в залежності від типу локації. Приклад одного з згенерованих рівнів зображено на рисунку 2.1, інший приклад на рисунку 2.2.

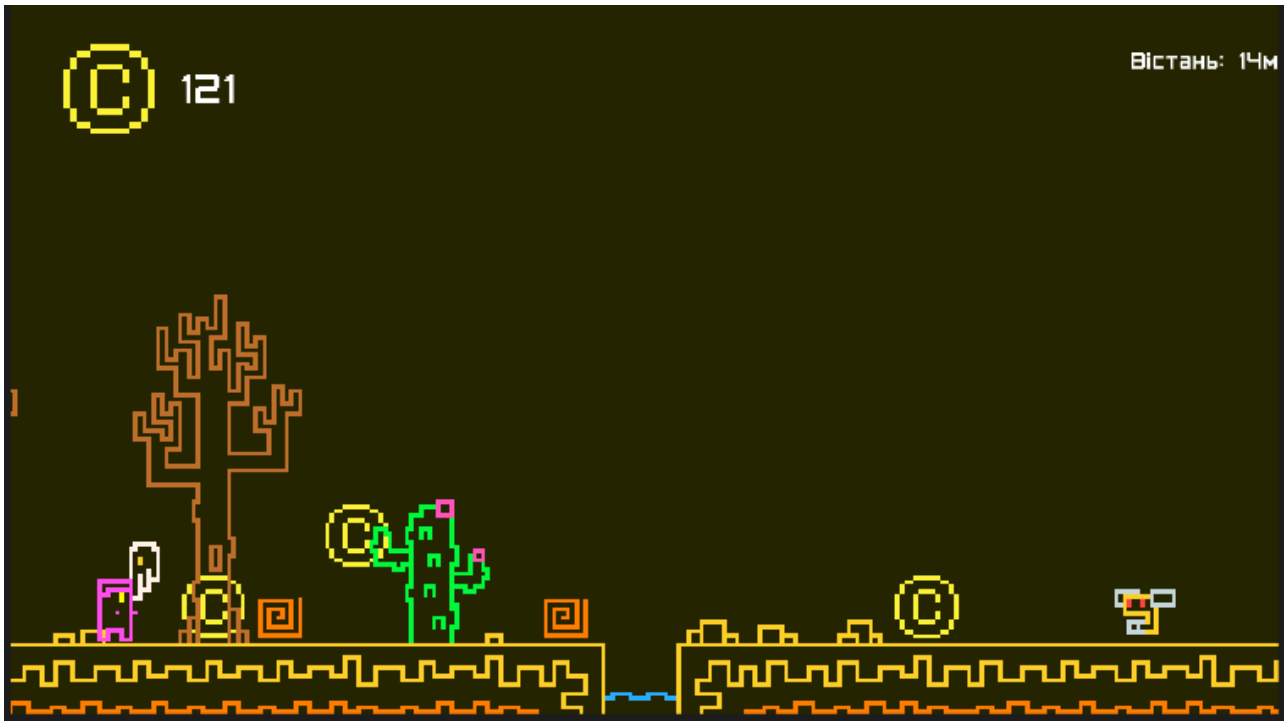


Рис. 2.1 – Приклад згенерованого рівня

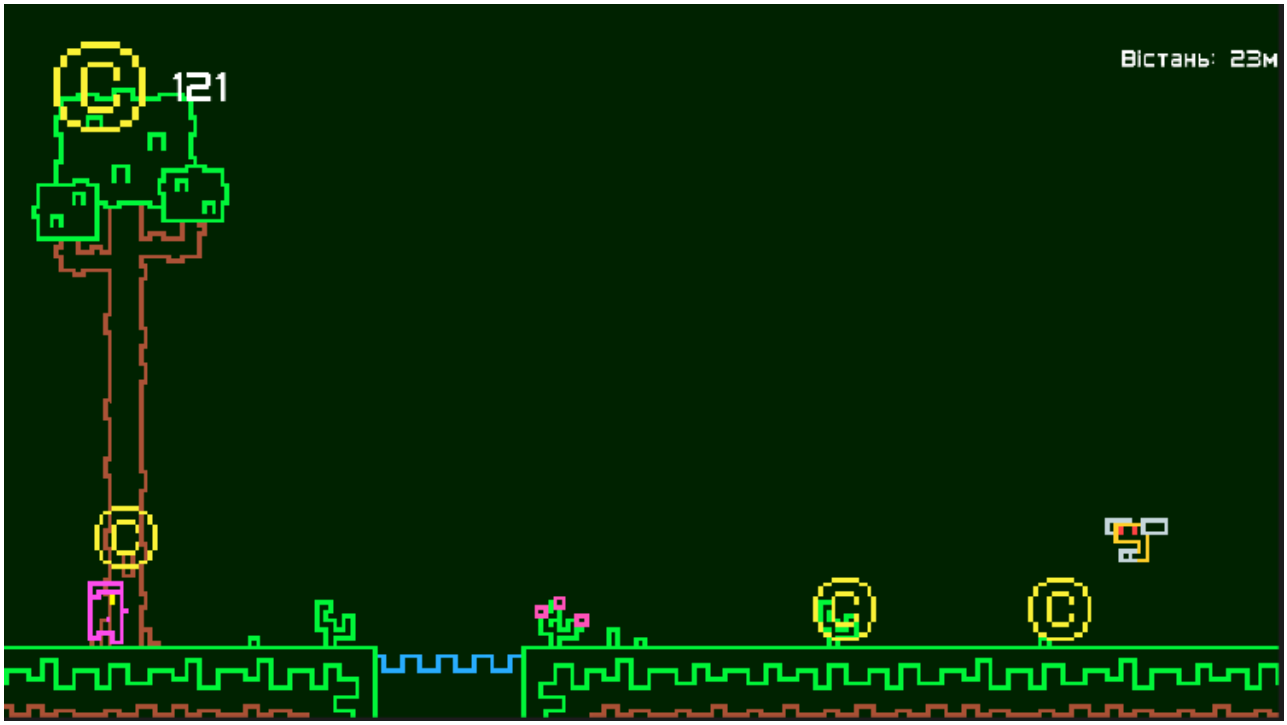


Рис. 2.2 – Приклад згенерованого рівня

2.4.3 Штучний інтелект

У грі процес створення ворога передбачає присвоєння кожному ворогу однієї з трьох можливих поведінок, які визначаються як типом ворога, так і елементом випадковості. Загалом існує три окремі категорії ворогів: статичні, наземні та повітряні.

При створенні ворогів ігрова система враховує різні фактори, щоб забезпечити різноманітний і непередбачуваний ігровий досвід. Завдяки поєднанню класифікації ворогів за типами та введенню елементу випадковості, ігрова механіка визначає, яку поведінку буде демонструвати ворог.

Перша категорія ворогів, відома як стаціонарні вороги. Ці вороги розташовані на землі та не рухаються.

Друга категорія – наземні вороги. Ці вороги можуть як стояти на землі, так і пересуватися по землі у бік гравця. Траекторія пересування гравця зображена на рисунку 2.2

Рис. 2.3 – Траєкторія пересування ворога наземного типу

Третя і остання категорія охоплює повітряних ворогів. Ці грізні вороги мають здатність перетинати небо, додаючи вертикальний вимір до ігрового процесу. Як і їхні наземні колеги, специфічна поведінка повітряних ворогів визначається як їхнім типом, так і елементом випадковості. Він може пересуватися тільки вертикально. Приклад тректорії пересування зображено на рисунку 2.3.

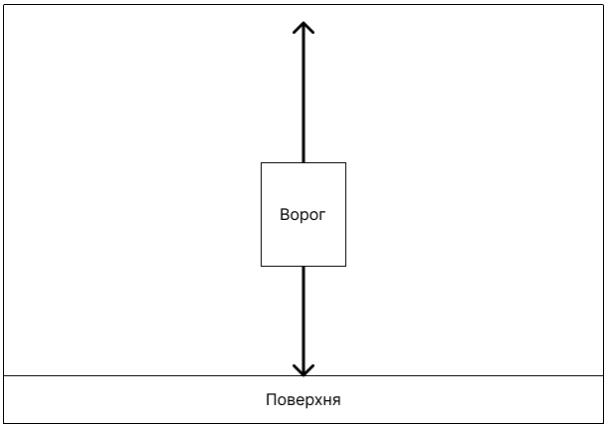


Рис. 2.4 – Приклад поведінки ворога повітряного типу

Завдяки такій багатогранній системі створення ворогів, гра надає гравцям широкий вибір ворогів, кожен з яких має унікальну поведінку та характеристики. Такий підхід не лише додає глибини та різноманітності ігровому процесу, але й посилює загальну складність та занурення, утримуючи гравців залученими та готовими протистояти постійним загрозам, які чекають на них в ігровому світі. Можливі поведінки ворога представлені в таблиці 2.1.

Таблиця 2.1

Типи поведінок ворогів

Назва поведінки	Опис поведінки
Спокій	Ворог стоїть на місці та не рухається. Може бути призначений на всі типи ворогів
Горизонтальний рух	Ворог пересувається в бік гравця із заданою швидкістю. У випадку якщо попереду річка, він падає в неї. Призначається на наземних та повітряних ворогів.
Вертикальний рух	Може отримати лише ворог, що може літати. Ворог літає вгору та вниз на задану відстань. З'являється в проміжку від землі то заданої висоти

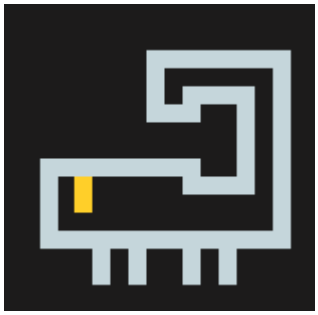
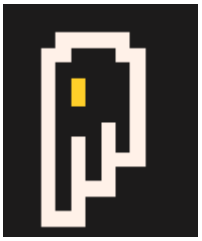
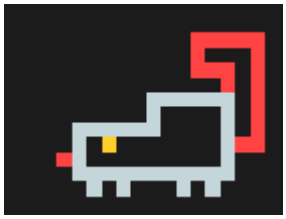
2.4.4 Вороги у грі

Таблиця 2.2

Характеристика ворогів

Назва ворога	Тип ворога	Зображення ворога
Курка	Наземний тип	
Змія	Наземний тип	

Продовження таблиці 2.2

Назва ворога	Тип ворога	Зображення ворога
Скорпіон	Наземний тип	
Привид	Повітряний тип	
Муха	Повітряний тип	
Слимак	Наземний тип	
Пацюк	Наземний тип	
Морква	Стаціонарний тип	

Продовження таблиці 2.2

Назва ворога	Тип ворога	Зображення ворога
Мураха	Наземний тип	

2.4.5 Процедурна генерація

Для того, щоб підвищити реіграбельність гри, було прийнято стратегічне рішення включити процедурний генератор в механіку її ігрового процесу. Цей підхід широко використовується в іграх жанру «*Runner*», що призвело до створення піджанру «*Endless Runner*». У цьому піджанрі гравці здійснюють нескінченні забіги через динамічно згенеровані рівні.

Процедурний генератор, що використовується у грі, будує рівні за допомогою блокової системи. Він має можливість генерувати різноманітні об'єкти, які сприяють покращенню ігрового процесу, в тому числі:

– поверхневі утворення;

– річки;

– елементи ландшафту;

– монети;

– вороги;

– статичні перешкоди.

Інтегруючи процедурний генератор у дизайн гри, гравці отримують безмежні можливості та унікальні виклики під час кожного проходження. Такий підхід гарантує, що не буде двох однакових ігрових досвідів, сприяючи азарту, несподіванкам та відчуттю дослідження. Динамічно згенеровані рівні, створені за допомогою процедурного генератора, сприяють реграбельності гри, заохочуючи гравців постійно прагнути до нових високих результатів і відкривати приховані секрети в ігровому світі, що постійно розвивається.

Процес генерації цих об'єктів відбувається за загальним алгоритмом, як показано на рисунку 2.13, де зображено циклічне створення блоків генератором.

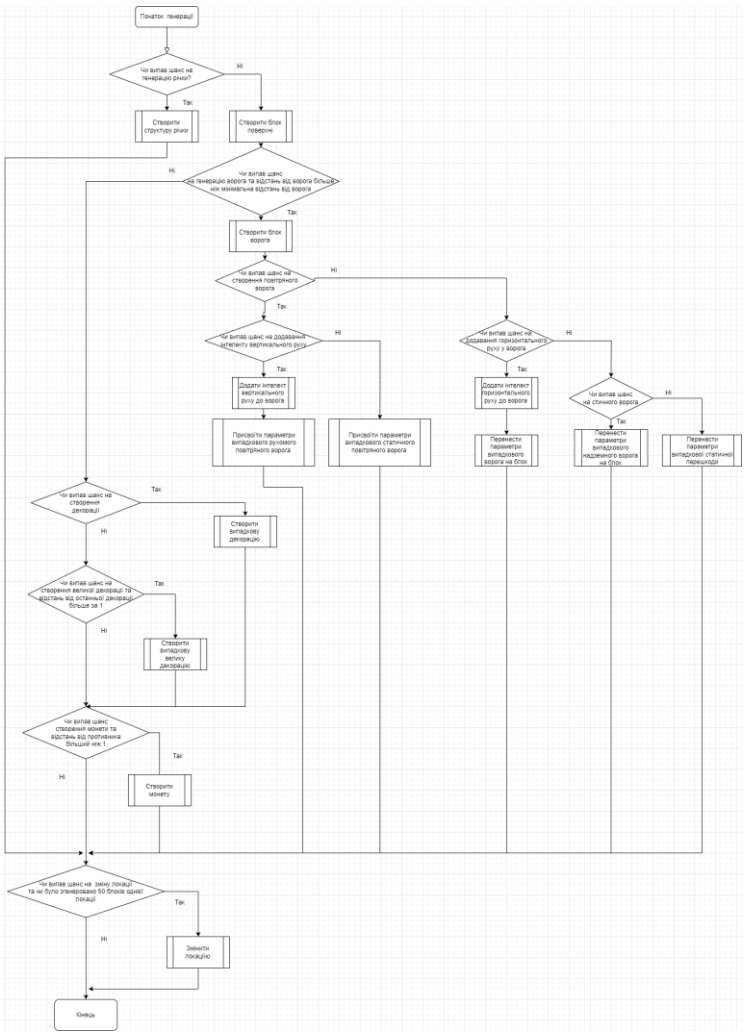


Рис. 2.5 – Алгоритм створення елемента рівня в процедурному генераторі

Генератор використовує комбінацію з функціонального та планувального підходів. Для планувального підходу існують наступні правила генерації:

1. Перші 30 блоків поверхні рівня генеруються без перешкод для створення спокійного ландшафту на початку гри.
2. Щоб змінити локацію, генератор повинен створити мінімум 50 блоків поверхні одної локації, щоб змінити локацію.
3. На початку гри вороги розміщуються з мінімальною відстанню в чотири блоки між ними, що виключає можливість з’явитися непрохідним перешкодам.

4. Річки, якщо вони з'являються, можуть мати максимальну ширину в два блоки, забезпечуючи природні водні об'єкти в межах рівнів.
5. Монети розміщуються щонайменше за один блок до або після ворога, надаючи гравцям можливість збирати монети с меншої небезпекою.
6. Вороги не можуть знаходитися в одному блоці з декораціями чи монетою, що зберігає чіткість ігрового процесу та полегшує виявленню ворога.
7. Великі декорації розташовані на відстані щонайменше одного блоку один від одного, щоб вони не накладались один на одного і зберігали візуальну привабливість.
8. Вороги та монети не можуть з’явитися над водою.
9. По мірі проходження гри відстань між ворогами та довжина річки збільшується на 1 блок.

Окрім правил, що описані для генератора, генератор має значення ймовірності, при якому може створитися той чи інший елемент рівня. Важливо зазначити, що генератор створить об’єкт тільки тоді коли виконуються умови та випадає ймовірність.

Генератор використовує функціональний підхід для визначення дії генератора. Для кожної дії існує власне значення ймовірності.

Ймовірності створення об’єктів показані в таблиці 2.3.

Таблиця 2.3

Ймовірність створення блоків та структур.

Назва дії	Ймовірність створення
Ймовірність змінити локацію	0,2
Ймовірність створити структуру «Річка»	0,05
Ймовірність створити малу декорацію	0,3

Продовження таблиці 2.3

Ймовірність створити велику декорацію	0,4
Ймовірність створити ворога	0,5
Ймовірність створити монету	0,5
Ймовірність створити повітряного ворога	0,5
Ймовірність додати вертикальний рух до повітряного ворога	0,5
Ймовірність створити горизонтальний рух до наземного ворога	0,5
Ймовірність створити статичного ворога	0,5
Ймовірність створити монету	0,5

Генератор працює автономно, використовуючи свої внутрішні можливості для створення рівнів без будь-якого втручання гравця. Завдяки широкому набору параметрів, генератор має можливість динамічно створювати велику кількість рівнів. Ці рівні розроблені таким чином, щоб бути дуже різноманітними, охоплюючи широкий спектр структур, ворогів та нагород. Процедурний генератор дотримується певних правил і вказівок, щоб гарантувати, що ігровий процес буде не лише збалансованим, але й цікавим для гравців. Такий підхід дозволяє створювати захоплююче та захоплююче середовище, яке гравці можуть досліджувати та взаємодіяти з ним. Більше того, ця система усуває необхідність прямої взаємодії з користувачем або кастомізації, надаючи гравцям свободу насолоджуватися унікальними світами гри без жодних додаткових зусиль. В результаті гравці можуть вирушати в подорож постійно мінливими ландшафтами, відкриваючи для себе нові виклики та сюрпризи на своєму шляху. Здатність процедурного генератора генерувати безліч динамічних рівнів гарантує, що гравці завжди матимуть свіжий і захоплюючий досвід, що підвищує реграбельність і довговічність гри.

2.4.5 Ігрові посилення

Гра має різноманітні посилення, які надають гравцям тимчасові переваги. Ці посилення додають ігровому процесу ще більшого азарту, а також роблять гру ще більш привабливою.

У грі гравці можуть отримати доступ до різних типів посилення, які надають їм унікальні здібності та переваги. Ці посилення покращують ігровий процес і сприяють досягненню успіху. Від прискорення, що дозволяє блискавично пересуватися, до щитів, що забезпечують тимчасову невразливість, спектр посиленнь забезпечує динамічну та захоплюючу ігрову сесію.

В процесі гри гравець може зібрати кілька посилень, при чому час дії кожного окремий. Якщо гравець випадково натрапляє на здібності того ж типу, то таймер для цього бонусу перезапускається, фактично обнуляючи тривалість його дії, так, ніби гравець щойно отримав його заново. Це гарантує, що гравці отримують справедливий і збалансований досвід, оскільки гра запобігає надмірному накопиченню бонусів, які потенційно можуть порушити рівновагу в грі.

Таким чином, гра пропонує різноманітний вибір бонусів, які дають гравцям тимчасові переваги. Одночасна активація декількох бонусів поєднує їхні ефекти, посилюючи можливості гравця. Водночас, коли гравцеві трапляються однотипні бонуси, таймер перезапускається, запобігаючи їх надмірному накопиченню. Завдяки цій механіці гравці можуть стратегічно використовувати ці бонуси на свою користь, занурюючись у захопливий і тактично багатий ігровий досвід. На таблиці 2.4 зображені ігрові посилення, що є у грі.

Таблиця 2.4

Ігрові посилення

Вид посилювача	Опис
Магніт	Притягує монети до гравця
Невразливість	Гравець невразливий до ворогів

Продовження таблицы 2.4

Вид посилювача	Опис
Уповільнення	Сповільнює швидкість гри
Мости	Ставить мости через річки
Потрійний стрибок	Додає додатковий стрибок

2.4.6 Платформа

Гра спеціально спроектована і розроблена для платформи *Android*, що забезпечує сумісність з широким спектром *Android*-пристроїв. Вона може похвалитися вражаючою особливістю – низькими системними вимогами, що дозволяє їй безперебійно працювати навіть на пристроях зі скромними технічними характеристиками. Ця особливість доступності гарантує, що ширша аудиторія може насолоджуватися грою, не відчуваючи жодних проблем з продуктивністю.

У майбутньому планується розширити доступність гри і для інших операційних систем. Це рішення спрямоване на залучення нових користувачів та подальше зростання популярності гри. Впроваджуючи гру на різні платформи, такі як *iOS* або *Windows*, розробники очікують значного збільшення кількості активних користувачів, оскільки це відкриває можливості для людей, які використовують пристрої за межами екосистеми *Android*, приєднатися до розваг.

Інформація про системні вимоги наведені в таблиці 2.5, в якій наведено вичерпну інформацію про технічні характеристики, необхідні для безперебійної роботи гри на мобільних пристроях. У цій таблиці наведено мінімальні та рекомендовані вимоги до апаратного та програмного забезпечення, яким повинні відповідати користувачі, щоб забезпечити оптимальний ігровий досвід на своїх пристроях *Android*.

Таблиця 2.5

Системні вимоги гри

Параметр	Мінімальні вимоги	Рекомендовані вимоги
ОС	*Android 7*	*Android 9*
Кількість памяті	50 Мб і більше	
Оперативна память	1 Гб і більше	
Процесор	Qualcom Snapdragon 400 або вище	

2.5 Функціональна специфікація

2.5.1 Модель гри

Запускаючи гру, гравець бачить меню гри. На фоні зображений початок рівня, що був створений процедурним генератором. На першому плані зображений текст «Натисни, щоб почати гру». При дотику на екран починається кат-сцена, після чого починається гра.

На верхньому правому боці екрану знаходяться кнопки регулювання музики та звуків у гри, де гравець може їх ввімкнути або вимкнути.

З правого боку знаходиться кнопка кастомізації, де гравець може змінити зовнішній вигляд свого персонажа за якого він буде грати. Якщо гравець змінює зовнішність, в головному меню зміниться зовнішність гравця на той, що обрав гравець.

Нижче від кнопки кастомізації знаходиться кнопка, що відкриває меню посилень. В цьому меню гравець може покращити посилювачі, а також придбати одноразові предмети, наприклад, можливість продовжити гру після програшу. З лівого верхнього боку знаходиться інформація про поточну кількість монет у гравця.

З лівого нижнього боку знаходиться кнопка виходу з гри, при натисканні якої гра перепитує чи справді гравець хоче вимкнути гру.

Схема меню зображена на рисунку 2.14, схема кастомізації на рисунку 2.15, схема меню покращень на рисунку 2.16, меню виходу з гри на рисунку 2.17.

Рис. 2.6 – Схема головного меню гри

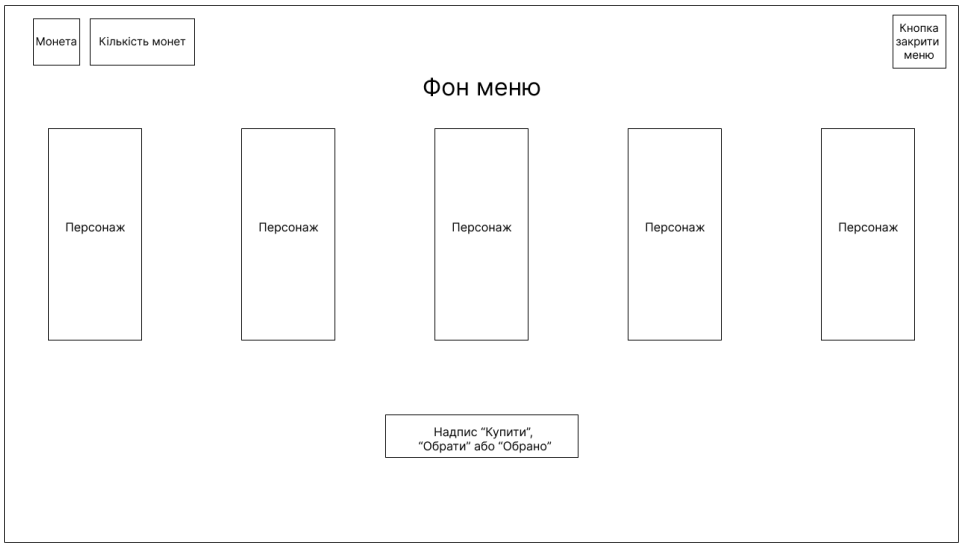


Рис. 2.7 – Схема меню кастомізації

Рис. 2.8 – Схема меню покращень

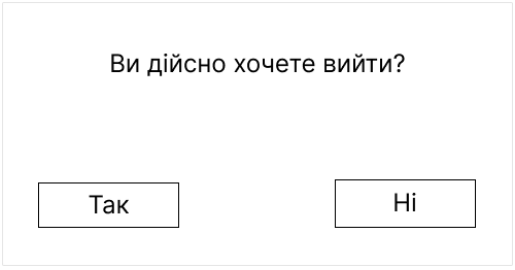


Рис. 2.9 – Схема меню виходу з гри

2.5.2 Кат-сцена

Після дотику на екран в головному меню гри, починається кат-сцена. Його алгоритм наступний:

1. Гравець стоїть на місці та дивиться в правий бік екрану.
2. Гігантська курка підходить до гравця з лівого боку.
3. Гравець повертається та лякається курки, при цьому над гравцем з’являється знак оклику та вмикається звук лякання гравця.
4. Гравець та гігантська курка починають бігти.
5. Камера поступово пересувається направо так, щоб гравець був на лівій частині екрану.

2.5.3 Формули

Для розробки гри було створено та взято кілька формул для правильної роботи гри.

Формула визначення сили стрибка

$$F_{ст} = mv \tag{2.1}$$

де: $F_{ст}$ – сила, яка потрібна для стрибка

m – маса гравця

v – вертикальна швидкість, яку потрібно досягнути.

Формула для визначення напрямку до гравця

$$\vec{d} = \vec{p} - \vec{e} \tag{2.2}$$

де: $\vec{d}$ – напрямок до гравця за векторами *xyz*

$\vec{p}$ – позиція гравця за векторами *xyz*

$\vec{e}$ –довільна точка за векторами *xyz*

Формула псевдовипадкових чисел(лінійний конгруентний метод):

$$X_{n+1} = (a * X_n + c) \bmod m \tag{2.3}$$

Де: X_{n+1} – наступне псевдовипадкове число

X_n– поточне псевдовипадкове число

a,c,m – константи, які визначаються при реалізації проекту

Значення X_0 (початкове значення) вибирається випадково або за певною системною характеристикою (наприклад, системним часом), і потім застосовується рекурентна формула для генерації наступних чисел.

Значення a, c і m обираються таким чином, щоб забезпечити добру рівномірність розподілу псевдовипадкових чисел і довгий період повторення перед появою циклів.

2.6 Графіка та анімації

Таблиця 2.6

Графічні елементи гри

Елемент	Опис
Головний герой	Головний герой
Тайлсет локацій	Набір елементів графіки для локацій
Противники	Наземні, повітряні та стаціонарні вороги
Елементи інтерфейсу	Графічні елементи інтерфейсу гри
Варіації головного героя	Різноманітні образи для головного героя для кастомізації
Монета	Зображення монети

Таблиця 2.7

Анімації гри

Назва анімації	Опис
Анімація бігу героя	Покадрова анімація бігу гравця
Анімація стрибка героя	Статична картина стрибаючого гравця
Анімація пригинання героя	Покадрова анімація укороченого героя
Анімація падіння героя	Статична картинка лежачого героя
Анімація спокою ворога	Покадрова анімація спокою
Анімація руху ворога	Покадрова анімація для наземних та повітряних ворогів

2.7 Звуки та музика

Таблиця 2.8

Музикальні ефекти гри

Елемент	Коментарі
Музика для меню гри	–
Звук при першому стрибкі	–
Звук при другому стрибкі	–
Звук при приземлені	–
Звук при програші	–
Звук при підбирані підсилювача	–
Звук при закінченні підсилювача	–
Звук при використанні предмету	–
Звук при натисканні на кнопку	–
Звук при ляканні гравця	–
Звук підбирання монети	–

Висновки до другого розділу

1. Було створено концептуальний документ гри, необхідний для презентації видавцям чи інвесторам.
2. Був розроблений дизайнерський документ гри, що описує основні особливості гри.

РОЗДІЛ 3
РОЗРОБКА ГРИ

3.1 Обґрунтування вибору засобів реалізації

Для того, щоб розробити мобільну гру був обраний ігровий рушій *Unity 2021.3.7f1*. *Unity* – це потужний ігровий рушій, що розроблений компанією *Unity Technologies* у 2005 році та активно розвивається. *Unity* дає змогу створювати програми різної складності, які можуть бути експортовані на більш ніж 25 платформ, що включає мобільні пристрої, консолі, персональні комп'ютери, інтернет-додатки та інші.

Редактор рушія *Unity* являє з себе візуальне середовище, яке має багато елементів, таких як: редактор анімацій, редактор сцени, редактор модулів та інші. Для функціонування фізики в *Unity* використовується фізичний рушій *PhysX* від компанії *NVIDIA*. Програма має візуально зрозумілий та простий інтерфейс, що робить її зручною платформою для простих чи середніх проектів, проте зі збільшенням масштабу гри розробка гри стає складнішою, через що потрібно вводити сторонні методи організації проекту.

Unity пропонує кілька ключових переваг, включаючи візуальне середовище розробки, модульну систему компонентів та велику спільноту. Попри всі переваги рушія він також має свої недоліки. Наприклад, гра, розроблена в *Unity* має обмеження продуктивності при розробці складних сцен, що попри зусилля розробників рушія ускладнює розробку та оптимізацію подібних ігор.

Розширені можливості *Unity* становлять значний виклик у навчанні, незважаючи на те, що платформа є зручною для початківців. Розробники, які вивчають складні функціональні можливості, такі як програмування шейдерів або просунуті фізичні симуляції, зіткнуться з необхідністю додаткового часу та зусиль, щоб повністю зрозуміти ці концепції.

Для програмування було вирішено використовувати середовище програмування *Visual Studio* 2022, що легко інтрегрується з редактором *Unity*. Окрім швидкого відкриття та можливості відстежувати помилки в коді ігрового рушія, інтеграція дозволяє використовувати автодоповнення середовища для компонентів *Unity*.

Для розробки було обрано мову програмування *C#*, що підтримується *Unity*. *C#* – це строго типізована об'єктно-орієнтована мова програмування, розроблена для платформи *.NET*. Розроблена Скотом Вілтамутом, Пітером Гольде та Андерсом Гейлсбергом під керівництвом *Microsoft Research*(при фірмі *Microsoft*).

Зазвичай, *C#* є більш зручною мовою програмування порівняно з іншими, якщо перший пріоритет – розробка гри, а потім вже робота над складними аспектами програмування [39].

В меню *Unity*, можна обрати потрібний проект або завантажити відсутню версію редактора, яка необхідна на даний момент (рис. 3.1).

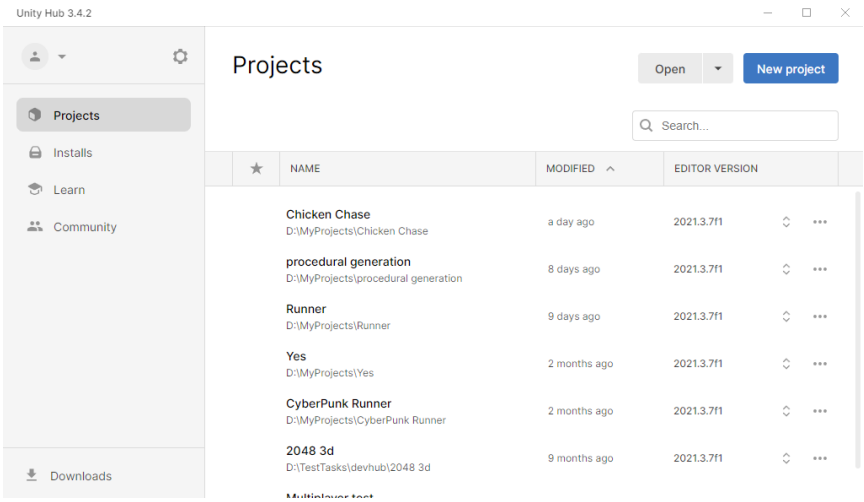


Рис. 3.1 – Вікно вибору проектів у *Unity Hub*

Для створення та редагування наявних спрайтів для гри використовується програма *Aseprite*. *Aseprite* – це запатентований редактор зображень, призначений переважно для малювання та анімації піксельного мистецтва. Він працює в *Windows*, *macOS* і *Linux* і містить різні інструменти для редагування

зображень і анімації, такі як: шари, кадри, підтримка атласів зображень, сценарії *Lua* тощо.

Дана програма користується великим попитом серед художників піксельного стилю, тому що була розроблена саме для них.

Лівий бік представляє набір кольорів, який можна обрати відповідною кнопкою в інтерфейсі. Також в редакторі присутня палітра, в якій додатково можна міняти прозорість кольору. Файли збережених проектів мають формат *.aseprite*. Експортовані спрайти мають формат *.png*, що підтримує прозорі фони.

Зображення інтерфейсу *Aseprite* показано на рисунку 3.2.

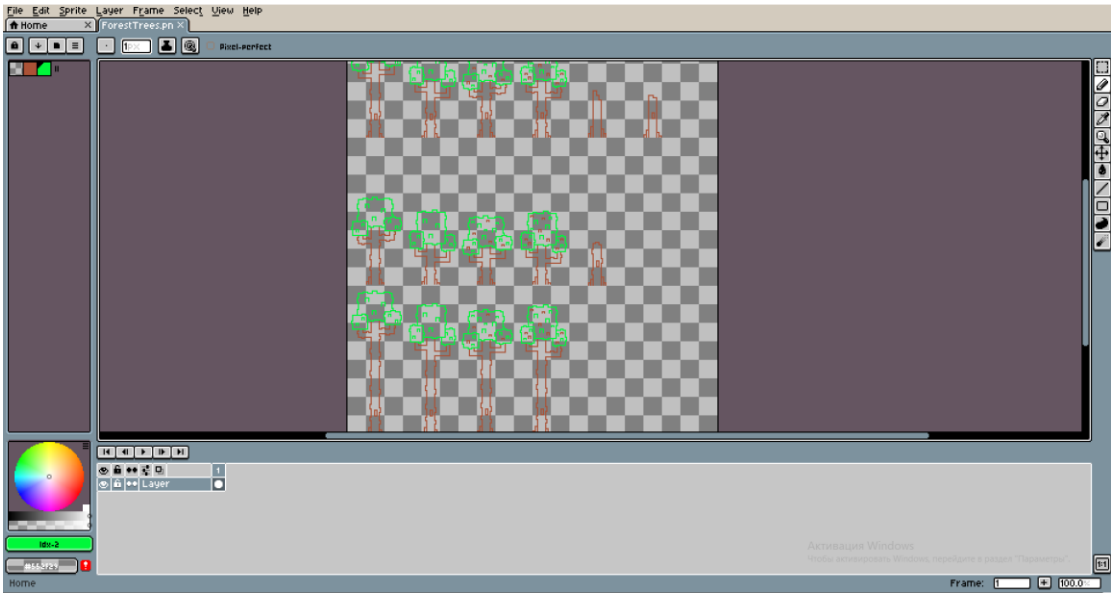


Рис. 3.2 – Інтерфейс програми *Aseprite*

3.2 Початок розробки

3.2.1 Створення головної сцени

При запуску будь-якого проекту з’являється головне вікно інтерфейсу, що відображає основні елементи інтерфейсу, що використовуються найчастіше: вікно сцени, вікно гри, інспектор, ієрархія, вікно файлів проекту та консоль.

Розташування вікна сцени та гри зосереджене в центрі екрану. Вони знаходяться в одному вікні, але можуть бути переключені за допомогою вкладок, розташованих над вікном. Вікно сцени використовується для створення

рівня, додавання нових об'єктів та маніпулювання існуючими елементами гри, тоді як вікно гри дозволяє спостерігати за грою з точки зору гравця. Розробники також можуть використовувати вікно гри для тестування функціоналу та візуального вигляду гри.

Ієрархія розташована ліворуч від вікна сцени і вона відображає всі ігрові об'єкти, розміщені на сцені. Крім того, вона показує пріоритетність одних об'єктів відносно інших (об'єкти, що знаходяться нижче по ієрархії, успадковують основні характеристики об'єктів, розташованих вище по ієрархії).

При виборі об'єкта в ієрархії у вікні інспектора є можливість редагувати його компоненти та додавати нові. Вікно інспектора розташоване праворуч від вікна сцени і містить всю інформацію про компоненти об'єкта.

Внизу під вікном сцени за замовчуванням розташоване вікно проекту. Воно відображає файли та директорії, які використовуються в проекті, зокрема програмні скрипти, анімації, префаби ігрових об'єктів, моделі, анімації та інше.

В редакторі також присутнє вікно консолі, яке виводить повідомлення та помилки, що виникають під час компіляції та під час гри.

Зображення головної сцени проекту зображено на рисунку 3.3

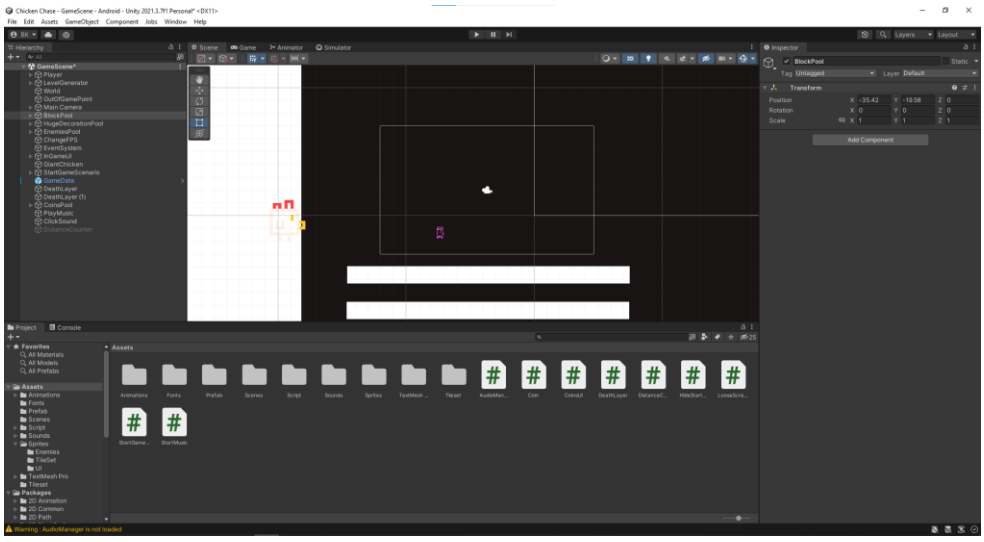


Рис. 3.3 – Головна сцена гри

Оскільки світ генерується процедурно, головна сцена складається з вказівників на генерацію, префаб гравця та елементи інтерфейсу. Для цього було створено наступні об'єкти: Гравець (*Player*), Головна камера (*MainCamera*),

Початковий сценарій (*StartScenario*), Генератор рівнів (*LevelGenerator*), Ігровий інтерфейс (*InGameUI*) та Світ (*World*). Щоб зручно зберігати префаби, які використовуються для генерації рівня, вони зберігаються у відповідних об'єктах: Пул блоків *BlockPool*, *HugeDecorationPool*, *EnemiesPool*, *CoinsPool*.

Через те, що світ генерується процедурно, головна сцена скаладається з вказівників до генерації, префаб гравця та елементи інтерфейсу. Для таких цілей були створені наступні об’єкти: *Player*, *MainCamera*, *StartScenario*, *LevelGenerator*, *InGameUI*, *World*.

3.2.2 Створення об’єктів для гравця

Для функціонування гравця до об’єкту у грі було додано компоненти:

1. *Sprite Renderer* – компонент, який відповідає за відображення спрайтів в ігровому полі. Він використовується для створення анімацій гравця (рис 3.4).

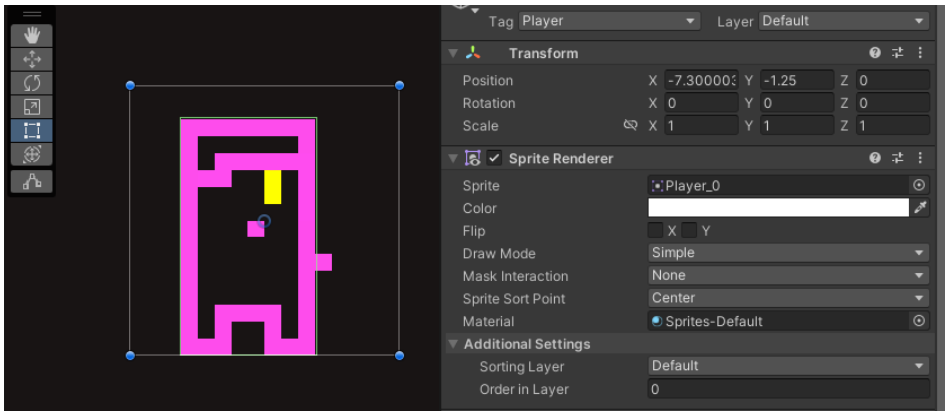


Рис. 3.4 – параметри компоненту *Sprite Renderer*

2. *Box Collider 2D* – компонент, який моделює тверде тіло в фізичному рушії, в формі прямокутника. Для реалізації механіки ковзання було створено два колайдера: один для звичайного бігу і один для ковзання. Тільки один з них може бути активним. Зображення колайдерів показано на рисунку 3.5.

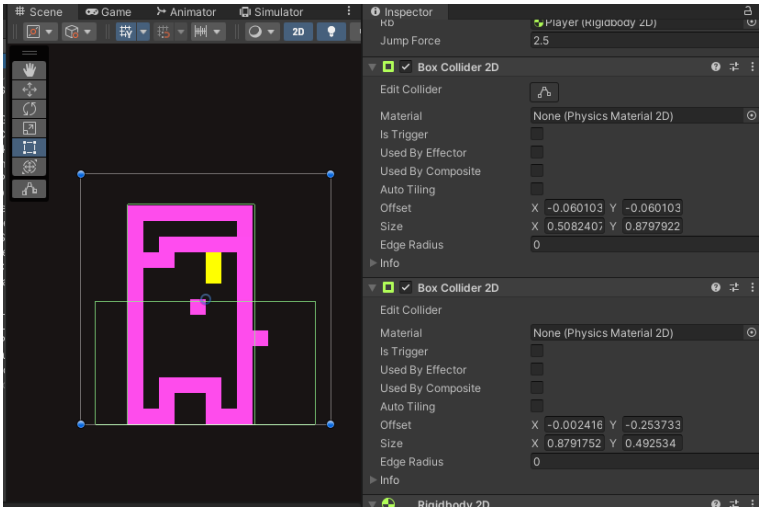


Рис. 3.5 – Колайдери гравця

3. *Rigidbody 2D* – це компонент, який додає фізичні взаємодії, такі як сила гравітації, маса, опір повітря та імпульсу. Він також дозволяє відслідковувати взаємодії між колайдерами та тригерами. Параметри компонента показані на рисунку 3.6.

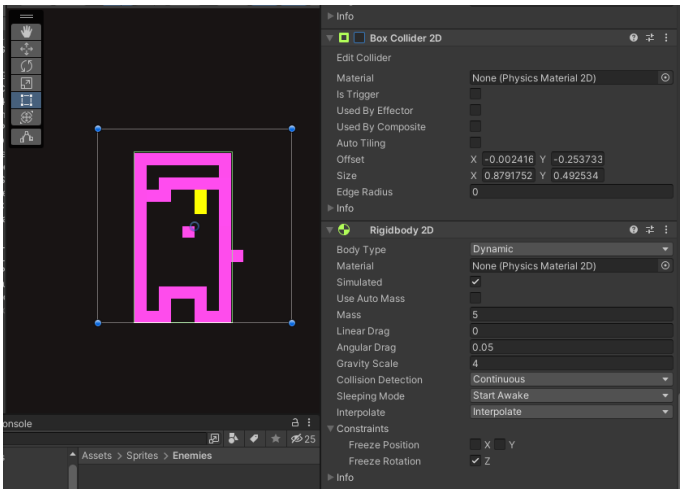


Рис. 3.6 – Компонент *Rigidbody 2D*

4. *Animator* – компонент, який відтворює анімації. Для редагування анімацій використовується вікно з відповідною назвою. Оскільки всі анімації будуть змінюватися в скрипті, зв'язки між анімаціями не потрібні (рис 3.7).

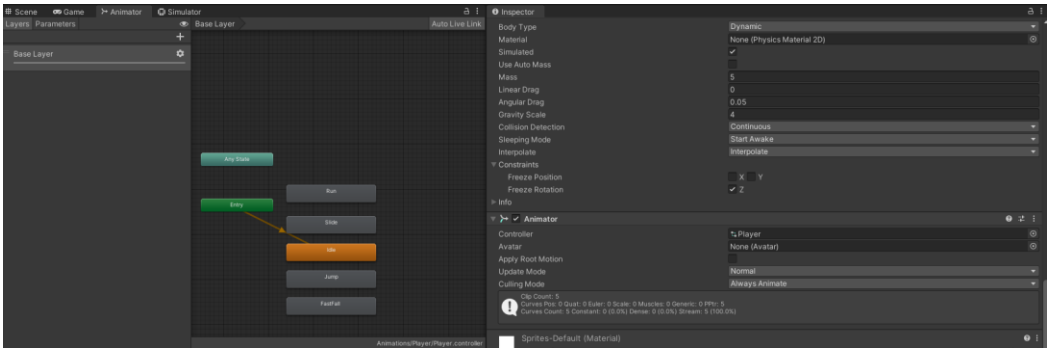


Рис. 3.7 – Анімації в аніматорі

Для керування гравцем за допомогою свайпу по екрану було додано невидимий об'єкт *TouchController*, який має наступні компоненти:

1. *Image* – компонент, який використовується для відображення спрайта в інтерфейсі. У даному випадку він використовується для відслідковування натискання пальця на екран;
2. *TouchController* – скрипт, який відстежує рухи пальця та передає дані, коли виконується свайп.

Зображення компоненів *TouchController* показано на рисунку 3.8.

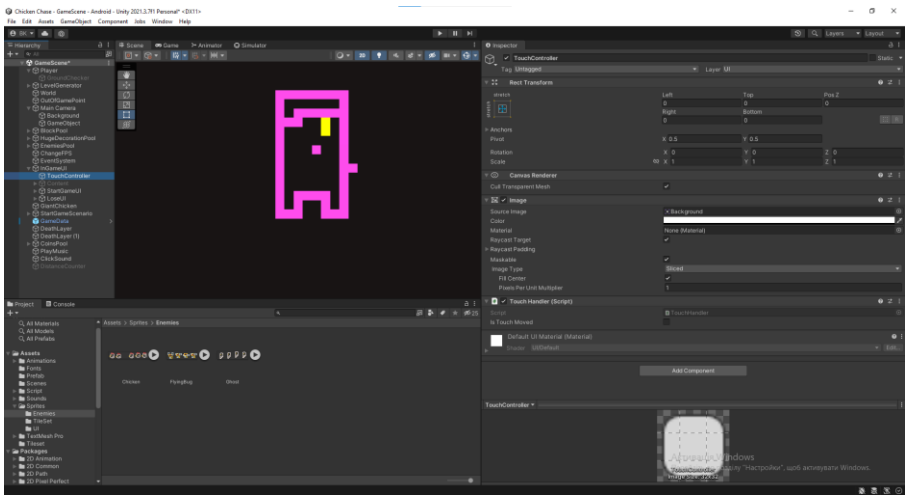


Рис. 3.8 – Компоненти об’єкту *TouchController*

Для того, щоб знати чи знаходиться гравець на поверхні було створено клас *GroundChecker*. Він включає наступні компоненти:

– *Box Collider 2D* – зона, яка перевіряється на наявність поверхні під гравцем;

– *GroundChecker* – скрипт, що оброблює наявність поверхні.

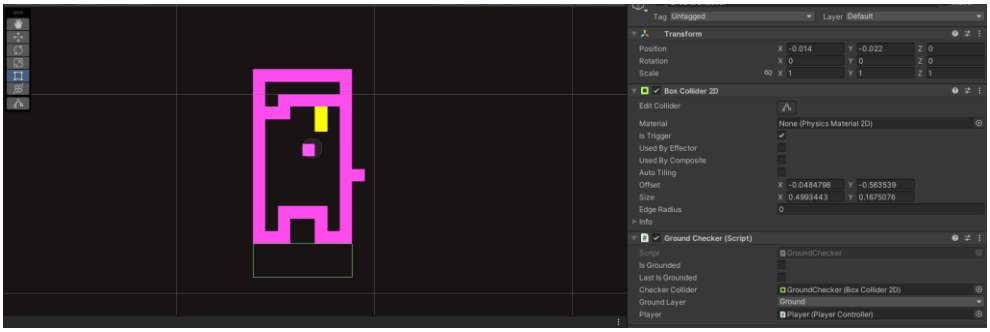


Рис. 3.9 – Вигляд об’єкту *GroundChecker*

3.4 Скрипти для роботи гравця

У головному скрипті гравця, *PlayerController*, наявні наступні параметри:

– *PlayerState* – відображає поточний стан гравця. Включає 5 станів: спокій (*Idle*), біг (*Run*), стрибок (*Jump*), швидке падіння (*FastFall*) і ковзання (*Slide*).

– *DefaultCollider* – стандартний колайдер для гравця;

– *GroundLayer* – шар, на якому розташовані блоки поверхні;

– *TouchHandler* – клас, що відповідає за відстеження дотиків по екрану;

– *SlideCollider* – колайдер, який замінює *DefaultCollider* під час ковзання;

– *GroundChecker* – клас, що перевіряє наявність поверхні для стрибка гравця;

– *Physics* – клас, що відповідає за фізичні взаємодії гравця;

– *Animation* – клас, що керує анімаціями гравця під час зміни стану;

– *MaxAdditionalJumps* – максимальна кількість додаткових стрибків у повітрі;

– *CurrentAdditionalJumps* – поточна кількість доступних додаткових стрибків у повітрі;

– *SlidingDuration* – тривалість перебування гравця в стані ковзання;

– *SlideTimeLeft* – залишений час для ковзання.

– *IsPlayingGame* – прапорець, що вказує, чи розпочата гра;

– *CoinPickSound* – звук, який відтворюється при підбиранні монети;

– *FirstJumpSound* – звук першого стрибка;

– *SecondJumpSound* – звук другого стрибка;

– *DeathSound* – звук, що відтворюється при смерті гравця від ворога;

– *DrowningSound* – звук, що відтворюється при смерті гравця від потоплення (рис 3.10).

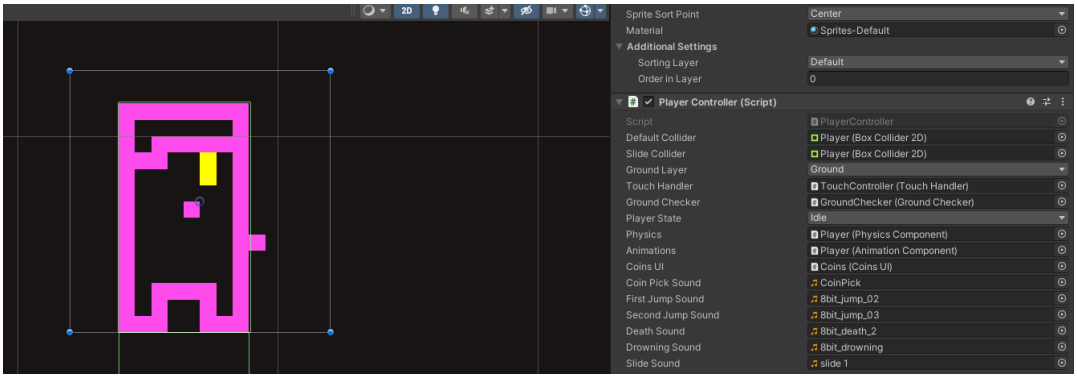


Рис. 3.10 – Компонент *PlayerController*

Оскільки гравець рухається тільки вертикально і не має горизонтального руху, скрипт гравця «підписується» на подію *OnTouchMove*, яка знаходиться в класі *TouchHandler*. Подія *OnTouchMove* спрацьовує, коли користувач здійснює свайп по екрану телефона.

Клас *TouchHandler* складається з наступних компонентів:

– Подія *OnTouchMove*, яка викликається при свайпі;

– *StartPosition* – координати точки, де палець доторкнувся до екрану;

– *MinDistanceToMove* – визначає мінімальну відстань, яку палець має подолати, щоб вважатися свайпом;

– *MoveVector* – представляє напрямок свайпу, обчислений на основі руху пальця;

– *IsTouchMoved* – вказує, чи здійснювався рух пальця після доторкання до екрану;

– *PrevTouchCount* – зберігає кількість пальців, які дотикалися екрану у попередньому кадрі.

Процес відстеження дотиків відбувається в методі *Update* класу *TouchHandler*, який виконується кожен кадр гри. В цьому методі обробляється

перший палець, що доторкнувся до екрану, та його фаза. Робота методу полягає у наступному:

1. При дотику до екрану зберігається початкова позиція дотику (*startPosition*).
2. При руху пальця обчислюється відстань, яку він подолав, щоб визначити, чи можна вважати цей рух свайпом. Ця відстань порівнюється з мінімальною відстанню (*minDistanceToMove*), встановленою для визнання свайпу.
3. Якщо рух пальця вважається достатньо великим для свайпу, то значення *startPosition* та *moveVector* обнуляються, і обчислюється напрямок свайпу на основі руху пальця.
4. В результаті спрацьовує подія *OnTouchMove* з передачею напрямку свайпу як аргумента.

Нижче наведений лістинг 3.1, який демонструє метод *Update* в класі *TouchHandler*.

Лістинг 3.1. Фрагмент коду метода *Update*

```
if(Input.touchCount > 0 && !isTouchMoved)
{
    Touch touch = Input.GetTouch(0);
      if (touch.phase == TouchPhase.Began)
      {
          startPosition = touch.position;
      }
      else if (touch.phase == TouchPhase.Moved)
      {

          if (Vector2.Distance(startPosition, touch.position) >
           minDistanceToMove)
            {
                isTouchMoved = true;
                    moveVector = (touch.position -
                     startPosition).normalized;
                OnTouchMove?.Invoke(moveVector);
                ResetMoveVector();
                ResetStartPosition();
            }
       }
           else if (touch.phase == TouchPhase.Ended || touch.phase ==
            TouchPhase.Canceled)
       {
            isTouchMoved = false;
```

```
        }
      }
         if(prevTouchCount != Input.touchCount && prevTouchCount == 1)
         {
             isTouchMoved = false;
         }
         prevTouchCount = Input.touchCount;
}
```

Після спрацьовування події *OnTouchMove* відбувається обробка напрямку свайпу в головному скрипті гравця. В цьому місці приймається рішення про подальші дії гравця: стрибати або ковзати.

Якщо гравець перебуває у повітрі, але потрібно здійснити ковзання, гравець спочатку починає швидко падати вниз, а потім починає ковзати.

При початку ковзання у гравця запускається таймер, і коли цей таймер досягає певного значення, гравець змінює свій стан на біг. При зміні стану гравця на ковзання викликається метод *SwitchCollider*, який змінює активність колайдерів (*defaultCollider* та *slideCollider*), як показано в лістингу 3.2.

Лістинг 3.2. Метод *SwitchCollider*

```
private void SwitchCollider()
{
    defaultCollider.enabled = playerState != PlayerState.Slide;
    slideCollider.enabled = playerState == PlayerState.Slide;
}
```

У випадку зі стрибком необхідно перевірити, чи гравець знаходиться на поверхні або має додаткові стрибки. Це визначає, чи буде витрачатися додатковий стрибок та який звук стрибка буде відтворений. У випадку, якщо гравець не знаходиться на поверхні та вичерпав усі додаткові стрибки, стрибок не буде виконаний.

При стрибку також обнуляється параметр *slideTimeLeft*, який використовується для відслідковування тривалості ковзання. Після цього стан гравця змінюється на стрибок, а метод *Jump* обробляє цю дію в параметрі *physics*.

Об’єкт *Physics* є екземпляром класу *PhysicsComponent* і має такі параметри:

– *Rb* – посилання на компонент *Rigidbody* гравця;

– *VerticalVelocity* – посилання на параметр *Rb*, який містить інформацію про поточну вертикальну швидкість гравця;

– *JumpForce* - сила, з якою гравець стрибне.

В класі присутні два методи: *Jump* та *FastFall*. Метод *Jump* виконує стрибок, в той час як метод *FastFall* примусово змінює вертикальну швидкість гравця на від’ємну, щоб гравець миттєво почав падати. Код метода *Jump* в *PhysicsComponent* представлений в лістингу 3.3.

Лістинг 3.3 Метод *Jump* в *PhysicsComponent*

```
public void Jump()
{
    Vector2 newVelocity = rb.velocity;
    newVelocity.y = jumpForce * rb.mass;
    rb.velocity = newVelocity;
}
```

У випадку поразки гравця в грі, в залежності від причини смерті, викликаються два методи: *DieFromEnemy* і *DieFromDrowning*. Відповідно до вибраного методу, відтворюється відповідний звук і виконується метод Die, який зупиняє гру, зберігає кількість зібраних монет і викликає подію *OnPlayerDie*. Коли подія *OnPlayerDie* викликається, відображається вікно з повідомленням про програш.

3.5 Горизонтальна прокрутка сцени

Для переміщення блоків, які генеруються у світі за допомогою процедурного генератора, використовується ігровий об'єкт *World*. Ці блоки будуть пересуватися з визначеною швидкістю. Клас *WorldMoving* створений з метою забезпечення переміщення кожного блоку всередині об'єкта *World* за допомогою відповідного скрипту. Ось параметри цього класу:

– *Player* – посилання на об'єкт гравця;

– *Speed* – поточне значення швидкості. Його значення можна отримати, але змінити поза межами класу неможливо;

– *PassedDistance* – значення пройденої відстані;

– *GrowingSpeedMultiplierPerSecond* – число, на яке збільшиться множник швидкості за секунду гри.

Всі дії з переміщенням блоків відбуваються в методі *FixedUpdate*, оскільки використання методу *Update* може призвести до помилок в фізичних взаємодіях.

Швидкість пересування блоків збільшується до 10 одиниць, після чого вона стає постійною. Також, при збільшенні швидкості, збільшується анімація бігу гравця.

Фрагмент з переміщенням блоків показано в лістингу 3.4.

Лістинг 3.4. Метод *FixedUpdate* в класі *WorldMoving*

```
private void FixedUpdate()
{
    if (!player.IsPlayingGame) { return; }
    var children = transform.GetComponentsInChildren<Transform>();
    foreach(var obj in children)
    {
        obj.transform.Translate(Vector3.left*Time.fixedDeltaTime *
         speed);
    }
    if (speed < 10)
    {
        speed += Time.fixedDeltaTime *
         growingSpeedMultiplierPerSecond;
    }
    player.ChangeAnimationSpeed(speed/4);
    PassedDistance += speed * Time.fixedDeltaTime;
}
```

3.6 Розробка процедурного генератора рівня

3.6.1 Блоки для процедурної генерації

Для процедурної генерації рівня було вирішено використовувати чотири категорії блоків: статичні блоки (*Block*), блоки ворогів (*EnemyBlock*), блоки

великих декорацій (*HugeDecoration*) та блоки монет (*CoinBlock*). Префаби, які є об'єктами даних типів, зображені на рисунку 3.11.

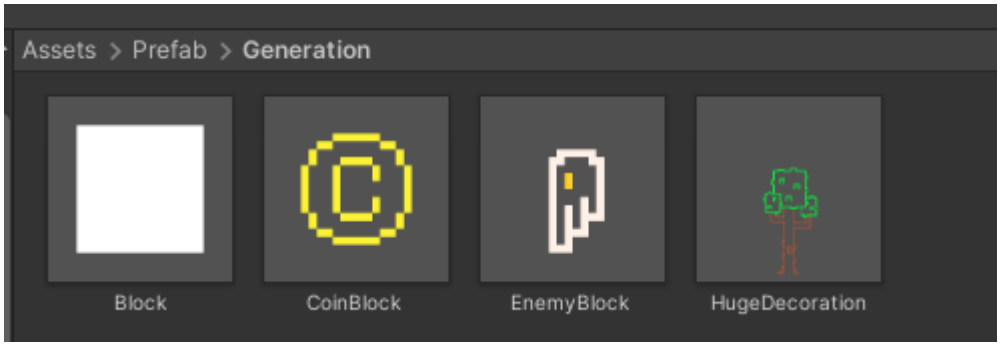


Рис. 3.11 – Префаби блоків процедурного генератора

Типи блоків мають компонент, що успадковується від класу *BlockBase* для універсалізації операцій в генераторі. Це дозволяє маніпулювати блоками при умові, коли неважливо знати який блок використовується. Всі типи блоків мають такі компоненти як: *SpriteRenderer* для зміни спрайту та *Transform* для зміни позиції блоку.

Кожен тип блоку виконує свої функції у генераторі, тому слід розглянути кожен з них окремо.

Тип *Block* у процедурному генераторі використовується для наступних дій:

– створення блоків поверхні;

– створення блоків води;

– створення малих декорації.

Блок поверхні, на відміну від блоків води та декорацій, має тверду поверхню. Саме для цього, тип *Block* має додатковий параметр *Collider*, що зберігає посилання на компонент *BoxCollider2D*, який дозволяє легко вимикати та вмикати колайдер, в залежності від підвиду блоку. Приклади блоків, якими може стати тип *Block* зображений на рисунку 3.12.

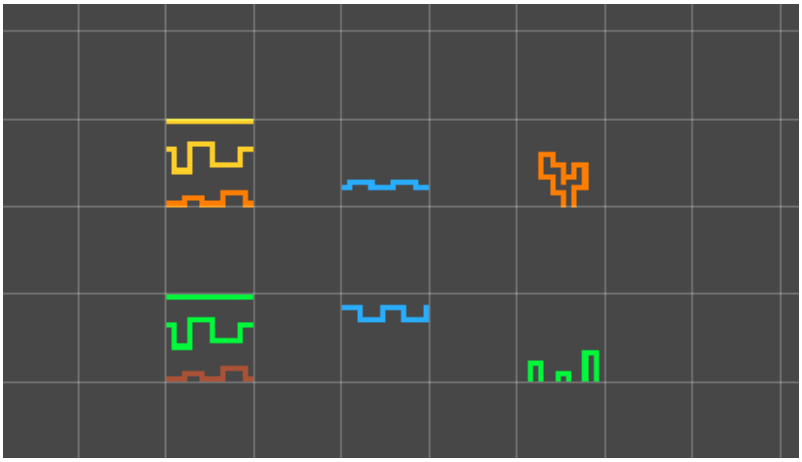


Рис. 3.12 – Приклад використання блоків типу *Block*

Тип *EnemyBlock* потрібний для створення ворогів. Окрім загальних параментрів у ворога присутні анімації та розподіл ворогів на види. В класі *EnemyBlock* присутній параметр *Type*, що зберігає дані про вид ворога, в який перетворився блок. Загалом є три види ворога: курка, привид та літаючий жук. Для анімацій клас *EnemyBlock* має параметр *Animator*, в який імпортується контролер аніматора, який запускає анімації у ворога. У випадку, якщо ворог літаючого типу, в даному випадку їх двоє (привид та літаючий жук), то є ймовірність додавання компоненту вертикального повітряного пересування на об’єкт та зберігання його посилання в параметр *VerticalMovement*, який по стандарту порожній.

Приклад створених ворогів зображений на рисунку 3.13.

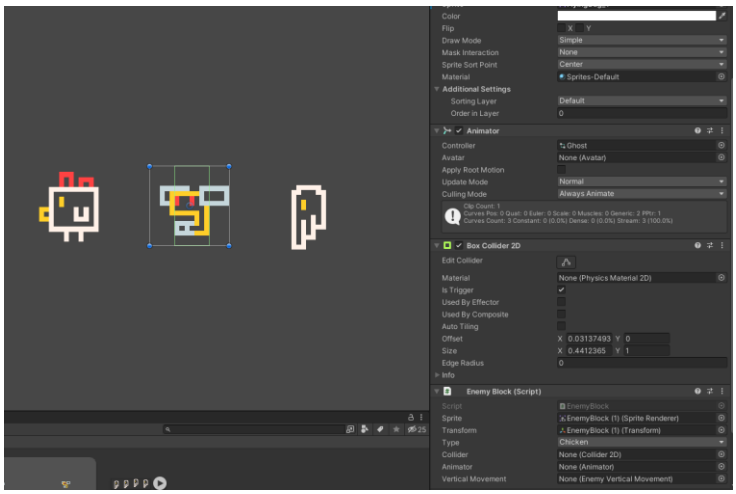


Рис. 3.13 – Приклад ворога «Літаючий жук», що має штучний інтелект вертикального руху

Тип *HugeDecoration* створений для розташування декорацій, що більші ніж стандартний блок. Вигляд таких блоків зображений на рисунку 3.14.

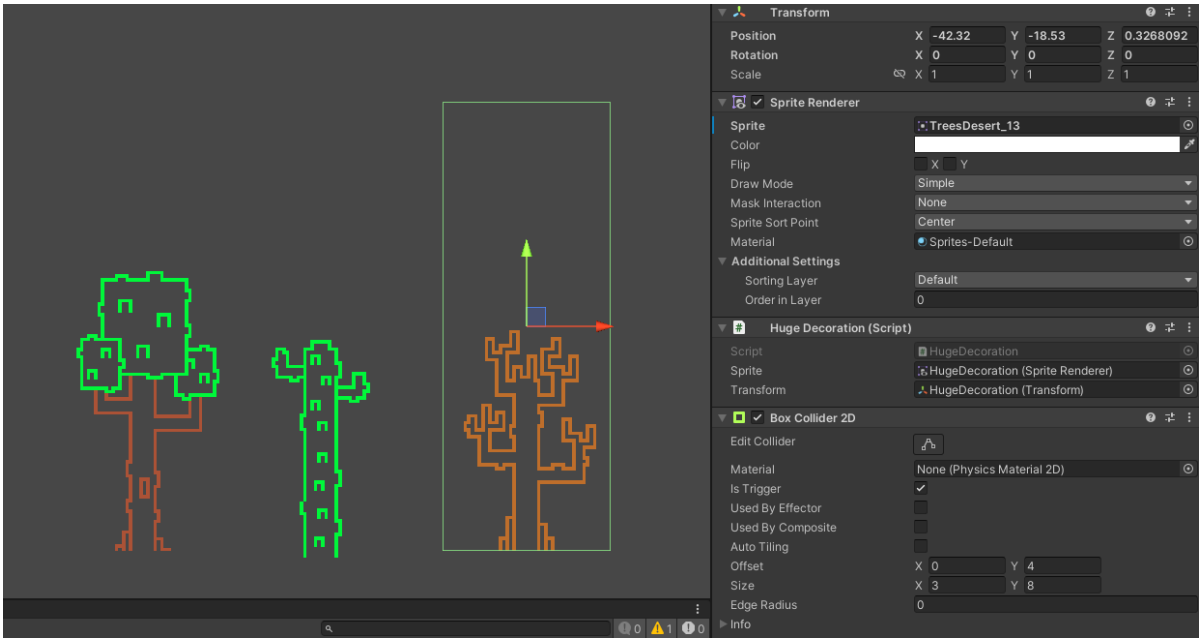


Рис. 3.14 – Вигляд блоків типу *HugeDecoration*

Блоки типу *CoinBlock* потрібні для розташування монет по рівню. При підбиранні монети гравець заробляє одну монету. Зображення монети показане на рисунку 3.15.

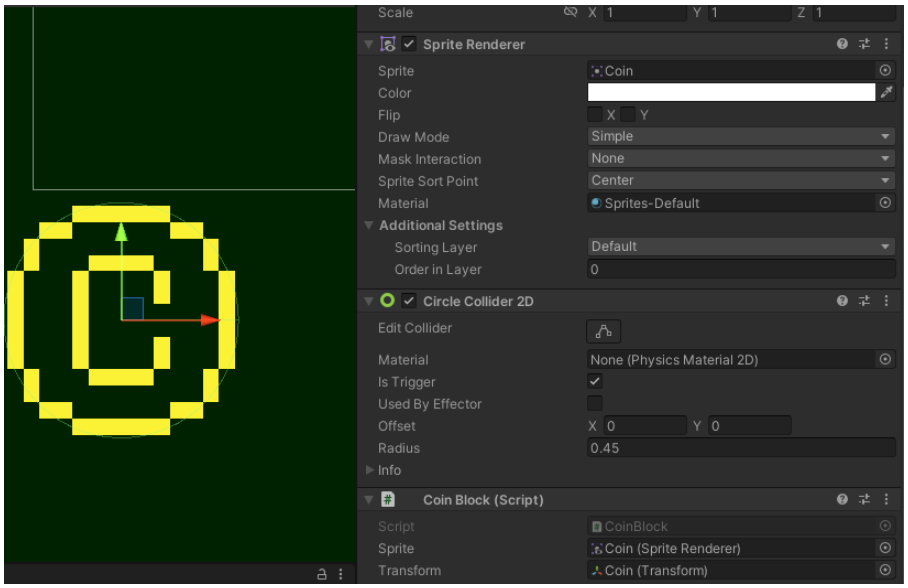


Рис. 3.15 – Зображення блока типу *CoinBlock*

3.6.2 Процедурний генератор

Процедурний генератор розташовує блоки у світі, створюючи динамічні на різноманітні рівні. Загальний вигляд всіх об'єктів на сцені під час роботи генератора зображено на рисунку 3.16.

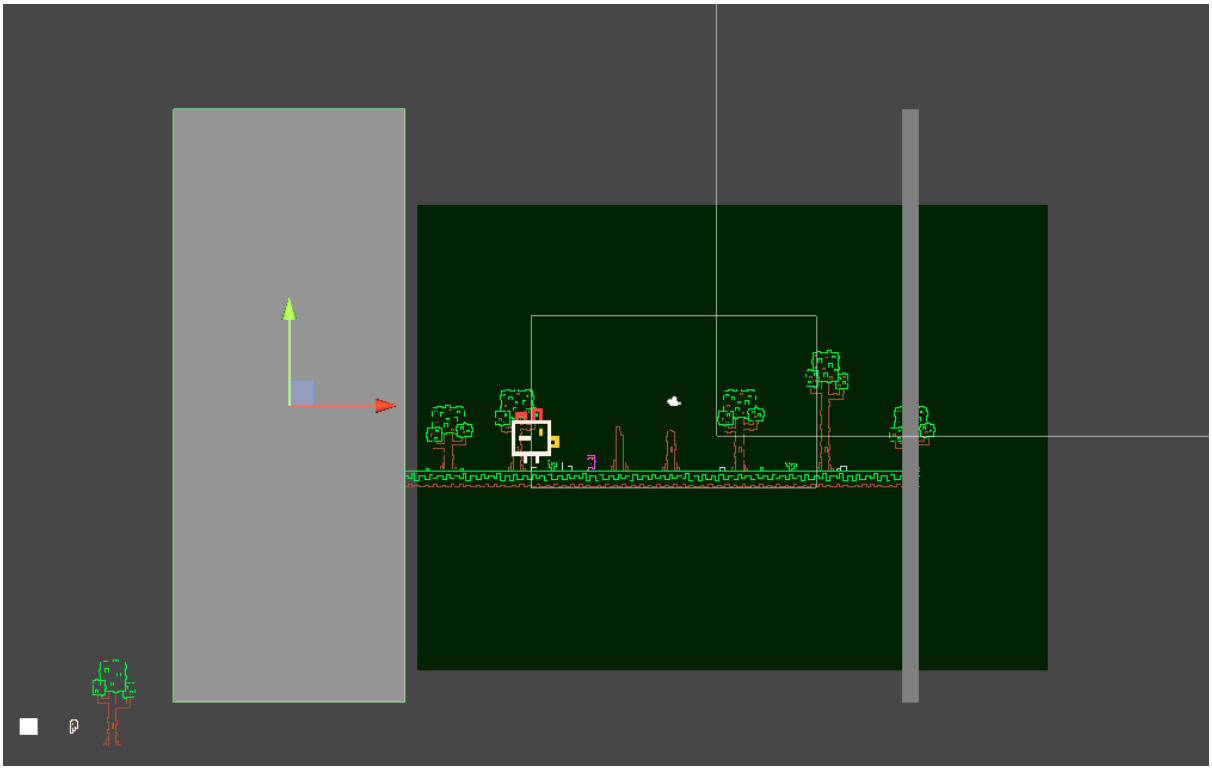


Рис. 3.16 – Зображення роботи процедурного генератора під час гри

Генератор може створити наступні елементи світу:

– блоки поверхні;

– малі декорації;

– великі декорації;

– монети;

– структури «Річка»;

– вороги.

Загальний вигляд компоненту зображено на рисунку 3.17.

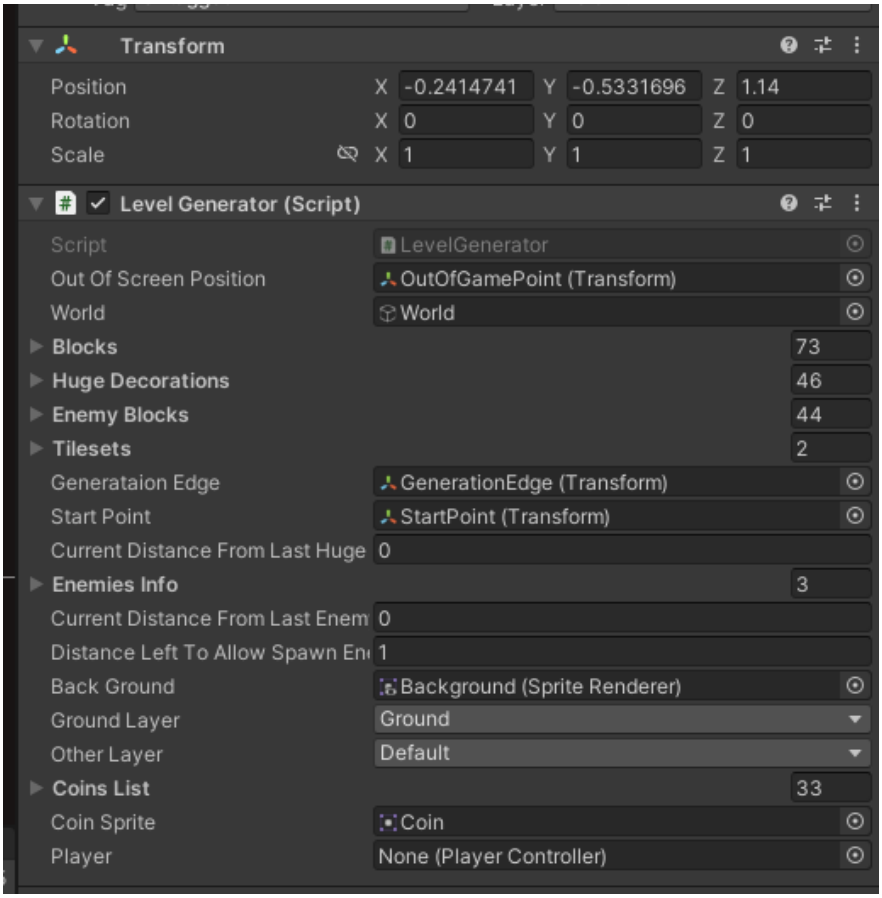


Рис. 3.17 – Компонент *LevelGenerator*

Кожна дія можлива тільки тоді, коли виконані всі умови та випадає ймовірність на створення відповідного елемента.

Робота генератора включає в себе два підходи до процедурної генерації: функціональний та планувальний. Також генератор поділяється на два типи: створення початку рівня та подальшу роботу генератора.

Для початку роботи генератора, а саме створення перших тридцяти блоків поверхні, потрібно мати інформацію про поточну локацію та місця, звідки почати генерувати рівень. Для збереження даних про локацію було створено клас *TileSet*, який успадковується від вбудованого в редактор *Unity* класу *ScriptableObject*, що дозволяє створювати об’єкти класу як файли проекту, що спрощує збереження даних. В генераторі існує масив доступних для генерації локацій під назвою *Tilesets*.

TileSet має наступні параметри:

– *Location* – вид локації;

– *Ground* – масив спрайтів звичайних блоків поверхні;

– *GroundRightEdge* – спрайт блоку кінця поверхні;

– *GroundLeftEdge* – спрайт блоку початку поверхні;

– *Water* – масив спрайтів блоку води;

– *Decoration* – масив спрайтів малих декорацій;

– *Huge* – масив спрайтів великих декорацій;

– *BackgroundColor* – колір заднього фону.

Генератор може отримати будь-який параметр з даних локації за допомогою методів, що випадковим чином видають елемент масиву з відповідного параметру, або напряму з параметра, взявши необхідний спрайт. Методів для отримання спрайтів чотири: *GetRandomGround* (дає випадковий блок поверхні), *GetRandomWater* (дає випадковий блок води), *GetRandomDecoration* (дає випадкову малу декорацію) та *GetRandomHugeDecoration* (дає випадкову велику декорацію).

Для вказівника початку генерації в сцені було створено об’єкт *StartPoint*. Його вигляд показаний на рисунку 3.18. Стрілки осей, що зображені на рисунку 3.18 вказують на позицію об'єкта.

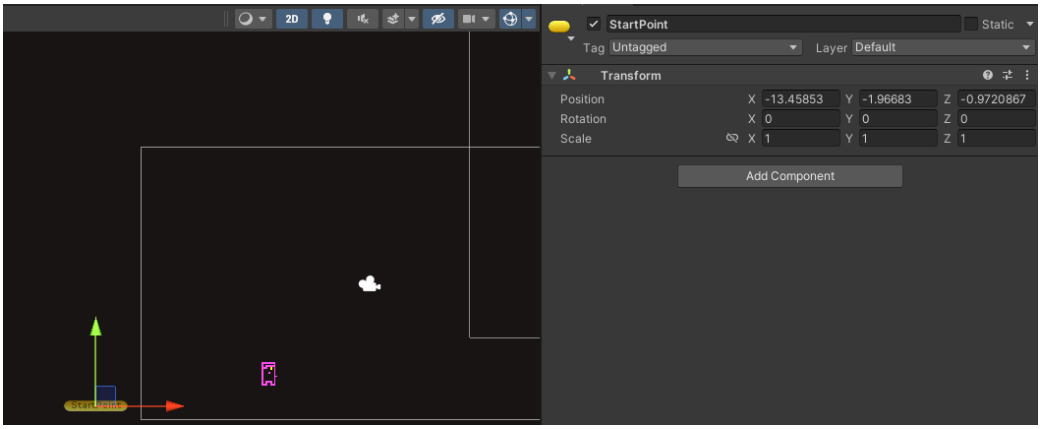


Рис. 3.18 – Вигляд об’єкта *StartPoint*

Всі блоки, що використовуються в об’єкті розташовані в структурі «Список» як параметр генератора рівня для кожного типу, а саме: *Blocks*, *HugeDecoration*, *EnemyBlocks* та *Coins*.

Для повторного використання блоків було створено структуру даних *PoolObject* [40], що дозволяє забирати вільні, та додавати щойно звільнені блоки.

Власна реалізація класу *PoolObject* заснована на структурі «Черга», що дозволяє використовувати об’єкти по принципу черги: перший ввійшов, перший вийшов. При цьому об’єкт що вийшов з черги видаляється з неї. В цьому об’єкті присутні два методи: для додавання об’єкта в чергу(*AddItem*) та для отримання предмета з неї (*PickAvailableItem*). Так як дана структура потрібна для збереження різних даних, для оптимізації розробки було вирішено зробити його загальним (*generic*), що вимагає уточнення типу даних, який буде використовуватися в *PoolObject*. Приклад створення такого класу показаний в листінгу 3.5.

Лістинг 3.5. Використання структури *PoolObject*.

```
private PoolObject<CoinBlock> coinPool = new PoolObject<CoinBlock>();
coinPool.AddItem(coin);
coinPool.PickAvailableItem();
```

На початку гри процедурний генератор має режим роботи створення початку рівня, проте спочатку він заповнює всі об’єкти для зберігання блоків: *BlocksPool*, *EnemiesPool*, *HugeDecorationPool* та *CoinsPool*, після чого обирає випадкову локацію та зберігає посилання на поточну локацію в окремий параметр *CurrentTileSet*. З планувального підходу генератору відомо, що потрібно створити тридцять блоків поверхні, причому він може додатково створювати лише декорації. Генератор створює лише один блок поверхні за раз. При кожному створенні блоку поверхні функціональним підходом вирішується зовнішній вигляд блоку та необхідність створення декорацій. В генерації великі та малі декорації не можуть бути розташовані в одному місці. Приклад згенерованого початку рівня показаний на рисунку 3.19.

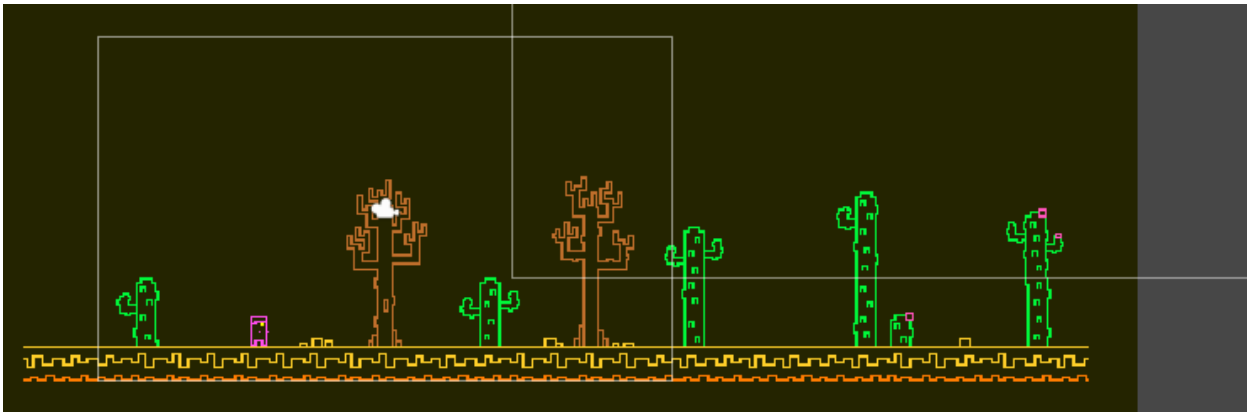


Рис. 3.19 – Приклад рівня, створеного процедурним генератором в режимі створення початку рівня

Після створення початку рівня процедурний генератор автоматично переходить в режим створення основного рівня. Даний режим роботи проходить в методі *FixedUpdate*, що викликається редактором *Unity* постійно через фіксований проміжок часу. Важливо зазначити, що циклом роботи генератора слід вважати лише ті виклики метода *FixedUpdate*, в яких був створений блок поверхні.

Для роботи генератора в режимі створення основного рівня потрібні додаткові параметри, а саме:

– *GenerationEdge* – це об’єкт в сцені, що вказує докуди буде працювати генератор (рис 3.20).

– *EnemiesInfo* – масив даних про ворогів. Зберігає дані про вид та тип ворога та анімацію (рис 3.21).

– *ChanceToSetWater* – ймовірність на створення структури «Річка»

– *ChanceToSpawnDecoration* – ймовірність на створення малої декорації

– *ChanceToSpawnHugeDecoration* – ймовірність на створення великої декорації

– *ChanceToSpawnEnemy* – ймовірність на створення ворога

– *ChanceToBeInAir* – ймовірність на створення повітряного ворога

– *ChanceToBeMovableInAir* – ймовірність на додання інтелекту вертикального пересування ворогу

– *ChanceToSpawnCoin* – ймовірність на створення монети

– *ChanceToChangeTileSet* – ймовірність змінити локацію.

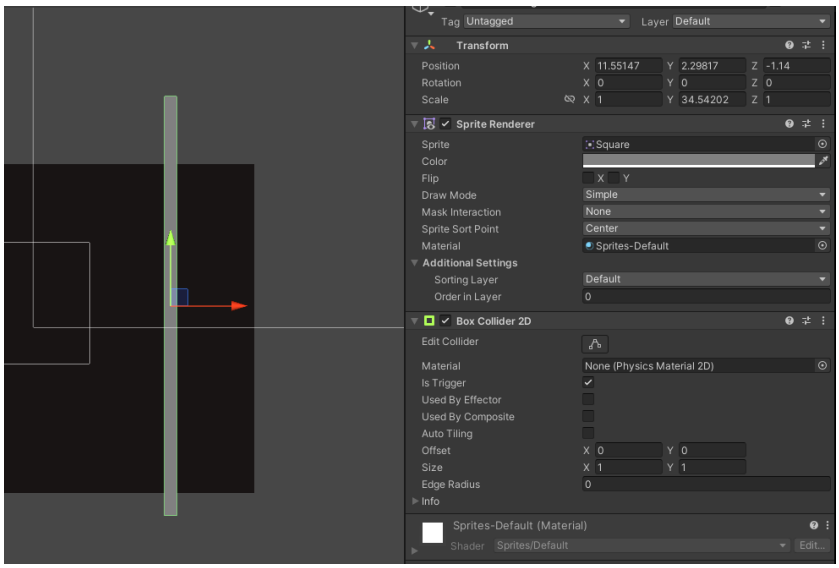


Рис. 3.20 – Вигляд об’єкту *GenerationEdge*

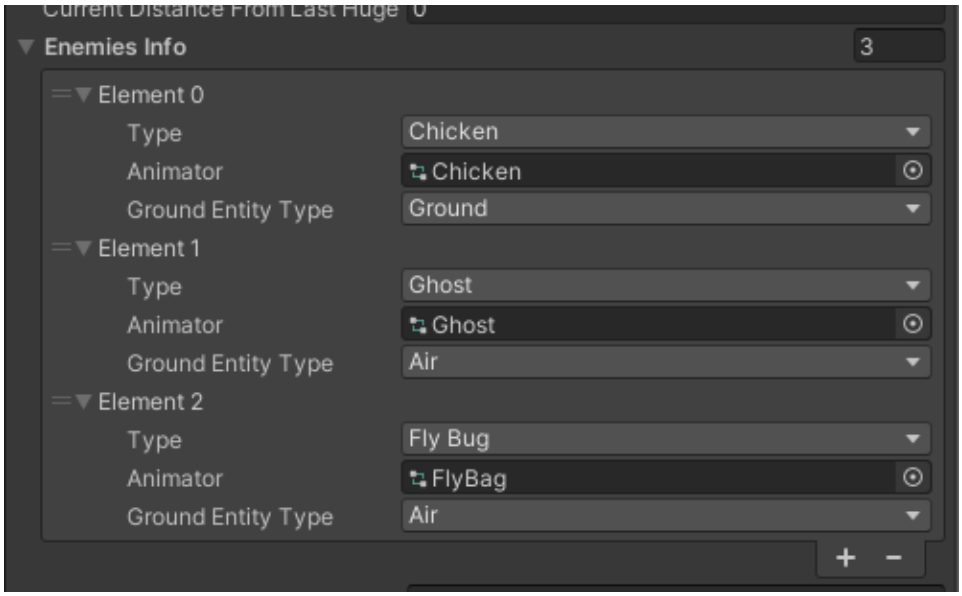


Рис. 3.21 – Вигляд параметра *EnemiesInfo*

Як і в режимі створення початку рівня, режим генерації основної частини створює один блок поверхні за один цикл. Створений блок поверхні стає останнім згенерованим блоком. Після створення поверхні, генератор функціональним підходом обчислює можливість появи додаткових блоків та структур. Важливо зазначити, що за один цикл роботи генератора може з’явитися лише один додатковий блок або одна структура. Робота процедурного генератора показана на рисунку 3.22.

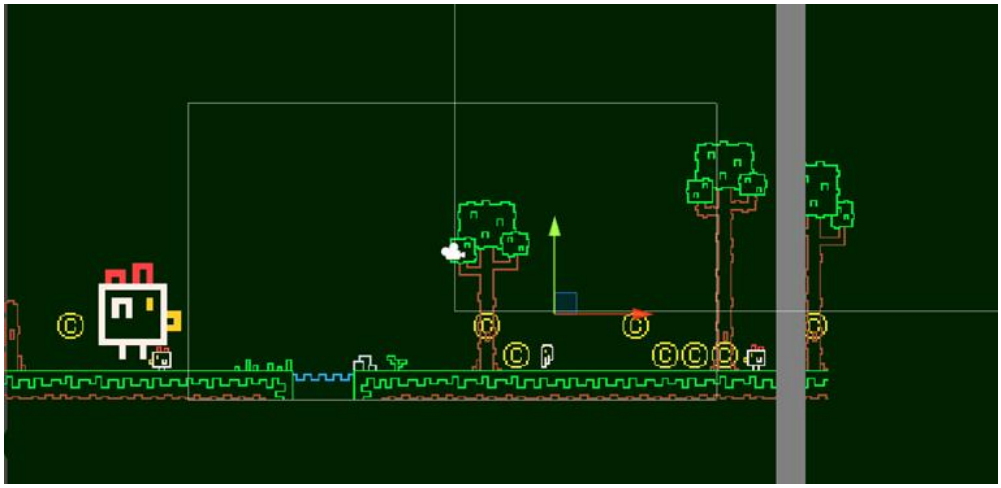


Рис. 3.22 – Зображення роботи процедурного генератора

Як показано на рисунку 3.22, генератор не створює нові блоки, поки за межею генерації знаходяться інші блоки. Циклом роботи генератора вважається тільки той цикл, в якому він створить блок поверхні.

Перш за все обчислюється можливість появи структури «Річка». У випадку, коли структура повинна з'явитися, викликається метод *PlaceWater*. Структура «Річка» складається з: трьох блоків поверхні зліва, випадкової кількості блоків води та трьох блоків поверхні справа. Кількість блоків води варіюється від одного до двох. Даний проміжок поступово збільшується кожної хвилини на 1, тобто через хвилину гри проміжок можливої кількості створеної води стане від одного до трьох. Важливо зазначити, що на блоках поверхні в структурі не з'являються ні вороги, ні монети. Так як річка ділить поверхні на частини, важливо позначити це візуально, для чого використовуються спрайти *GroundRightEdge* та *GroundLeftEdge* з поточної локації для зображення правого та лівого краю поверхні відповідно (рис. 3.23).

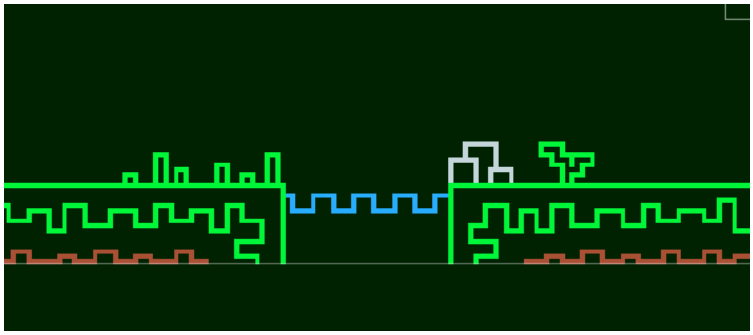


Рис. 3.23 – Приклад генерації структури «Річка»

У випадку, коли структура «Річка» не створюється, функціональним підходом визначається чи може з'явитися ворог. У випадку, коли ворог може з'явитися планувальний метод вирішує чи дозволено створити ворога згідно описаних умов: не менше чотирьох блоків між ворогами та один блок відстані між монетами та ворогом. Кожної хвилини мінімальна відстань між ворогами збільшується на один блок, тобто через хвилину мінімальна відстань між ворогами стане п'ять блоків.

Створення ворога проходить в кілька етапів. Спочатку функціональним способом обирається ворог з масиву *EnemyInfo*. Потім ворог розташовується над останнім згенерованим блоком поверхні, після чого прораховується чи може ворог бути в повітрі та чи був обраний ворог, що може знаходитися у повітрі. У випадку, коли ця умова виконується функціональним способом вирішується чи додавати ворогу інтелект вертикального пересування, після чого у випадку виконання умов ворог розташовується у проміжку пересування випадковим чином, після чого щойно доданий компонент зберігається у об’єкті класу *EnemyBlock*. Інакше, повітряний ворог розташовується над землею таким чином, щоб гравець мав можливість ковзати під ним, при цьому ворог не рухається. Всі надземні вороги завжди розташовані як статична перешкода над землею. Приклад створення ворогів зображений на рисунку 3.24.

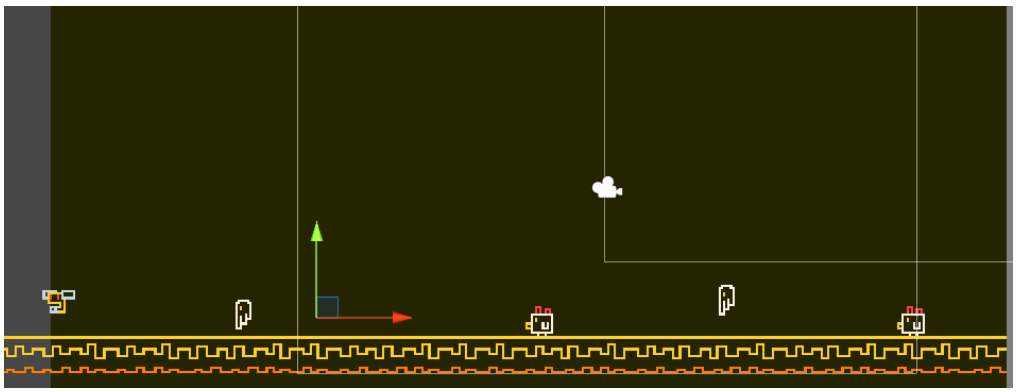


Рис. 3.24 – Приклад генерації ворогів в рівні

У випадку, коли ворог не створюється, функціональним способом обчислюється можливість створення малої декорації. При створенні малої декорації обирається випадковий спрайт декорації з поточної локації. Якщо мала

декорація не створюється, то вираховується можливість створення великої декорації. В разі створення великої декорації використовується планувальний підхід щоб вирішити чи слід створювати велику декорацію. Правилом для великої декорації є те, що відстань між великими декораціями повинна бути як мінімум один блок. Приклад генерації декорації зображений на рисунку 3.25.

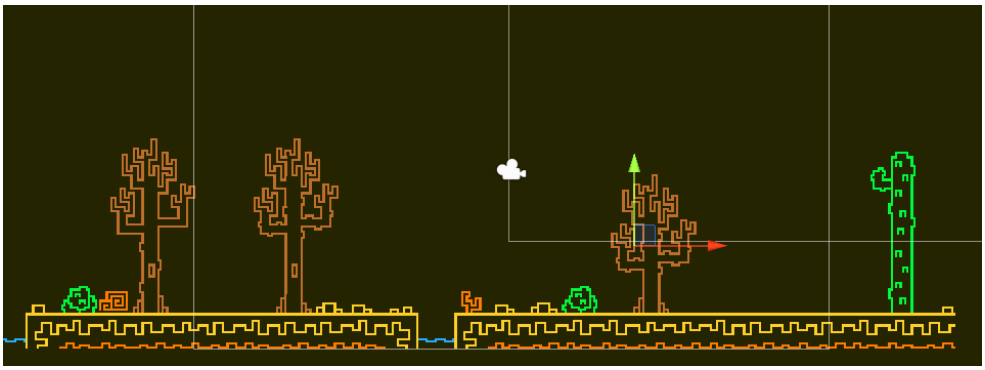


Рис. 3.25 – Приклад генерації декорацій в рівні

Разом з декораціями, прораховується й створення монет на рівні. Віни створюються лише у випадку, коли не створюється структура «Річка» та вороги, проте можуть бути створені разом з декораціями. Випадковим чином вирішується чи буде створюватися монета, а також обирається випадкова висота над блоком поверхні. Приклад згенерованих монет зображений на рисунку 3.26.

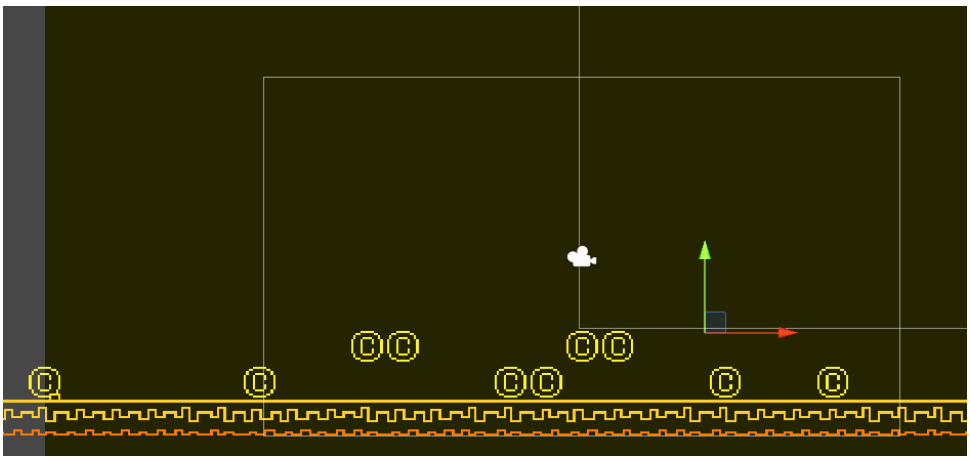


Рис. 3.26 – Приклад згенерованих монет в рівні

Таким чином, процедурний генератор використовує функціональний та планувальний підходи для створення різноманітних рівнів.

Коли генератор розмістить 50 блоків з однієї локації, з’являється ймовірність змінити локацію на випадкову, що перевіряється в кожному циклу.

3.6.3 Повторне використання блоків

Коли блоки проходять повз камеру, вони потрапляють в колайдер об'єкта *BackEdge*. Цей колайдер оброблює блоки, які досягли його, і викликає подію *OnPassingBackEdge* в класі *LevelGenerator* (рис. 3.27).

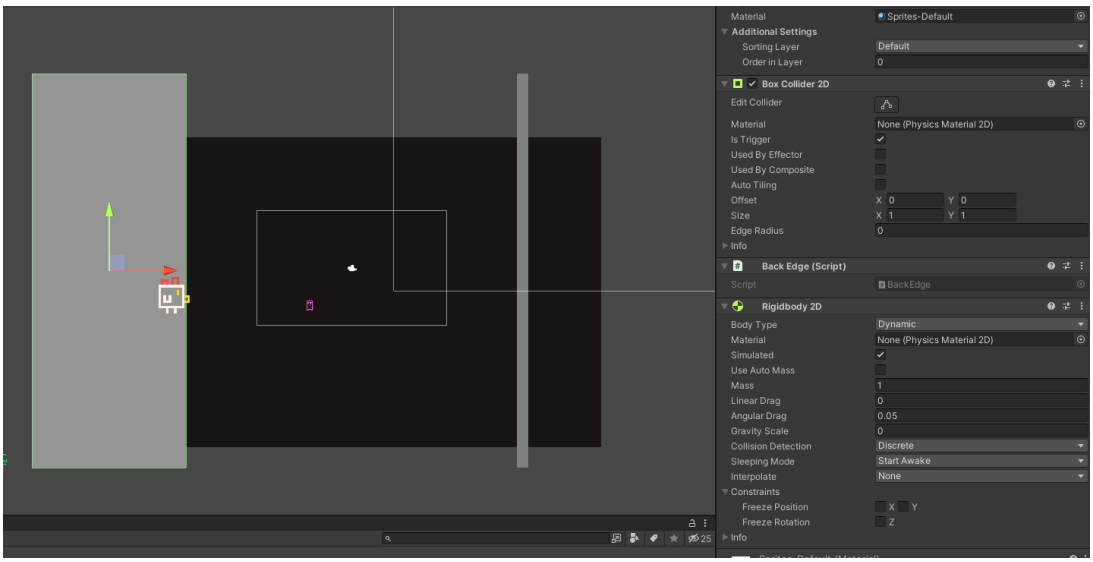


Рис. 3.27 – Зображення об’єкту *BackEdge*

Після того, як скрипт впізнає блоки генерації, вони передаються генератору для повторного використання. Генератор повертає ці блоки до початкового стану, переміщує їх до відповідного місця для готових блоків і додає використані блоки до відповідних структур *PoolObject*. Для блоків ворога генератор додатково знищує компонент вертикального пересування, якщо він є. Фрагмент коду, що описує дії генератора після використання блоків, показаний в лістингу 3.6.

Лістинг 3.6. Метод *OnBlockBecameInvisible* класу *LevelGenerator*.

```
private void OnBlockBecameInvisible(BlockBase block)
{
    block.Transform.parent = null;
    block.Sprite.sprite = null;
```

```
        block.Transform.position = OutOfScreenPosition.position;
        block.gameObject.layer = otherLayer;

        if (block is Block)
        {
            blocksPool.AddItem((Block)block);
        }
        else if(block is HugeDecoration)
        {
            hugeDecorationPool.AddItem((HugeDecoration)block);
        }
        else if(block is EnemyBlock)
        {
            EnemyBlock enemyBlock = (EnemyBlock)block;
            Destroy(enemyBlock.verticalMovement);
            enemiesPool.AddItem(enemyBlock);
        }
        else if(block is CoinBlock)
        {
            coinPool.AddItem((CoinBlock)block);
        }
    }
```

3.7 Інтерфейс гри

Гра починається з головного екрану, де присутній процедурний генератор, що створює безпечну локацію. Усі елементи інтерфейсу початку гри знаходяться в об'єкті *StartGameUI*. На екрані можна побачити наступні елементи:

– *StartText* – повідомлення, що пояснює, як розпочати гру;

– *Coins* – елемент, що відображає кількість монет;

– *MusicToggle* – кнопка, яка дозволяє увімкнути або вимкнути музику;

– *SoundToogle* – кнопка, яка дозволяє увімкнути або вимкнути звуки.

На задньому плані інтерфейсу можна побачити зображення гравця, який стоїть. Зовнішній вигляд інтерфейсу початку гри показаний на рисунку 3.28.

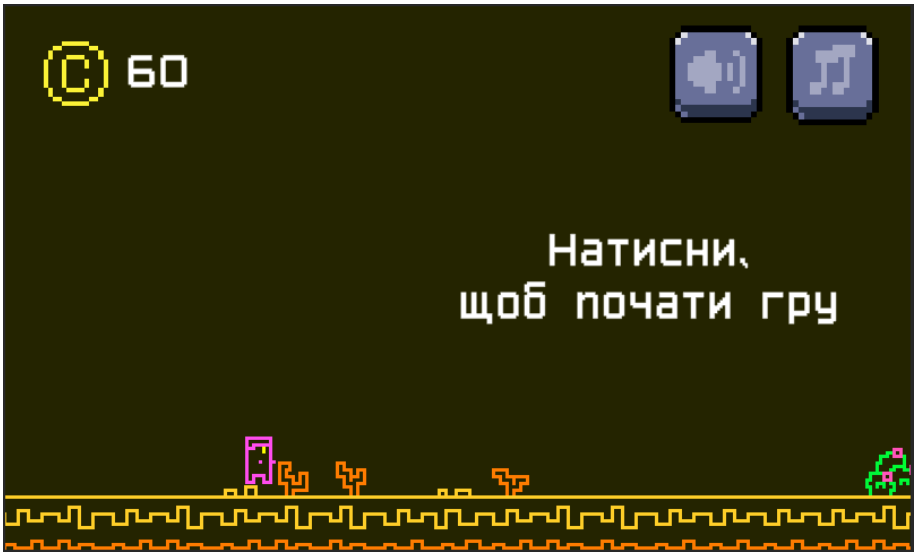


Рис. 3.28 – Головний екран гри

Інтерфейс гри розміщений в об'єкті *Content* і містить наступні елементи:

– *Distance* – відображає інформацію про пройдену відстань гравцем;

– *Coins* – відображає інформацію про поточну кількість монет.

Зображення інтерфейсу гри показане на рисунку 3.29.

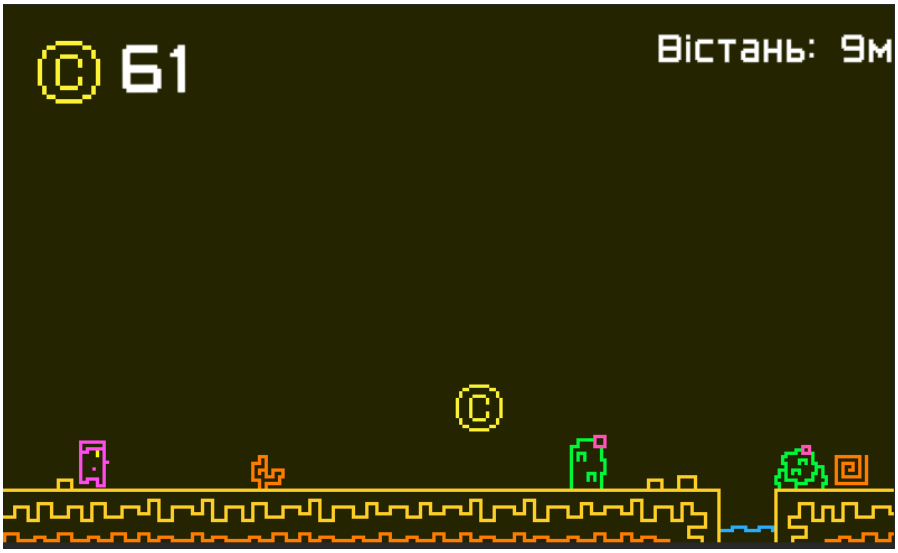


Рис. 3.29 – Інтерфейс гри

Інтерфейс програшу знаходиться в об'єкті *LoosePanel* і складається з наступних елементів:

– *GoToMenu* – кнопка, яка перенаправляє гравця в меню початку гри;

– *MainText* – текст, який повідомляє про програш гравця;

– *MadeDistanceText* – надпис «Пройдена відстань»;

– *MadeDistanceCount* – відображає, скільки відстані пройшов гравець за цю спробу.

Вигляд інтерфейсу програшу показаний на рисунку 3.30

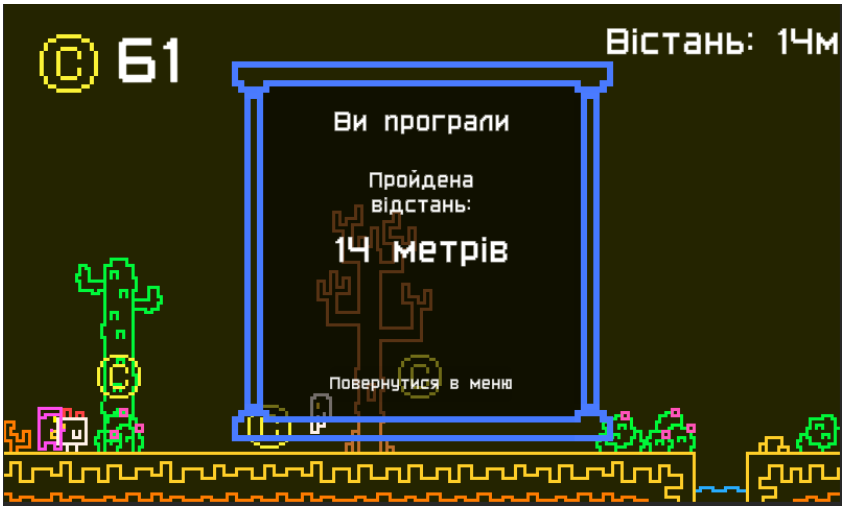


Рис. 3.30 – Інтерфейс екрану програшу

3.8 Кат-сцена

У грі присутня недовготривала кат-сцена, що символізує початок гри. Її сценарій досить простий: до гравця підходить велитенська курка, гравець через секунду починає бігти, а курка переслідує його, після чого камера плавно пересувається направо, щоб гравець був зліва. Даний сценарій був реалізований за допомогою скрипта під назвою *StartGameScenario*.

– *StartGameScenario* має наступні параметри:

– *GameCamera* – ігрова камера;

– *ChickenTransform* – компонент *Transform* у курки;

– *ToChickenPosition* – позиція, куди йде курка по сценарію;

– *ChickenAnimator* – компонент *Animator* у курки;

– *ToCameraPosition* – позиція, куди камера переміститься по сценарію;

– *Player* – гравець;

– *GameUI* – інтерфейс гри.

Сценарій починається при дотику до екрану в головному екрані гри. Зображення об’єкта *StartGameScenario* зображений на рисунку 3.31

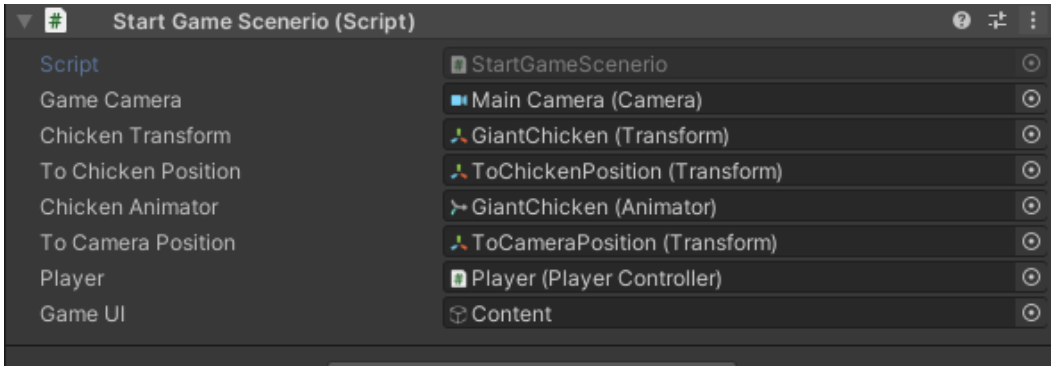


Рис. 3.31 – Компонент кат-сцени

Висновок до третього розділу

1. У ході проведеної роботи було обрано ігровий рушій *Unity 2021.3.7f1*. як засіб для створення гри. Для створення програмного коду гри було обрано редактор *Visual Studio 2022*. Для створення візуальної складової гри було обрано графічний редактор *Aseprite*.
2. Було описано детальну роботу головного персонажу гри. Показані головні компоненти головного персонажу гри та розробка його основних можливостей.
3. Було продемострovано розробку процедурного генератора. Показаний детальний опис роботи процедурного генератора та позначені підходи процедурної генерації, що використовуються.
4. Проект був зкомпільований та при його тестуванні не було виявлено жодних проблем та помилок.

РОЗДІЛ 4
ЕКОНОМІЧНІ РОЗРАХУНКИ ПРОЕКТУ

В умовах відкритої ринкової економіки розширюється діапазон оцінки ефективності науково-технічних розробок, а отже, збільшується кількість основних видів ефективності НДДКР, які необхідно визначити з метою цієї оцінки[41]. До них належать:

1. ***Науково-технічний ефект***. Розробка розширяє поняття процедурної генерації.
2. ***Економічний ефект***. Розробка гри має на увазі отримання швидкого прибутку при закінченні розробки технології. Прибуток, що можна отримати напряму залежить від просування гри та якості зробленої продукції.
3. ***Соціальний ефект***. Створена гра має ряд переваг та недоліків в соціумі. Серед переваг можна вважати хороший спосіб скоротати час та розважальний контент. Серед головних недоліків слід вважати те, що надмірне проводження часу за грою може привести до погіршення здоров’я чи зменшення соціальної взаємодії у реальному житті.
4. ***Маркетинговий ефект***. Ігри подібного жанру популярні у широкій аудиторії, тому дослідження нових механік у іграх жанру «*Runner*» є доцільним.
5. ***Екологічний ефект***. Розробка гри є цілком екологічним, тому що витрачається мінімум природніх ресурсів.

Для того, щоб розрахувати потенційні витрати на гру потрібно дослідити аналоги світового ринку та вирахувати приблизні витрати на розробку, публікацію та рекламу гри.

Майже монополістом на операційній системі *Android* є *Play Market*, що є майже на кожному *Android* пристрої, саме тому він буде досліджуватися. Серед ігор жанру «*Runner*» спостерігається велика кількість одноманітних клонів

оригінальних ігор, з яких брали приклад. З цього можна зробити висновок, що конкуренція велика, тому потрібно використовувати рекламу.

Реклама гри може відбуватися за допомогою платформи *Google Ads*, що є популярним рішенням для компаній. Платформа пропонує низку послуг, серед них: пошукова реклама *Google*, банерна реклама *Google* та відео реклама на платформі *Youtube*. Платформа пропонує гнучкий вибір використання реклами за будь яким бюджетом, проте і якість буде залежати від обраного бюджета, що буде витрачатися протягом місяця. При цьому, кошти витрачаються лише тоді, коли потенційний клієнт перейде по рекламі. Для малої компанії витрачати близько 1000 доларів буде доцільним(36791,56 гривні), проте з ростом компанії, витрати на рекламу також можуть збільшуватися, поки це збільшує прибутки компанії, деякі компанії можуть дійти до витрат в розмірі 10000 доларів на місяць реклами, що станом на 04.05.2023 становить 367915,60 гривні.

Для того, щоб опублікувати гру на платформі *Play Market*, потрібно мати обліковий запис розробника, що коштує 25 доларів(919,79 гривні).

Окрім витрат на рекламу та облікового запису, потрібно оплачувати заробітну плату своїм працівникам. Припустимо, що для роботи над базовим ранером потребується один розробник, один 3д художник та художник інтерфейсів середнього рівня, а гру робити мають на протязі пів року. Їх середня оплата за місяць роботи є 20000, 26500 та 25000 гривні відповідно. За півроку їм необхідно оплатити 429 000 гривні за їх працю.

Для розрахунку витрат власного проекту буде використано середні витрати на 1 розробника, що буде робити гру з нуля, з безкоштовних моделів та зображень на ноутбуці середньої комплектації протягом року та рекламою продукту на протязі двох місяців.

Витратами є:

1. Електроенергія: за рік роботи за середнім ноутбуком буде спожито 297.2 кВт енергії, що буде становити 428.52 гривні.
2. Ноутбук: ноутбук середньої комплектації буде достатньо для розробки гри. Подібний пристрій коштує в середньому 22 000 гривні

3. Інтернет: за рік використання тарифу інтернету в 100 мбіт/с у провайдера Укртелеком буде витрачено 4200 гривні.
4. Заробітна плата працівникам: для того, щоб виконати даний проект достатньо одного розробника рівня *Junior*, що станом на 04.05.2023 буде коштувати в середньому 20 000 гривень в місяць. Зображення та моделі будуть використовуватися безкоштовні, тому оплата за них непотрібна. Так як гру буде розроблювати власник гри плати він не потребує.
5. Обліковий запис розробника на *Play Market*: для розповсюдження гри в світовому ринку вирішено використовувати платформу *Play Market*, яка має величезну аудиторію та є на майже на кожному пристрої з операційною системою *Android*. Для створення облікового запису необхідно заплатити 25 доларів, що станом на 04.05.2023 року становить 919,79 гривні
6. Реклама на платформі *Google Ads* протягом двох місяців. Для поширення гри серед великої кількості ігор буде використовуватися реклама. Так як платформа *Google Ads* має гнучкий бюджет для показу реклами вирішено витратити 500 доларів в місяць, тобто 1000 за два місяці. Це 36791,56 гривні.

Підсумовуючи, загальні витрати на гру на протязі року становлять 47549.65 гривні, при умові, що гру робить власник гри, в іншому випадку витрати становлять 287549.65 грн. Потім після випуску гри необхідно витратити 36791,56 гривні для реклами самої гри.

Науково-технічну ефективність (НТЕ) результатів прикладних робіт визначають на основі показників науково-технічного рівня. Оцінка науково-технічної ефективності НДДКР відбувається на основі показника ($О_{НТЕ}$), який представляє собою ступінь досягнення максимально можливого рівня, значення якого дорівнює 1 (одиниці) [41]:

$$О_{НТЕ} = К^{Ф}_{НТЕ} / К^{П}_{НТЕ} \quad , \qquad (4.1)$$

де $К^{Ф}_{НТЕ}$ – показник (коефіцієнт) фактичного рівня науково-технічної ефективності;

$К^{П}_{НТЕ}$ – показник (коефіцієнт) потенційно можливого рівня науково-технічної ефективності (дорівнює одиниці).

Значення показника $К^{Ф}_{НТЕ}$ визначають на основі шкали експертних оцінок (табл. 4.1).

Таблиця 4.1

Шкала експертних оцінок для виміру рівня науково-технічної ефективності проектів

№	Групи показників	Характеристика показників	Інтервал рейтингового числа	Коефіцієнт значущості показників
1	Науково-технічний рівень	Перевищує кращі світові аналоги	10	0,35
		Відповідає світовому рівню	7 – 9	
		Нижче кращих світових аналогів	5 – 6	
		Перевищує кращі вітчизняні аналоги	3 – 4	
		Відповідає вітчизняному рівню	1 – 2	
		Нижче вітчизняного рівня	0	
2	Перспективність	Першочергова значущість	8 – 10	0,35
		Значущий	5 – 7	
		Корисний	1 – 4	

Продовження Таблиці 4.1

№	Групи показників	Характеристика показників	Інтервал рейтингового числа	Коефіцієнт значущості показників
	Потенційний масштаб практичного використання	Світовий ринок	10	0,20
		Галузі національної економіки		
		Галузь (регіон)		
4	Ступінь вірогідності досягнення позитивних результатів	Великий	10	0,10
		Середній	5 – 9	
		Малий	1 – 4	

Примітка: об'єкт оцінки і аналог(и), які порівнюють за однаковими показниками, наведеними у співставленому вигляді відхилення в значеннях кожного з показників, мають бути однаковими для варіантів, що порівнюються.

Проведення оцінки

Визначають $K^{Ф}_{НТЕ}$ на основі експертної оцінки науково-технічного рівня розробки.

З цією метою:

1. Розроблюють перелік специфічних показників, необхідних для виміру науково-технічного рівня розробки; формують групу аналогів, які реалізовані на світовому і вітчизняному ринках.
2. Здійснюють відповідні розрахунки для співставлення показників і визначення балів по табл. 4.1.

До числа специфічних показників відносять:

1. **Для нової техніки**: продуктивність, споживання інженерних ресурсів на виробітку одиниці продукції, потреба в робочих, які обслуговують обладнання, експлуатаційні витрати на одиницю продукції.

2. **Для нових матеріалів і речовин**: вміст корисних речовин для виробітки готової продукції, питома вага відходів у загальному обсязі переробленої сировини, вартість одиниці нового матеріалу.
3. **Для нових технологій**: якість виробленої продукції, енергоємність і трудомісткість продукції, собівартість одиниці продукції[41].

З метою спрощення визначення $К^{Ф}_{НТЕ}$ у табл. 2 не введено показника витрат на одиницю продукції.

Таблиця 4.2

Порівняльні показники для виконання оцінки НТЕ

ПОКАЗНИКИ	Варіанти технології	
	розробленої	співвідносної (аналога)
Рівень новізни	світовий	-

На основі співставлення даних таблиці встановлюють бали по характеристиках чотирьох груп і на цій основі розраховують значення інтегрального показника НТЕ:

$$НТЕ = \Sigma Б_i \times К_i^3 , \quad (4.2)$$

де $i = 1 \div 4$,

Бі – бали (рейтингове число),

К – коефіцієнт значущості показників.

Рівень науково-технічної ефективності НДДКР розраховано на основі наведених даних прикладу (табл. 4.3).

Таблиця 4.3

Експертна оцінка і розрахунок величини інтегрального показника НТЕ

№	Групи показників	Рейтинг експертів			Середня за експертними оцінками	НТЕ
		1	2	3		
1	Науково-технічний рівень	8	8	8	8	2,8 (8 х 0,35)
2	Перспективність	5	7	6	6	2,1 (1 (6 х 0,35)
3	Потенційний масштаб практичного використання	10	10	10	10	2 (10 х 0,20)
4	Ступінь вірогідності досягнення позитивних результатів	5	5	5	5	0,5 (5 х 0,10)
В С Ь О Г О						7,4

НТЕ = 8·0,35+6·0,35+10·0,2+ 5·0,1 = 2,8+2,1+ 2 +0,5 = 7,4

Отриманий результат слід порівняти з максимально можливим значенням, яке дорівнює 10 балам (10·0,35 + 10·0,35 + 10·0,2 + 10·0,1).”[41]

Отже, оцінка рівня НТЕ може бути зроблена за допомогою інтегрального коефіцієнта оцінки НТЕ ($К_{НТЕ}$):

$$К_{НТЕ} = \frac{НТЕ}{10} \cdot 100\ \% .$$

На основі даних табл. 3.3 можно дійти до висновку, що $К_{НТЕ}$ відповідає 74.0 %, тобто:

$$\frac{7.4}{10} * 100\% = 74{,}0\ \%.$$

В тому випадку, коли значення $K_{НТЕ}$ перевищує середнє значення, яке дорівнює 5,0, має бути зроблено висновок про достатній рівень НТЕ:

– цілком достатній 5,0 – 6,0;

– достатній 6,1 – 8,0;

– достатньо високий 8,1 – 9,0;

– високий 9,1 – 10.

Таким чином, рівень НТЕ технології можна визнати достатнім. Отже, розроблену технологію пропонується впроваджувати у виробництво.

Висновки до четвертого розділу

1. Було проведено оцінку розробки аналогів, обчислено вартість власної розробки та обчислено науково технічку ефективність розробки гри жанру «Runner» з процедурною генерацією рівнів.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Вимоги безпеки при виконанні робіт на робочому місці

При виконанні робіт на робочому місці програміста-розробника потрібно дотримуватись встановлених вимог безпеки праці, а саме – Нормативно-правові акти з охорони праці (НПАОП) 0.00-7.15-18 «Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

НПАОП 0.00-7.15-18 – замінює НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», затверджені наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 №65. Затверджено наказом Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 №207 та Зареєстровано в Міністерстві юстиції України: 25.04.2018 за № 508/31960.

5.1.1 Загальні положення

Згідно з НПАОП 0.00-7.15-18 було встановлено такі загальні положення:

1. Не має значення яка форма власності, організаційно-правова форма і який вид діяльності встановлений на суб'єкті. Вимоги, щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями будуть поширюватися на них всі не зважаючи на їх типи та моделі.
2. Вимоги безпеки і захисту здоров'я та життя працівників під час роботи з екранними пристроями не обмежують права роботодавця встановлювати більш жорсткі та спеціальні вимоги, якщо це не суперечить чинному законодавству.

3. Вимоги не поширюються на:

– робочі місця, які використовуються для навчання у освітніх закладах;
– робочі місця працівників, які виконують управління транспортними засобами (водії, пілоти оператори транспортних засобів);
– робочі місця працівників, які займаються ремонтом та налагодженням екранних пристроїв.

Значення вживаних термінів:

– екранні пристрої – засоби відтворення будь-якої графічної інформації;
– робоче місце – сукупність приладів, що включає екранний пристрій, який може доповнюватись іншими пристроями та програмним забезпеченням.

5.1.2 Вимоги безпеки до робочих місць працівників з екранними пристроями

На підставі НПАОП 0.00-7.15-18 було встановлено такі вимоги безпеки до робочих місць працівників з екранними пристроями:

1. Робочі місця повинні бути спроектовані так, щоб працівники мали простір для руху та зміни робочого положення.

2. Усе випромінювання має бути зведено до граничного допустимого рівня.

3. Організація робочого місця працівника з екранними пристроями має забезпечувати відповідність усіх психофізіологічних вимог, антропологічних та ергономічних вимог з урахуванням виконуваних робіт.

4. Освітлення робочого місця працівника з екранними пристроями має відповідати вимогам Державних санітарних правил і нормам(ДСанПіН) роботи з візуальними дисплейними терміналами електронно-обчислювальних машин 3.3.2.007-98 та створювати відповідний контраст між екраном і навколишнім середовищем.

5. Мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень Державні санітарні норми(ДСН) 3.3.6.042-99.

6. Робоча поверхня повинна мати достатній розмір для гнучкого розміщення екрана, клавіатури, документів і відповідного устаткування та мати поверхню з низькою відбивною здатністю.

7. Робоче крісло повинно бути стійким та дозволяти працівнику легко рухатись і займати зручне положення. Сидіння повинне мати регулювання по висоті та нахилу, також за потрібності передбачити наявність підніжки.

5.1.3 Мінімальні вимоги безпеки під час роботи з екранними пристроями

Згідно з НПАОП 0.00-7.15-18 було встановлені такі мінімальні вимоги безпеки під час роботи з екранними пристроями:

1. Потрібно щодня проводити прибирання та очищення робочого місця та самого екранного пристрою.
2. Після завершення роботи екранні пристрої необхідно відключати від електричної мережі.
3. У разі виникнення аварійної ситуації необхідно негайно відключити усі екранні пристрої та усі електронні прилади від електричної мережі.
4. Не допускається:
   - виконання технічних робіт (ремонт, обслуговування, налагодження) з екранними пристроями підчас роботи та на робочому місці працівника;
   - відключати захисні пристрої, самотужки проводити технічні роботи та зміни у конструкції екранних пристроїв;
   - працювати з несправними екранними пристроями та пристроями, які мають нестабільне зображення та сигналізують про несправність.

5. Підчас виконання робіт у приміщеннях повинні дотримуватись оптимальні умови мікроклімату відповідно до вимог Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99.

5.1.4 Мінімальні вимоги безпеки до екранних пристроїв

На підставі НПАОП 0.00-7.15-18 було встановлено такі мінімальні вимоги безпеки до екранних пристроїв:

1. Екранні пристрої не мають бути джерелом ризику для працівників.

2. Усе випромінювання, за винятком видимої частини електромагнітного спектра, має бути зведене до незначного рівня з погляду безпеки і охорони здоров’я працівників.

3. Між символами і рядками символів має бути належна відстань та символи на екранних пристроях мають бути чіткими й дотримуватись відповідного розміру.

4. Зображення на екрані повинно бути чітким та стабільним, без ознак неправильної роботи екранного пристрою.

5. Яскравість та контрастність повинні мати змогу бути налаштованими працівником підчас роботи з екранним пристроєм та швидко адаптуватись до навколишнього середовища.

6. Обираючи екранний пристрій потрібно враховувати, що він повинен мати змогу налаштування працівником нахилу та висоти екрану.

7. За необхідності може використовуватись регульований стіл або підставка для екрану.

8. Екран не повинен відбивати світло чи відблискувати під час роботи працівника за екранним пристроєм, щоб не викликати дискомфорту.

9. Вибираючи клавіатуру, слід надавати перевагу клавіатурам, які відокремлені від екранного пристрою (автономні) та мають функції налаштування висоти, щоб працівник мав змогу зайняти зручну робочу позу й уникнути втоми рук.

10. Поверхня клавіатури не повинна відблискувати чи відбивати світло, щоб уникнути віддзеркалювання. Клавіши та розташування клавіш повинні полегшувати роботу з клавіатурою. Усі клавіши повинні бути позначені контрастними кольорами та бути розбірливими.

11. Прилади, які входять до робочої станції не повинні виділяти надлишкового тепла для комфортної роботи працівника.

12. Під час роботи з екранними приладами, роботодавець повинен надавати таке програмне забезпечення, яке відповідає розв'язуваним задачам і є простим у використанні, а де необхідно адаптованим під працівника.

5.1.5 Правила сидіння за комп'ютером

При тривалому користуванні комп'ютером слід дотримуватися правил користування комп'ютером:

1. Сидіти треба глибоко на твердому стільці з високою спинкою, що має вигин для попереку, – це вирівняє спину і дасть підтримку шиї. Край стільця не повинен тиснути на судини під колінами.

2. Відстань до монітора повинна бути 50-70см.

3. Використовувати мишку відповідних розмірів, зручної форми.

4. Робити перерву в сидячій роботі, вставати і ходити 15-20 хвилин кожні 1-2 години.

5. Правильно організуйте освітлення робочого місця. При слабкому світлі очі напружуються і болять. Стримайте яскравість екрану. Літери і цифри на екрані це маленькі світлові промені, які йдуть прямо в очі. Потрібно відрегулювати їх контрастність, щоб світло не був дуже яскравим.

6. Час від часу відводите очі вбік, щоб дати відпочити своєму зору.

7. Переміщайте погляд по всій площі екрану, намагайтеся не дивитися напружено в одну точку. Нехай поперемінно працюють всі м'язи очей, а не окремі групи, на які в цьому випадку буде падати максимальне навантаження.

Приклад сидіння за комп'ютером представлений на рисунку 5.1

Рис. 5.1 – Зображення правильного сидіння за комп’ютером

5.2 Шкідливі виробничі фактори на робочому місці

5.2.1 Характеристика шкідливих факторів на робочому місці

У зв'язку із процесом глобальної комп'ютеризації в житті людини виникають питання про вплив комп'ютера на його здоров'я. Існує перелік основних шкідливих факторів, що діють на людину за комп'ютером:

– підвищене навантаження на зір;

– електромагнітне випромінювання;

– сидяче положення протягом тривалого часу;

– перевантаження суглобів кистей;

– комп'ютер – серйозне джерело алергенів;

– вплив комп'ютера на психічне здоров'я.

Отже трохи детальніше розглянемо ці шкідливі фактори й розглянемо рекомендації, щодо усунення їх або максимальне зниження їх впливу на працівників [42]:

1. Постійна напруга зору неминуча при тривалій роботі на комп'ютері. При роботі за комп'ютером очі постійно дивляться в одному напрямку, м'язи, що управляють очними яблуками, слабшають, також працюючи за монітором, людина набагато рідше моргає, що призводить до пересушування кон’юнктиви

ока. У результаті робота на комп'ютері серйозно перевантажує наші очі. У користувача погіршується зір, очі починають сльозитися, з'являється головний біль, стомлення, двоїння зображення.

Для зменшення втоми очей рекомендується використання правильного освітлення – гарне природнє освітлення, у темний час доби лампа повинна освітлювати тільки кімнату, але не екран монітора – це допоможе уникнути відблисків, що ускладнюють роботу. Також важливим фактором збереження здоров’я зору є обмеження часу роботи на комп'ютері та періодичне застосування зорової гімнастики.

Світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати відблисків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк. При цьому освітленість робочих поверхонь столів має становити не нижче 400 лк.

2. Електромагнітне випромінювання є не менш серйозною проблемою. Ввімкнений комп'ютер створює навколо себе поле із широким частотним спектром, який є найнебезпечнішим джерелом електромагнітного випромінювання серед побутових приладів.

По-перше, від екрана йде м'яке рентгенівське випромінювання, по-друге, котушки усередині монітора генерують змінне електромагнітне випромінювання, воно поширюється в основному в різні боки й назад, по-третє, використовувана в електронно-променевих трубках висока напруга призводить до появи електростатичного поля поза монітором.

Усі перераховані вище джерела електромагнітного випромінювання встановлювались у застарілу техніку, тому до рекомендації з максимального зниження випромінювання відноситься – купівля сучасного обладнання та обмеження часу роботи за комп’ютером, потрібно робити перерви підчас роботи за ним.

3. При роботі за комп'ютером користувач сидить в розслабленій позі, однак вона є вимушеною й мало приємною. Довге знаходження у сидячому положенні загрожує різного роду захворюваннями. Одні з них легко

проявляються й досить легко лікуються, а от патологічні захворювання, що відбуваються усередині організму дуже небезпечні. Наприклад, сколіоз при запущеній формі загрожує тим, що, викривляючи хребет, защемляє нервову систему й відбувається зсув внутрішніх органів. Також існує ще багато інших хвороб, які можуть бути викликані сидячим образом життя.

Для того, щоб зменшити шкідливий вплив незручної пози та довгого знаходження у сидячому положенні, необхідно правильно підібрати робочі меблі. Вони допоможуть зберегти правильне положення тіла під час роботи за комп'ютером і зменшити навантаження на м’язи.

Крісло повинне бути на роликах, з регульованою висотою сидіння й спинкою, з підлокітниками та мало змогу обертатись навколо своєї осі та стіл повинен мати спеціальну висувну дошку для клавіатури, або мати достатньо місця для розміщення клавіатури та інших необхідних пристроїв. У перервах між роботою на комп'ютері необхідно влаштовувати розминку.

4. Для роботи з комп'ютером, в основному використовуються комп’ютерна миш й клавіатура, однак ці обладнання змушують людину робити тисячі одноманітних рухів, саме це, у сукупності з постійною напругою м'язів руки, призводить до защемлення нервів в зап'ястному каналі й до болю в зап'ястях. Щоб тривала робота на комп'ютері не призвела до виникнення синдрому зап'ястного каналу, досить виконувати нескладні рекомендації з організації свого робочого місця й режиму роботи, щогодини робити короткі перерви, під час яких виконати кілька вправ для кистей рук.

5. Комп'ютер – серйозне джерело алергенів. Електростатичні заряди, що виникають на моніторі, притягають пил з повітря, цей пил осідає не тільки на екран, але й на особу, яка працює на ньому, а пилу в свою чергу містить велику кількість мікроорганізмів та алергенних часток. Таким чином, багатогодинне сидіння за комп'ютером здатне призвести до поганого самопочуття, сухості шкіри й алергійної реакції.

Основні способи профілактики алергії внаслідок осідання пилу досить очевидні: це підтримка чистоти і порядку на робочому місці, а також зміцнення імунної системи.

6. Робота за комп'ютером – це чисто інтелектуальна праця, тому основна частина навантаження доводиться на нервову систему, а саме на головний мозок. Часто виникають психічні порушення, які є наслідком стресу, частота таких розладів як тривога, дратівливість і пригніченість в працівників, які працюють за комп'ютерами коливається від 25 до 70%. У них частіше, ніж у представників інших професій, спостерігається безсоння і втрата апетиту, можливе виникнення захворювань шкіри обличчя і рук.

До рекомендацій стосовно зменшення навантаження на нервову систему відносяться: прогулянку по свіжому повітрі, спілкування з людьми у реальному світі та також потрібно робити перерви при роботі з комп'ютерною технікою.

5.3 Дії працівників у надзвичайних ситуаціях

5.3.1 Дії працівників при отриманні електротравми.

Так як ми розглядаємо робоче місце працівника, який працює за комп'ютером пропонується розглянути дії працівників у випадку отримання електротравми.

Згідно Порядку надання домедичної допомоги постраждалим при уражені електричним струмом та блискавкою зареєстрованого в Міністерстві юстиції України 7 липня 2014 р. за № 775/25552 послідовність дій надання домедичної допомоги постраждалим при уражені струмом чи блискавкою:

1. Переконатися у відсутності небезпеки.
2. Якщо постраждалий перебуває під дією електричного струму, за можливістю вимкнути джерело струму, відкинути електричний провід або відштовхнути постраждалого за допомогою любого підручного діелектрика.

3. Оглянути постраждалого та з'ясувати чи находиться він у свідомості та має змогу дихати.
4. Викликати медичну допомогу.
5. Якщо постраждалий не дихає почати надавати першу медичну допомогу, а саме серцево-легеневу реанімацію.
6. Якщо постраждалий дихає, проте без свідомості, то потрібно обережно привести його до стабільного положення.
7. Накласти стерильні пов’язки на місця опіків.
8. Забезпечити постійний нагляд за постраждалим до приїзду швидкої допомоги.
9. При погіршені стану постраждалого до приїзду швидкої допомоги повторно зателефонувати диспетчеру швидкої.

5.3.2 Дії працівників при пожежі

На підприємстві при виникненні пожежі дії адміністрації та персоналу слід спрямувати на забезпечення безпеки та евакуації людей.

Кожний працівник, який виявив пожежу, повинен:

1. Негайно повідомити про це по телефону 101 пожежну охорону (при цьому слід указати адресу об’єкта, поверховість будівлі, місце виникнення пожежі, обстановку на пожежі, наявність людей, а також назвати своє прізвище, ім’я та по батькові).
2. Ужити (по змозі) заходів для евакуації людей, гасіння (локалізації) пожежі та збереження матеріальних цінностей.
3. Повідомити про пожежу керівника чи відповідну компетентну посадову особу та (або) чергового по підприємству або організації;при потребі – викликати інші аварійно-рятувальні служби (медичну, газорятувальну і т.ін.).

Посадова особа підприємства, що прибула на місце пожежі, повинна.

1. Перевірити, чи викликана пожежна охорона (продублювати повідомлення), довести до відома власника підприємства про пожежу.

2. У разі загрози для життя людей негайно організувати їх порятунок (евакуацію), використовуючи для цього наявні сили і засоби.

3. Вивести за межі небезпечної зони всіх працівників, не пов’язаних з ліквідацією пожежі.

4. Припинити роботи в приміщенні, крім заходів з ліквідації пожежі.

5. При потребі – відключити електроенергію (за винятком систем протипожежного захисту), зупинити транспортери, агрегати, апарати, перекрити сировинні, газові та парові комунікації, зупинити системи вентиляції в аварійному та суміжних з ним приміщеннях (за винятком пристроїв протидимового захисту) і вжити інших заходів, які сприяють недопущенню розвитку пожежі та задимлення в приміщенні.

6. Перевірити включення повідомлення людей про пожежу, установок пожежогасіння, протидимового захисту.

7. Одночасно з гасінням пожежі організувати евакуацію людей у відповідності до схеми і захист матеріальних цінностей.

8. Забезпечити дотримання вимог безпеки працівниками, які беруть участь у гасінні пожежі.

9. Організувати зустріч підрозділів пожежної охорони, допомогти у виборі найкоротшого шляху для під’їзду до осередку пожежі та в установці на водні джерела.

10. По прибутті на пожежу пожежних підрозділів слід забезпечити їх безперешкодний доступ на територію підприємства.

11. Після прибуття пожежного підрозділу адміністрація та технічний персонал підприємства зобов’язані брати участь у консультуванні керівника гасіння про конструктивні і технологічні особливості підприємства, де виникла пожежа, прилеглих будинків, організувати залучення сил та засобів підприємства до вжиття належних заходів, пов’язаних з ліквідацією пожежі та попередженням її розвитку.

Висновки п’ятого розділу

1. Було описано вимоги до безпеки при виконанні роботи на робочому місці. Вказані мінімальні вимоги щодо роботи за екранними пристроями та описані правила сидіння за комп’ютером.
2. Вказано шкідливі виробничі фактори на робочому місці, вказано причини їх виникнення та наслідки для користувача.
3. Було описано дії працівників при виникненні надзвичайних ситуацій, а саме: дії працівників при отриманні електротравми та дії працівників при пожежі.

ЗАГАЛЬНІ ВИСНОВКИ

1. У результаті аналізу предметної області було виявлено основні характеристики ігор жанру «*Runner*», показано історію його становлення та розвитку. Досліджено основні методи процедурної генерації, а саме: імітаційний, функціональний та планувальний. Вказано для генерації якого контенту частіше всього використовуються перелічені методи. Зазначено, що комбінації різних методів процедурної генерації можуть додатково підвищити рівень варіативності та складності генерованого ігрового контенту.

2. Проаналізовано сучасні аналоги ігор жанру «*Runner*». Виявлено їх переваги та недоліки.

3. Здійснено постановку задачі з зазначенням бажаного результату.

4. Було створено концептуальний документ гри, необхідний для презентації видавцям чи інвесторам.

5. Був розроблений дизайнерський документ гри, що описує основні особливості гри.

6. У ході проведеної роботи було обрано ігровий рушій *Unity 2021.3.7f1*. як засіб для створення гри. Для створення програмного коду гри було обрано редактор *Visual Studio 2022*. Для створення візуальної складової гри було обрано графічний редактор *Aseprite*.

7. Було описано детальну роботу головного персонажу гри. Показані головні компоненти головного персонажу гри та розробка його основних можливостей

8. Було продемострovано розробку процедурного генератора. Показаний детальний опис роботи процедурного генератора та позначені підходи процедурної генерації, що використовуються.

9. Проект був зкомпільований та при його тестуванні не було виявлено жодних проблем та помилок.

10. Було проведено оцінку розробки аналогів, обчислено вартість власної розробки та обчислено науково технічку ефективність розробки гри жанру «Runner» з процедурною генерацією рівнів.

11. Було описано вимоги до безпеки при виконанні роботи на робочому місці. Вказані мінімальні вимоги щодо роботи за екранними пристроями та описані правила сидіння за комп’ютером.

12. Вказано шкідливі виробничі фактори на робочому місці, вказано причини появлення та наслідки для користувача.

13. Було описано дії працівників при виникненні надзвичайних ситуацій, а саме: дії працівників при отриманні електротравми та дії працівників при пожежі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Pong* – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Pong* (останнє звернення 01.06.2023).

2. DOOM – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Doom_(1993_video_game)* (останнє звернення 01.06.2023).

3. Subway Surfers – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Subway_Surfers* (останнє звернення 01.06.2023).

4. Jetpack Joyride – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Jetpack_Joyride* (останнє звернення 01.06.2023).

5. Temple Run – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Temple_Run* (останнє звернення 01.06.2023).

6. *Platform game* – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Platform_game* (останнє звернення 01.06.2023).

7. B.C. Quest for Tires – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/B.C.%27s_Quest_for_Tires* (останнє звернення 01.06.2023).

8. Canabalt – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Canabalt* (останнє звернення 01.06.2023).

9. Micropayment – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Micropayment* (останнє звернення 01.06.2023).

10. *Procedural generation* – Вікіпедія. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Procedural_generation (останнє звернення 01.06.2023).

11. *Survival game* – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Survival_game* (останнє звернення 01.06.2023).

12. Map seed – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Map_seed* (останнє звернення 01.06.2023).

13. Minecraft – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Minecraft* (останнє звернення 01.06.2023).

14. *Borderlands* (video game) – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Borderlands_(video_game)* (останнє звернення 01.06.2023).

15. Spore (2008 video game) – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Spore_(2008_video_game)* (останнє звернення 01.06.2023).

16. The Binding of Isaac: Rebirth – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/The_Binding_of_Isaac:_Rebirth* (останнє звернення 01.06.2023).

17. Just Cause (video game) – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Just_Cause_(video_game)* (останнє звернення 01.06.2023).

18. No Man's Sky – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/No_Man%27s_Sky* (останнє звернення 01.06.2023).

19. Skovbo R. J. Procedural world generation: The simulation, functional and planning approaches – [Електронний ресурс]. – Режим доступу*: https://www.gamedeveloper.com/design/*. (останнє звернення 01.06.2023).

20. The Sims – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/The_Sims* (останнє звернення 01.06.2023).

21. Civilization – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Civilization* (останнє звернення 01.06.2023).

22. Diablo III – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Diablo_III* (останнє звернення 01.06.2023).

23. Despicable Me (film) – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Despicable_Me_(film)* (останнє звернення 01.06.2023).

24. Loot box – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Loot_box* (останнє звернення 01.06.2023).

25. Cookie Run (video game) – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Cookie_Run_(video_game)* (останнє звернення 01.06.2023).

26. Crash Bandicoot – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Crash_Bandicoot* (останнє звернення 01.06.2023).

27. Android (operating system) – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Android_(operating_system)* (останнє звернення 01.06.2023).

28. IOS – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/IOS* (останнє звернення 01.06.2023).

29. Windows Phone – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Windows_Phone* (останнє звернення 01.06.2023).

30. Microsoft Windows – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Microsoft_Windows* (останнє звернення 01.06.2023).

31. Nintendo Switch – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://ru.wikipedia.org/wiki/Nintendo_Switch* (останнє звернення 01.06.2023).

32. PlayStation 4 – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/PlayStation_4* (останнє звернення 01.06.2023).

33. PlayStation 5 – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/PlayStation_5* (останнє звернення 01.06.2023).

34. Xbox One – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Xbox_One* (останнє звернення 01.06.2023).

35. Xbox Series X and Series S – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Xbox_Series_X_and_Series_S*

36. Linux – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/Linux* (останнє звернення 01.06.2023).

37. The Game Awards 2017 – Вікіпедія. [Електронний ресурс]. – Режим доступу: *https://en.wikipedia.org/wiki/The_Game_Awards_2017* (останнє звернення 01.06.2023).

38. *Andrew Rollings, David Morris. Game Architecture and Design. – New Riders,* 2003. – 960*p*.

39. *R. Nystrom, Game Programming Patterns. – Genever Benning,* 2014. – 345*p*.

40. *J. Hocking, Jesse Schell, Unity in Action: Multiplatform Game Development in C# with Unity 5. – Manning Publications, 2022 –* 386*p*.

41. Басюркіна Н.Й., Свистун Т.В. Методичні вказівки до оцінки науково-технічної ефективності розробки нової технології, нового обладнання та інших інновацій. – Одеса: ОНАТУ, 2022 р. – С. 18.

42. Демчан І. С. Шкідливі фактори при роботі з комп'ютерами. [Електронний Ресурс]. – Режим доступу: https://www.slideshare.net/d_iruna/ (останнє звернення 01.06.2023).