

ПВНЗ «ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»

КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

СПЕЦІАЛЬНІСТЬ 121 ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА “ІНЖЕНЕРІЯ ПРОГРАМНОГО  
ЗАБЕЗПЕЧЕННЯ

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему: Розробка мобільної гри з використанням Unreal Engine.

Студент

Шкарпіта Іван Андрійович

Науковий керівник

Савченко Я.О.

Допущено до захисту “ ”

202 р.

Завідувач кафедри  
наук, доцент

Сушинський О. Є., доктор технічних

Декан факультету

Яровий Р. О.

Завідувач навчальної частини

Козинець А. О.

Київ – 2025

ПВНЗ «ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»

КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

СПЕЦІАЛЬНІСТЬ 121 ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА “ІНЖЕНЕРІЯ ПРОГРАМНОГО  
ЗАБЕЗПЕЧЕННЯ  
ЗАВДАННЯ

на випускню кваліфікаційну бакалаврську роботу студентів:

Шкарпіта Іван Андрійович

1. Тема роботи

Розробка мобільної гри з використанням Unreal Engine

2. Затверджена наказом по університету від “ ” 202 р. №

3. Цільова установка на вихідні дані до роботи

Основою для виконання дипломної роботи є актуальність тематики, обрана ціль та завдання дослідження, технічні можливості Unreal Engine 5, специфіка мобільних платформ, а також вимоги до розробки і оптимізації мобільної гри. Вихідними даними слугують науково-технічна література, офіційна документація Unreal Engine, практичні приклади реалізації ігрових механік, результати тестування прототипу на мобільних пристроях та особисті напрацювання автора.

4. Перелік графічного матеріалу (з точним визначенням обов'язкових креслень)

Скріншоти з редактора Unreal Engine.

5. Консультанти по роботі, із зазначенням розділів, що їх стосуються:

| Розділ   | Науковий керівник | Підпис, дата   |                  |
|----------|-------------------|----------------|------------------|
|          |                   | Завдання видав | Завдання отримав |
| Розділ 1 | Савченко Я. О.    | 04.11.24       | 04.11.24         |
| Розділ 2 | Савченко Я. О.    | 03.03.25       | 03.03.25         |
| Розділ 3 | Савченко Я. О.    | 24.03.25       | 24.03.25         |

6. Зміст випускної кваліфікаційної бакалаврської роботи (перелік питань)

Зміст

Вступ

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МОБІЛЬНИХ ІГОР НА БАЗІ UNREAL ENGINE 5

1.1. Тенденції розвитку ринку мобільних ігор

1.1.2. Обґрунтування вибору Unreal Engine 5

1.1.3. Порівняння з традиційними підходами до розробки

1.2. Сучасний стан галузі розробки мобільних ігор

1.2.1. Огляд ринку мобільних ігор та його обсяги

1.2.2. Основні тенденції та перспективи розвитку

1.2.3. Аналіз технічних обмежень та викликів мобільних платформ

- 1.2.4. Популярні жанри мобільних ігор та їх особливості
- 1.3. Методологія розробки мобільних ігор
  - 1.3.1. Особливості життєвого циклу розробки мобільної гри
  - 1.3.2. Проектування архітектури з урахуванням обмежень мобільних пристроїв
  - 1.3.3. Методи оптимізації продуктивності та енергоспоживання
  - 1.3.4. Підходи до тестування мобільних ігор
- 1.4. Порівняльний аналіз інструментів розробки мобільних ігор
  - 1.4.1. Порівняння Unreal Engine 5 з іншими рушіями для мобільної розробки (Unity, Godot)
  - 1.4.2. Переваги та недоліки використання UE5 для мобільних проєктів
  - 1.4.3. Аналіз доступних плагінів та готових рішень для мобільної розробки
  - 1.4.4. Кросплатформені можливості Unreal Engine 5

## РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ГРИ ТИПУ “ENDLESS RUNNER” НА UNREAL ENGINE 5

- 2.1. Жанр "Endless Runner": історія, особливості та ключові механіки
  - 2.1.1. Історія розвитку жанру "Endless Runner"
  - 2.1.2. Основні характеристики та особливості жанру
  - 2.1.3. Ключові механіки гри "Endless Runner"
- 2.2. Основні підходи до розробки ігор жанру "Endless Runner"
  - 2.2.1. Процедурна генерація рівнів: алгоритми та методи
  - 2.2.2. Розробка системи управління та інтерфейсу користувача (UI)
  - 2.2.3. Створення анімацій та візуальних ефектів
- 2.3. Unreal Engine 5 як інструмент для розробки ігор
  - 2.3.1. Огляд можливостей Unreal Engine 5

2.3.2. Переваги використання Unreal Engine 5 для розробки мобільних ігор

2.4. Психологічні та поведінкові аспекти дизайну ігор жанру "Endless Runner"

2.4.1. Дослідження мотивації та залучення гравців у мобільних іграх

2.4.2. Аналіз поведінки гравців у мобільних іграх жанру "Endless Runner"

2.4.3. Вплив візуального та аудіо дизайну на залучення гравців

2.5.1. Аналіз ринку мобільних ігор жанру "Endless Runner"

2.5.2. Соціальні аспекти мобільних ігор жанру "Endless Runner"

2.5.3. Монетизація мобільних ігор жанру "Endless Runner"

## РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ГРИ “ENDLESS RUNNER” НА UNREAL ENGINE 5

3.1. Планування та підготовка до розробки

3.1.1. Визначення основних механік та функціоналу гри

3.1.2 Налаштування проекту Unreal Engine 5

3.2. Розробка ключових механік гри

3.2.1. Створення Персонажа та реалізація механік руху

3.2.2. Генерація шляху та перешкод

3.2.3. Розробка системи збору ігрової валюти та підсилень

3.3. Розробка інтерфейсу користувача (UI)

3.3.1. Створення віджетів для відображення інформації

3.3.2 Створення віджетів для відображення поразки та меню паузи

3.3.3 Створення головного меню гри та меню вибору персонажа.

ВИСНОВОК

7. Календарний план виконання випускної кваліфікаційної  
бакалаврської роботи

| №<br>з/п | Назва етапів випускної<br>кваліфікаційної<br>бакалаврської<br>роботи        | Термін виконання етапів випускної<br>кваліфікаційної бакалаврської<br>роботи |                     |
|----------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------|---------------------|
|          |                                                                             | За планом                                                                    | Фактично            |
| 1.       | Обрання теми випускної<br>кваліфікаційної<br>магістерської<br>роботи (ВКБР) | 05.08.24 – 13.09.24                                                          | 05.08.24 – 13.09.24 |
| 2.       | Опрацювання літератури та<br>інших джерел для<br>написання<br>ВКБР          | 16.09.24 – 01.11.24                                                          | 16.09.24 – 01.11.24 |
| 3.       | Виконання 1 розділу ВКБР                                                    | 04.11.24 – 28.02.25                                                          | 04.11.24 – 28.02.25 |
| 4.       | Виконання 2 розділу ВКБР                                                    | 03.03.25 – 21.03.25                                                          | 03.03.25 – 21.03.25 |
| 5.       | Виконання 3 розділу ВКБР,<br>висновки                                       | 24.03.25 – 18.04.25                                                          | 24.03.25 – 18.04.25 |
| 6.       | Передзахист та проведення<br>перевірки на плагіат                           | 21.04.24 – 02.05.25                                                          | 21.04.24 – 02.05.25 |
| 7.       | Відгук керівника, рецензія                                                  | 01.05.25 – 12.05.25                                                          | 01.05.25 – 12.05.25 |

Керівник

Завдання прийняв до виконання

## Анотація

У бакалаврській кваліфікаційній роботі розглянуто процес розробки мобільної гри жанру *Endless Runner* з використанням рушія Unreal Engine 5. Робота складається з теоретичного аналізу сучасного стану індустрії мобільних ігор, особливостей розробки ігор для мобільних платформ, а також порівняння існуючих рушіїв для створення ігор. Особливу увагу приділено вивченню ключових механік жанру *Endless Runner*, процедурної генерації контенту, користувацького інтерфейсу та механік руху персонажа.

Практична частина роботи присвячена створенню прототипу мобільної гри: реалізовано основні ігрові механіки, інтерфейс користувача, систему збору бонусів та генерацію перешкод. Проведено тестування гри на мобільному пристрої для виявлення технічних обмежень та шляхів оптимізації.

Результатом дослідження є повнофункціональний прототип мобільної гри, що демонструє можливості Unreal Engine 5 у контексті розробки продуктивного, кросплатформеного ігрового продукту.

Ключові слова: Unreal Engine 5, мобільна гра, Endless Runner, процедурна генерація, UI, розробка ігор.

## **Abstract**

This bachelor's qualification work examines the process of developing a mobile game in the Endless Runner genre using the Unreal Engine 5. The work includes a theoretical analysis of the current state of the mobile game industry, the specifics of mobile game development, and a comparison of existing game engines. Special attention is given to the study of key mechanics of the Endless Runner genre, procedural content generation, user interface design, and character movement mechanics.

The practical part of the work focuses on the creation of a mobile game prototype: core game mechanics, user interface, bonus collection system, and obstacle generation have been implemented. The game was tested on a mobile device to identify technical limitations and explore optimization methods.

The result of the research is a fully functional prototype of a mobile game that demonstrates the capabilities of Unreal Engine 5 in the context of developing a performant, cross-platform game product.

**Keywords:** Unreal Engine 5, mobile game, Endless Runner, procedural generation, UI, game development.

## Зміст

|                                                                                               |    |
|-----------------------------------------------------------------------------------------------|----|
| Вступ.....                                                                                    | 10 |
| Розділ 1. Теоретичні основи розробки мобільних ігор на базі Unreal Engine 5                   |    |
| 1.1. Тенденції розвитку ринку мобільних ігор.....                                             | 12 |
| 1.1.2. Обґрунтування вибору Unreal Engine 5.....                                              | 13 |
| 1.1.3. Порівняння з традиційними підходами до розробки.....                                   | 14 |
| 1.2. Сучасний стан галузі розробки мобільних ігор                                             |    |
| 1.2.1. Огляд ринку мобільних ігор та його обсяги.....                                         | 14 |
| 1.2.2. Основні тенденції та перспективи розвитку.....                                         | 15 |
| 1.2.3. Аналіз технічних обмежень та викликів мобільних платформ.....                          | 18 |
| 1.2.4. Популярні жанри мобільних ігор та їх особливості.....                                  | 20 |
| 1.3. Методологія розробки мобільних ігор                                                      |    |
| 1.3.1. Особливості життєвого циклу розробки мобільної гри.....                                | 23 |
| 1.3.2. Проектування архітектури з урахуванням обмежень мобільних пристроїв.....               | 24 |
| 1.3.3. Методи оптимізації продуктивності та енергоспоживання.....                             | 24 |
| 1.3.4. Підходи до тестування мобільних ігор.....                                              | 25 |
| 1.4. Порівняльний аналіз інструментів розробки мобільних ігор                                 |    |
| 1.4.1. Порівняння Unreal Engine 5 з іншими рушіями для мобільної розробки (Unity, Godot)..... | 25 |
| 1.4.2. Переваги та недоліки використання UE5 для мобільних проектів.....                      | 26 |
| 1.4.3. Аналіз доступних плагінів та готових рішень для мобільної розробки.....                | 27 |
| 1.4.4. Кросплатформені можливості Unreal Engine 5.....                                        | 27 |
| Розділ 2. Теоретичні основи розробки гри типу “Endless Runner” на Unreal Engine 5             |    |
| 2.1. Жанр "Endless Runner": історія, особливості та ключові механіки.....                     | 28 |
| 2.1.1. Історія розвитку жанру "Endless Runner".....                                           | 28 |
| 2.1.2. Основні характеристики та особливості жанру.....                                       | 29 |
| 2.1.3. Ключові механіки гри "Endless Runner".....                                             | 30 |
| 2.2. Основні підходи до розробки ігор жанру "Endless Runner"                                  |    |
| 2.2.1. Процедурна генерація рівнів: алгоритми та методи.....                                  | 31 |
| 2.2.2. Розробка системи управління та інтерфейсу користувача (UI).....                        | 33 |
| 2.2.3. Створення анімацій та візуальних ефектів.....                                          | 36 |
| 2.3. Unreal Engine 5 як інструмент для розробки ігор                                          |    |
| 2.3.1. Огляд можливостей Unreal Engine 5.....                                                 | 38 |
| 2.3.2. Переваги використання Unreal Engine 5 для розробки мобільних ігор.....                 | 40 |
| 2.4. Психологічні та поведінкові аспекти дизайну ігор жанру "Endless Runner"                  |    |

|                                                                               |     |
|-------------------------------------------------------------------------------|-----|
| 2.4.1. Дослідження мотивації та залучення гравців у мобільних іграх.....      | 42  |
| 2.4.2. Аналіз поведінки гравців у мобільних іграх жанру "Endless Runner"..... | 45  |
| 2.4.3. Вплив візуального та аудіо дизайну на залучення гравців.....           | 47  |
| 2.5 Особливості мобільних ігор жанру "Endless Runner"                         |     |
| 2.5.1. Аналіз ринку мобільних ігор жанру "Endless Runner".....                | 49  |
| 2.5.2. Соціальні аспекти мобільних ігор жанру "Endless Runner".....           | 52  |
| 2.5.3. Монетизація мобільних ігор жанру "Endless Runner".....                 | 53  |
| <br>Розділ 3. Практична реалізація гри “Endless Runner” на Unreal Engine 5    |     |
| 3.1. Планування та підготовка до розробки                                     |     |
| 3.1.1. Визначення основних механік та функціоналу гри.....                    | 54  |
| 3.1.2 Налаштування проекту Unreal Engine 5.....                               | 56  |
| 3.2. Розробка ключових механік гри                                            |     |
| 3.2.1. Створення Персонажа та реалізація механік руху.....                    | 60  |
| 3.2.2. Генерація шляху та перешкод.....                                       | 73  |
| 3.2.3. Розробка системи збору ігрової валюти та підсилень.....                | 89  |
| 3.3. Розробка інтерфейсу користувача (UI)                                     |     |
| 3.3.1. Створення віджетів для відображення інформації.....                    | 103 |
| 3.3.2 Створення віджетів для відображення поразки та меню паузи.....          | 111 |
| 3.3.3 Створення головного меню гри та меню вибору персонажа.....              | 126 |
| <br>ВИСНОВОК.....                                                             | 141 |

## **ВСТУП**

### **Актуальність теми**

Сучасний ринок мобільних ігор є одним з найбільш швидкозростаючих сегментів індустрії розваг. За даними аналітичної компанії Newzoo, глобальний ринок мобільних ігор у 2023 році досяг обсягу понад 103,5 мільярда доларів, що становить більше половини загального ринку відеоігор. Прогнозується подальше зростання цього сегменту на 5-7% щорічно протягом наступних п'яти років. Така динаміка зумовлена постійним збільшенням кількості користувачів смартфонів, підвищенням доступності мобільного інтернету та технологічним прогресом у галузі мобільних пристроїв.

### **Мета і завдання дослідження**

#### **Мета дослідження**

Метою дипломної роботи є дослідження та практичне застосування можливостей Unreal Engine 5 для розробки мобільної гри типу endless runner, а також розробка методичних рекомендацій щодо оптимізації подібних проектів для мобільних платформ.

#### **Завдання дослідження**

Для досягнення поставленої мети необхідно виконати наступні завдання:

- Провести комплексний аналіз технічних можливостей Unreal Engine 5 для розробки мобільних ігор, визначити особливості та обмеження рушія при роботі з мобільними платформами.

- Дослідити сучасні методики оптимізації мобільних ігор, включаючи оптимізацію графічного рендерингу, управління пам'яттю та енергоспоживанням.
- Вивчити підходи до проектування архітектури мобільних ігор, з урахуванням специфіки жанру endless runner та особливостей цільових платформ.
- Розробити прототип мобільної гри жанру endless runner з використанням Unreal Engine 5, реалізуючи основні ігрові механіки та візуальні елементи.
- Провести тестування продуктивності розробленого прототипу на різних мобільних пристроях та виконати оптимізацію проекту для забезпечення стабільної частоти кадрів та енергоефективності.
- На основі отриманого досвіду розробити методичні рекомендації щодо створення та оптимізації мобільних ігор з використанням Unreal Engine 5.

## **Об'єкт і предмет дослідження**

### **Об'єкт дослідження**

Об'єктом дослідження є процес розробки мобільних ігор з використанням сучасних ігрових двигунів, зокрема технології та методології, що застосовуються при створенні мобільних ігрових проектів.

### **Предмет дослідження**

Предметом дослідження є методи та інструменти розробки мобільної гри жанру endless runner з використанням Unreal Engine 5, включаючи технічні аспекти реалізації ігрових механік, оптимізацію продуктивності та адаптацію проекту для різних моделей мобільних пристроїв.

## **РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МОБІЛЬНИХ ІГОР НА БАЗІ UNREAL ENGINE 5**

### **1.1. Тенденції розвитку ринку мобільних ігор**

За останні роки спостерігається суттєва зміна тенденцій на ринку мобільних ігор. Якщо раніше мобільні ігри асоціювалися переважно з простими казуальними проектами з примітивною графікою, то сьогодні користувачі очікують від мобільних ігор візуальної якості та глибини геймплею, наближеної до консольних та ПК-проектів. Сучасні флагманські смартфони оснащені потужними графічними процесорами, що підтримують такі передові технології, як трасування променів, фізично правильне освітлення та високоякісні текстури. Це створює підґрунтя для розробки технологічно складних та візуально вражаючих ігрових проектів на мобільних платформах.

#### **1.1.2. Обґрунтування вибору Unreal Engine 5**

У відповідь на ці тенденції розробники ігрових двигунів адаптують свої продукти для задоволення зростаючих вимог мобільної розробки. Unreal Engine 5, представлений компанією Epic Games у 2021 році, є одним із найпрогресивніших рішень у цій сфері. Даний рушій пропонує революційні технології, які дозволяють створювати візуально вражаючі проекти навіть для мобільних платформ, оптимізуючи їх для збереження продуктивності та енергоефективності.

Вибір Unreal Engine 5 як інструмента для розробки мобільних ігор обґрунтований кількома ключовими факторами:

1. UE5 пропонує інноваційні технології Nanite та Lumen, які радикально змінюють підхід до деталізації та освітлення сцени. Nanite – система віртуалізованої геометрії, що дозволяє

використовувати надзвичайно деталізовані моделі без значного впливу на продуктивність. Хоча повноцінна реалізація Nanite поки що доступна лише для платформ високого класу, адаптовані версії цієї технології можуть бути застосовані і для мобільних проєктів. Lumen забезпечує динамічне глобальне освітлення, що значно підвищує реалістичність візуального представлення гри.

2. UE5 пропонує розширені інструменти оптимізації, спеціально адаптовані для мобільних платформ: система рівнів деталізації (LOD), динамічне масштабування розділення текстур, оптимізація процесів рендерингу. Ці інструменти дозволяють балансувати між візуальною якістю та продуктивністю, що критично важливо для мобільних пристроїв з обмеженими обчислювальними ресурсами та необхідністю економії заряду батареї.
3. Unreal Engine 5 надає унікальну систему візуального програмування Blueprint, яка значно спрощує процес розробки та прототипування ігрових механік. Це особливо важливо в контексті мобільної розробки, де швидкість виведення продукту на ринок часто є ключовим фактором успіху.

### **1.1.3. Порівняння з традиційними підходами до розробки**

Порівнюючи Unreal Engine 5 з традиційними підходами до розробки мобільних ігор, варто відзначити кілька суттєвих переваг. На відміну від нативної розробки з використанням мов програмування Swift (iOS) або Kotlin/Java (Android), UE5 забезпечує кросплатформеність, дозволяючи створювати проєкт одночасно для обох основних мобільних операційних систем. Це значно скорочує часові та фінансові витрати на розробку.

У порівнянні з іншими популярними двигунами для мобільної розробки, такими як Unity, Unreal Engine 5 пропонує більш досконалі

технології візуалізації, що дозволяють досягти фотореалістичної графіки навіть на мобільних пристроях. Крім того, інтеграція з великою екосистемою Epic Games, включаючи маркетплейс з готовими ассетами та систему розподілу доходів, створює додаткові переваги для розробників.

Однак використання потужного рушія як Unreal Engine 5 для мобільної розробки пов'язане з певними викликами, зокрема з необхідністю глибокого розуміння методів оптимізації та адаптації високоякісних ассетів для пристроїв з обмеженими ресурсами. Це робить актуальним дослідження ефективних методик та підходів до розробки мобільних ігор з використанням UE5, які дозволяють максимально задіяти потенціал даного інструмента, одночасно забезпечуючи оптимальну продуктивність на цільових платформах.

Таким чином, дослідження можливостей та особливостей розробки мобільних ігор з використанням Unreal Engine 5 є актуальною темою, що має як теоретичну цінність у контексті вивчення сучасних технологій ігрової розробки, так і практичне значення для створення конкурентоспроможних мобільних ігрових продуктів.

## **1.2. Сучасний стан галузі розробки мобільних ігор**

### **1.2.1. Огляд ринку мобільних ігор та його обсяги**

Ринок мобільних ігор на сучасному етапі є домінуючим сегментом світової ігрової індустрії, демонструючи стабільне зростання та інноваційний розвиток. За даними аналітичної агенції Newzoo, у 2023 році глобальний дохід від мобільних ігор склав понад 103,5 мільярда доларів США, що становить приблизно 51% загального обсягу ігрового ринку.

Прогнозується, що до кінця 2025 року цей показник досягне 116 мільярдів доларів.

Регіональна структура ринку мобільних ігор демонструє значну диверсифікацію. Найбільшими регіональними ринками залишаються Азіатсько-Тихоокеанський регіон (особливо Китай, Японія та Південна Корея), який формує близько 50% глобального доходу, Північна Америка (приблизно 25%) та Європа (близько 18%). При цьому найбільш динамічне зростання демонструють ринки Південно-Східної Азії, Індії, Латинської Америки та Близького Сходу, де спостерігається швидке збільшення кількості користувачів смартфонів та покращення доступу до мобільного інтернету.

Показовим є той факт, що активна користувацька база мобільних ігор у 2023 році перевищила 2,8 мільярда гравців, що становить близько 36% світового населення. Такий масштаб аудиторії зумовлений демократизацією доступу до смартфонів, відносно низьким порогом входу для користувачів (більшість мобільних ігор розповсюджуються за безкоштовною моделлю free-to-play) та постійним розширенням різноманіття ігрових продуктів.

Варто відзначити, що бізнес-модель free-to-play з внутрішньоігровими мікротранзакціями залишається домінуючою на ринку мобільних ігор, генеруючи понад 90% загального доходу. При цьому на 20% найактивніших гравців (так званих "китів") припадає близько 65% усіх витрат на мікротранзакції. Водночас набирають популярності й альтернативні моделі монетизації, зокрема підписки (наприклад, Apple Arcade, Google Play Pass), гібридні моделі з рекламною інтеграцією та преміальні ігри з одноразовою оплатою.

Конкуренція на ринку мобільних ігор залишається надзвичайно високою. За даними App Annie, в магазинах App Store та Google Play щоденно публікується понад 500 нових ігрових додатків. При цьому спостерігається яскраво виражена тенденція до консолідації ринку, де найбільші видавці (такі як Tencent, NetEase, Activision Blizzard King, Supercell) контролюють значну частку доходів.

### 1.2.2. Основні тенденції та перспективи розвитку

Сучасний ринок мобільних ігор характеризується низкою виразних тенденцій, що визначатимуть його розвиток у найближчі роки:

- **Підвищення технологічної складності продуктів.** Спостерігається стійкий тренд до створення технологічно досконалих ігор з високоякісною графікою та складними ігровими механіками. Мобільні ігри все частіше наближаються за своїм технологічним рівнем до консольних та ПК-проектів. Цьому сприяє постійне вдосконалення мобільних пристроїв та поява спеціалізованих ігрових смартфонів з підвищеною продуктивністю.
- **Розвиток крос-платформеності.** Зростає популярність ігор, що підтримують синхронізацію прогресу та мультиплеер між мобільними пристроями, ПК та консолями. Такі проекти як Fortnite, Genshin Impact, та Call of Duty: Mobile демонструють успішність цього підходу, розширюючи аудиторію та збільшуючи залученість гравців.
- **Впровадження технологій доповненої (AR) та віртуальної (VR) реальності.** Хоча AR-ігри вже досягли певного успіху (зокрема, Pokémon GO), розвиток VR для мобільних платформ все ще знаходиться на ранній стадії. Проте з розвитком технологій та

зниженням вартості VR-гарнітур очікується зростання цього сегменту.

- **Інтеграція елементів соціальних мереж.** Сучасні мобільні ігри все частіше включають функції соціальної взаємодії, дозволяючи гравцям спілкуватися, формувати спільноти та змагатися один з одним. Ця тенденція сприяє формуванню стійких ігрових спільнот та підвищує рівень залученості користувачів.
- **Розвиток хмарних ігрових сервісів.** Технології на кшталт Google Stadia, Microsoft xCloud та NVIDIA GeForce Now починають адаптуватися для мобільних пристроїв, що потенційно дозволить запускати технологічно вимогливі ігри на смартфонах з обмеженими ресурсами.
- **Імплементація технологій штучного інтелекту.** ШІ використовується не лише для створення більш реалістичних неігрових персонажів, але й для персоналізації ігрового досвіду, адаптації складності та аналізу поведінки користувачів з метою оптимізації бізнес-показників.
- **Розвиток мобільного кіберспорту.** Такі ігри як PUBG Mobile, Mobile Legends: Bang Bang та Free Fire сформували масштабні кіберспортивні екосистеми з професійними лігами та міжнародними турнірами з великими призовими фондами.
- **Адаптація механік автобатлерів, королівських битв та інших сучасних жанрів.** Успішні жанри з ПК та консольного сегменту швидко адаптуються для мобільних платформ, часто з додаванням унікальних механік, оптимізованих для сенсорного управління.
- **Інтеграція блокчейн-технологій та NFT.** Незважаючи на певну контroversійність, ігри з інтеграцією криптовалют та незамінних токенів продовжують розвиватися, формуючи нішовий, але перспективний сегмент ринку.

Ці тенденції вказують на подальше ускладнення та диверсифікацію ринку мобільних ігор, що створює як нові можливості для розробників, так і підвищує рівень вхідного бар'єру для нових проектів.

### **1.2.3. Аналіз технічних обмежень та викликів мобільних платформ**

Незважаючи на постійне вдосконалення мобільних пристроїв, розробка ігор для цієї платформи все ще пов'язана з низкою технічних обмежень та викликів:

- **Фрагментація апаратного забезпечення.** На ринку представлені тисячі моделей мобільних пристроїв з різними характеристиками (розмір екрану, продуктивність процесора, об'єм оперативної пам'яті). Це ускладнює процес розробки та тестування, вимагаючи створення адаптивних рішень для забезпечення стабільної роботи на широкому спектрі пристроїв.
- **Обмеження обчислювальних ресурсів.** Навіть найпотужніші смартфони значно поступаються ПК та консолям за обчислювальною потужністю, що обмежує графічну складність та масштаб ігрових проектів. При цьому розробники мають балансувати між візуальною якістю та продуктивністю.
- **Енергоспоживання.** Мобільні ігри повинні бути оптимізовані для мінімізації споживання енергії, оскільки надмірне навантаження на батарею суттєво погіршує користувацький досвід. Це створює додаткові вимоги до оптимізації графічного рендерингу та фонових процесів.
- **Обмеження оперативної та постійної пам'яті.** Мобільні пристрої часто мають обмежений обсяг пам'яті, що вимагає ретельного управління ресурсами та оптимізації ассетів. Крім того, існують обмеження на розмір додатків у магазинах додатків, що змушує

розробників використовувати технології динамічного завантаження контенту.

- **Особливості сенсорного управління.** Відсутність фізичних контролерів створює певні обмеження для реалізації складних систем управління, особливо в іграх з динамічним геймплеем. Це вимагає створення інтуїтивних та ергономічних інтерфейсів, адаптованих для сенсорних екранів.
- **Проблеми з підключенням до мережі.** Більшість сучасних мобільних ігор використовують онлайн-компоненти, проте якість мобільного інтернету може бути нестабільною. Це вимагає реалізації механізмів толерантності до затримок та роботи в офлайн-режимі.
- **Теплові обмеження.** Інтенсивні обчислення призводять до нагрівання мобільних пристроїв, що може викликати термічне дроселювання (thermal throttling) — автоматичне зниження продуктивності для запобігання перегріву. Це змушує розробників впроваджувати динамічне масштабування навантаження в залежності від температурного режиму пристрою.
- **Різноманітність операційних систем та їх версій.** Необхідність підтримки різних версій Android та iOS ускладнює процес розробки та тестування, особливо з огляду на різну імплементацію стандартів у різних версіях ОС.
- **Обмеження магазинів додатків.** App Store та Google Play встановлюють суворі правила щодо контенту, монетизації та технічної реалізації додатків, що може обмежувати творчу свободу розробників та впливати на бізнес-моделі.

Для подолання цих обмежень розробники мобільних ігор використовують різноманітні підходи до оптимізації, включаючи:

- Імплементацію системи рівнів деталізації (LOD)

- Використання білінійної та анізотропної фільтрації текстур
- Оптимізацію шейдерів та матеріалів
- Використання різних рівнів якості в залежності від потужності пристрою
- Застосування технологій кешування та передрозрахунків
- Оптимізацію логіки оновлення ігрових об'єктів
- Впровадження асинхронного завантаження ассетів

#### 1.2.4. Популярні жанри мобільних ігор та їх особливості

Ринок мобільних ігор характеризується значним жанровим різноманіттям, проте можна виділити кілька категорій, що демонструють найбільшу популярність та комерційний успіх:

- **Гіперказуальні ігри (Hyper-casual)** — найпростіший та найдоступніший жанр з мінімалістичним дизайном, інтуїтивним керуванням та короткими ігровими сесіями. Такі ігри як Helix Jump, Stack Ball, Aquapark.io зазвичай монетизуються через рекламу та орієнтовані на максимально широку аудиторію. Особливості жанру:
  - Надзвичайно простий геймплей (часто достатньо лише одного дотику)
  - Мінімалістичний дизайн та швидкий цикл розробки
  - Висока відтворюваність (replayability)
  - Орієнтація на короткі ігрові сесії (до 2 хвилин)
- **Пазли та головоломки (Puzzle)** — інтелектуальні ігри, засновані на розв'язанні логічних задач. Представники: Candy Crush Saga, Gardenscapes, Homescapes. Характерні риси:
  - Покроковий геймплей, що не вимагає швидкої реакції
  - Поступове ускладнення рівнів
  - Наративна обгортка для підвищення залученості

- Механіки соціальної взаємодії
- **Казуальні ігри (Casual)** — доступні ігри з простим геймплеєм, орієнтовані на широку аудиторію. Приклади: Angry Birds, Cut the Rope, Fruit Ninja. Особливості:
  - Інтуїтивне управління
  - Розширена система прогресії порівняно з гіперказуальними іграми
  - Світлий, яскравий візуальний стиль
  - Орієнтація на сімейну аудиторію
- **Безкінечні бігуни (Endless runners)** — ігри, де персонаж постійно рухається вперед, а гравець керує ухиленням від перешкод та збором бонусів. Представники жанру: Subway Surfers, Temple Run, Sonic Dash. Характеристики:
  - Проста, але захоплююча механіка
  - Система поступової прогресії (збір валюти, покращення персонажів)
  - Орієнтація на встановлення рекордів
  - Сесійний геймплей з можливістю короткочасної гри
- **Стратегії та симулятори управління (Strategy and Management)** — ігри, засновані на стратегічному плануванні та управлінні ресурсами. Приклади: Clash of Clans, Rise of Kingdoms, Fallout Shelter. Особливості:
  - Глибокі механіки та довготривала прогресія
  - Соціальні елементи (клани, альянси)
  - Асинхронний PvP
  - Регулярні події та оновлення
- **Рольові ігри (RPG)** — проекти з фокусом на розвитку персонажа та наративі. Представники: Genshin Impact, Raid: Shadow Legends, AFK Arena. Характеристики:

- Комплексні системи прогресії персонажів
- Глибокий лор та сюжетні лінії
- Гача-механіки (випадкове отримання персонажів)
- Кооперативні та змагальні елементи
- **Мультиплеєрні онлайн-битви (MOBA та Battle Royale)** — змагальні ігри, орієнтовані на PvP-взаємодію. Приклади: PUBG Mobile, Mobile Legends: Bang Bang, Brawl Stars. Особливості:
  - Фокус на конкурентному геймплеї
  - Складні механіки, що потребують майстерності
  - Сезонна структура та рейтингові системи
  - Розвинений соціальний компонент
- **Спортивні та перегонові ігри (Sports & Racing)** — симулятори реальних видів спорту та автоперегонів. Представники: FIFA Mobile, Asphalt, 8 Ball Pool. Характеристики:
  - Реалістична фізика та управління
  - Ліцензований контент (реальні команди, спортсмени)
  - Щорічні оновлення для спортивних симуляторів
  - Поєднання одиночного та багатокористувацького режимів
- **Карткові ігри (Card Games)** — цифрові адаптації традиційних карткових ігор або оригінальні проекти. Приклади: Hearthstone, Magic: The Gathering Arena, Solitaire. Особливості:
  - Колекційні елементи
  - Тактична глибина
  - Регулярне оновлення мета-гри
  - Доступність для початківців з високим порогом майстерності

Важливо відзначити, що сучасні мобільні ігри часто поєднують елементи різних жанрів, створюючи гібридні продукти для розширення цільової аудиторії. Наприклад, багато RPG інкорпоруєть елементи

стратегій, а казуальні ігри запозичують механіки прогресії з більш хардкорних жанрів.

Жанр endless runner, який є фокусом даної роботи, залишається одним із найбільш успішних та стабільних на мобільному ринку. Його популярність пояснюється органічним поєднанням простого, інтуїтивно зрозумілого геймплею з довготривалою системою прогресії, що робить ігри цього жанру доступними для новачків, але цікавими для довготривалого залучення користувачів.

### **1.3. Методологія розробки мобільних ігор**

#### **1.3.1. Особливості життєвого циклу розробки мобільної гри**

Розробка мобільних ігор має свої унікальні особливості, які відрізняють її від розробки ігор для інших платформ.

- **Життєвий цикл розробки:** Життєвий цикл мобільної гри включає кілька етапів: концепція, прототипування, розробка, тестування, випуск та підтримка. Кожен етап вимагає уваги до специфічних особливостей мобільних платформ, таких як обмежені ресурси та різноманітність пристроїв. Розробники повинні враховувати ці фактори, щоб забезпечити стабільну роботу гри на всіх цільових пристроях.
- **Часті оновлення:** Мобільні ігри часто вимагають частіших оновлень для виправлення помилок та додавання нового контенту. Це допомагає підтримувати інтерес гравців та адаптувати гру до змін у ринкових умовах.

#### **1.3.2. Проектування архітектури з урахуванням обмежень мобільних пристроїв**

Проектування архітектури мобільної гри вимагає врахування обмежень мобільних пристроїв, таких як обмежена потужність процесора, об'єм оперативної пам'яті та енергоспоживання.

- **Модульна архітектура:** Використання модульної архітектури дозволяє розділити гру на окремі компоненти, які можна легко оновлювати та оптимізувати. Це допомагає забезпечити гнучкість та масштабованість проекту.
- **Оптимізація ресурсів:** Розробники повинні оптимізувати використання ресурсів, таких як текстури, моделі та звуки, для зменшення навантаження на пристрій. Це включає використання технік, таких як компресія текстур та зменшення кількості полігонів у моделях.

### 1.3.3. Методи оптимізації продуктивності та енергоспоживання

Оптимізація продуктивності та енергоспоживання є ключовими аспектами розробки мобільних ігор.

- **Оптимізація продуктивності:** Для забезпечення плавної роботи гри на мобільних пристроях необхідно використовувати такі методи, як зменшення кількості викликів до графічного процесора, оптимізація фізичних розрахунків та використання ефективних алгоритмів. Це допомагає зменшити затримки та підвищити продуктивність гри.
- **Економія енергоспоживання:** Оптимізація енергоспоживання включає зменшення використання процесора та графічного процесора в фоновому режимі, а також ефективне управління живленням. Це допомагає продовжити час роботи пристрою від одного заряду батареї.

### 1.3.4. Підходи до тестування мобільних ігор

Тестування мобільних ігор є важливим етапом, який допомагає виявити та виправити помилки перед випуском гри.

- **Функціональне тестування:** Перевірка всіх ігрових функцій на різних пристроях та операційних системах для забезпечення їх правильної роботи. Це включає тестування ігрових механік, інтерфейсу користувача та взаємодії з сервером.
- **Тестування продуктивності:** Вимірювання продуктивності гри на різних пристроях для виявлення проблем з продуктивністю, таких як затримки або зниження частоти кадрів. Це допомагає оптимізувати гру для стабільної роботи на всіх цільових пристроях.
- **Тестування сумісності:** Перевірка сумісності гри з різними версіями операційних систем та пристроями для забезпечення стабільної роботи на всіх популярних платформах.

Ці методологічні підходи допомагають розробникам створювати якісні мобільні ігри, які відповідають вимогам гравців та ринку.

## **1.4. Порівняльний аналіз інструментів розробки мобільних ігор**

### **1.4.1. Порівняння Unreal Engine 5 з іншими рушіями для мобільної розробки (Unity, Godot)**

Unreal Engine 5 (UE5) є одним з провідних ігрових рушіїв, але на ринку також присутні такі популярні рушії, як Unity та Godot. Кожен з них має свої особливості та переваги.

- **Unreal Engine 5:** UE5 вирізняється своїми передовими графічними можливостями, такими як Nanite та Lumen, які дозволяють створювати деталізовані та реалістичні сцени. Цей рушій ідеально підходить для розробки ігор з високими вимогами до графіки та

продуктивності. Однак, він може вимагати більше ресурсів для оптимізації на мобільних пристроях.

- **Unity:** Unity є універсальним рушієм, який пропонує широкий спектр інструментів для розробки ігор на різних платформах. Він має велику спільноту розробників та багато доступних плагінів. Unity добре підходить для розробки 2D-ігор та ігор з меншими вимогами до графіки.
- **Godot:** Godot — це легкий та відкритий ігровий рушій, який набирає популярності завдяки своїй простоті та гнучкості. Він добре підходить для інди-розробників та невеликих проектів. Godot пропонує хорошу підтримку 2D-ігор та має низький поріг входу для нових розробників.

#### 1.4.2. Переваги та недоліки використання UE5 для мобільних проектів

##### Переваги:

- **Графіка:** UE5 пропонує найкращі графічні можливості на ринку, що дозволяє створювати візуально вражаючі ігри.
- **Інструменти оптимізації:** UE5 має вбудовані інструменти для оптимізації продуктивності, такі як Level of Detail (LOD) та Nanite, які допомагають адаптувати гру для мобільних пристроїв.
- **Спільнота та підтримка:** Велика спільнота розробників та доступна документація роблять процес розробки простішим.

##### Недоліки:

- **Ресурси:** UE5 може вимагати більше часу та ресурсів для оптимізації на мобільних пристроях.
- **Складність:** Високий поріг входу для нових розробників, особливо для тих, хто не має досвіду в програмуванні на C++.

### 1.4.3. Аналіз доступних плагінів та готових рішень для мобільної розробки

UE5 пропонує широкий вибір плагінів та готових рішень, які можуть спростити процес розробки мобільних ігор.

- **Плагіни:** Доступні плагіни для інтеграції реклами, аналітики, соціальних функцій та інших можливостей. Це дозволяє розробникам швидко додавати нові функції до своїх ігор.
- **Готові рішення:** UE5 пропонує готові рішення для таких завдань, як робота з мережею, багатокористувацькі функції та інтеграція з популярними сервісами.

### 1.4.4. Кросплатформені можливості Unreal Engine 5

UE5 підтримує розробку ігор для різних платформ, включаючи мобільні пристрої, ПК та консолі.

- **Кросплатформеність:** Розробники можуть створювати гру один раз і випускати її на кількох платформах, що економить час та ресурси.
- **Сумісність:** UE5 забезпечує сумісність з популярними операційними системами, такими як iOS та Android, що дозволяє охопити широку аудиторію гравців.

Цей аналіз допоможе вам зрозуміти переваги та недоліки різних ігрових рушіїв та вибрати найкращий інструмент для вашого проекту.

Unreal Engine 5 (UE5) пропонує потужні інструменти та технології для розробки мобільних ігор, які дозволяють створювати високоякісні та імерсивні ігрові досвіди. Основні теоретичні основи включають використання передових графічних технологій, таких як Nanite та Lumen,

які забезпечують деталізовану графіку та динамічне освітлення. Крім того, UE5 пропонує інструменти для оптимізації продуктивності, такі як Level of Detail (LOD) та системи частинок Niagara, які допомагають адаптувати гру для обмежених ресурсів мобільних пристроїв. В наступному розділі можливості Unreal Engine будуть розглянуті детальніше.

## **РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ГРИ ТИПУ “ENDLESS RUNNER” НА UNREAL ENGINE 5**

### **2.1. Жанр "Endless Runner": історія, особливості та ключові механіки**

Жанр "endless runner" (безкінечний бігун) є одним з найпопулярніших у мобільній ігровій індустрії. Його особливість полягає в безперервному русі персонажа, керування яким зведено до простих жестів. Цей жанр поєднує в собі елементи аркади, платформера та гонки, створюючи динамічний та захоплюючий ігровий процес.

#### **2.1.1. Історія розвитку жанру "Endless Runner"**

Історія жанру "endless runner" бере свій початок з ранніх аркадних ігор. Одним з перших прикладів можна вважати гру "Pac-Man" (1980), де персонаж безперервно рухався лабіринтом, уникаючи ворогів. Проте, повноцінно жанр сформувався значно пізніше.

- **Ранні аркади (1980-ті):**

- Ігри, такі як "Pac-Man" (1980), заклали основи безперервного руху та уникнення перешкод. Хоча "Pac-Man" не є класичним

"endless runner", він представив концепцію безперервного руху лабіринтом, що стало важливим елементом жанру.

- Інші аркадні ігри, такі як "Frogger" (1981) та "Paperboy" (1985), також мали елементи безперервного руху та уникнення перешкод.
- Ці ігри були популярні в аркадних автоматах і заклали фундамент для майбутнього розвитку жанру.

- **Консольні ігри (1990-ті - 2000-ні):**

- З розвитком консольних ігор з'явилися платформери з елементами безперервного руху, такі як серія "Sonic the Hedgehog" (1991).
- Ці ігри мали швидкий темп та елементи уникнення перешкод, але вони ще не мали випадкової генерації рівнів, яка стала ключовою особливістю сучасних "endless runner".

- **Мобільні ігри:**

- Справжній розквіт жанру відбувся з появою мобільних ігор. Ігри, такі як "Canabalt" (2009), "Temple Run" (2011) та "Subway Surfers" (2012), визначили основні механіки та особливості жанру.
- Сучасні мобільні endless runner ігри, характеризуються яскравою графікою, динамічним геймплеем та соціальною складовою.

### **2.1.2. Основні характеристики та особливості жанру**

- **Безперервний рух:**

- Персонаж автоматично рухається вперед, а гравець контролює лише його дії.

- **Випадкова генерація рівнів:**

- Рівні генеруються процедурно, що забезпечує безкінечний та різноманітний ігровий процес.
- **Прості елементи управління:**
  - Керування зведено до простих жестів, таких як свайпи та торкання екрана.
- **Висока динаміка:**
  - Ігровий процес характеризується високою динамікою та швидким темпом.
- **Змагальний елемент:**
  - Багато ігор жанру мають таблиці лідерів та соціальні функції, що стимулюють змагання між гравцями.

### 2.1.3. Ключові механіки гри "Endless Runner"

- **Стрибки:**
  - Механіка стрибків дозволяє персонажу долати перешкоди та збирати предмети, що знаходяться у повітрі.
- **Ковзання:**
  - Ковзання дозволяє персонажу прослизати під низькими перешкодами.
- **Збір предметів:**
  - Збір монет, бонусів та інших предметів є важливою частиною ігрового процесу.
- **Уникнення перешкод:**
  - Головна мета гри - уникнути зіткнення з перешкодами, які з'являються на шляху персонажа.
- **Покращення:**
  - Часто, в іграх даного жанру, є можливість покращувати персонажа, або купувати нові скіни, або бонуси, які допомагають у проходженні гри.

## **2.2. Основні підходи до розробки ігор жанру "Endless Runner"**

### **2.2.1. Процедурна генерація рівнів: алгоритми та методи**

Процедурна генерація рівнів (ПГР) є ключовим елементом ігор жанру "endless runner". Вона дозволяє створювати безкінечні та різноманітні рівні, що забезпечує високу реіграбельність та динамічний ігровий процес. ПГР використовує алгоритми для автоматичного створення контенту, що значно спрощує процес розробки та дозволяє створювати унікальні ігрові світи.

#### **Опис різних алгоритмів та їх застосування в генерації рівнів:**

- **Алгоритм випадкового блукання (Random Walk Algorithm):**
  - Цей алгоритм використовує випадкові кроки для створення шляху, яким рухається гравець.
  - Він може бути використаний для генерації простих рівнів з випадковими перешкодами та предметами.
  - Перевага цього алгоритму полягає в його простоті та швидкості виконання.
  - Однак, він може створювати непередбачувані та незбалансовані рівні.
- **Алгоритм клітинного автомата (Cellular Automata Algorithm):**
  - Цей алгоритм використовує сітку клітин, кожна з яких має свій стан.
  - Стан клітин змінюється в залежності від стану сусідніх клітин за певними правилами.
  - Алгоритм клітинного автомата може бути використаний для створення складних та органічних рівнів.
  - Він дозволяє створювати різноманітні структури, такі як печери, тунелі та лабіринти.

- **Алгоритм на основі блоків (Tile-Based Algorithm):**

- Цей алгоритм використовує набір попередньо створених блоків рівнів.
- Блоки випадково з'єднуються між собою, створюючи безкінечний рівень.
- Цей алгоритм дозволяє створювати різноманітні та збалансовані рівні.
- Він також дозволяє легко контролювати складність та дизайн рівнів.

### **Переваги та недоліки процедурної генерації:**

- **Переваги:**

- Безкінечна реіграбельність: ПГР дозволяє створювати безкінечні та різноманітні рівні, що забезпечує високу реіграбельність.
- Економія часу та ресурсів: ПГР дозволяє автоматично створювати контент, що значно спрощує процес розробки.
- Унікальність: ПГР дозволяє створювати унікальні ігрові світи, які неможливо створити вручну.

- **Недоліки:**

- Непередбачуваність: ПГР може створювати непередбачувані та незбалансовані рівні.
- Складність контролю: ПГР може бути складно контролювати, що може призвести до непередбачуваних результатів.
- Необхідність у оптимізації: алгоритми процедурної генерації, можуть сильно навантажувати систему, тому потрібна оптимізація.

### **Створення блоків рівнів та їх випадкове з'єднання:**

Одним з найпоширеніших методів ПГР в іграх жанру "endless runner" є використання блоків рівнів. Блоки рівнів - це попередньо створені сегменти рівня, які можуть містити перешкоди, предмети та інші елементи гри. Алгоритм ПГР випадково вибирає та з'єднує блоки рівнів, створюючи безкінечний рівень.

Цей метод дозволяє:

- Контролювати складність та дизайн рівнів.
- Створювати різноманітні та збалансовані рівні.
- Легко додавати новий контент.

Використання процедурної генерації рівнів є ключовим аспектом розробки ігор такого жанру.

### **2.2.2. Розробка системи управління та інтерфейсу користувача (UI)**

Розробка ефективної системи управління та інтуїтивно зрозумілого інтерфейсу користувача є ключовим фактором успіху будь-якої мобільної гри, особливо в жанрі "endless runner", де швидкість реакції та простота управління відіграють вирішальну роль.

#### **Особливості сенсорного управління в мобільних іграх:**

- **Простота та інтуїтивність:**
  - Сенсорне управління має бути простим та інтуїтивно зрозумілим, щоб гравці могли швидко освоїти його.
  - В іграх жанру "endless runner" зазвичай використовуються прості жести, такі як свайпи (проведення пальцем по екрану) для стрибків та ковзань.
- **Чутливість та точність:**

- Сенсорне управління має бути чутливим та точним, щоб гравці могли швидко та точно реагувати на події в грі.
- Важливо забезпечити мінімальну затримку між жестом гравця та реакцією персонажа в грі.
- **Адаптація до різних розмірів екранів:**
  - Мобільні пристрої мають різні розміри екранів, тому система управління має бути адаптована до різних розмірів екранів.
  - Важливо забезпечити зручне розташування елементів управління на різних екранах.
- **Зручність:**
  - Управління повинно бути зручним для тривалої гри.

### **Принципи створення інтуїтивно зрозумілого інтерфейсу:**

- **Мінімалізм:**
  - Інтерфейс має бути мінімалістичним та не перевантаженим зайвими елементами.
  - Важливо відображати лише необхідну інформацію, таку як рахунок, відстань та бонуси.
- **Чіткість та ясність:**
  - Елементи інтерфейсу мають бути чіткими та ясними, щоб гравці могли легко їх розпізнати.
  - Важливо використовувати зрозумілі іконки та текст.
- **Консистентність:**
  - Інтерфейс має бути консистентним, тобто всі елементи мають бути виконані в одному стилі.
  - Це допоможе гравцям швидко освоїти інтерфейс та зрозуміти його логіку.
- **Відгук:**

- Інтерфейс повинен мати відгук на дії користувача. Наприклад, зміна кольору кнопки при натисканні.
- **Адаптивність:**
  - Інтерфейс повинен адаптуватися до різних розмірів екранів та орієнтацій пристроїв.

## **Використання віджетів для відображення інформації та елементів управління:**

- **Відображення рахунку та відстані:**
  - Віджети можуть використовуватися для відображення рахунку, відстані та інших важливих показників гри.
  - Важливо відображати цю інформацію у зручному та доступному місці на екрані.
- **Відображення бонусів та предметів:**
  - Віджети можуть використовуватися для відображення бонусів та предметів, які гравець зібрав під час гри.
  - Це допоможе гравцям відстежувати свій прогрес та використовувати бонуси.
- **Елементи управління:**
  - Віджети можуть використовуватися для створення елементів управління, таких як кнопки, джойстики та свайп-зони.
  - Важливо розташувати елементи управління у зручному місці на екрані, щоб гравці могли легко їх використовувати.
- **Меню та налаштування:**
  - Віджети використовуються для розробки меню, налаштувань, таблиць лідерів, та іншого.

### **2.2.3. Створення анімацій та візуальних ефектів**

Анімації та візуальні ефекти відіграють важливу роль у створенні динамічного та захоплюючого ігрового процесу в іграх жанру "endless runner". Вони допомагають підкреслити ключові моменти гри, зробити її більш візуально привабливою та залучити гравців.

#### **Роль анімацій у створенні динамічного ігрового процесу:**

- **Підкреслення дій персонажа:**
  - Анімації використовуються для відображення дій персонажа, таких як біг, стрибки, ковзання та збір предметів.
  - Плавні та реалістичні анімації роблять ігровий процес більш динамічним та захоплюючим.
- **Створення відчуття швидкості:**
  - Анімації використовуються для створення відчуття швидкості, що є важливим елементом ігор жанру "endless runner".
  - Розмиття руху, ефекти прискорення та інші візуальні ефекти допомагають передати відчуття швидкості.
- **Покращення візуального стилю:**
  - Анімації використовуються для створення унікального візуального стилю гри.
  - Стилізовані анімації, ефекти та переходи можуть зробити гру більш візуально привабливою та запам'ятовуваною.

#### **Використання візуальних ефектів для підкреслення ключових моментів гри:**

- **Підкреслення збору предметів:**
  - Візуальні ефекти використовуються для підкреслення збору предметів, таких як монети, бонуси та інші нагороди.

- Яскраві ефекти, такі як спалахи, блискітки та вибухи, роблять збір предметів більш задовольняючим.
- **Підкреслення уникнення перешкод:**
  - Візуальні ефекти використовуються для підкреслення уникнення перешкод, таких як стрибки та ковзання.
  - Ефекти сповільнення часу, розмиття руху та інші візуальні ефекти роблять уникнення перешкод більш драматичним.
- **Підкреслення досягнень:**
  - Візуальні ефекти використовуються для підкреслення досягнень гравця, таких як встановлення рекордів та отримання бонусів.
  - Ефекти феєрверків, спалахів та інших святкових ефектів роблять досягнення більш пам'ятними.

### **Принципи оптимізації анімацій та візуальних ефектів для мобільних пристроїв:**

- **Використання простих анімацій:**
  - Використовуйте прості анімації з мінімальною кількістю кадрів.
  - Це допоможе зменшити навантаження на процесор та графічний процесор.
- **Використання оптимізованих текстур:**
  - Використовуйте оптимізовані текстури з низькою роздільною здатністю.
  - Це допоможе зменшити розмір гри та навантаження на пам'ять.
- **Використання ефектів з низькою полігональністю:**
  - Використовуйте ефекти з низькою полігональністю, такі як частинки та сплайни.

- Це допоможе зменшити навантаження на графічний процесор.
- **Використання спрайтів замість 3D-моделей:**
  - Використовуйте спрайти замість 3D-моделей для простих анімацій та ефектів.
  - Це допоможе зменшити навантаження на процесор та графічний процесор.
- **Тестування на різних пристроях:**
  - Тестуйте анімації та візуальні ефекти на різних мобільних пристроях, щоб переконатися, що вони працюють плавно та без проблем.
  - Це допоможе виявити та виправити проблеми з продуктивністю.

## 2.3. Unreal Engine 5 як інструмент для розробки ігор

### 2.3.1. Огляд можливостей Unreal Engine 5

Unreal Engine 5 (UE5) – це передовий ігровий рушій, розроблений компанією Epic Games, який надає розробникам безпрецедентні можливості для створення високоякісних ігор та інтерактивних 3D-додатків. UE5 вирізняється своєю потужністю, гнучкістю та інноваційними технологіями, які дозволяють створювати надзвичайно реалістичні та захоплюючі ігрові світи.

- **Візуальне програмування Blueprints:**
  - Система Blueprints – це візуальний інструмент програмування, який дозволяє розробникам створювати ігрову логіку без написання коду на C++.
  - Замість текстових інструкцій використовуються візуальні блоки (ноди), які з'єднуються між собою, утворюючи графічну схему.

- Це значно спрощує процес розробки та робить його доступним для художників, дизайнерів та інших творчих фахівців, які не мають глибоких знань у програмуванні.
- Blueprints дозволяє швидко прототипувати ідеї, експериментувати з різними механіками та створювати складні інтерактивні системи.
- **Система Nanite:**
  - Nanite – це технологія віртуалізованої геометрії, яка дозволяє відображати мільярди полігонів у реальному часі.
  - Це усуває обмеження, пов'язані з деталізацією геометрії, і дозволяє використовувати моделі з кінематографічною якістю без втрати продуктивності.
  - Nanite автоматично обробляє рівні деталізації (LOD), що значно спрощує процес створення складних сцен.
  - Ця технологія відкриває нові можливості для створення надзвичайно реалістичних ігрових світів з безпрецедентним рівнем деталізації.
- **Система Lumen:**
  - Lumen – це система глобального освітлення в реальному часі, яка забезпечує надзвичайно реалістичне освітлення та відображення.
  - Вона дозволяє створювати динамічні сцени, де освітлення змінюється в залежності від часу доби, погоди та інших факторів.
  - Lumen усуває необхідність запікати освітлення, що значно прискорює процес розробки та дозволяє створювати більш інтерактивні сцени.

- Ця технологія забезпечує високу якість освітлення навіть на мобільних пристроях, що робить UE5 ідеальним вибором для розробки мобільних ігор.

- **MetaHumans:**

- MetaHumans – це інструмент для створення високоякісних цифрових персонажів.
- Він дозволяє швидко та легко створювати реалістичні моделі людей з різними зовнішніми характеристиками.
- MetaHumans надає широкий спектр інструментів для налаштування зовнішності персонажів, включаючи риси обличчя, волосся, одяг та аксесуари.
- Цей інструмент значно спрощує процес створення персонажів і дозволяє розробникам зосередитися на інших аспектах гри.

Unreal Engine 5 є потужним інструментом для розробки ігор, який надає розробникам широкий спектр можливостей для створення високоякісних та захоплюючих ігор.

### **2.3.2. Переваги використання Unreal Engine 5 для розробки мобільних ігор**

Unreal Engine 5 (UE5) пропонує ряд значних переваг для розробників мобільних ігор, які прагнуть створювати високоякісні та продуктивні ігри для платформ iOS та Android.

- **Висока якість графіки:**

- UE5 надає потужні інструменти для створення візуально приголомшливих мобільних ігор.

- Система Lumen забезпечує реалістичне глобальне освітлення в реальному часі, що дозволяє створювати динамічні та атмосферні ігрові середовища.
- Технологія Nanite дозволяє використовувати моделі з кінематографічною якістю без втрати продуктивності, що відкриває нові можливості для створення деталізованих ігрових світів.
- **Оптимізація продуктивності:**
  - UE5 має інструменти для оптимізації продуктивності, що дозволяє створювати плавні та швидкі мобільні ігри.
  - Рушій автоматично масштабує графіку та ресурси в залежності від можливостей пристрою.
  - UE5 підтримує різні методи оптимізації, такі як LOD, occlusion culling та інші, що дозволяє досягти високої продуктивності навіть на менш потужних мобільних пристроях.
- **Кросплатформність:**
  - UE5 дозволяє розробляти ігри для різних мобільних платформ, включаючи iOS та Android, з однієї кодової бази.
  - Це значно спрощує процес розробки та дозволяє охопити широку аудиторію.
- **Візуальне програмування Blueprints:**
  - Система Blueprints дозволяє розробникам створювати ігрову логіку без написання коду на C++.
  - Це значно спрощує процес розробки та робить його доступним для широкого кола розробників, включаючи художників та дизайнерів.
  - Blueprints дозволяє швидко прототипувати ідеї, експериментувати з різними механіками та створювати складні інтерактивні системи.

- **Велика спільнота та підтримка:**

- UE5 має велику спільноту розробників та широку підтримку від Epic Games, що забезпечує доступ до безлічі ресурсів та допомоги.
- Це дозволяє розробникам швидко знаходити рішення на проблеми та отримувати допомогу від досвідчених фахівців.

- **Інструмент MetaHumans:**

- MetaHumans це інструмент для створення високоякісних цифрових персонажів.
- Це дає змогу швидко та легко створювати реалістичні моделі людей з різними зовнішніми характеристиками.

Unreal Engine 5 є потужним та універсальним інструментом для розробки мобільних ігор, який надає розробникам широкий спектр можливостей для створення високоякісних та успішних ігор.

## **2.4. Психологічні та поведінкові аспекти дизайну ігор жанру "Endless Runner"**

Розробка успішних мобільних ігор вимагає не лише технічних знань, але й розуміння психологічних та поведінкових аспектів, які впливають на залучення та утримання гравців.

### **2.4.1. Дослідження мотивації та залучення гравців у мобільних іграх**

Мобільні ігри жанру "endless runner" мають величезну популярність, і розуміння того, що мотивує гравців та утримує їх, є ключовим для успішної розробки.

- **Аналіз психологічних факторів, що впливають на залучення гравців:**

- **Теорія самодетермінації:**

- Ця теорія стверджує, що люди мають три основні психологічні потреби: автономію, компетентність та спорідненість.
- **Автономія:** Гравці хочуть відчувати контроль над своїми діями та мати можливість вибору. У контексті "endless runner" це може бути досягнуто шляхом надання гравцям можливості вибирати персонажів, налаштовувати їх зовнішній вигляд або вибирати різні шляхи проходження гри.
- **Компетентність:** Гравці хочуть відчувати, що вони покращують свої навички та досягають прогресу. В "endless runner" це може бути досягнуто шляхом поступового підвищення складності гри, надання гравцям можливості покращувати свої результати та демонструвати свої навички.
- **Спорідненість:** Гравці хочуть відчувати зв'язок з іншими гравцями та бути частиною спільноти. В "endless runner" це може бути досягнуто шляхом інтеграції соціальних функцій, таких як таблиці лідерів, змагання з друзями та можливість ділитися своїми досягненнями.

- **Теорія потоку:**

- Ця теорія описує стан, коли людина повністю занурена в діяльність, відчуючи задоволення та зосередженість.
- Для досягнення стану потоку гра повинна мати оптимальний баланс між викликом та навичками гравця.

- В "endless runner" це може бути досягнуто шляхом поступового підвищення складності гри, що відповідає зростанню навичок гравця, а також шляхом створення динамічного та захоплюючого ігрового процесу.
- **Використання елементів гейміфікації для підвищення мотивації:**
  - **Системи нагород:**
    - Винагороди за досягнення, такі як монети, бонуси та нові предмети, стимулюють гравців до подальшої гри.
    - Важливо, щоб нагороди були цінними та відповідали зусиллям гравців.
  - **Таблиці лідерів:**
    - Змагання з іншими гравцями за високі результати мотивує гравців покращувати свої навички.
    - Важливо, щоб таблиці лідерів були чесними та відображали реальні досягнення гравців.
  - **Соціальні функції:**
    - Можливість ділитися досягненнями, спілкуватися з іншими гравцями та грати разом підвищує соціальну взаємодію та залучення.
    - Важливо, щоб соціальні функції були інтегровані в гру органічно та не порушували ігровий процес.
- **Дослідження впливу різних механік гри на залучення гравців:**
  - **Випадкова генерація рівнів:**
    - Непередбачуваність рівнів підтримує інтерес гравців та запобігає нудзі.
    - Важливо, щоб випадкова генерація рівнів була збалансованою та не створювала неможливих ситуацій.
  - **Прогресія складності:**

- Поступове підвищення складності гри забезпечує постійний виклик та стимулює гравців до розвитку.
- Важливо, щоб прогресія складності була пальною та відповідала зростанню навичок гравців.

#### **2.4.2. Аналіз поведінки гравців у мобільних іграх жанру "Endless Runner"**

Розуміння того, як гравці взаємодіють з вашою грою, є важливим для її успіху. Аналіз поведінки гравців дозволяє виявити типові моделі поведінки, помилки та стратегії, що, своєю чергою, допомагає покращити дизайн гри та баланс складності.

- **Дослідження моделей поведінки гравців:**

- **Аналіз даних телеметрії:**

- Телеметрія - це збір даних про дії гравців під час гри. Ці дані можуть включати час гри, результати, використання предметів, частоту помилок та багато іншого.
- Аналіз даних телеметрії дозволяє виявити типові моделі поведінки гравців, такі як:
  - Які перешкоди є найскладнішими для гравців.
  - Які предмети є найпопулярнішими.
  - Як гравці використовують бонуси.
  - Як гравці реагують на різні рівні складності.
- Ці дані можуть бути використані для покращення дизайну гри та балансу складності.

- **Аналіз записів ігрового процесу:**

- Записи ігрового процесу - це відеозаписи гри, які показують, як гравці взаємодіють з грою.

- Аналіз записів ігрового процесу дозволяє спостерігати за діями гравців та виявляти їхні стратегії та помилки.
- Цей метод дозволяє отримати більш якісні дані, ніж аналіз даних телеметрії, але він є більш трудомістким.
- **Виявлення типових помилок та стратегій гравців:**
  - Аналіз даних телеметрії та записів ігрового процесу дозволяє виявити типові помилки, які роблять гравці, такі як:
    - Неправильне використання бонусів.
    - Неправильне уникнення перешкод.
    - Неправильне використання елементів керування.
  - Аналіз даних також дозволяє виявити типові стратегії, які використовують гравці для досягнення високих результатів, такі як:
    - Використання певних комбінацій бонусів.
    - Використання певних маршрутів.
    - Використання певних елементів керування.
- **Використання аналізу поведінки гравців для покращення дизайну гри та балансу складності:**
  - На основі аналізу поведінки гравців можна:
    - Змінити дизайн рівнів, щоб зробити їх більш цікавими та складними.
    - Налаштувати баланс складності, щоб зробити гру більш доступною для новачків та більш складною для досвідчених гравців.
    - Змінити частоту появи предметів, щоб зробити гру більш збалансованою.
    - Змінити елементи керування, щоб зробити їх більш зручними та інтуїтивно зрозумілими.

- Використовувати ці дані для розробки нових механік гри.

### **2.4.3. Вплив візуального та аудіо дизайну на залучення гравців**

Візуальний та аудіо дизайн відіграють важливу роль у створенні захоплюючого та привабливого ігрового досвіду. Вони впливають на емоційний стан гравців, їхнє занурення в ігровий процес та загальне задоволення від гри.

- **Дослідження впливу кольорової палітри, стилю графіки та анімацій на емоційний стан гравців:**

- **Кольорова палітра:**

- Кольори можуть викликати різні емоції та асоціації. Наприклад, яскраві та насичені кольори можуть викликати відчуття енергії та радості, тоді як темні та приглушені кольори можуть викликати відчуття напруги та тривоги.
- В іграх жанру "endless runner" часто використовуються яскраві та насичені кольори, щоб підтримувати високий рівень енергії та динаміки.

- **Стиль графіки:**

- Стиль графіки визначає загальний візуальний вигляд гри. Він може бути реалістичним, мультяшним, стилізованим або абстрактним.
- Вибір стилю графіки залежить від цільової аудиторії та загального настрою гри.

- **Анімації:**

- Анімації використовуються для відображення дій персонажів, рухів об'єктів та візуальних ефектів.

- Плавні та динамічні анімації роблять ігровий процес більш захоплюючим та реалістичним.
- Важливим є, що анімації повинні бути оптимізовані для мобільних пристроїв.
- **Аналіз впливу звукового дизайну (музика, звукові ефекти) на занурення в ігровий процес:**
  - **Музика:**
    - Музика створює атмосферу гри та підсилює емоційний вплив.
    - Динамічна та ритмічна музика підтримує високий рівень енергії та динаміки гри.
    - Вдало підібрана музика, може значно покращити залучення гравця.
  - **Звукові ефекти:**
    - Звукові ефекти використовуються для відображення дій персонажів, зіткнень, вибухів та інших подій у грі.
    - Реалістичні та якісні звукові ефекти підсилюють відчуття присутності в грі.
    - Звукові ефекти, також, можуть давати корисну інформацію гравцю, попередження про перешкоди, чи сповіщення про бонуси.
- **Використання візуальних та аудіо ефектів для підкреслення ключових моментів гри та створення відчуття динаміки:**
  - **Візуальні ефекти:**
    - Візуальні ефекти, такі як розмиття руху, ефекти прискорення та спалахи, підсилюють відчуття швидкості та динаміки гри.

- Візуальні ефекти використовуються для підкреслення важливих моментів гри, таких як збір предметів, уникнення перешкод та досягнення високих результатів.
- **Аудіо ефекти:**
  - Звукові ефекти, такі як гучні звуки зіткнень, прискорення та вибухи, підкреслюють важливі моменти гри та створюють відчуття динаміки.
  - Звукові ефекти, допомагають створити відчуття нагороди, при зборі предметів, чи бонусів.

## **2.5 Особливості мобільних ігор жанру "Endless Runner"**

### **2.5.1. Аналіз ринку мобільних ігор жанру "Endless Runner"**

Ринок мобільних ігор є динамічним та швидкозростаючим, а жанр "endless runner" залишається одним з найпопулярніших. Розуміння тенденцій ринку, аналіз популярних ігор та виявлення ключових факторів успіху є важливими для розробників.

- **Дослідження тенденцій розвитку ринку мобільних ігор:**
  - **Зростання мобільного геймінгу:** Мобільні ігри продовжують набирати популярність, завдяки доступності смартфонів та планшетів.
  - **"Free-to-play" модель:** Більшість мобільних ігор використовують модель "free-to-play", де гра є безкоштовною, але гравці можуть купувати внутрішньоігрові предмети або бонуси.
  - **Соціальна інтеграція:** Соціальні функції, такі як таблиці лідерів, змагання з друзями та можливість ділитися досягненнями, стають все більш важливими для залучення та утримання гравців.

- **Гіперказуальні ігри:** Гіперказуальні ігри, які характеризуються простим геймплеєм та миттєвою доступністю, стають все більш популярними. Жанр "endless runner" часто використовує гіперказуальний стиль.
- **eSports та стрімінг:** Мобільні ігри починають завойовувати популярність в eSports та на стрімінгових платформах, таких як Twitch та YouTube.
- **Аналіз популярних ігор жанру "Endless Runner" та їхніх бізнес-моделей:**
  - **Subway Surfers:** Ця гра є однією з найпопулярніших у жанрі, завдяки динамічному геймплею та яскравій графіці. Її бізнес-модель базується на внутрішньоігрових покупках для персоналізації персонажів та отримання бонусів.
  - **Temple Run:** Ця гра також є дуже популярною, завдяки інноваційному 3D-геймплею та атмосфері пригод. Її бізнес-модель також базується на внутрішньоігрових покупках.
- **Виявлення ключових факторів успіху на ринку мобільних ігор:**
  - **Простий та захопливий геймплей:** Ігри повинні бути легкими для освоєння, але складними для досягнення високих результатів.
  - **Яскрава та приваблива графіка:** Візуальний дизайн відіграє важливу роль у залученні гравців.
  - **Соціальна інтеграція:** Можливість змагатися з друзями та ділитися досягненнями підвищує залучення.
  - **Ефективна монетизація:** Бізнес-модель повинна бути збалансованою та не відштовхувати гравців.
  - **Регулярні оновлення:** Додавання нового контенту та функцій підтримує інтерес гравців.

### **2.5.2. Соціальні аспекти мобільних ігор жанру "Endless Runner"**

Соціальні функції відіграють важливу роль у залученні та утриманні гравців у мобільних іграх, особливо в жанрі "endless runner". Вони створюють відчуття спільноти, стимулюють змагання та підвищують залучення.

- **Дослідження впливу соціальних функцій на залучення гравців:**
  - **Таблиці лідерів:**
    - Таблиці лідерів дозволяють гравцям порівнювати свої результати з результатами інших гравців, що стимулює змагання та прагнення до досягнення високих результатів.
    - Вони створюють відчуття конкуренції та мотивують гравців покращувати свої навички.
  - **Змагання з друзями:**
    - Можливість змагатися з друзями робить гру більш захоплюючою та соціальною.
    - Гравці можуть ділитися своїми досягненнями та змагатися за високі результати.
  - **Обмін досягненнями:**
    - Можливість ділитися своїми досягненнями в соціальних мережах або через внутрішньоігрові функції підвищує соціальну взаємодію та залучення.
    - Гравці можуть демонструвати свої навички та отримувати визнання від інших гравців.
- **Аналіз впливу соціальних мереж та стримінгових платформ на популярність мобільних ігор:**

- **Соціальні мережі:**
  - Соціальні мережі, такі як Facebook, Twitter та Instagram, використовуються для обміну інформацією про ігри, демонстрації досягнень та створення спільнот гравців.
  - Вони також використовуються для реклами та просування ігор.
- **Стримінгові платформи:**
  - Стримінгові платформи, такі як Twitch та YouTube, використовуються для демонстрації ігрового процесу та створення контенту, пов'язаного з іграми.
  - Вони допомагають залучити нових гравців та підвищити популярність ігор.
- **Використання соціальних функцій для створення спільноти гравців:**
  - **Внутрішньоігрові чати та форуми:**
    - Внутрішньоігрові чати та форуми дозволяють гравцям спілкуватися між собою, обмінюватися порадами та створювати спільноти.
    - Вони сприяють соціальній взаємодії та підвищують залучення.
  - **Гільдії та клани:**
    - Гільдії та клани дозволяють гравцям об'єднуватися в групи та грати разом.
    - Вони створюють відчуття спільноти та сприяють соціальній взаємодії.
  - **Події та турніри:**
    - Події та турніри дозволяють гравцям змагатися між собою та отримувати нагороди.

- Вони створюють відчуття змагання та підвищують залучення.

Соціальні функції є важливим елементом успішних мобільних ігор, тому їх використання, є важливим.

### **2.5.3. Монетизація мобільних ігор жанру "Endless Runner"**

Монетизація є важливим аспектом розробки мобільних ігор, оскільки вона дозволяє розробникам отримувати прибуток та підтримувати подальший розвиток гри.

- **Аналіз різних моделей монетизації:**

- **Реклама:**

- Реклама є однією з найпоширеніших моделей монетизації в мобільних іграх.
- Вона може включати банерну рекламу, відеорекламу та інтерактивну рекламу.
- Перевага реклами полягає в тому, що вона не вимагає від гравців прямих витрат.
- Однак, надмірна кількість реклами може відштовхнути гравців.

- **Внутрішньоігрові покупки:**

- Внутрішньоігрові покупки дозволяють гравцям купувати віртуальні предмети, бонуси та інші переваги.
- Ця модель є дуже популярною в жанрі "endless runner", де гравці можуть купувати нових персонажів, покращення та бонуси.
- Перевага внутрішньоігрових покупок полягає в тому, що вони дозволяють гравцям персоналізувати свій досвід та покращити свої результати.

- Однак, важливо підтримувати баланс, щоб гра не стала "pay-to-win".
- **Підписки:**
  - Підписки дозволяють гравцям отримувати доступ до ексклюзивного контенту або функцій за щомісячну плату.
  - Ця модель менш поширена в жанрі "endless runner", але вона може бути використана для надання гравцям ексклюзивних персонажів, рівнів або бонусів.
- **Дослідження впливу різних моделей монетизації на залучення та утримання гравців:**
  - Важливо враховувати, що агресивна монетизація може відштовхнути гравців та зменшити їхнє залучення.
  - Збалансована монетизація, яка не порушує ігровий процес та не створює відчуття "pay-to-win", є більш ефективною для утримання гравців.
  - Важливо проводити тестування та аналізувати дані, щоб визначити, які моделі монетизації є найбільш ефективними для вашої гри.
- **Розробка стратегії монетизації, яка відповідає цільовій аудиторії та особливостям гри:**
  - Стратегія монетизації повинна відповідати цільовій аудиторії гри.
  - Наприклад, якщо ваша гра орієнтована на випадкових гравців, то реклама може бути більш ефективною, ніж внутрішньоігрові покупки.
  - Якщо ваша гра орієнтована на хардкорних гравців, то внутрішньоігрові покупки можуть бути більш ефективними.

- Важливо враховувати особливості гри, такі як геймплей, візуальний стиль та соціальні функції.
- Наприклад, якщо ваша гра має велику кількість персоналізації, то внутрішньоігрові покупки для персоналізації можуть бути ефективними.

## **РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ГРИ “ENDLESS RUNNER” НА UNREAL ENGINE 5**

### **3.1. Планування та підготовка до розробки**

Для розробки прототипу/базової гри необхідно визначити основні механіки гри та знайти або створити асети для використання. Один з сервісів, який надає безкоштовні 3D асети це Fab, офіційний ресурс від компанії Epic Games.

#### **3.1.1. Визначення основних механік та функціоналу гри**

Основний функціонал гри:

- Безперервний рух персонажа, зі здатністю змінювати лінії, ковзати під перешкодами, та стрибати.
- Нескінченний спавн шляху з перешкодами, яким рухатиметься персонаж.
- Підбирання підсилень та ігрової валюти.
- Можливість вибору персонажа.
- Можливість відродження після смерті за ігрову валюту.

Додавання UI елементів

Головне меню, те що бачить гравець при запуску гри, через яке запускається гра, або вихід з неї.

UI елементи під час гри: кількість зібраної валюти, рекорд здобутих балів, та теперішня кількість здобутих балів.

Меню, що з'являється після смерті: пропонує відродження, рестарт, або вихід до головного меню.

### **3.1.2 Налаштування проекту Unreal Engine 5**

Для запаковування проекту для на андроїд треба попередньо встановити Android studio, SDL, NDK, JDK.

Завантаження Android Studio відбувається через [developer.android.com](https://developer.android.com), підійде остання версія програми.

По завершенню завантаження в самій програмі обирається завантаження SDK та NDK та версії Android.

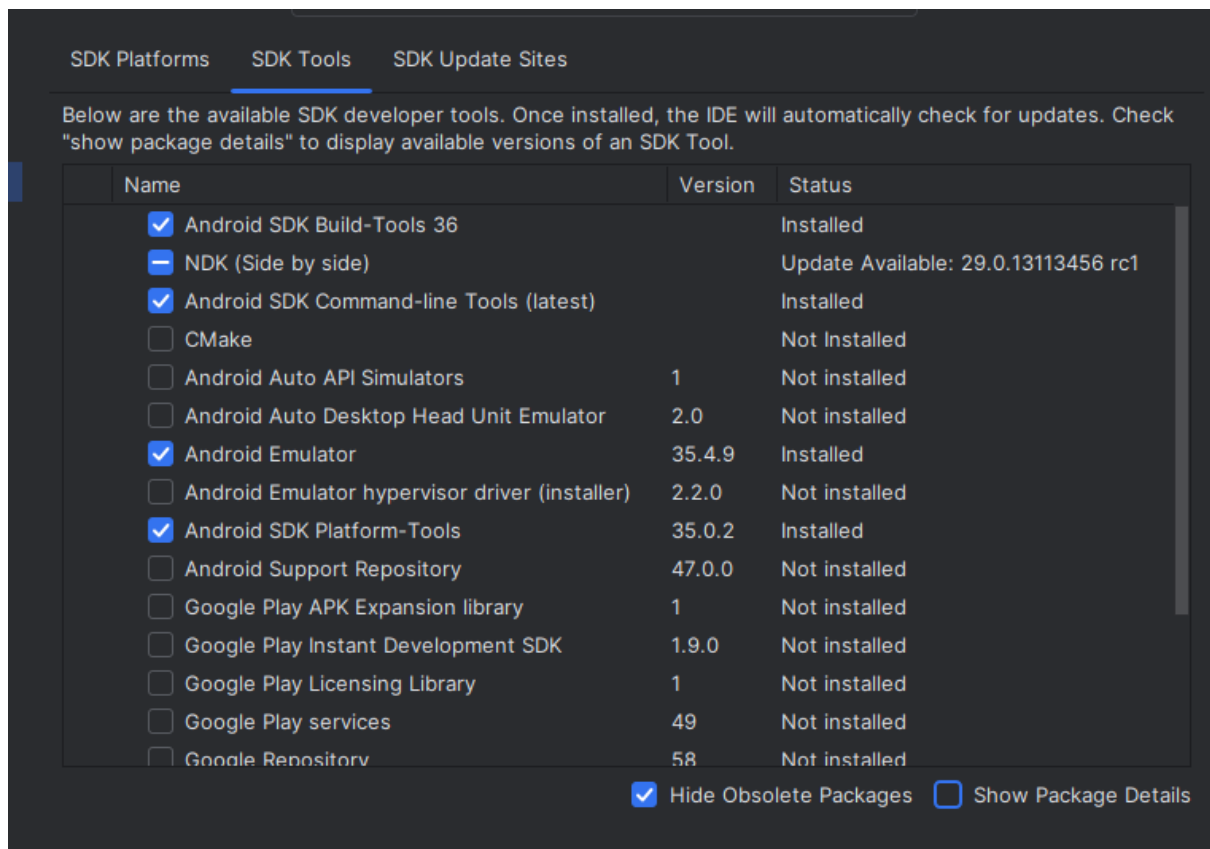


рис. 3.1.2.1.

JDK завантажити з сайту [oracle.com](https://oracle.com), шлях завантаженої теки додається в Environment Variables з назвою JAVA\_HOME

| Variable  | Value                    |
|-----------|--------------------------|
| JAVA_HOME | D:\Android\jdk-17.0.6+10 |

рис. 3.1.2.2.

Безпосередньо в едіторі Unreal Engine в project settings→Platforms→Android SDK встановлюємо шляхи до завантажених SDK, NDK та JDK.

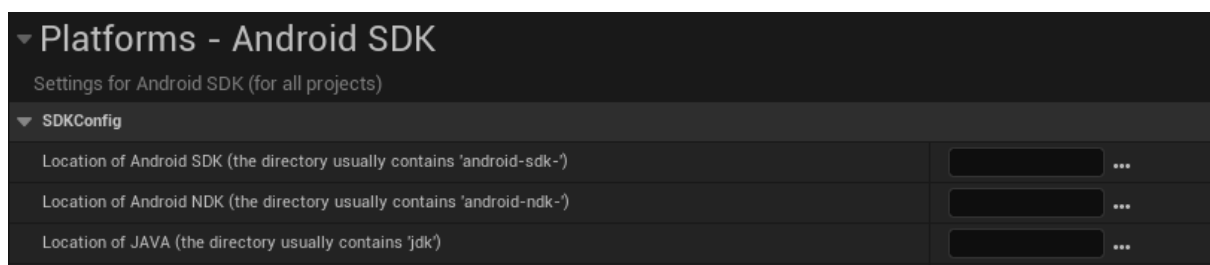


рис. 3.1.2.3.

Базова локація SDK та NDK - AppData/Local/Android/Sdk та AppData/Local/Android/Sdk/ndk/версія, відповідно

Тепер в Viewport→Platforms після оновлення статусу платформ має з'явитись іконка андроїду

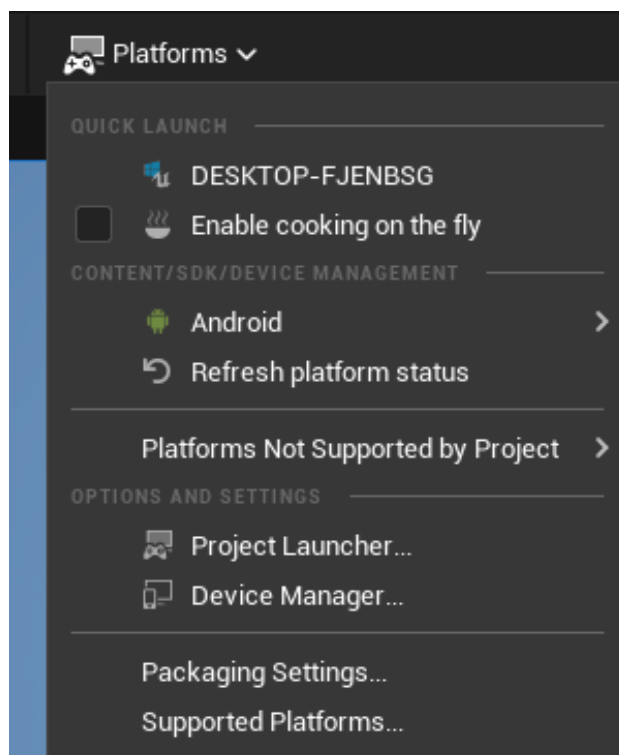


рис. 3.1.2.4.

В Project Settings →Projects→Packaging під розділом Packaging розгортаємо налаштування Advanced всередині в List of maps to include in a packaged build додаємо всі мапи що використовуються в грі.

В розділі Project обираємо Build Configuration → Shipping. В Project Settings→Platforms→Android в розділі APK Packaging першим стоїть перевірка

конфігурації , треба натиснути **Configure Now**, якщо горить червоним



рис. 3.1.2.5.

Наступним в тому ж розділі обов'язково вводимо **Android Package Name** по шаблону який представлено, виставляємо **minimum SDK version** та **Target SDK version**, **Target SDK Version** необхідно встановити відповідний з **Android Studio**. Поставимо галочку навпроти **Package game data inside .apk?**, щоб запаковану гру можна було повністю встановити за допомогою арк-файлу.

|                                                                                       |                                                  |
|---------------------------------------------------------------------------------------|--------------------------------------------------|
| Build Folder                                                                          | <input type="button" value="Open Build Folder"/> |
| Android Package Name ('com.Company.Project', [PROJECT] is replaced with project name) | <input type="text"/>                             |
| Store Version (1-2147483647)                                                          | <input type="text" value="1"/>                   |
| Store Version offset (arm64)                                                          | <input type="text" value="0"/>                   |
| Store Version offset (x86_64)                                                         | <input type="text" value="0"/>                   |
| Application Display Name (app_name), project name if blank                            | <input type="text"/>                             |
| Version Display Name (usually x.y)                                                    | <input type="text" value="1.0"/>                 |
| Minimum SDK Version (26=8.0.0, 27=8.1.0, 28=9, 29=10, 30=11, 31=12)                   | <input type="text" value="26"/>                  |
| Target SDK Version (26=8.0.0, 27=8.1.0, 28=9, 29=10, 30=11, 31=12)                    | <input type="text" value="34"/>                  |
| Install Location                                                                      | <input type="text" value="Auto"/>                |
| Enable Lint depreciation checks                                                       | <input type="checkbox"/>                         |
| Package game data inside .apk?                                                        | <input checked="" type="checkbox"/>              |

рис. 3.1.2.6.

Це усі необхідні налаштування для пакування проекту. Тепер перейдемо до реалізації геймплею.

## 3.2. Розробка ключових механік гри

### 3.2.1. Створення Персонажа та реалізація механік руху

Створюємо блюпрінт з parent class - Character, називаємо BP\_RunnerCharacter

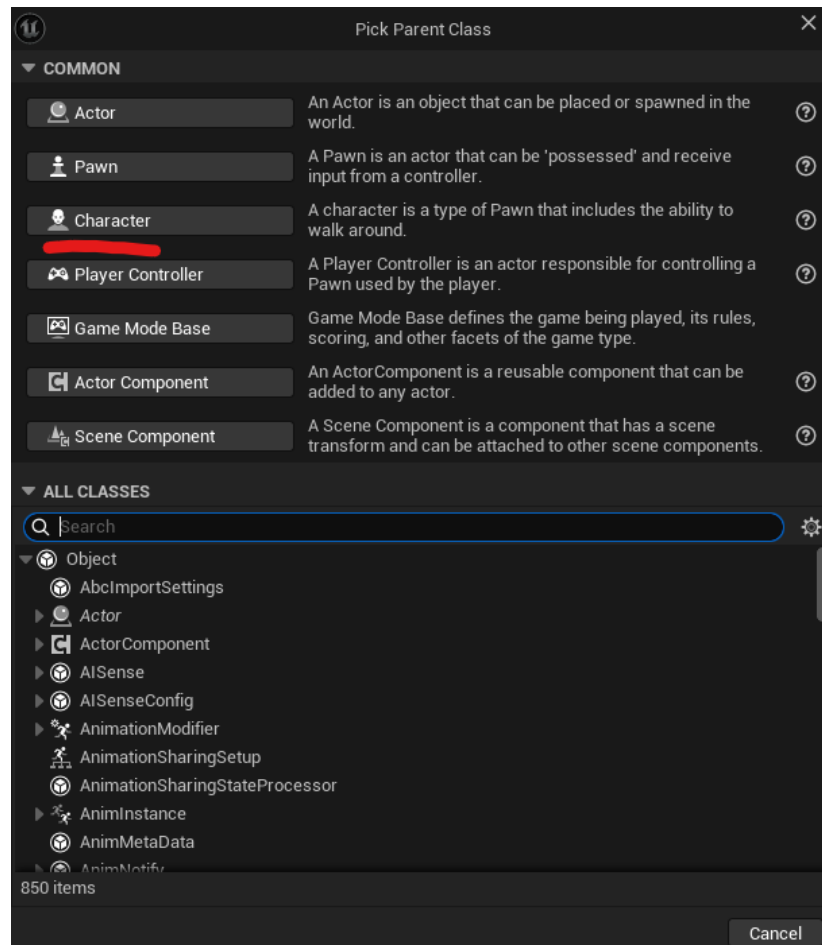


рис. 3.2.1.1.

Всередині в Components, до self додаємо Spring Arm і Followed Camera до компонента Spring Arm.

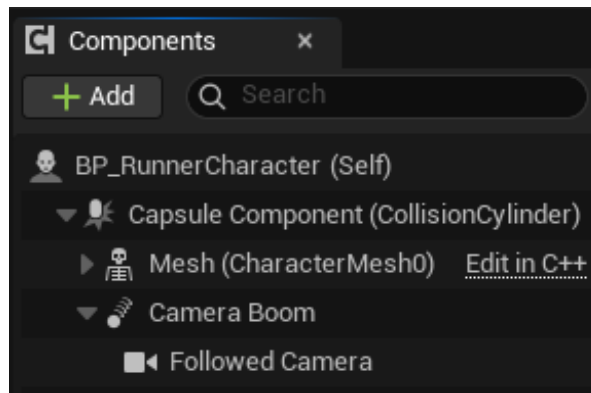


рис. 3.2.1.2.

В mesh component обираємо skeletal mesh asset манекена від Unreal Engine SKM\_Manny, він йде в комплекті з базовими анімаціями – ABP\_Manny, обираємо його в розділі Animation→Anim Class. В spring arm встановлюємо значення target arm length на 400, та do collision test на false

Переходимо в Event Graph, до Tick додаємо node Add movement input. Отримуємо обертання персонажа за допомогою Get Control Rotation, return value підключаємо до Break Rotator, Z підключаємо до Z в Make rotator, return value до Get Forward Vector, та до World Direction в Add Movement Input.

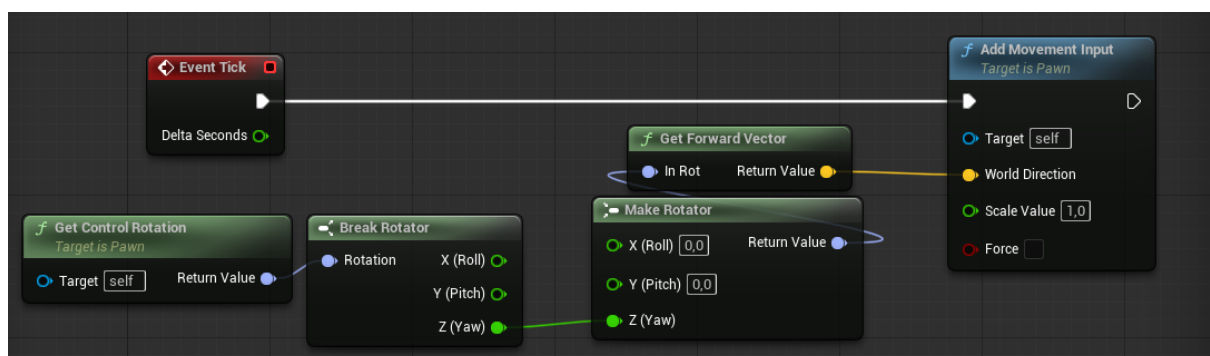


рис. 3.2.1.3

Із цим, персонаж рухатиметься вперед.

## Реалізація ковзання

Створюємо Input Mapping Context з назвою IMC\_RunnerDefault в окремій папці, там же створюємо Input Action - Touch\_IA.

Відкриваємо Mapping Context в Mappings додаємо Touch\_IA в Select the key value обираємо TOUCH 1.

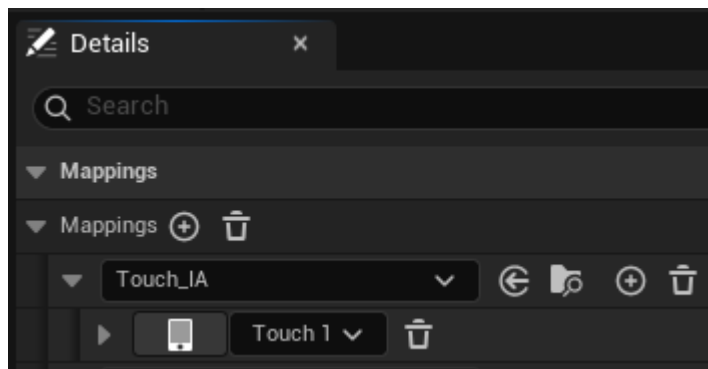


рис. 3.2.1.4.

Переходимо в BP\_RunnerCharacter, в Event Graph створюємо Custom Event з назвою SlideFunction, з Components переносимо на Event Graph Character Movement, з нього викликаємо Add Impulse, підключаємо ехес пін до SlideFunction.

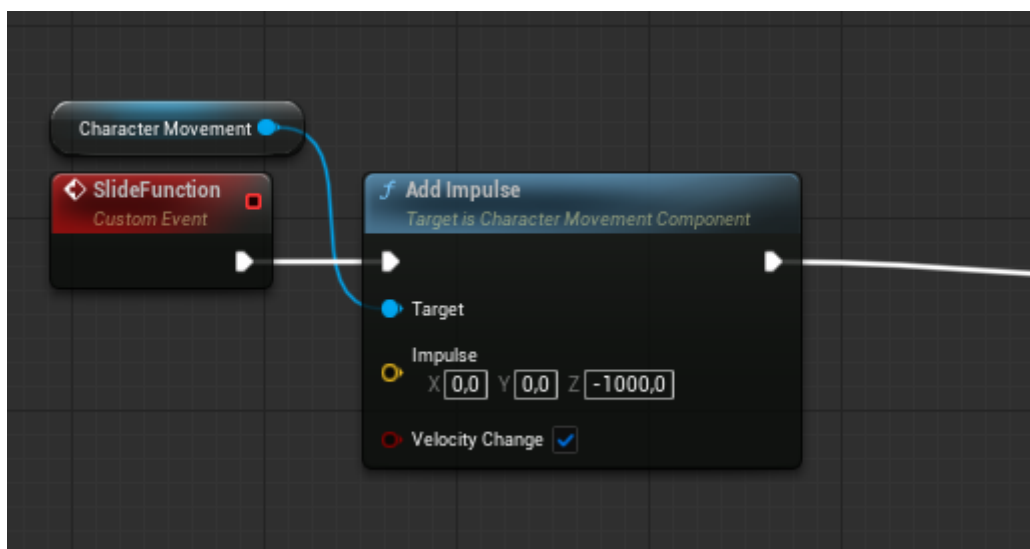


рис. 3.2.1.5.

З Add Impulse ехес пін додаємо Branch, для Condition, додаємо Character Movement, з нього викликаємо Get Velocity та IsFalling. З

Velocity викликаємо VectorLength з Return value викликаємо функцію Greater(>), в другий пін значення 60. З IsFalling викликаємо boolean Not. Підключаємо Return Value з Greater та Boolean not до AND Boolean, його return value підключаємо до Condition ноду Branch.

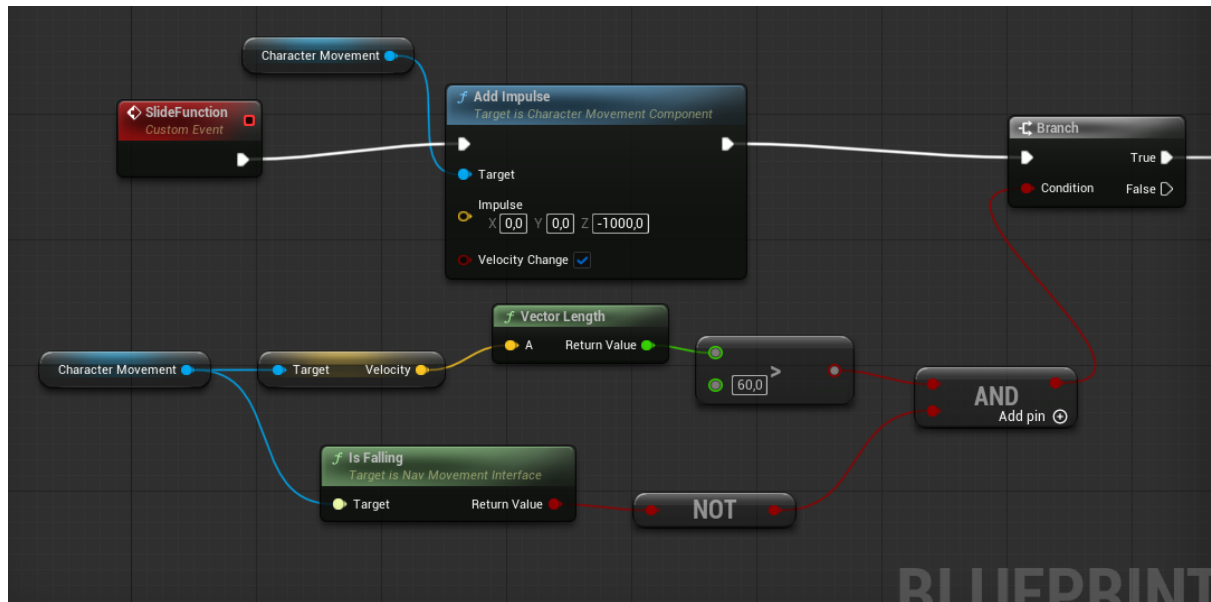


рис. 3.2.1.6.

З Branch True викликаємо Do Once нод, з Components перетаскуємо Capsule Component на вільне місце, з нього викликаємо нод Set Collision Response to Channel, підключаємо його до піну Completed ноду Do Once, New Response ставимо на Ignore.

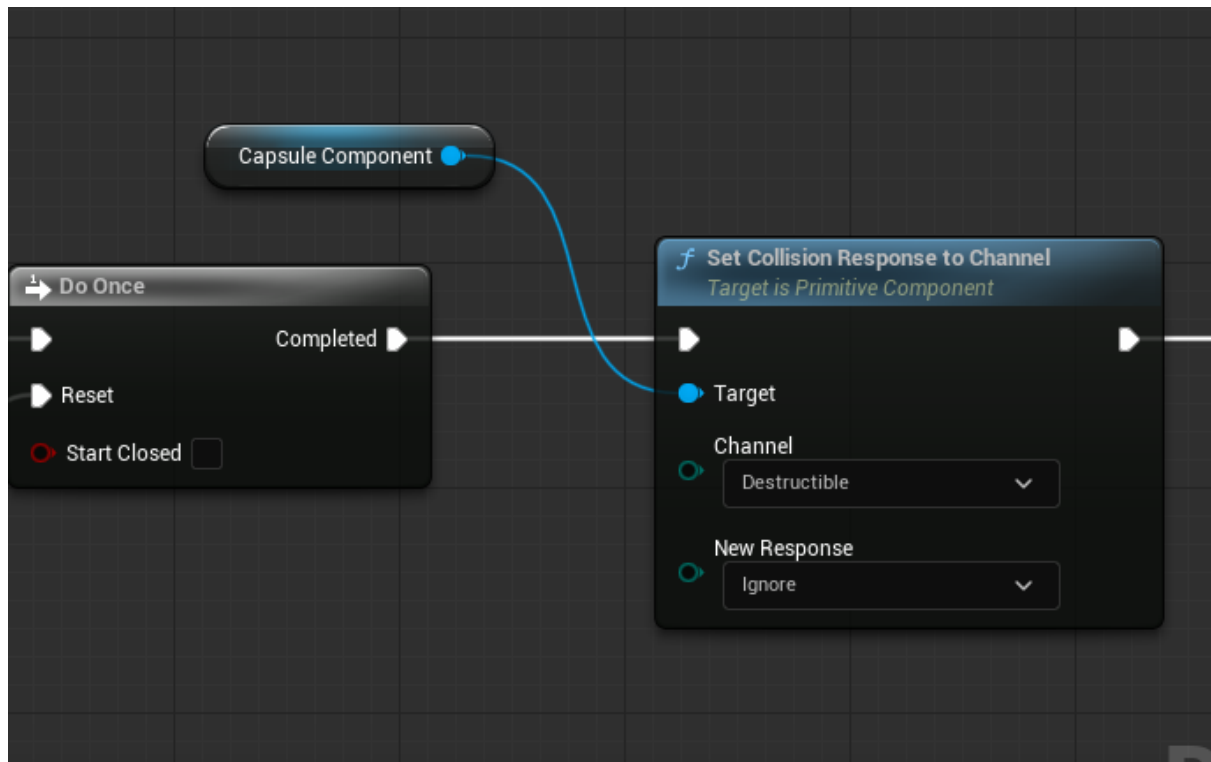


рис. 3.2.1.7.

Створюємо змінну `IsSliding` типу `boolean`, з Exec піну `Set Collision` викликаємо `Set IsSliding`, ставимо значення на `true`.

З `components` перетягуємо `mesh` на пусте місце, з нього викликаємо `Get Anim Instance`, з `Return Value` викликаємо `Montage Play`, exec пін `set IsSliding` підключаємо до `Montage Play`.

В пін `Montage to Play` обираємо `Animation Montage` існуючої анімації ковзання.

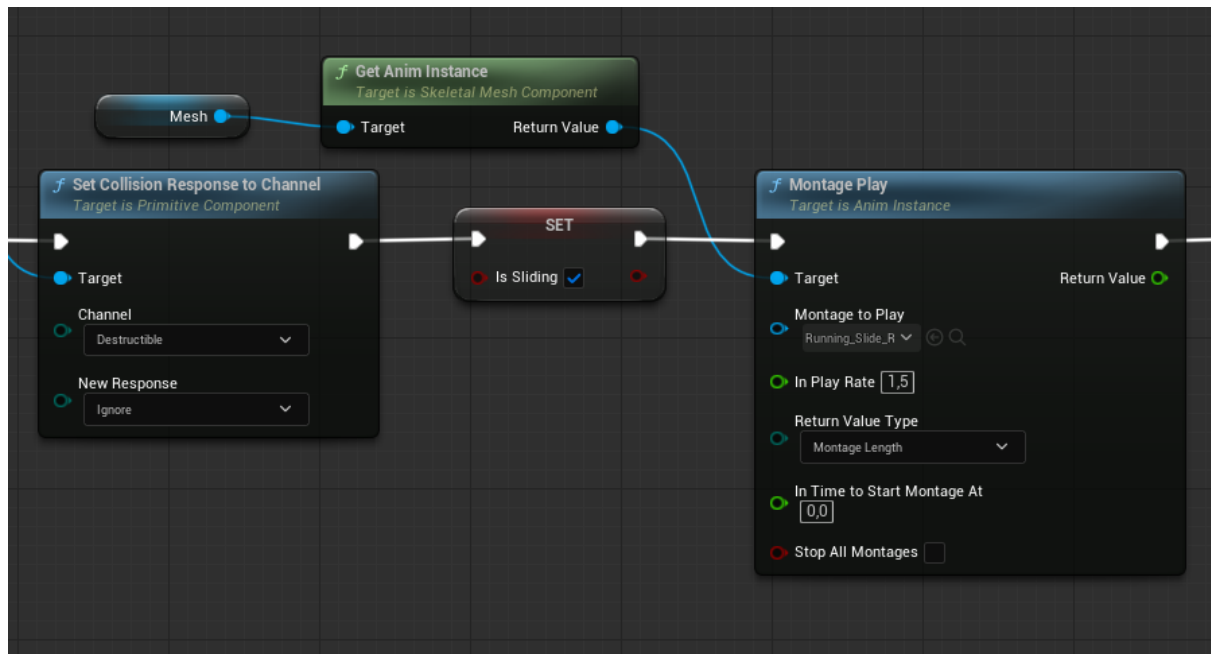


рис. 3.2.1.8.

З Montage Play ехес пін викликаємо delay 0.7 секунд, додаємо на пусте місце Capsule Component з нього додаємо Set Collision Response to Channel, New Response ставимо на Overlap, з нього додаємо Set IsSliding на false ехес пін підключаємо до Reset функції Do Once

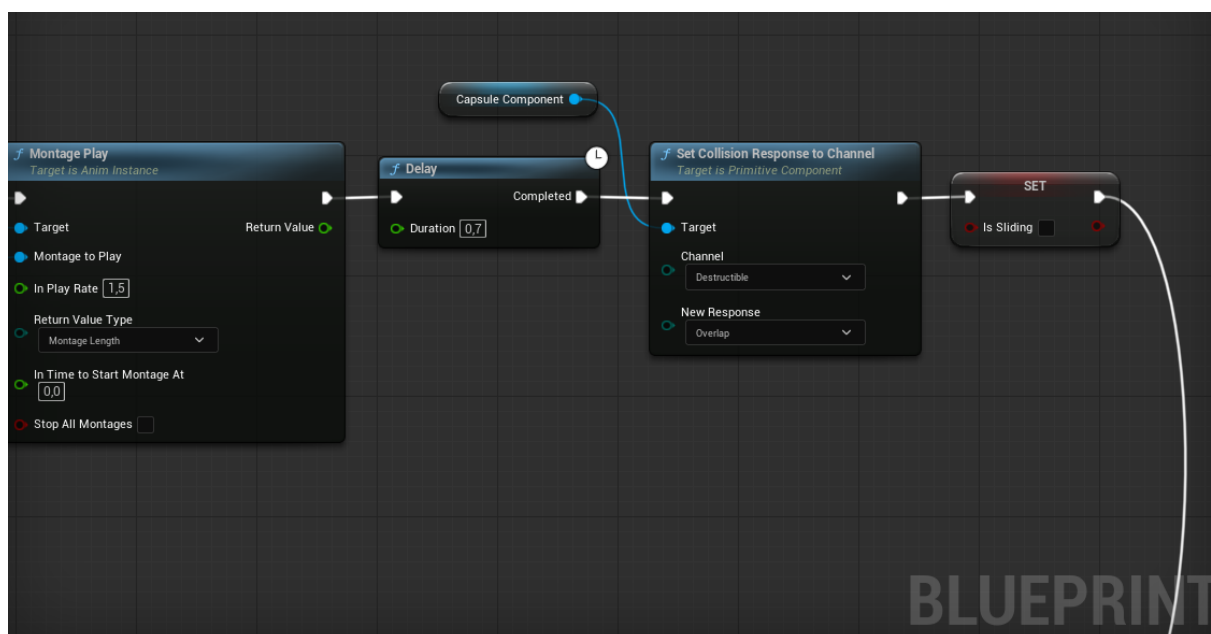


рис. 3.2.1.9.

## Реалізація зміни ліній (переміщення вправо/вліво)

В Event Graph створюємо Custom Event Change Lanes, натискаємо на створений івент, в деталях додаємо Inputs з типом int назвою Shift Lanes.

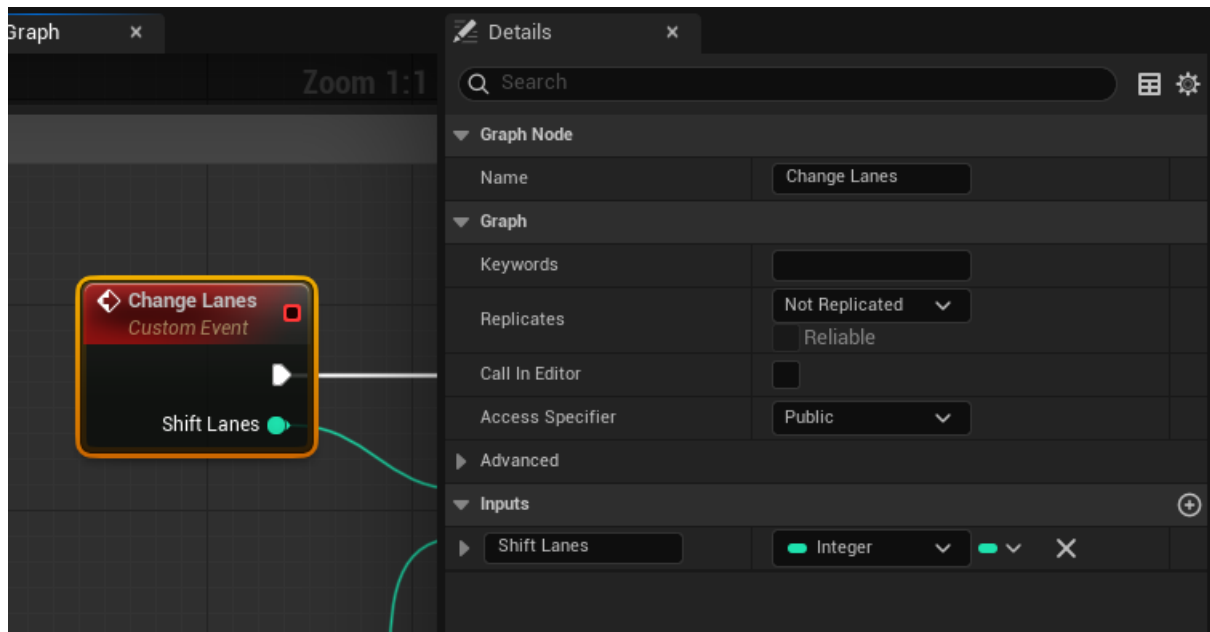


рис. 3.2.1.10.

Створюємо 2 змінні типу int: Current Index та Target Lane Index. Current Index базове значення ставимо 1 (лінія посередині). Викликаємо Set Target Lane Index, з'єднуємо з Exec пін івенту, до значення змінної під'єднуємо Clamp (Integer), min - 0, max - 2. До Value під'єднуємо нод Addition до неї під'єднуємо Shift Lanes з івенту та Current Index змінну.

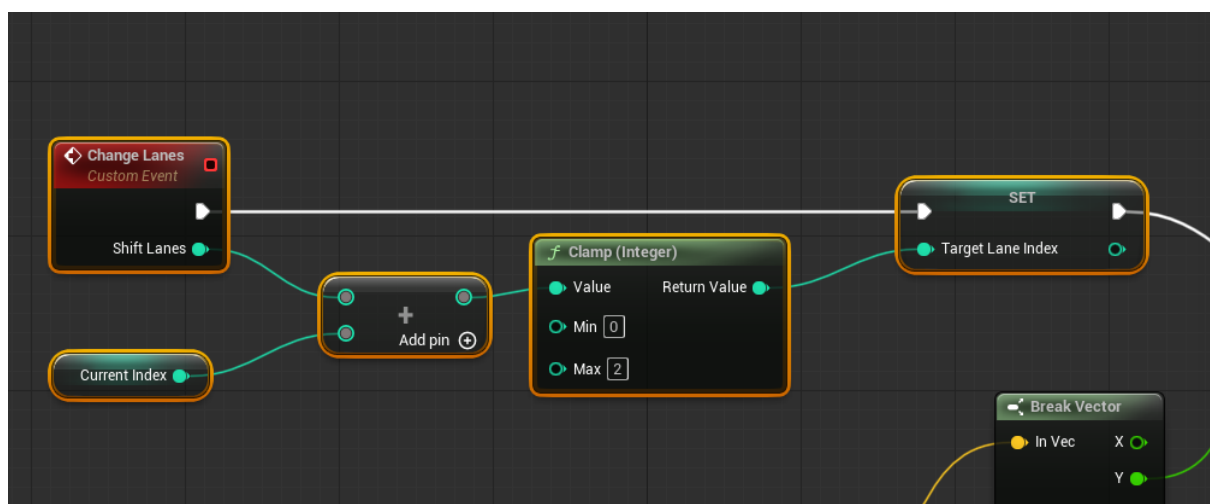


рис. 3.2.1.11.

Створюємо змінну типу float CurrentY Position, викликаємо Set нод цієї змінної, Exec пін до піну Set Target Lane Index. Отримуємо локацію персонажа через нод Get Actor Location, з Return value викликаємо Break Vector, під'єднуємо Y до піну змінної CurrentY Position.

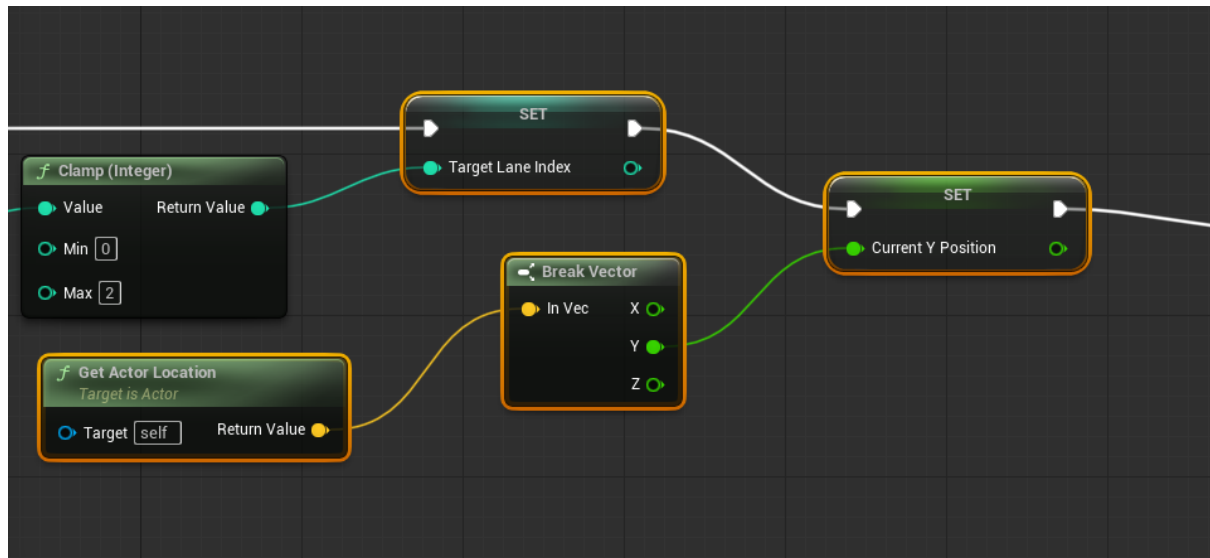


рис. 3.2.1.12.

З exec піну додаємо Timeline, підключаємо до Play From Start.

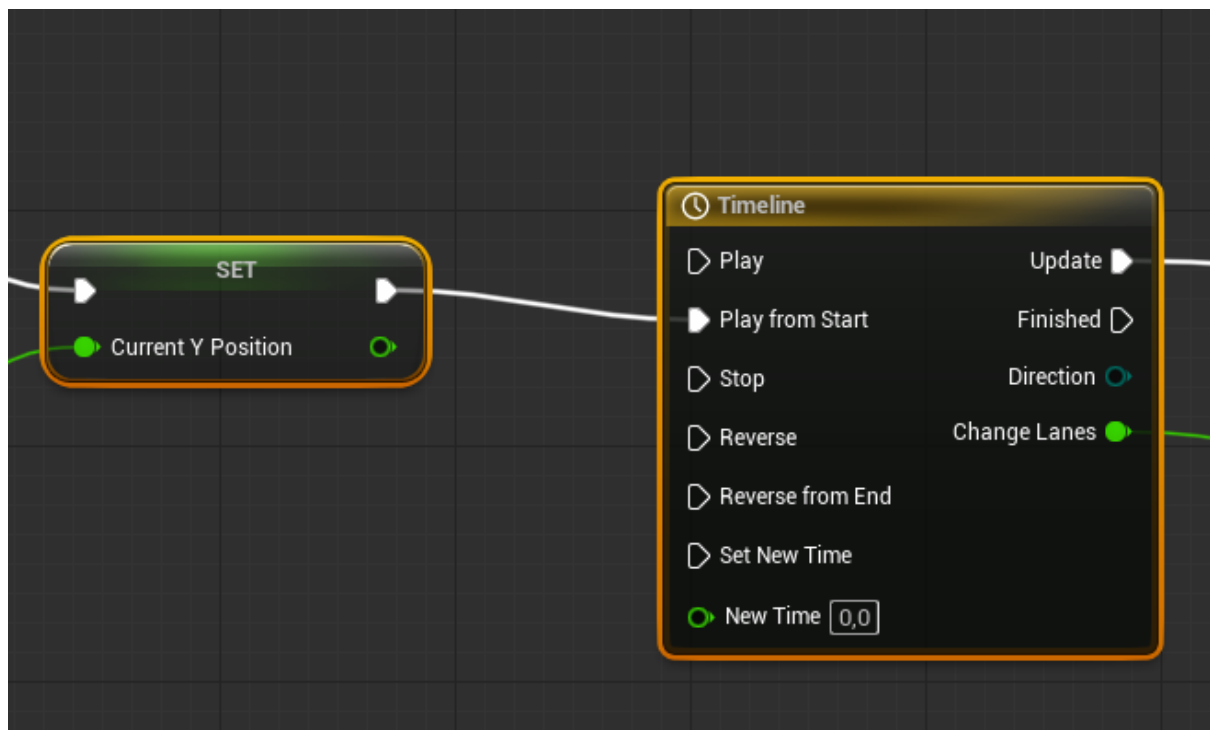


рис. 3.2.1.13.

Двічі натискаємо на Timeline, в редакторі таймлайну натискаємо + Track → Add Float Track, називаємо Change Lanes. Встановлюємо Length на 0.10, додаємо 2 ключових кадри (keyframes). 1 встановлюємо Time – 0, Value – 0. 2 встановлюємо Time – 0.1, Value – 1.

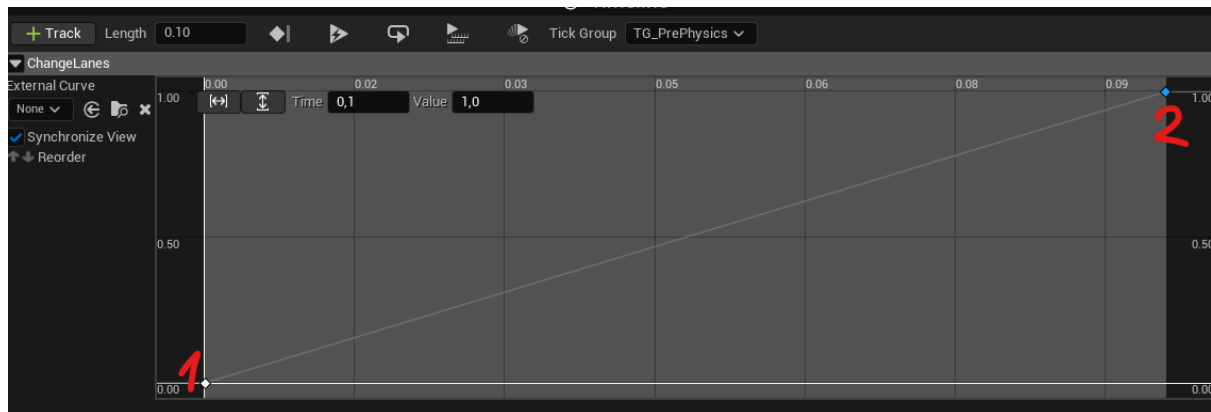


рис. 3.2.1.14.

з Update додаємо Set Current Index, до піну змінної під'єднуємо Target Lane Index. З exec піну Set викликаємо Set Actor Location, Sweep ставимо на True.

Для New Location піну:

Отримуємо локацію за допомогою Get Actor Location з нього Break Vector, потім Make Vector і під'єднуємо X і Z до відповідних пінів Return Value до New Location.

Створюємо змінну-масив LanePosition типу float, додаємо 3 базові значення: 0 індекс ставимо -325.0, 1 індекс – 0, 2 індекс – 325.

Додаємо нод Get (a copy) для масивів, в масив пін ставимо LanePosition, в індекс ставимо TargetLaneIndex. Додаємо нод Lerp (Linear Interpolation), А пін з'єднуємо зі змінною Current Y Position, В з output піном Get, Alpha з Change Lanes з Timeline, Return Value Lerp з'єднуємо з Y на ноді Make Vector.

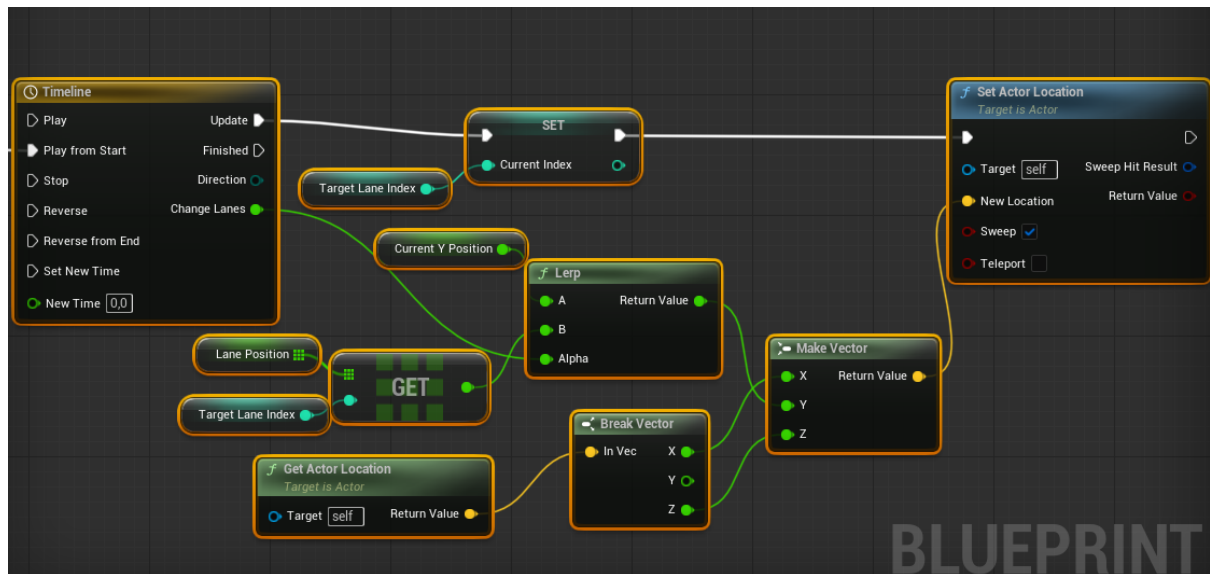


рис. 3.2.1.15.

### Реалізація дотиків по екрану

В Begin Play, отримуємо контроллер через Get Controller, викликаємо Cast to player controller, передаємо return get controller до Object, ехес пін з'єднуємо з Begin Play. З As Player Controller отримуємо Enhanced Input Local Player Subsystem, з ехес піну викликаємо Is Valid нод, до Input Object передаємо return з enhanced input subsystem. Is Valid пін з'єднуємо з нодом Add Mapping Context, Target передаємо Subsystem, в Mapping Context обираємо створений IMC\_RunnerDefault.

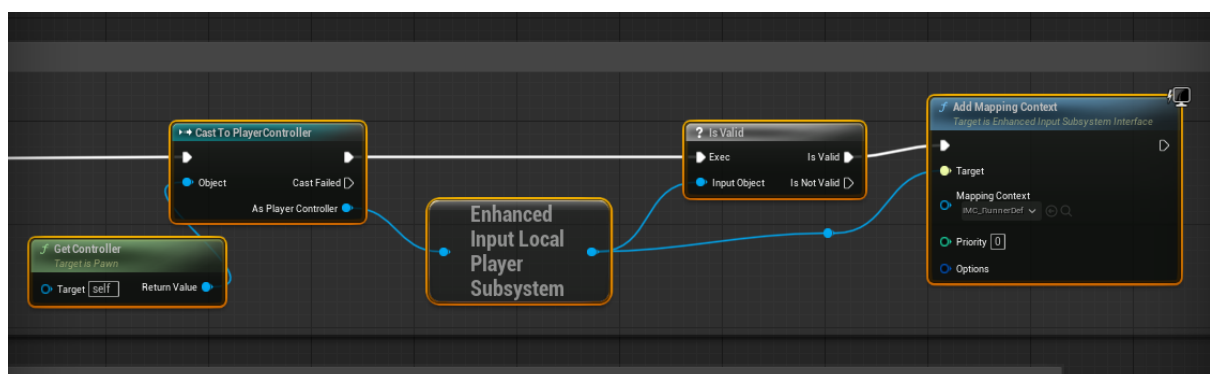


рис. 3.2.1.16.

Додаємо в Event Graph Enhanced Input Action Touch\_IA. Створюємо змінну типу Boolean IsTriggered та змінну Vector2d – TouchStart2D. В Started з Touch\_IA отримуємо Set IsTriggered, ставимо на True. З ехес пін отримуємо Set TouchStart2D. В значення – отримуємо Player Controller з return value → Get Input Touch State, Finger Index – Touch 1. Додаємо нод Make Vector 2D, з'єднуємо Location X, Location Y з Get Input Touch State з X та Y. Return Value в значення Set. Встановлюємо початкову точку дотику.

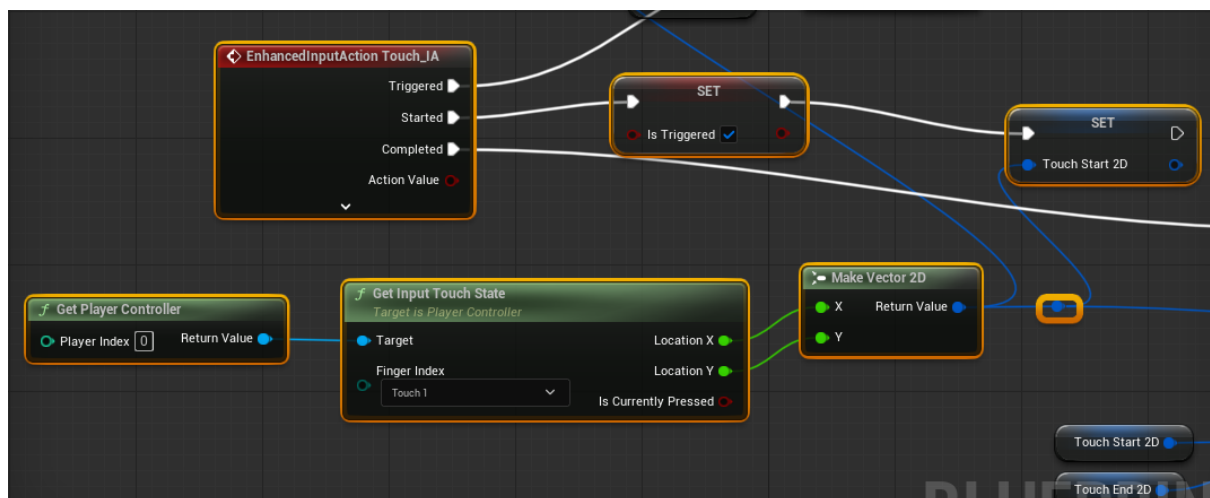


рис. 3.2.1.17.

З Triggered додаємо Branch, в condition під'єднуємо IsTriggered. З True додаємо ще один Branch. На пустому місці отримуємо Get TouchStart2D з нього викликаємо Subtract, в другий пін під'єднуємо MakeVector2D return value. З Return віднімання отримуємо Vector2D Length, з return value додаємо Greater(>) нод, в нижній пін ставимо значення 100, Return Value під'єднуємо до Condition другого Branch. Для запобігання випадкових дотиків, перевірка чи він був достатньо довгий для активації.

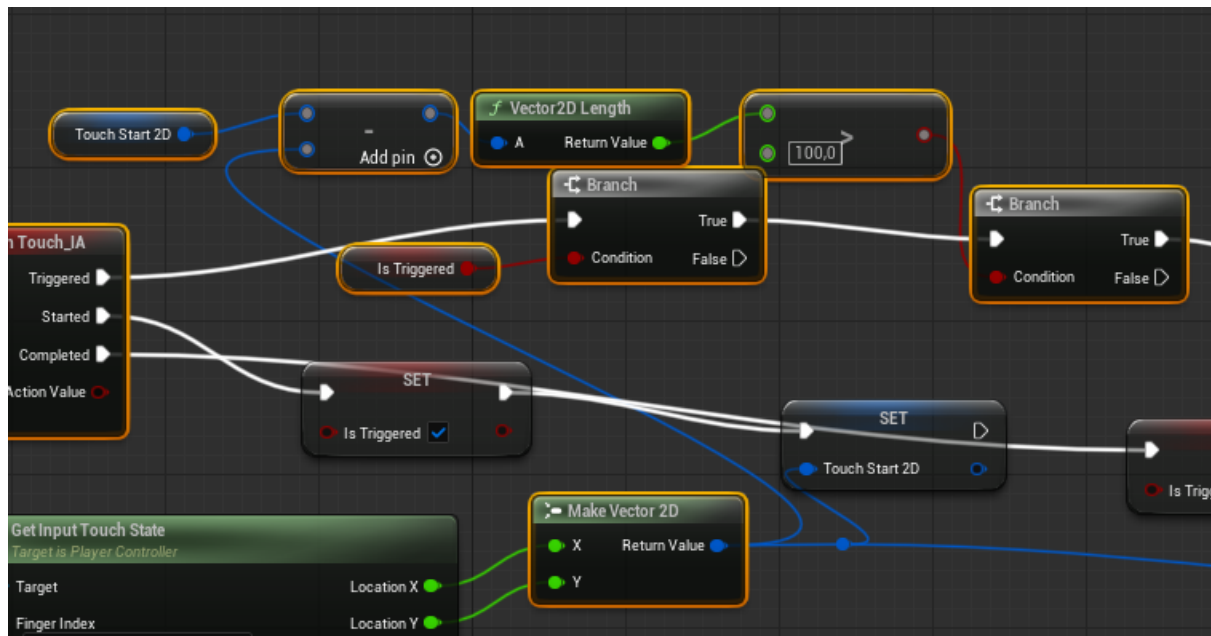


рис. 3.2.1.18.

Додаємо Do Once, під'єднуємо True з Branch до ехес піну. Створюємо змінну Touch End 2D типу Vector2D. З Completed піну додаємо Set TouchEnd2D, в значення змінної передаємо MakeVector2D.

З ехес піну додаємо Branch. На пустому місці додаємо GET TouchStart2D, та TouchEnd2D. Під'єднуємо їх до Subtract ноду, з return value додаємо BreakVector2D. З X та Y кожен під'єднуємо до Absolute (float) нод, з них додаємо Greater (>) нод, X до A піну, Y до B піну. Return Value під'єднуємо до Branch Condition.

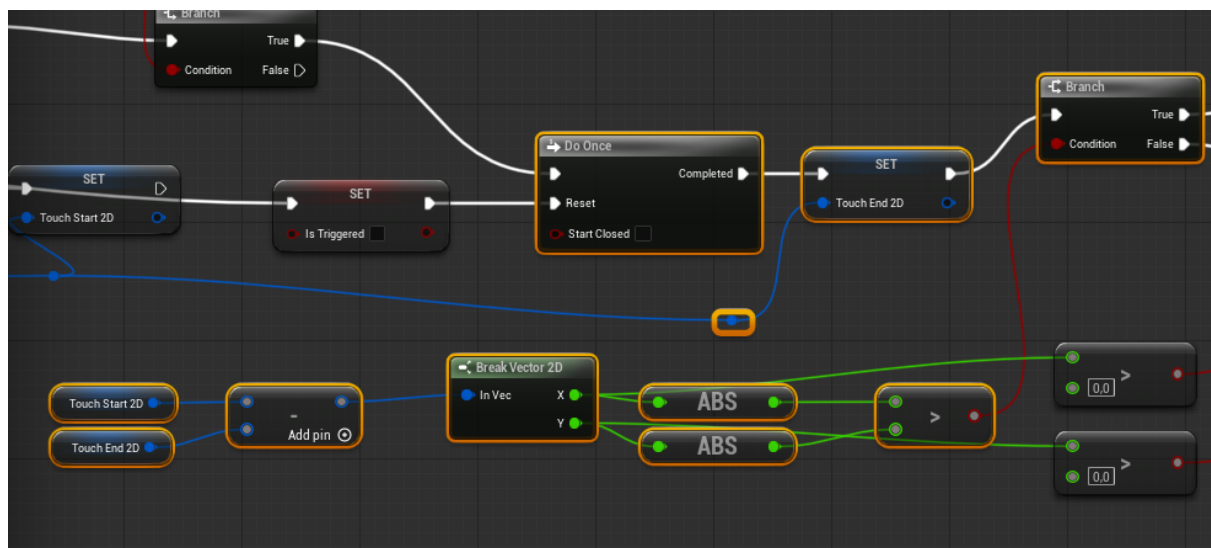


рис. 3.2.1.19.

З True додаємо ще один Branch. До Condition додаємо з Break Vector2D ноду пін X→Greater(>), В пін значення 0, Return Value під'єднуємо до умови.

З True додаємо івент Change Lanes зі значенням Shift Lanes -1. З False додаємо Change Lanes зі значенням Shift Lanes 1.

З False попереднього Branch додаємо ще один Branch. До Condition під'єднуємо пін Y з Break Vector2D, Greater(>) В пін значення 0, Return Value до умови.

З True додаємо нод Jump. З False додаємо івент SlideFunction.

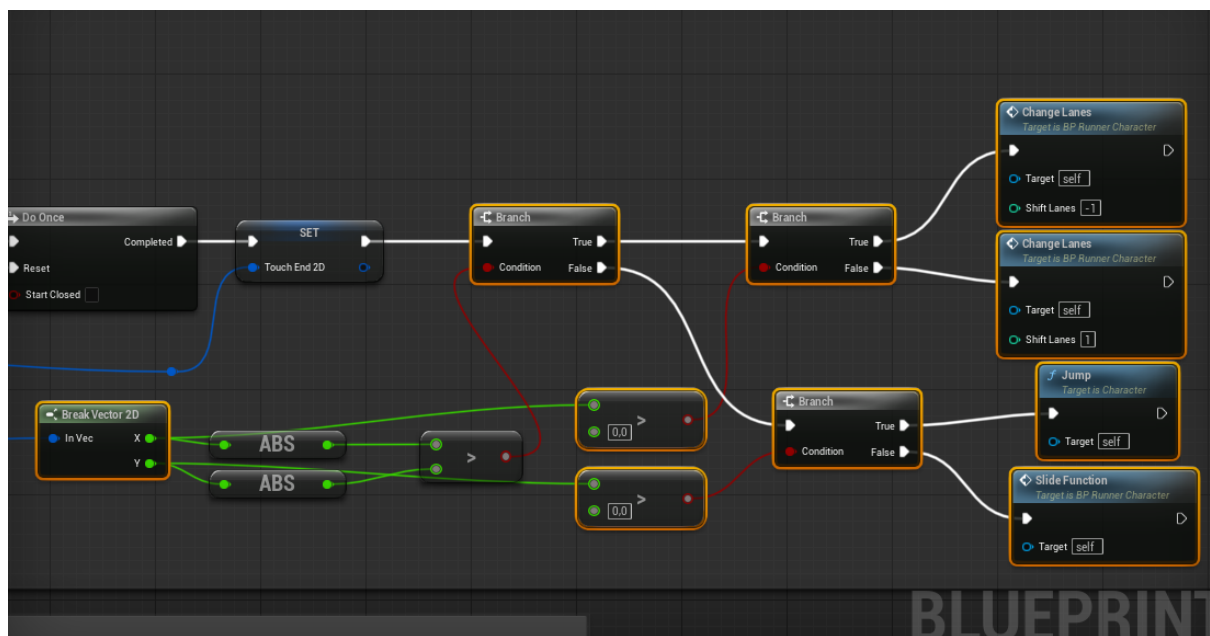


рис. 3.2.1.20.

### 3.2.2. Генерація шляху та перешкод

Створюємо блюпрінт з Parent class Actor, називаємо BP\_Floor. Всередині до Default scene root додаємо Static Mesh Cube (Floor) , Box Collision (Box) та Arrow

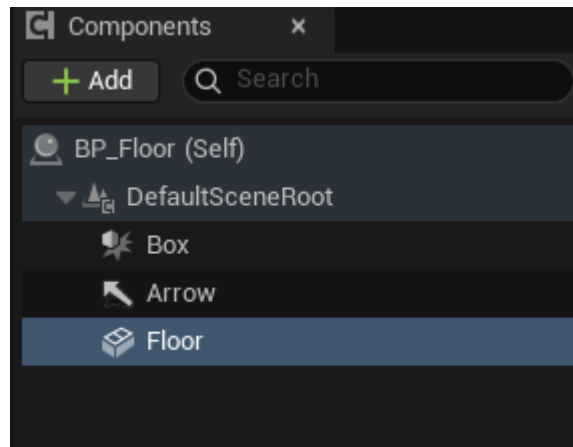


рис. 3.2.2.1.

Елемент Floor Location ставимо 0. 0. 0., rotation 90 градусів, scale 10, 10, 0.5.

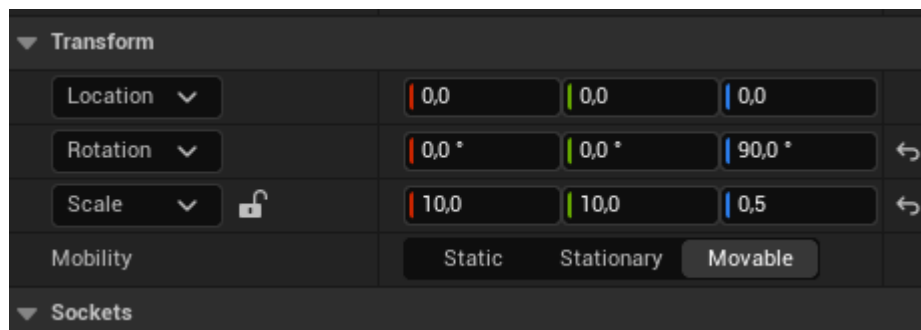
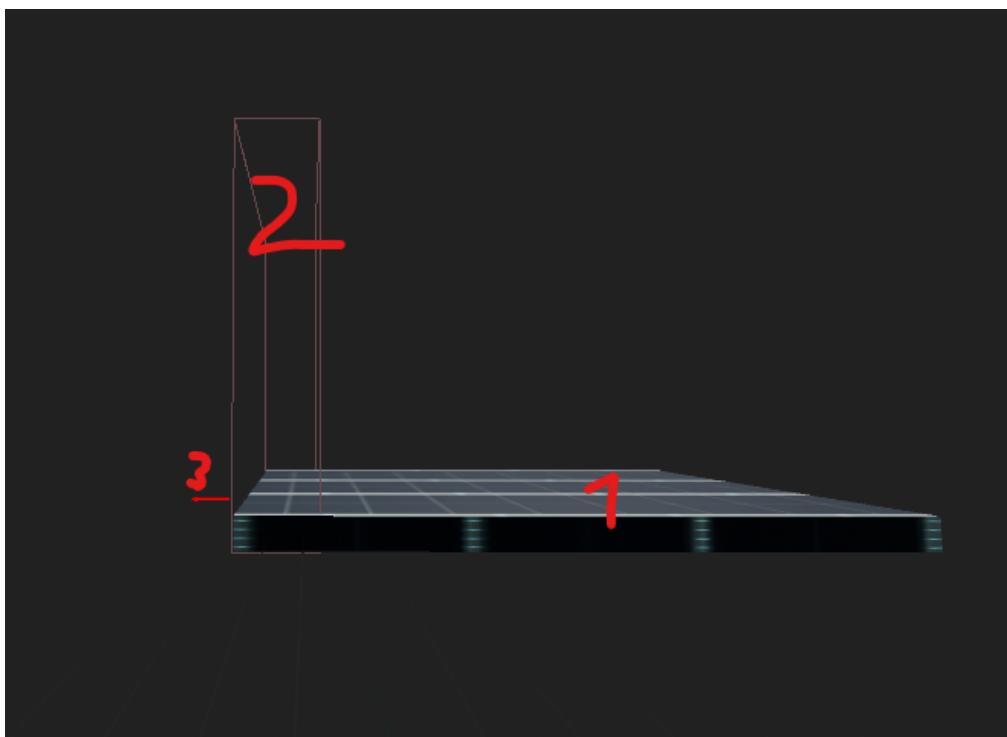


рис. 3.2.2.2.

Розширюємо Box Collision, збільшуємо Scale Y та Z, і ставимо його локацію на кінці Floor, трохи далі на рівні Floor (Location Z = 0) ставимо Arrow.



1 - Floor, 2 - Box Collision, 3 - Arrow

рис. 3.2.2.3.

В Event Graph створюємо функцію Get Attack Point, це стрілка Arrow, по локації якої буде генеруватись новий шлях

Всередині функції додаємо Return node, з Components перетаскуємо Arrow та викликаємо функцію Get World Transform, Return value під'єднуємо в return node.

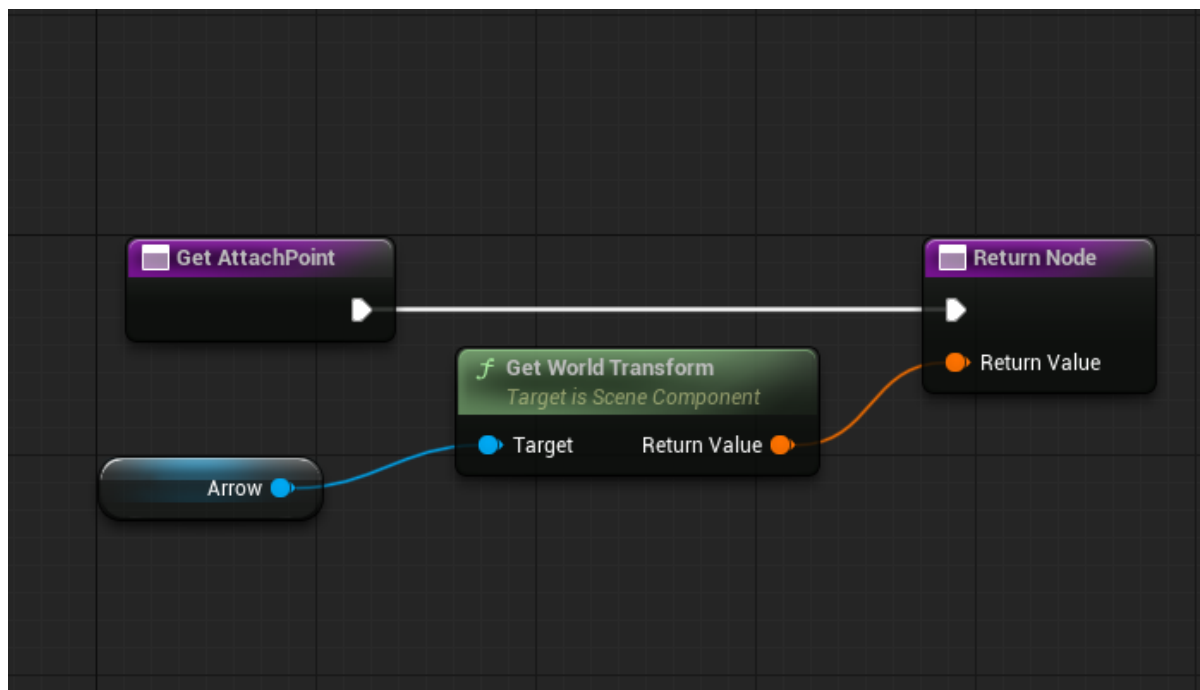


рис. 3.2.2.4.

В Components обираємо Box (box collision) в Details в кінці в розділі Events натискаємо + OnComponentBeginOverlap

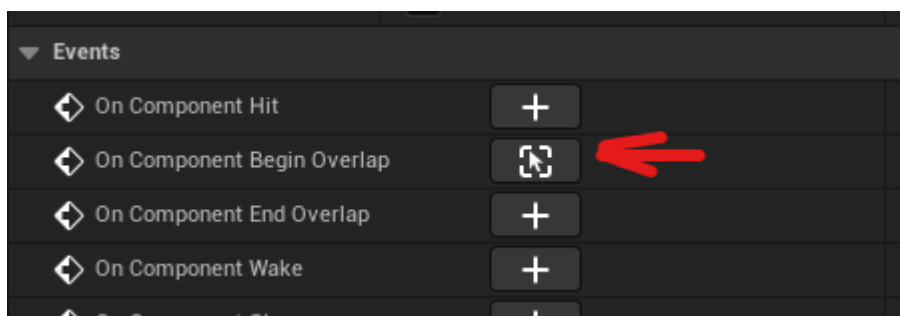


рис. 3.2.2.5.

В Event Graph має з'явитись Event OnComponentBeginOverlap, повернемось до нього потім.

Створюємо новий блюпрінт Game Mode Base - BP\_RunnerGameMode. В Project Settings встановлюємо Default Game mode на створений.

Всередині блюпрінта створюємо функцію з назвою Add Floor. В функції додаємо SpawnActor, обираємо клас BP\_Floor, зі SpawnTransform переносимо на пусте місце і в меню обираємо Promote to variable, називаємо Floor attach point.

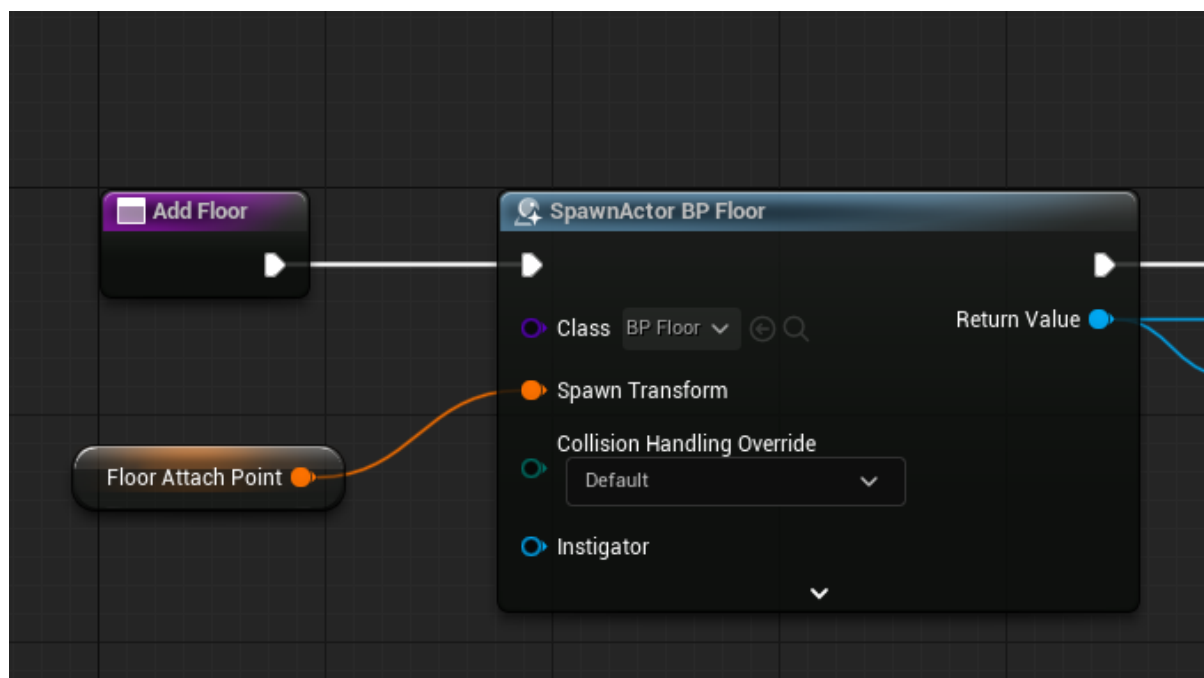


рис. 3.2.2.6.

З Return Value піну Spawn Actor викликаємо функцію яку створили в BP\_Floor - Get Attach Point, викликаємо Set змінної Floor Attach Point і return value функції під'єднуємо до Set.

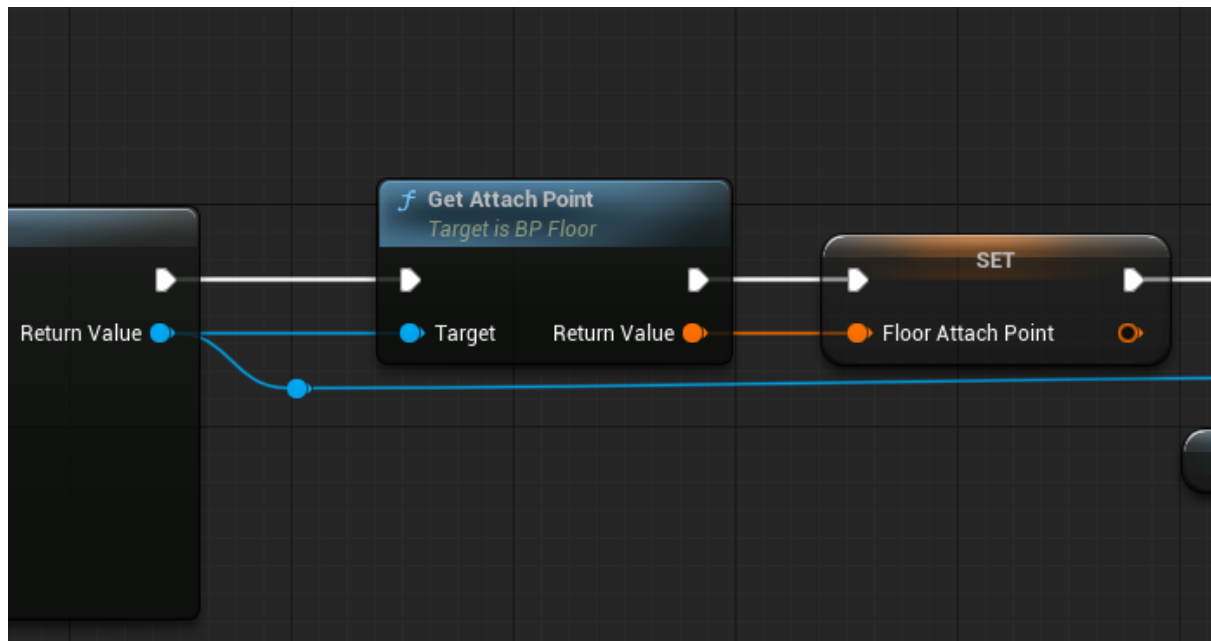


рис. 3.2.2.7.

Створюємо змінну Floor tiles типу BP\_Floor Array.

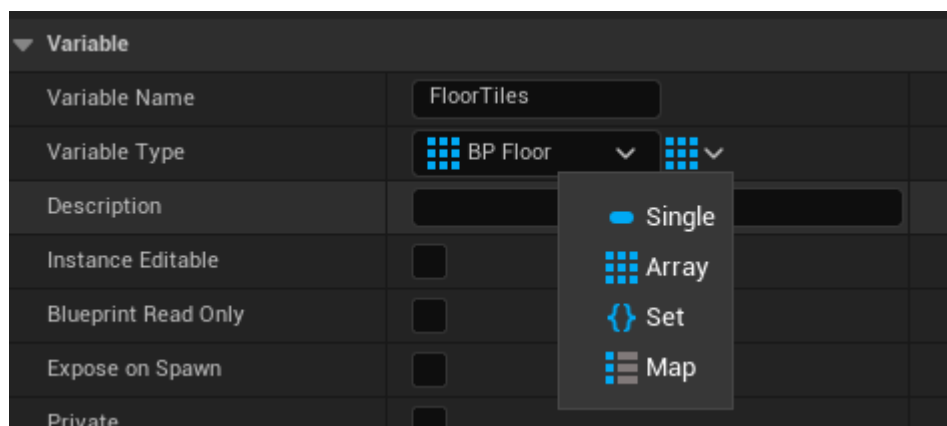


рис. 3.2.2.8.

Зі змінної Floor Tiles викликаємо нод Add для array, з піну exes, ноду Set для змінної Floor Attach Point під'єднуємо до Add, а Return value з SpawnActor до піну new item

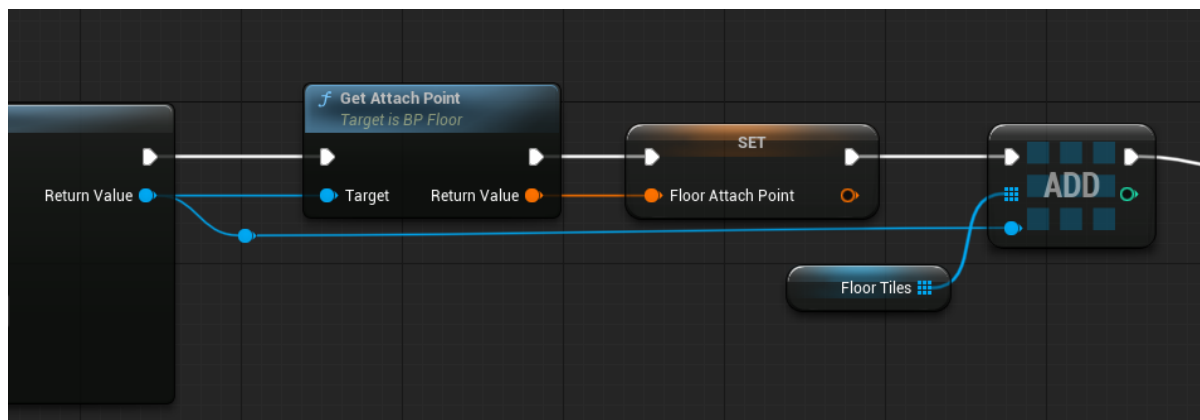


рис. 3.2.2.9.

Переходимо в EventGraph, з Begin Play додаємо нод Sequence, з нього For Loop, з Loop body додаємо функцію Add Floor.

Створюємо змінну типу int Number of Tiles, зі змінної додаємо нод віднімання, встановлюємо на 1 і під'єднуємо до Last index в For Loop.

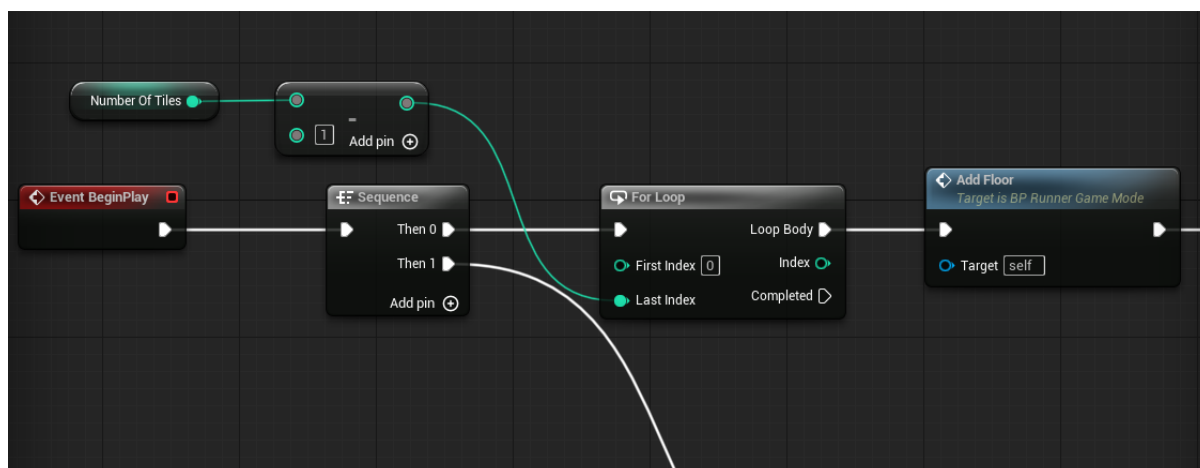


рис. 3.2.2.10.

Створюємо функцію Extend Floor, під'єднуємо до Add Floor, потім до Branch. Додаємо Get змінної Floor Tiles викликаємо функцію Length.

Додаємо Number of tiles функцію Addition (+), в B пін – 10.

Викликаємо функцію Greater Than (>), верхній пін підключаємо Return Value з Length, нижній - Return Value з Addition.

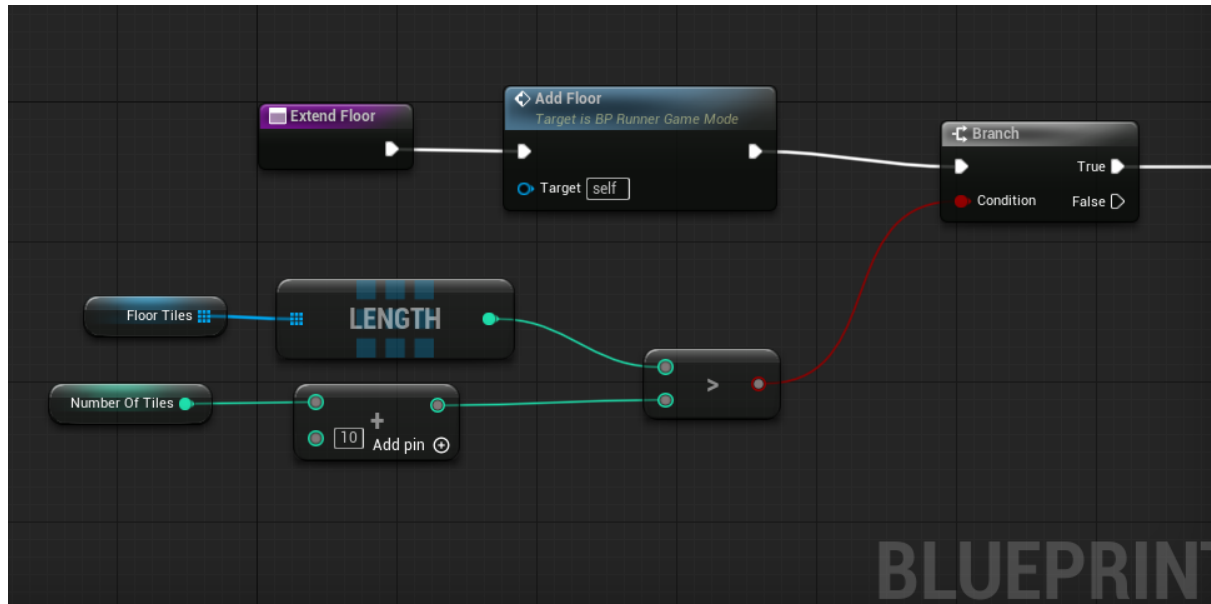


рис. 3.2.2.11.

З True викликаємо Destroy Actor, зі змінної Floor Tiles викликаємо Get за індексом 0, Output пін до Target піну Destroy Actor . З Exec піну Destroy викликаємо Remove для array, Target Array - Floor Tiles, Item - Output пін з Get.

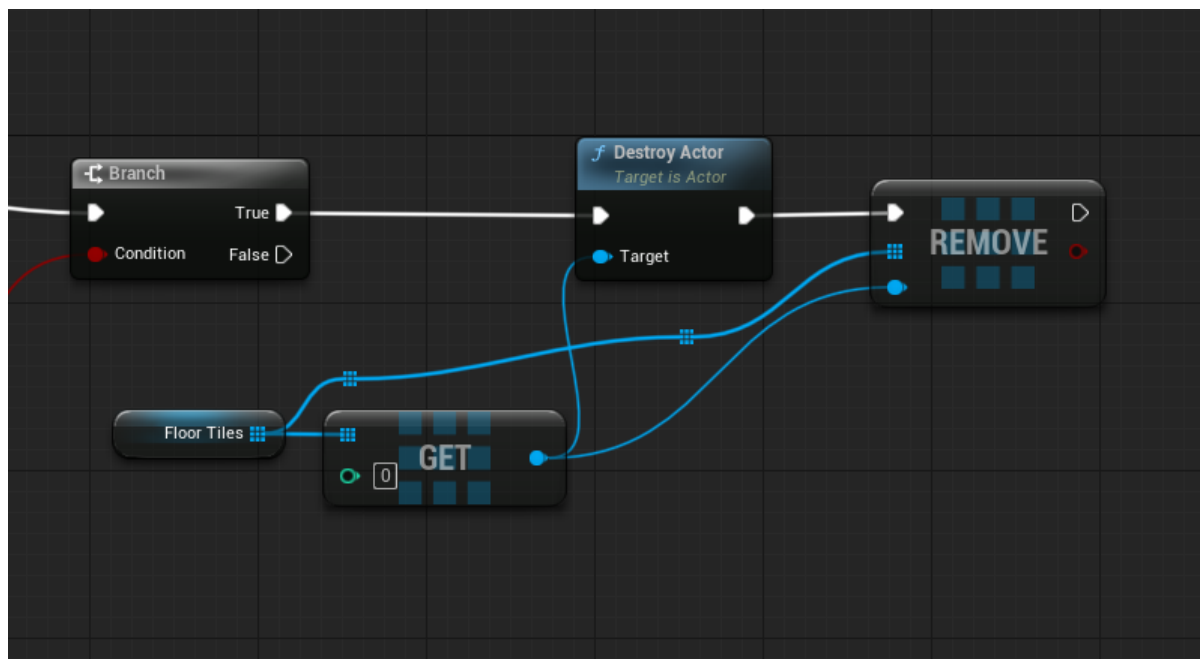


рис. 3.2.2.12.

Повертаємось до OnComponentBeginOverlap (Box) в BP\_Floor, на пустому місці викликаємо Get Game Mode з нього викликаємо Cast to BP\_RunnerGameMode, підключаємо його Exec пін до OnComponentBeginOverlap. З As BP Runner Game Mode викликаємо ExtendFloor функцію.

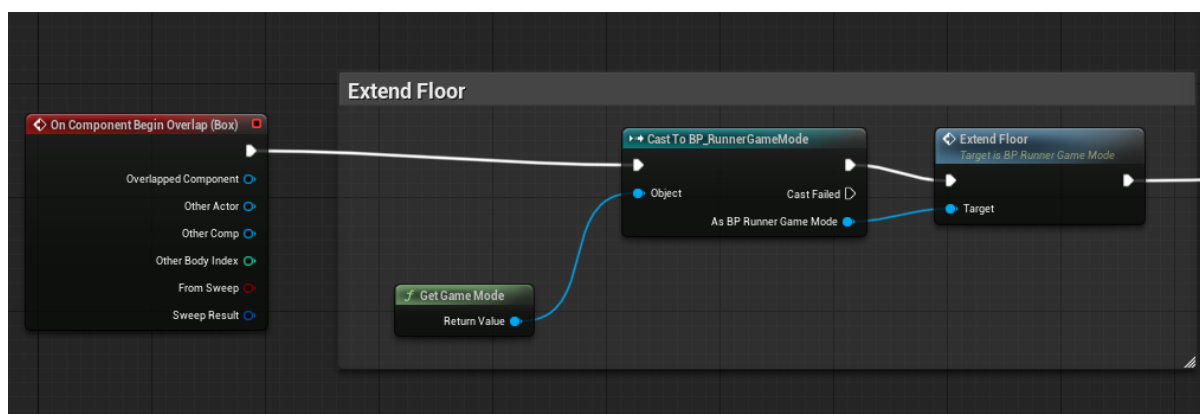


рис. 3.2.2.13.

Для підвищення швидкості руху з часом, з Exec піну Extend Floor отримуємо персонажа через Get

Player Character→Cast to BP\_RunnerCharacter. З As BP\_RunnerCharacter отримуємо Character Movement з нього отримуємо MaxWalkSpeed, додаємо Multiplication нод, в пін А з'єднуємо MaxWalkSpeed, в В ставимо 1.01. З Return value викликаємо Clamp (float) нод, в min 700, max 1500.

З Exec піну Cast to BP\_RunnerCharacter викликаємо Set MaxWalkSpeed. В значення змінної під'єднуємо Result Value з Clamp, в Target з'єднуємо отриманий Character Movement.

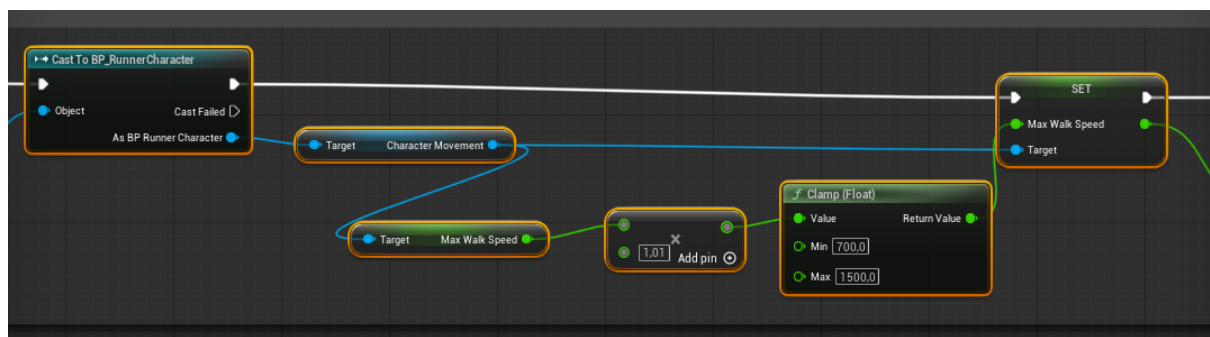


рис. 3.2.2.14.

Шлях з'являється, і з кожним пройденим блоком швидкість підвищується, тепер додамо перешкоди.

### Додавання перешкод

В BP\_Floor створимо функцію Setup Spawnpoints, щоб додати і налаштувати точки на яких будуть з'являться перешкоди.

З Exec піну додаємо For Each Loop, в пін Array під'єднуємо Lane Position змінну. З Loop Body додаємо For Loop, First Index – 0, Last Index – 1.

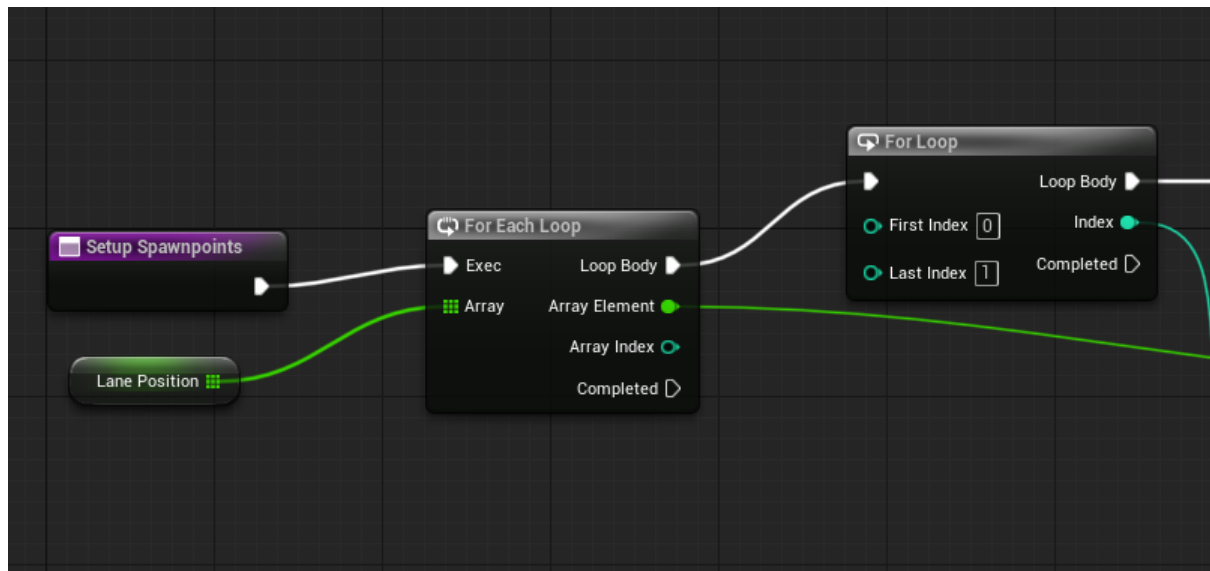


рис. 3.2.2.15.

З Loop Body додаємо Add Arrow Component. З Index додаємо Multiplication, пін В змінюємо на float (натиснути правою кнопкою миші → to float), значення 650.0 – щоб додати відстань між об'єктами і вони не спавнились один в одному. З Return Value отримуємо Make Vector, в пін Y під'єднуємо Array Element з For Each Loop, Z – 0.0. З Return Value отримуємо Make Transform, його Return Value з'єднуємо з Relative Transform ноду Add Arrow Component.

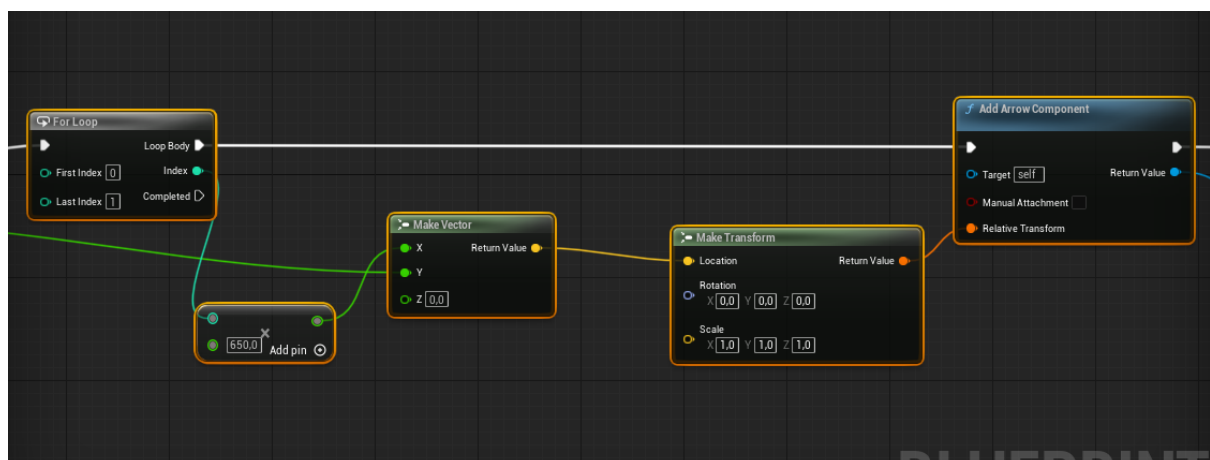


рис. 3.2.2.16.

Створюємо змінну-масив типу Transform SpawnPoints, з Exec піну викликаємо Add для масиву, в масив під'єднуємо змінну. З Return Value Add Arrow Component отримуємо Get World Transform, його Return Value під'єднуємо до New Item піну в Add ноді.

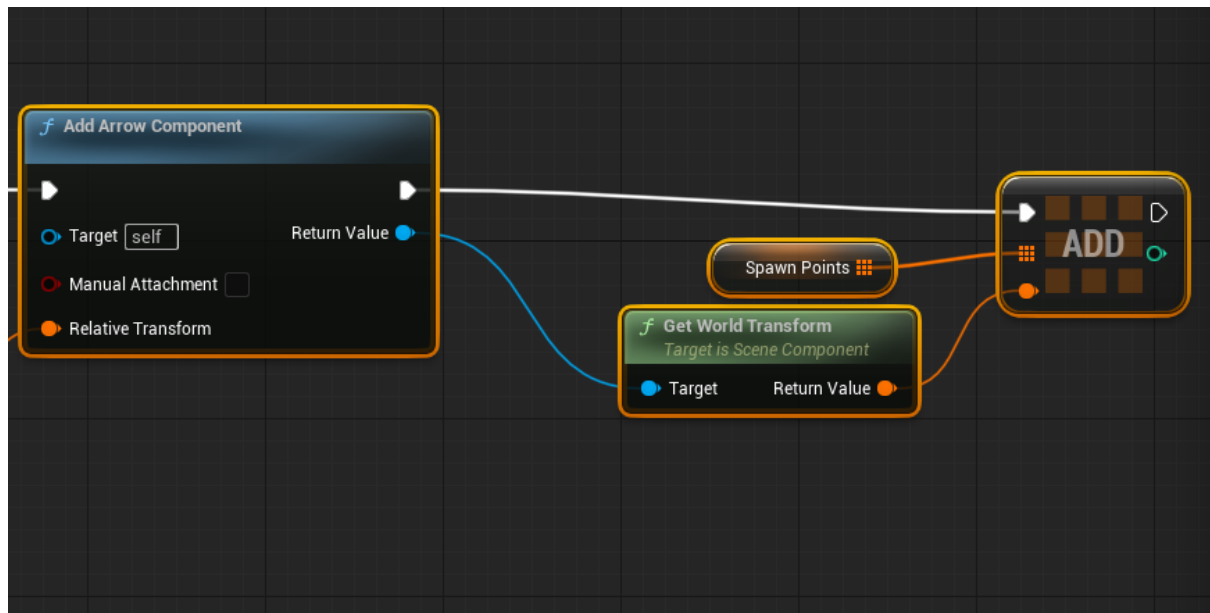


рис. 3.2.2.17.

Створюємо нову функцію Spawn Obstacles. В вкладці Local Variables створюємо змінну-масив Random Spawn Points типу transform. З exec піну викликаємо Set цієї змінної, в значення передаємо SpawnPoints. З піну повернення змінної під exec піном додаємо нод Shuffle, щоб перемішати точки спавну, з exec піну додаємо For Loop, First Index – 0, Last Index –додаємо нод Random integer in range 0-1.

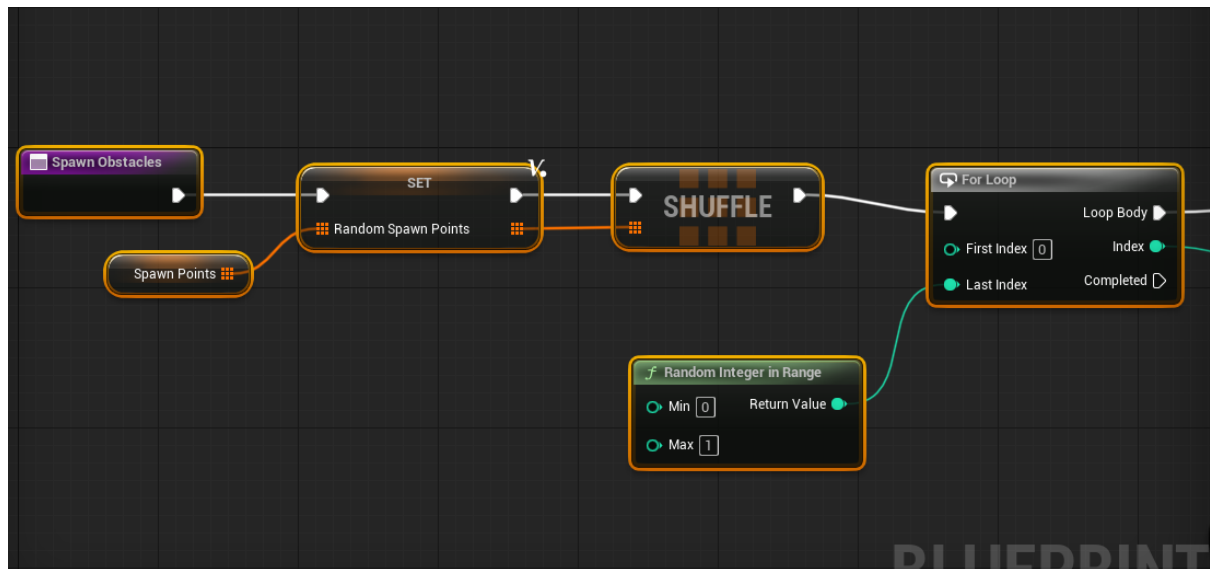


рис. 3.2.2.18.

З Loop Body додаємо Switch On Int, додаємо 3 піни 0-3. В Selection додаємо Random Integer in Range min 0, max 3.

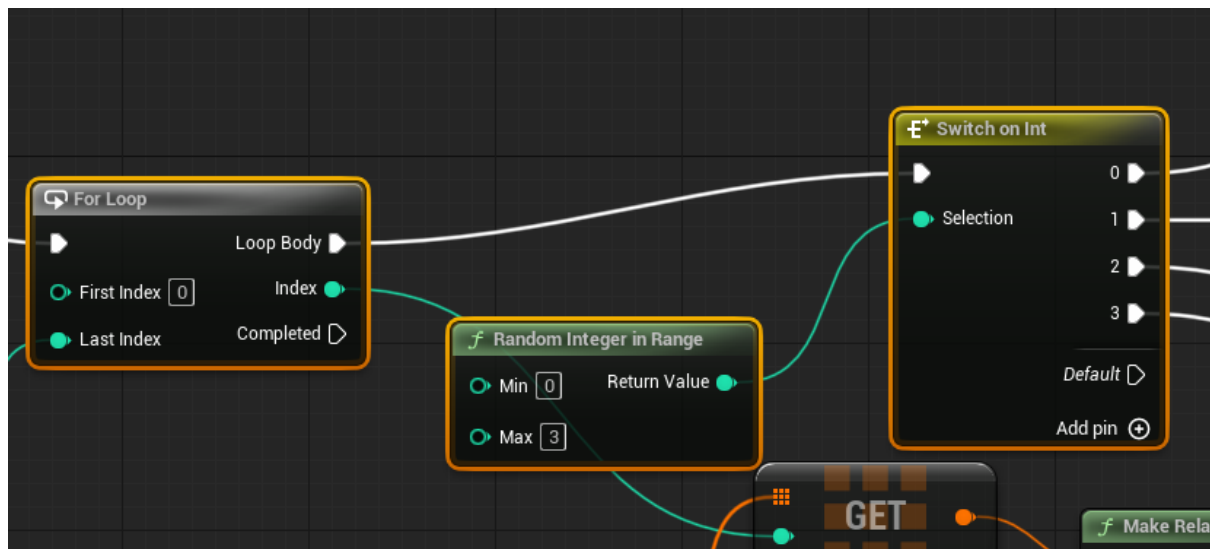


рис. 3.2.2.19.

Отримуємо Get Random Spawn Points, з нього додаємо Get для масиву, в пін індексу під'єднуємо Index з For Loop. З Return Value додаємо Make Relative Transform. В пін Relative to додаємо Get Actor Transform нод.

З 0 та 1 піну Switch додаємо Add Child Actor Component в relative transform під'єднуємо Return Value з Make Relative Transform.

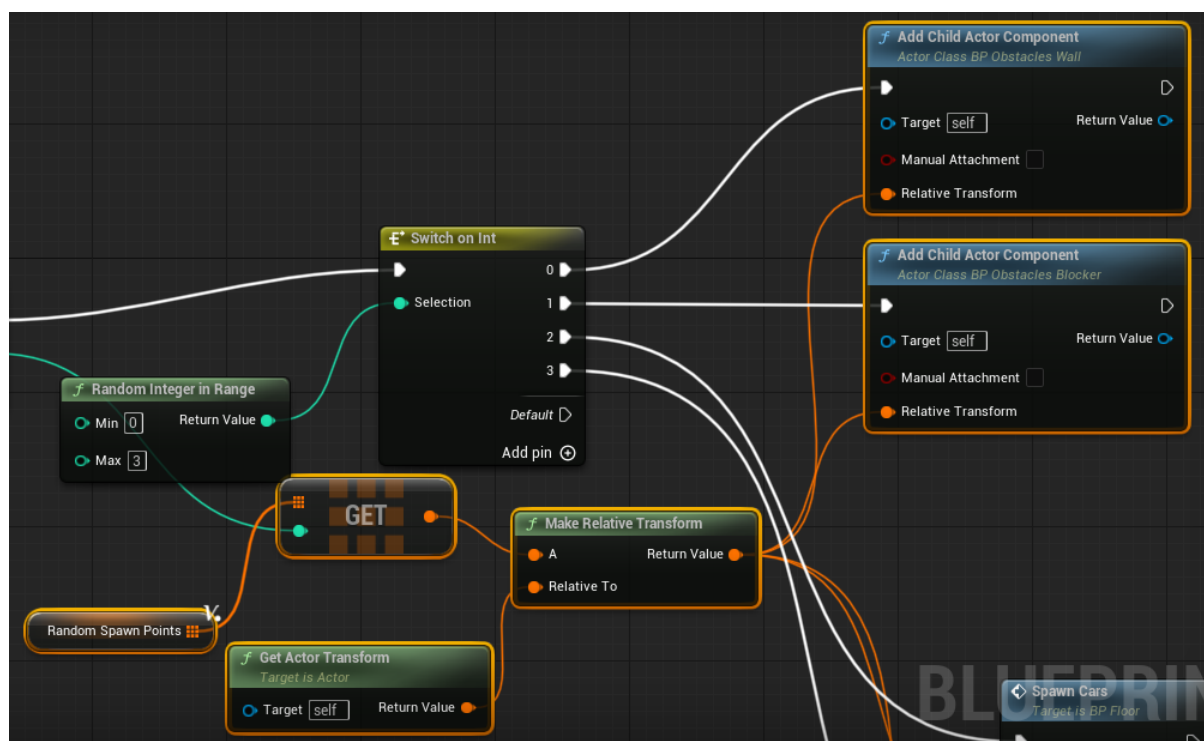


рис. 3.2.2.20.

Для перешкод треба створити нові блюпрінти.

Створюємо блюпрінт класу Actor BP\_Obstacles, це буде батьківський клас. З нього створюємо 2 child класи, натискаємо правою кнопкою миші → Create Child Blueprint Class. Один BP\_Obstacles\_Blocker, інший BP\_Obstacles\_Wall.

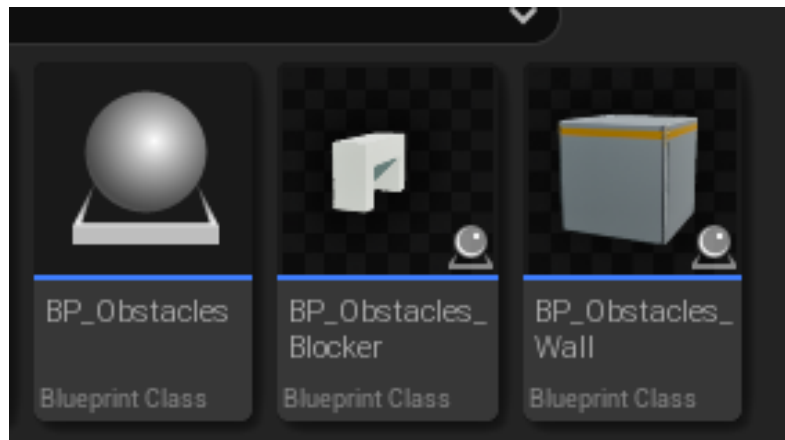


рис.3.2.2.21.

В BP\_Obstacles\_Wall додати StaticMesh та box collision в Details Static Mesh в розділі Static Mesh вибрати будь яку модель, або просто куб, підлаштувати розміри box collision під модель.

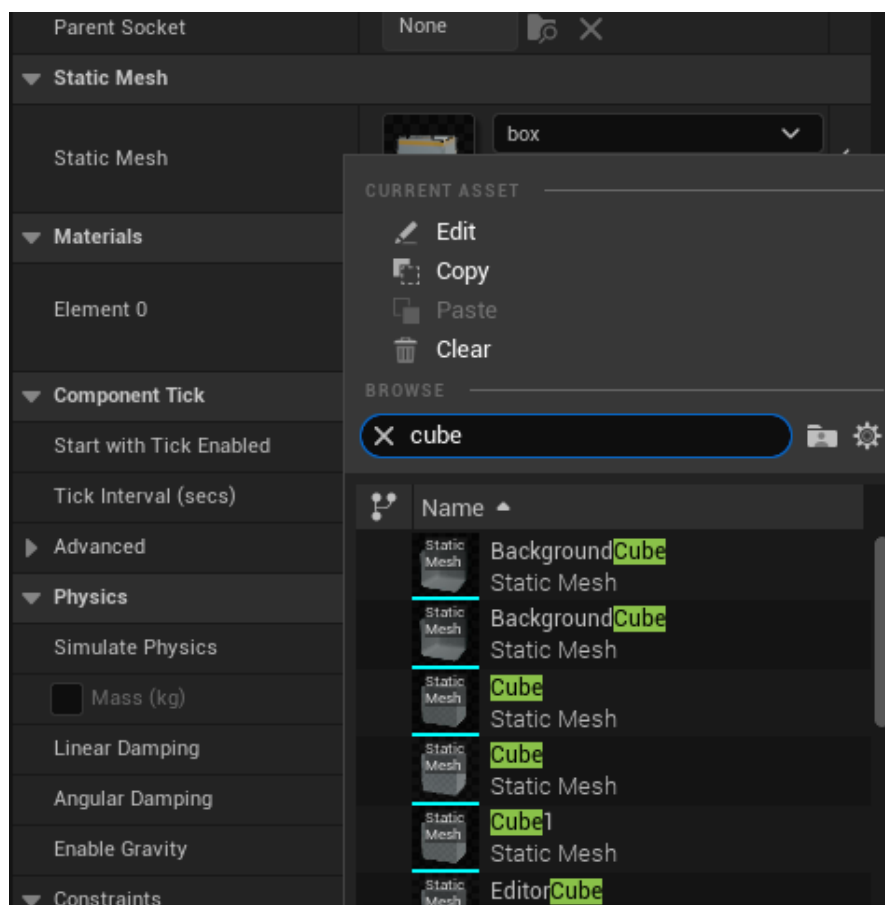


рис. 3.2.2.22.

В BP\_Obstacles\_Blocker додаємо 3 Static mesh Shape\_Cube та Box Collision, змінюємо Transform таким чином

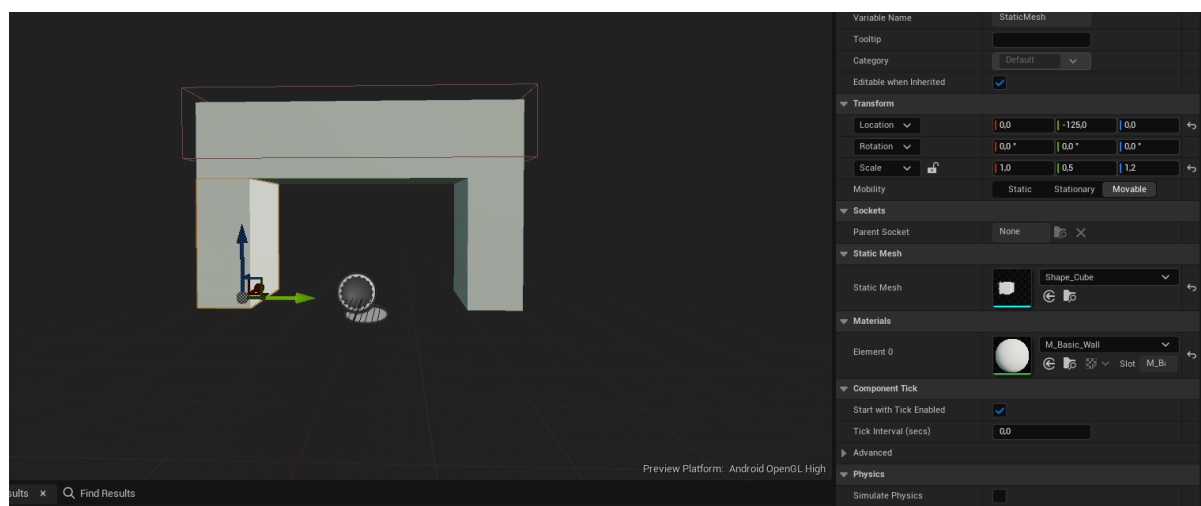


рис. 3.2.2.23.

Тепер коли є блюпрінти для перешкод додаємо їх в BP\_Floor в Details в Add Child Actor Component для 0 піну в Child Actor Class обираємо BP\_Obstacles\_Wall, для 1 обираємо BP\_Obstacles\_Blocker.

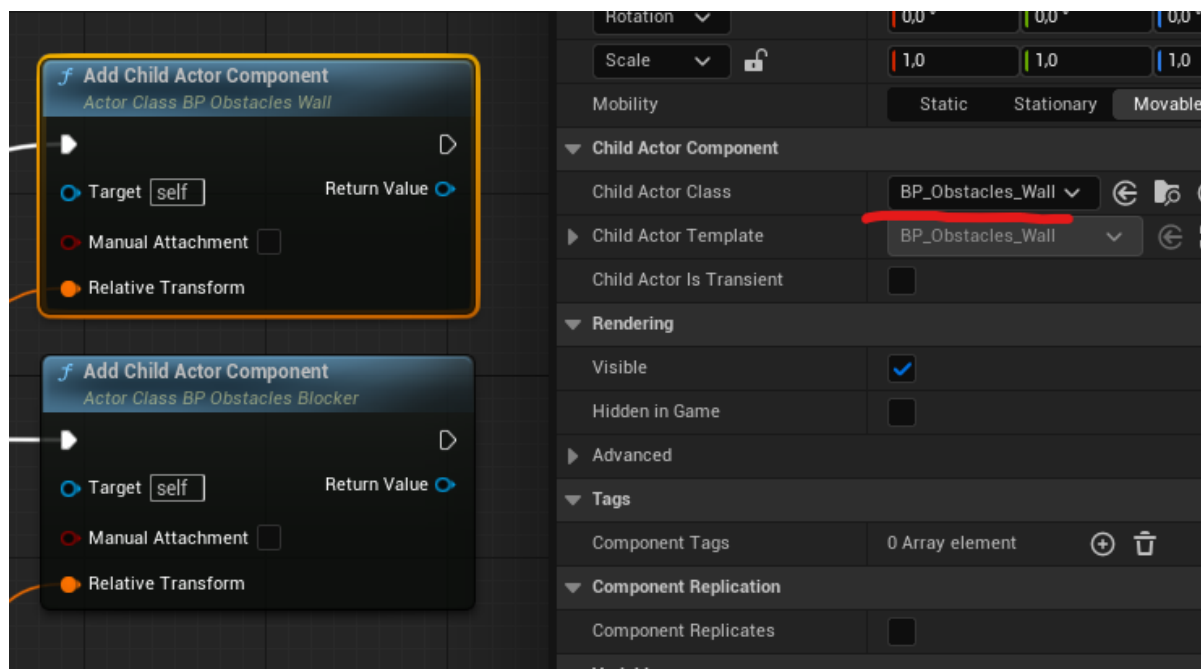


рис. 3.2.2.24.

Тепер в BP\_Floor переходимо в Construction Script і додаємо до ехес піну Setup Spawnpoints та Spawn Obstacles.

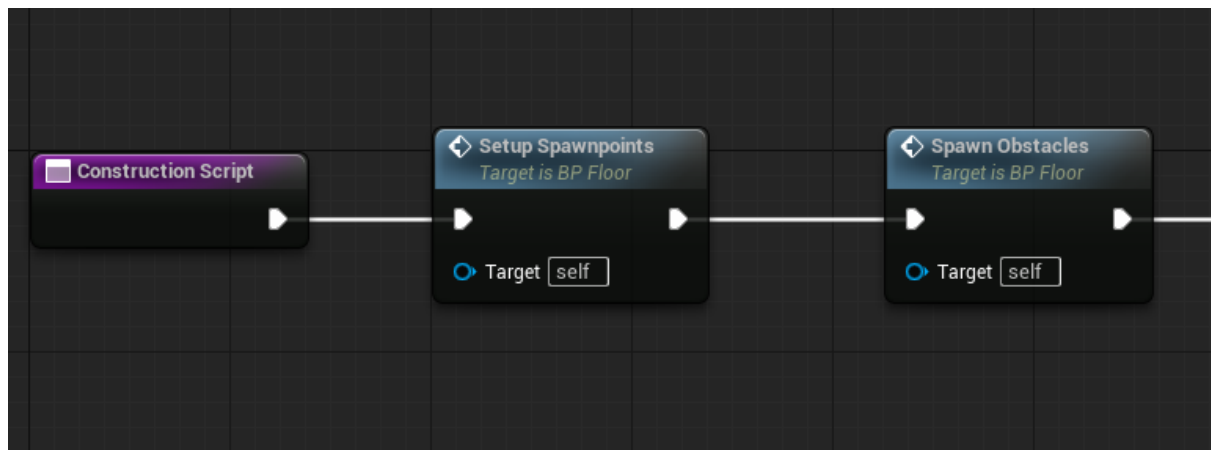


рис. 3.2.2.25.

В Viewport до актора при компіляції будуть з'являтися перешкоди.

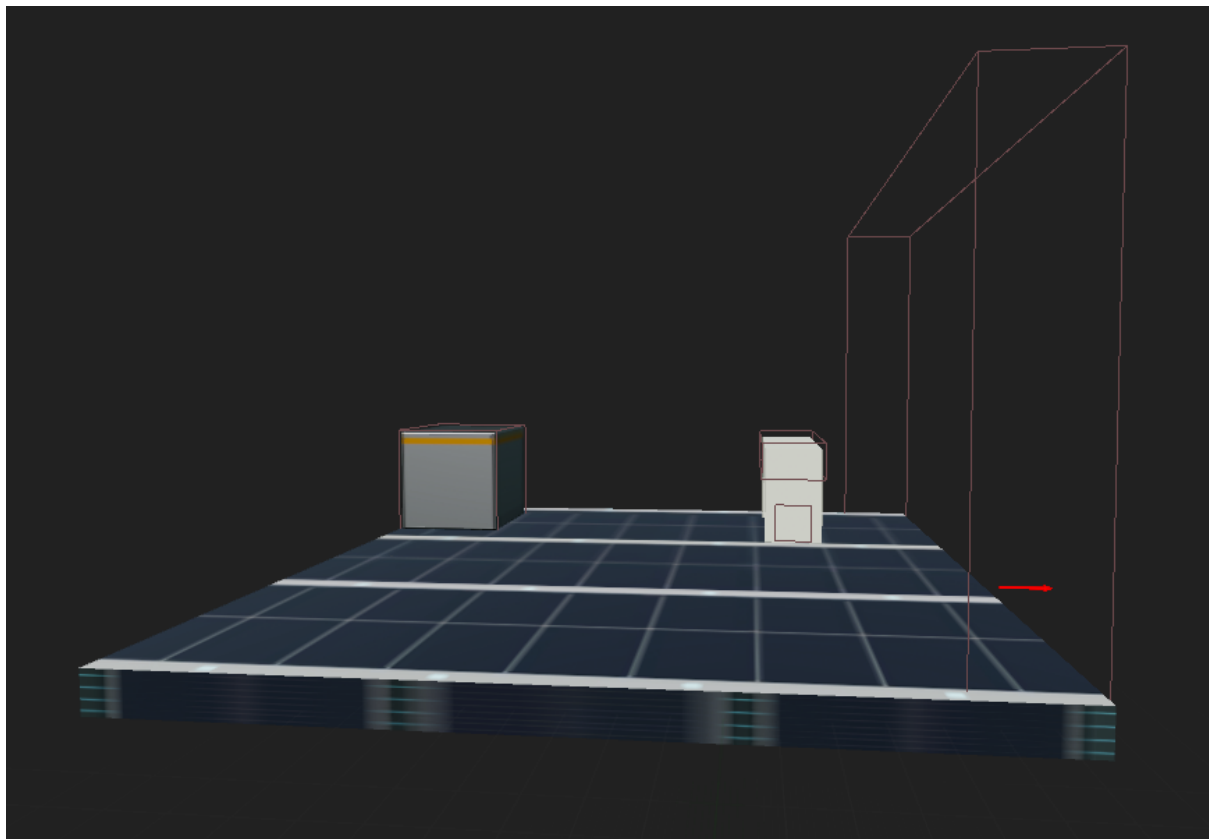


рис. 3.2.2.26.

Створимо окремий блюпрінт де додамо тільки пустий Static Mesh з блюпрінту BP\_Floor назвемо FloorTiles, на локації 0 0 0 ставимо BP\_Floor, перед ним двічі ставимо FloorTiles, Player Start на початку, щоб коли спавнився гравець було трохи пустого шляху. Зберігаємо мапу з назвою Runner\_map.

### 3.2.3. Розробка системи збору ігрової валюти та підсилень

#### Реалізація спавну і підбирання монет

Створюємо блюпрінт Actor BP\_Coin. Додаємо Static Mesh перейменовуємо в Coin, в модель обираємо модель монети, або будь чого іншого, що може репрезентуватиме ігрову валюту. В BP\_Coin (self) додаємо RotatingMovement для обертання.

В Coin в Details обираємо Events→OnComponentBeginOverlap і додаємо, в Eventh Graph з'являється івент.

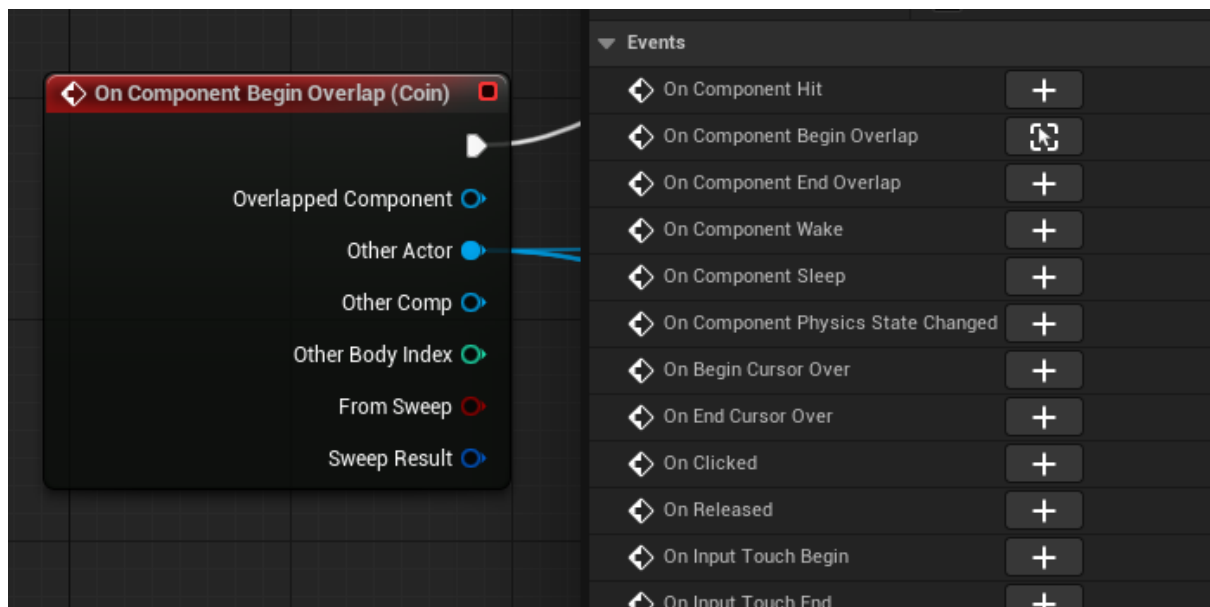


рис. 3.2.3.1.

З Exec пін додаємо Sequence додаємо додатковий пін, і в then 1 з Other Actor додаємо Cast to BP\_Obstacles\_Wall з нього додаємо Destroy Actor, щоб якщо монета заспавиться всередині перешкоди то вона знищилась.

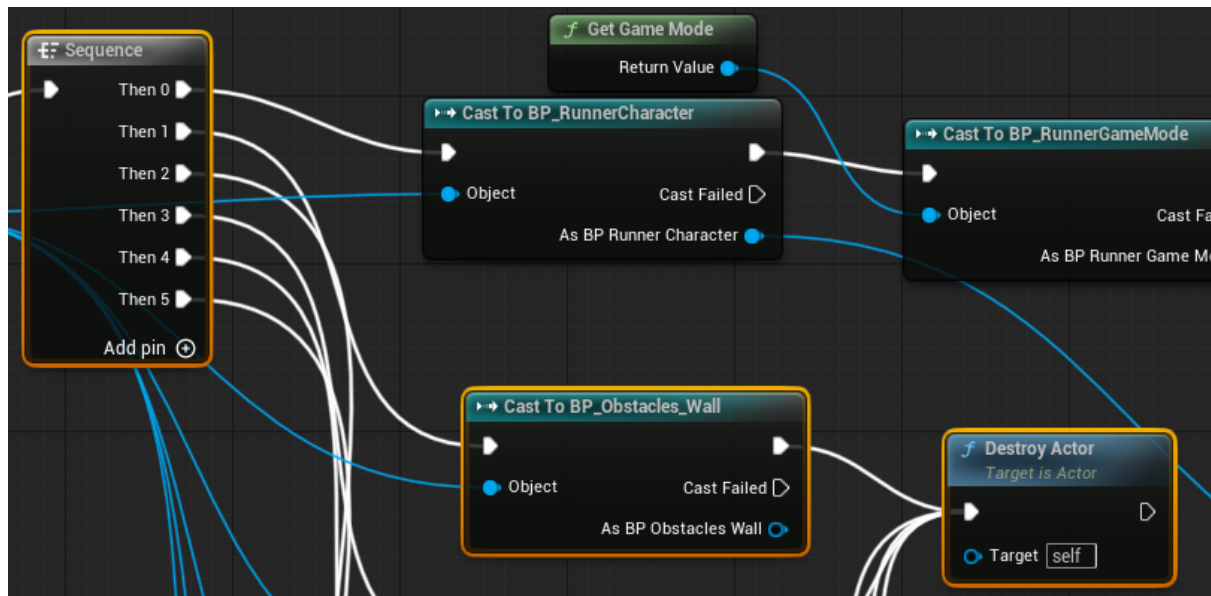


рис. 3.2.3.2.

В then 0 з Other Actor додаємо Cast to BP\_RunnerCharacter, на пусте місце додаємо Get Game Mode, з return value → Cast to BP\_RunnerGameMode, під'єднуємо до Exec піну попереднього ноду.

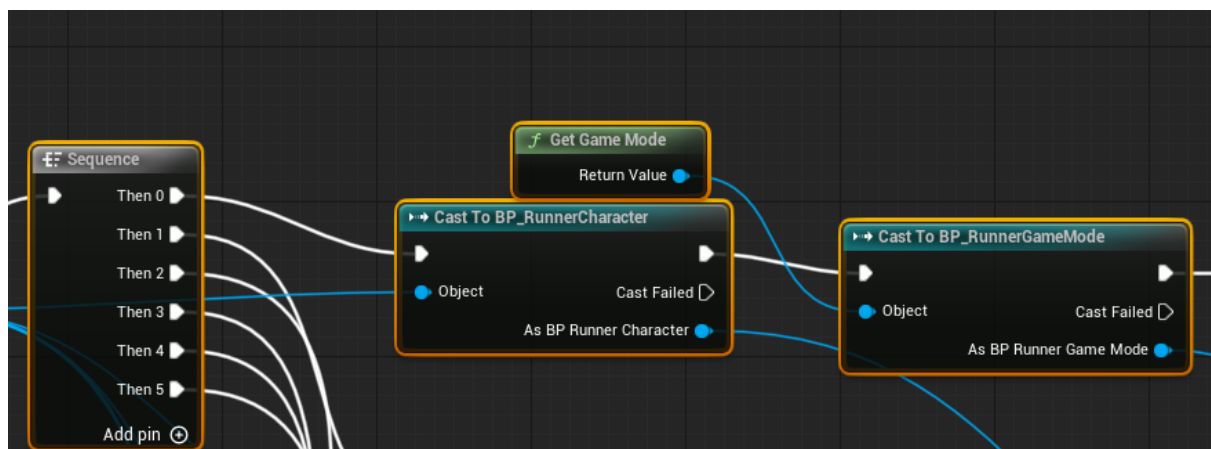


рис. 3.2.3.3.

В BP\_RunnerGameMode створюємо змінну `int Coins` для збереження кількості монет.

В BP\_Coin з As BP Runner Game Mode отримуємо цю змінну, з неї додаємо нод Increment (++), з'єднуємо ехес піни, з другого ехес піну додаємо Play Sound 2D для додавання звуку підбирання валюти, обираємо звук та з Ехес піну викликаємо Destroy Actor нод.

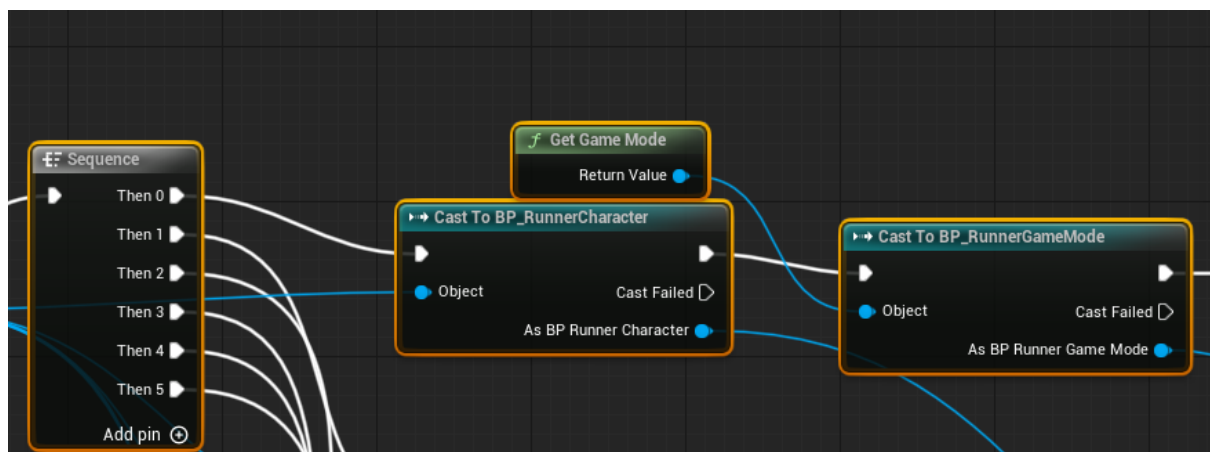


рис. 3.2.3.4.

Щоб зберігати монети і завантажувати їх значення, створюємо блюпрінт з назвою Runner\_SaveGame батьківський клас – SaveGame. В ньому створюємо змінну `CoinValue` типу `int`.

Переходимо до BP\_RunnerCharacter, створюємо функцію UpdateCoins, на пустому місці додаємо Get Game Mode→Cast to BP\_RunnerGameMode, з'єднуємо ехес пін з піном функції. З другого ехес піну касту додаємо нод Does Save Game Exist, в SlotName пін вводимо ім'я змінної створеної в Runner\_SaveGame – CoinValue. З Return value додаємо Branch.

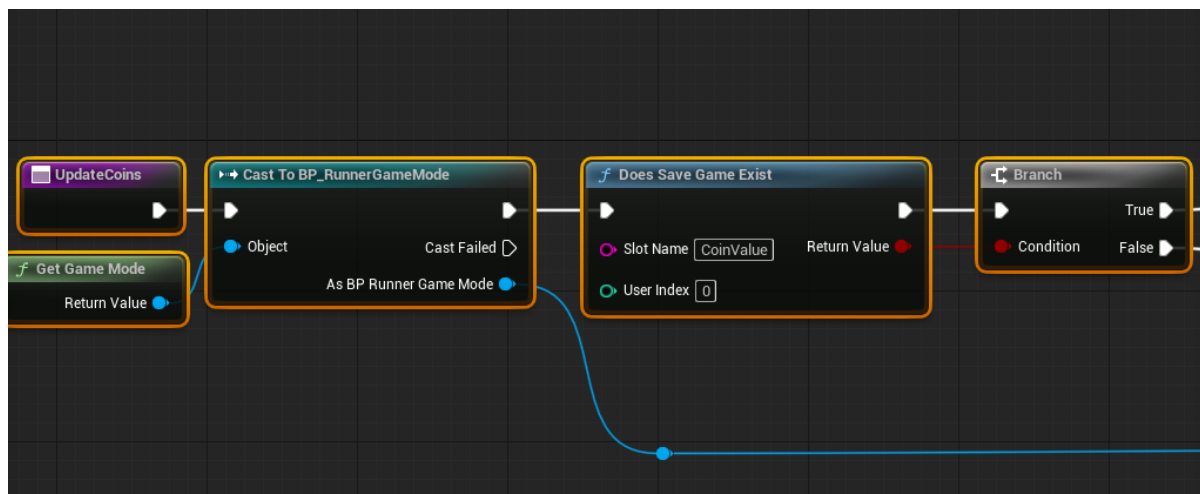


рис. 3.2.3.5.

З True додаємо Load Game from Slot, Slot Name – CoinValue. Створюємо змінну Saver Sub Class класу SaveGame. З ехес піну додаємо Set змінної, з value змінної додаємо Cast to Runner\_SaveGame. З As Runner Save Game додаємо Set CoinValue. З першого ноду Cast to BP\_RunnerCharacter з піну As BP Runner Character отримуємо змінну Coins та з'єднуємо з Coin Value. З ехес піну додаємо Set Game To Slot нод, в Save Game Object отримуємо Get Saver Sub Class і з'єднуємо.

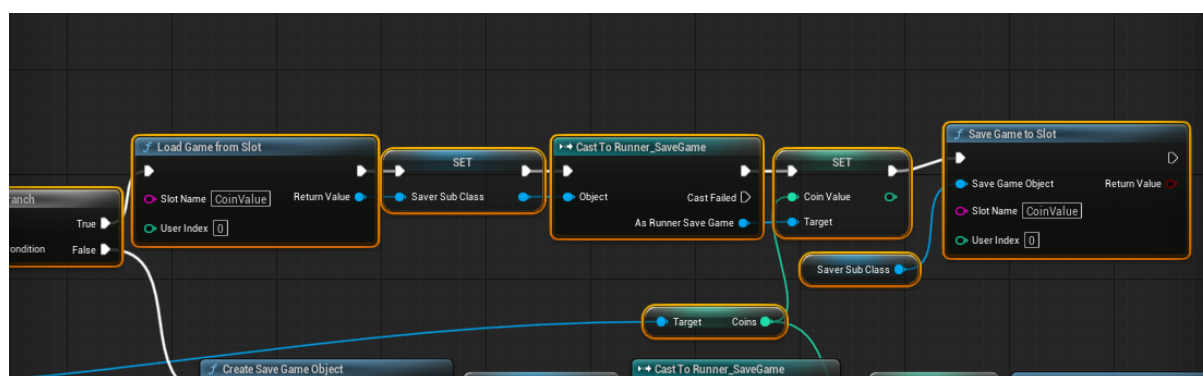


рис. 3.2.3.6.

З False додаємо нод Create Save Game Object обираємо створений блюпрінт. З Return value додаємо Set Saver Sub Class з піну повернення змінної додаємо Cast To Runner Save Game, з As Runner Save Game

додаємо Set Coin Value до піну Coin Value під'єднуємо Coins змінну. З Exec піну додаємо нод Save Game To Slot, Save game object з'єднуємо з Save Sub Class.

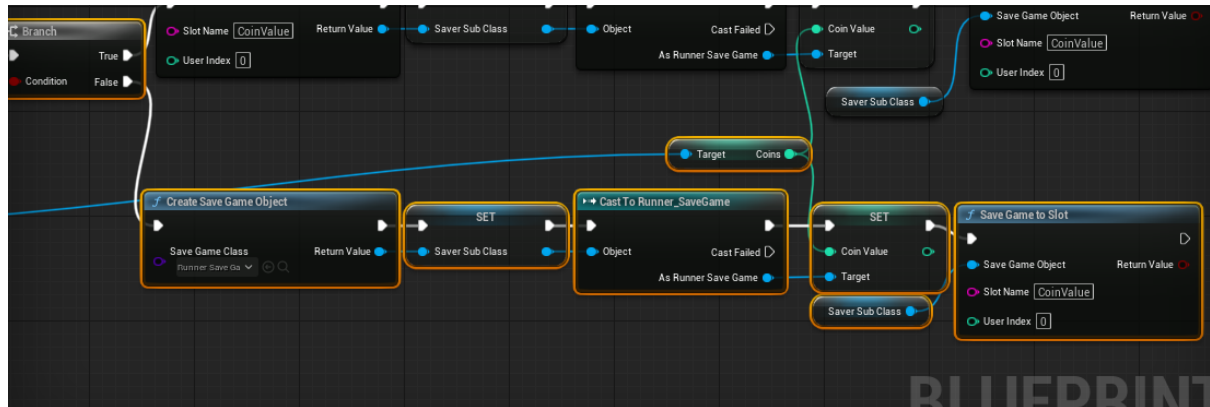


рис. 3.2.3.7.

Тепер в BP\_Coin в On Component Begin Overlap з ноду Cast to BP\_RunnerCharacter піну As BP Runner Character після Destroy Actor додаємо UpdateCoins.

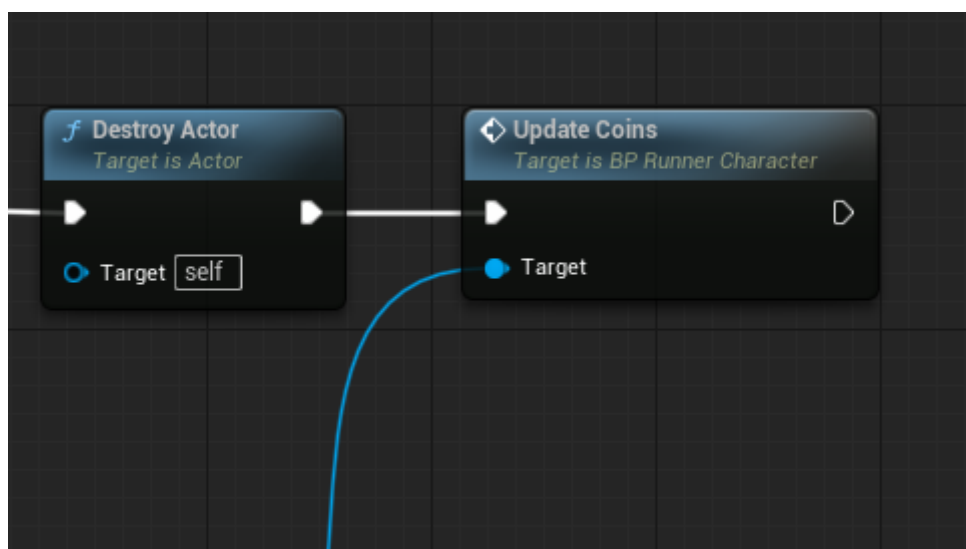


рис. 3.2.3.8.

Для того, щоб вони спавнились разом із перешкодами, переходимо до BP\_Floor. Копіюємо змінну SpawnPoints, називаємо CSpawnPoints.

Копіюємо функцію Setup SpawnPoints називаємо Coin Spawn Points, замінюємо змінні й функції які використовують SpawnPoints на CSpawnPoints. For Loop Last Index ставимо 9, значення multiplication ноду – 75.

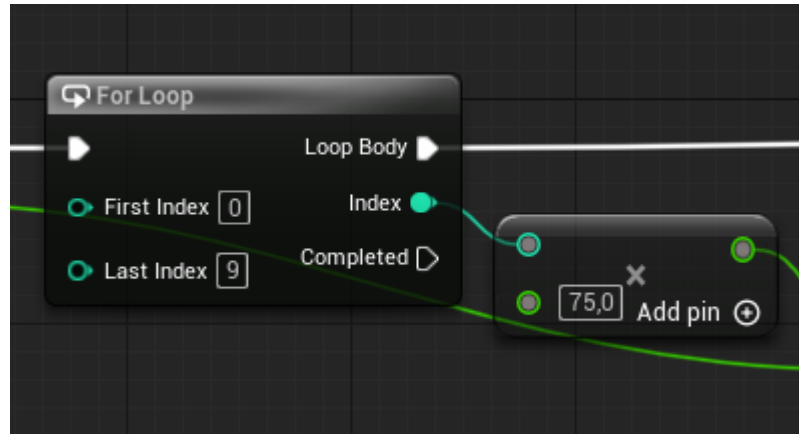


рис. 3.2.3.9.

Копіюємо функцію Spawn Obstacles, називаємо Spawn Coins. В Switch on int видаляємо всі піни окрім 0. З Selection видаляємо random integer і ставимо 0. В For Loop в Last Index де використовується Random Integer in Range ставимо 5-9. Замінюємо використання змінних на CSpawnPoints.

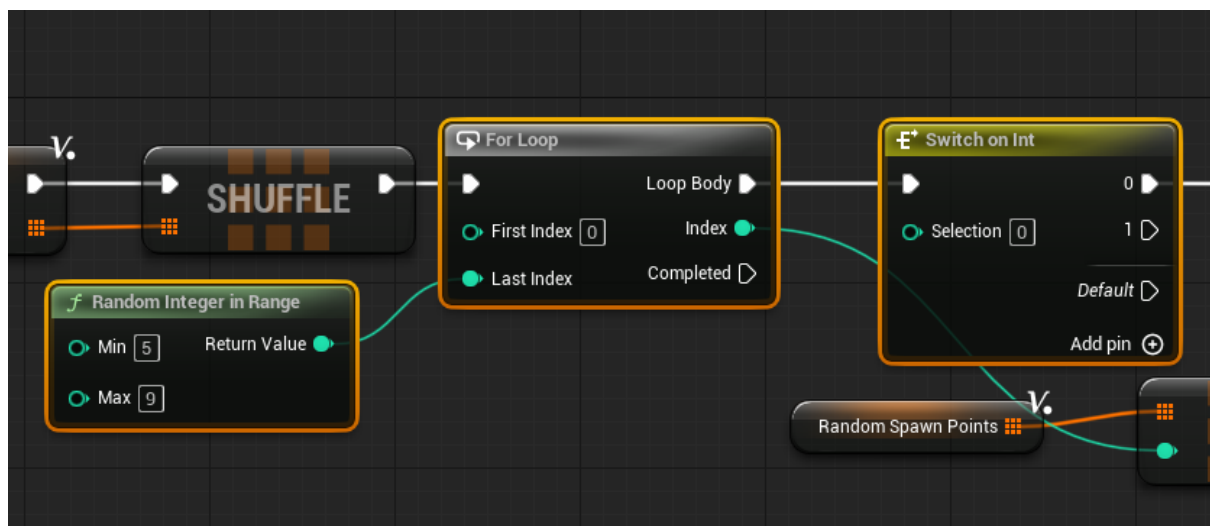


рис. 3.2.3.10.

В Switch on int 0 в Add Child Actor Component обираємо child actor class – BP\_Coin. В Construction Script додати Coin Spawn Points та Spawn Coins.

### Реалізація підсилення магніту

Створюємо блюпрінт актора BP\_Powerup\_Magnet всередині додаємо static mesh в scene root та Rotating Movement в self, в static mesh обираємо модель магніту або будь чого іншого, що репрезентуватиме підсилення.

В Event Graph в івенті ActorBeginOverlap з ехес піну додаємо sequence, додаємо додатковий пін, з other actor → Cast To BP\_Obstacles\_Wall → Destroy Actor, щоб не з'являлось всередині перешкоди.

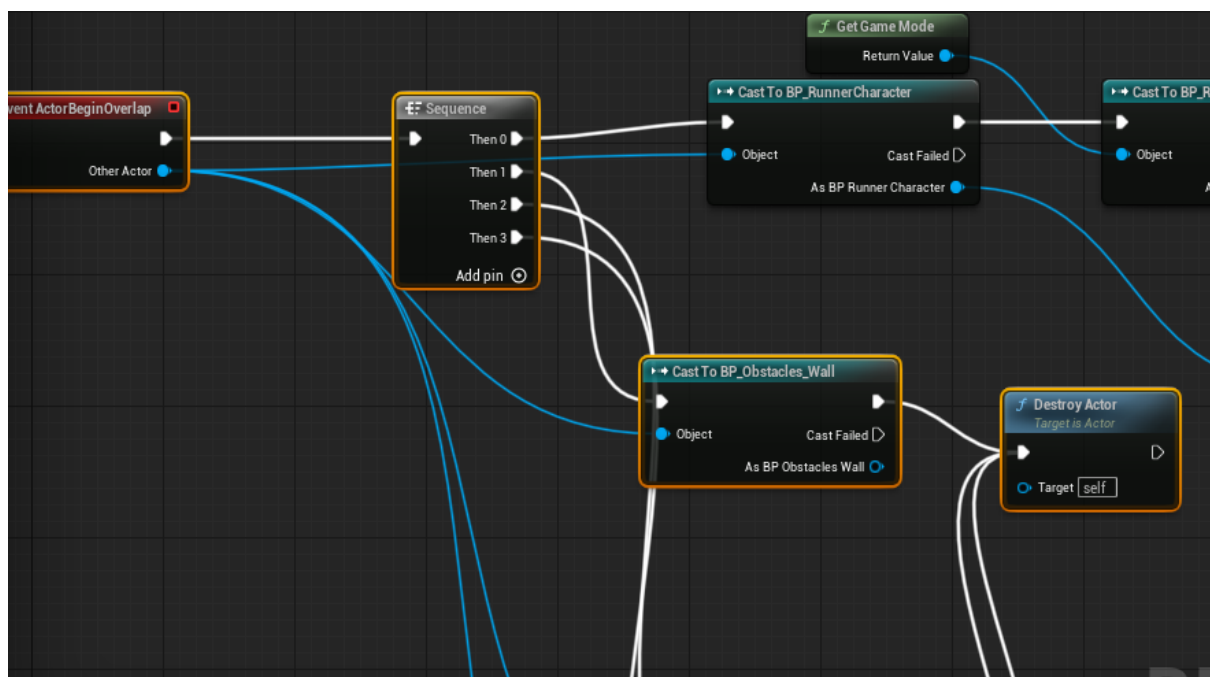


рис. 3.2.3.11.

З other actor → Cast To BP\_RunnerCharacter з'єднуємо ехес пін з then 0. На пустомі місці додаємо Get Game Mode→Cast To BP\_RunnerGameMode, з'єднуємо з

попереднім ехес піном. Далі додаємо Play Sound 2D для звуку при підбиранні, з'єднуємо з піном, та Set Actor Hidden In Game з New Hidden true.

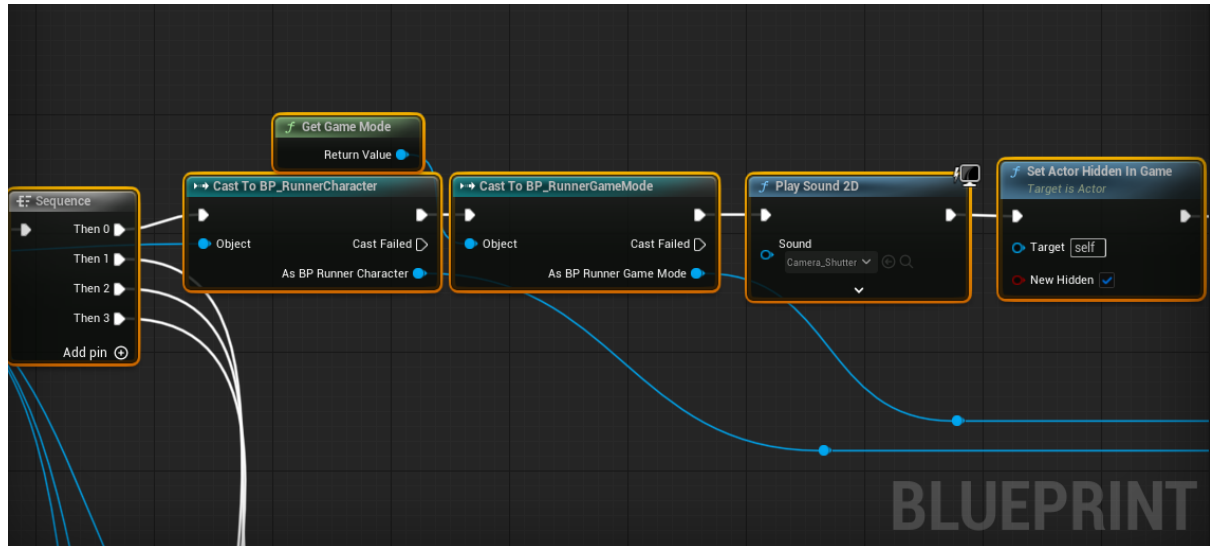


рис. 3.2.3.12.

В BP\_RunnerCharacter створюємо змінну типу boolean IsMagnetActive. В BP\_Powerup\_Magnet додаємо Branch в Condition з Cast to BP\_RunnerCharacter отримуємо IsMagnetActive, з'єднуємо з попереднім ехес піном. З False додаємо Set IsMagnetActive ставимо змінну на true. В BP\_RunnerGameMode створюємо змінну типу float MagnetTimer. З Set IsMagnetActive з Cast to BP\_RunnerGameMode отримуємо Set MagnetTimer значення змінної змінюємо на 6.

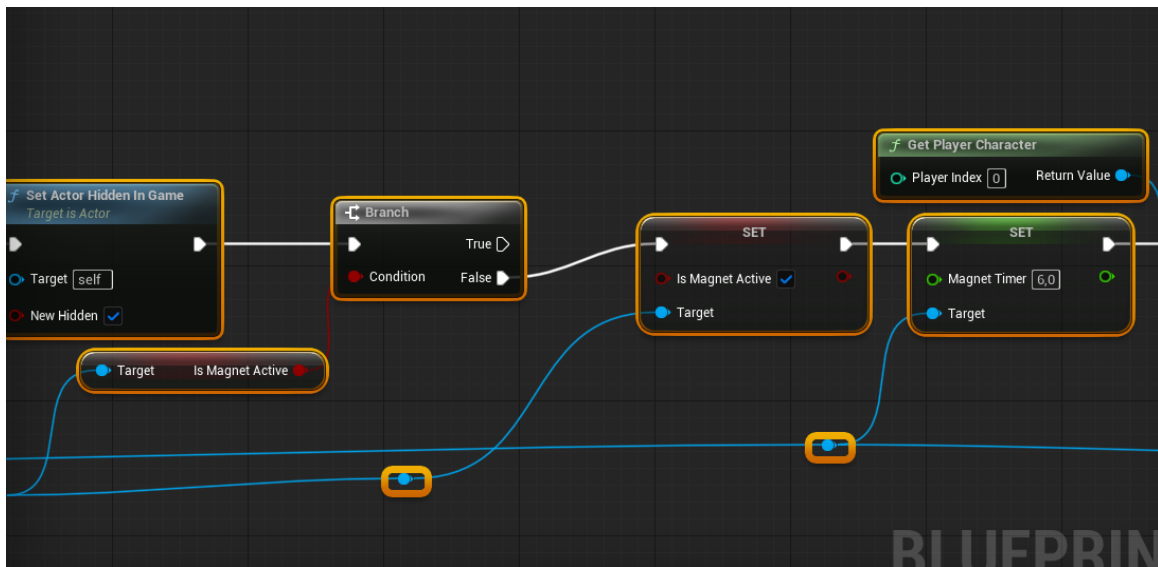


рис. 3.2.3.13.

З ехес піну додаємо Sequence, з then 0 додаємо For Loop with delay, Last index – 5, duration – 1. З RunnerGameMode отримуємо Magnet Timer → Decrement float нод, під'єднуємо до Loop Body. Додаємо Set Magnet Timer, з'єднуємо з ехес піном декременту, з result піну декременту додаємо нод Max пін B – 0, return value з'єднуємо з значенням змінної на Set Magnet Timer.

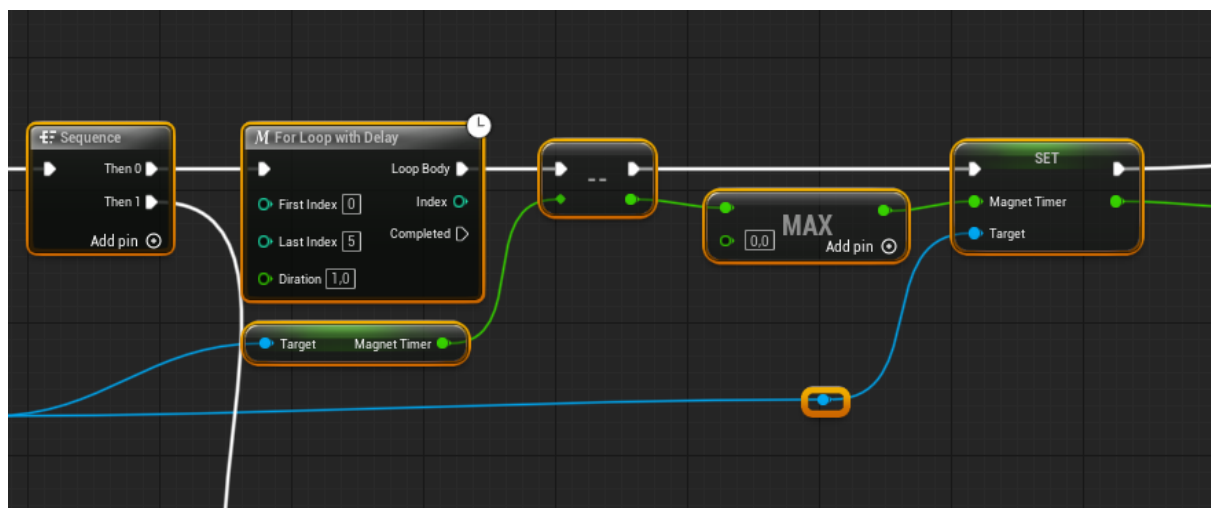


рис. 3.2.3.14.

До Sequence додаємо ще один пін then 1, з нього додаємо Delay duration 5.2. На пустому місці додаємо Get Player Character → Cast To BP\_RunnerCharacter, з'єднуємо з ехес піном Delay. Додаємо Set Is Magnet Active з As BP Runner Character та з ехес піну викликаємо Destroy Actor.

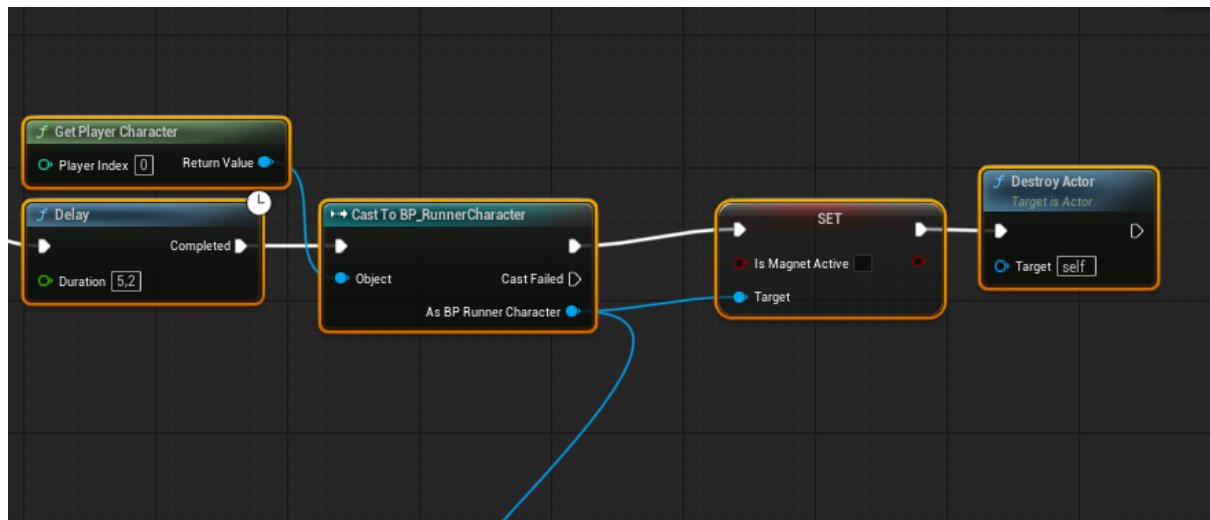


рис. 3.2.3.15.

В BP\_Floor копіюємо змінну SpawnPoints називаємо PSpawnPoints. Копіюємо функцію Setup Spawn Points називаємо PowerUpsSpawnPoints. В ній замінюємо використання SpawnPoints на PSpawn Points. В For Loop first index – 0, last index – 1, значення multiplication – 250. Створюємо функцію SpawnPowerUps всередині отримуємо  $\text{GetGameMode} \rightarrow \text{Cast to BP\_RunnerGameMode} \rightarrow \text{Branch}$ .

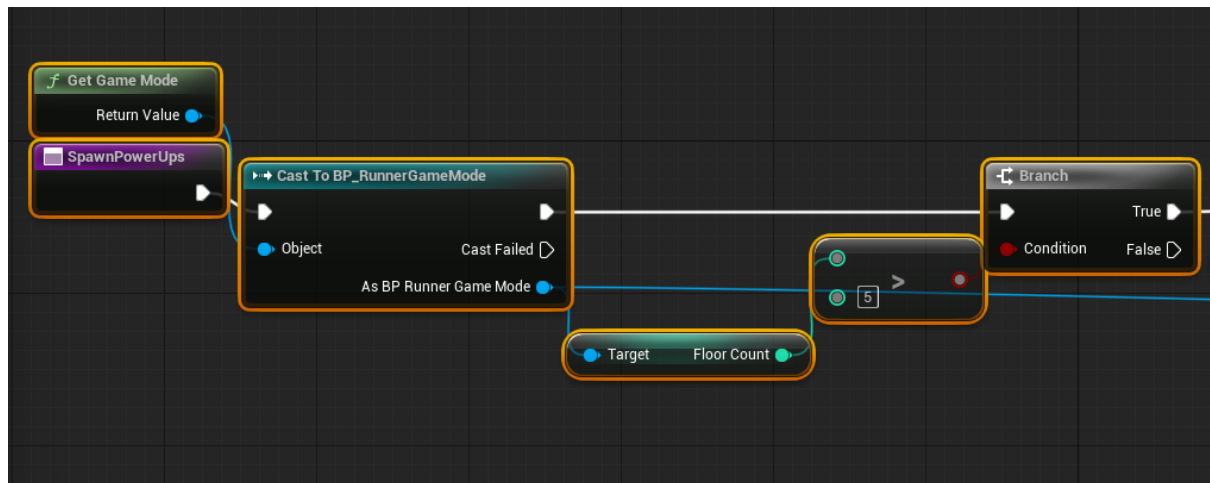


рис. 3.2.3.16.

В BP\_RunnerGameMode створюємо змінну Floor Count типу int. В функції Add Floor в кінці додаємо increment для floor count, та Set Floor count від результату інкременту.

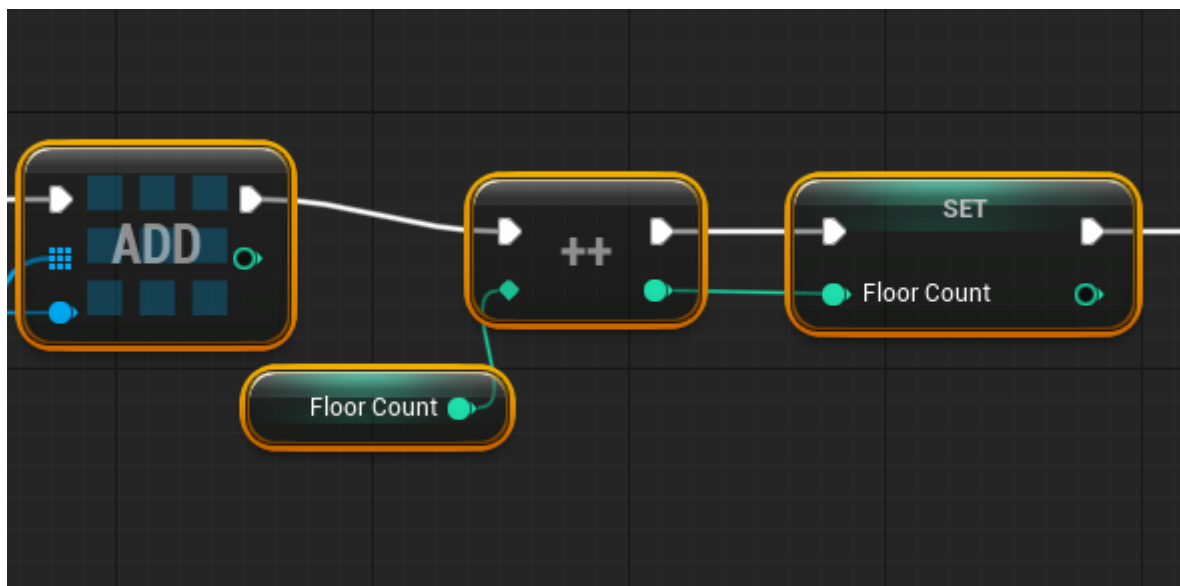


рис. 3.2.3.17.

В BP\_Floor з As Runner Game Mode отримуємо floor count → Greater(>) пін В значення 5, під'єднуємо до branch умови.

Створюємо локальну змінну Random Spawn Points. Викликаємо Set цієї змінної з піну True. В значення змінної передаємо PSpawnPoints. Зі повернення значення змінної додаємо Shaffle, з нього Switch On Int. В Selection під'єднуємо нод Random Integer In Range min – 0, max – 1. З 0 на Switch додаємо Add Child Actor Component, в ньому обираємо клас BP\_Powerup\_Magnet. З As BP Runner Game Mode викликаємо Set Floor Count, значення ставимо на 0 і з'єднуємо з останнім ехес піном.

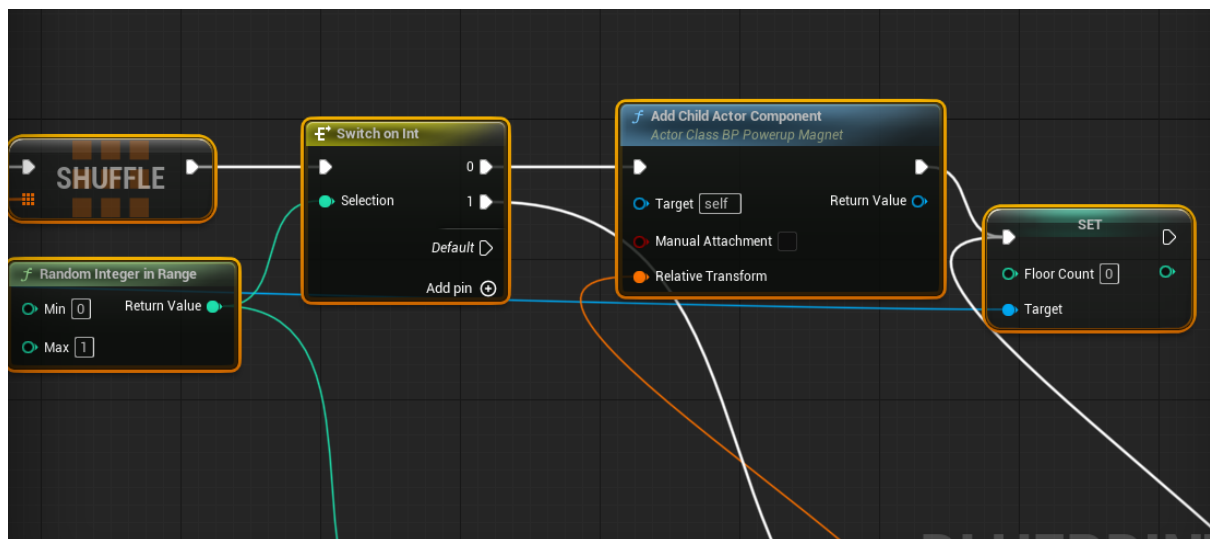


рис. 3.2.3.18.

Додаємо на пустому місці змінну RandomSpawnPoints→Get(copy) для масивів, до індексу під'єднуємо return value з random integer in range. Отримуємо Get Actor Transform, на пустому місці додаємо Make Relative Transform. Return Value з actor transform до Relative to піну, до піну A return Get. Return value ноду під'єднуємо до Relative Transform ноду Add Child Actor.

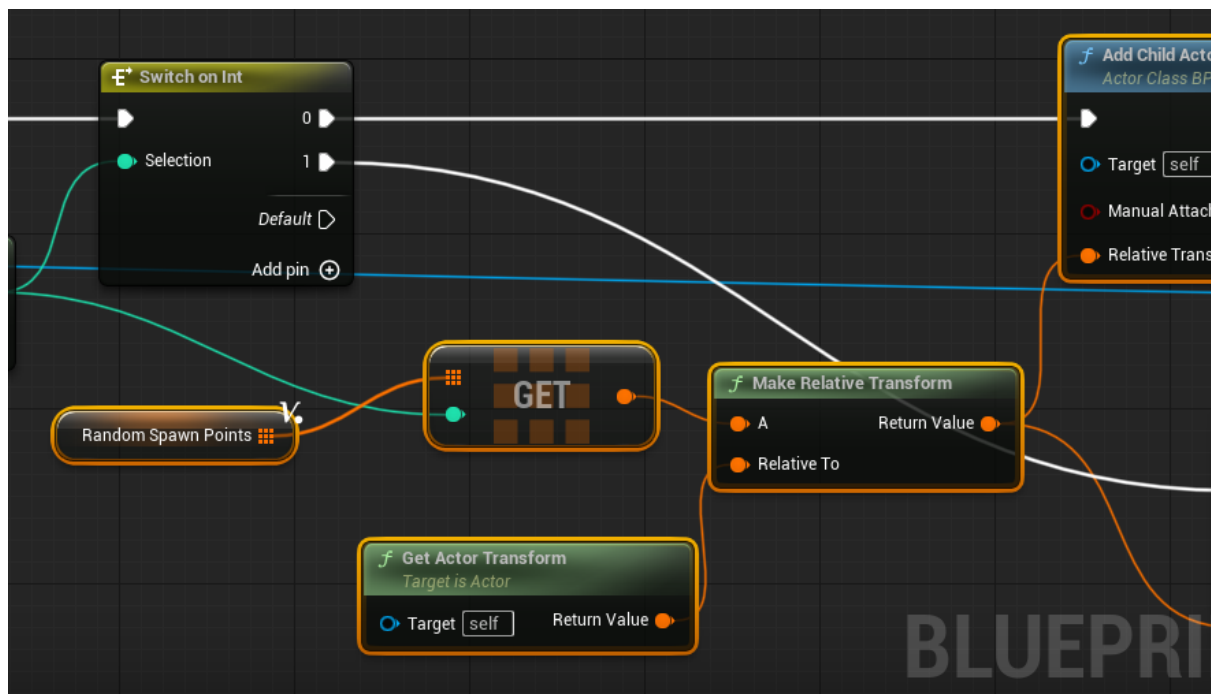


рис. 3.2.3.19.

В BP\_Coin додаємо до Default Scene Root Sphere Collision, ставимо scale на 30 30 30.

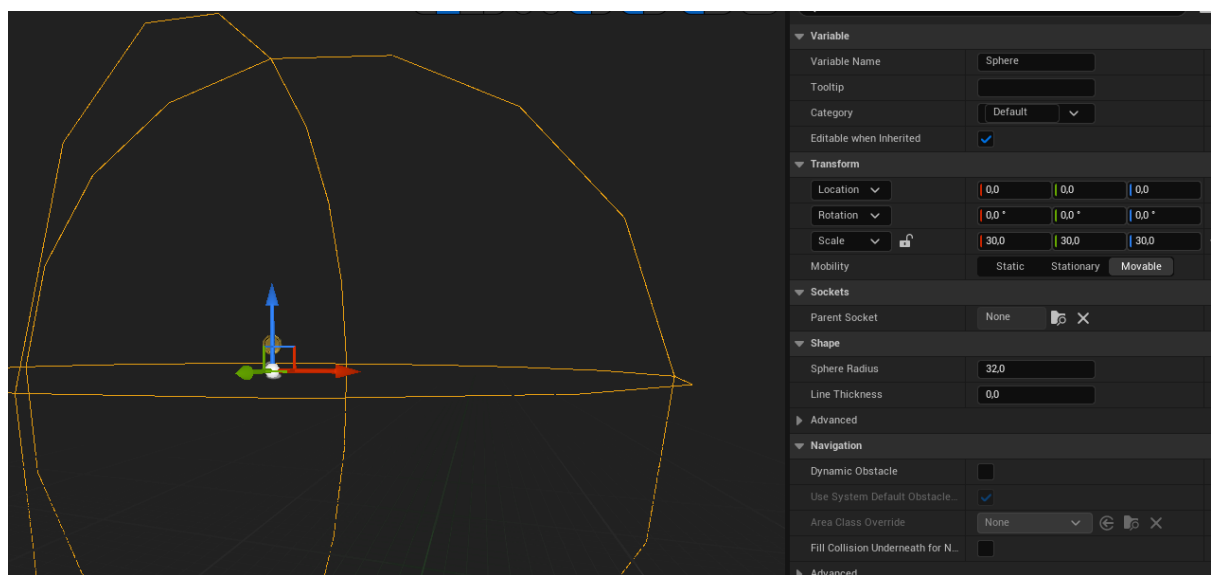


рис. 3.2.3.20.

В sphere collision додаємо event On Component Begin Overlap. З піну Other Actor → Cast To BP\_Runner

Character, As BP Runner Character → Is Magnet Active → Branch, True → Timeline

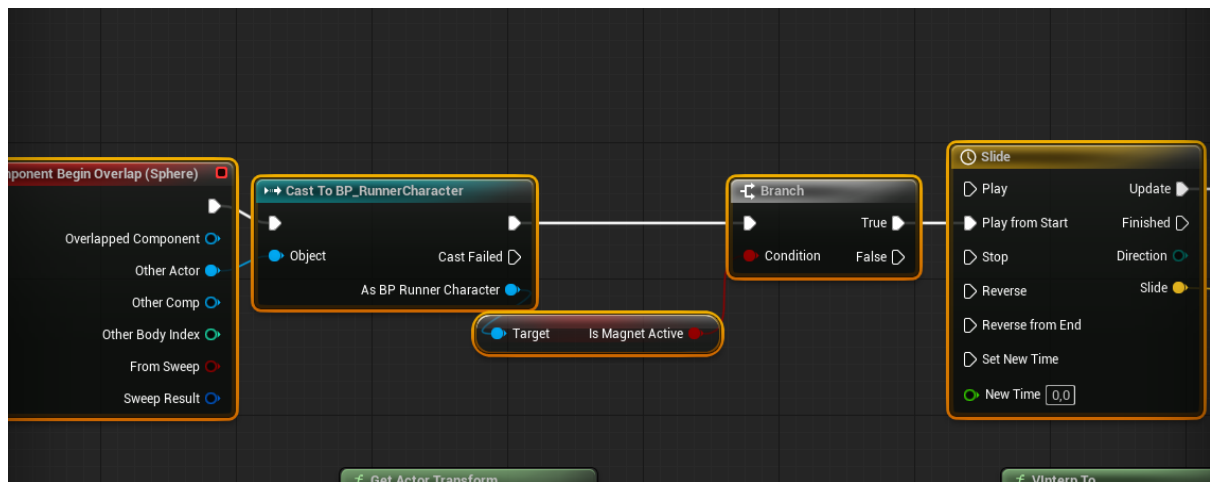


рис. 3.2.3.21.

В Timeline додаємо track float Length ставимо 2.

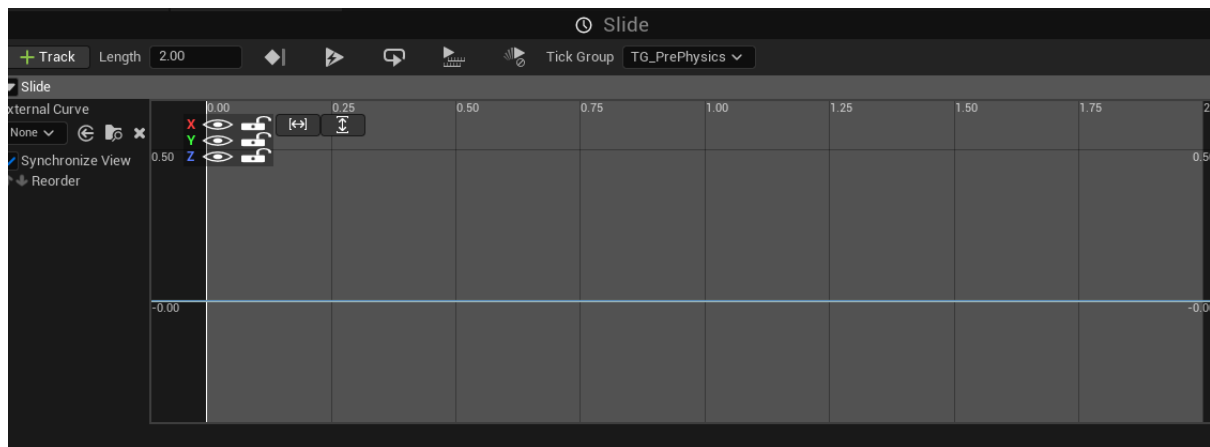


рис. 3.2.3.22.

3 Timeline Update → Set Actor Location. Додаємо Get Actor Transform → Break Transform, Location → Vinterp to. Додаємо Get Player Character → Get Actor Location, Return Value до Target ноду Vinterp To. В Delta time → Get World Delta Seconds. Додаємо нод Addition (+), в пін

A → Slide з Timeline, в пін B → Return value Vinterp To, під'єднуємо до New Location.

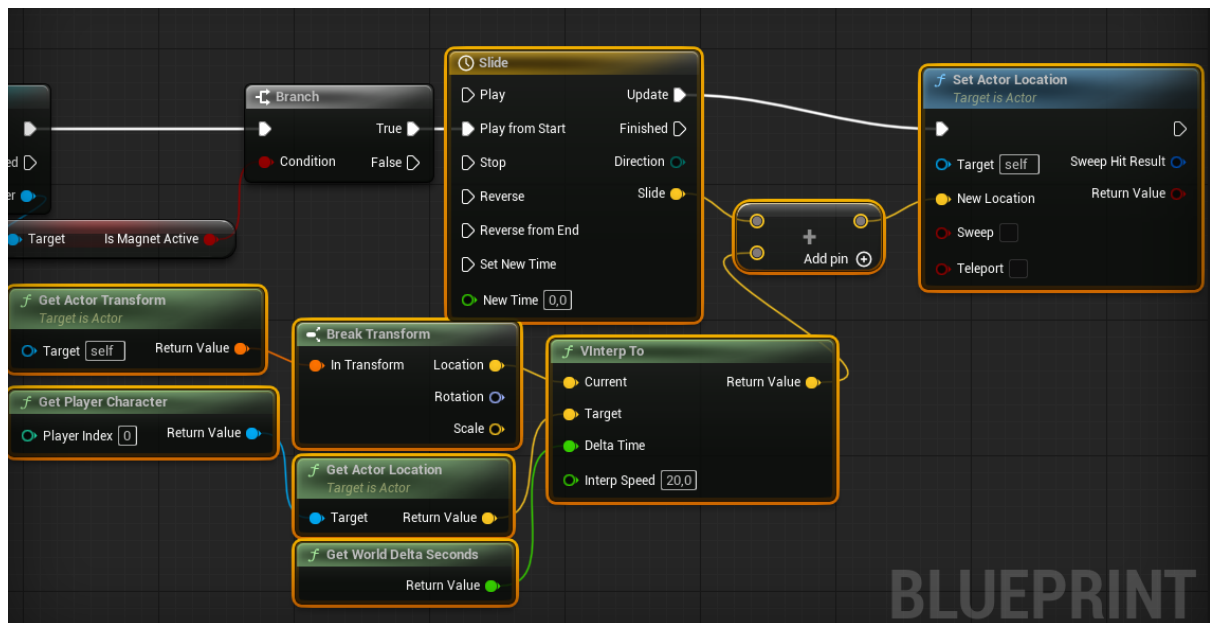


рис. 3.2.3.23.

### 3.3. Розробка інтерфейсу користувача (UI)

#### 3.3.1. Створення віджетів для відображення інформації

##### Основний ігровий віджет

Створюємо Widget Blueprint Gameplay\_UI. Додаємо в дизайнері Canvas Panel, на нього додаємо елементи які мають відображатись. Верхній правий кут – кнопка паузи (Button), під нею – зібрані монети(Image та Text Block), зверху посередині – індикатор часу підсилень (Progress Bar), зверху справа – поточний рахунок (Text Block та Image), знизу під ним – найкращий рахунок (Text Block та Image).

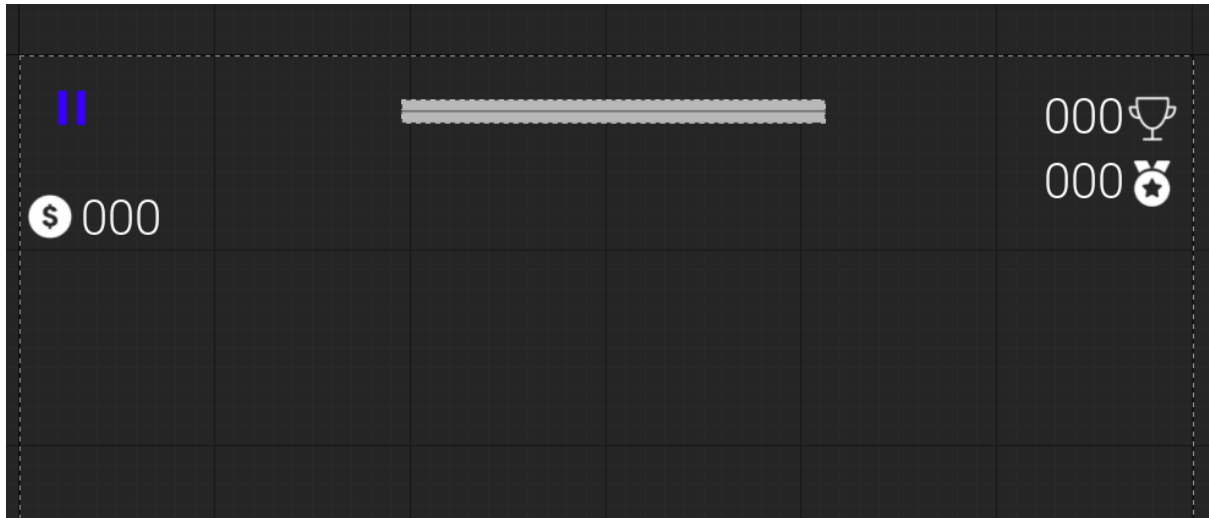


рис. 3.3.1.1.

### Відображення зібраних монет

Обираємо елемент Text Block, який використовується для монет, в Details→Content→Text Block натискаємо на кнопку справа і Create Binding, називаємо Get\_Coins\_Text.

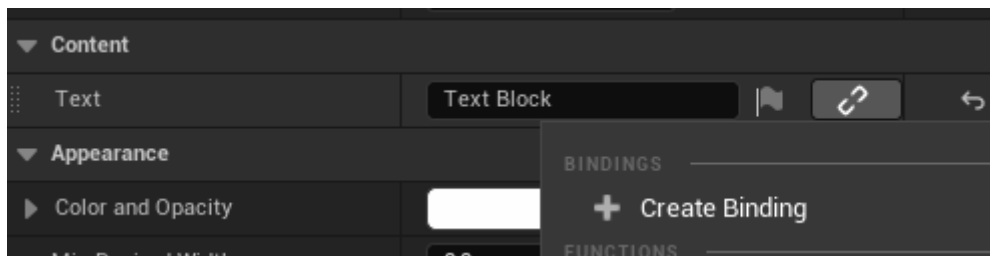


рис. 3.3.1.2.

Всередині з ехес піну додаємо Load Game From Slot в Slot Name вписуємо назву для збережених монет – CoinValue. З return value отримуємо Cast To Runner\_SaveGame→As Runner Save Game→CoinValue→ToText. Створюємо return node на

пустому місці, з'єднаємо з return value ноду ToText та попереднім ехес піном.

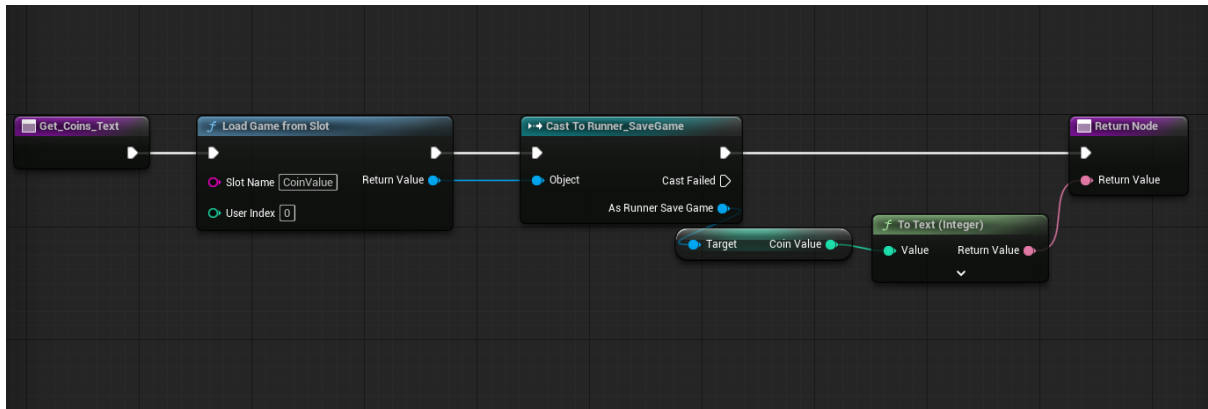


рис. 3.3.1.3.

### Відображення поточного та найкращого рахунку

Для реалізації рахунку, в BP\_RunnerGameMode створимо змінні типу int CurrentPoints та PointsMulti.

Переходимо в BP\_Floor, в On Component Begin Overlap (Box) після Set Max Walk Speed отримуємо BP\_RunnerGameMode через GetGameMode→Cast To BP\_RunnerGameMode. З As BP Runner Game Mode отримуємо CurrentPoints та PointsMulti, з PointsMulti додаємо multiplication (\*) на 1. З return value додаємо addition (+) в В пін під'єднуємо Current Points. З As BP Runner Game Mode додаємо Set Current Points, результат додавання передаємо в значення змінної.

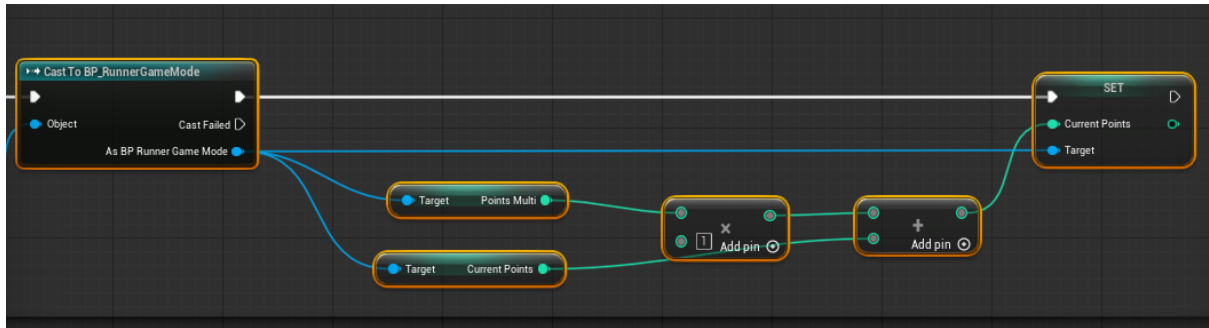


рис. 3.3.1.4.

В Runner\_SaveGame створюємо змінну int з назвою HighScoreValue.

В BP\_RunnerCharacter створюємо функцію SaveHighScore. Всередині отримуємо BP\_RunnerGameMode через GetGameMode→Cast to BP\_RunnerGameMode, з'єднуємо з ехес піном, з нього додаємо Does save game exist нод в Slot Name – HighScoreValue. З Return Value викликаємо Branch.

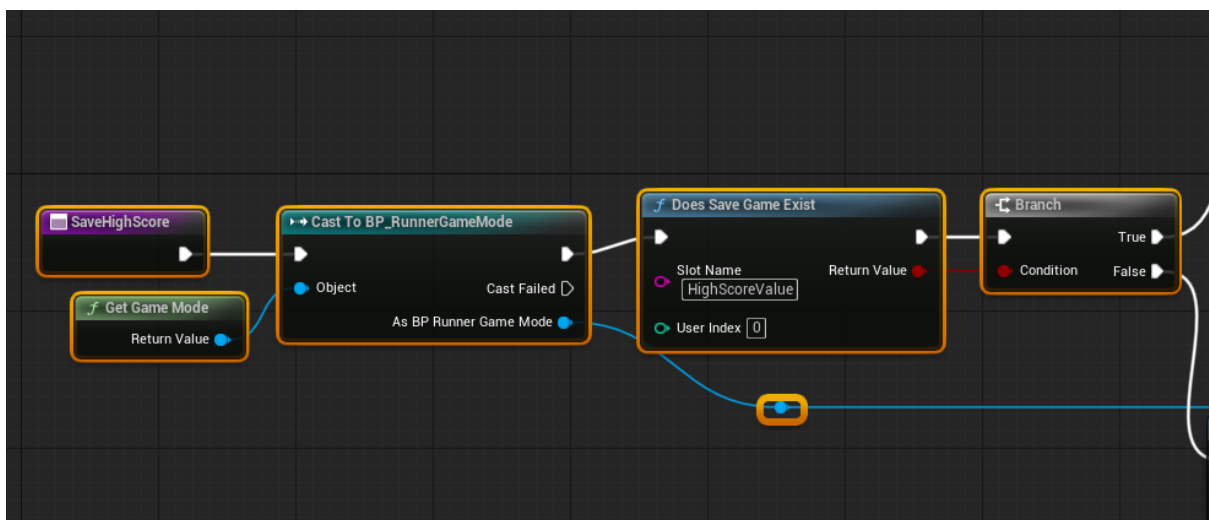


рис. 3.3.1.5.

З True→Load Game From Slot, Slot Name – HighScoreValue, Return Value → Set Saver Sub Class,

повернення змінної → Cast To Runner\_SaveGame, As Runner Save Game → Get High Score Value → Greater(>) в пін В (нижній), в пін А передаємо Current Points з As BP Runner Game Mode. З Return Value додаємо Branch.

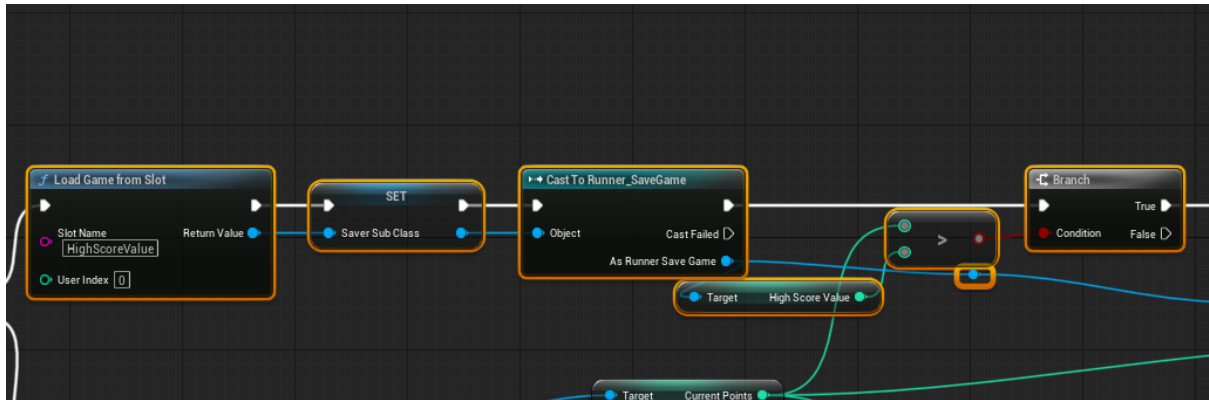


рис. 3.3.1.6.

З True з As BP Runner Save Game додаємо Set High Score Value, в значення змінної – Current Points. З exec пін → Save Game To Slot, Slot Name – HighScoreValue, в Save Game Object передаємо Saver Sub class.

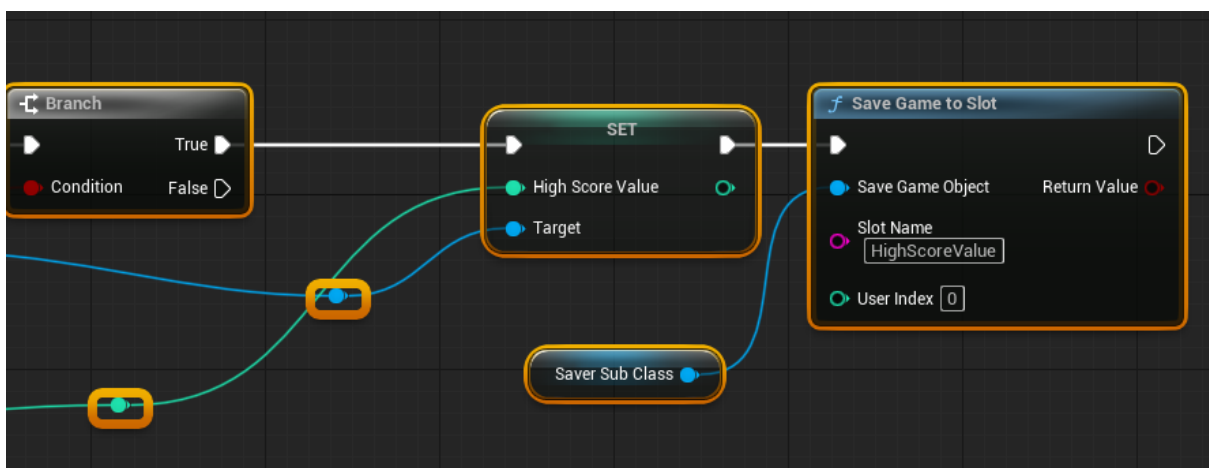


рис. 3.3.1.7.

З першого Branch з False додаємо Create Save Game Object обираємо клас Runner Save Game. З Return Value

→ Set Saver Sub Class → Cast to Runner\_SaveGame, As Runner Save Game → Set High Score Value, в значення змінної з'єднуємо CurrentPoints. З Exec піну викликаємо Save Game To Slot, Slot Name – HighScoreValue, Save Game Object під'єднуємо Saver Sub Class.

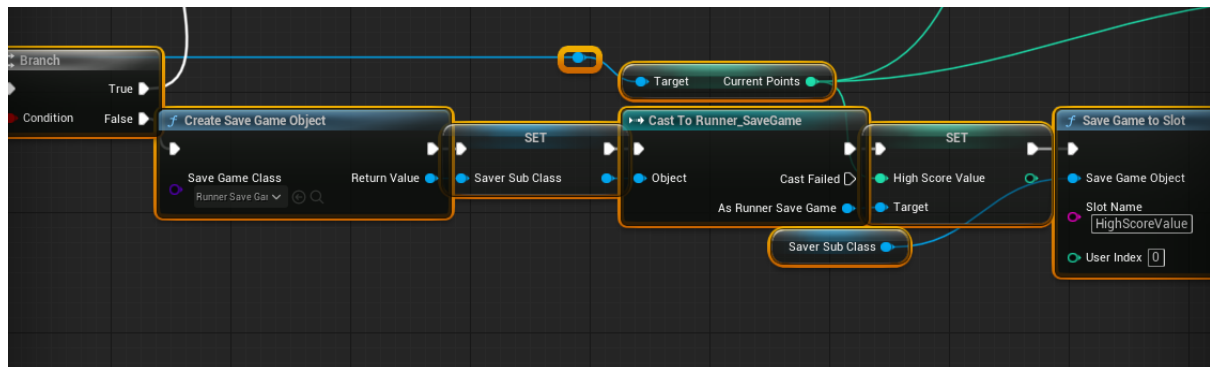


рис. 3.3.1.8.

Тепер HighScore зберігається і можна додати відображення в UI.

В Gameplay\_UI в текстовому блоці який використовуватиметься для Current Score додаємо Binding функцію Get\_CurrentScore\_Text. Всередині отримуємо Get Game Mode→Cast to BP\_RunnerGameMode, As BP Runner Game Mode→Get CurrentPoints→ToText додаємо Return node, з'єднуємо з return value та exec піном.

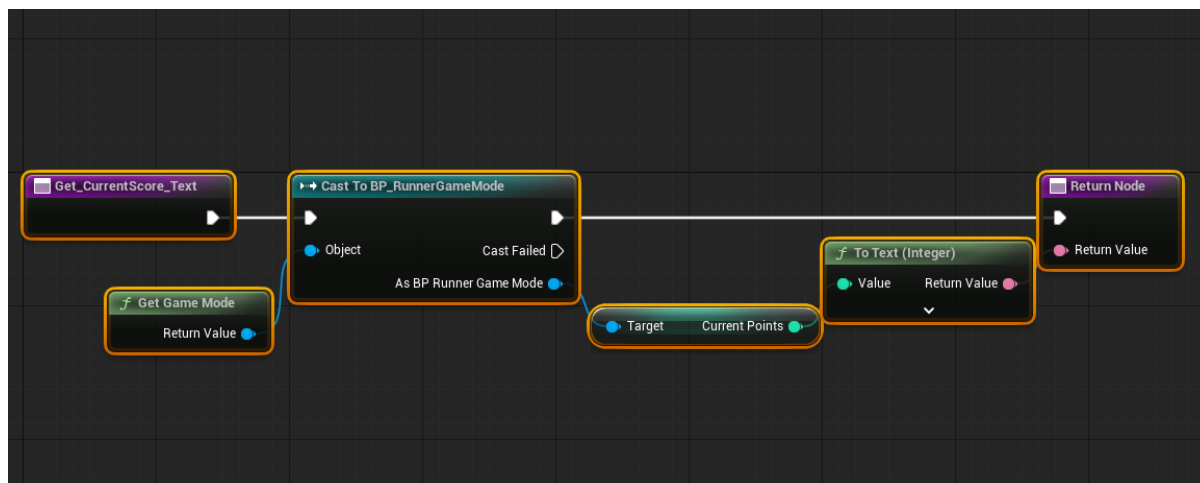


рис. 3.3.1.9.

В текстовому блоці що використовуватиметься для BestScore додаємо прив'язану функцію Get\_BestScore\_Text, всередині додаємо Load Game From Slot, Slot Name – HighScoreValue, Return Value → Cast to Runner\_SaveGame, As Runner Save Game→Get HighScoreValue→To Text, Return Value з'єднуємо з Return нодом.

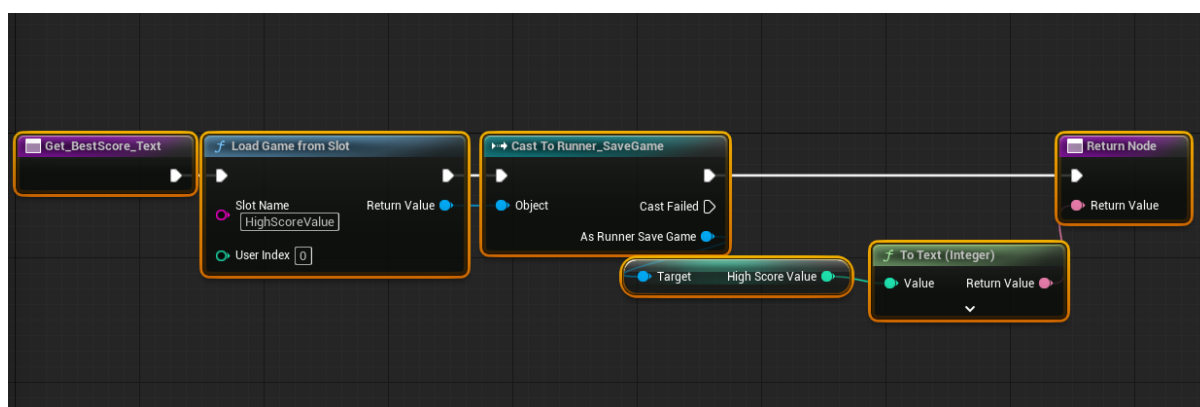


рис. 3.3.1.10.

В Progress Bar що репрезентує магніт в Details біля імені ставимо галочку Is Variable, ім'я – Magnet\_PB в Behavior→Visibility ставимо Collapsed. Додаємо

прив'язану функцію `Get_Magnet_Percent`, всередині додаємо `Get Game Mode` → `Cast to BP_RunnerGameMode`, `As BP Runner Game Mode` → `Magnet Timer` → `Division (float)` на 5. `Return Value` до `Return` ноду.

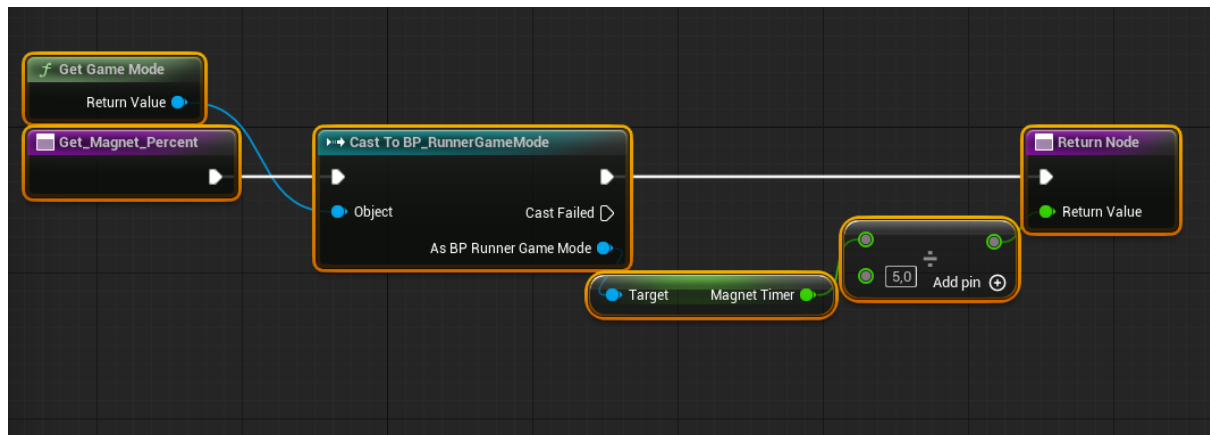


рис. 3.3.1.11.

В `BP_RunnerCharacter` в `Event Begin Play` перед усім іншим додаємо → `Create Widget`, `Class` – `Gameplay UI`, `return value` → `promote to variable`, ім'я – `MainUIRef` → `Add To Viewport`.

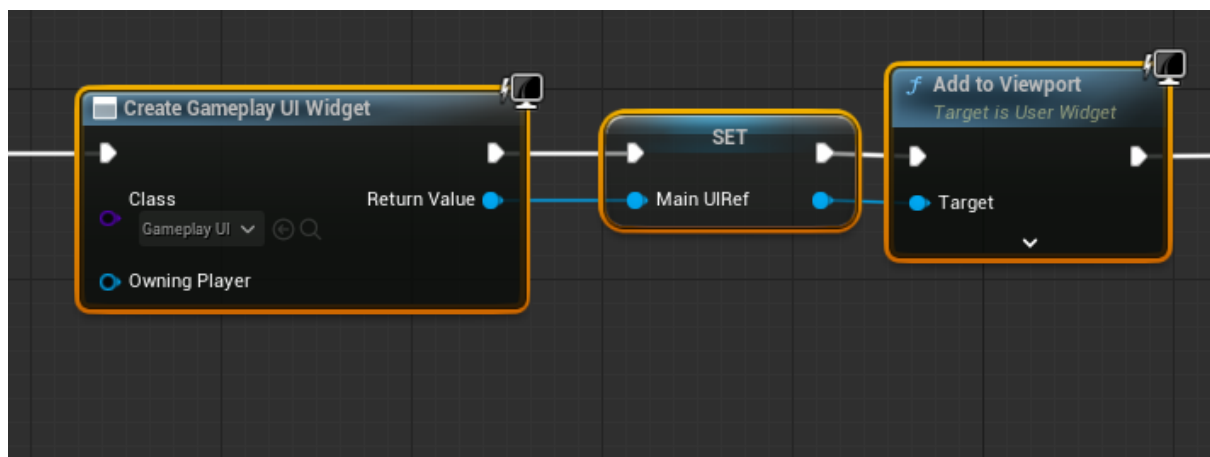


рис. 3.3.1.12.

В BP\_Powerup\_Magnet між Set Magnet Timer та Sequence, додаємо Get Player Controller → Cast to BP\_RunnerCharacter, As BP Runner Character→Get Main UIRef → Get Magnet PB → Set Visibility, In Visibility – Visible.

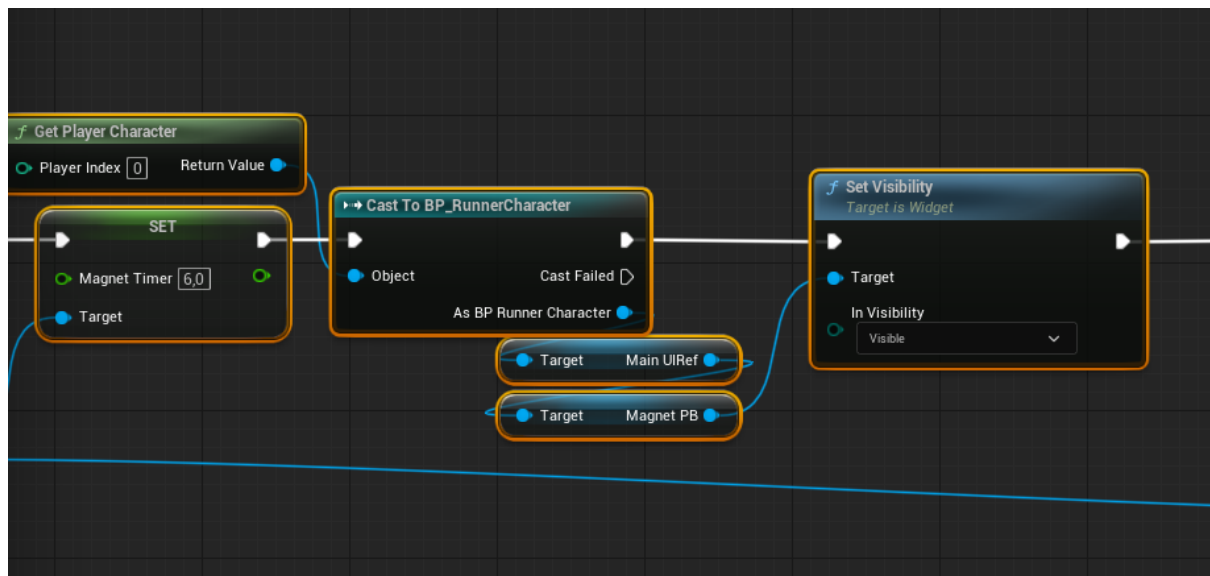


рис. 3.3.1.13.

В Sequence then 1 з Cast BP\_RunnerCharacter додаємо те саме тільки In Visibility ставимо Collapsed.

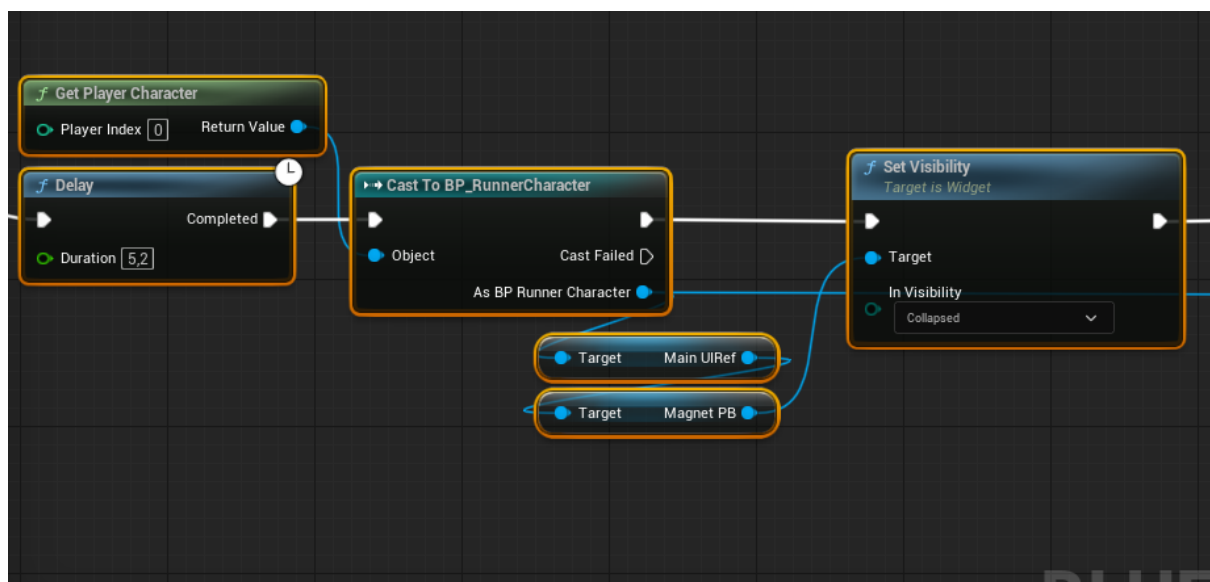


рис. 3.3.1.14.

### 3.3.2 Створення віджетів для відображення поразки та меню паузи

#### Реалізація віджету паузи

Кнопка для віджету паузи була додана раніше – знаходиться в верхньому лівому куті основного геймплейного віджету. Але треба створити окремий віджет блюпрінт для відображення.

Створюємо віджет блюпрінт з назвою `PauseMenu_UI`. Всередині в дизайнері додаємо `Canvas panel`, туди додаємо `Background blur` для ефекту розмиття, текст `PAUSED` та 3 кнопки – `Resume`, `Options`, `Main Menu`, в будь-якому порядку.



рис. 3.3.2.1.

В кнопці `Resume` Додаємо `Event On Clicked`, звідти додаємо `Branch` в `condition` → `Is Game Paused`, з `True` → `Set Game Paused false` → `Set Input Mode Game Only` в

Player Controller → get player controller, з нього ж Set Show Mouse Cursor false → Remove From Parent.

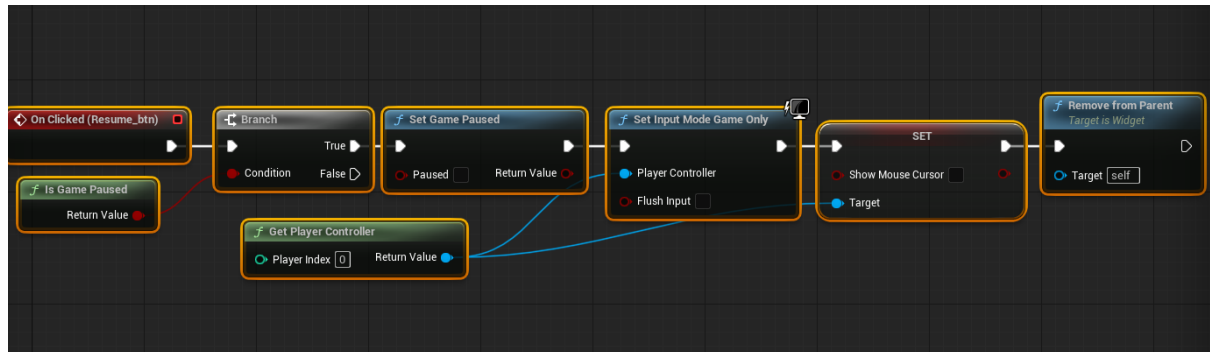


рис. 3.3.2.2.

В Gameplay\_UI, заходимо в кнопку паузи, в Details→Event додаємо On Clicked, з івенту додаємо Branch в condition → нод Is Game Paused. з False → Set Game Paused, Paused = true → Set Input Mode UI Only, в Player Controller → Get Player Controller.

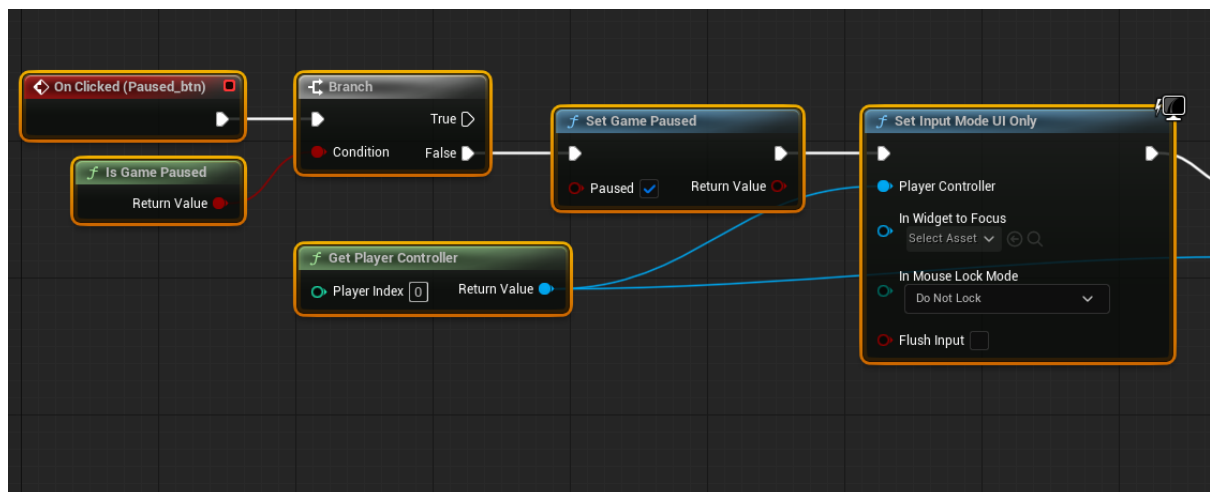


рис. 3.3.2.3.

з exec піну → Set Show Mouse Cursor False → Construct Object from Class обираємо Pause Menu UI, return value → Add To Viewport.

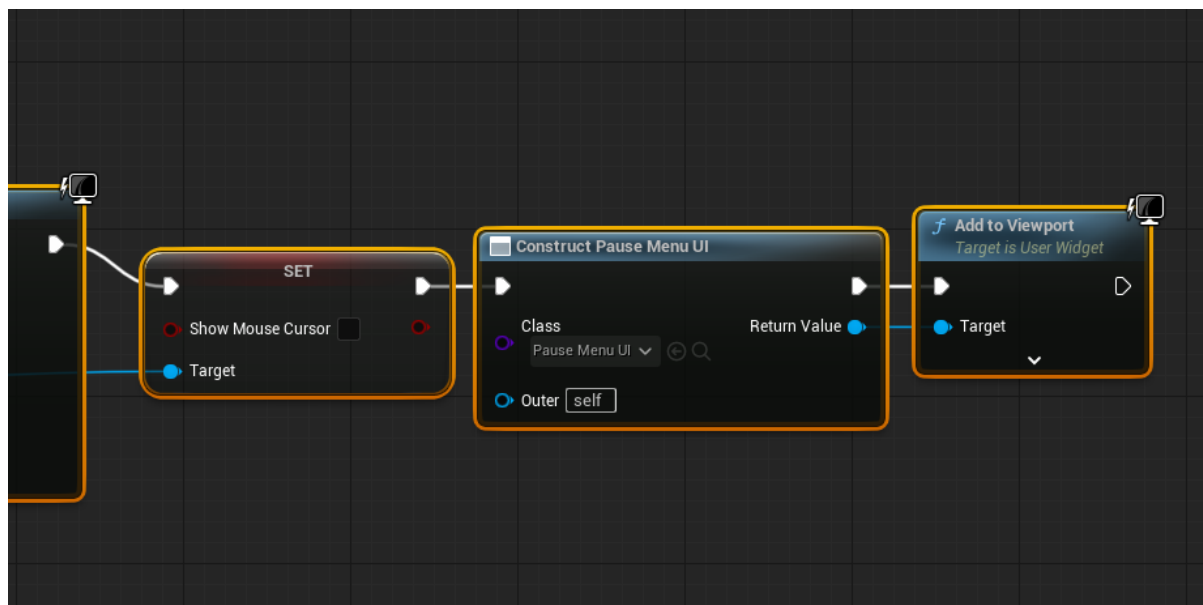


рис. 3.3.2.4.

### Віджет поразки

Додамо віджет поразки, який з'являтиметься при зіткенні з перешкодами. Створюємо віджет блюпрінт `GameOver_UI`, всередині майже як із `PauseMenu_UI`, 3 кнопки – `Resume`, `Restart`, `Main Menu`, надпис `GAME OVER`.

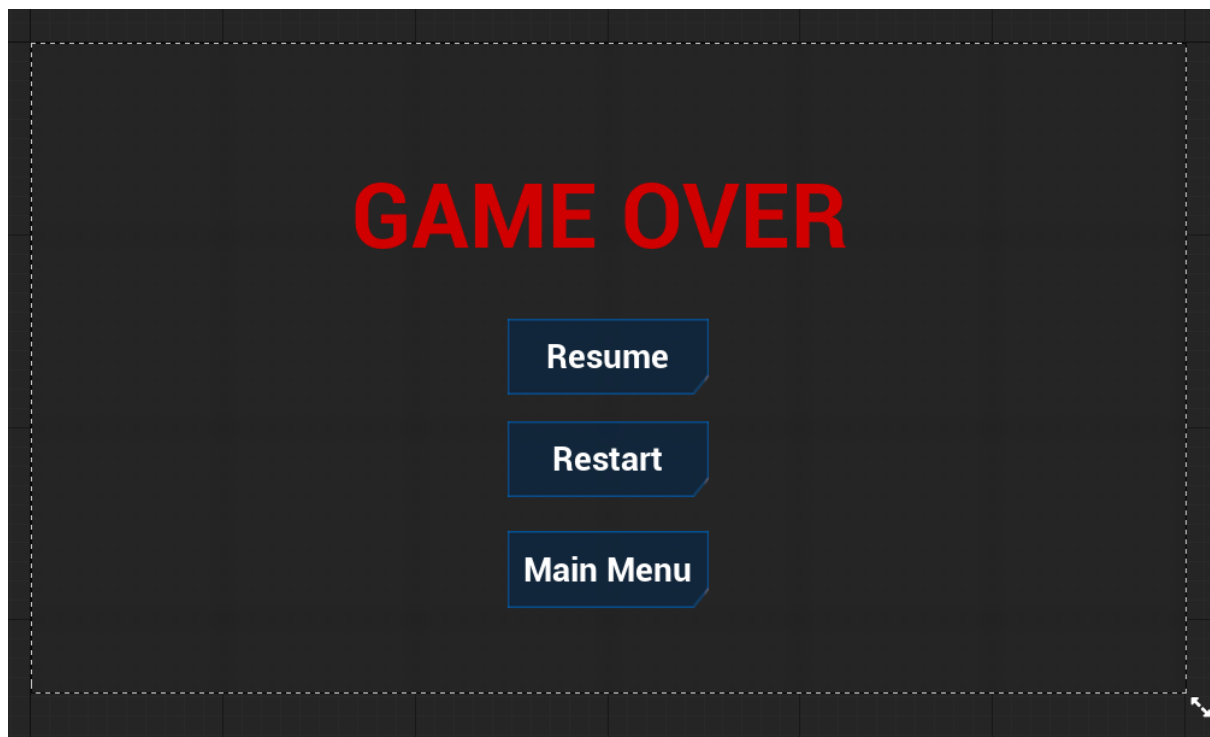


рис. 3.3.2.5

Для кнопки Restart додаємо OnClicked івент → Execute console command, command – restartlevel.

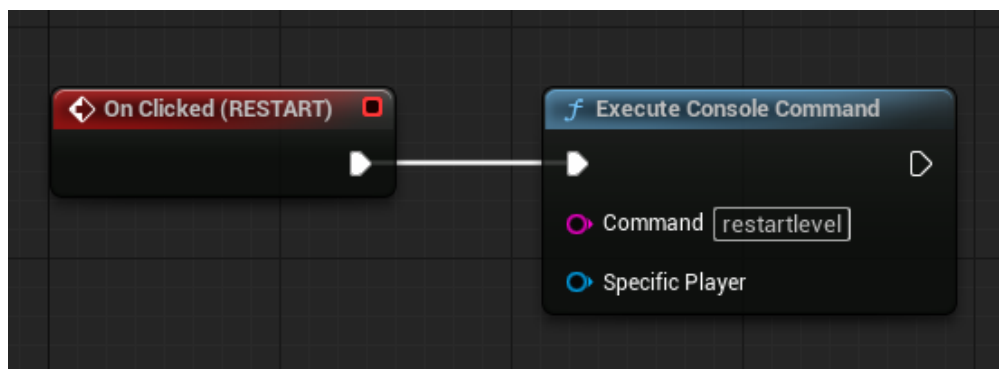


рис. 3.3.2.6.

Тепер в BP\_RunnerCharacter створимо функцію, яка реалізовуватиме поразку. Створюємо змінну isDead типу boolean. Створюємо функцію з назвою Death. Всередині додаємо Set IsDead True, з Components додаємо на пусте місце Character Movement з нього

Disable Movement → Disable Input додаємо 3 Components  
 mesh → Set Simulate Physics Simulate true.

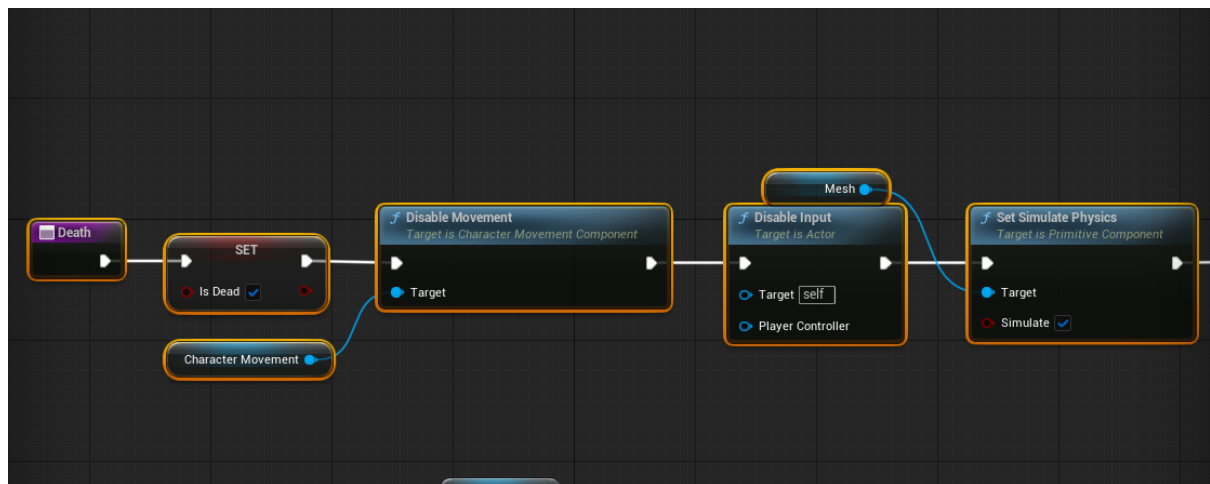


рис. 3.3.2.7.

3 exec піну → Remove All Widgets → Create Widget,  
 class – Game Over UI, Return Value → Add To Viewport.  
 Додаємо Get Player Controller → Set Show Mouse Cursor  
 true → Save High Score → Update Coins.

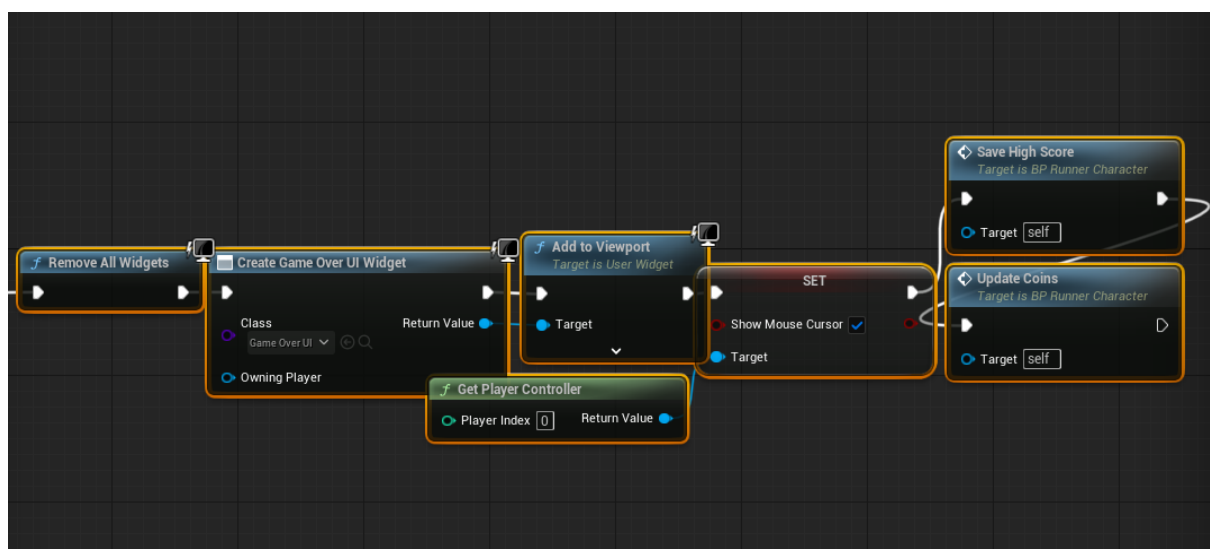


рис. 3.3.2.8.

В BP\_Obstacles\_Blocker в Event Actor Begin Overlap, з other actor → Cast to BP\_RunnerCharacter → Branch. з As BP Runner Character → isSliding з'єднуємо з Condition. False → Death, Target → As BP Runner Character. з Exec піну → Delay 1.0 → Set Game Paused true.

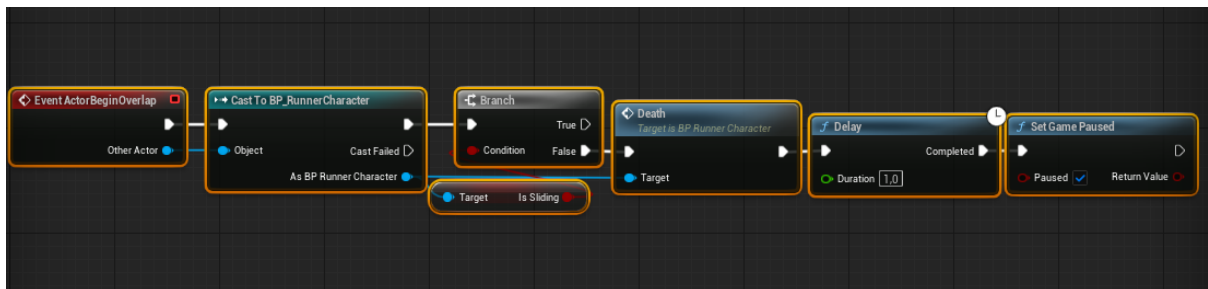


рис. 3.3.2.9.

В BP\_Obstacles\_Wall додаємо те саме що в BP\_Obstacles\_Blocker, тільки видаляємо Branch з IsSliding.

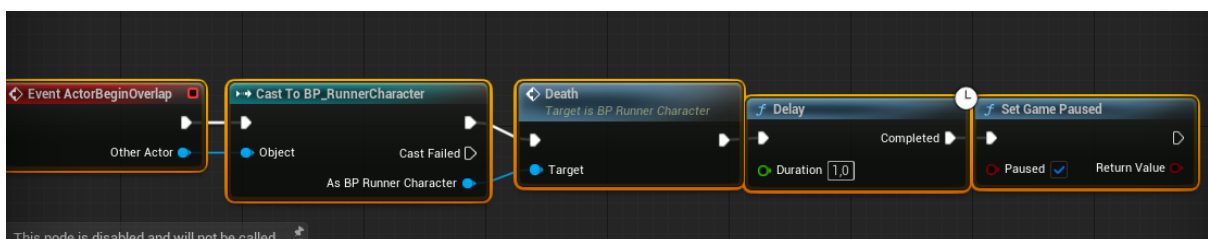


рис. 3.3.2.10.

Тепер персонаж помиратиме від зіткнень з перешкодами та з'являтиметься меню поразки. Додамо реалізацію відновлення гри після поразки.

### Відновлення гри після поразки

Створимо блюпрінт актора BP\_Obstacle\_Destroyer в Default Scene root додамо Sphere Collision поставимо scale на 60 60 60.

В Event Actor Begin Overlap, додаємо Get Player Controller → Cast to BP\_Runner Character, з'єднуємо з івентом. З ехес піну додаємо Branch. З As BP Runner Character → Get IsDead з'єднуємо з Condition.

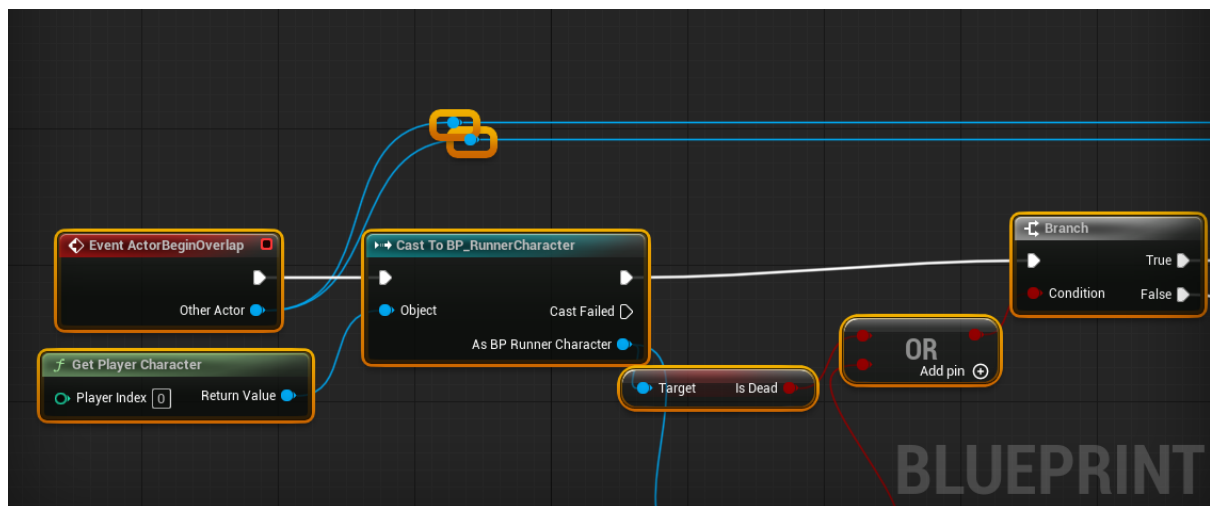


рис. 3.3.2.11.

З False додаємо Destroy Actor. З True → Sequence, з then 0 → з other actor піну в ноді івенту → Cast to BP\_Obstacle\_Blocker, As BP Obstacle Blocker → Destroy Actor → Delay → Destroy Actor.

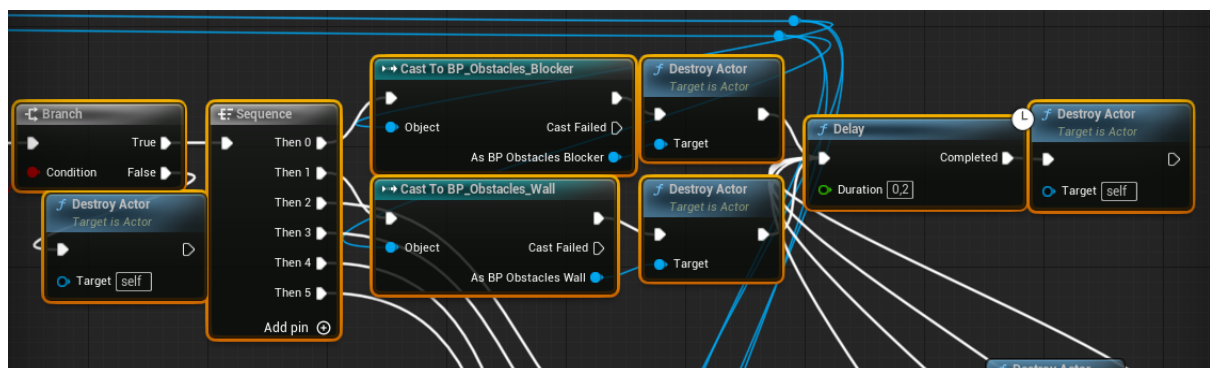


рис. 3.3.2.12.

В BP\_RunnerCharacter створюємо функцію Resume Game. Всередині додаємо IsDead→Branch. Додаємо Mesh component → Set relative Transform Split struct pin, потім split transform location pin, Location Z = -90, Teleport true.

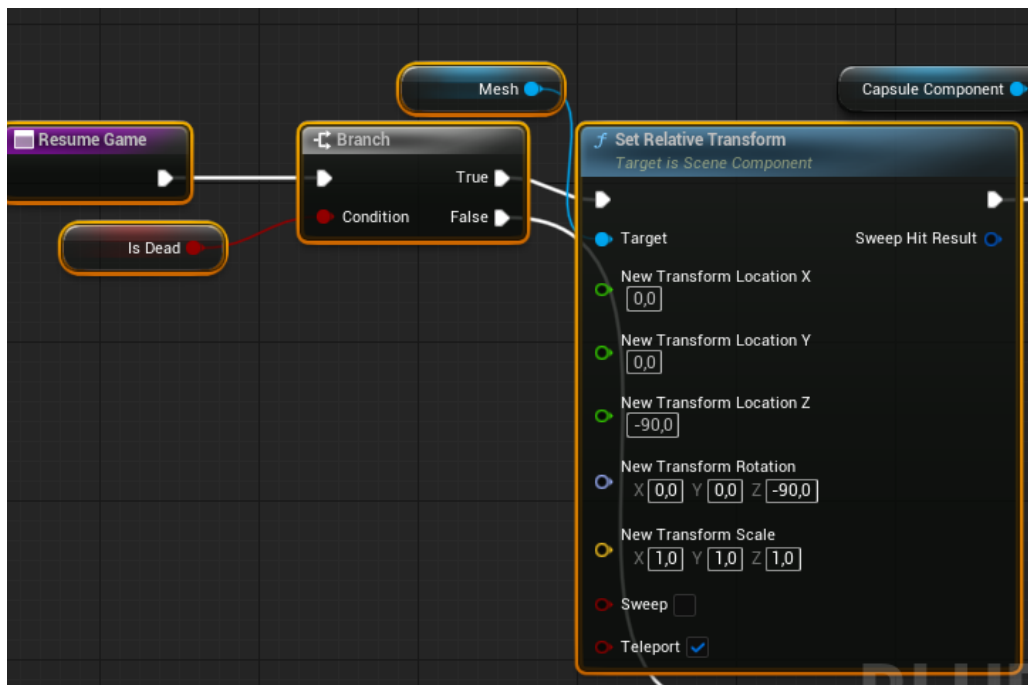


рис. 3.3.2.13.

3 exec піну → Set current index = 1 → Set Actor Location, split transform pin. Додаємо Capsule Component → Get Relative Transform, Split transform, split location pin, X до X Set actor location, Z до Z. 3 Exec піну → Attach Component to Component, Target → Mesh, Parent → Capsule Component.

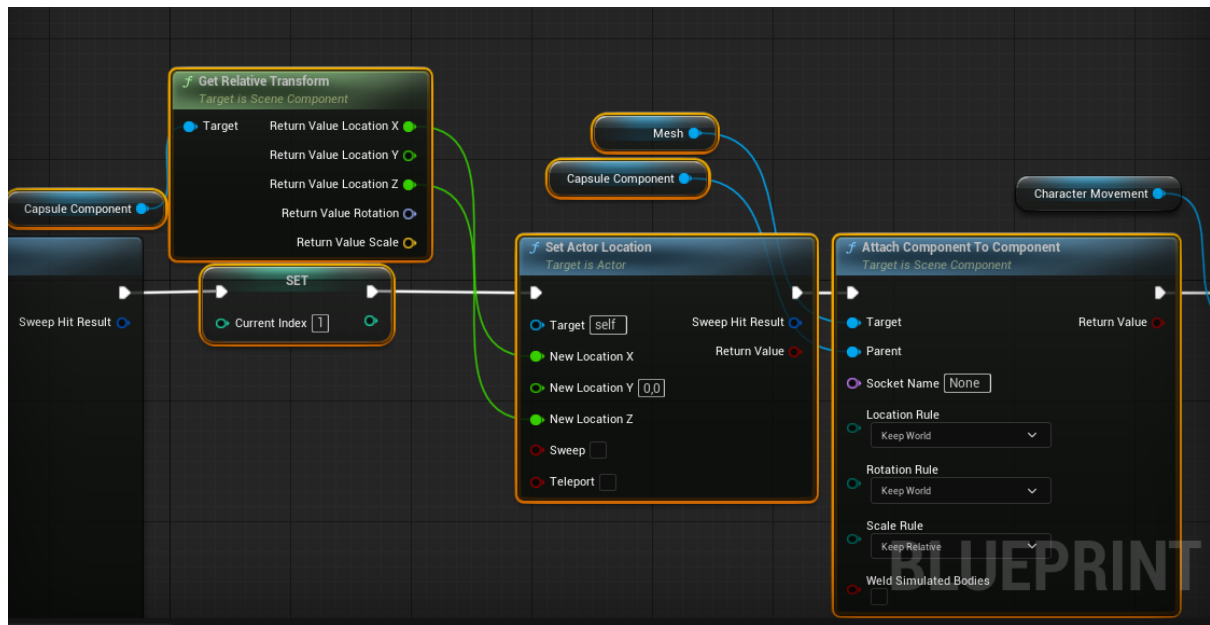


рис. 3.3.2.14.

3 Ехес піну → Додаємо Character Movement → Set Movement mode Walking → Enable Input → додаємо Mesh → Set Simulate Physics.

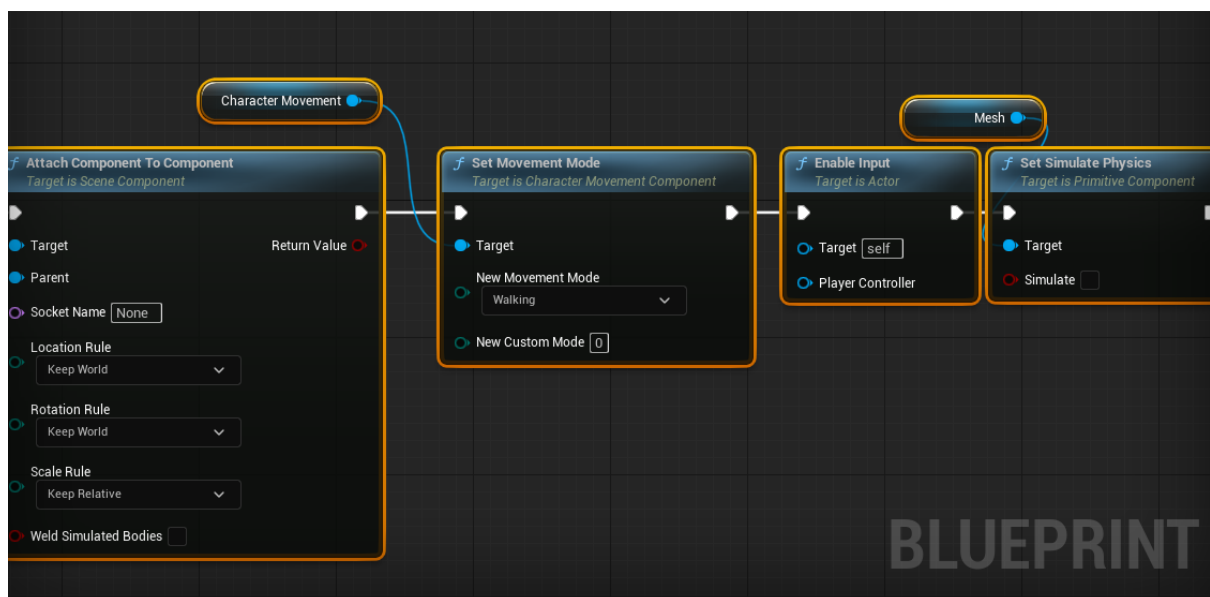


рис. 3.3.2.15.

3 ехес піну → Remove All Widgets → Create Widget class – Gameplay UI, Return Value → Set Main UIRef, піл

повернення змінної → Add To Viewport, Get Player Controller → Set Input Mode Game Only, → від Player Controller → Set Show Mouse Cursor false.

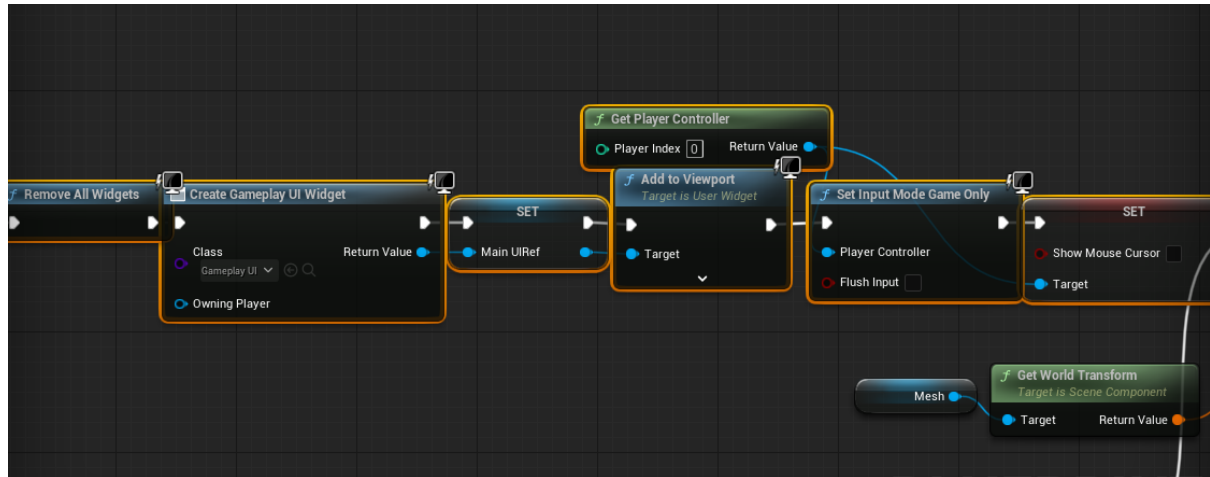


рис. 3.3.2.16.

З ехес піну → SpawnActor From Class, обираємо клас – BP\_Obstacle\_Destroyer. В Spawn Transform → додаємо Mesh→Get World Transform, return value під'єднуємо до spawn transform.

False з Branch під'єднуємо до Spawn Actor. Зі Spawn Actor → Branch, condition → Is Game Paused. True → Set game Paused False → set is Dead false. False → Set is Dead false.

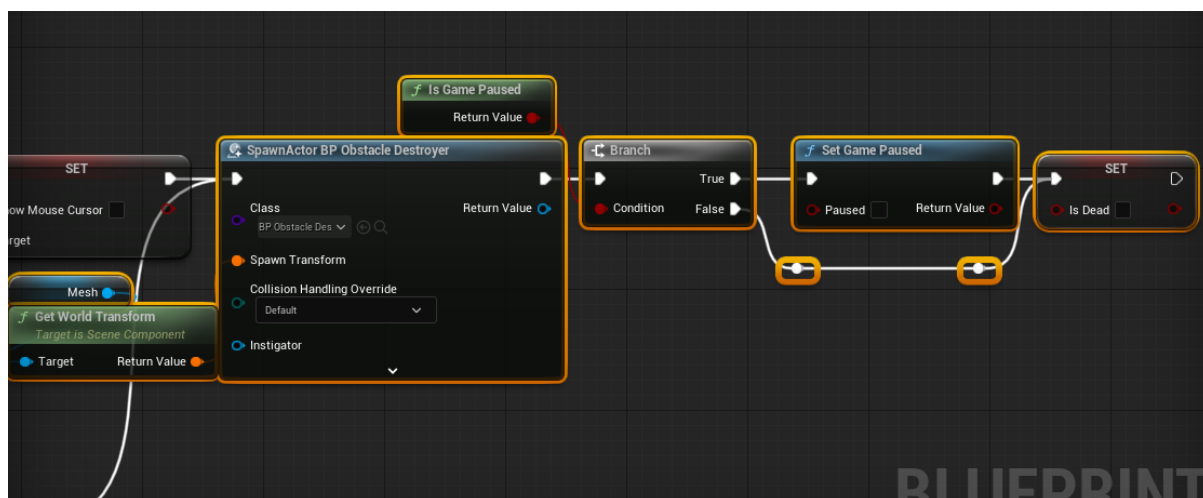


рис. 3.3.2.17.

Тепер для Resume в GameOver\_UI. Спочатку створимо віджет блюпрінт ResumePopup\_WBP. Всередині обов'язкові елементи: текст посередині – назва Warning Text, поставити галочку Is Variable, кнопка YES, назва Yes\_btn, галочка Is Variable, кнопка Close – Close\_btn, галочка Is Variable, кнопка No – No\_btn, галочка Is Variable.

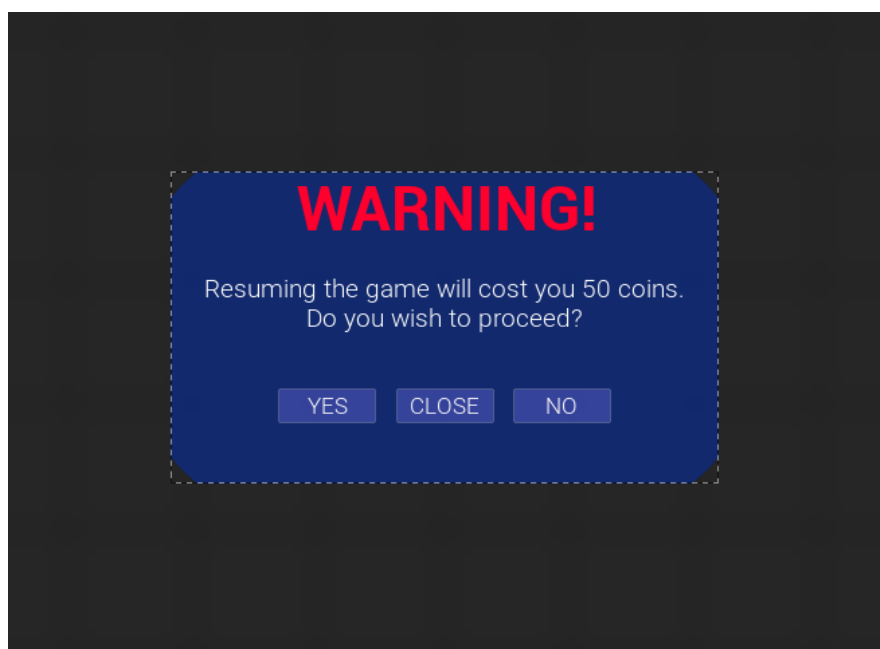


рис. 3.3.2.18.

B event graph створюємо custom event get coins value→Load Game From Slot, Slot name – CoinValue, Return Value → Cast to Runner Save Game. 3 As Runner Save Game → Get Coin Value → Less (<), B = 50, Return value → Branch.

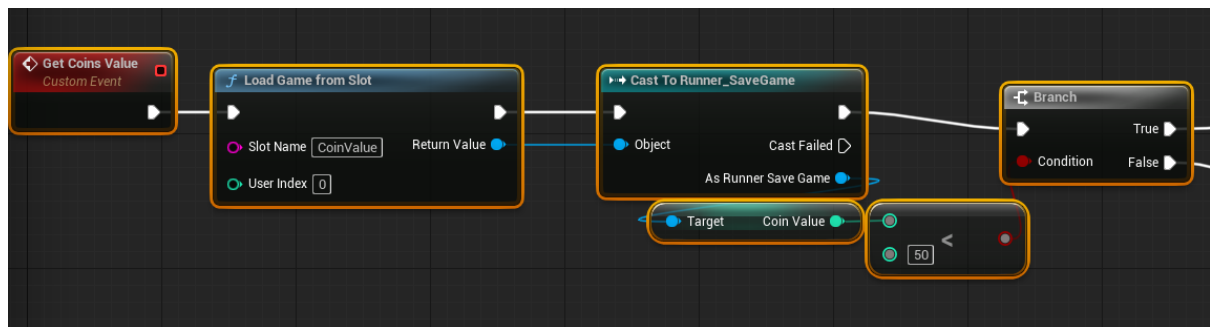


рис. 3.3.2.19.

True → 3 variables додаємо WarningText→Set Text, в текст: You need at least 50 coins to resume the game. Додаємо No\_btn та Yes\_btn, під'єднуємо до Set Visibility Hidden. Додаємо Close\_btn → Set Visibility Visible.

False→ WarningText→Set Text, в текст: Resuming the game will cost you 50 coins. Do you wish to proceed?. Yes\_btn та No\_btn під'єднуємо до Set Visibility Visible → Close\_btn до Set Visibility Hidden.

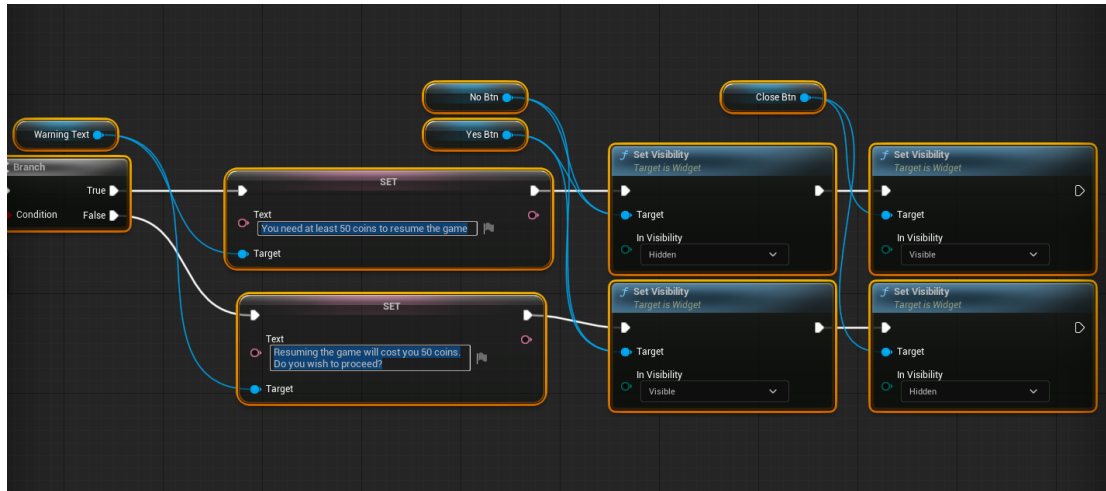


рис. 3.3.2.20.

В Event Construct додати Get Coins Value.

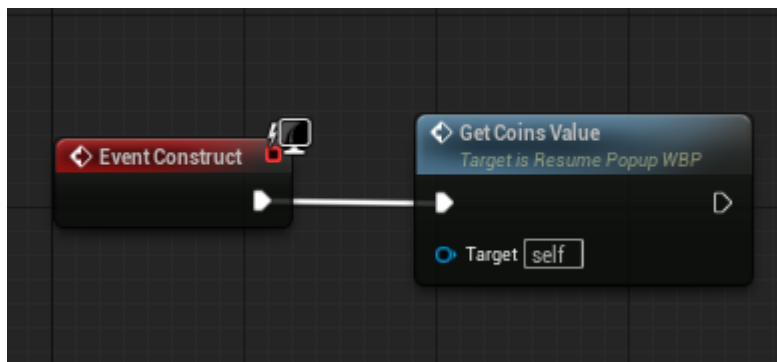


рис. 3.3.2.21.

Додати Custom Event Subtract Coins → Load Game From Slot CoinValue, Return Value → Cast To Runner Save Game, As Runner Save Game → Less (<) 50, Return Value → Branch.

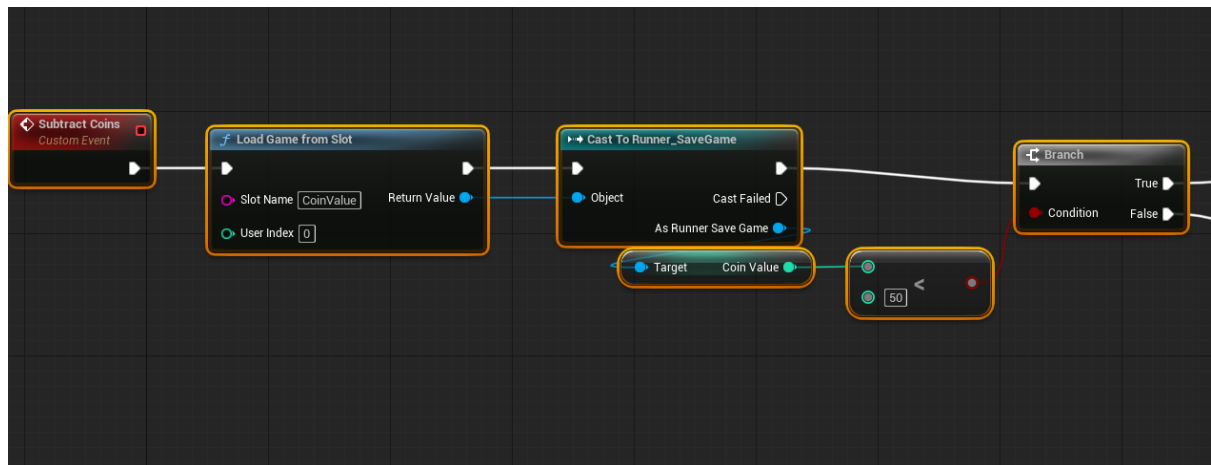


рис. 3.3.2.22.

3 True копіюємо те саме що в Get Coins Value. 3 False, Get Player Character → Cast to BP\_RunnerCharacter з'єднуємо до false. Get Game Mode → Cast to BP\_RunnerGameMode, з'єднуємо до попереднього ехес піну. As BP Runner Game Mode → Get Coins → Subtract 50 → As BP Runner Game Mode Set Coins.

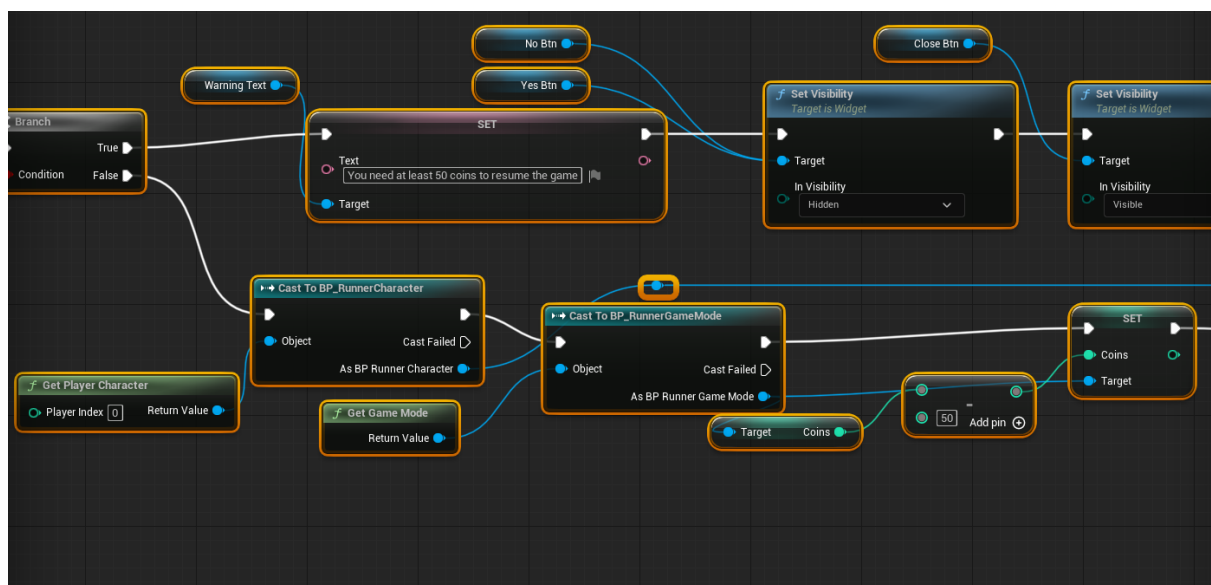


рис. 3.3.2.23.

3 As BP Runner Character → Update Coins → Resume Game.

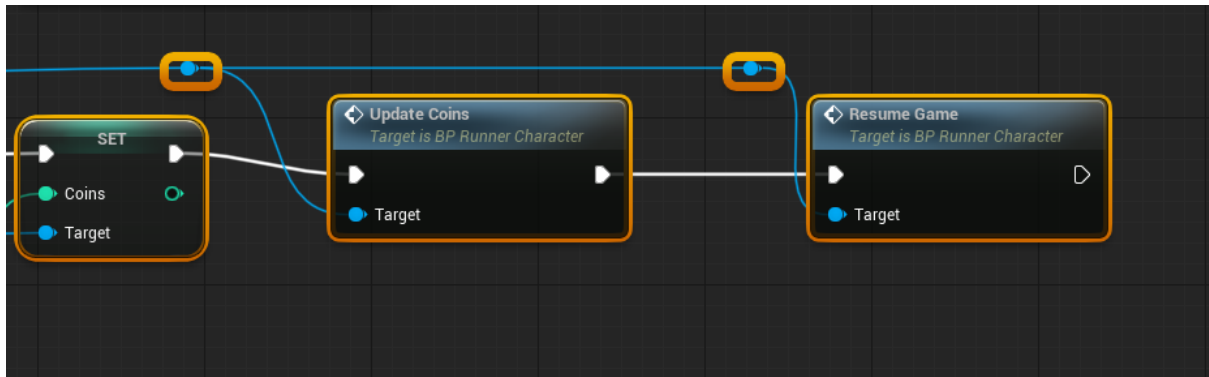


рис. 3.3.2.24.

В кнопку Yes\_btn додаємо OnClicked→Subtract Coins

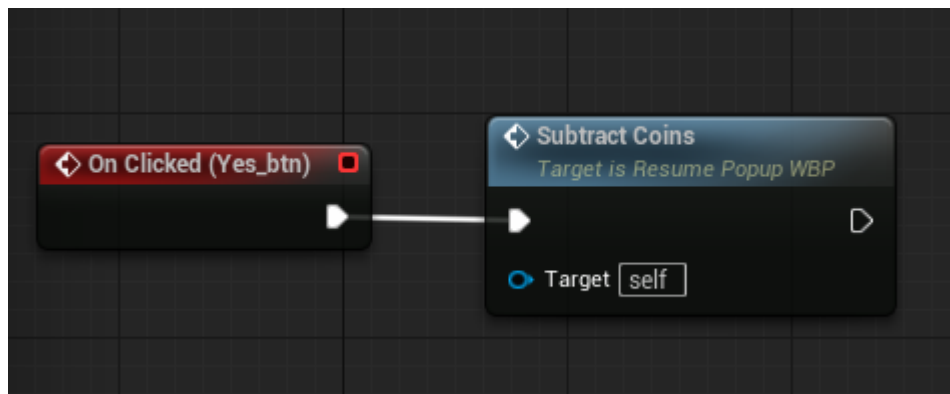


рис. 3.3.2.25.

В кнопки No\_btn та Close\_btn → Remove From Parent → Create Widget from class, class – Game Over UI → Add to Viewport.

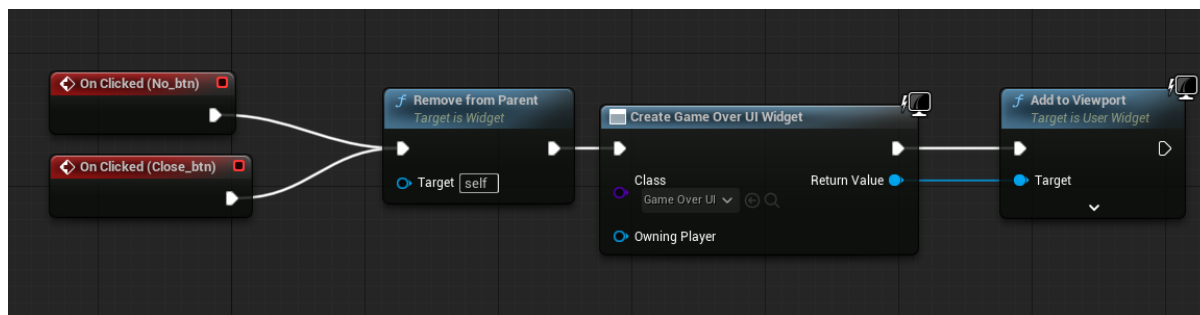


рис. 3.3.2.26.

### 3.3.3 Створення головного меню гри та меню вибору персонажа.

#### Головне меню

Створюємо нову мапу MainMenu. Створюємо Widget Blueprint MainMenu\_UI. Всередині 2 текстових блоки, верхній найкращий результати, нижче – кількість монет. Посередині кнопки.

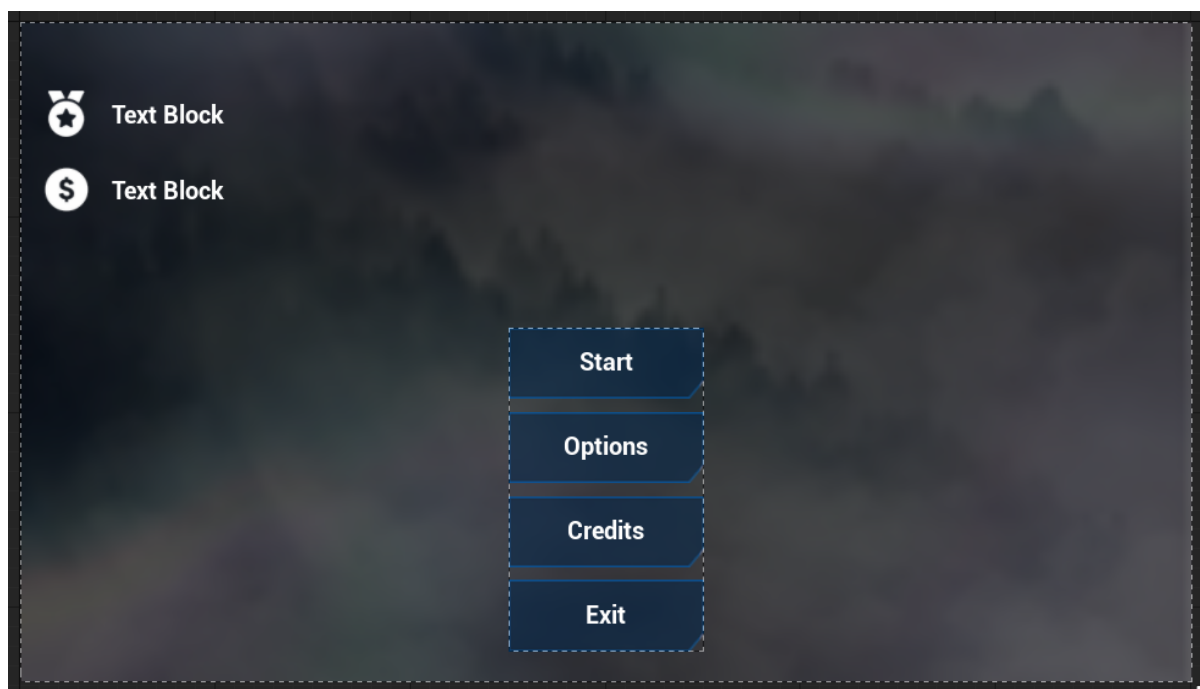


рис. 3.3.3.1.

Для рекорду створимо функцію GetHighScoreText → Load Game From Slot, slot name – HighScoreValue, Return

Value → Cast to Runner\_SaveGame, As Runner Save Game → High Score Value → To Text → Return Node.

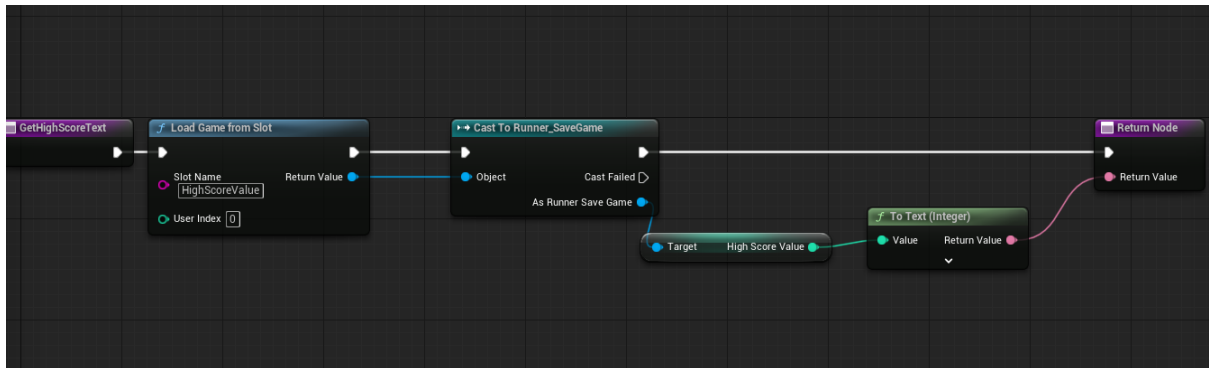


рис. 3.3.3.2.

Для монет створюємо функцію `GetCoinText`, копіюємо те що в `GetHighScoreText`. Замінюємо `SlotName` на `CoinValue`, з `As Runner Save Game` → `Coin Value`.

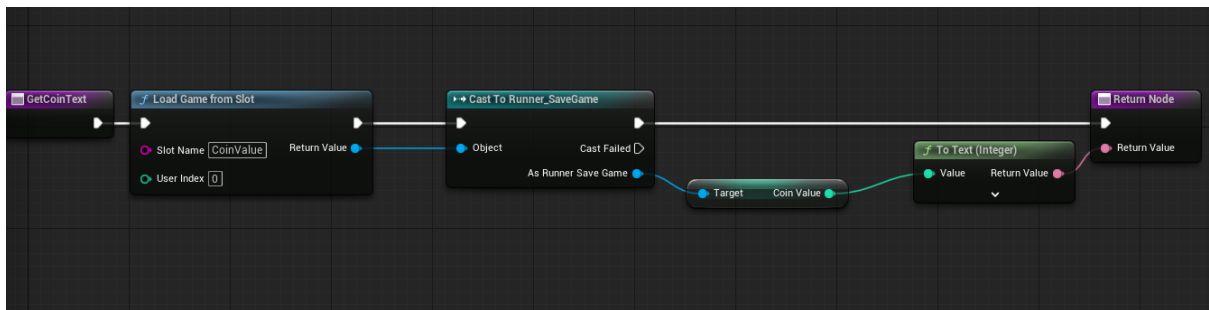


рис. 3.3.3.3.

Для кнопки старту додаємо event `OnClicked` з нього `Open Level (by reference)`, обираємо `Runner_Map`.

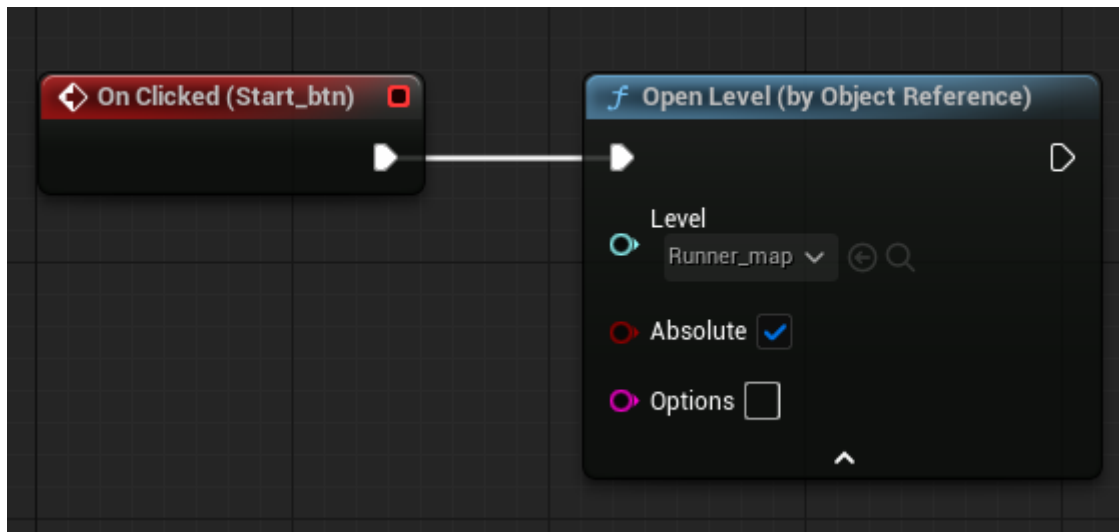


рис. 3.3.3.4.

Для кнопки Quit (Exit) додаємо OnClicked → Execute console command, command = exit.

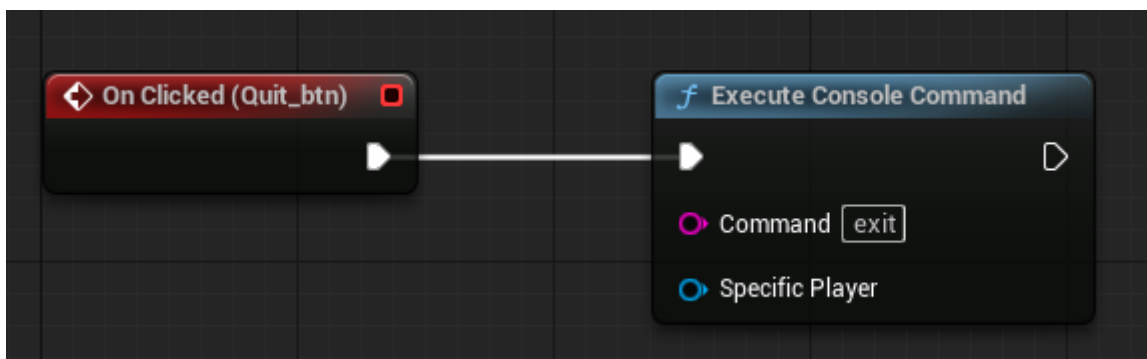


рис. 3.3.3.5.

В level blueprint мапи головного меню в event begin play додаємо Get Player Controller з нього Set Input Mode UI Only, з'єднуємо з ехес піном. 3 Get Player Controller → Set Show Mouse Cursor True → Create Widget, class – Main Menu UI → Add To Viewport.

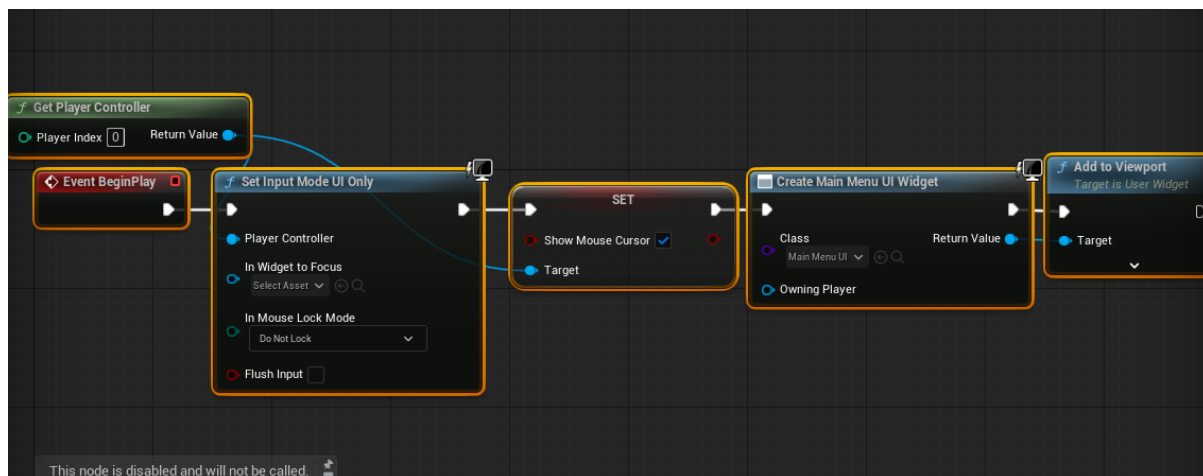


рис. 3.3.3.6.

В Project settings Game Default Map обираємо MainMenu.

## Меню вибору персонажа

Створимо мапу CS\_Map. Створимо блюпрінт Актора BP\_CSCharacters. Всередині в Default scene root додамо Skeletal mesh, в mesh asset вибрати SKM\_Manny, в anim class вибрати відповідний анімаційний блюпрінт – ABP\_Manny.

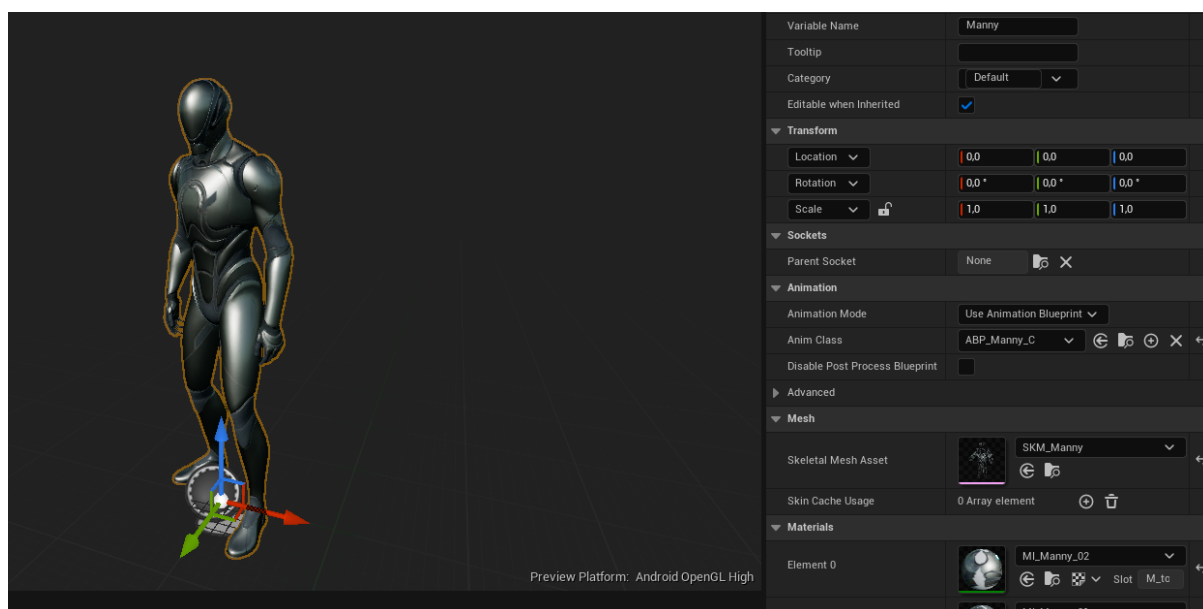


рис. 3.3.3.7.

Так само додаємо усіх персонажів які є. В Event Graph створюємо event Show Manny, переносимо з Components його mesh на пусте місце і викликаємо Set Hiddent in Game false. Для всіх інших персонажів які є викикаємо те саме і ставимо New Hidden True. Для кожного створюємо такий самий івент.

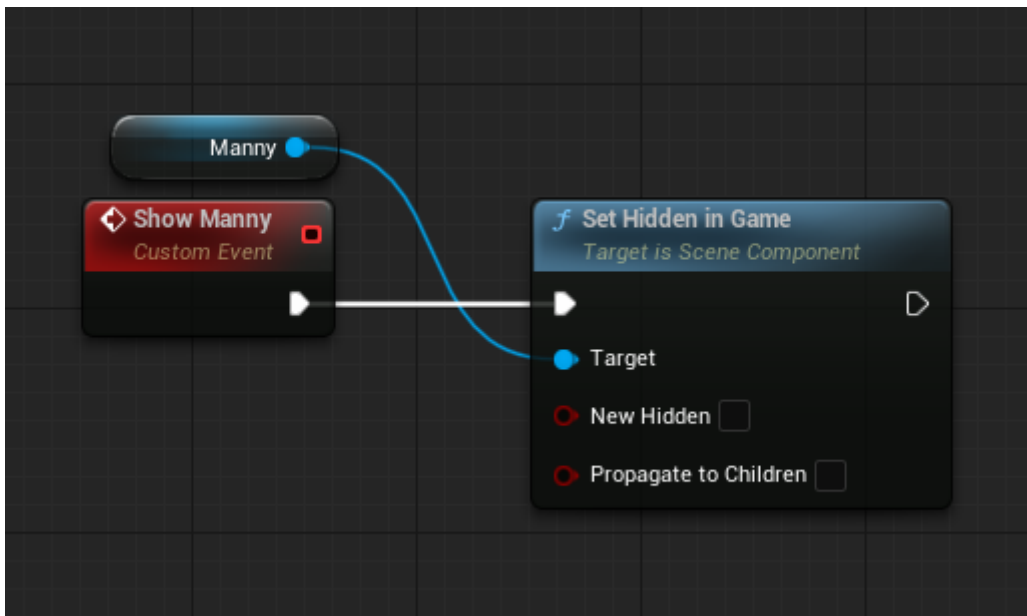


рис. 3.3.3.8.

Створюємо івент Spawn Manny → Cast to Pawn Class, обираємо блюпрінт з персонажем. As Pawn → promote to variable.

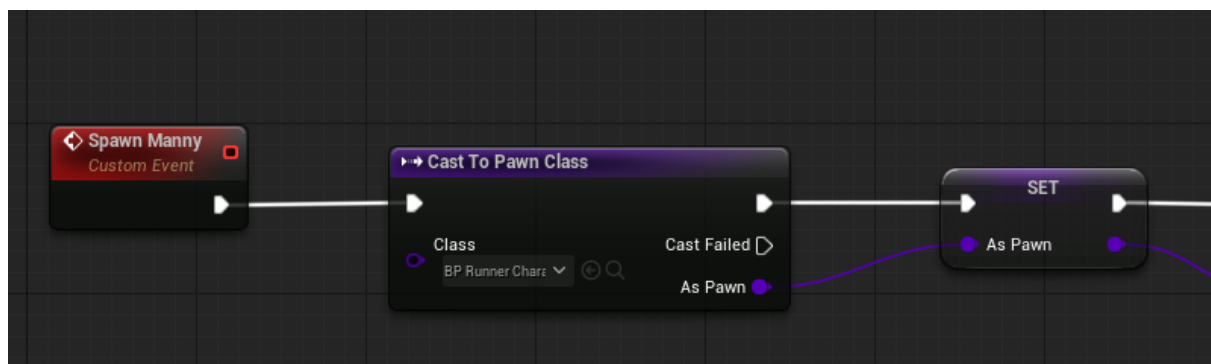


рис. 3.3.3.9.

В Runner\_SaveGame створюємо змінну типу Pawn PlayerCharacter. В BP\_CSCharacters створюємо функцію SaveCharacters → DoesSaveGameExist, SlotName PlayerCharacter. Return value → Branch. True→Load Game From Slot, slot name PlayerCharacter, return value → promote to variable name = Saver Sub Class.

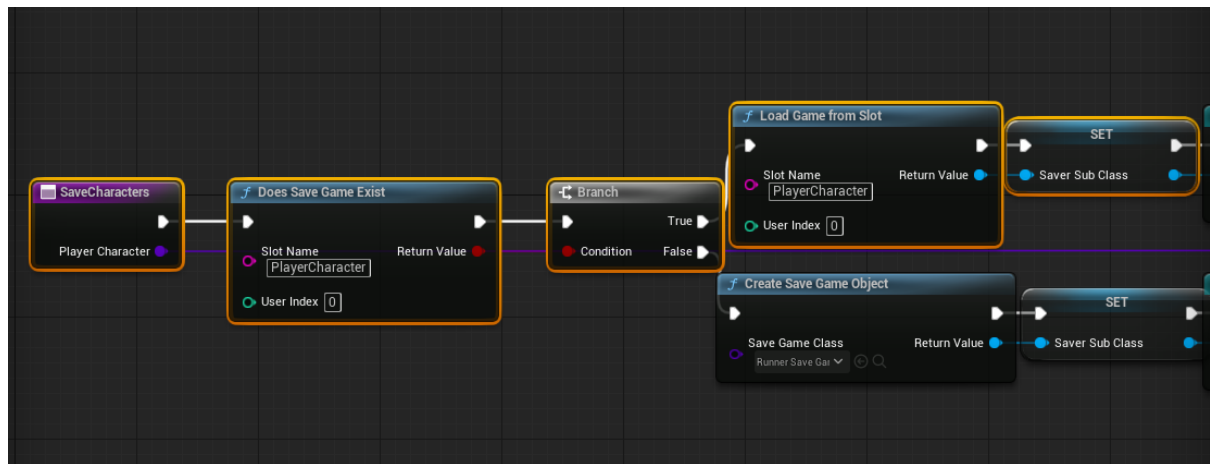


рис. 3.3.3.10.

З повернення змінної → Cast to Runner\_SaveGame, As Runner Save Game → Set Player Character. Натискаємо на початковий нод SaveCharacters і додаємо в Inputs параметр PlayerCharacter типу Pawn. Його передаємо в PlayerCharacter в ноді Set. З ехес піну → Save Game To Slot, SlotName PlayerCharacter → Open Level (By Object reference) обираємо Runner\_map.

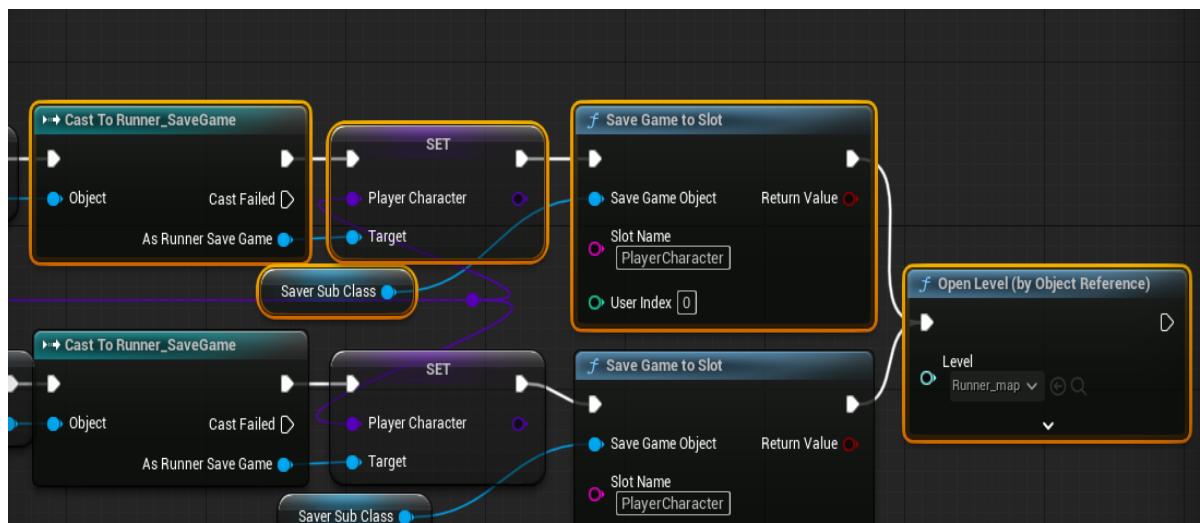


рис. 3.3.3.11.

В False копіюємо те саме що в True, тільки замість Load Game From Slot додаємо Create Save Game object, Save game class → Runner Save Game.

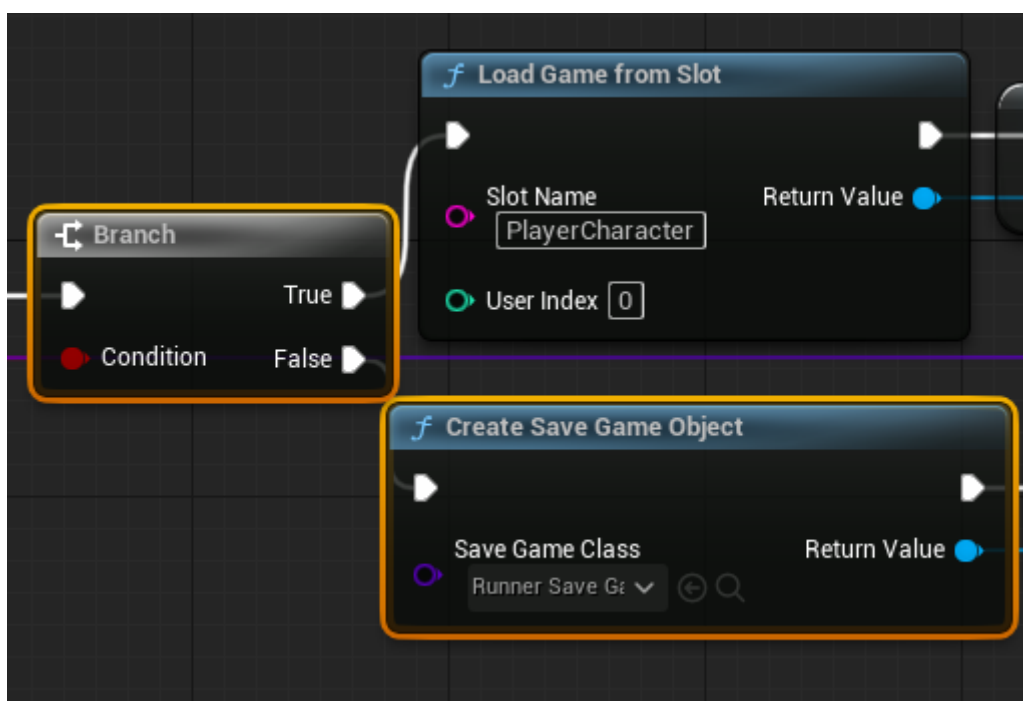


рис. 3.3.3.12.

В event graph в Spawn Manny після Set As Pawn додаємо Save Characters.

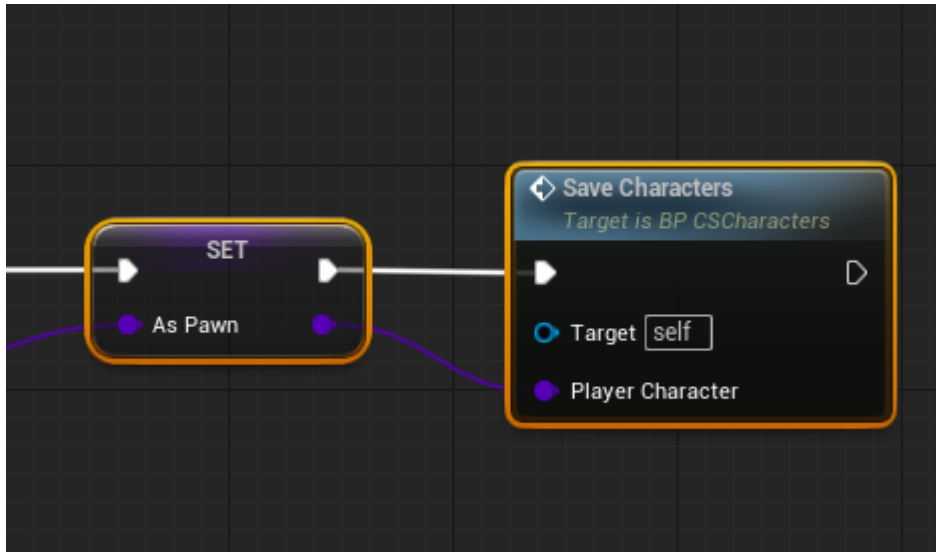


рис. 3.3.3.13.

Створюємо віджет блюпрінт CSelection\_UI. Всередині в Canvas Panel додаємо 3 кнопки, посередині кнопки < > Для перемикавання персонажів, в правому верхньому куту кнопку Start

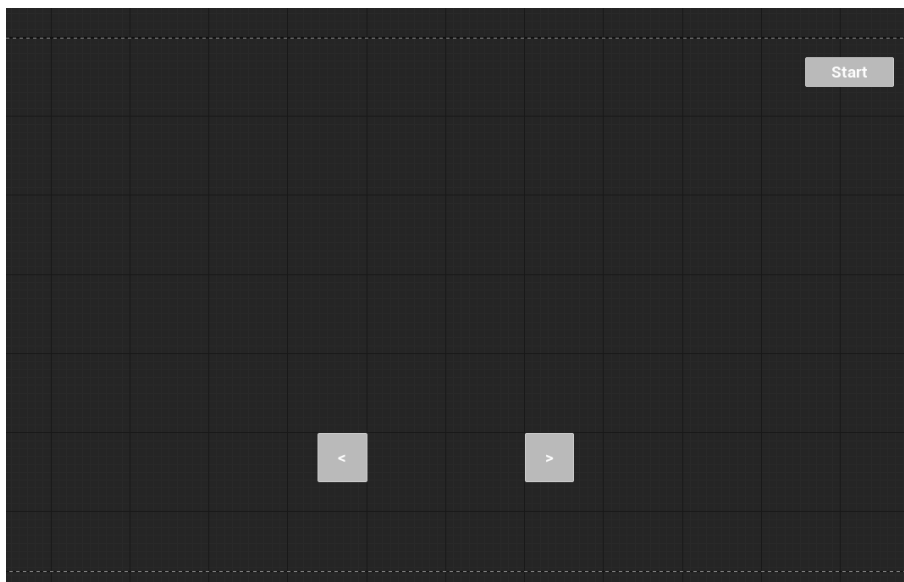


рис. 3.3.3.14.

В Event Graph створюємо змінну CS\_Characters типу BP\_CSCharacters. З Event Construct → Get Actor of Class,

клас обираємо BP CSCharacters, return value → set CS Characters.

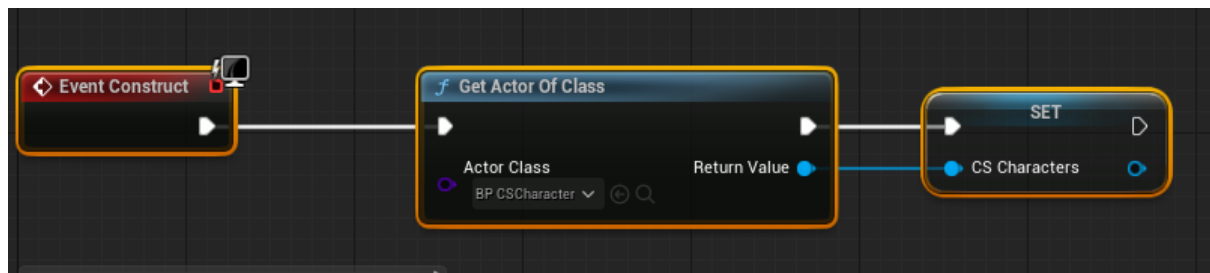


рис 3.3.3.15.

Створюємо змінну Character Index типу int. Створюємо Custom event Select Character, в ньому додаємо Inputs CharacterIndex типу int, з нього → Clamp 0–2, Return Value → Set Character Index.

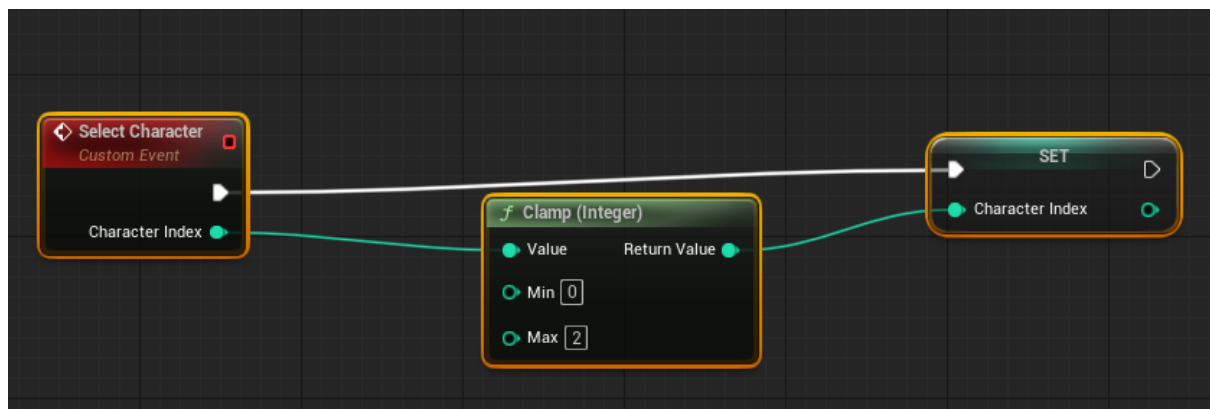


рис. 3.3.3.16.

Створюємо Custom Event Show Characters. з ехес піну → Sequence, then 0 → Branch. На пусте місце додаємо Character Index → Equeal(==) нод, пін B = 0, return value до Condition. Get CS\_Characters на пустому місці → Show Manny.

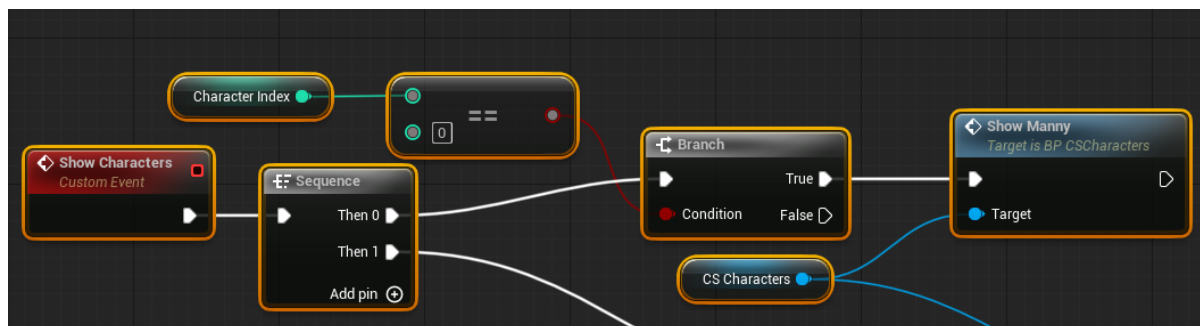


рис. 3.3.3.17.

Для інших персонажів додаємо нові піни в sequence та повторюємо те саме, збільшуючи значення з яким порівнюємо Character index на 1 та викликаючи функцію для відповідного персонажа.

В кнопці Start додаємо івент On Clicked, копіюємо те саме що в Show Characters, така сама логіка, тільки замість Show викликаємо Spawn.

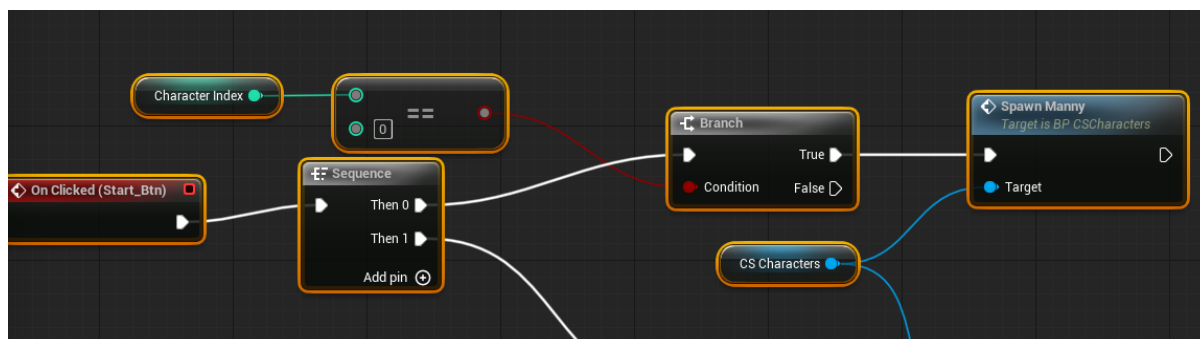


рис. 3.3.3.18.

Для кнопки < (left) додаємо OnClicked. Отримуємо Character Index → декремент → Set Character Index з піну змінної → Select Character → Show Characters.

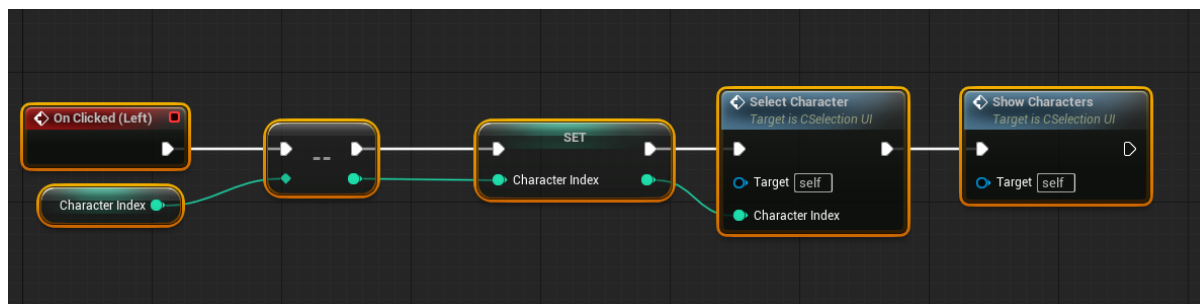


рис. 3.3.3.19.

Для кнопки > (right) додаємо OnClicked → Sequence. Додаємо Get Character Index → інкремент, з'єднуємо з then 0. З повернення змінної → Set Character Index → Select Character → Show Characters.

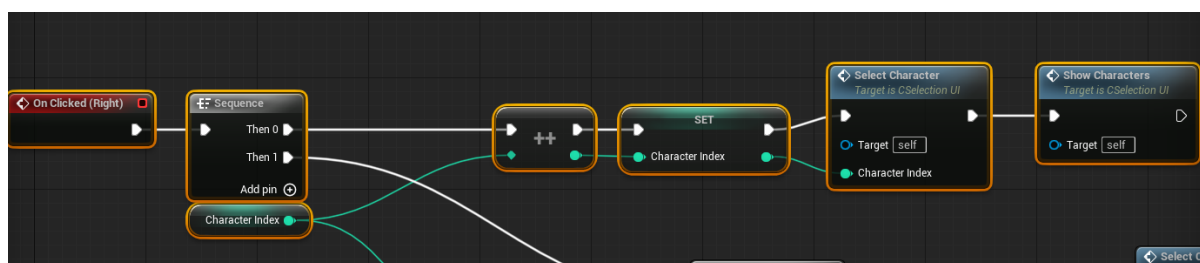


рис. 3.3.3.20.

З Character Index → Greater than or equal (>=) 2, якщо персонажів 2, якщо один то 1 і т.д.. З return value → Branch, True → Set Character Index 0 → Select Character → Show Characters.

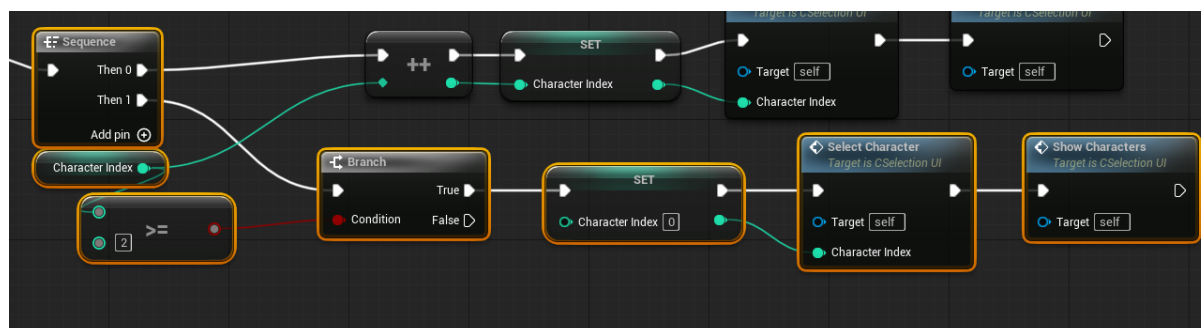


рис. 3.3.3.21.

В мапі, додаємо платформу, CineCameraActor та BP\_CSCharacters.

ставимо камеру та актора щоб актор був посередині в вигляді від камери.

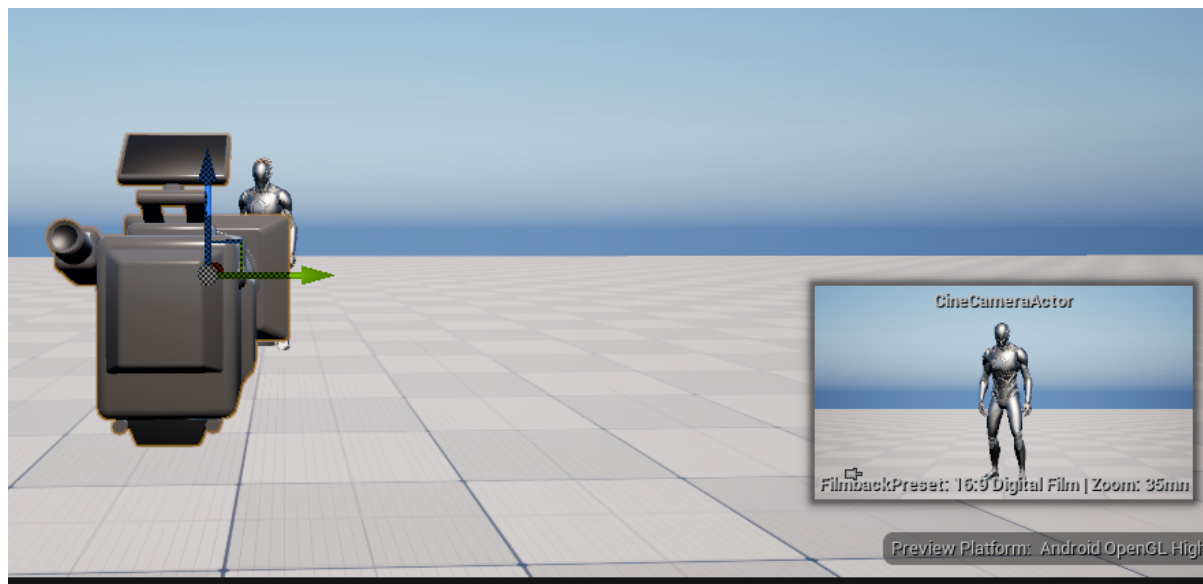


рис. 3.3.3.22.

В Viewport мапи обираємо актора камеру, відкриваємо level blueprint, на пустому місці натискаємо правою кнопкою миші, там з'явиться варіант Create a reference на актора який виділений в Viewport, створюємо посилання на камеру з нього → Set View Target With Blend, додаємо Get Player Controller з'єднуємо з піном Target. Під'єднуємо ехес пін до Begin Play.

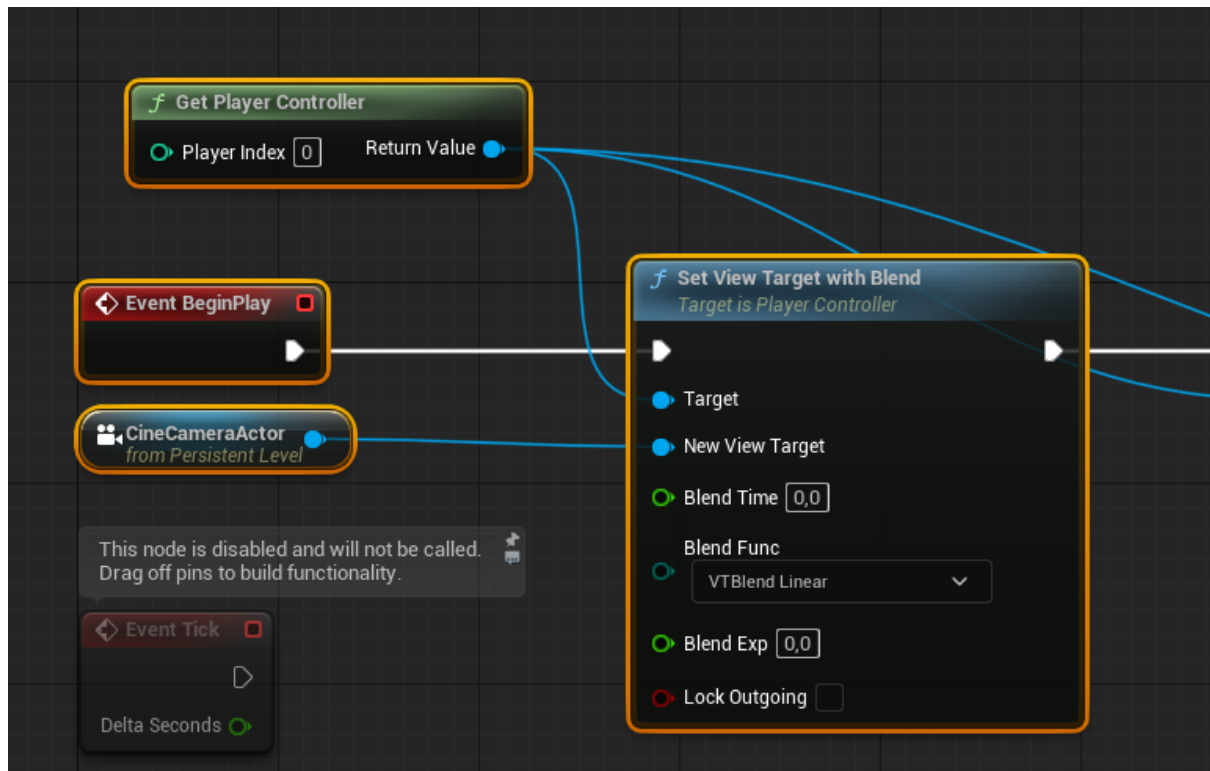


рис. 3.3.3.23.

3 Get Player Controller викликаємо Set input mode UI only, далі з нього же → Set Show Mouse Cursor True → Create Widget, class – CSelection UI, return value → Add To Viewport.

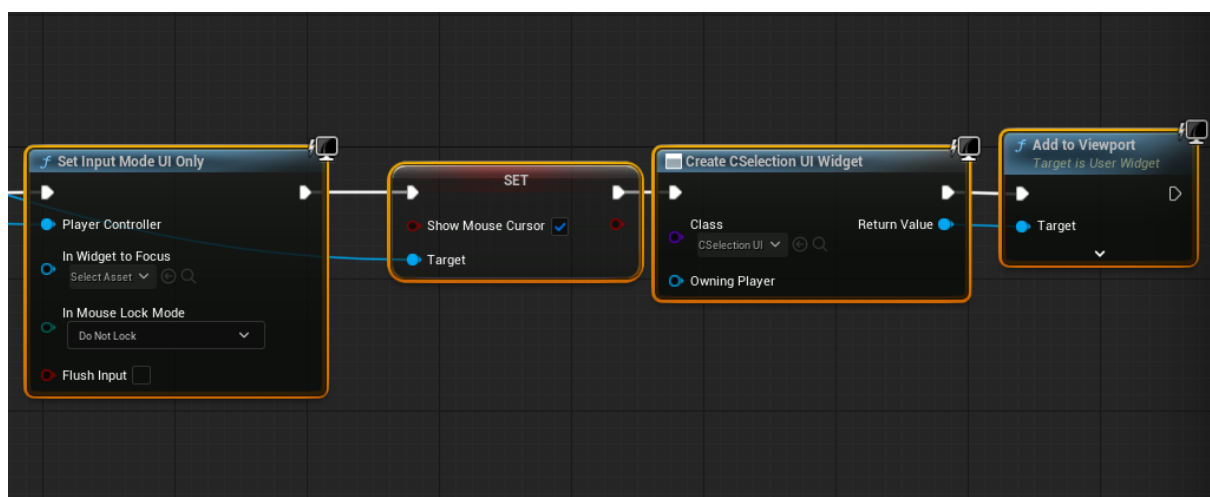


рис. 3.3.3.24.

В мапі Runner\_map, Begin play, отримемо збереженого персонажа зі слоту: додаємо Get Player Controller → Set Input Mode Game Only → Load Game From Slot, SlotName = PlayerCharacter, Return value → Cast To Runner\_SaveGame.

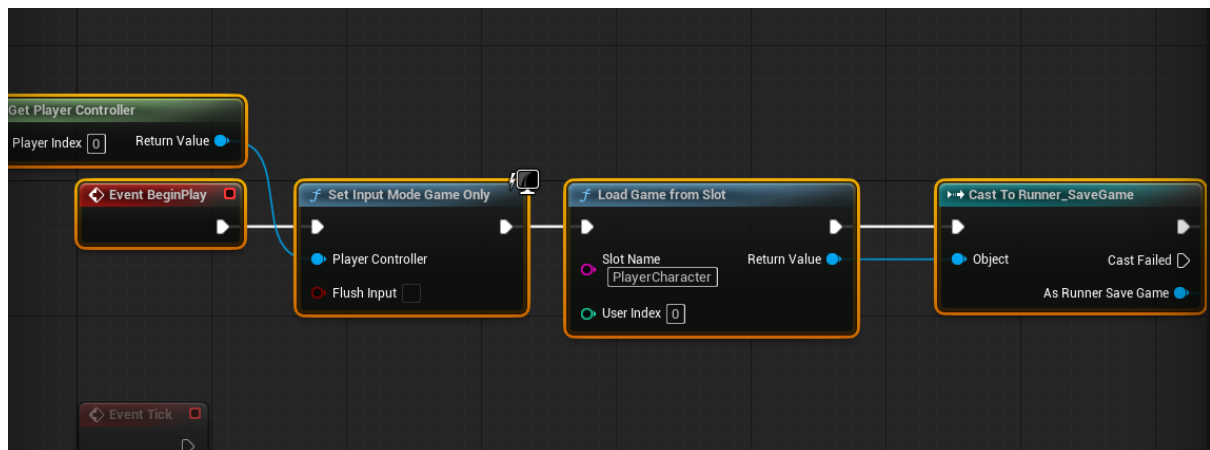


рис. 3.3.3.25.

3 As Runner Save Game отримемо збережену змінну → Get Player Character → Spawn Actor. В Spawn transform → Promote to variable Spawn Transform. В Default Values додаємо значення Location щоб гравець спавнився коректно. 3 return value → Posses. Targer → Get Player Controller.

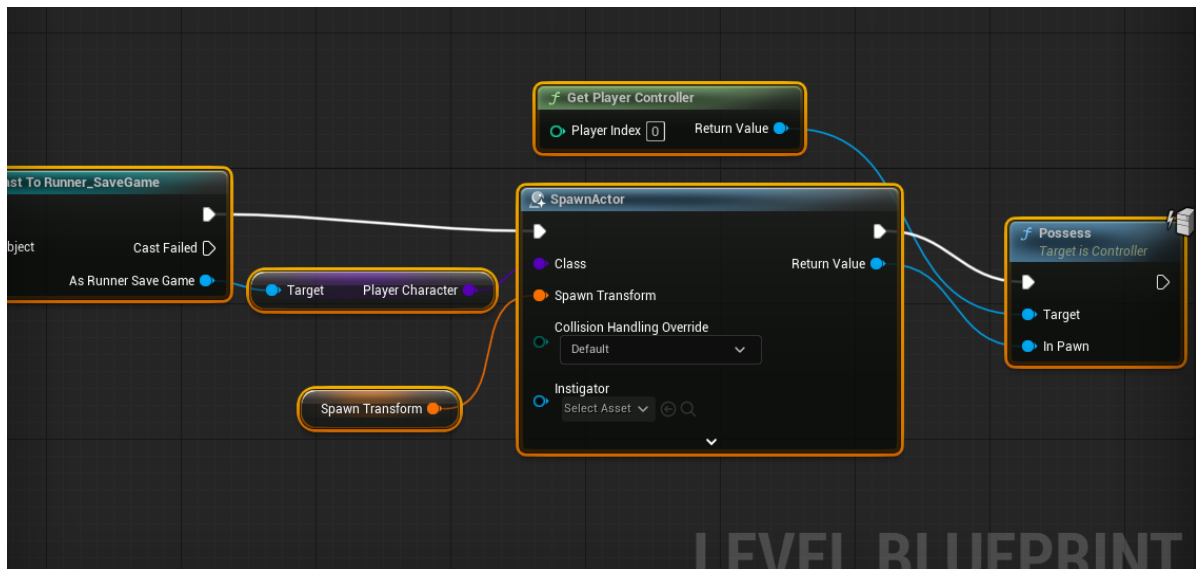
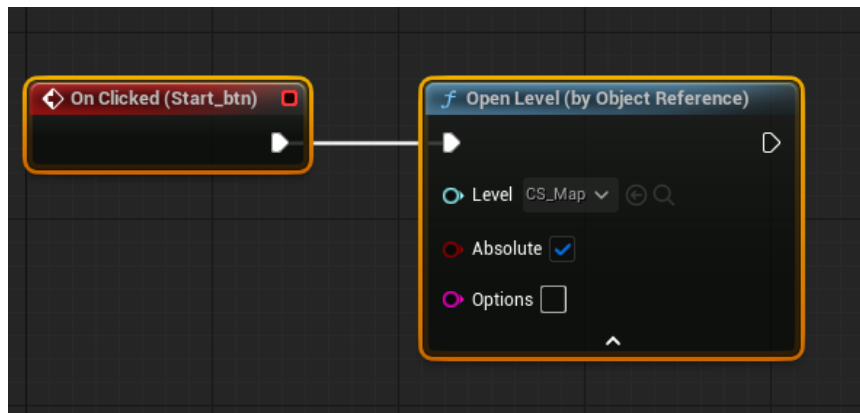


рис. 3.3.3.26.

Тепер в MainMenu\_UI, в OnClicked для Start кнопки  
обираємо Level → CS\_Map.



3.3.3.27.

Тепер щоб запакувати гру в APK натискаємо  
Platforms → Android, обираємо Android (ETC2) Та  
натискаємо Package Project і обираємо папку в яку воно  
збереже файли.

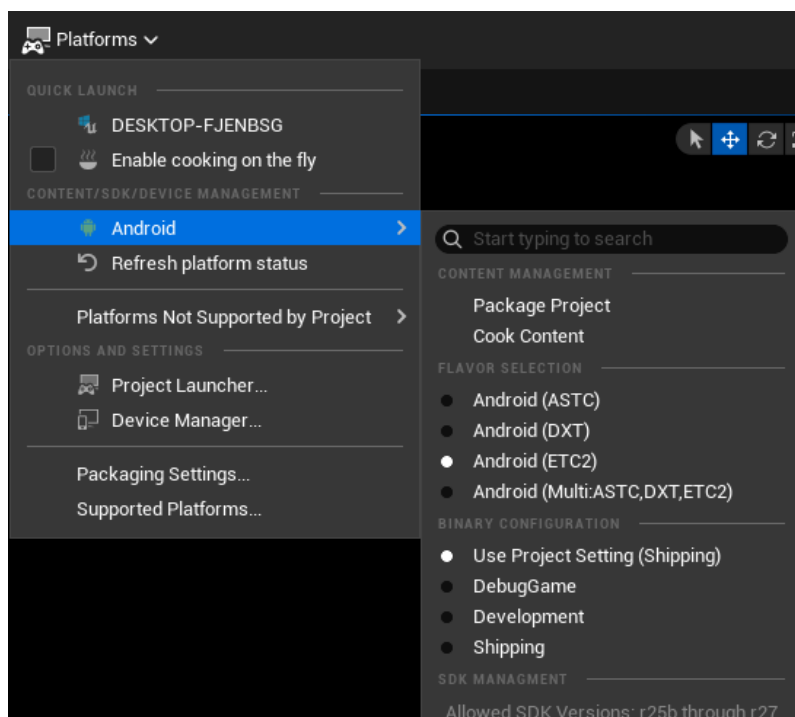


рис. 3.3.3.28.

## ВИСНОВОК

У межах дипломної роботи було розроблено прототип гри жанру endless runner. Реалізовано базовий геймплей: персонаж автоматично рухається вперед, гравець має змогу збирати монети, використовувати підсилення (наприклад, магніт, який автоматично притягує монети), а також змінювати ігрового персонажа.

Додано простий користувацький інтерфейс, що відображає кількість очок та зібраних монет. Також реалізовано можливість поставити гру на паузу та відродитися після поразки.

Гру можна значно покращити, розширивши механіки: додати нові види перешкод, різні типи підсилень, більше варіантів персонажів. Окрему увагу слід приділити заміні безкоштовних асетів на створені власноруч — це підвищить унікальність гри та дасть змогу більш точно оптимізувати 3D моделі та текстури для роботи на мобільному пристрої, порівняно з завантаженими асетами.

Загалом, розроблений прототип демонструє основи геймплею подібних ігор і може слугувати хорошою базою для подальшого розвитку повноцінного проекту.