ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра механіки, автоматизації та інформаційних технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«2-D казуальна стратегія. Бізнес-логіка гри»

«2-D casual strategy. Business logic of the game»

Виконав(ла): здобувач(ка) денної (очної) форми навчання спеціальності 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Кобзар Костянтин Вадимович
(прізвище, ім'я, по-батькові здобувача)

Керівник к.ф.-м.н., доц. Рачинська А.Л. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент викл. Царенко О.П.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
механіки, автоматизації та інформаційних технологій
№ ___ від ____.____. 20___ р.

Завідувачка кафедри
_____ Алла РАЧИНСЬКА
(підпис)

Захищено на засіданні ЕК № _____
Протокол № ___ від ____.____. 20 ___ р.

Оцінка _____/_____/_____

Голова ЕК
_____ Микола МАЛАКСІАНО
(підпис)

Одеса 2024

АНОТАЦІЯ

Дипломна робота присвячена розробці та реалізації 2-D казуальної стратегії на ігровому движку Unreal Engine з використанням Blueprints і плагіна "PaperZD". Основною метою роботи є дослідження можливостей Unreal Engine для створення казуальних ігор, а також розробка повнофункціональної гри з різноманітними механіками та інтерактивними елементами.

У роботі розглядаються основні аспекти розробки гри, включаючи освоєння інтерфейсу та інструментів Unreal Engine, створення анімацій персонажів, розробку інтерфейсу користувача, проектування та реалізацію системи інвентарю, а також механіки будівництва та управління об'єктами. Особлива увага приділяється бізнес-логіці гри, яка реалізована за допомогою візуального програмування на Blueprints.

Гра розповідає про героя-космонавта, котрий досліджує різні планети з метою збору ресурсів і вдосконалення свого космічного корабля. В процесі гри реалізовані такі механіки, як будівництво, інвентар, автозбереження, налаштування гри, пересування різними видами транспорту, боротьба з ворогами та покращення будівель і транспорту.

Результати роботи включають повністю функціональну гру, опис процесу розробки, а також технічну документацію та вихідний код ключових частин проекту.

Дипломна робота містить аналіз досягнутих результатів та рекомендації щодо можливих покращень і подальшого розвитку проекту.

ANNOTATION

The thesis is dedicated to the development and implementation of a 2-D casual strategy game using the Unreal Engine with Blueprints and the "PaperZD" plugin. The main goal of the work is to explore the capabilities of Unreal Engine for creating casual games, as well as to develop a fully functional game with various mechanics and interactive elements.

The work addresses the main aspects of gamedev, including mastering the Unreal Engine interface and tools, creating character animations, developing the user interface, designing and implementing an inventory system, as well as building mechanics and object management. Special attention is given to the game's business logic, which is implemented using visual programming in Blueprints.

The game tells the story of an astronaut hero whoexplores different planets t o gather resources andupgrade his spaceship. The game features mechanics such as construction, inventory, autosave, game settings, transportation, combat with enemies, and improvements to buildings and transport.

The results of the work include a fully functional game, a description of the development process, as well as technical documentation and the source code of keyparts of the project.

The thesis includes an analysis of the achieved results and recommendations for possible improvements and further development of the project.

ЗМІСТ

ВСТУП

У сучасному світі індустрія відеоігор стрімко розвивається, стаючи однією з найвпливовіших галузей розваг. Ігри різних жанрів, зокрема казуальні стратегії, приваблюють мільйони гравців завдяки своїй доступності, цікавим ігровим механікам та інтерактивності. Казуальні стратегії особливо популярні завдяки своїй простоті та захопливості, що робить їх привабливими для широкої аудиторії.

Розробка 2-D казуальної стратегії на базі Unreal Engine з використанням Blueprints і плагіна "PaperZD" є актуальною темою, оскільки дозволяє дослідити можливості цих інструментів для створення якісних ігрових продуктів. Unreal Engine є одним з найпотужніших ігрових движків, що надає розробникам широкі можливості для реалізації своїх ідей, а використання Blueprints дозволяє спростити процес програмування за рахунок візуального кодування.

Гра розповідає про героя-космонавта, якому доручено досліджувати віддалені куточки всесвіту. Його керівництво сподівається таким чином позбутися його, однак герой не здається і впевнено рухається вперед.

Головна мета гри полягає в дослідженні та освоєнні кожної планети для подальшого вдосконалення свого космічного корабля. У процесі розвитку на кожній планеті герой отримує все більше ресурсів для покращення як свого корабля, так і власних можливостей у галактичних дослідженнях. Кінцевою метою гри є досягнення центру галактики та подальший прохід у чорну діру.

Основні механіки гри включають:

- Будівництво: створення різноманітних об'єктів і будівель для покращення бази та корабля.
- Інвентар: управління зібраними ресурсами і предметами.
- Автозбереження: автоматичне збереження прогресу гравця.
- Налаштування гри: можливість змінювати налаштування гри.

- Переміщення: різні способи переміщення, включаючи машини, поїзди, космічні кораблі та пішохідний хід.

- Атмосфера та тиск: вплив атмосферних умов на ігровий процес.

- Вороги: наявність противників, з якими потрібно боротися.

- Покращення будівель та транспорту: можливість покращувати створені об'єкти і транспортні засоби.

ПОСТАНОВКА ЗАДАЧІ

Основною метою даного дипломного проекту є розробка 2-D казуальної стратегії.

Бізнес-логіка гри.

Ця система повинна забезпечувати управління ігровими даними, створення та управління ігровими елементами, а також аутентифікацію користувачів для підвищення рівня інтерактивності та захисту даних у грі.

Для досягнення ключових результатів необхідно вирішити наступні підзадачі:

а) Порівняти існуючі методи та програмні продукти для розробки казуальних стратегій:

1) Провести детальний аналіз сучасних ігрових систем та інструментів для розробки 2-D казуальних стратегій;
2) Виявити їхні переваги та недоліки, щоб визначити оптимальний підхід для розробки нашої гри.

б) Проаналізувати та обґрунтувати вибір технологій, необхідних для реалізації гри:

1) Обрати найбільш відповідні технології, які забезпечують високу продуктивність, надійність та гнучкість системи;
2) Це включає вибір мов програмування, ігрових рушіїв та баз даних.

в) Розробити архітектуру гри, що забезпечує її надійність, масштабованість та гнучкість:

1) Спроектувати систему з урахуванням потреб користувачів;
2) Визначити основні компоненти та їх взаємодію;
3) Забезпечити можливість легкої інтеграції з іншими ігровими системами та сервісами.

г) Спроектувати та реалізувати ігровий додаток для управління ігровими даними та створення ігрових елементів:

1) Створити веб-додаток або стаціонарну версію з інтуїтивно зрозумілим інтерфейсом;
2) Додаток дасть змогу легко додавати, редагувати та видаляти ігрові дані;
3) Генерувати та управляти ігровими елементами, такими як персонажі, ресурси та інші об'єкти.

Здобуття теоретичних та практичних навичок

Ще одним важливим аспектом у досягненні мети є здобуття теоретичних та практичних навичок проектування та реалізації казуальних стратегій, що включає:

а) Вивчення сучасних технологій та методів забезпечення інтерактивності та безпеки в іграх;

б) Практичне застосування цих знань у розробці функціональних модулів гри.

Комплексний підхід до розробки

Таким чином, реалізація даного проекту передбачає комплексний підхід до розробки 2-D казуальної стратегії, що забезпечить:

а) Високий рівень інтерактивності та безпеки;

б) Зручність у використанні;

в) Можливість подальшого розвитку та адаптації гри під специфічні потреби користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

Область відеоігор є широкою та розмаїтою та охоплює такі процеси як - проектування, розробка, тестування, тощо.

Один з важливих аспектів створення відеоігор - це їх концептуалізація. Одне з головних завдань для розробників створювати ідеї для гри, котрі будуть цікавими і захоплюючими для гравців. Концептуалізація включає в себе - визначення жанру гри, написання історії, створення персонажів, проектування ігрового світу та ігрового процесу.

Наступний крок після концептуалізації - це фаза розробки, до якої входять малювання графічних елементів, написання звуків та музики, написання коду гри та її тестування.

1.1 Класифікація комп'ютерних ігор

Це міждисциплінарна категорія, яка вивчається в рамках філософії, культурології та естетики. Активно застосовується підхід, котрий ґрунтується на системі правил, за аналогією класифікації фільмів.

Категоріювання За Кроуфордом

«Мистецтво проектування комп'ютерних ігор» - найбільш рання робота з класифікації «електронних ігор» була проведена геймдизайнером Крісом Кроуфордом у 1984 році. Результатом проведеної роботи став розподіл ігор на два етапи. На першому етапі гри поділялися на дві категорії. Розподіл наведено у табл. 1.1. [14]

Таблиця 1.1. - Найбільш рання класифікація комп'ютерних ігор

Категорії	Синоніми	Опис
Skill-and-action games	S&A games, Екшн ігри	Акцентує увагу на швидкості та точності рухів
Strategy	Cognitive games, Стратегії	Акцентує увагу на когнітивній активності

За характером ігрової методики ігри можна поділити на:

-Предметні;

-Сюжетні;

-Рольові;

-Ділові;

-Імітаційні;

-Ігри-драматизації.

1.2 Аналіз засобів розробки

Наразі відбуваються процеси удосконалення декількох основних ігрових рушіїв, які використовуються під час розробки відеоігор.

Розвиток нової гілки фізики ігрових рушіїв під назвою HAVOK дозволяє використовувати високополігональну модель матерії, яка покриває будь-який об'єкт, а після – руйнує його в залежності від програмованого алгоритму.

Вибухи, бризки, світлові промені – це все FX-ефекти, їх якість та видовищність безпосередньо залежить від ігрового рушію, тому наразі один

з пріоритетів розробки ігрових рушіїв це системи візуалізації частинок, що допоможуть покращити якість ігрових програм.

Одним з перших етапів розробки комп'ютерної гри є вибір ігрового рушію. Для вибору засобу розробки проведемо аналіз декількох популярних ігрових рушіїв за основними параметрами такими як: продуктивність, якість графіки, якість звуку. Аналіз продуктивності приведено в табл. 1.2. [11, 12]

Таблиця 1.2. Аналіз ігрових рушіїв

Рушій	Швидкість фізичної симуляції	Швидкість обробки гри	Швидкість завантаження гри	Швидкість рендерингу
Unreal Engine	Висока	Висока	Середня	Висока
Unity	Висока	Висока	Середня	Висока
GameMaker	Середня	Середня	Висока	Середня
Godot	Висока	Висока	Середня	Висока
RPG Maker M	Середня	Середня	Висока	Середня
Buildbox	Низька	Середня	Висока	Середня

Кожен ігровий рушій має свої плюси та мінуси. До прикладу, найбільш популярні на сьогоднішній день рушії, це Unity та Godot через їх простоту в використанні. Але якщо брати сучасні технології та швидкість їх розвитку, то можна розглядати такі рушії як: Unreal Engine та GameMaker.

Далі розглянемо якість звукоефектів. Аналіз приведено в табл. 1.3. [12, 13]

Таблиця 1.3. Порівняння якості звукових ефектів

Рушій	Якість звукових ефектів	Якість діалогів та голосів персонажів	Якість музики	Простірний звук	Підтримка 3D-звуку
Unreal Engine	Висока	Висока	Висока	Так	Так
Unity	Висока	Висока	Висока	Так	Так
GameMaker	Середня	Середня	Висока	Ні	Ні
Godot	Висока	Висока	Висока	Так	Так
RPG Maker MZ	Середня	Середня	Середня	Ні	Ні
Buildbox	Середня	Середня	Середня	Ні	Ні

1.3 Загальний алгоритм реалізації

1.3.1 Опис процесу тестування та налагодження гри

Процес тестування та налагодження є ключовим етапом розробки гри, який дозволяє виявити та виправити помилки, забезпечити стабільну роботу та покращити загальну якість гри.

Основні кроки процесу тестування та налагодження:

а) Планування тестування:

1) Визначаємо основні цілі та завдання тестування.

2) Розробляємо план тестування, який включає різні типи тестів (функціональні, регресійні, стресові, тощо).

б) Розробка тестових сценаріїв:

1) Створюємо детальні тестові сценарії, які охоплюють усі основні аспекти гри (ігровий процес, інтерфейс, продуктивність, сумісність).

2) Використовуємо автоматизовані тести, де це можливо, для зменшення часу на тестування.

в) Виконання тестування:

1) Залучаємо команду тестувальників для проведення тестування гри.

2) Використовуємо інструменти для відстеження помилок (наприклад, Jira, Trello) для документування виявлених проблем.

г) Налагодження гри:

1) Аналізуємо виявлені помилки та розробляємо план їх виправлення.

2) Використовуємо дебагінг інструменти Unreal Engine для виявлення та виправлення помилок у коді та логіці гри.

1.3.2 Призначення гри

Двовимірна (2D — від англ. two dimensions — «два виміри») комп'ютерна графіка класифікується на кшталт подання графічної інформації, і з алгоритмами обробки зображень. Зазвичай комп'ютерну графіку поділяють на векторну та растрову, хоча відокремлюють ще й фрактальний тип подання зображень.

Комп'ютерні ігри завоювали світ, надаючи мільйонам гравців можливість відчути себе героями неймовірних пригод, стратегами, спортсменами та навіть богами у віртуальних світах. Ці ігри створюють незабутні відчуття, вони здатні перенести нас у захопливий світ історій та непередбачуваних пригод.

1.3.3 Обладнання для реалізації ігрового контенту

Вибір правильного обладнання може кардинально змінити ігровий досвід. Від швидкодії процесора до якості зображення на екрані кожен компонент має значення.

Вибір між геймерським ноутбуком та настільним ПК – це вибір між мобільністю та потужністю. Геймерський ноутбук забезпечує портативність, що є ідеальним для тих, хто часто переміщається. Однак, у плані продуктивності та можливостей апгрейду, настільний ПК часто перевершує ноутбуки. Найважливішими компонентами є процесор та відеокарта.

Вибір монітора відіграє ключову роль визначенні візуального досвіду в іграх. Висока роздільна здатність екрана забезпечує чіткість і деталізацію зображення, що особливо важливо в сучасних іграх з високоякісною графікою. Частота оновлення екрана також є критичним фактором — чим вища частота, тим плавнішими та реалістичними будуть рухи у грі.

Для професійних геймерів часто рекомендується монітор з частотою оновлення щонайменше 144 Гц. Крім того, варто враховувати розмір екрану. Великі монітори забезпечують більш досвід, що занурює, але вимагають відповідного простору на робочому столі. Важливо вибирати монітор, що поєднує в собі оптимальні для вас характеристики роздільної здатності, частоти оновлення та розміру, щоб максимально підвищити якість та комфорт ігрового процесу.

2 ОСВОЄННЯ UNREAL ENGINE 5

Unreal Engine 5 – це один з найпотужніших ігрових рушіїв, що надає розробникам широкий спектр можливостей для створення високоякісних ігрових проектів. Освоєння цього інструменту розпочинається з вивчення його інтерфейсу та основних інструментів. [4, 6]

2.1 Інтерфейс

Інтерфейс Unreal Engine 5 включає в себе декілька основних частин:

а) Панель Level Viewport - головне вікно, де розробники можуть бачити і редагувати свою сцену або рівень та область, яку бачить користувач на екрані, коли заходить на сторінку сайту з будь-якого пристрою. Вікно перегляду рівня загалом може відображати вміст рівня двома способами:

1) Перспектива - є тривимірним виглядом, за яким можна переміщатися, щоб побачити вміст вікна перегляду з різних кутів.
2) Ортографічне зображення - є двовимірним виглядом, який дивиться вниз по одній із головних осей (X, Y або Z).

б) Панель Details містить інформацію, утиліти та функції специфічні для вибраного об'єкта у панелі Viewport.

в) Панель Content Drawer та Content Browser – область для управління усіма файлами проекту, включаючи текстури, моделі, матеріали та інші ресурси, також необхідна для створення, імпорту, сортування та перегляду асетів у Unreal Engine.

г) Панель World Outliner - це ієрархія всіх елементів на рівні, яка надає інструменти для вибору, пошуку, видалення, батьківського контролю, групування та відображення/приховування елементів.

ґ) Details Panel - панель, що показує властивості вибраного об'єкту та дозволяє їх редагувати.

д) Blueprint Editor - інструмент для візуального програмування ігрової логіки.

е) Панель Modes - це інструмент, що спрощує процедуру створення оточення в Unreal Engine 5. Подібно до контенту браузеру Place Mode фокусується на тих асетах (тобто об'єктах), які можна розмістити на вашому рівні. Коли вікно активно, ви отримуєте швидкий доступ до всіх розміщених об'єктів, які існують у вашому проекті, без необхідності переміщатися папками як ви б робили в контент браузері.

2.2 Редактор

Редактор рівнів Unreal Engine 5 - це основний редактор, у якому ми створюємо рівні гри. Тут визначається ігровий простір для гри, додаються різні типи акторів, об'єктів та геометрії, візуальні сценарії Blueprints, Niagara, тощо. При створенні або відкритті проекту редактор рівнів відкривається за замовчуванням.

Контент браузер Unreal Engine 5 дозволяє керувати асетами проекту, такими як моделі, текстури, анімації та звуки. Крім основних функцій, Unreal Engine 5 також пропонує низку просунутих можливостей, таких як система Niagara для створення складних ефектів частинок.

Редактор Niagara Editor призначений для створення спеціальних ефектів шляхом використання повністю модульної системи ефектів частинок, що складається з окремих випромінювачів частинок для кожного ефекту. Випромінювачі можна зберегти в Content Browser для подальшого використання.

Unreal Engine 5 надає ряд інструментів та функцій для оптимізації продуктивності, таких як система LOD, оптимізація освітлення та матеріалів, а також налаштування параметрів проекту.

Blueprint – інструмент який використовується для створення елементів ігрового процесу (наприклад, керування актором або створення сценарію події), модифікації матеріалів або реалізації інших функцій Unreal Engine без необхідності писати код C++.

Material Editor - відповідає за зовнішній вигляд та властивості поверхні всіх об'єктів сцени, таких як Static Mesh (стандартні об'єкти зі статичним типом геометрії), Skeletal Mesh (об'єкти зі скелетною анімацією), ландшафт, інтерфейс користувача та візуальні ефекти.

Static Mesh - використовується для попереднього перегляду вигляду, колізій та УФ-відображення, а також налаштовуються властивості статичних сіток та LOD.

2.3 Основні компоненти

Структури Даних (Data Structures)

Для зберігання інформації про предмети інвентарю використовуються структури даних, такі як масиви або словники. Це дозволяє організувати предмети та отримувати до них швидкий доступ.

Класи Предметів (Item Classes)

Кожен предмет інвентарю зазвичай представлений окремим класом, що зберігає інформацію про властивості предмета, такі як назва, опис, іконка, кількість тощо.

Віджети (Widgets)

Для відображення інвентарю на екрані використовуються віджети UMG. Це можуть бути списки предметів, іконки, кількість предметів та інші елементи.

Системи Управління (Management Systems)

Система управління інвентарем відповідає за додавання, видалення, сортування та використання предметів. Вона також може включати функції обміну між персонажами або зберігання предметів у сховищах.

Інтерфейс користувача (скорочено ІК), (англ. user interface, UI) — засіб зручної взаємодії користувача (людини) з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача. У графічних системах інтерфейс користувача втілюється багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їх розташуванням, сортуванням елементів вікон, гнучкими налагодженнями як самих вікон, так і окремих їх елементів (файли, папки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

Інтерфейс користувача, в галузі промислового дизайну, взаємодії між людиною та комп'ютером - простір, де відбувається співпраця між людьми та машинами. Мета цієї взаємодії полягає у забезпеченні досконалої роботи та керуванні машиною з боку людини, а машина одночасно надає інформацію, яка допомагає ухваленню рішень операторами.

Прикладами цієї широкої концепції інтерфейсів користувача, є:

- інтерактивні можливості комп'ютерних операційних систем;
- ручні інструменти;
- операторські засоби керування важким обладнання;
- та елементи контролю над технологічними процесами.

Конструктивні міркування, що застосовуються під час створення інтерфейсів користувача, пов'язано з такими галузями, як ергономіка, когнітивні науки та психологія.

Здебільшого, мета створення інтерфейсу користувача полягає у тому, щоб зробити інтерфейс ефективним для керування машиною, задля забезпечення бажаного результату.

Це означає, що оператор повинен застосовувати щонайменші зусилля для досягнення очікуваного результати, а також, аби машина зменшувала небажані наслідки для людини. З посиленням використання персональних комп'ютерів та відносним зниженням обізнаності суспільства про важкі машини, термін «інтерфейс користувача», здебільшого, передбачає графічний інтерфейс користувача, тоді як промислові панелі керування та механізми контролю за обладнанням, частіше стосуються людино-машинної взаємодії.

Інші терміни для інтерфейсу користувача — це інтерфейс «людина-машина» (MMI), коли відповідним пристроєм, є комп'ютерний інтерфейс «людина-комп'ютер».

2.4 Розробка, виконання та оптимізація тестових сценаріїв для перевірки функціоналу гри

Тестові сценарії є ключовим елементом процесу тестування, що дозволяють систематично перевіряти різні аспекти гри.

Основні кроки для розробки та виконання тестових сценаріїв:

а) Розробка тестових сценаріїв:

1) Визначаємо основні функціональні області гри, які потребують тестування (наприклад, бойова система, система інвентарю, механіка руху).
2) Створюємо детальні сценарії, що описують конкретні дії гравця та очікувані результати.

б) Виконання тестових сценаріїв:

1) Тестувальники виконують тестові сценарії, фіксуючи результати та виявлені проблеми.
2) Використовуємо автоматизовані тести для перевірки повторюваних або складних сценаріїв.

в) Аналіз результатів тестування:

1) Аналізуємо результати тестових сценаріїв для виявлення загальних проблем або патернів помилок.
2) Пріоретизуємо виправлення помилок на основі їх впливу на ігровий процес та стабільність гри.

Оптимізація проекту є важливим етапом для забезпечення плавної роботи гри на різних пристроях та платформам.

Основні кроки для оптимізації проекту:

а) Оптимізація коду:

1) Аналізуємо код гри для виявлення неефективних або надлишкових частин.
2) Оптимізуємо алгоритми та структури даних для зменшення навантаження на процесор та пам'ять.

б) Оптимізація ресурсів:

1) Зменшуємо розмір текстур, моделей та інших ресурсів без значної втрати якості.
2) Використовуємо рівні деталізації (LOD) для моделей, щоб зменшити навантаження на графічний процесор на великій відстані.

в) Оптимізація логіки гри:

1) Оптимізуємо логіку ігрових механік для зменшення кількості обчислень у реальному часі.
2) Використовуємо асинхронні процеси та багатопоточність для покращення продуктивності.

г) Тестування продуктивності:

1) Проводимо тестування продуктивності на різних пристроях для виявлення потенційних проблем.
2) Використовуємо інструменти профілювання Unreal Engine для аналізу продуктивності та оптимізації критичних частин гри.

3. ПРОЦЕС РОЗРОБКИ ГРИ

3.1 Бізнес-логіка гри

Встановлення Unreal Engine 5 та його запуск в подальшому відбувається через Epic Games Launcher. Перед початком розробки 2D стратегії завантажуємо PaperZD з магазину додатка. (рис. 3.1). Це плагін який дозволить робити 2D анімацію для нашого персонажа та все що з ним пов'язано. Плагін є безкоштовним. [5]

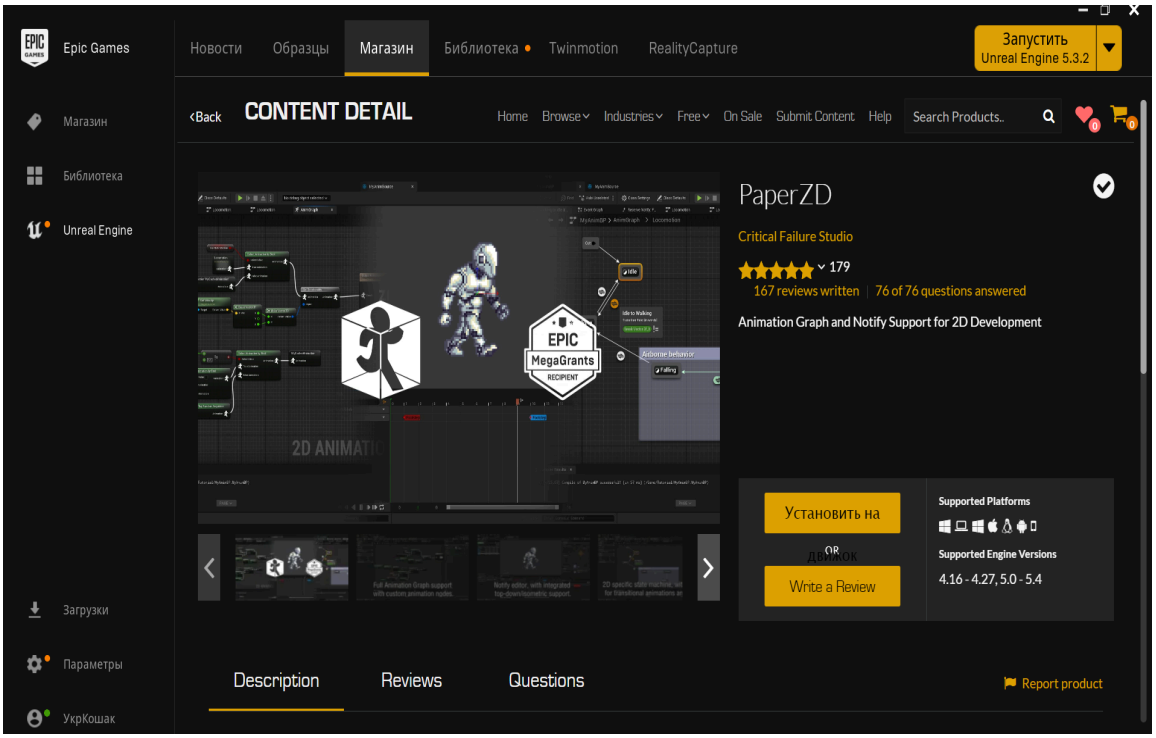

Рисунок 3.1 - Завантаження PaperZD

3.2 Створення проекту

Для початку процесу розробки треба обрати тип проекту. Переходимо до вкладки “Games” (рис. 3.2).

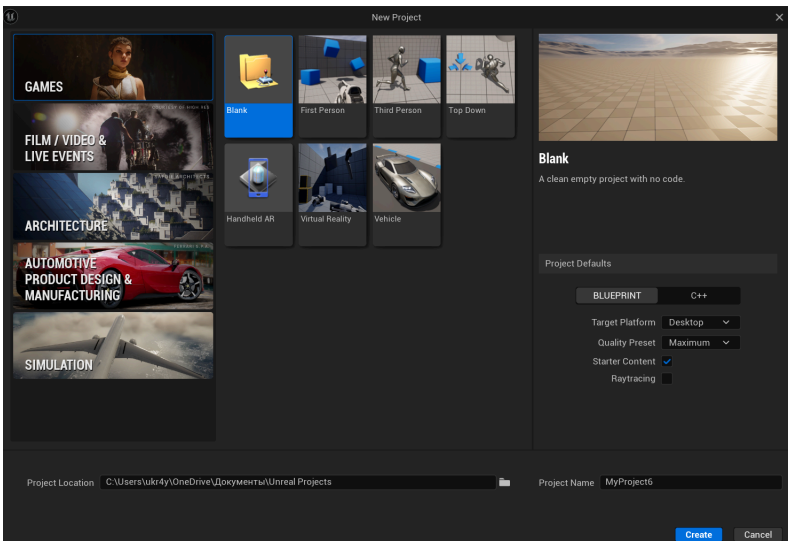

Рисунок 3.2. - Вибір проекту

Далі відразу обираємо шаблон нашої гри (рис. 3.3). Обираємо Blank. Цей шаблон пустий, але нам не потрібно буде шукати зайве.

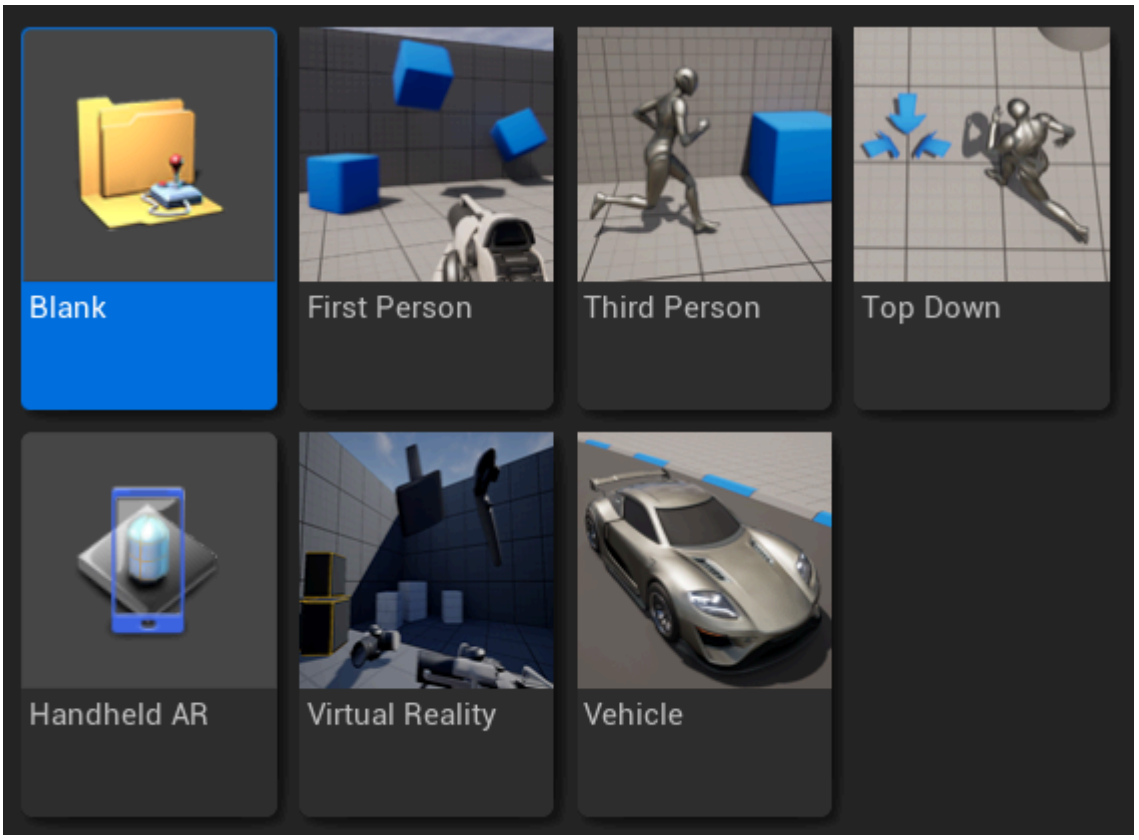

Рисунок 3.3 Вибір пустого шаблону

Виставляємо основні налаштування проекту (рис. 3.4). Вказуємо назву проекту, та обираємо Blueprints, проект розробляється для персональних комп’ютерів, додатково вмикаємо параметр Starter Content

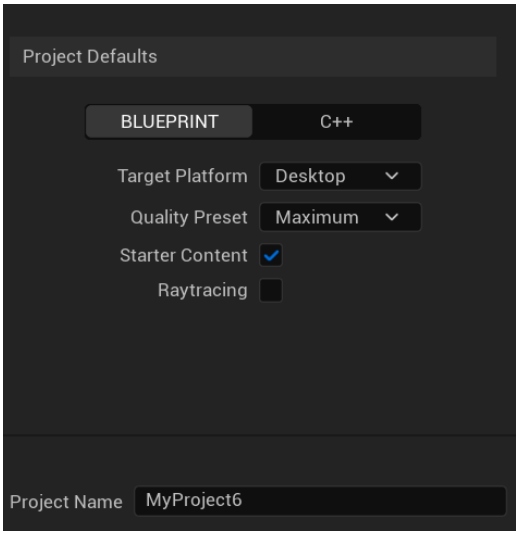

Рисунок 3.4. - Налаштування проекту

Після створення проекту ми бачимо пусту сцену з відкритим редактором рівня, на якому можемо виставляти наші об’єкти (рис. 3.5)

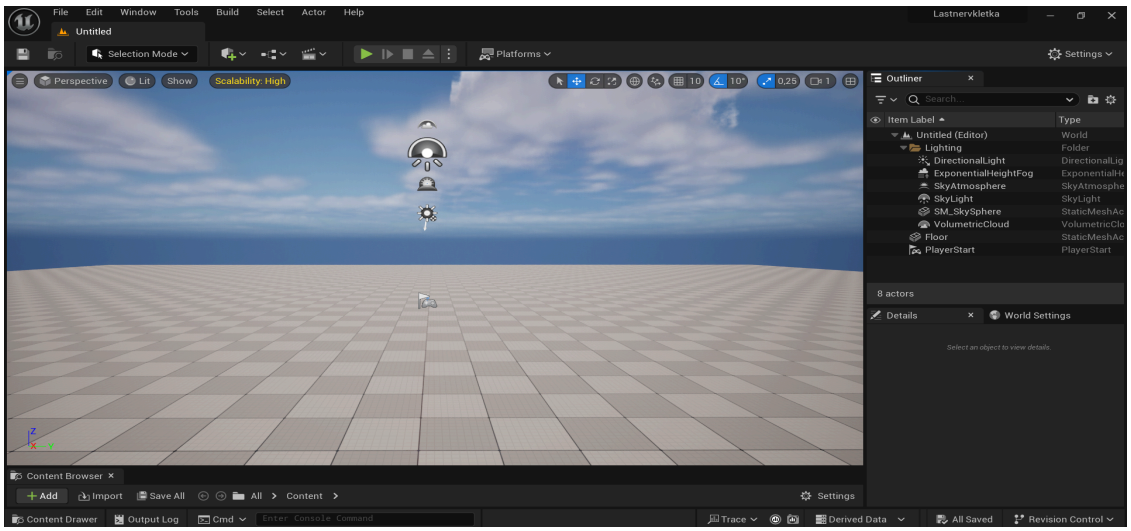

Рисунок 3.5. - Основне вікно Unreal engine

3.3 Створення персонажу

Так як гра в 2D світі, нам потрібно створити не об’ємного персонажа.

Для створення та використання нашого персонажа, нам потрібно завантажити ігрові асети створені в CAD. Створюємо папки “character” – “People”, та розподіляємо наші асети по наступним папкам (рис. 3.6) [8]:

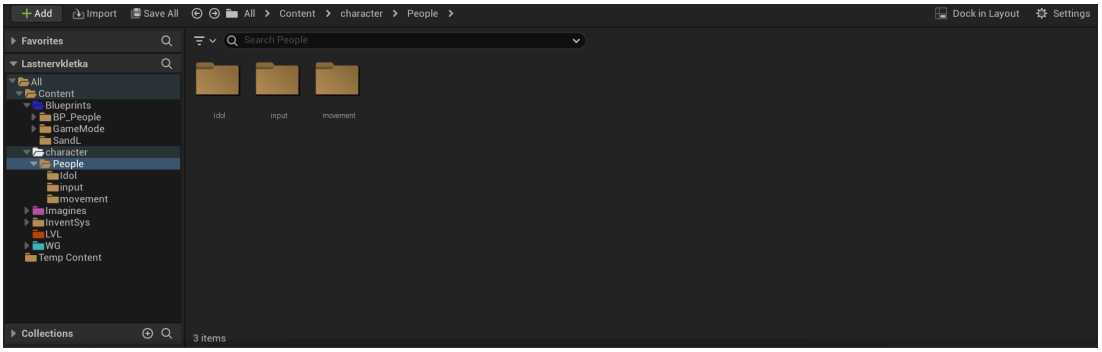

Рисунок 3.6. - Файловий менеджер з папками персонажу

Idle – папка з анімаціями простою (рис. 3.7)

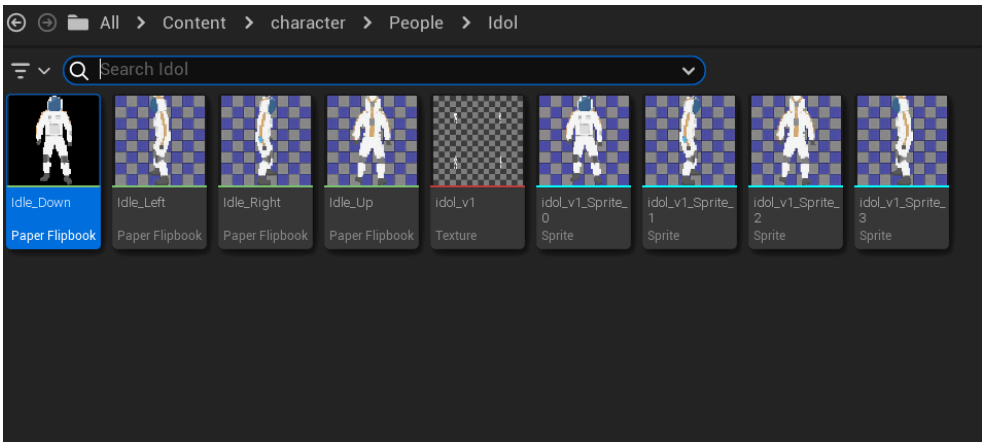

Рисунок 3.7. - Анімації спокою персонажа

Movement – папка з анімаціями руху (рис. 3.8)

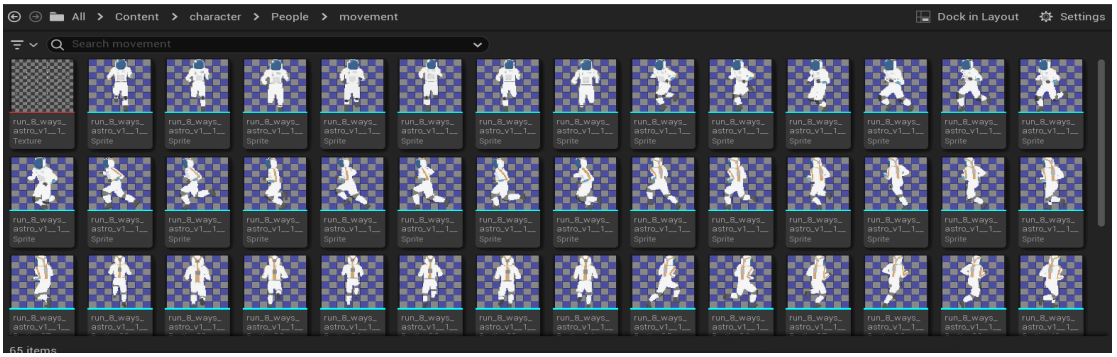

Рисунок 3.8. - Анімації переміщення персонажа

Після підготовки усіх асетів додаємо папку Input та в ній створюємо Input Action під назвою Move_Ia та TD_Move (рис. 3.9).

Ці файли будуть зберігати в собі налаштування кнопок та вектор руху для переміщення персонажу (рис. 3.10) .

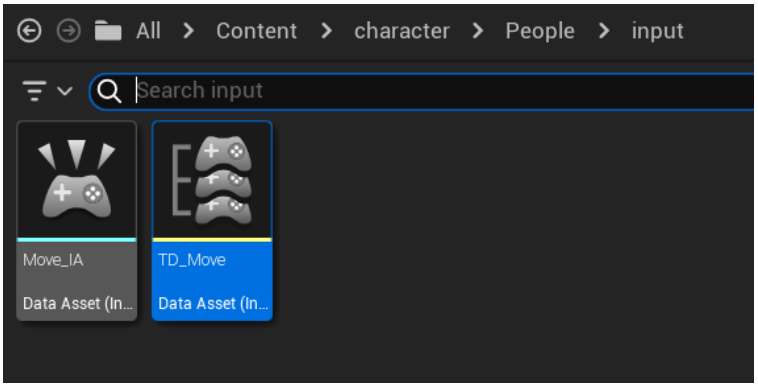

Рисунок 3.9. - Файли управління персонажем

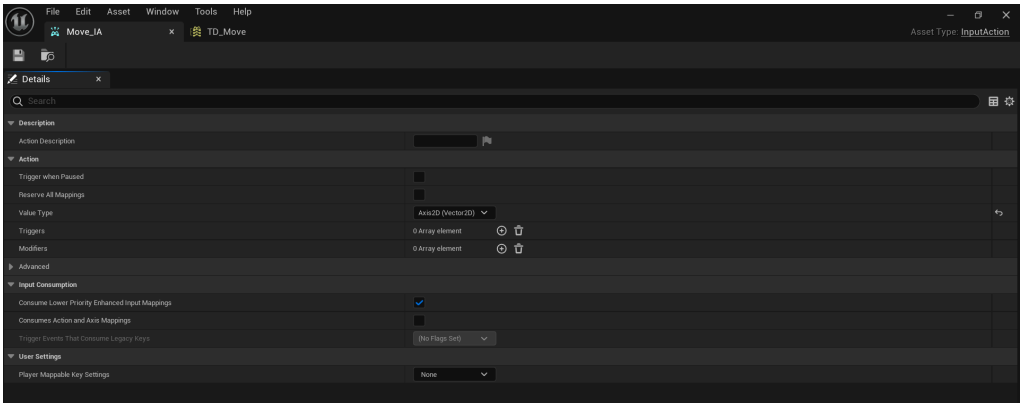

Рисунок 3.10.1 - файл прив’язки до 2D

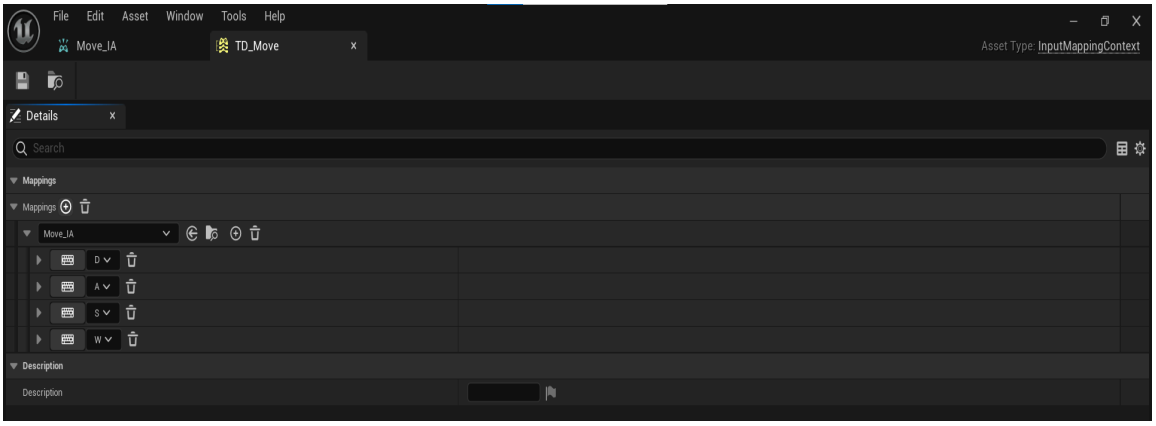

Рисунок 3.10.2 - Файл прив’язки до клавіш

Використовуючи асети персонажа та плагін PaperZD створюємо FlipBook, який буде створювати анімацію з декількох зображень космонавта (рис. 3.11).

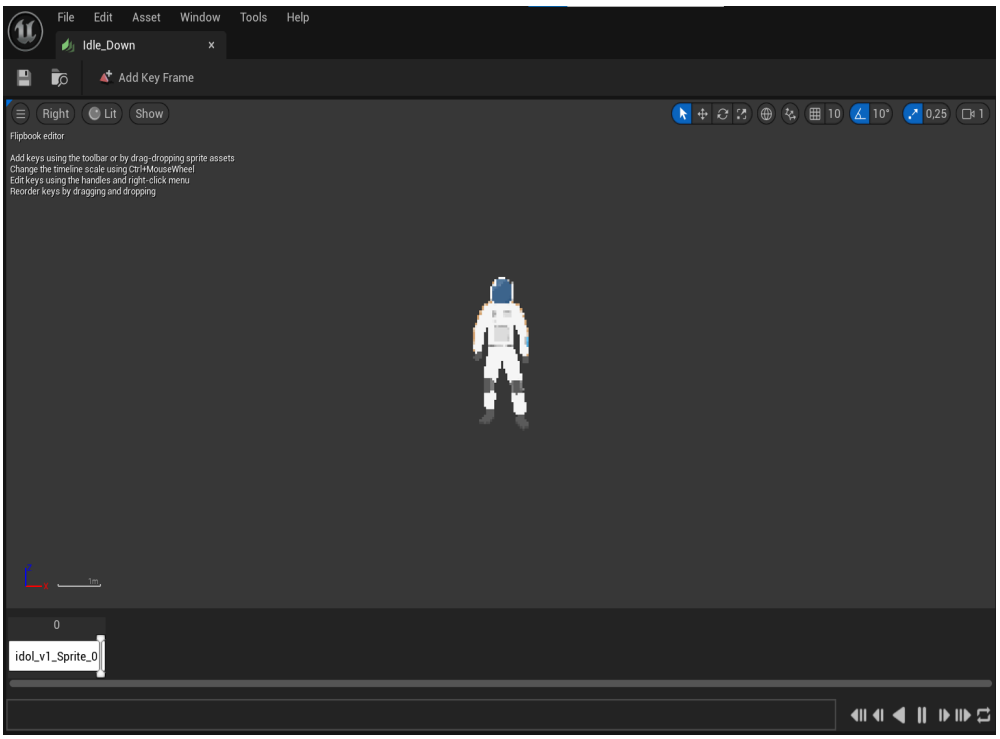

Рисунок 3.11. - FlipBook з IDLE анімацією

Завдяки наборам Flipbook, ми маємо змогу створити анімацію та запрограмувати її відтворення на потрібну нам клавішу (рис. 3.12).

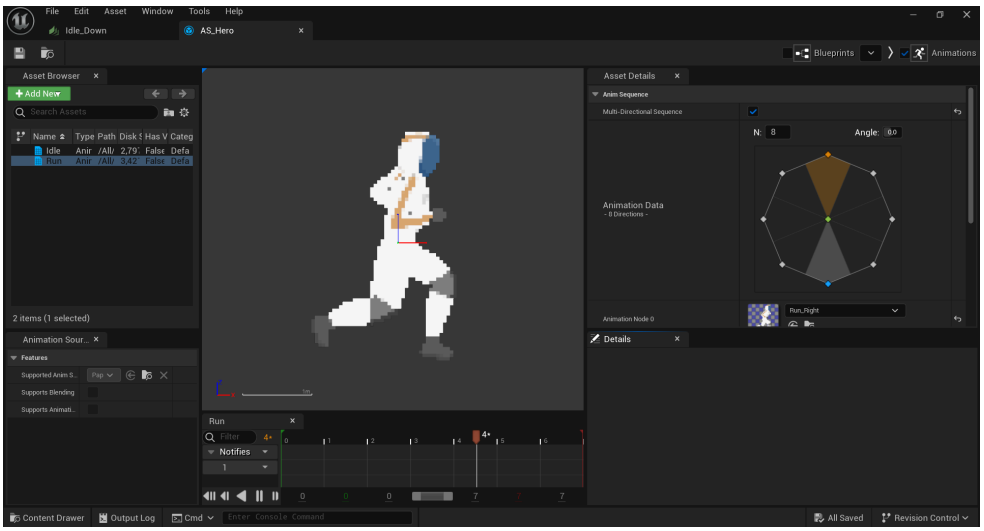

Рисунок 3.12 - Набір анімацій для відтворення

Для того, щоб ми могли користуватися та перемикатися між нашими Flispbook безпосередньо з гри, нам потрібно створити Blueprint, який завдяки прив’язці до AnimSequences, буде самостійно сповіщати гру який саме Flipbook треба зараз відтворити. Це можливо завдяки нашим Blueprints (рис. 3.13).

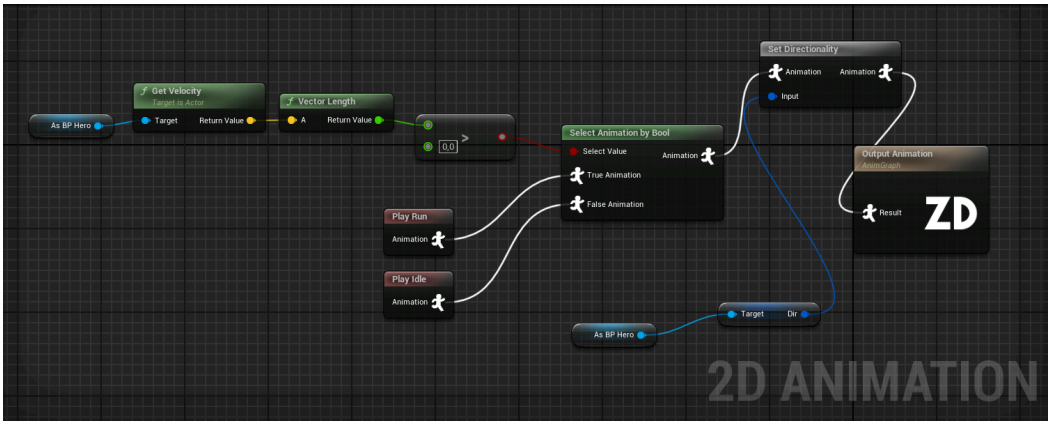

Рисунок 3.13 - Візуальний код зміни анімацій

Додатково нам знадобиться прив’язати до нашого персонажу камеру, задля того, щоб при переміщенні користувач слідкував саме за героєм. Для цього ми створимо ще один Blueprint котрий буде основою нашого

персонажа. Саме через цей Blueprint буде йти весь контроль грою (рис. 3.14).

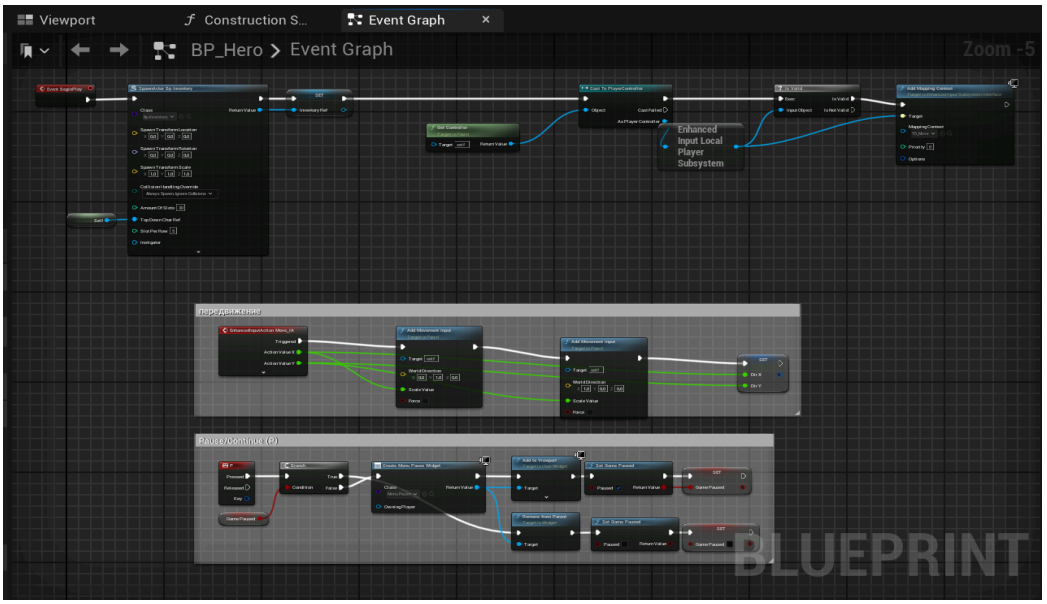

Рисунок 3.14 - Основний Blueprint управління персонажем та грою

Для того щоб прив’язати камеру нам знадобиться перейти до ViewPort та додати компонент Capsule, він буде відповідати за зіткнення з об’єктами та не дасть персонажу проходити крізь них. Після Capsule треба додати SpringArm, який буде виступати в ролі повідця між нашою камерою та персонажем. І вже на SpringArm ми додаємо нашу камеру (рис. 3.15).

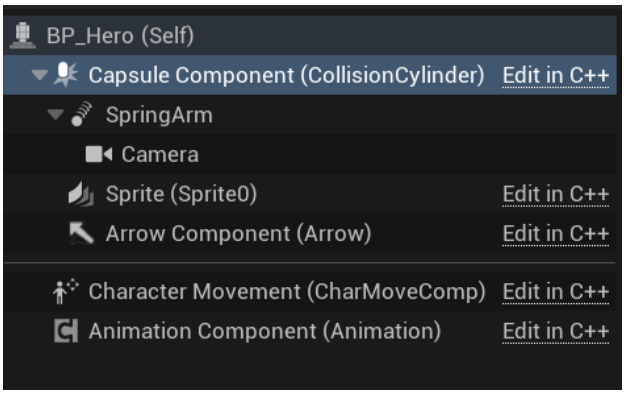

Рисунок 3.15 - Прив’язка камери до персонажу

Нам залишилось перетягнути нашого космонавта на рівень, та спробувати пересуватися (рис 3.16). Для пересування будемо використовувати ноди переміщення (рис 3.17).

Рисунок 3.16. - Персонаж на локації

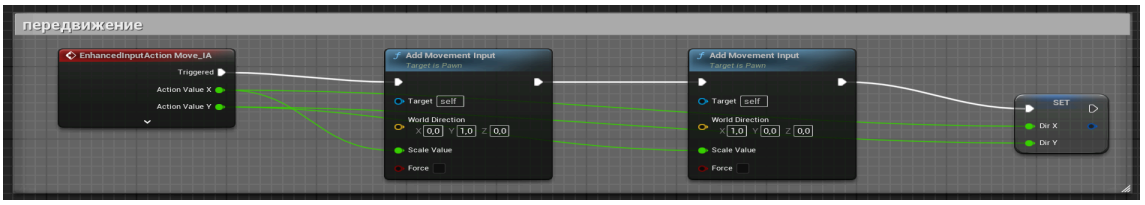

Рисунок 3.17. - Код переміщення персонажу

3.4 Головне меню

Головне меню буде складатися з основного меню яке ми бачимо при вході в гру та меню паузи яке викликається під час перебуванні на локаціях. Основне меню буде з’являтися при вході в гру завдяки налаштуванням проекту. Візуальну частину ми реалізуємо як новий Віджет на сцені, що дасть нам можливість переключатися між сценами та програмувати їх по окремості.

Отже спочатку створимо папки з рівнями LWL та з Віджетами.

Папка LVL відповідає за збереження наших ігрових локацій і перша така локація буде MainMenu. Щоб створити локацію достатньо перейти в папку та після натискання правої кнопки миші вибрати Level (рис. 3.18).

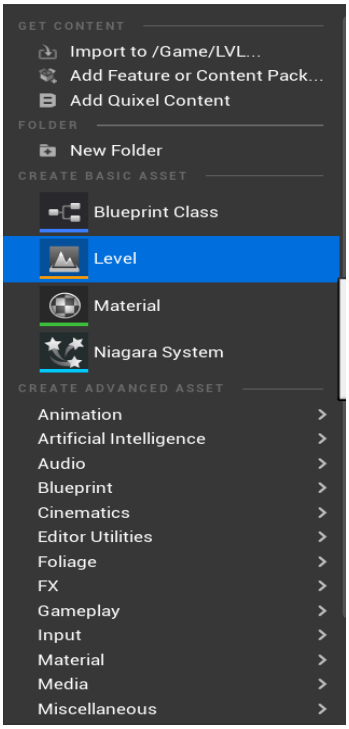

Рисунок 3.18. - Створення нового уровня

Даємо назву рівню та відкриваємо його. В налаштуваннях проекту поставимо MainMenu рівнем за замовчуванням, щоб вся гра починалася саме з нього.

Далі нам потрібна папка в якій ми будемо зберігати усі наші Blueprints з Віджетами. Вони будуть невід’ємною візуальною частиною нашого проекту. В папці створимо перший віджет та назвемо його MenuWG. Цей віджет замінює нам фон. Далі буде віджет Buttons котрий і буде зберігати основну частину інформації. Відкрив віджет ми добавляємо в нього 5 кнопок на які потім вставляємо текст (рис. 3.19)

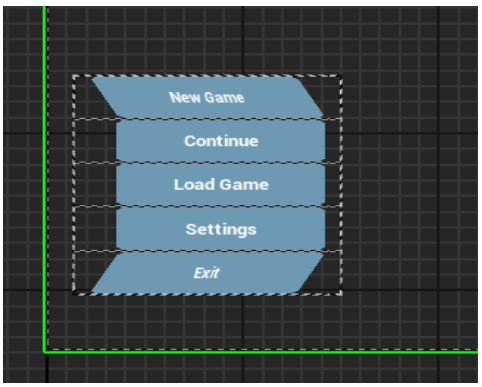

Рисунок 3.19. - Блок з кнопками

Для програмувань кнопок будемо використовувати Blueprint. В нашому випадку нам потрібен Event Construct котрий буде запускатися разом з грою щоб програвати анімацію, а також івенти on clicked для кожної з кнопок. Кожна кнопка буде виконувати функцію 1 раз, для того щоб випадково не призвести до появи багів, будемо використовувати ноду Do Once. Через наявність анімацій, їх також треба підключити до функції. Обираємо Play Animation та одразу виставляємо затримку перед наступною нодою (Delay). Щоб старий віджет зник, треба скрити його. Зробимо це завдяки Remove From Parent, та одразу створимо новий віджет і додаємо його у Viewport (рис. 3.20). Робимо так для кожної кнопки

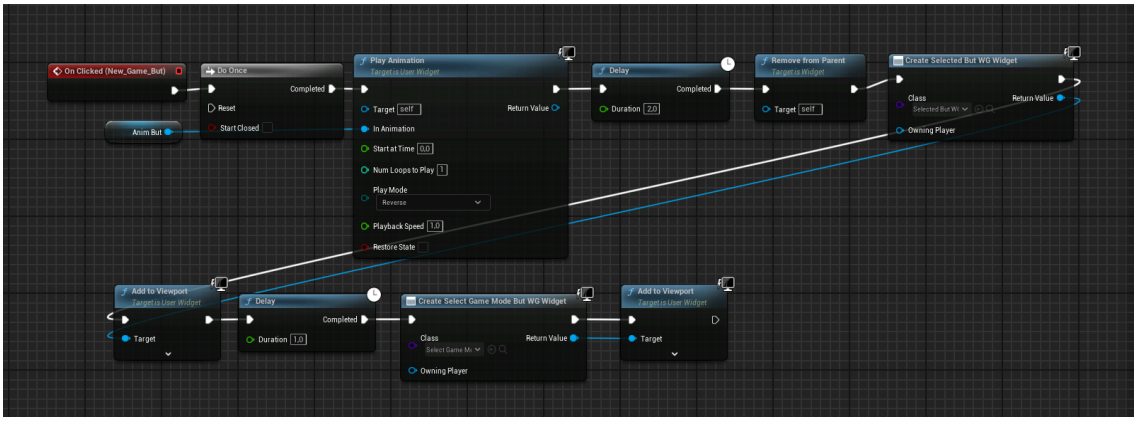

Рисунок 3.20. - Відтворення переходу на інший віджет

Для використання меню паузи створюємо новий віджет (рис. 3.21), який буде викликатися при натисканні кнопки I на клавіатурі (рис. 3.22), а також ставити гру на паузу завдяки функції задля запобігання проблем.

Для цього будемо використовувати нові ноди, такі як: Branch та set game pause. Нода Branch в Blueprint, замінює нам True/False. Нода set game pause просто заморожує час в самій грі.

Рисунок 3.21. - Новий віджет паузи

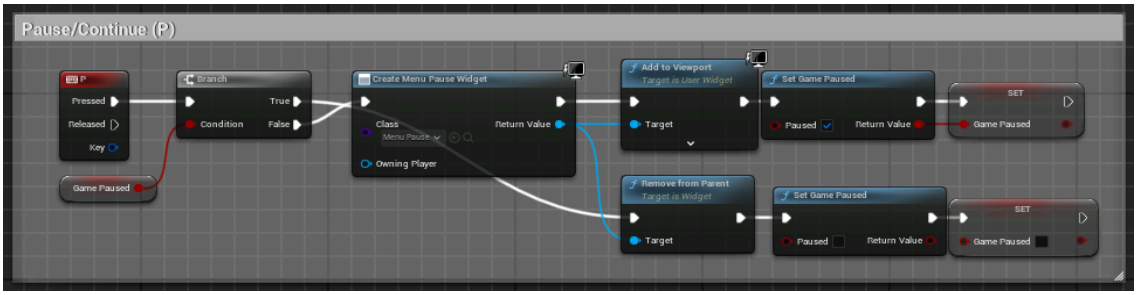

Рисунок 3.22. - Код реалізації паузи

3.5 Створення системи інвентарю

На даному етапі ми вже маємо персонажа та вміння працювати з віджетами. Наступна частина це створення та реалізація системи інвентарю для кращої взаємодії зі світом нашої гри. Для цього нам будет потрібно підібрати акторів, використовувати функції, масив для зберігання нашого інвентаря, Enum та Struct, щоб було легше розрізняти елементи. Створюємо нову папку де ми будемо зберігати усі наші файли. [2, 9, 10]

Почнемо з Enum. Щоб створити його натискаємо пкм, обираємо Add - Blueprint - Enumeration. Для початку добавляємо два айтеми:

Heal - зілля для відновлення життів (Рис. 3.23)

Iron - матеріал для крафту майбутніх будівель (Рис. 3.24)

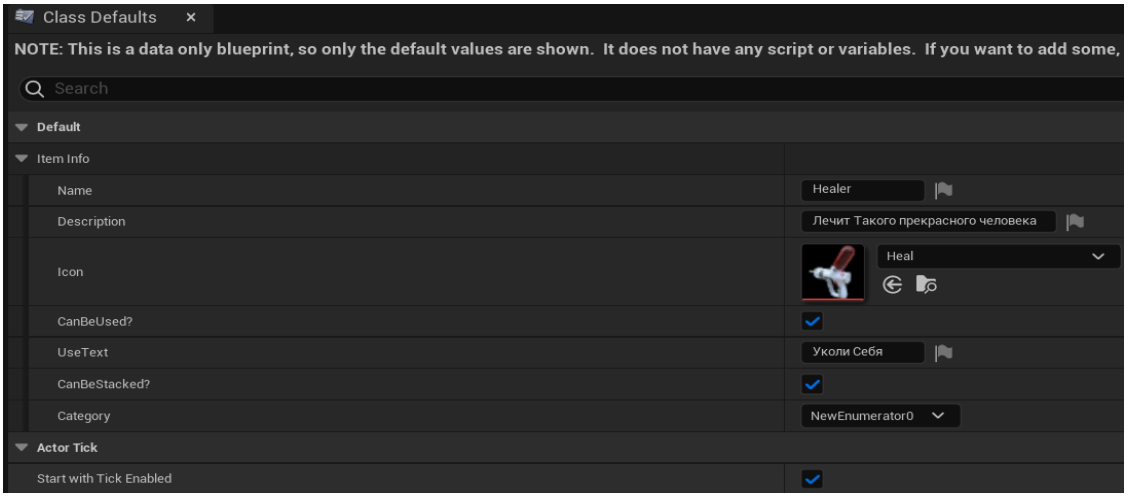

Рисунок 3.23. - Предмет Healer

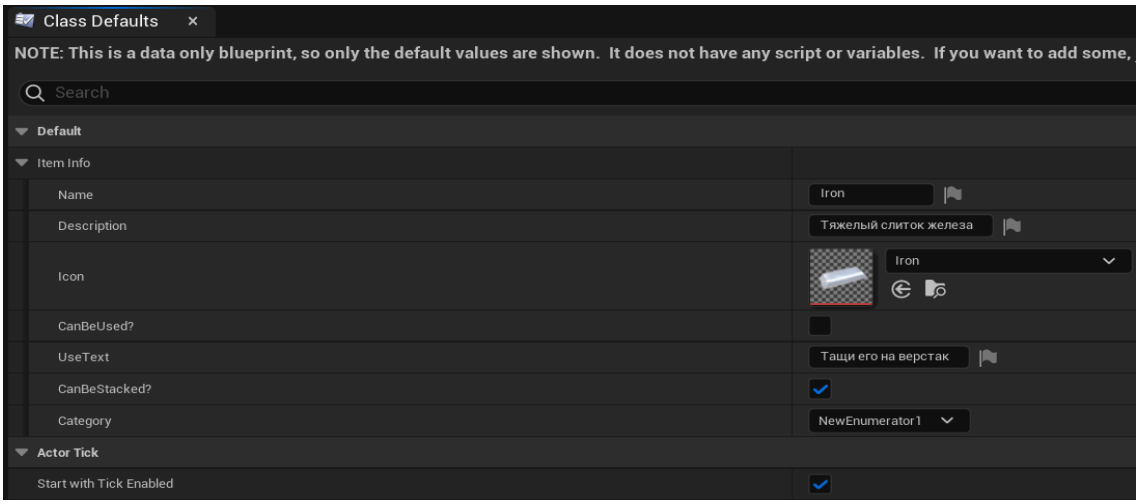

Рисунок 3.24. - Предмет Iron

Зараз нам потрібно задати для предметів якусь властивість та інформацію, тобто справочну інформацію. Цю інформацію ми будемо брати з нового типу файлів structure, а саме з s_itemInfo (рис. 3.25).

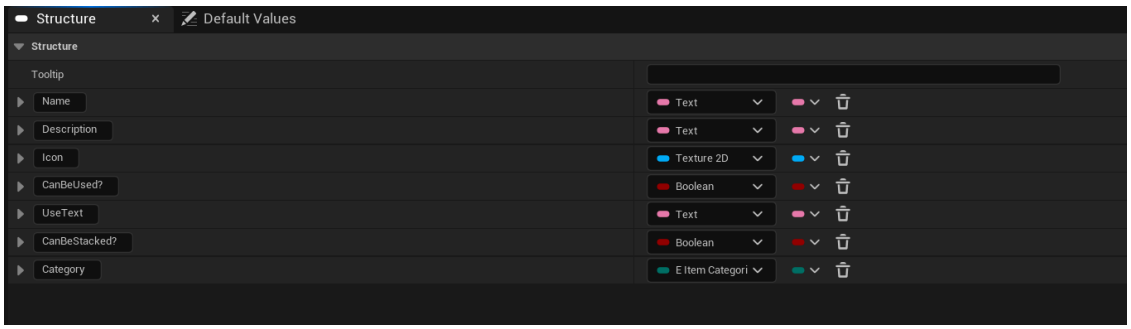

Рисунок 3.25. - Інформація для предметів

В цьому файлі ми зберігаємо все що пов’язане з нашими предметами, від назви до категорії, але задля того щоб скористатися таким поняттям як категорія, нам потрібно створити перерахований тип.

Для цього скористаємося Enum. Enum дозволяє ставити параметри для окремих видів предметів. Наразі створимо два типа: Hp та Item. тепер при перегляді предмета в грі ми можемо побачити до якої категорії він відноситься та що з ним можно робити.

Перейдемо до створення віджету інвентаря. Спочатку ми опрацюємо віджет для зберігання. Будемо використовувати UniformGrid для зберігання Itemslots(рис. 3.26).

Рисунок 3.26. - Вигляд слотів в інвентарі

Зараз треба додати код. Для початку створюємо сам інвентар і виставляємо максимальну кількість стовпців 5 (рис. 3.27), а також кнопку виходу (рис. 3.28).

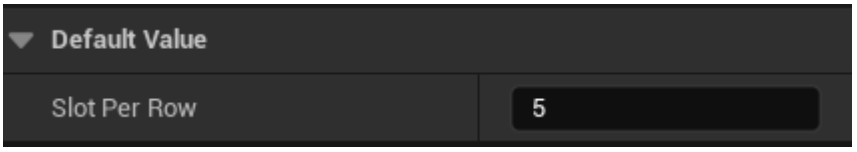

Рисунок 3.27. - Кількість слотів по горизонталі

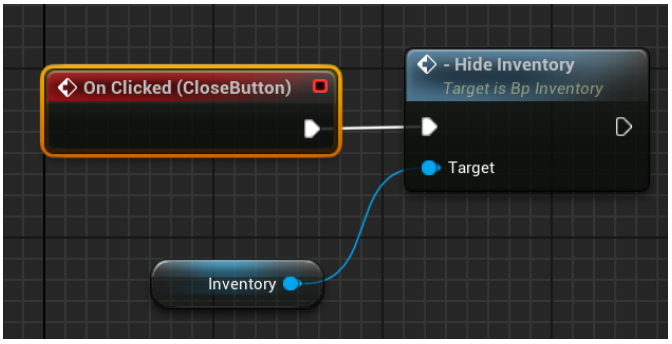

Рисунок 3.28. - Івент натискання на кнопку

Тепер працюємо над ItemSlots. Спочатку виставляємо розмір слота (рис. 3.29).

Рисунок 3.29. - Висота та ширина віджету

Далі додаємо: розмір, зображення фону, зображення предмета, кнопку для використання, текст з кількістю (рис. 3.30).

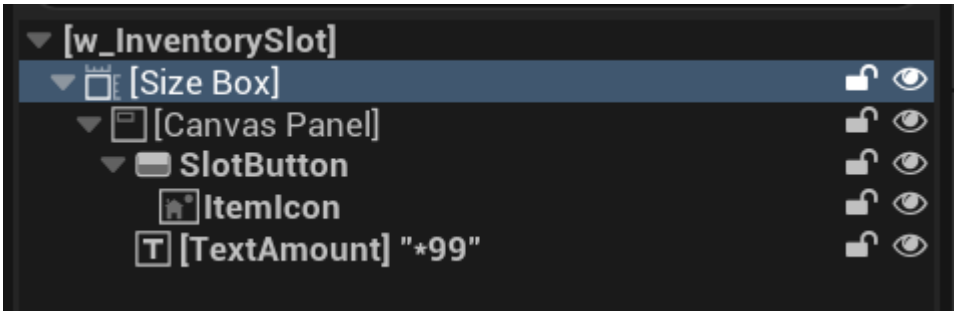

Рисунок 3.30. - Ієрархія

Також розробимо ThrowWidget з контекстним меню де будуть кнопки: використати, розділити, викинути. Всі ці кнопки ми програмуємо прямо в нашому віджеті.

В кожному віджеті ми маємо створити змінні для індексу запасів (Index - буде зберігати індекс нашого інвентаря) (рис. 3.31).

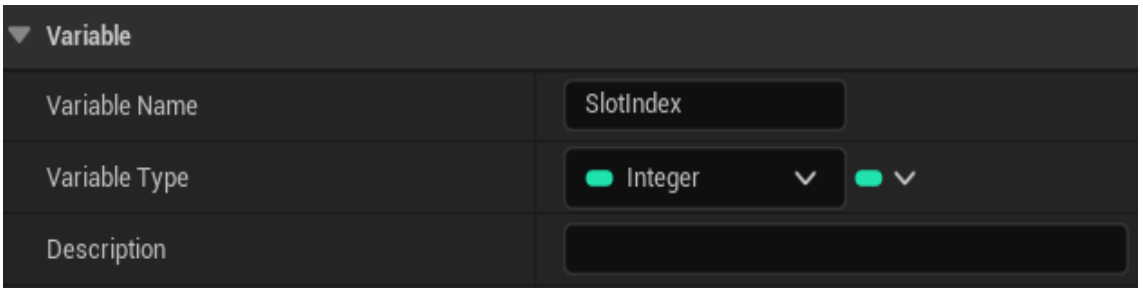

Рисунок 3.31. - Змінна

Також нам потрібне посилання на батьківський віджет (рис. 3.32).

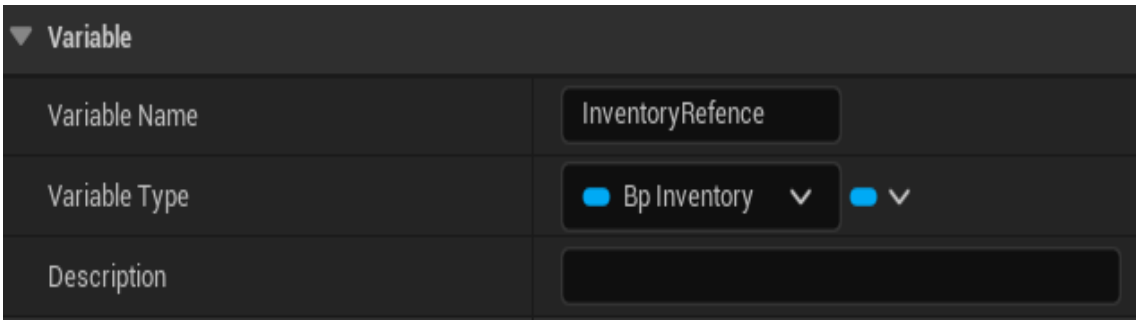

Рисунок 3.32. - Змінна посилання

Повертаючись до віджету інвентаря. Маємо поєднати Індекс предмету та Батьківський віджет. Далі переходимо до ItemSlot, щоб додати код. Спочатку нам потрібна посилання на екземпляр гри, щоб отримати доступ до масиву інвентаря. Ми отримаємо його при будівництві слота (рис. 3.33).

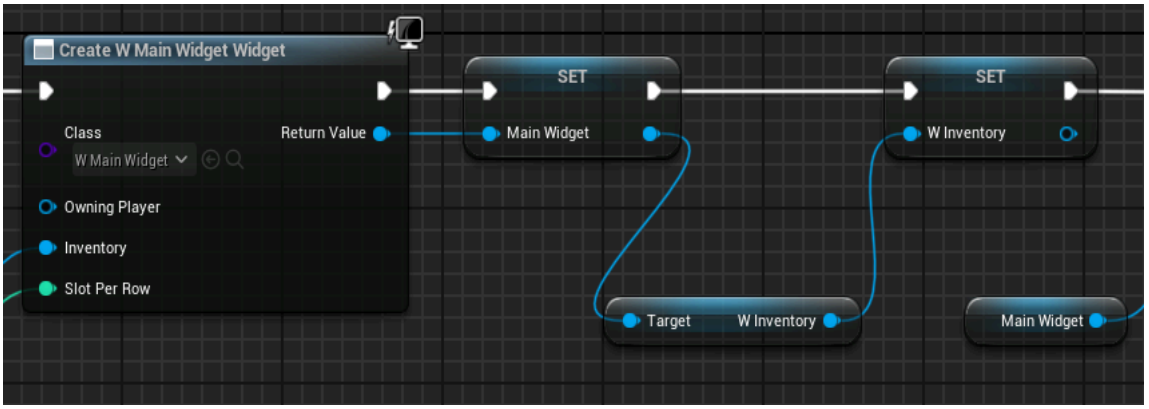

Рисунок 3.33. - Посилання на екземпляри

Далі встановити ItemImage. Для цього повертаємося в візуальний дизайнер, обираємо ItemImage, в розділі Bind, Create Binding. Ми використаємо переміну Index, для отримання типу елементу з масиву та зображення зі списку. Це допоможе програмі обрати яке зображення використовувати (рис. 3.34).

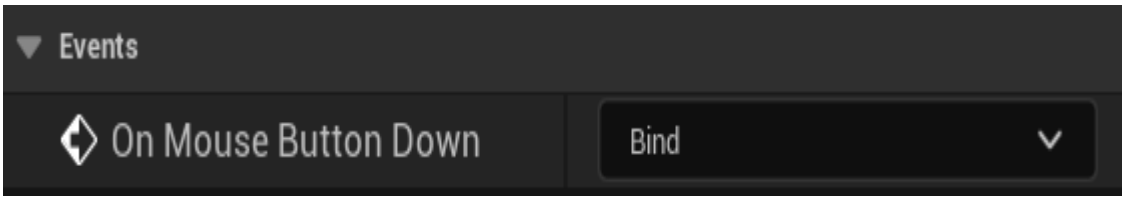

Рисунок 3.34. - Bind зображення

Повертаючись до Event Graph, добавляємо код для приховування кнопок (рис. 3.35).

Далі пишемо код для кнопки Використати (рис. 3.36).

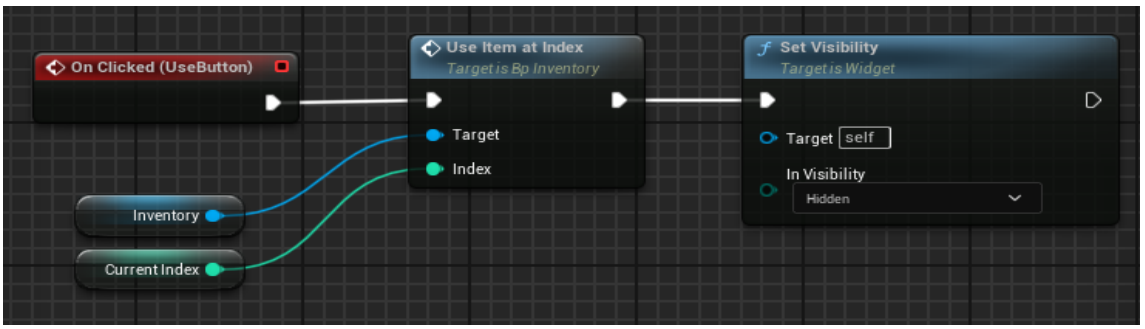

Рисунок 3.35. - Приховування після натиску

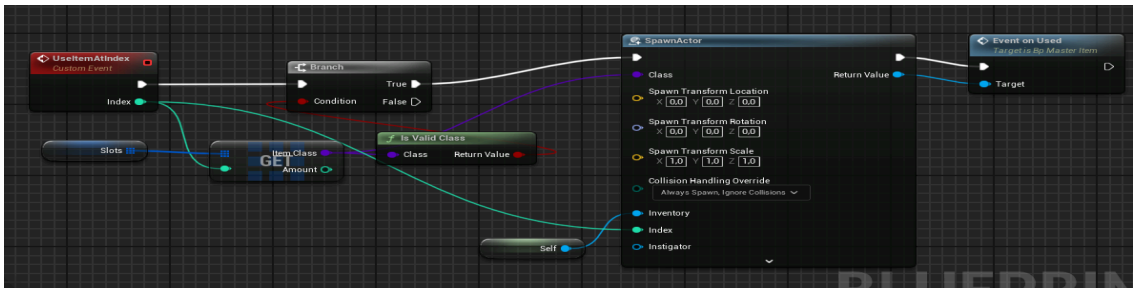

Рисунок 3.36. - Код кнопки Використати

Залишається тільки додати код для відображення головного віджету інвентаря. Це ми зробимо в Blueprint головного героя (рис. 3.37).

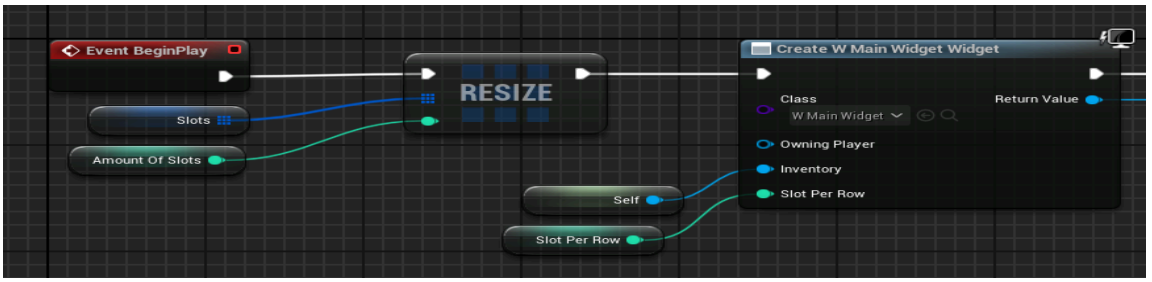

Рисунок 3.37. - Відображення головного віджету

3.5.1 Функція підбирання предметів

Створюємо нового актора, та називаємо його PickUp_Master. Цей актор допоможе нам створювати дочірні класи які ми можемо розміщувати в ігровому просторі (на локації) (рис. 3.38).

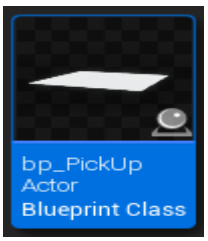

Рисунок 3.38. - Blueprint класу

Для актора нам знадобиться Mesh та Collision Hull. В налаштуваннях нам потрібно відредагувати налаштування зіткнення (рис. 3.39).

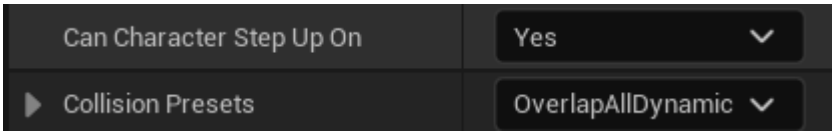

Рисунок 3.39. - Налаштування

Зараз нам потрібно додати візуальний код для створення підняття.

Актор перекритий кимось - Перевірте, чи був це гравець, привівши його до персонажа - Отримати ігровий екземпляр і застосувати його до нашого ігрового екземпляра - Отримати з нього інвентар - Додати елемент в інвентар, використовуючи створений тип змінної - Знищити актора (рис. 3.40).

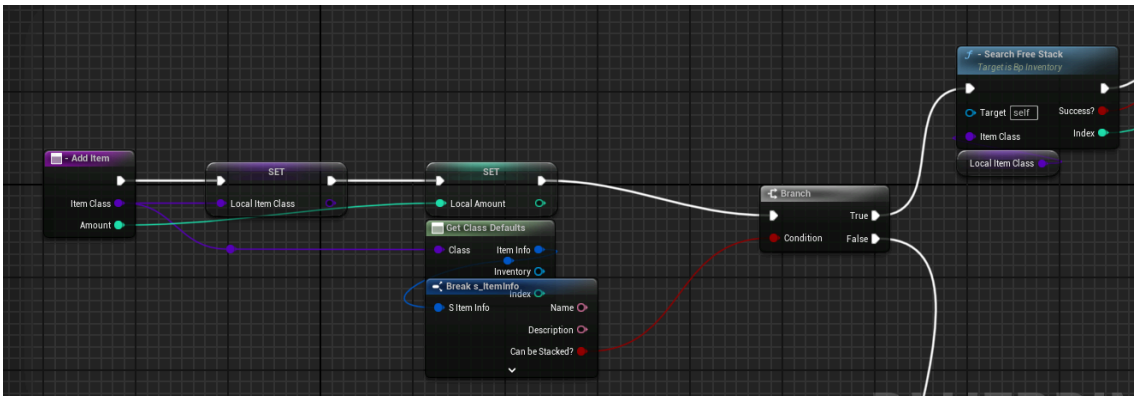

Рисунок 3.40. - Додавання предмету

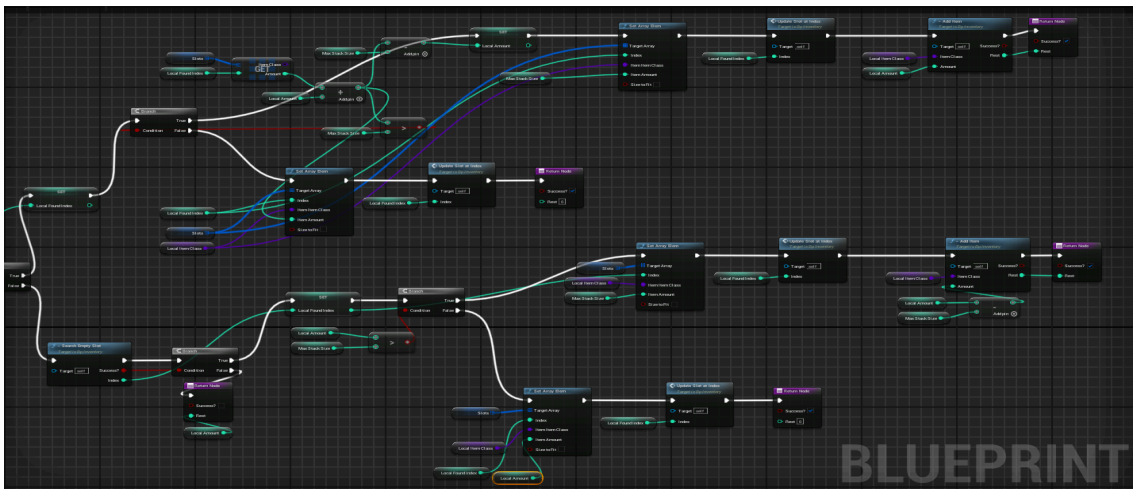

Рисунок 3.41.1 - Додавання предмету

Для додання нового предмету нам потрібно створити дочірній клас від Blueprint - PickUp_Master. Для цього натискаємо пкм по батьківському блюпрінту і створити дочірній клас. Якщо цей предмет можно буде використовувати, ставимо галочку напроти функції Use (рис. 3.42).

Рисунок 3.42. - Наявність флажків

3.5.2 Викидання предмету

Для реалізації цієї функції ми звернемося до віджету ActionWidget та для кнопки Throw додаємо код який і буде виконувати функцію викидання. Логіка буде схожа с доданням, а головна функція буде викликатися з блюпринта інвентаря (рис. 3.43).

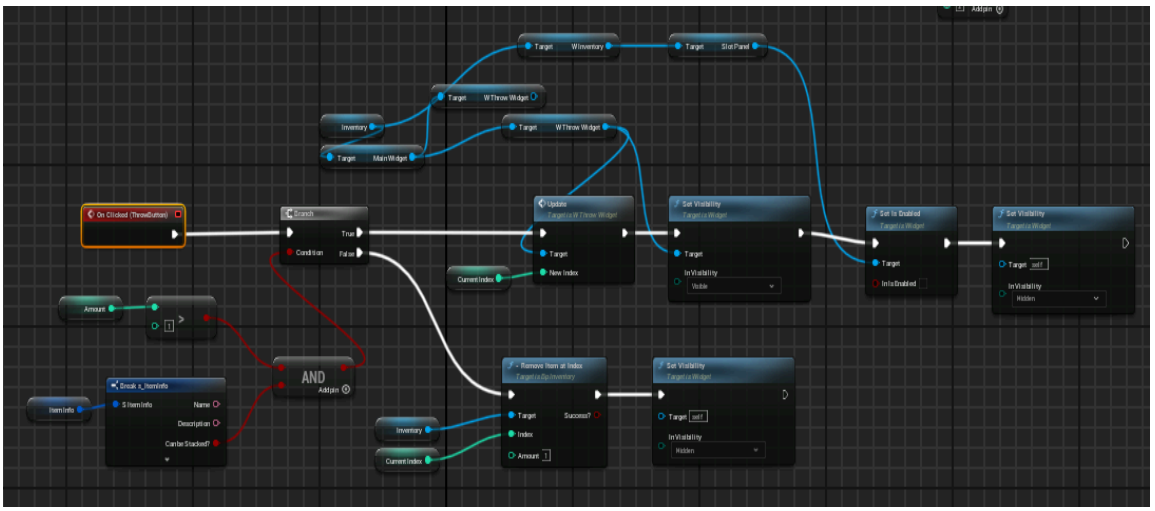

Рисунок 3.43. - Код кнопки викидання

3.5.3 Поділ стака

Також звертаємося до віджету кнопок та викликаємо основну функцію з блюпринта інвентаря. Логіка полягає в пошуку вільного слота через індекс та поділ стака на дві цілі частини, або за вибором користувача (скільки потрібно розділити). Це робиться для коректного використання місця в інвентарі (рис. 3.44).

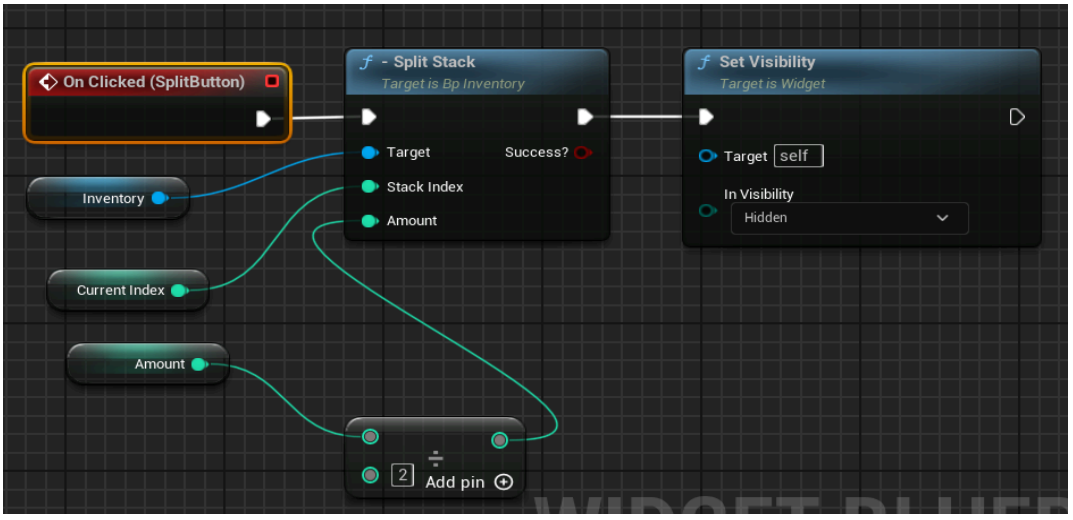

Рисунок 3.44. - Кнопка поділу стака

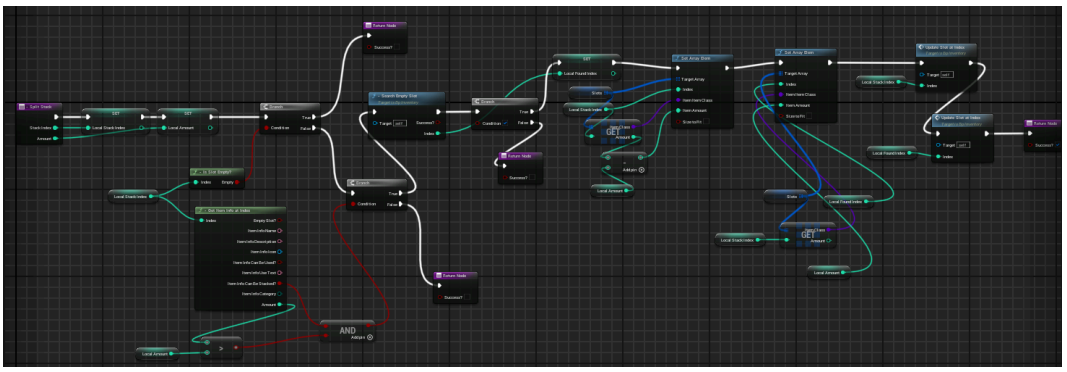

Рисунок 3.44.1. - Поділ стака

3.5.4 Використання предмету

Використання буде реалізовано завдяки кнопці Використати та логіки видалення предмету зі сцени без можливості подальшого його використання. Це надасть нам можливість робити вичерпні предмети котрі будуть мати свою цінність (рис. 3.45).

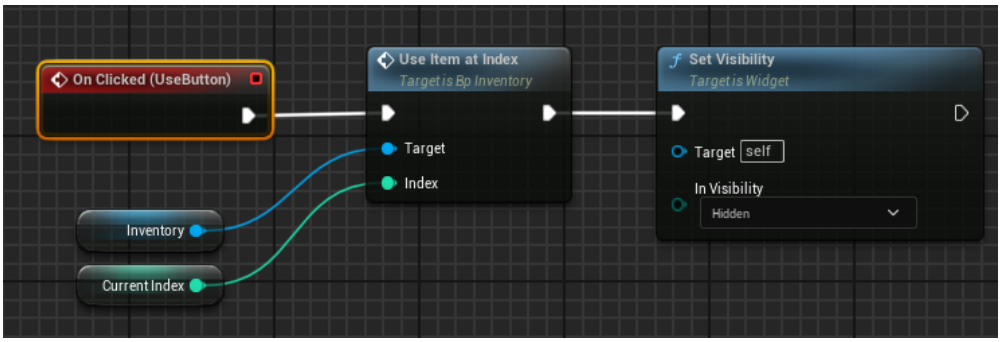

Рисунок 3.45. - Кнопка використання

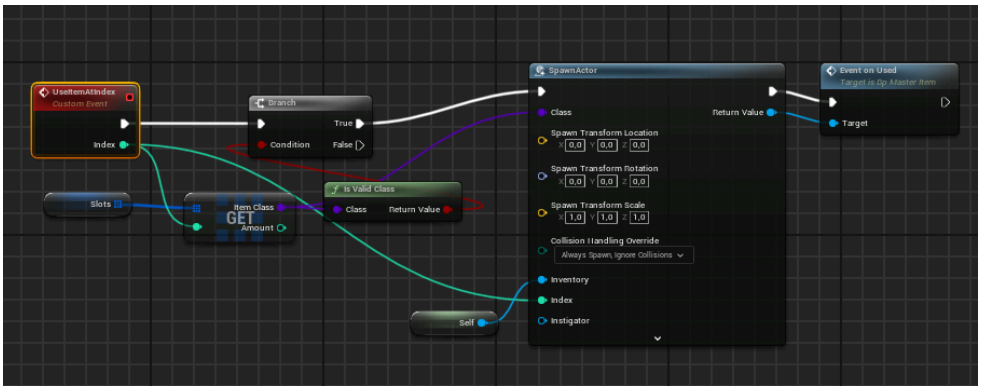

Рисунок 3.45.1 - Кнопка використання

3.5.5 Drag & drop

Drag and drop - це функція, що дозволяє користувачам переміщати елементи інтерфейсу, або об’єкти (в нашому випадку об’єкти в інвентарі). Ця функція використовується для інтерактивних і зручних інтерфейсів користувача. Працює функція з UMG з використання Blueprint, що дає можливість писати візуальний код без використання C++. Основними подіями являються: OnDragDetected, OnDrop, OnDragCancelled. [3]

Використовувати Drag and drop ми будемо для переміщення віджетів та предметів в віджеті інвентаря.

Для початку нам треба створити Drag Widget в якості батьківського класу. Це надасть нам можливості передавати інформацію як частину дії переміщення (рис. 3.46).

Рисунок 3.46. - Віджет Drag and drop

У віджеті створюємо Референс з параметрами Instance Editable, Expose on spawn (рис. 3.47).

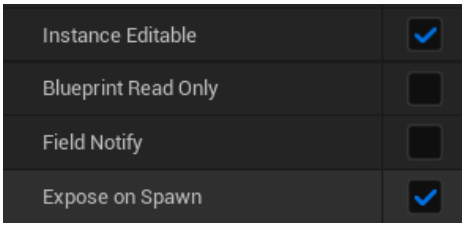

Рисунок 3.47. - Параметри

Також нам знадобиться переміна Vector2D з такими ж флажками. Це буде зміщенням, звідки буде відтворюватися переміщення.

Настройка Віджетів.

На цьому етапі ми будемо визначати коли треба відтворювати функцію переміщення по натисканню лівої кнопки миші.

Для цього нам потрібно переоприділити OnMouseButtonDown та OnDragDetected. Це створить вкладки в графіку подій.

Далі будемо добавляти код для подій переміщення (рис. 3.48).

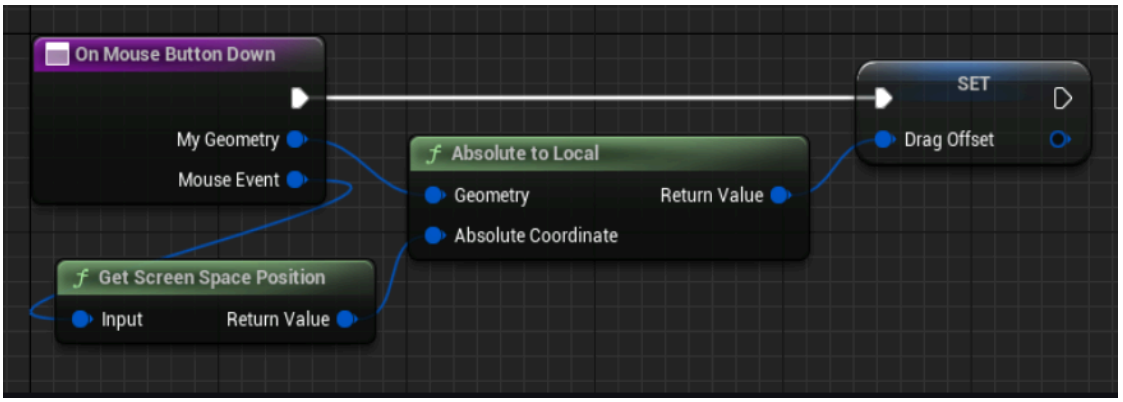

Рисунок 3.48. - Код подій переміщення

Для виявлення функції переміщення нам потрібна вкладка OnDragDetected. Додаємо вузол Створити Віджет, та встановлюємо клас Drag Widget, а також значення Миша вниз. (рис. 3.49).

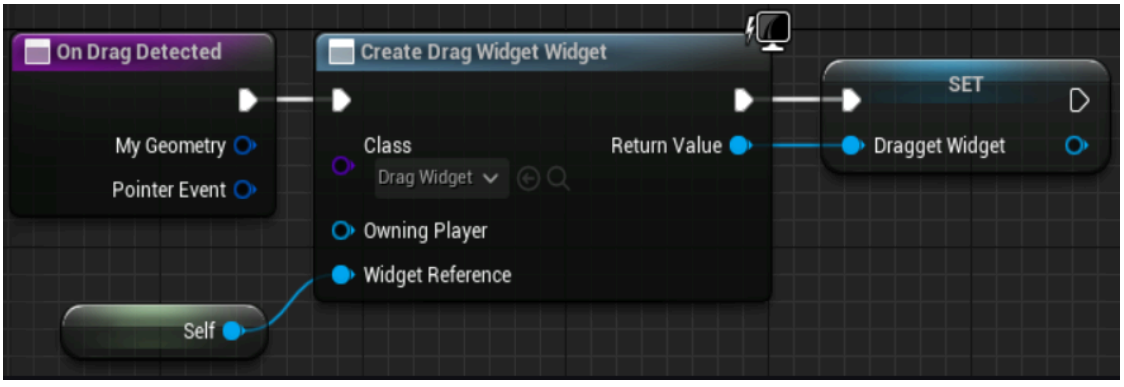

Рисунок 3.49. - Створення віджету

Для налаштування OnDrop нам потрібно налаштовувати вже основний віджет та переоприділити що відбувається при виконанні функції (рис. 3.50).

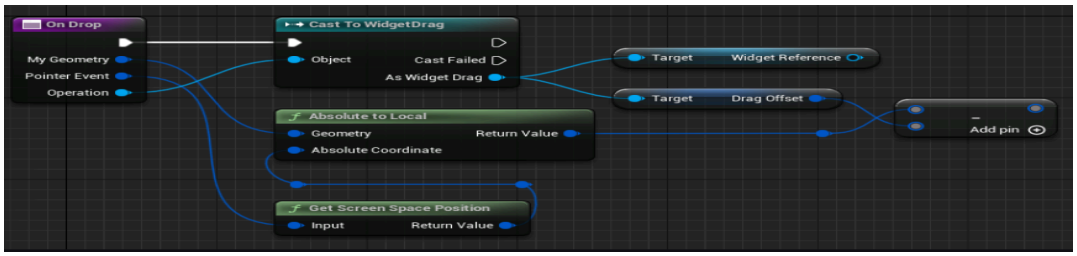

Рисунок 3.50. - Налаштування функції

Також нам понадобиться встановити контакт з нодами геометрії для правильного переміщення на екрані. Це дозволить віджетам переміщатися разом з курсором миші в 2D просторі (рис. 3.51).

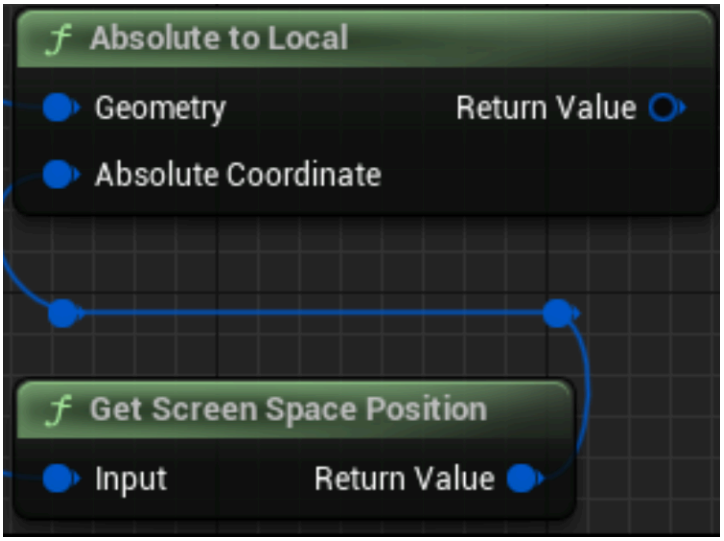

Рисунок 3.51. - Ноди геометрії

Для коректного відображення на екрані ми будемо вимикати основний віджет на передавати його на віджет переміщення, після закінчення переміщення обратно вмикаємо основний віджет але на новому місці (рис. 3.52).

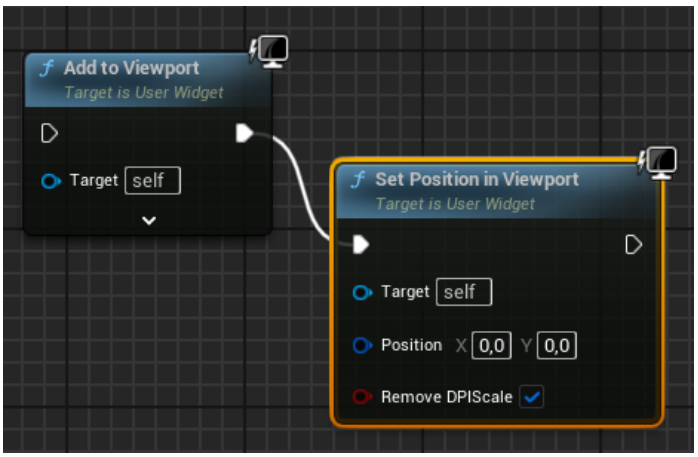

Рисунок 3.52. - Додавання до Viewport

Кінцевим результатом являється переміщення, або заміна між собою слотів інвентаря, а також переміщення всього інвентаря по екрану.

3.6 Створення ігрового простору

Ігровий простір, це основа Майже всієї гри. Неможливо грати в щось коли в тебе немає відкритої карти, або правильно спроектованого рівня. [7]

Tile set — це колекція графічних елементів, які користувачі використовують для створення ігрових локацій, тобто для будування TileMap. Кожен тайл може бути відмальован або завантажений з інтернету. Також мі можемо використовувати одне і те ж саме зображення декілька раз для створення великої локації. Зазвичай чю механіку використовують в 2D платформерах, або стратегіях.

Для створення зображення будемо використовувати програмне забезпечення Adobe Photoshop. Завдяки цьому забезпеченню ми спокійно можемо намалювати тайли схожі на траву (рис. 3.53).

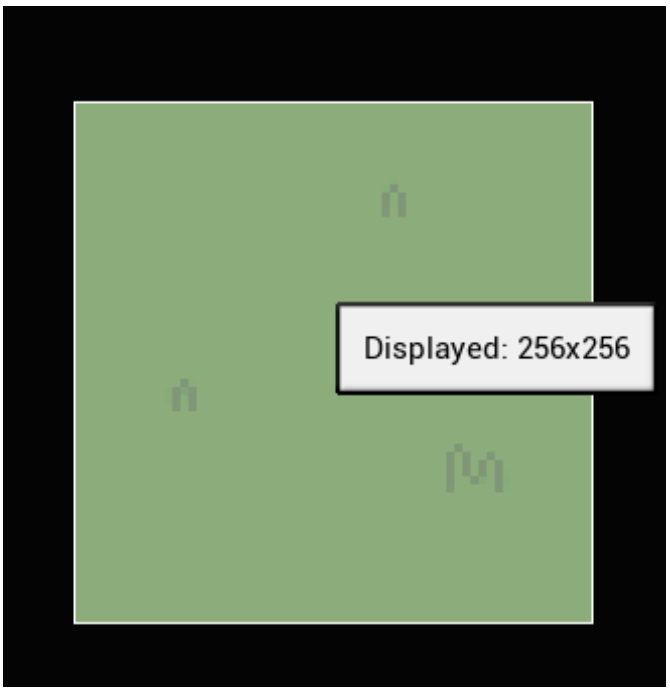

Рисунок 3.53. - Зображення Tile

Користуватися TileSet не важко. Головне розуміти що ти хочеш отримати від твоєї ігрової карти і як правильно розміщувати їх на TileMap.

Ціль для нас - Створити базову невелику карту (TileMap) на якій користувач може пересуватися, та опановувати механіки нової для нього гри.

Тож чому саме TileMap. Це проста в опануванні механіка яка дозволяє створювати великі карти, та ділити всю локацію на під уровні. Також будування нової локації використовуючи вже готові тайли це набагато швидше і більше піддається форматуванню. Карти створені завдяки TileSet займають набагато менше місця, що дає змогу зекономити місце на пристроях користувачів.

Для наглядної роботи цієї механіки, побудуємо карту 50 на 50 тайлов в один слой (рис. 3.54).

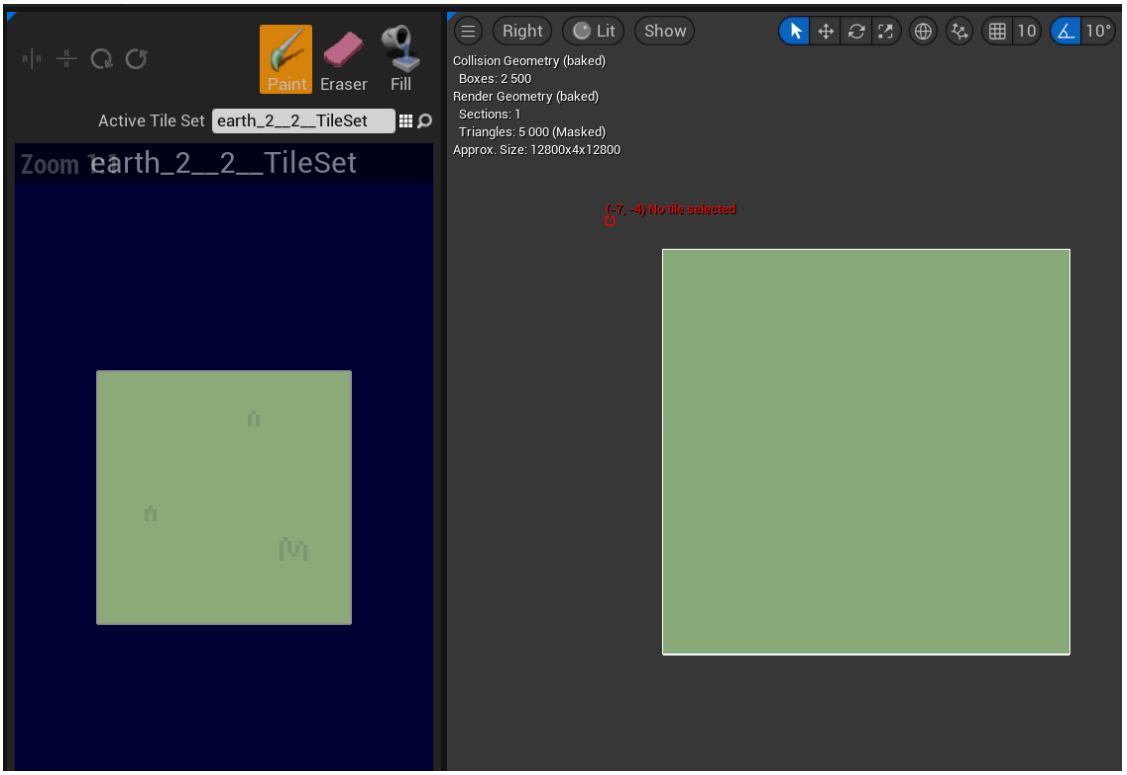

Рисунок 3.54. - Вигляд простору роботи з TileMap

ВИСНОВОК

Розвиток індустрії відеоігор, зокрема казуальних стратегій, відображає сучасні тенденції у світі розваг, де простота та захопливість гри приваблюють широкий спектр гравців. Створення 2-D казуальної стратегії за допомогою Unreal Engine, використовуючи Blueprints і плагін "PaperZD", демонструє потужні можливості цих інструментів у реалізації креативних ідей та забезпеченні високої якості ігрового продукту.

Проект, де гравець виступає в ролі космонавта, який досліджує віддалені куточки всесвіту, представляє захопливий геймплей з багатьма цікавими механіками. Будівництво, управління інвентарем, автозбереження, налаштування гри, різноманітні способи переміщення, атмосферні умови, боротьба з ворогами та покращення будівель і транспорту створюють глибокий і насичений ігровий досвід. Вся ця система дозволяє гравцеві зануритися у світ гри, розвивати свої навички та отримувати задоволення від процесу.

Таким чином, реалізація подібного проєкту на базі Unreal Engine з використанням сучасних технологій є важливим кроком у розвитку казуальних стратегій і підтверджує перспективи використання цих інструментів для створення захопливих і доступних ігор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Tutorial] Simple Inventory System. - Development / Programming & Scripting - Epic Developer Community Forums [Електронний ресурс] // Режим доступу: https://forums.unrealengine.com/t/tutorial-simple-inventory-system/1316560
2. [Tutorial] Simple Inventory System [Електронний ресурс] // Режим доступу: https://forums.unrealengine.com/t/tutorial-simple-inventory-system/1316560
3. Creating Drag and Drop UI [Електронний ресурс] // Режим доступу: https://dev.epicgames.com/documentation/en-us/unreal-engine/creating-drag-and-drop-ui-in-unreal-engine?application_version=5.0
4. Unreal Engine 5.0 Documentation [Електронний ресурс] // Режим доступу: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-0-documentation?application_version=5.0
5. Unreal Engine 5 Створення першої 2D гри з 0 [Електронний ресурс] // Режим доступу: https://www.youtube.com/watch?v=FbXD5VlsRGA&list=PLJCeoSg-p6hbInSTKa8a6MNVqRei9J4U_&index=2&ab_channel=GoodyIT
6. Unreal Engine 4 Documentation [Електронний ресурс] // Режим доступу: https://docs.unrealengine.com/4.27/en-US/
7. Best practices when using tilesets in UE4 [Електронний ресурс] // Режим доступу: https://www.nexatli.com/blog/best-practices-when-using-tilesets-in-ue4
8. Створюємо 2D платформер за допомогою Unreal Engine 4. [Електронний ресурс] // Режим доступу: https://habr.com/ru/articles/237409/
9. Система інвентарю [Електронний ресурс] // Режим доступу:

https://www.youtube.com/playlist?list=PLuqxJ8gJ4DlEy0Z36E6CsaBPA-rCC-nhb

10. How to Make an Inventory System in Unreal Engine 5 [Електронний ресурс] // Режим доступу: https://www.youtube.com/playlist?list=PL4G2bSPE_8umjCYXbq0v5IoV-Wi_WAxC3
11. Unreal Engine 5 [Електронний ресурс] // Режим доступу: https://www.unrealengine.com/en-US
12. Unity [Електронний ресурс] // Режим доступу: https://unity.com/ru
13. Godot [Електронний ресурс] // Режим доступу: https://godotengine.org/
14. Види Ігор [Електронний ресурс] // Режим доступу: https://games-yes-no.webnode.com.ua/vidi-igor/
15. Кобзар К. В. Вплив Інформаційних Технологій на циклічність розвитку економічних систем [Електронний ресурс] // Режим доступу: https://drive.google.com/file/d/11tUmgDWXXxDfcbPv_UaWg7PTW4vwA8Ce/view?usp=drive_link
16. Костянтин Кобзар, Марина Кіріліна. «INDUSTRY 4.0» : ВПЛИВ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ НА ПОПИТ НА КОМПЕТЕНЦІЇ [Електронний ресурс] // Режим доступу: https://conf.krok.edu.ua/ISR/ISR-2021/paper/view/964