МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

ФАКУЛЬТЕТ *ІНФОРМАТИКИ*
КАФЕДРА МУЛЬТИМЕДІЙНИХ СИСТЕМ

РОЗРОБКА ГРИ НА UNREAL ENGINE 5

Текстова частина до курсової роботи
за спеціальністю «Комп’ютерні науки» 122

Керівник курсової роботи
ст.в. Борозенний С. О.

_____
(підпис)

Виконав студент 3 курсу
Пожаров Д. С.

«____»_____2023 р.

Київ 2023

Зміст

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,

Жежерун О.П.

_____

(підпис)

«___» _____ 2023 р

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту 3-го курсу, факультету інформатики Пожарову Дмитру

Тема: Розробка гри на Unreal Engine 5

Вихідні дані:

Зміст ТЧ до курсової роботи:

Зміст

Анотація

Вступ

1. Загальна інформація про Unreal Engine 5
2. Дослідження актуальності теми
3. Реалізація гри на Unreal Engine 5

Висновок

Список використаної літератури

Дата видачі «____»_____ 2023 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Розробка гри на Unreal Engine 5:

№	Назва етапу курсового проєкту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	10.09.2022	
2.	Аналіз матеріалів за темою курсової роботи	15.11.2022	
3.	Розробка гри на Unreal Engine 5	08.01.2023	
4.	Написання текстової частини	22.03.2023	
5.	Надання роботи на перевірку керівником	12.05.2023	
6.	Коригування на основі результатів перевірки	13.05.2023	
7.	Остаточне оформлення роботи та презентації	14.05.2023	
8.	Захист курсової роботи	23.05.2023	

Пожаров Д.С. _____

Борозенний С. О. _____

«____» _____ _____р.

Анотація

В даній курсовій роботі було проведено дослідження процесу розробки гри на Unreal Engine 5 з використанням візуальної мови програмування Blueprint. Основним завданням роботи було розроблення гри "Морський бій" в 3D, що дозволило продемонструвати можливості потужного інструменту для розробки ігор. В процесі роботи були досліджені та використані різні функції та інструменти, які надає рушій.

Гравець може вибрати складність, розташувати свої кораблі на ігровому полі, або згенерувати випадкове розташування. Ціль гри — знищити увесь флот ворога до того, як ворог знищить флот гравця. На полегшеній складності супротивник буде стріляти навмання, на стандартній буде грати за стратегією, тому гра стане справжнім випробуванням для гравця.

Вступ

Швидкість розвитку технологій у світі на сьогоднішній день є надзвичайно високою. Кожного року з’являються нові рішення, які вносять значні зміни в повсякденне життя. За останні роки швидкість розвитку технологій в світі зросла експоненційно, що зумовило появу нових можливостей та викликів у багатьох галузях. Однією з таких галузей є ігрова індустрія, яка стала однією з найпопулярніших та найприбутковіших галузей в світі. За даними звіту Global Games Market Report 2022, обсяг ринку відеоігор складає більше 190 мільярдів доларів США, а кількість гравців вже перевищило 3.2 мільярди [1].

Розробка гри — це завжди складна задача, що потребує великої кількості вмінь та знань. Для досягнення мети зазвичай потрібна ціла команда розробників, включаючи програмістів, графічних дизайнерів, тестувальників, менеджерів, геймдизайнерів та інших спеціалістів. Кожен з цих фахівців виконує свою власну задачу та вносить свій вклад. Наприклад, програмісти створюють код та алгоритми, які визначають логіку та поведінку продукту, геймдизайнери придумують геймплей, механіки та загальну концепцію гри, а графічні дизайнери створюють моделі персонажів, об'єктів та локацій. Саме через такий обсяг різноманітної роботи, одним з ключових факторів успіху у створенні якісних ігор є використання потужних інструментів для їх розробки.

Unreal Engine 5 є одним з найпопулярніших та найпотужніших ігрових рушіїв який володіє багатим функціоналом та забезпечує зручну роботу над проєктами. Він надає потужні інструменти для створення фізики, анімації, графіки тощо. Крім того, Unreal Engine 5 має ряд унікальних технологій, таких як Nanite, що дозволяє відображати мільйони геометричних об'єктів в реальному часі без значного впливу на продуктивність, та Lumen, що забезпечує реалістичне освітлення в ігровому світі.

Багато відомих ігор було створено на базі Unreal Engine, таких як Ark II, Hogwarts Legacy, Life is Strange, S.T.A.L.K.E.R. 2 та інші, що є чудовими прикладами його ефективності та можливостей [2].

1. Загальна інформація про Unreal Engine 5

1.1 Історія Unreal Engine 5

Перша версія Unreal Engine була випущена у 1998 році компанією Epic Games. Першою грою на рушії була гра “Unreal”, яка з’явилася того ж року. Розробники значно удосконалили систему виявлення зіткнень, штучний інтелект, видимість об’єктів та керування файловою системою і об’єднали в одне програмне забезпечення. Ця версія рушія мала великий вплив на індустрію відеоігор і встановила новий стандарт якості та можливостей для розробників гри. Підтримувалися багато платформ, таких як PC, PlayStation 2, Xbox та інші.

Згодом, у 2002 році вийшов Unreal Engine 2. Рушій був значно покращений та отримав нові можливості, такі як підтримка більшої кількості платформ та покращений інструментарій для розробки.

В 2005 році вийшов Unreal Engine 3, який ще більше покращив графіку, фізику та інші можливості рушія.

19 березня 2014 року Epic Games презентувала 4 версію написану на мові C++. Спочатку рушій розповсюджувався за 19 доларів на місяць, але вже в 2015 році став безкоштовним для всіх користувачі, за умови сплати 5% від доходу за продукти, які приносять понад 3000 доларів на квартал. Це зробило Unreal дуже привабливим для невеликих проєктів. Четверта версія рушія підтримує всі сучасні платформи для розробки.

У травні 2020 було анонсовано Unreal Engine 5. Одна з головних інновацій — це Nanite, нова технологія, яка спроможна рендерити надзвичайно деталізовані об'єкти в реальному часі, а також дозволяє додавати на сцену мільйони об’єктів, не впливаючи на продуктивність. Друга інновація — це нова система освітлення Lumen, яка дозволяє вираховувати освітлення на сцені в реальному часі. Також додали багато нових технологій, наприклад, World Partition — нова система розподілу світу на сегменти, та Niagara — нова система частинок. [3]

1.2 Основні функції та можливості Unreal Engine 5.

Unreal Engine 5 — це дуже потужний інструмент, створений для розробки ігор, симуляцій, віртуальної реальності та інших програм, що вимагають високої рівня графіки та інтерактивності. Відкритий початковий код дозволяє переписати рушій під будь які вимоги. А візуальна мова програмування Blueprint полегшує розробку механік, анімацій, штучного інтелекту тощо.

Основні сфери в яких використовується Unreal Engine:

1. **Ігри.** Рушій створений для розробки ігор, отже має безліч інструментів для цього. Unreal Engine зручний для роботи над проєктом будь-якого масштабу та з будь-якою кількістю людей в команді. Підтримка рендерингу з трасуванням променів, динамічних тіней, наявність універсальних інструментів освітлення та гнучкого редактору матеріалів дає можливість легкої розробки якісного контенту.
2. **Архітектура.** Завдяки підтримці фотореалістичних моделей та текстур користувачі можуть втілити будь-які свої ідеї. Рушій дозволяє візуалізувати різноманітні будівлі, приміщення та об'єкти. Також рушій підтримує інтеграцію з додатками віртуальної реальності, що дає змогу створювати віртуальні тури та дослідження об'єктів.
3. **Кіноіндустрія.** Візуалізація в реальному часі допомагає студіям зменшити час та витрати. За допомогою Unreal Engine можна створити фотореалістичні візуалізації для кіно. Рушій надає можливість створювати сцени зі складними ефектами, такими як вибухи, руйнування тощо. Unreal Engine був використаний при зйомках серіалу “Мандалорець”, де він допоміг створити візуальні ефекти та пейзажі, а в фільмі “Форд проти Феррарі” він дозволив створити візуалізації гоночних трас та автомобілів з високою деталізацією. [4]
4. **Трансляції та прямі етери.** Unreal Engine дозволяє створювати реалістичну графіку у реальному часі, що в поєднанні з доповненою реальністю дає можливість створювати вражаючі спеціальні ефекти в прямому етері. Це ідеально підходить для різних шоу або спортивних

подій, адже він дозволяє створювати віртуальні анімовані студії, що зацікавлять глядача. Саме за допомогою Unreal Engine, на події “Super Bowl” було створено графіку, 3D-гравців, тренерів та співачку Ріанну. [5]

5. **Анімація.** Unreal Engine має всі необхідні інструменти для створення анімацій. Завдяки просунутій фізиці і інструментам для створення реалістичної поведінки волосся, хутра, а також візуалізації обличчя розробники можуть створювати якісні анімації для використання в іграх, мультфільмах та інших проєктах.
6. **Автомобілі.** Завдяки вбудованим інструментам, рушій використовується у віртуальних шоурумах відомих автомобільних компаній, таких як “Audi”, “Chevrolet”, “Volkswagen”, “Porsche” тощо. А також для тестування автономних транспортних засобів безпечно для людей. Unreal Engine використовувався для рендерингу в реальному часі реклами в “Формула-1”.[6]
7. **Симуляції.** Завдяки Unreal Engine симуляції стали реалістичними і точними. Рушій дає можливість зануритися у віртуальну реальність і досконало вивчити галузь. Наприклад, хірурги можуть навчитися оперувати, а механіки ремонтувати літаки. А за допомогою підтримки VR-технологій, занурення стає ще глибшим. [7]

1.3 Засоби розробки в Unreal Engine 5: Blueprint та C++.

В Unreal Engine є два основних засоби розробки: Blueprint та C++. Blueprint - це система візуального програмування. У Blueprint використовуються “ноди” (вузли) замість написання коду достатньо розмістити їх, задати властивості і з’єднати між собою. На рисунку 1.1 наведений приклад алгоритму, написаний за допомогою Blueprints.

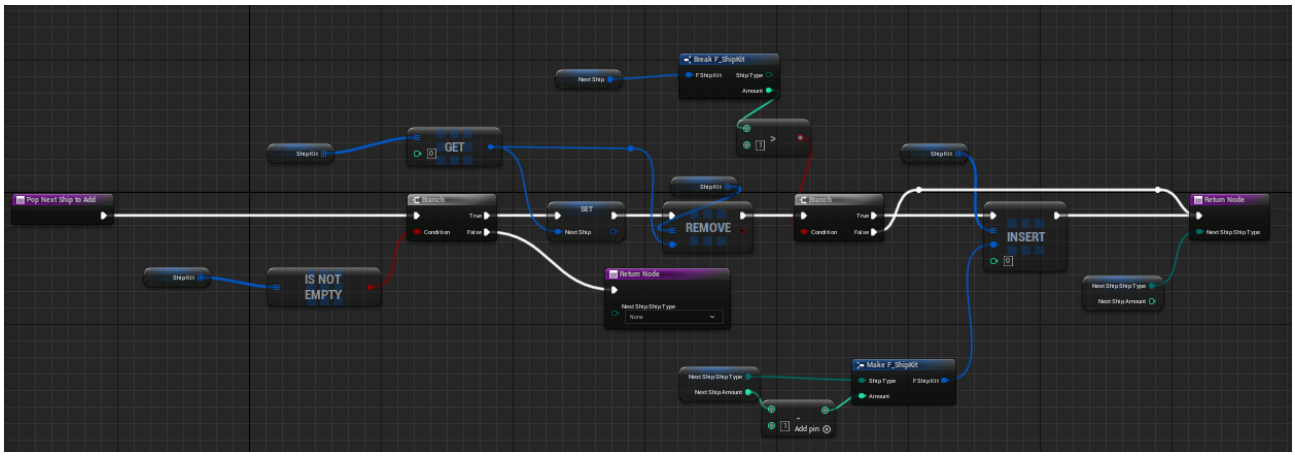

Рисунок 1.1 —Blueprints.

Розробка за допомогою нодів досить обмежена, порівняно з C++, проте значно полегшує читабельність алгоритму і процес розробки. Blueprint може бути використаний для створення складних ігрових логік, анімацій, штучного інтелекту, взаємодії з гравцем та багато іншого. Використання C++ дає програмістам повний контроль над процесом розробки, а також дає більшу ефективність. За допомогою обох методів можна створювати проєкти будь-якої складності, від простих ігор до складних великобюджетних проєктів, віртуальних реальностей та додатків для різних платформ. Вибір між засобами залежить від ваших потреб та рівня знань програмування. Наприклад, для розробки простих ігор достатньо використовувати Blueprint, а для складних ігор або спеціалізованих інструментів може бути кориснішим використовувати C++. Часто методи використовуються разом, наприклад, коли в Blueprint недостатньо функціоналу для вирішення певної задачі, можна використовувати код. Також можна створювати власні ноди, які виконують функції, написані на C++. Це дозволяє спростити процес розробки і забезпечити повну гнучкість у виборі рівня програмування для розв'язання певних задач. В даній курсовій роботі було

досліджено створення гри за допомогою Blueprint, адже розмір проєкту не потребував використання коду, а складність розробки не вимагала такого рівня програмування. Однак, для більш складних програм або специфічних потреб, використання C++ може бути необхідним. В будь-якому випадку, важливо знати обидва засоби розробки, щоб мати можливість вибрати оптимальний метод для вирішення конкретних задач.

1.4 Багатоплатформна підтримка Unreal Engine 5

Unreal Engine 5 підтримує всі сучасні платформи та операційні системи, такі як Windows, macOS, Linux, iOS, Android, PlayStation, Xbox та інші. Тобто розробники можуть розповсюджувати свій продукт на різних платформах, використовуючи один і той самий код. Unreal Engine 5 також підтримує різні API, такі як DirectX, OpenGL, Metal та Vulkan, що дозволяє використовувати їх залежно від платформи, для якої вони розробляються. Unreal Engine 5 може експортувати проєкти в форматі HTML5, що дозволяє запускати гру в браузері без необхідності встановлювати будь-який додатковий софт на комп'ютері користувача. Також рушій має спеціальні інструменти для розробки ігор з підтримкою віртуальної або доповненої реальності. Підтримуються всі сучасні VR пристрої.

1.5 Ресурси для вивчення Unreal Engine 5

Для вивчення Unreal Engine є дуже обширна офіційна документація [8]. Вона містить докладні пояснення та приклади для різних аспектів розробки. Крім того, в документації можна знайти відповіді на багато питань, які можуть виникнути під час роботи з рушієм.

Спільнота Unreal-розробників досить велика, тому існує багато форумів, чатів, курсів та навчальних відеоматеріалів, які можуть допомогти вивчити рушій. Наприклад, на форумі Unreal Engine [9] або в відповідній темі на Reddit [10] можна знайти відповіді на різноманітні запитання, а також побачити проєкти, які розробляють інші користувачі. Крім цього, Unreal Engine має свій власний навчальний ресурс Unreal Online Learning [11], який надає безкоштовний доступ

до багатьох курсів, різного рівня складності. Крім того, є багато онлайн-курсів, які можна знайти на інших платформах.

2. Дослідження актуальності теми

2.1 Аналіз ринку ігор

Згідно з звіту про світовий ринок ігор “The Global Games Market 2022” від Newzoo, ігрова індустрія дуже велика і швидко розвивається. Тільки в 2022 році 3.2 мільярди гравців витратили майже 200 мільярдів доларів, що на 2% більше за 2021. До 2025 року, за прогнозом, кількість гравців зросте до 3,5 мільярда (рисунок 2.1), які згенерують 225 мільярдів доларів (Рисунок 2.2),. Це вражаючи цифри, що свідчить про популярність сфери, отже розробка ігор є важливою і перспективною галуззю, яка привертає увагу інвесторів. [1]

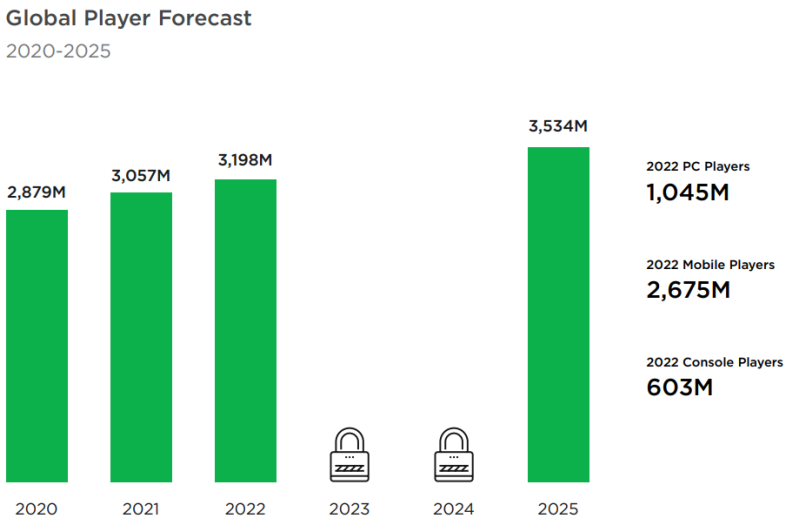

Рисунок 2.1 — Прогноз кількості гравців від Global Games Market Report 2022 [1].

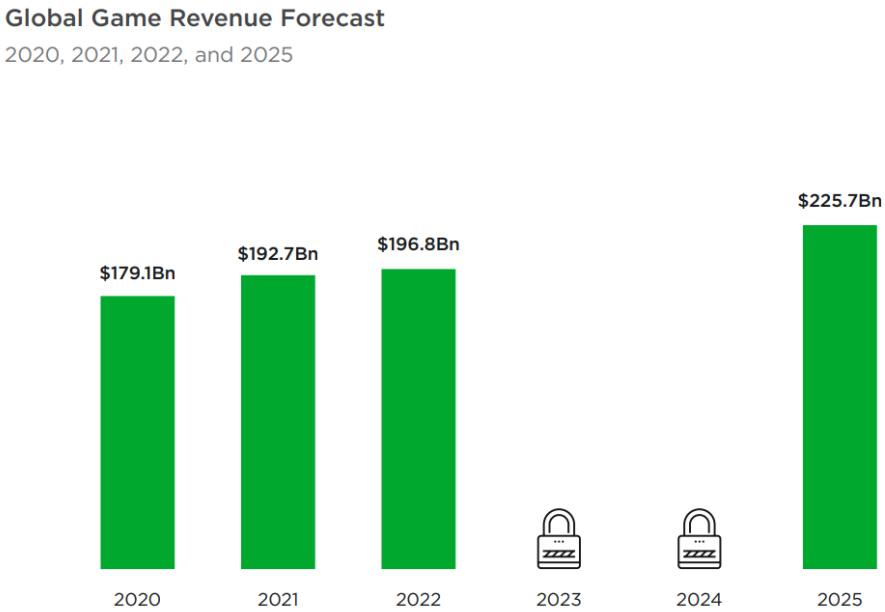

Рисунок 2.2 — Прогноз прибутків від ігор від Global Games Market Report 2022 [1].

2.2 Аналіз існуючих ігор, розроблених на Unreal Engine.

На Unreal Engine 5 розробляються сотні ігор кожен рік. Рушієм користуються як великі студії, для розробки AAA-проєктів, так і розробники-початківці, для створення невеликих симуляторів. 2022 рік вважається одним із успішних для розробників Unreal у всьому світі. За цей рік вийшло багато довгоочікуваних ігор, серед яких пригодницькі бойовики, файтинги, рольові ігри, ігри VR та багато іншого. [12]

Ігри, розроблені на Unreal Engine продаються за вищими цінами, що Unreal є кращим рушієм для масштабних проєктів. 25% ігор на Unreal стартують за ціною $29,99+, порівняно з лише 6% ігор на Unity. Цінову позначку $49.99+ намагаються досягти майже виключно ігри, розроблені на Unreal або на кастомних рушіях. Тим часом, 85% всіх інших рушіїв випускаються за ціною до $19.99. (Рисунок 2.3). Ігри на Unreal займають 15.34% від всіх ігор на 2021 рік у Steam, за версією Game Developer [13] (Рисунок 2.4).

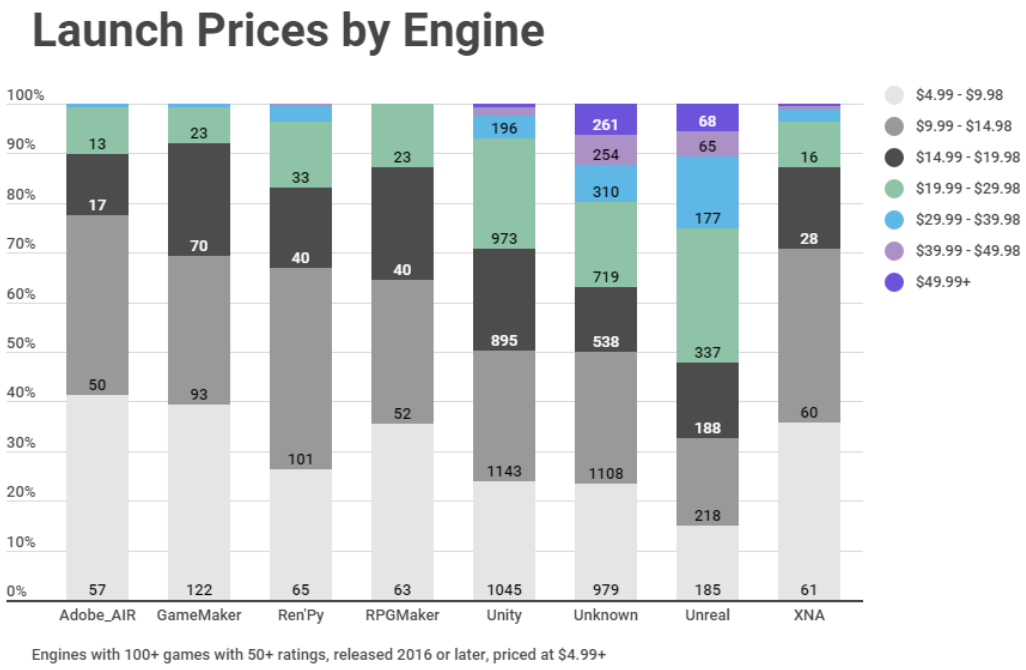

Рисунок 2.3 Ціни на ігри під час випуску [13]

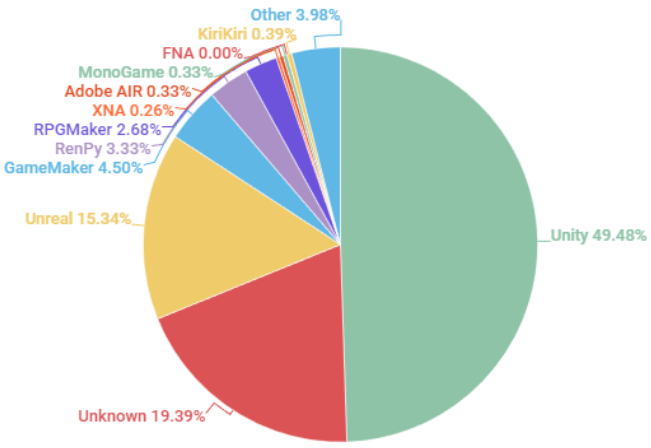

Рисунок 2.4 Відсоток використання рушіїв у Steam станом на 2021 рік [13]

2.2 Порівняння Unreal Engine 5 з головним конкурентом, Unity.

Unity — це головний конкурент Unreal Engine на сьогодні. Обидва рушії мають свої переваги та недоліки. Unity використовує мову програмування C#, яка вважається легкою, має дуже інтуїтивно зрозумілий інтерфейс та просту архітектуру, що дозволяє розробникам створювати ігри швидко та ефективно. Також Unity надзвичайно зручний для розробки 2D та мобільних ігор. Він відомий зручним інтерфейсом, отже підходить для початківців.

У той же час, Unreal більше підходить для масштабних проєктів з якісною, а іноді фотореалістичною графікою. Unreal вважається рушієм для великобюджетних AAA-ігор. А система Blueprint дозволяє створювати складні алгоритми, не знаючи мови програмування, що робить його більш доступним для аніматорів, художників та інших професій.

Візуальні ефекти

Обидва рушії створюють візуальні ефекти дуже високої якості, але візуальні ефекти, які створються в Unreal вважаються кращими, ніж в Unity.

Рендеринг

Unreal підтримує швидший рендеринг, ніж Unity

Анімація

В Unreal інструменти анімацій складніші, бо розраховані на професійних розробників ігор.

Мова програмування

Unreal Engine використовує C++, тоді як Unity — C#.

Інструменти

Більшість інструментів в обох рушіях схожі за функціональністю, але є такі, які треба згадати окремо. Наприклад, штучний інтелект. Система Behavior Tree дозволяє з легкістю створювати складні сценарії поведінки.

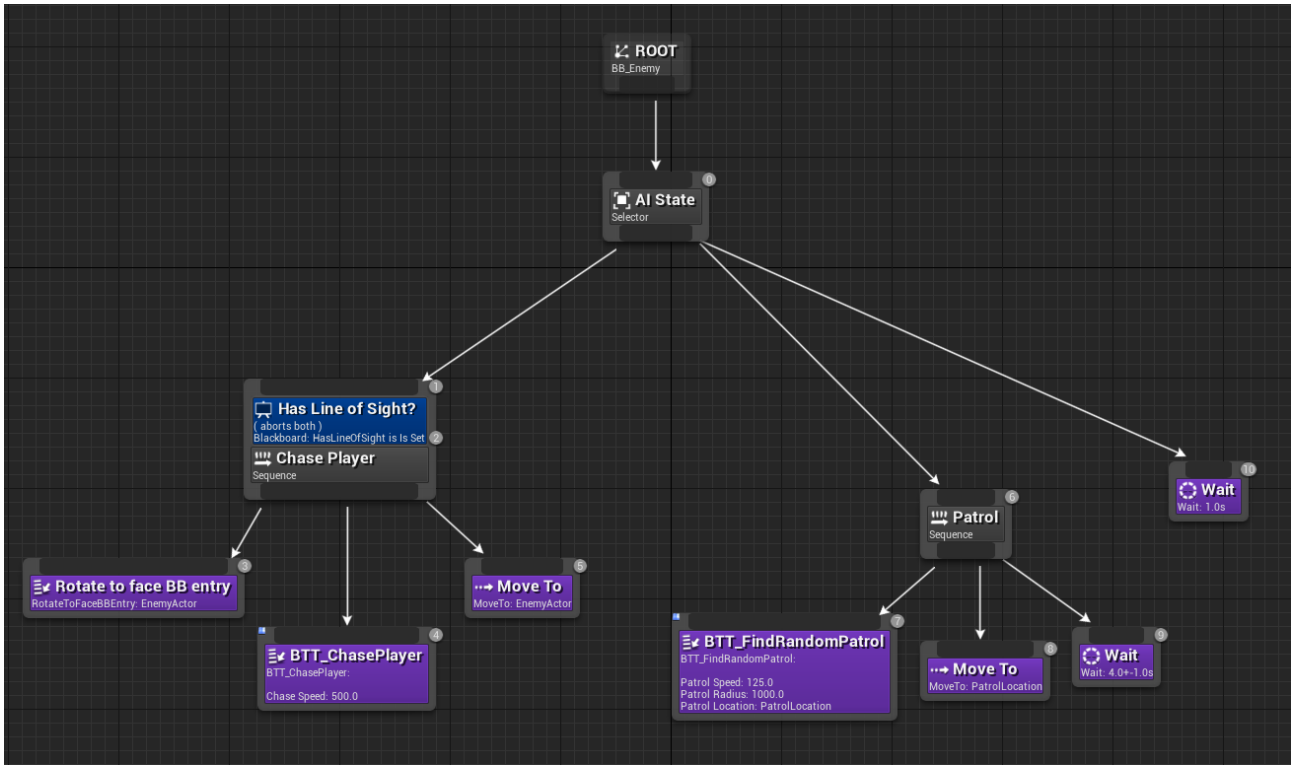

Рисунок 2.5 Behavior Tree [14]

Масштаб проєкту

Unity дуже популярний серед інді-розробників. А Unreal — серед великих студій, На Unreal Marketplace багато додаткових плагінів, а також ресурсів, які можна використовувати для створення прототипів. Коли справа доходить до

великих проєктів, Unreal підходить набагато більше, оскільки великі світи є слабким місцем для Unity.

Ще однією причиною, чому великі розробники обирають Unreal, є те, що він має відкритий вихідний код. Ви можете переписати рушій повністю під вимоги вашого проєкту. Так роблять багато студій, як, наприклад, Epic Games для гри “Fortnite”.[15]

Кодування

Обидва рушії мають подібний візуальний редактор коду (Blueprint в Unreal Engine 4 та Prefab в Unity). Однак вони відрізняються: Blueprint в Unreal - це фрагмент коду, скомпільований кодовим генератором Unreal для отримання дійсного класу C++. Unity's Prefab — це просто інструмент інтерфейсу користувача, який допомагає вам зв'язати різні скрипти разом. В Unreal ви можете створювати гру, використовуючи лише Blueprint, тому що вони є повноцінними частинами коду. Його перевага у продуктивності стає очевидною зі збільшенням складності ігрових сцен.

Продуктивність

Із ростом ігрового світу, Unity збільшує час виконання, тоді як Unreal Engine цього не робить.

Вибір між Unity та Unreal Engine залежить від потреб розробника та вимог конкретного проєкту. Якщо проєкт вимагає більшої графічної складності та фотореалістичної графіки, то Unreal Engine може бути кращим вибором. Але якщо метою є створення меншої гри, зокрема мобільної чи 2D-ігри, то Unity може бути більш вдалим вибором. [16]

3. Реалізація гри на Unreal Engine 5

3.1 Опис гри

Основне завдання — створити гру “Морський бій” на на Unreal Engine 5, використовуючи систему нодів Blueprint.

Гра умовно поділяється на 3 етапи:

- **Меню.** Меню — це перше, що бачить користувач. На екрані з’являється дві кнопки: “*Почати гру*” і “*Вихід*”. Після натискання на першу, меню запропонує обрати складність: “*Спрощена*” і “*Звичайна*”. Після натискання на другу, гра закривається.
- **Розміщення кораблів.** Після вибору складності, користувачу треба розмістити свої кораблі на ігровому полі 10*10. За допомогою пересування миші можна обрати розташування корабля. За допомогою кнопки “R” можна обертати корабель (на 90 градусів за 1 клік). Якщо обране розташування корабля неможливе, корабель підсвічується червоним, в протилежному випадку — зеленим. За допомогою лівої клавіші миші можна поставити корабель, якщо це можливо. Також, доступна функція випадкового розташування відразу всього флоту на клавішу “G”.
- **Битва.** Після розташування кораблів починається “Битва” з штучним інтелектом. З’являється пусте поле ворога, по якому можна “стріляти” за допомогою миші. Штучний інтелект може працювати за двома сценаріями: на спрощеній складності кожна клітинка для пострілу буде вибиратися випадково, на звичайній — вибір клітинки буде залежати від минулих влучань. На звичайній складності поведінка штучного інтелекту подібна до людської. Інтелект стріляє навмання, поки не влучить, після влучання ШІ намагається зрозуміти позицію корабля і буде намагатися його знищити. Після перемоги чи поразки, користувач повертається на етап меню. Він може або почати ще одну гру, або закрити додаток.

3.2 Реалізація. Огляд класів.

Почнемо з основних класів

1. *BP_MainCharacter*. Клас персонажу. Дочірній клас вбудованого класу “Character”. Містить стандартний клас-камеру і клас BP_Clipboard () з полем ворога (Рисунок 3.1.1). Також клас містить логіку з пересування BP_Clipboard (Рисунок 3.1.2).

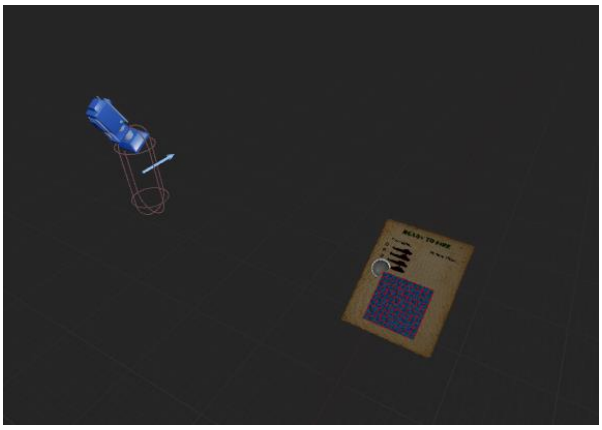

Рисунок 3.1.1 BP_MainCharacter

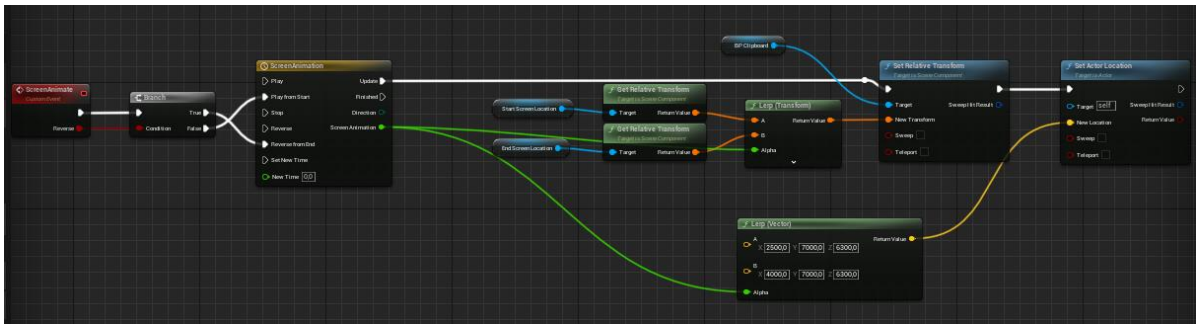

Рисунок 3.1.1 Логіка класу BP_MainCharacter

2. *BP_MainPlayerController*. Клас контролера. Дочірній клас вбудованого класу “PlayerController”. Містить в собі компонент “AC_FieldInteraction”(3).

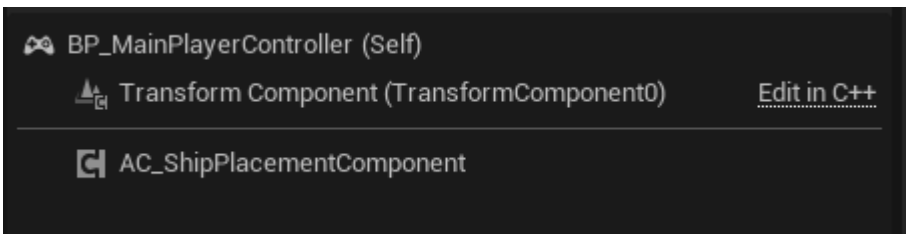

Рисунок 3.2.1 BP_MainPlayerController.

3. AC_FieldInteraction. Дочірній клас вбудованого класу “ActorComponent”. Містить налаштування керування. “Читає” необхідні натискання клавіш та повідомляє всі необхідні класи про зміни (Рисунок 3.3.1). Кожен тік оновлює

позицію миші за допомогою функції “UpdateMousePosition” (Рисунок 3.3.3), вираховує координату клітини на полі і повідомляє про це поле (Рисунок 3.3.2). Функції “UpdateMousePosition” надсилає промінь від камери у її напрямку і чекає на перетин з полем для гри. Після цього по координатам поля і координатам перетину проміння знаходить координату клітинки. Функція повертає знайдене поле і координати клітинки.

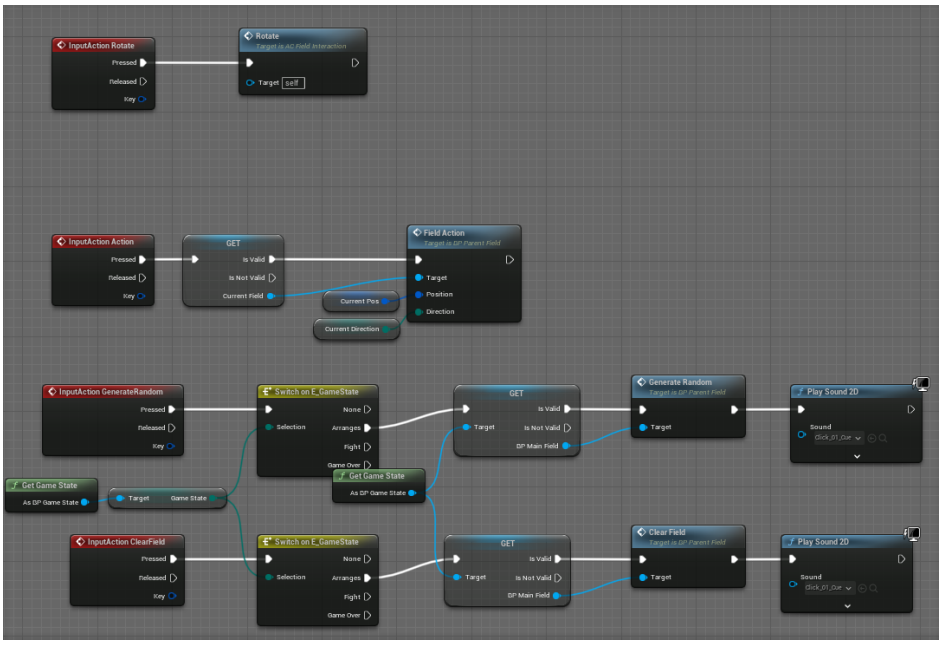

Рисунок 3.3.1 Логіка класу AC_FieldInteraction.

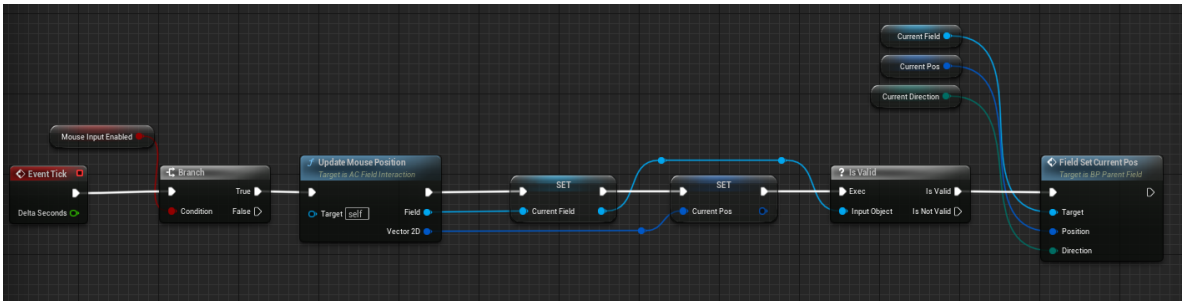

Рисунок 3.3.2 Логіка класу AC_FieldInteraction.

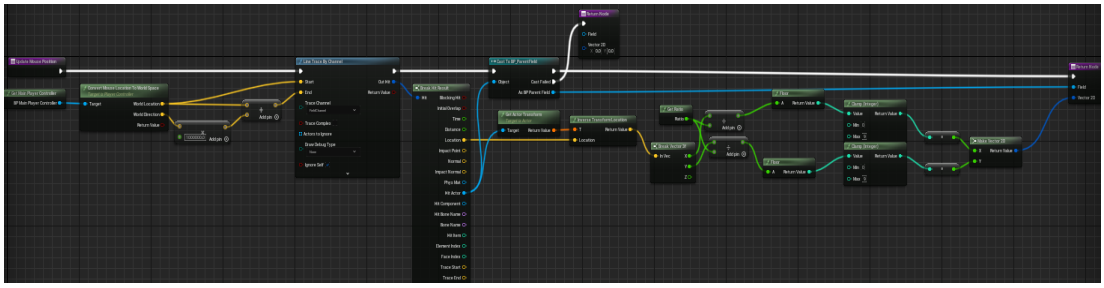

Рисунок 3.3.2 Функція UpdateMousePosition.

4. *BP_FunctionLibrary*. Бібліотека функцій. Дочірній клас вбудованого класу “BlueprintFunctionLibrary”. Містить основні універсальні функції, які використовуються у різних класах проєкту (Рисунок 3.4.1).

   Функція “PossibleToPlace” перевіряє, чи можна встановити корабель. (Рисунок 3.4.2).

   Функція “PosIsValid” перевіряє чи належать координати ігровому полю (Рисунок 3.4.3).

   Функція “GetAllCordinatesFromFirst” із однієї координати, типу корабля і напряму знаходить усі координати корабля (Рисунок 3.4.4).

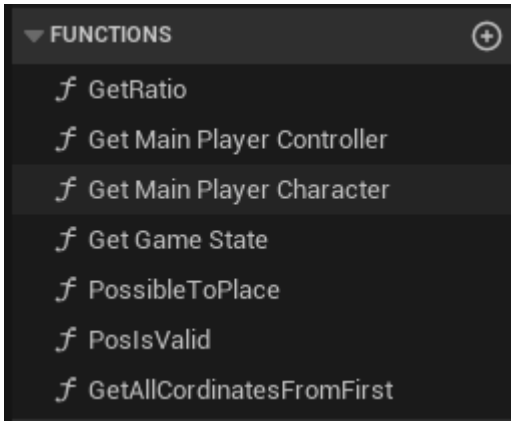

Рисунок 3.4.1 Функції класу “BlueprintFunctionLibrary”.

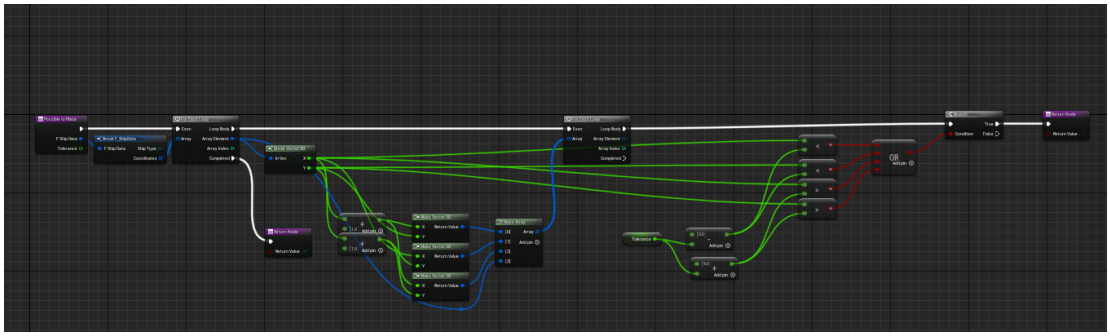

Рисунок 3.4.2 Функція “PossibleToPlace”.

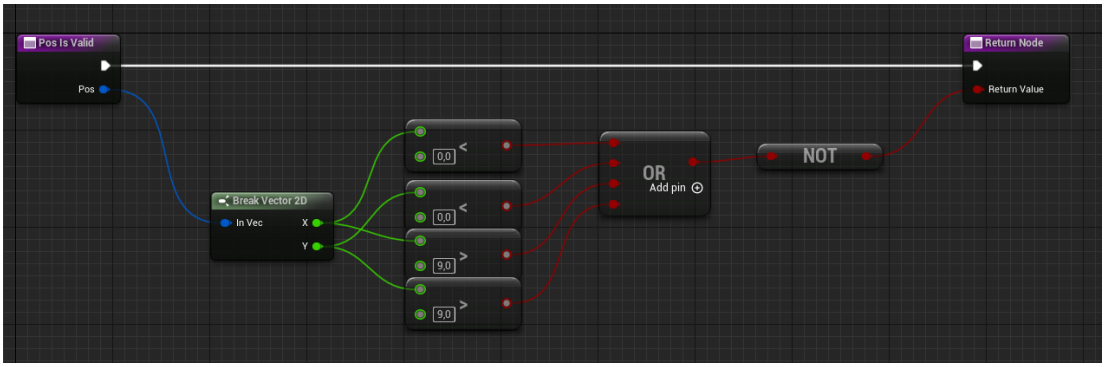

Рисунок 3.4.3 Функція “PosIsValid”.

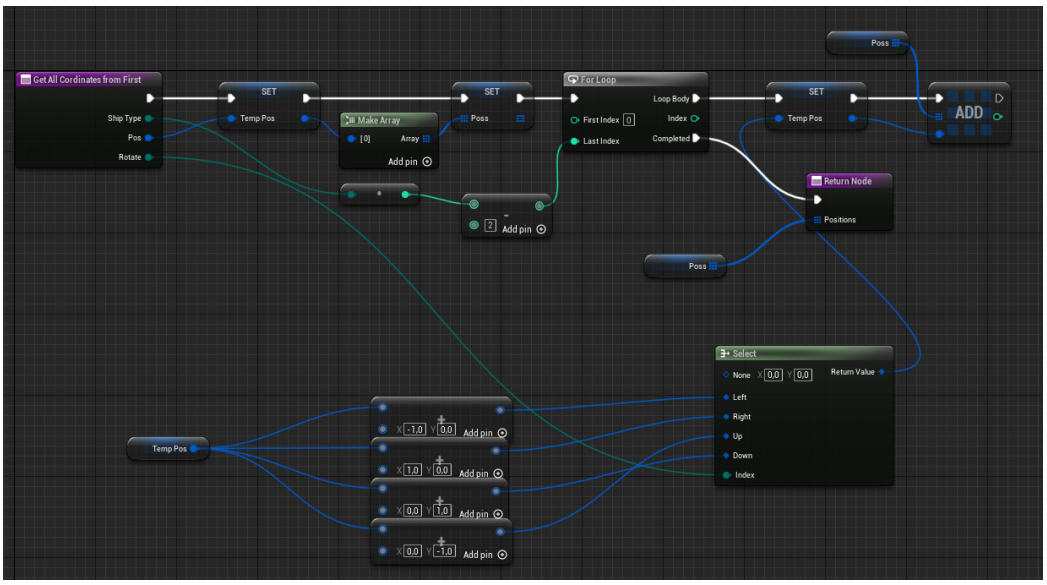

Рисунок 3.4.4 Функція "GetAllCordinatesFromFirst".

5. *BP_GameState.* Дочірній клас вбудованого класу "GameStateBase". Саме цей клас відповідає за послідовність гри. Клас на початку встановлює значення змінної "GameState" на "Arranges", це етап розташування кораблів. Після цього створюється об'єкт класу "BP_MainField" (Рисунок 3.5.1). Після успішного розташування флота, спрацює делегат "ShipsPlaced". Тепер створюється полу супротивника, а "GameState" встановлюється на "Fight" (Рисунок 3.5.2-3.5.3).

   Після виклику делегата "OnShot" обирається чия черга грати (Рисунок 3.5.4). Якщо черга штучного інтелекту, то викликається функція "AI Fire" у класі "AC_ShotAI" (Рисунок 3.5.5). Після виклику делегата " AllShipsDestroyed" "GameState" встановлюється на "GameOver", на екрані ствоюється віджет "WBP_GameOverMenu" (Рисунок 3.5.6).

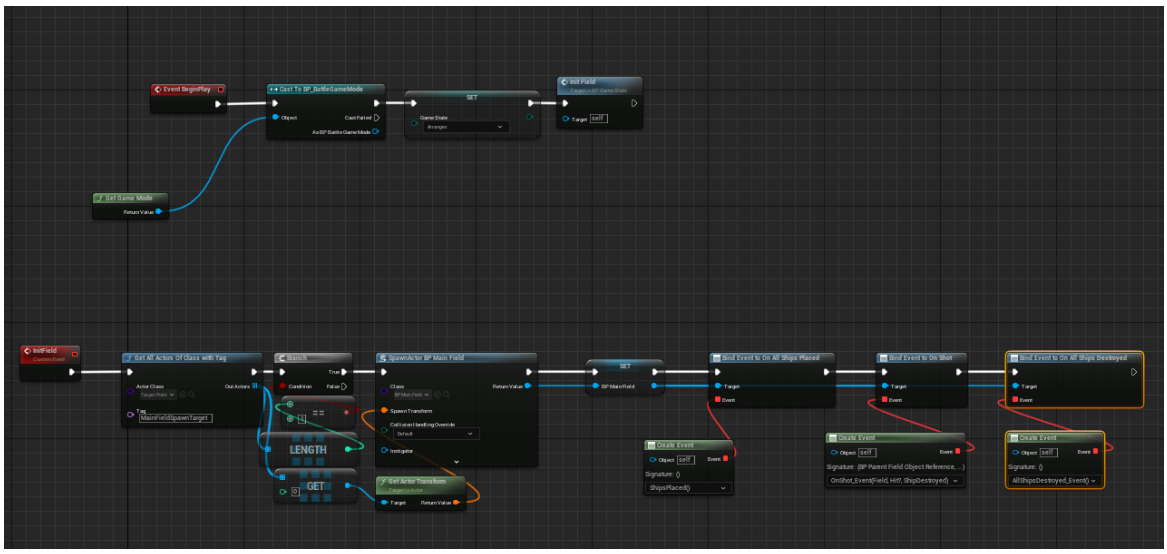

Рисунок 3.5.1 Логіка класу “BP_GameState”.

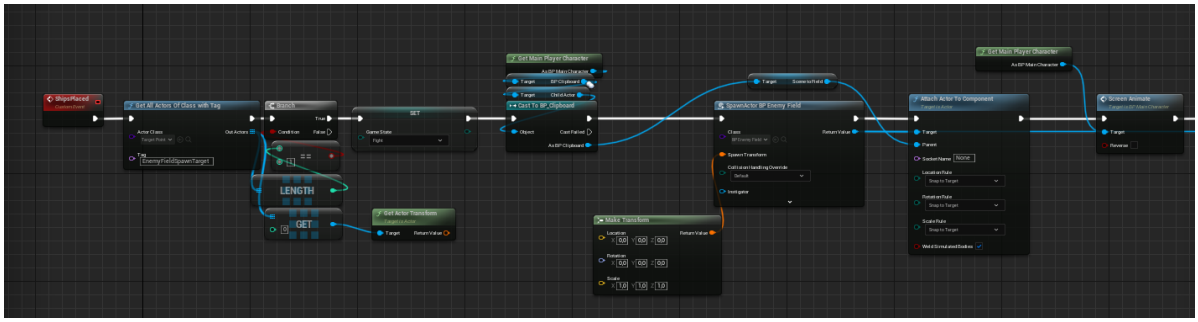

Рисунок 3.5.2 Логіка класу “BP_GameState”.

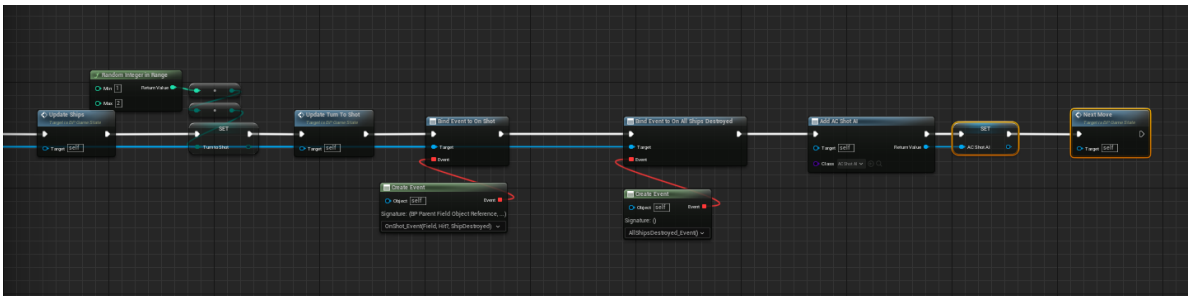

Рисунок 3.5.3 Логіка класу “BP_GameState”.

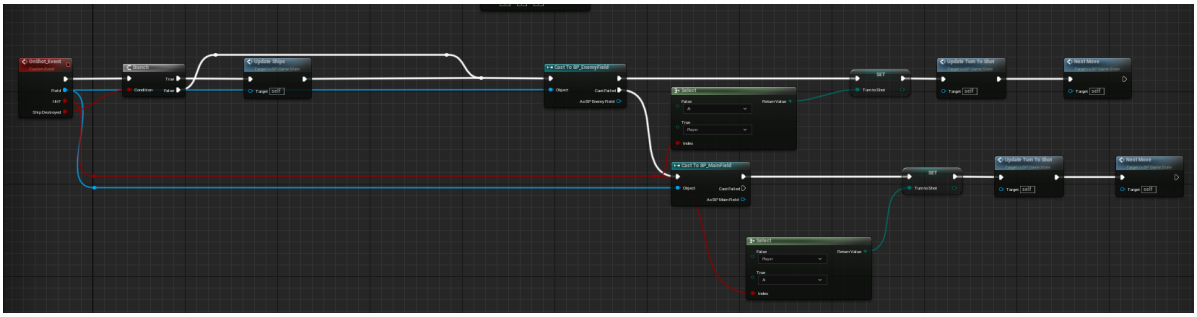

Рисунок 3.5.4 Логіка класу “BP_GameState”.

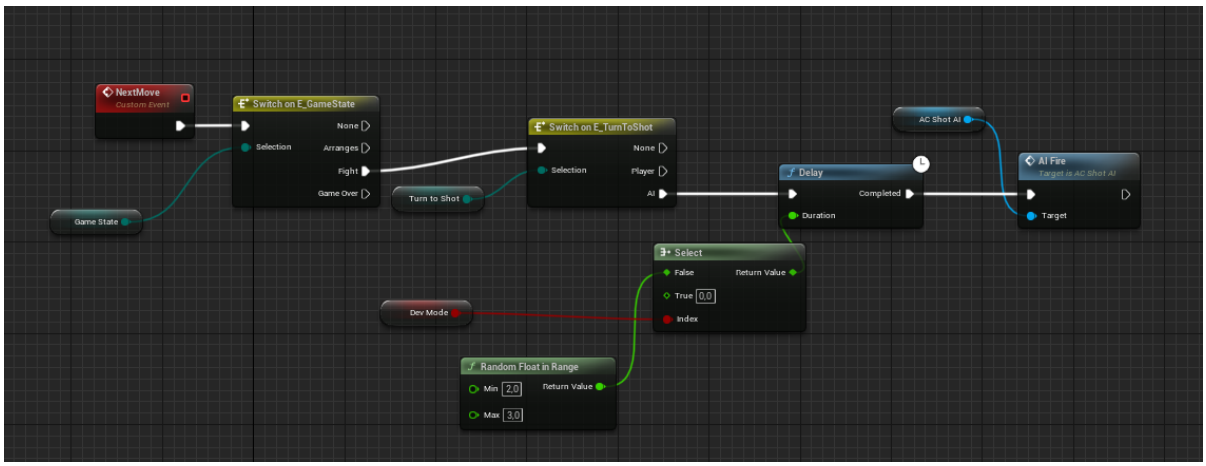

Рисунок 3.5.5 Логіка класу "BP_GameState".

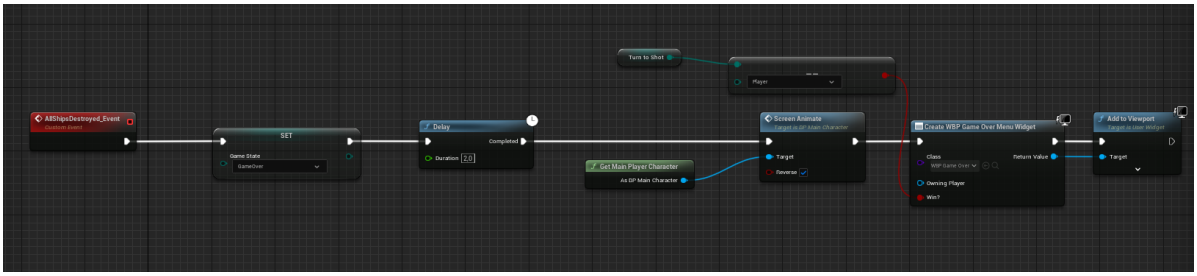

Рисунок 3.5.6 Логіка класу "BP_GameState".

Рисунок 3.5.7 Змінні класу "BP_GameState".

6. *BP_ParentField.* Батьківський клас ігрового поля. У ConstructionScript задається сітка поля (Рисунок 3.6.1). На початку виконання клас за допомогою функції "ImportShipKit" (Рисунки 3.6.2-3.6.3) імпортує список кораблів для гри із таблиці "DT_ShipKit" (Рисунок 3.6.4).

   В класі BP_ParentField багато функцій, які будуть використовуватися в дочірніх класах.

   Функції:

   - *"FieldSetCurrentPos"* викликається при наведені курсора на клітинку поля. В дочірніх класах переписується (Рисунок 3.6.5).
   - *"Field Action"* викликається при натискані на клітинку поля. В дочірніх класах переписується (Рисунок 3.6.6).

- *“Get Neighbours”* знаходить усі клітинки “сусіди” з даною (Рисунок 3.6.7).
- *“Possible To Place On Field”* перевіряє чи нема клітинок даного корабля серед заповнених, тобто перевіряє, чи можливо поставити корабель (Рисунок 3.6.8).
- *“Clear Field”* Очищує поле від кораблів. Заново імпортує список не розставлених кораблів (Рисунок 3.6.9).
- *“Pop Next Ship to Add”* Вилучає і повертає один корабель із списку не розставлених кораблів (Рисунок 3.6.10).
- *“SetCurrentShip”* встановлює змінну поточного корабля та викликає делегат “OnAllShipsPlaced”, якщо поточного корабля немає (Рисунок 3.6.11).
- *“Generate Random”* знаходить випадкову розстановку флоту (Рисунок 3.6.12-3.6.13).
- *“Place Ship”* Додає клітинки, які займає посталений корабель до масиву, додає дані корабля до відповідного масиву (Рисунок 3.6.14).
- *“Get Location From Field”* конвертує позицію на полі в локальні координати (Рисунок 3.6.15).
- *“Get Ship Location From Field”* конвертує позицію корабля на полі в локальні координати (Рисунок 3.6.16).
- *“Shot”* робить “постріл”: перевіряє координати, створює відповідну мітку в залежності від влучання, створює ефекти, викликає делегат “OnShot”, повертає успішність пострілу, чи є влучання, чи є знищення корабля (Рисунок 3.6.17-3.6.19).
- *“Update Ships”* Відмічає всі сусідні клітинки знищеного корабля (Рисунок 3.6.20-3.6.21).
- *“Spawn Effects”* Створює ефекти на координатах. В дочірніх класах переписується (Рисунок 3.6.22).

Змінні (Рисунок 3.6.23):

- *Field* зберігає створені стінки поля.
- *FilledCells* зберігає клітинки, на яких є кораблі.
- *DataShipKit* містить таблицю кораблів.
- *Ship Kit* зберігає не розставлені кораблі.
- *CurrentShip* зберігає вибраний корабель.
- *Ships* зберігає дані створених кораблів.
- *CheckedCells* зберігає перевірені клітинки поля.
- *Hits* зберігає створені мітки влучання.

Диспатчери (делегати):

- OnAllShipsPlaced викликається, коли користувач розташував всі кораблі.
- OnAllShipsDestroyed викликається, коли всі кораблі однієї з команд був знищені.
- OnShot викликається, коли один з гравців “стріляє”.

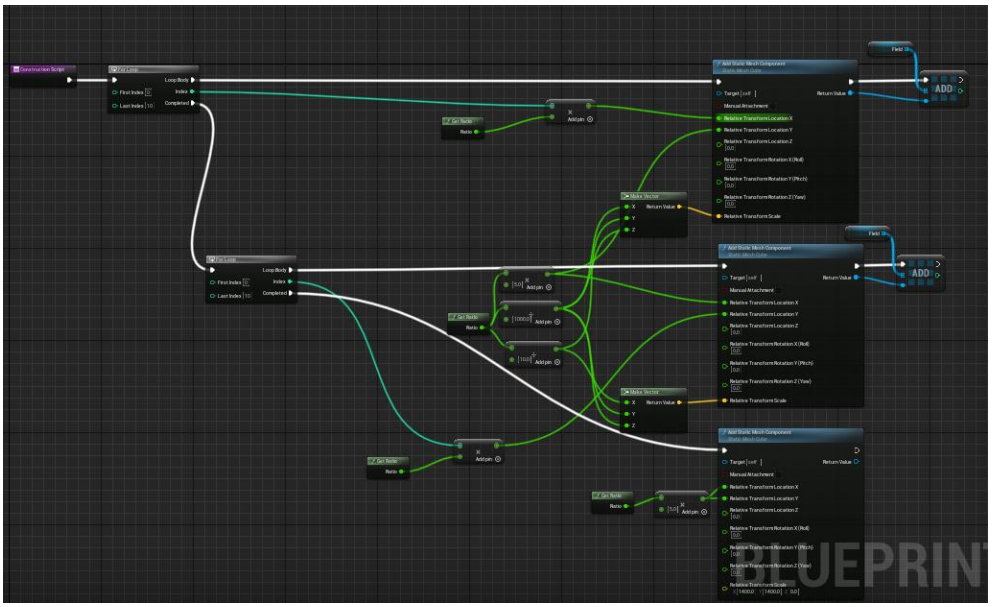

Рисунок 3.6.1 Логіка класу “BP_ParentField”.

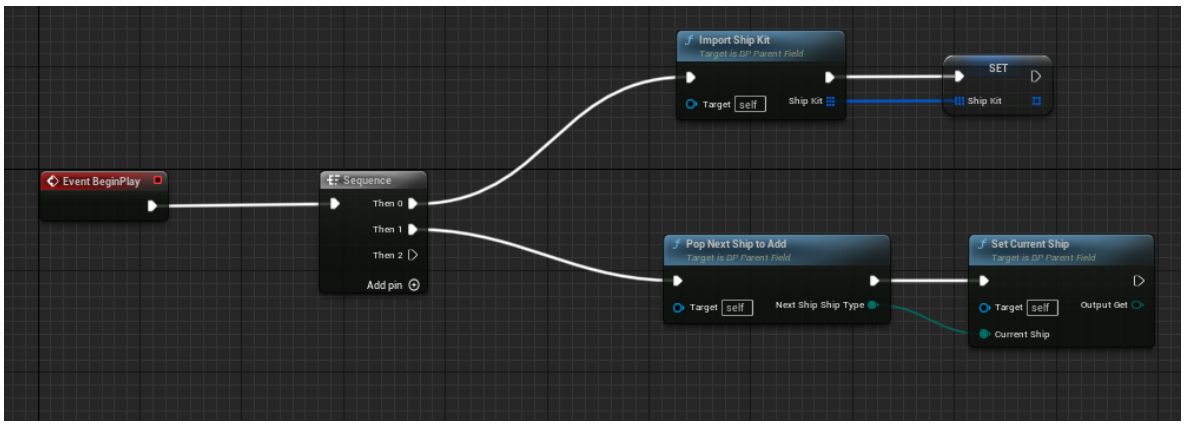

Рисунок 3.6.2 Логіка класу "BP_ParentField".

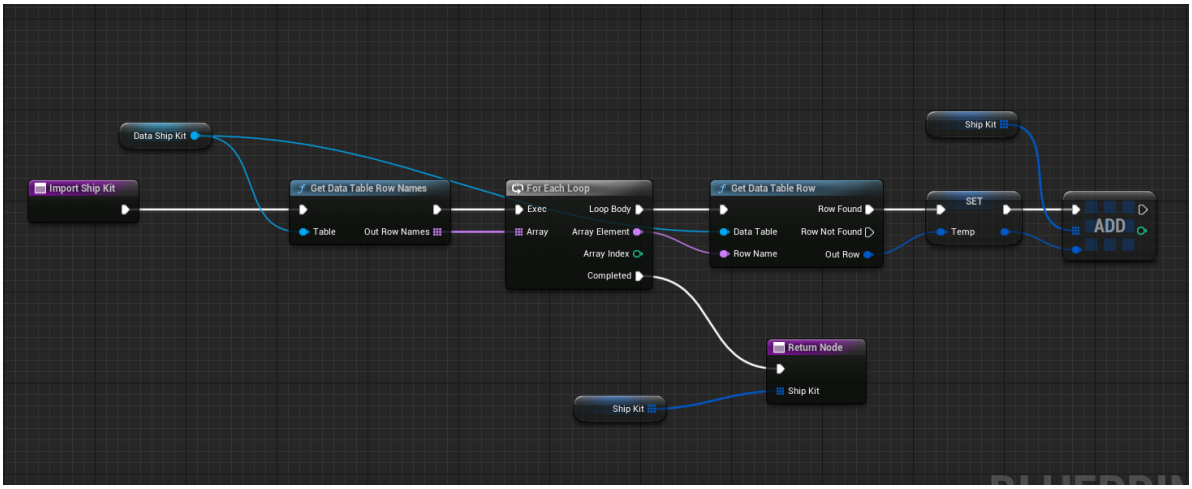

Рисунок 3.6.3 Логіка класу "BP_ParentField".

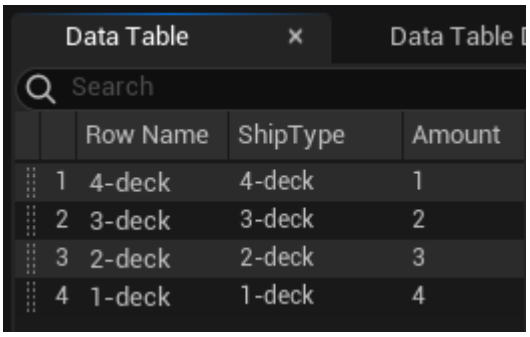

Рисунок 3.6.4 Таблиця "DT_ShipKit".

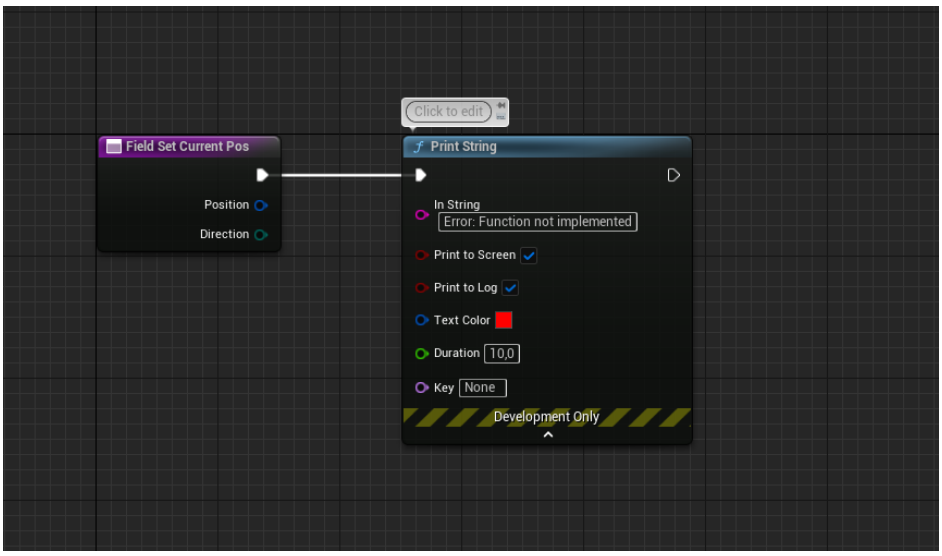

Рисунок 3.6.5 Функція "Field Set Current Pos" класу " BP_ParentField".

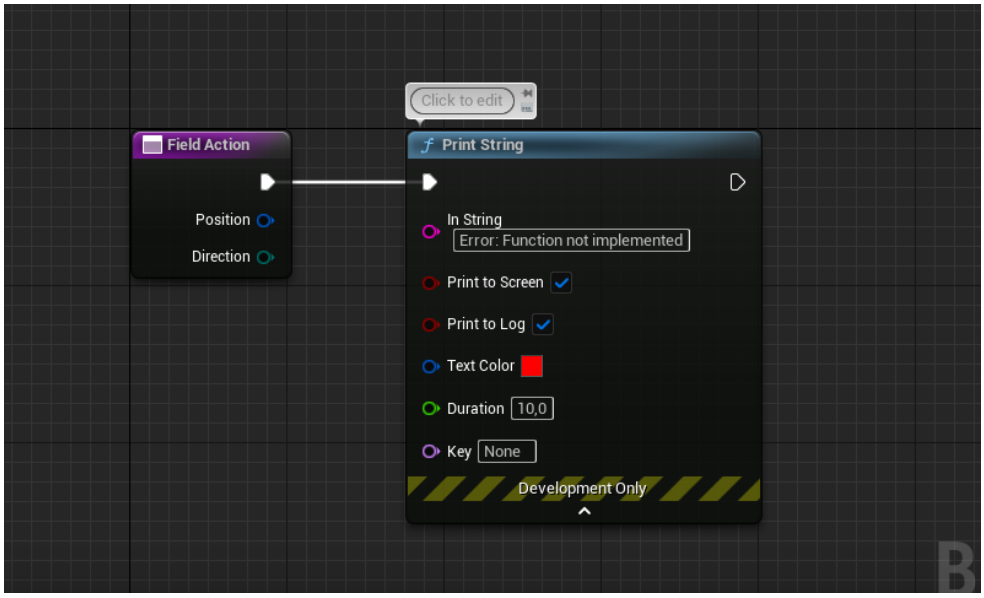

Рисунок 3.6.6 Функція "Field Action" класу " BP_ParentField".

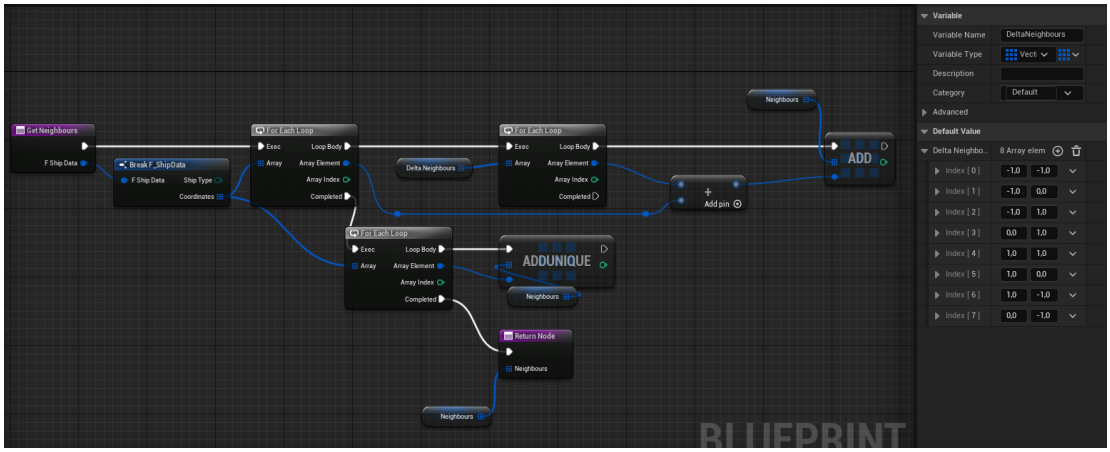

Рисунок 3.6.7 Функція "Get Neighbours" класу " BP_ParentField".

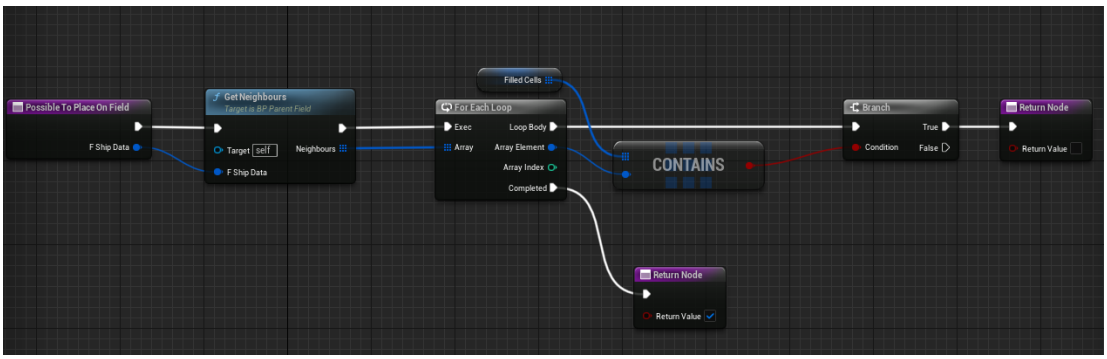

Рисунок 3.6.8 Функція "Possible To Place On Field" класу "BP_ParentField".

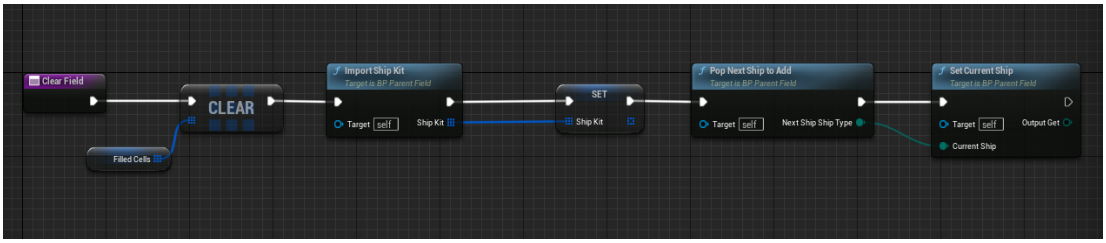

Рисунок 3.6.9 Функція "Clear Field" класу "BP_ParentField".

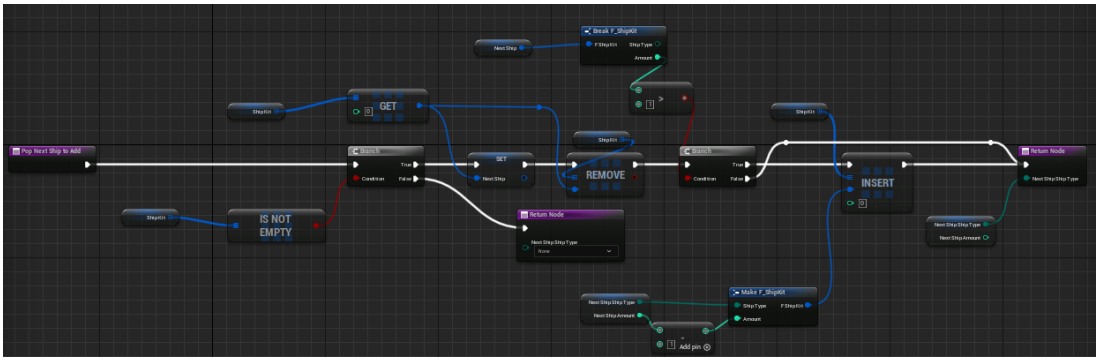

Рисунок 3.6.10 Функція "Pop Next Ship to Add" класу "BP_ParentField".

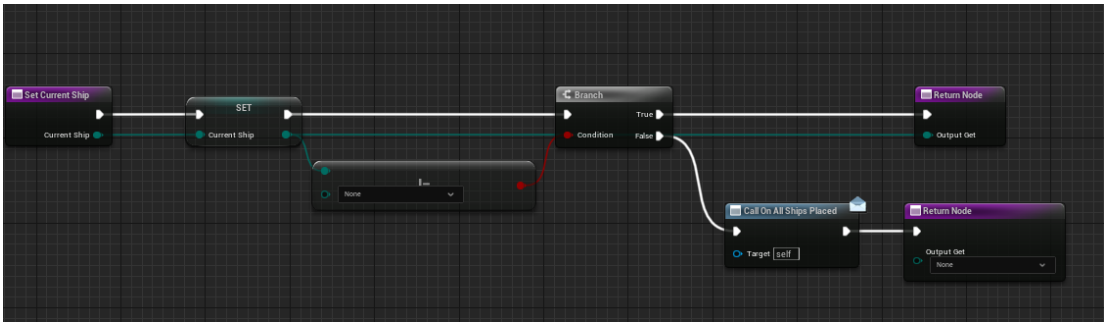

Рисунок 3.6.11 Функція "SetCurrentShip" класу "BP_ParentField".

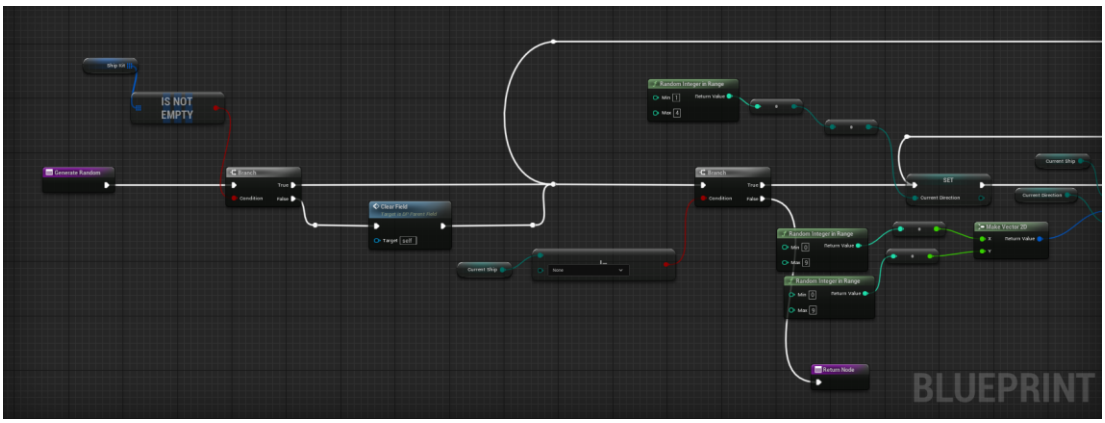

Рисунок 3.6.12 Функція "Generate Random" класу "BP_ParentField".

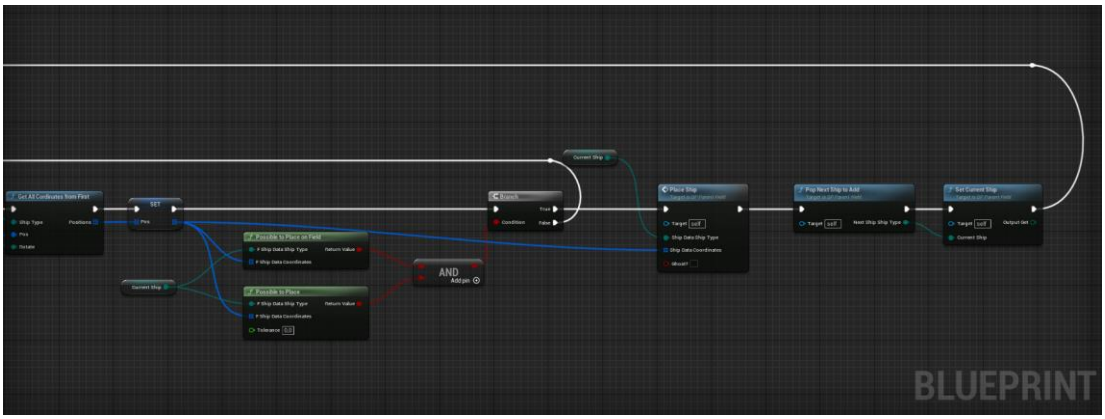

Рисунок 3.6.13 Функція "Generate Random" класу "BP_ParentField".

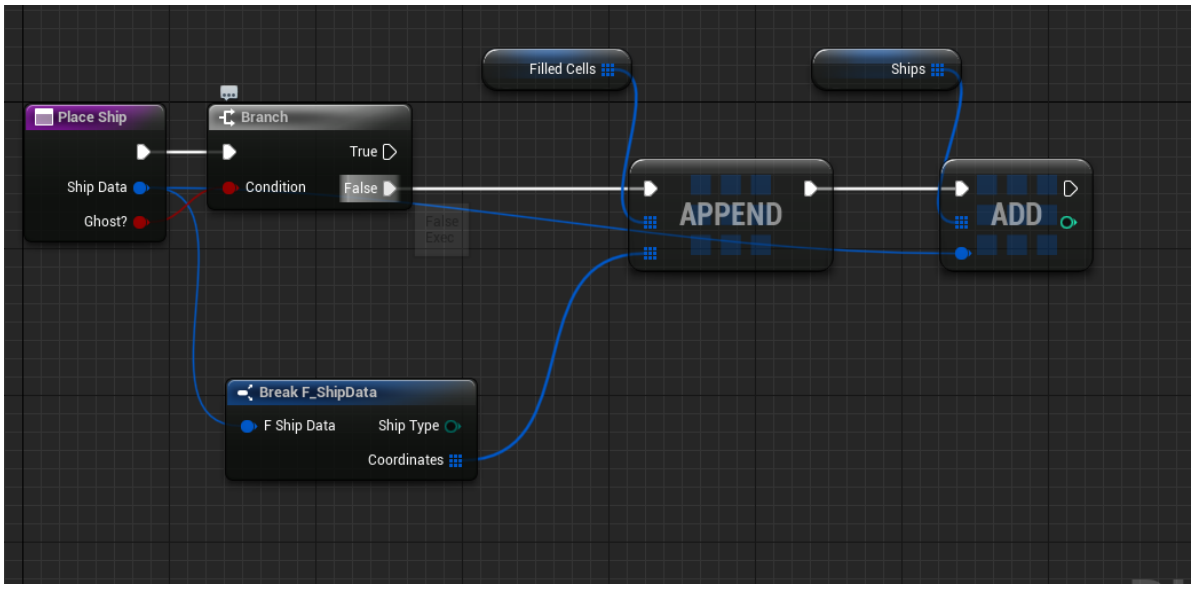

Рисунок 3.6.14 Функція "Place Ship" класу "BP_ParentField".

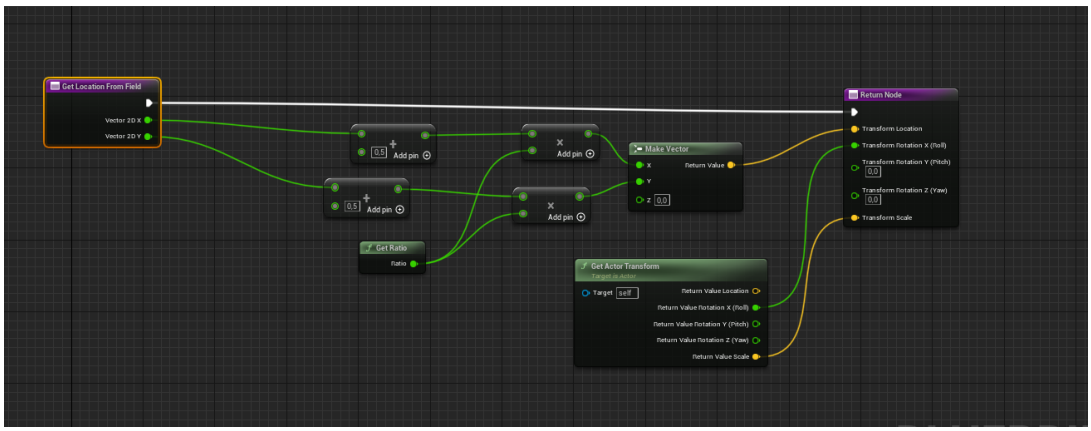

Рисунок 3.6.15 Функція "Get Location From Field" класу "BP_ParentField".

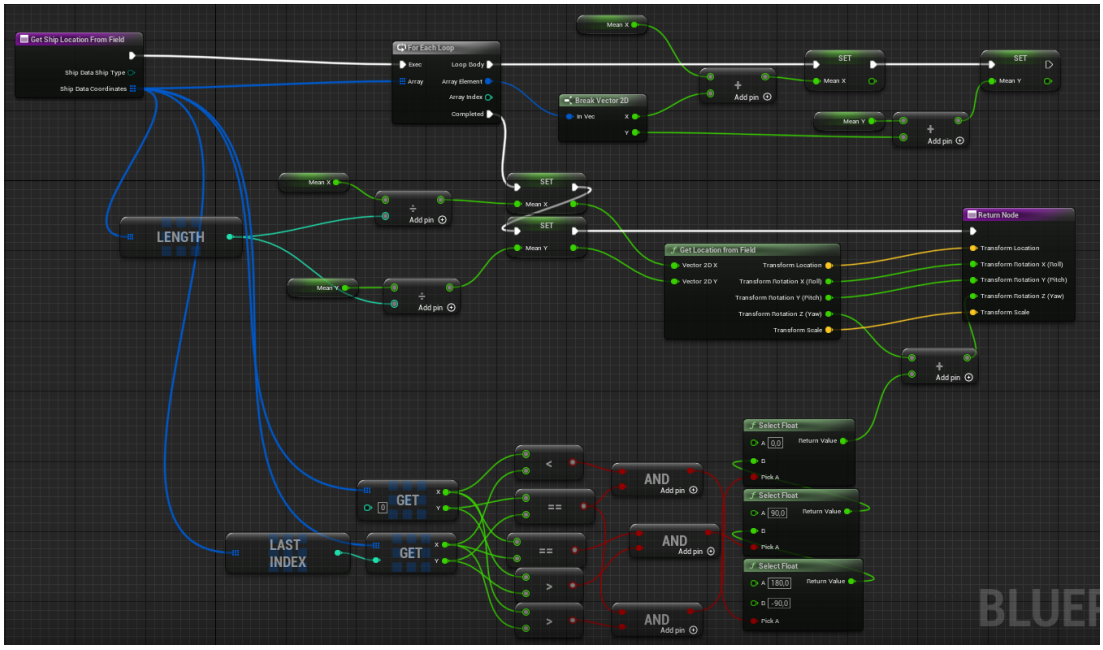

Рисунок 3.6.16 Функція "Get Ship Location From Field" класу "BP_ParentField".

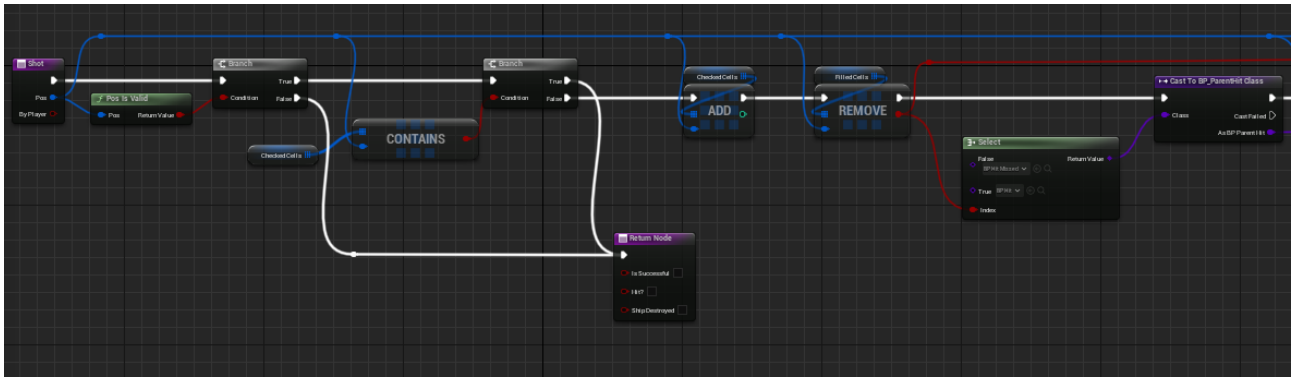

Рисунок 3.6.17 Функція "Shot" класу "BP_ParentField".

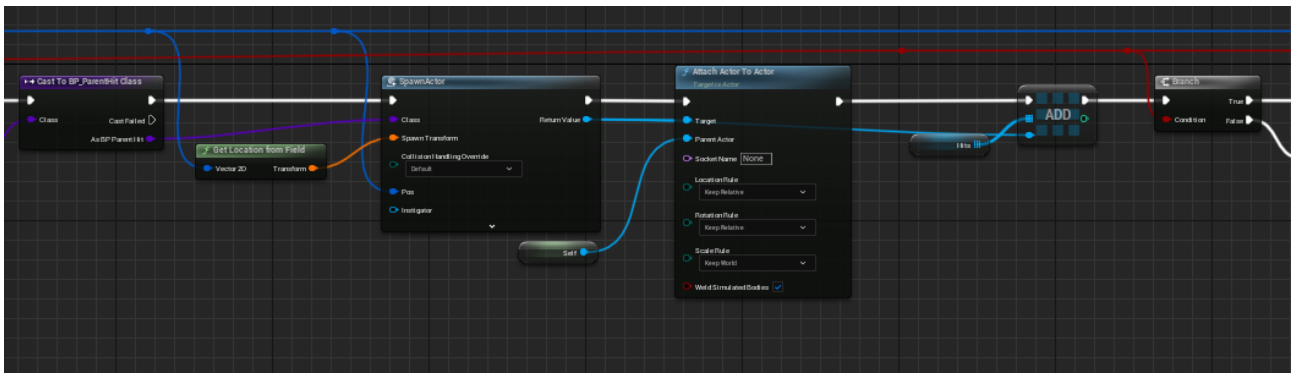

Рисунок 3.6.18 Функція "Shot" класу "BP_ParentField".

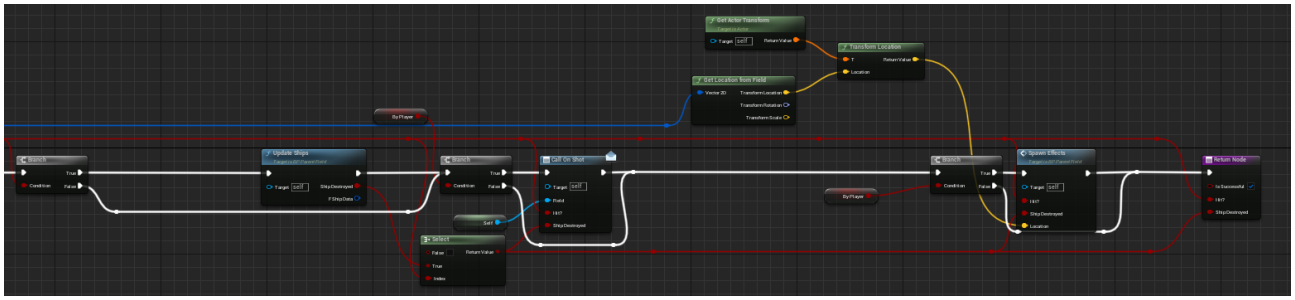

Рисунок 3.6.19 Функція "Shot" класу "BP_ParentField".

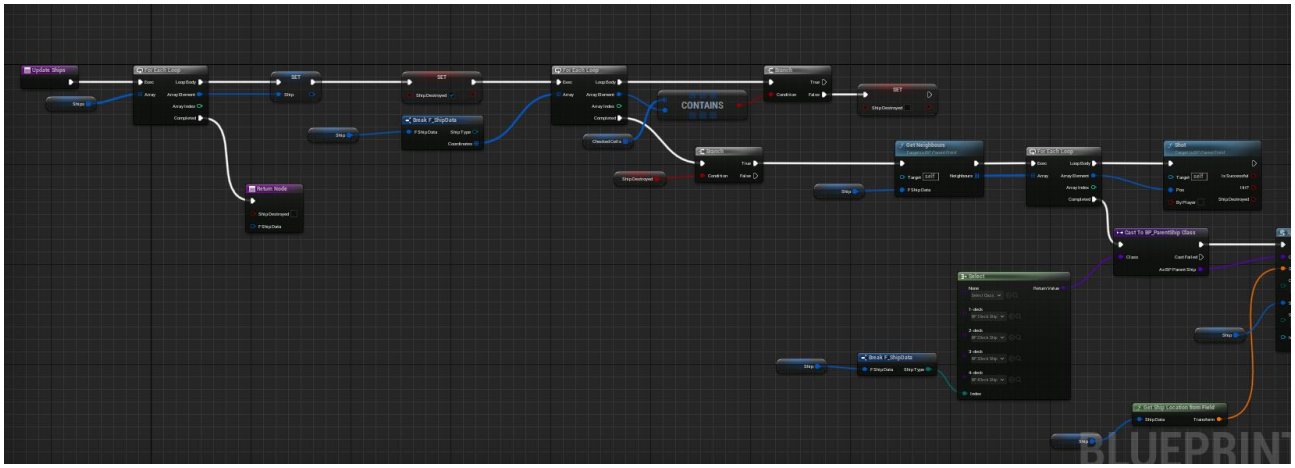

Рисунок 3.6.20 Функція "Update Ships" класу "BP_ParentField".

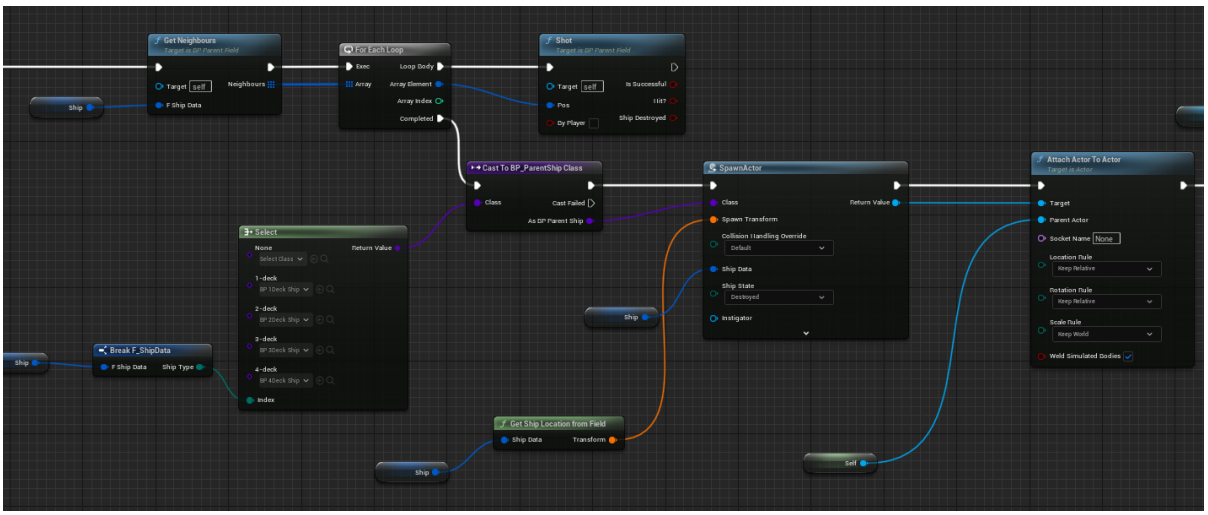

Рисунок 3.6.21 Функція "Update Ships" класу "BP_ParentField".

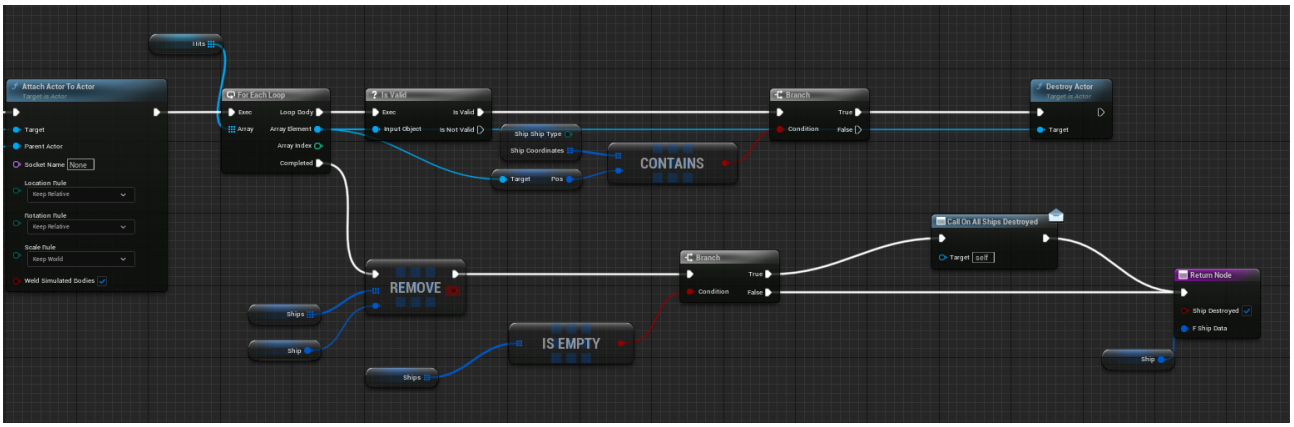

Рисунок 3.6.22 Функція "Update Ships" класу "BP_ParentField".

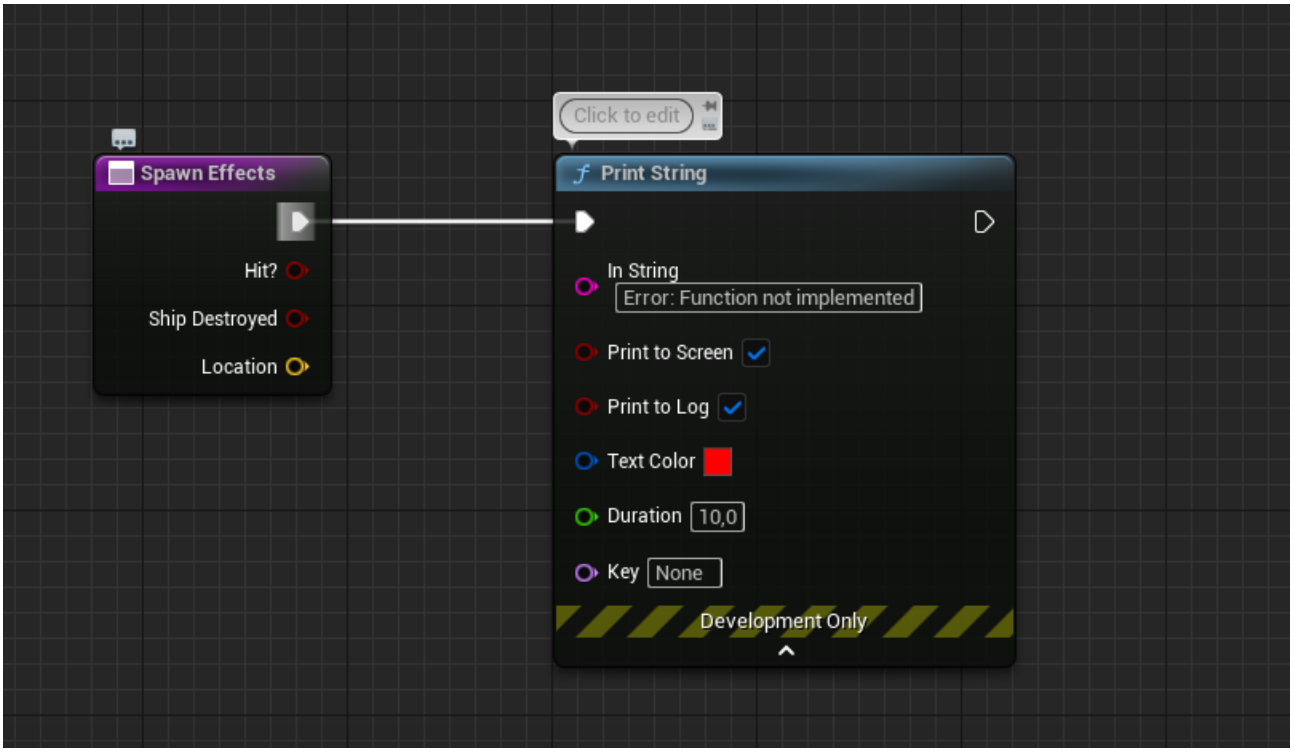

Рисунок 3.6.22 Функція "Spawn Effects" класу "BP_ParentField".

Рисунок 3.6.23 Змінні класу "BP_ParentField".

Рисунок 3.6.24 Диспатчери класу "BP_ParentField".

7. *BP_MainField.* Клас наслідується від класу "BP_ParentField".

Функції:

- *"FieldSetCurrentPos"* Перевизначає функцію батьківсьго класу. Розміщує "прозорий" корабля на полі (Рисунок 3.7.1).
- *"FieldAction"* Перевизначає функцію батьківсьго класу. Розміщує корабель на полі (Рисунок 3.7.2).
- *"ClearField"* Перевизначає функцію батьківсьго класу. Викликає функцію батьківського класу, видаляє створені кораблі (Рисунок 3.7.3).
- *"SetCurrentShip"* Перевизначає функцію батьківсьго класу. Викликає функцію батьківського класу, оновлює розташування прозорого корабля (Рисунок 3.7.4).
- *"PlaceShip"* Перевизначає функцію батьківсьго класу. Викликає функцію батьківського класу, створює корабель (Рисунок 3.7.5).
- *"Spawn Effects"* Перевизначає функцію батьківсьго класу. Створює ефекти (Рисунок 3.7.6).

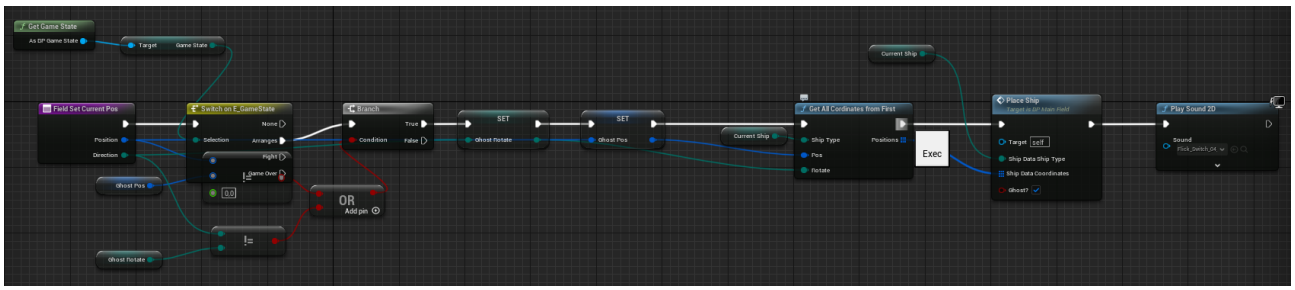

Рисунок 3.7.1 Функція "Field Set Current Pos" класу "BP_MainField".

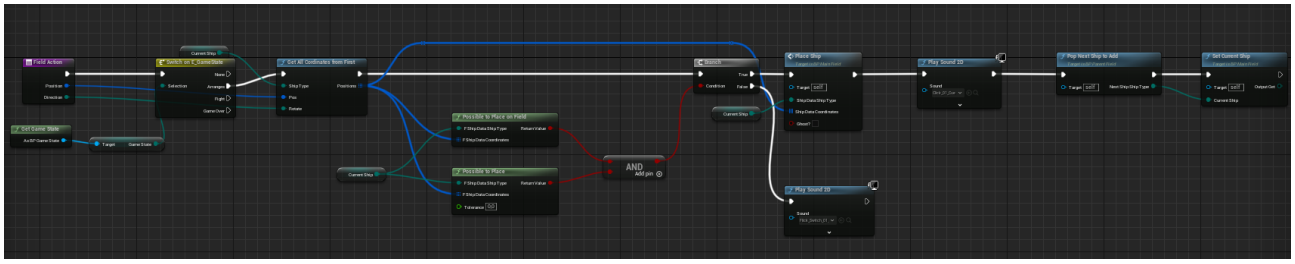

Рисунок 3.7.2 Функція "Field Action" класу "BP_MainField".

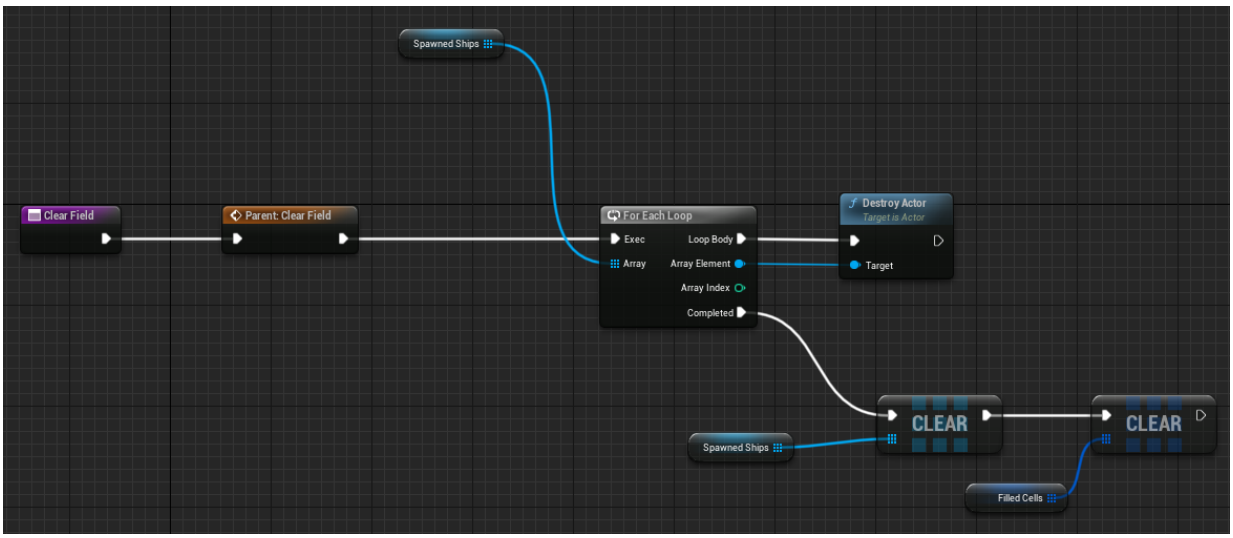

Рисунок 3.7.3 Функція "Clear Field" класу "BP_MainField".

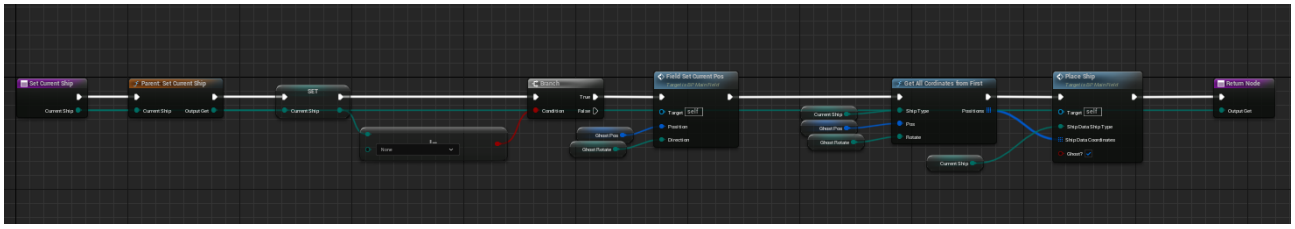

Рисунок 3.7.4 Функція "SetCurrentShip" класу "BP_MainField".

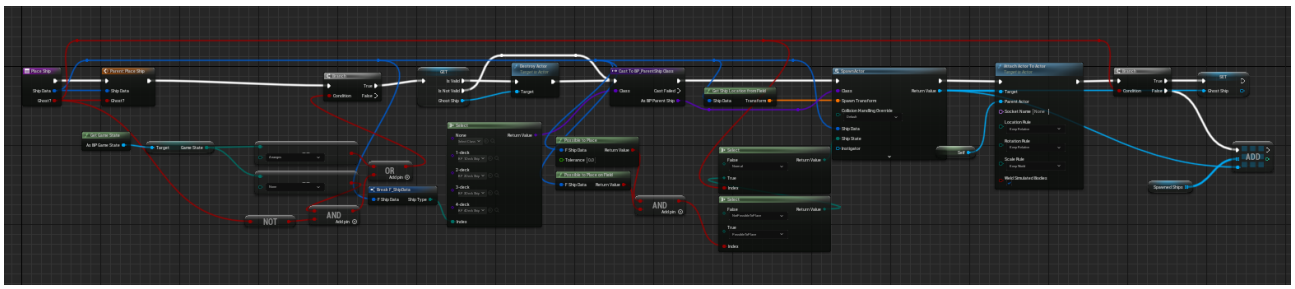

*Рисунок 3.7.5 Функція "*PlaceShip*" класу "BP_MainField".*

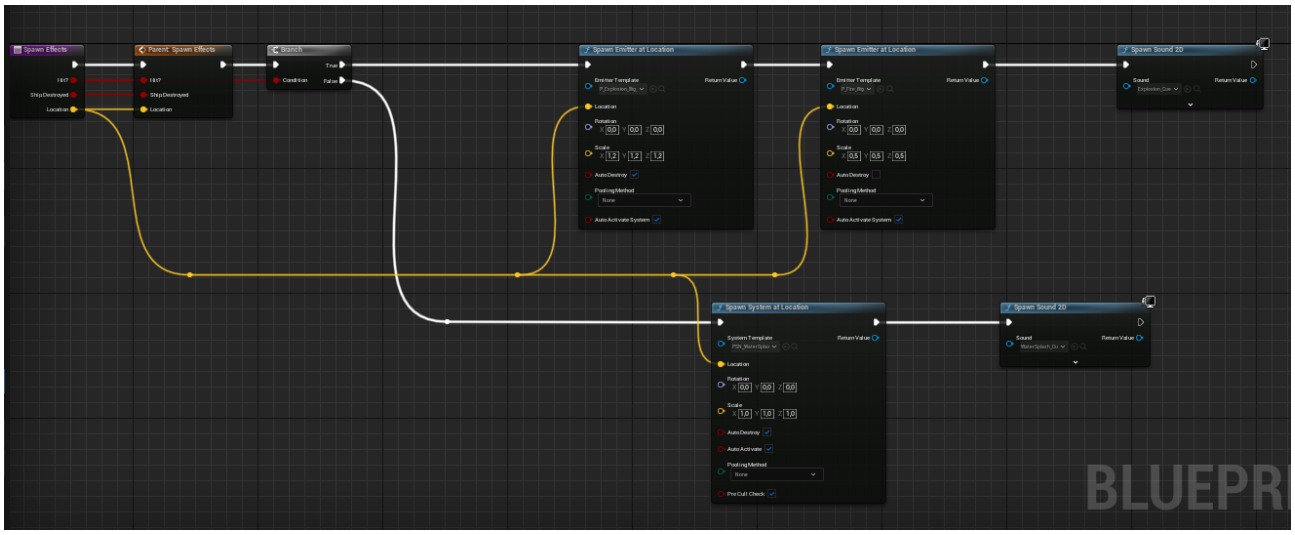

Рисунок 3.7.6 Функція " Spawn Effects" класу "BP_MainField".

8. *BP_EnemyField*. Клас наслідується від класу "BP_ParentField". На старті виконання запускає функцію батьківського класа (Рисунок 3.8.1).

   Функції:

   - *"FieldSetCurrentPos"* Перевизначає функцію батьківсього класу. Розміщує приціл на полі (Рисунок 3.8.2-3.8.3).
   - *"FieldAction"* Перевизначає функцію батьківсього класу. Здійснює постріл (Рисунок 3.8.4).
   - *"Spawn Effects"* Перевизначає функцію батьківсього класу. Створює ефекти (Рисунок 3.8.5).

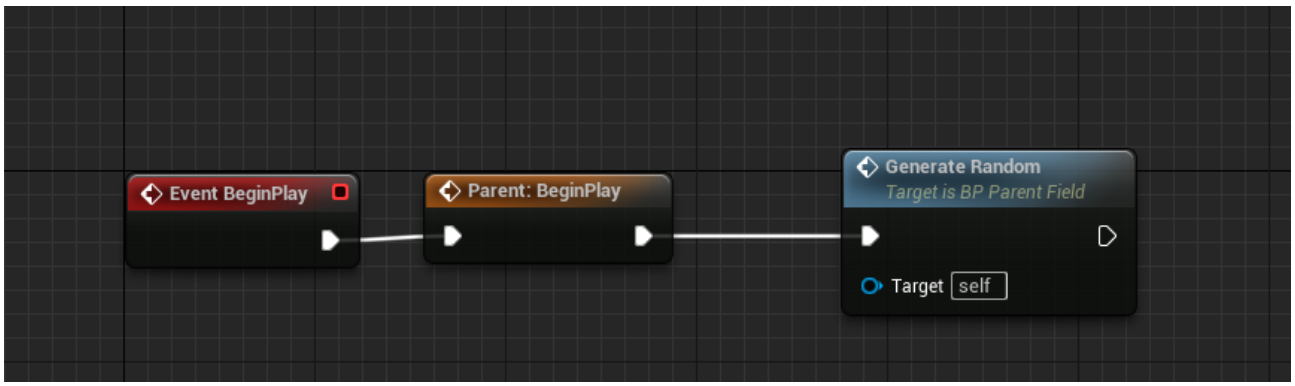

Рисунок 3.8.1 Клас "BP_EnemyField".

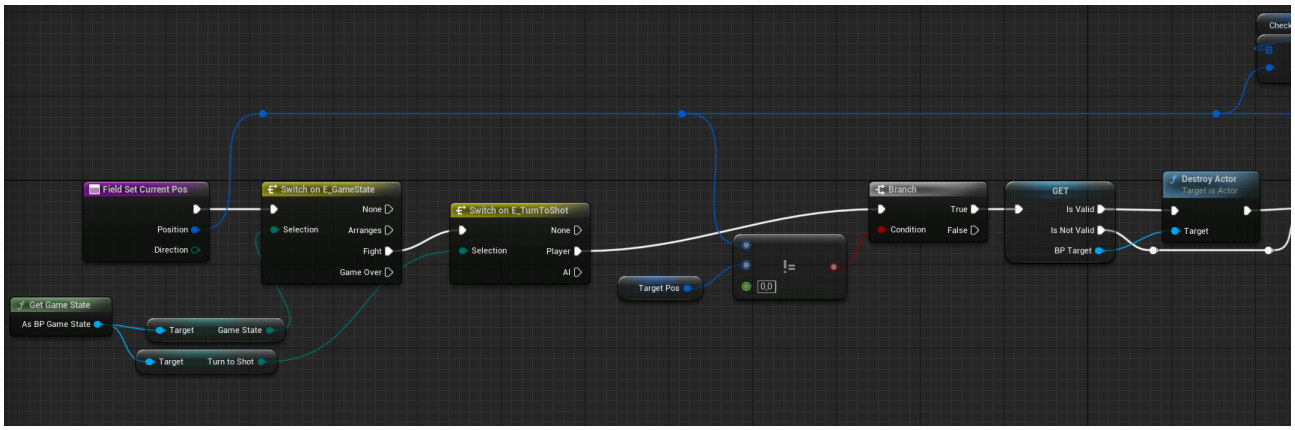

Рисунок 3.8.2 Функція "FieldSetCurrentPos" класу "BP_EnemyField".

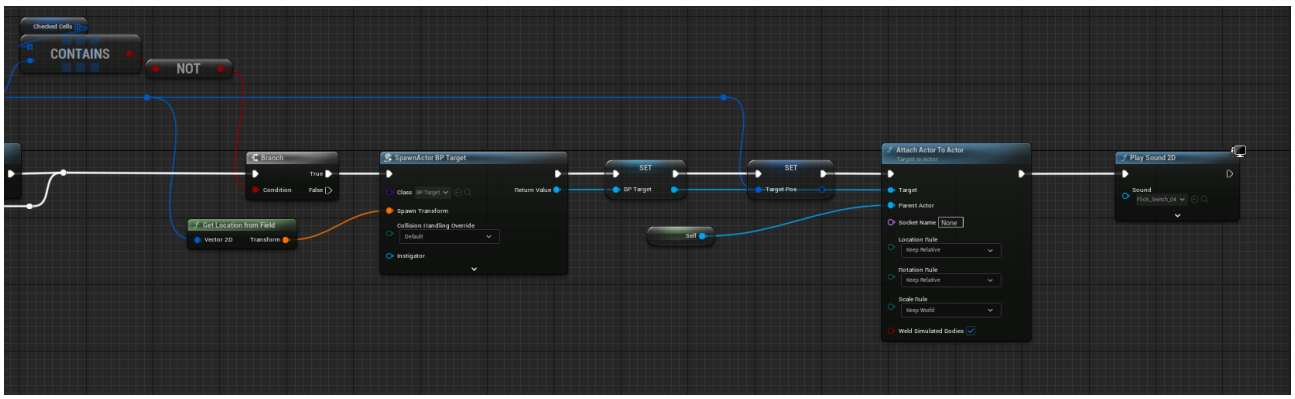

Рисунок 3.8.3 Функція "FieldSetCurrentPos" класу "BP_EnemyField".

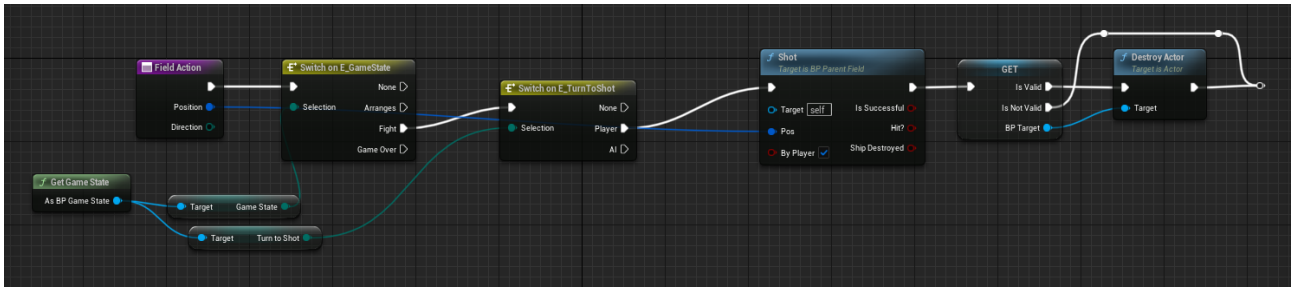

Рисунок 3.8.4 Функція "FieldAction" класу "BP_EnemyField".

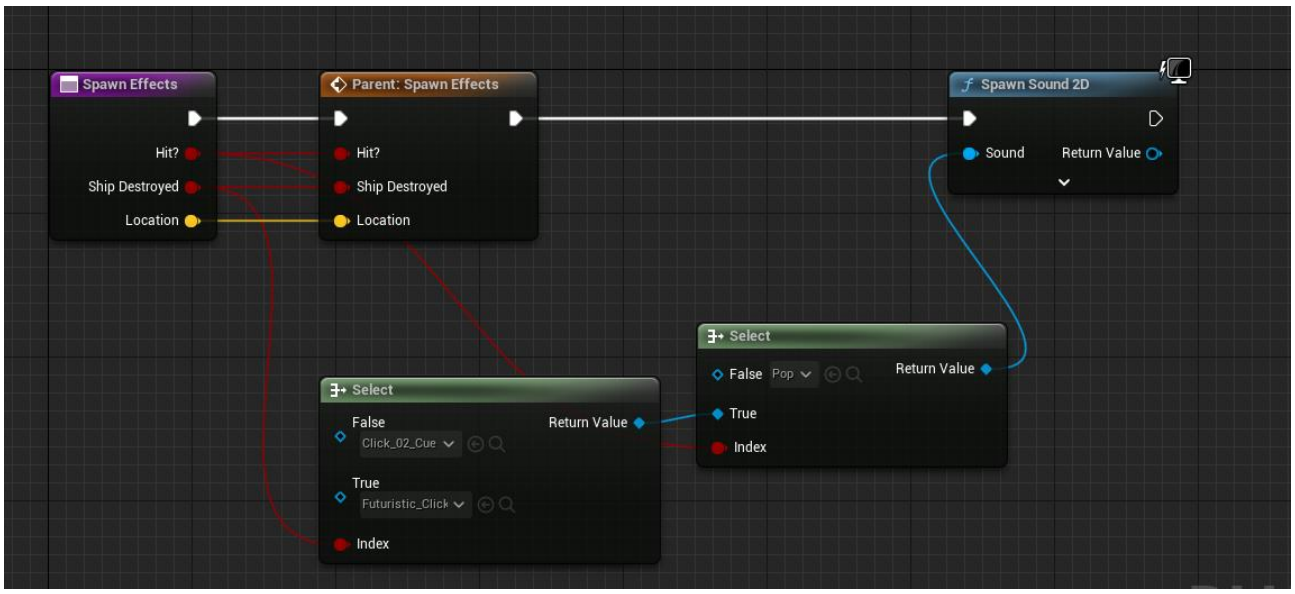

Рисунок 3.8.5 Функція "Spawn Effects" класу "BP_EnemyField".

9. *AC_ShotAI.* Допоміжний компонент в якому прописана логіка штучного інтелекту:

Функції:

GetNextCellToFire. Функція вибирає наступну клітинку.

AI Fire. Функція перевіряє вибрану клітинки і здійснює постріл.

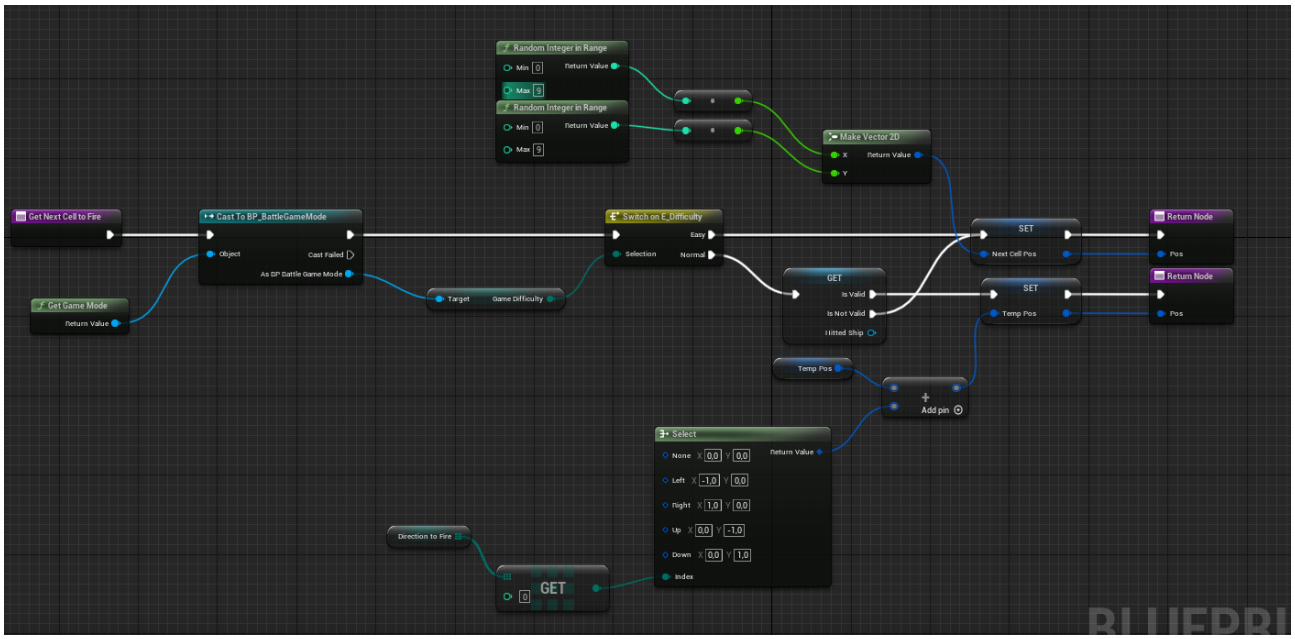

Рисунок 3.9.1 Функція "GetNextCellToFire" класу "AC_ShotAI".

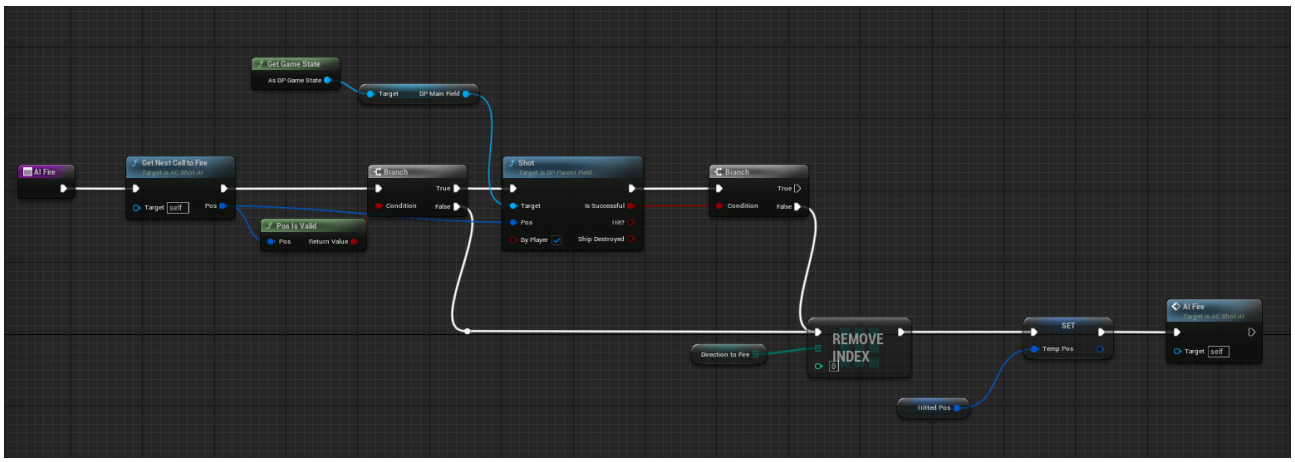

Рисунок 3.9.1 Функція "AI Fire" класу "AC_ShotAI".

10. *BP_Clipboard.* Клас "планшета" (Рисунок 3.10.1). Клас містить віджет "WBP_Clipboard", поле супротивника і фон з текстурою води [17]. Має 2 функції, які оновлюють кількість кораблів для відображення і готовність вогню.

Рисунок 3.10.1 Клас "BP_Clipboard".

11. *WBP_Clipboard.* Клас віджету. Показує кількість кораблів і готовність вогню (Рисунок 3.11.1).

    Функції:

    UpdateShips. Оновлює кількість кораблів та створює об'єкти класу "WBP_ShipCounter" (Рисунок 3.11.2).

    "Update Turn To Shot" оновлює готовність вогню (Рисунок 3.11.3).

Рисунок 3.11.1 Клас "WBP_Clipboard".

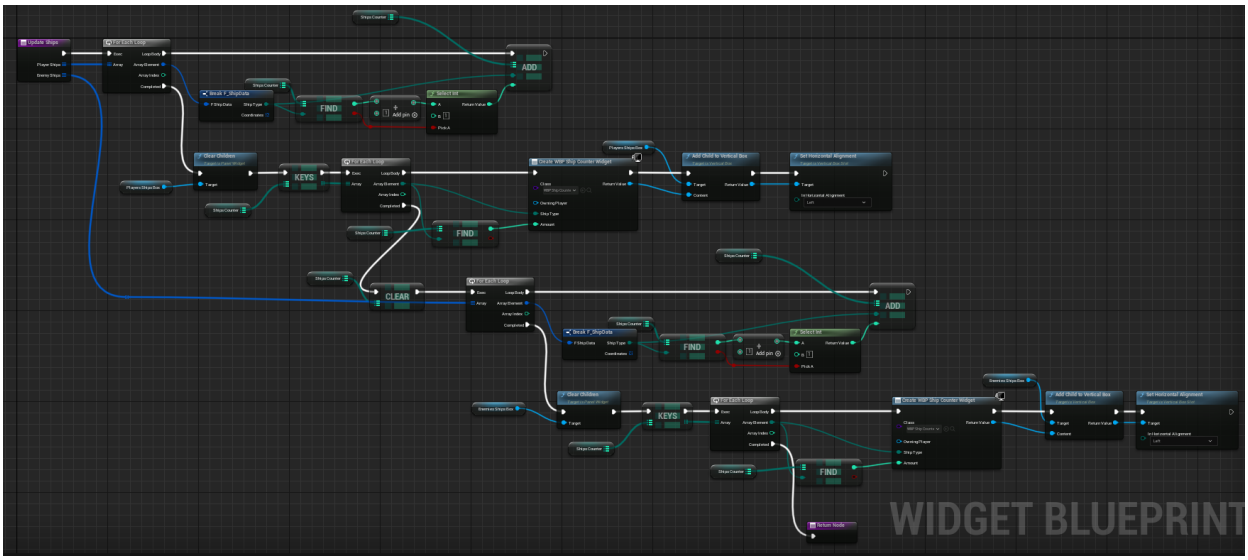

Рисунок 3.11.2 Функція "Update Ships" класу "WBP_Clipboard".

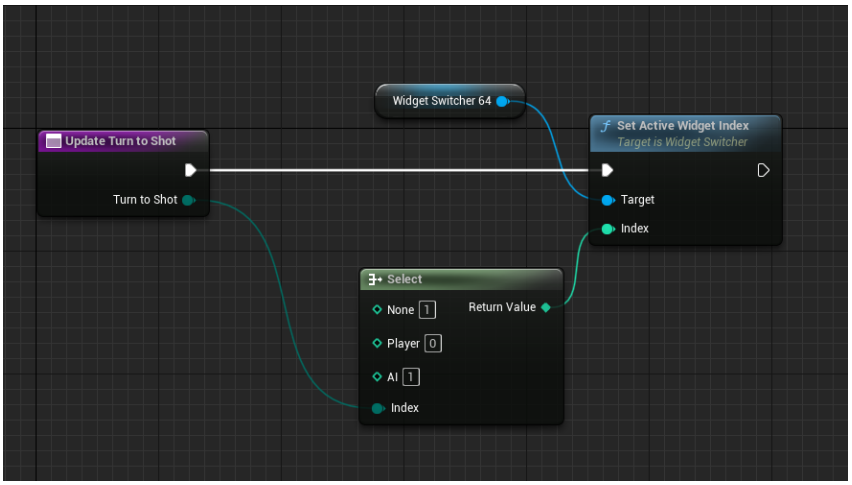

Рисунок 3.11.3 Функція “ Update Turn to Shot” класу “WBP_Clipboard”.

12. *WBP_ShipCounter*. Клас для одного рядка кількості кораблів в “WBP_Clipboard”. В залежності від параметрів, які надаються при створені об’єкту, показує тип корабля і кількість (Рисунок 3.12.1-3.12.2).

Рисунок 3.12.1 Клас “WBP_ShipCounter”.

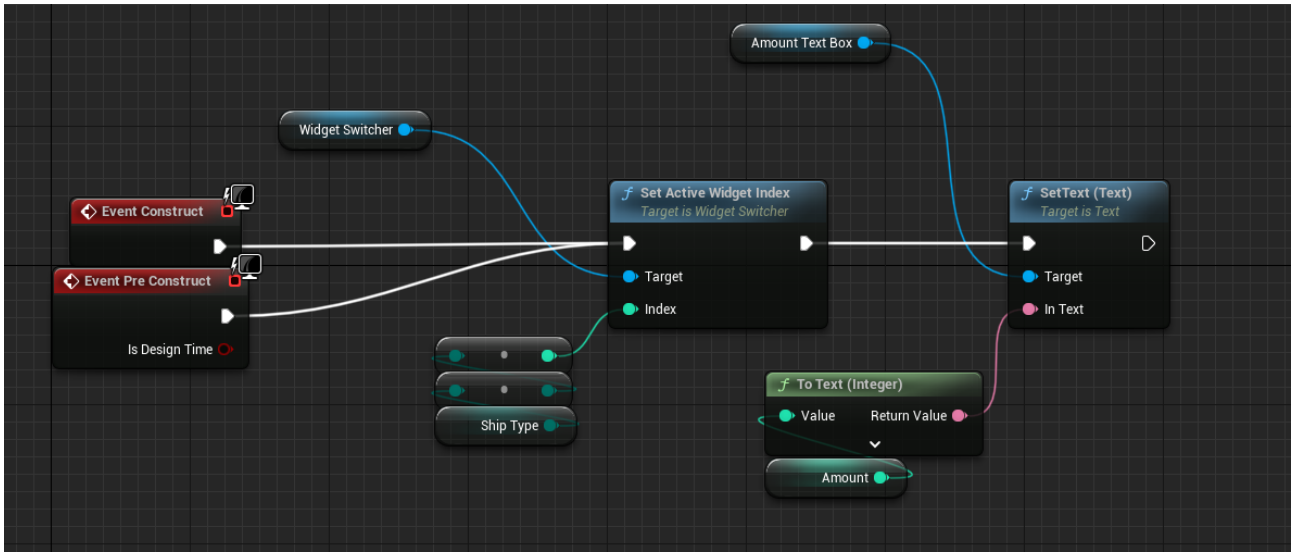

Рисунок 3.12.2 Клас “WBP_ShipCounter”.

13. *WBP_MainMenu.* Клас головного меню. Має два рівні (Рисунок 3.13.1 і 3.13.2) на першому вибір між початком гри і виходом, на другому — вибір складності. При натисканні на кнопку “Start game” меню переходить на другий рівень. При натисканні на “Exit” гра закривається. При натисканні на кнопки складності відкривається новий рівень, де і буде основний геймплей рівні (Рисунок 3.13.3).

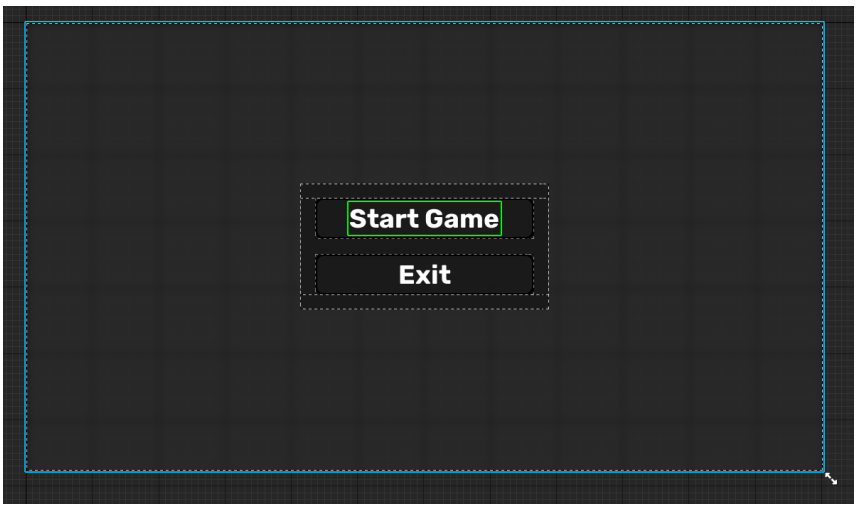

Рисунок 3.13.1 Клас "WBP_MainMenu".

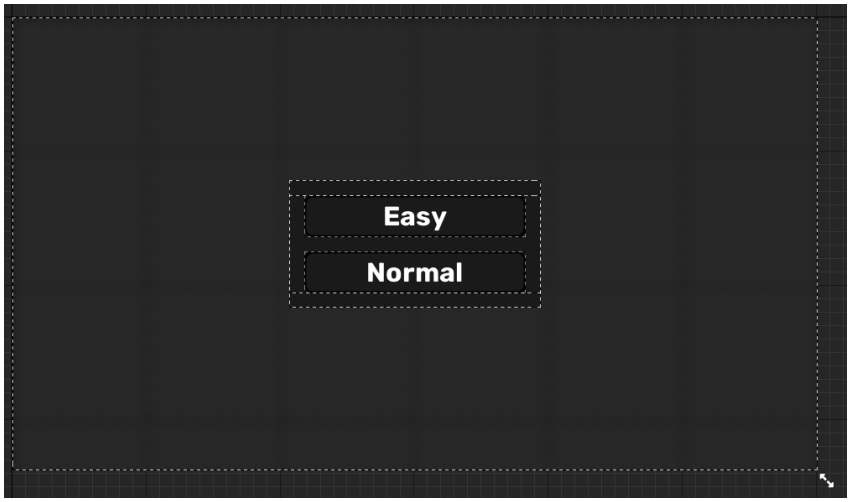

Рисунок 3.13.2 Клас "WBP_MainMenu".

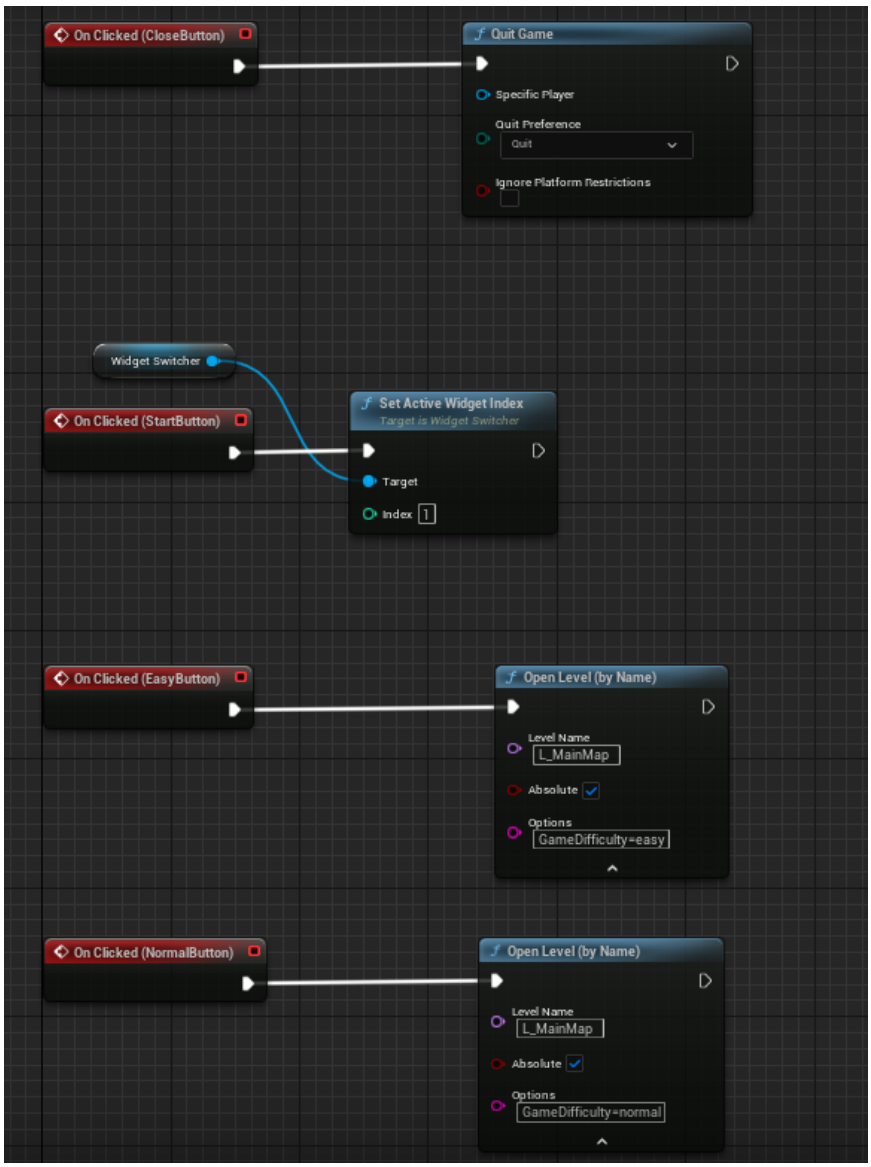

Рисунок 3.13.3 Клас "WBP_MainMenu".

В грі існує два рівні: "L_Menu" (Рисунок 3.14.1) і "L_MainMap" (Рисунок 3.14.3). Перший відповідає за головне меню, другий — за основний геймплей.

На рівні "L_Menu" знаходиться інструмент "Кран" з камерою. Він повільно крутиться навколо своєї осі, роблячи меню динамічним. Також є поле "BP_MainField" і вода [17]. Після відкриття рівня на полі генеруються кораблі і встановлюється режим керування для UI. (Рисунок 3.14.2). Також створюється два таймера.

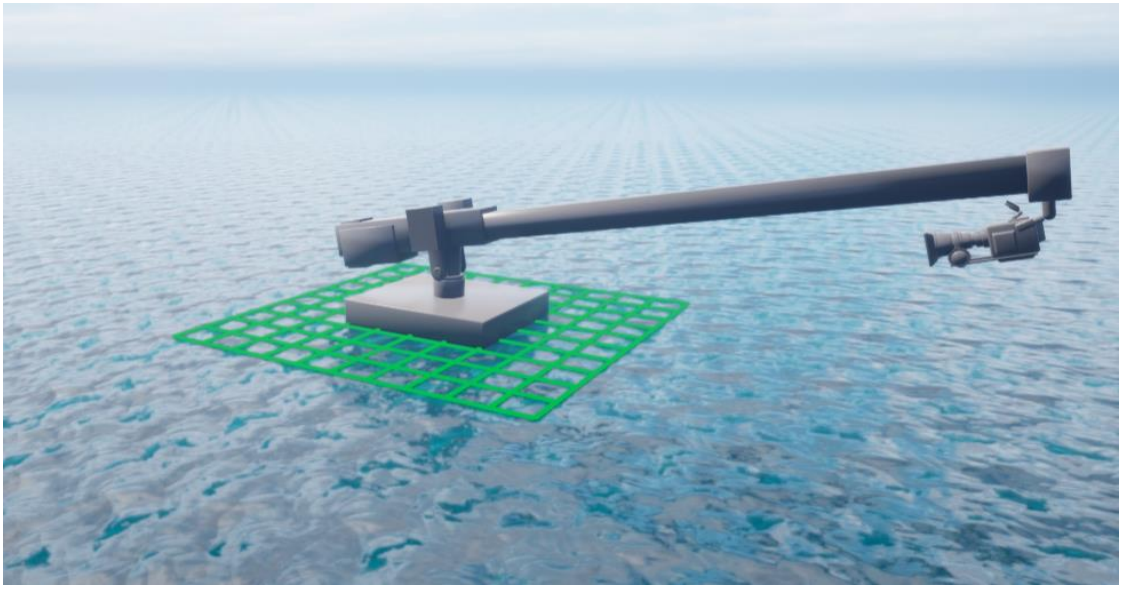

Рисунок 3.14.1 Рівень “ L_Menu ”.

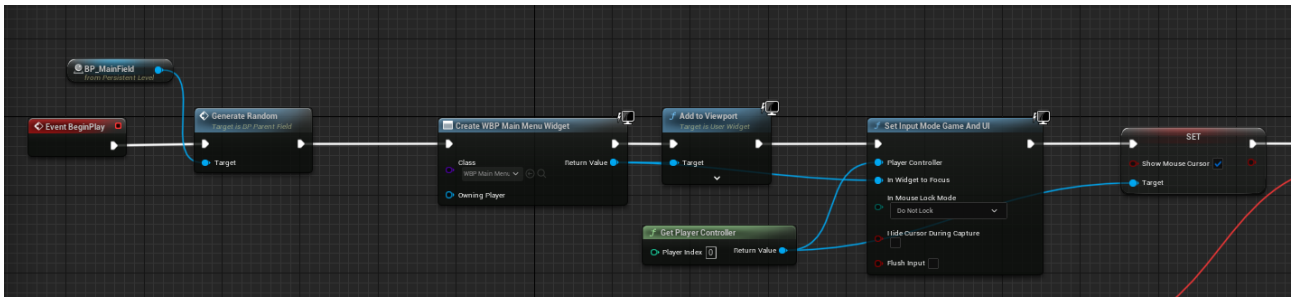

Рисунок 3.14.2 Рівень “ L_Menu ”.

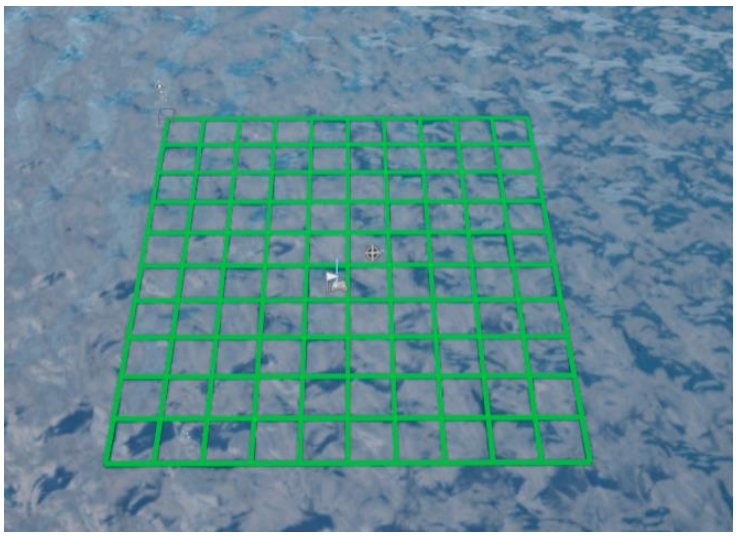

Рисунок 3.14.3 Рівень “ L_MainMap”.

3.3 Реалізація. Огляд гри.

Після запуску гри, користувач бачить меню з динамічним фоном (Рисунок 3.15.1). Після натискання на кнопку “Start Game” відкривається меню вибору

складності (Рисунок 3.15.2). Після вибору складності користувач має можливість розставити всі кораблі на сітці. Корабель показується зеленим, якщо його можна розмістити на вибраній позиції (Рисунок 3.15.3), якщо не можна — червоним (Рисунок 3.15.4). Коли користувач вибрав розташування для всіх кораблів, починається наступний етап битви (Рисунок 3.15.5). В цьому етапі користувач і ШІ по черзі намагаються влучити у кораблі супротивника. Влучені гравцем кораблі позначаються хрестиками на полі ворога, промах позначається точкою. Після знищення гравцем корабля, він замінюється моделькою знищеного корабля. При влучені в корабель гравця, клітинка позначається хрестиком та ефектом вибуху і вогню [18]. Промах — точкою. (Рисунок 3.15.6). Після втрати одною з сторін всіх кораблів, гра закінчується: зникає меню бою та відкривається меню, яке пропонує почати нову гру, або закрити додаток (Рисунок 3.15.7).

Рисунок 3.15.1 Меню з динамічним фоном

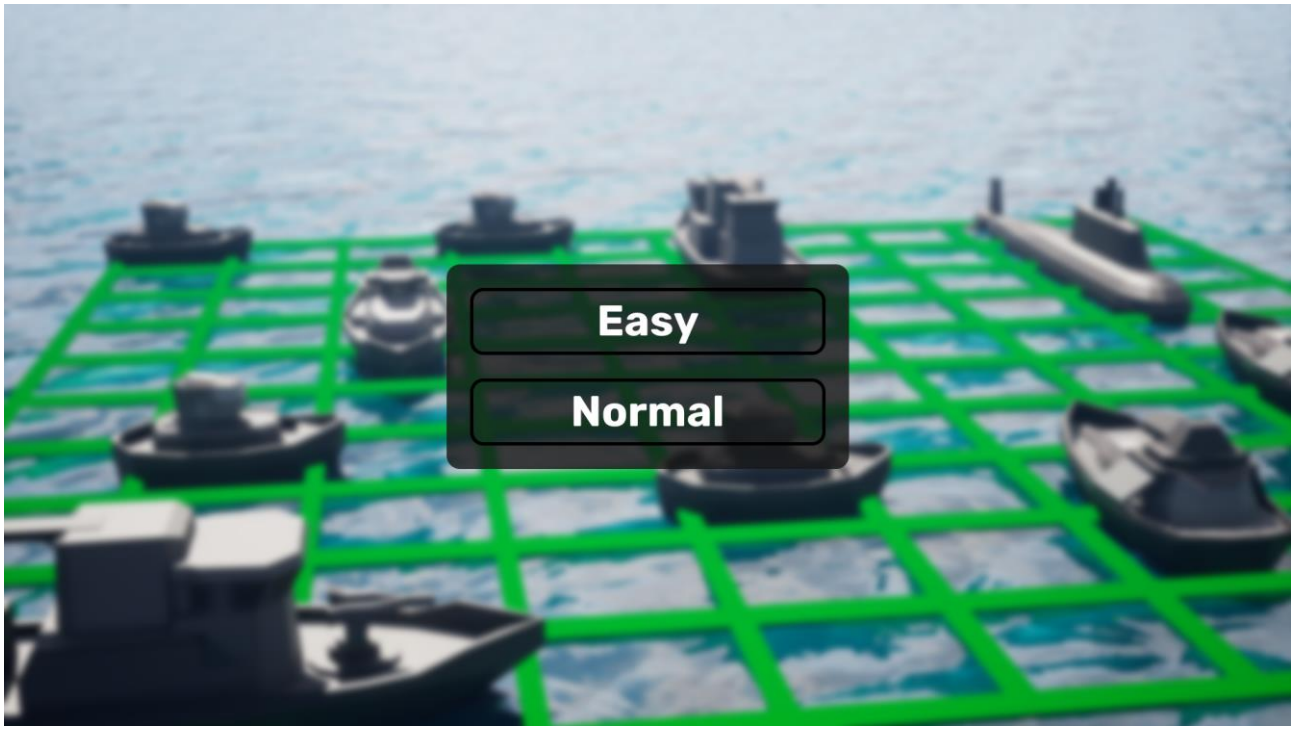

Рисунок 3.15.2 Меню з динамічним фоном

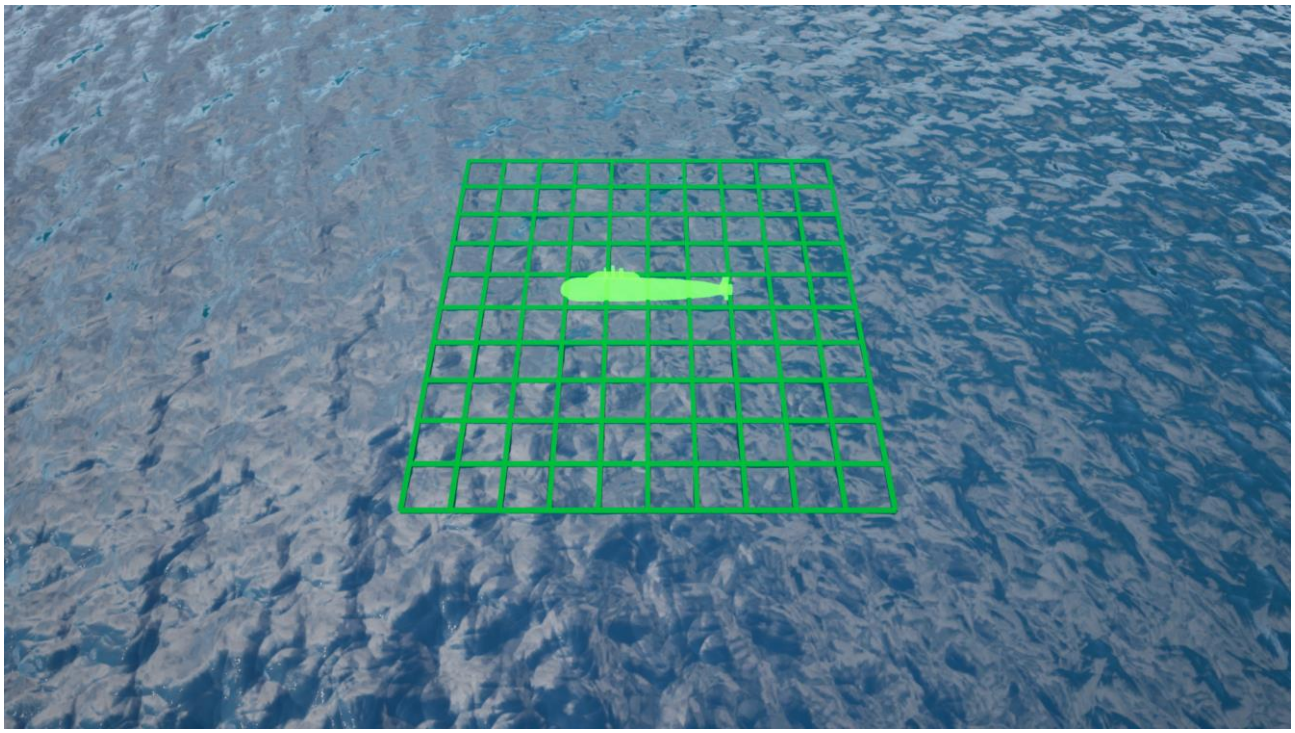

Рисунок 3.15.3 Корабель можна розмістити

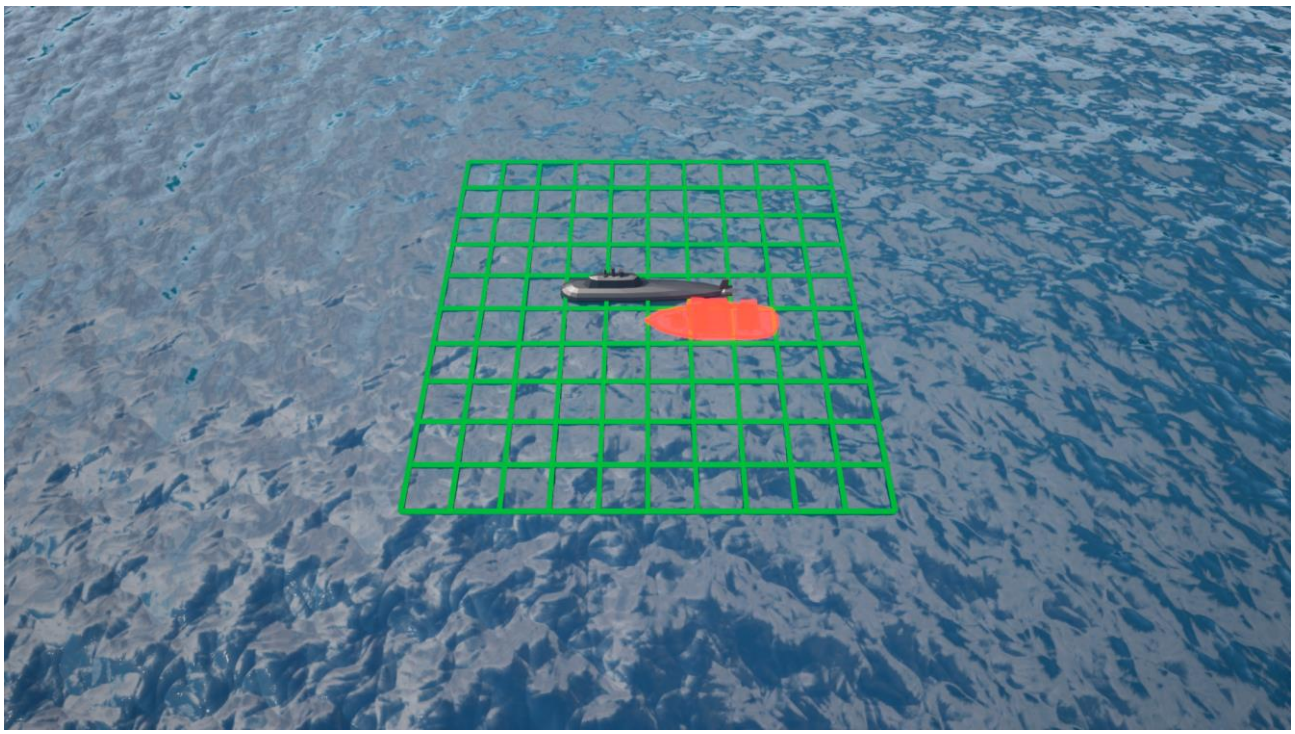

Рисунок 3.15.4 Корабель не можна розмістити

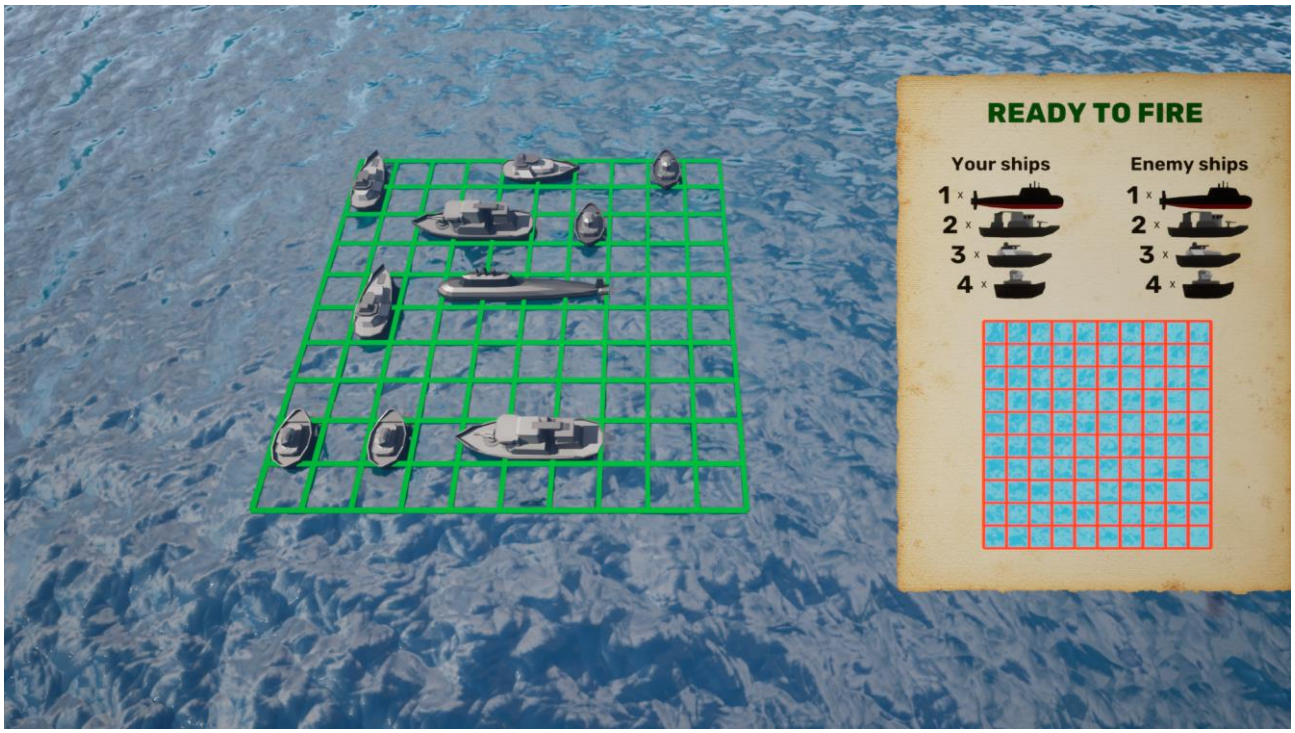

Рисунок 3.15.5 Етап битви.

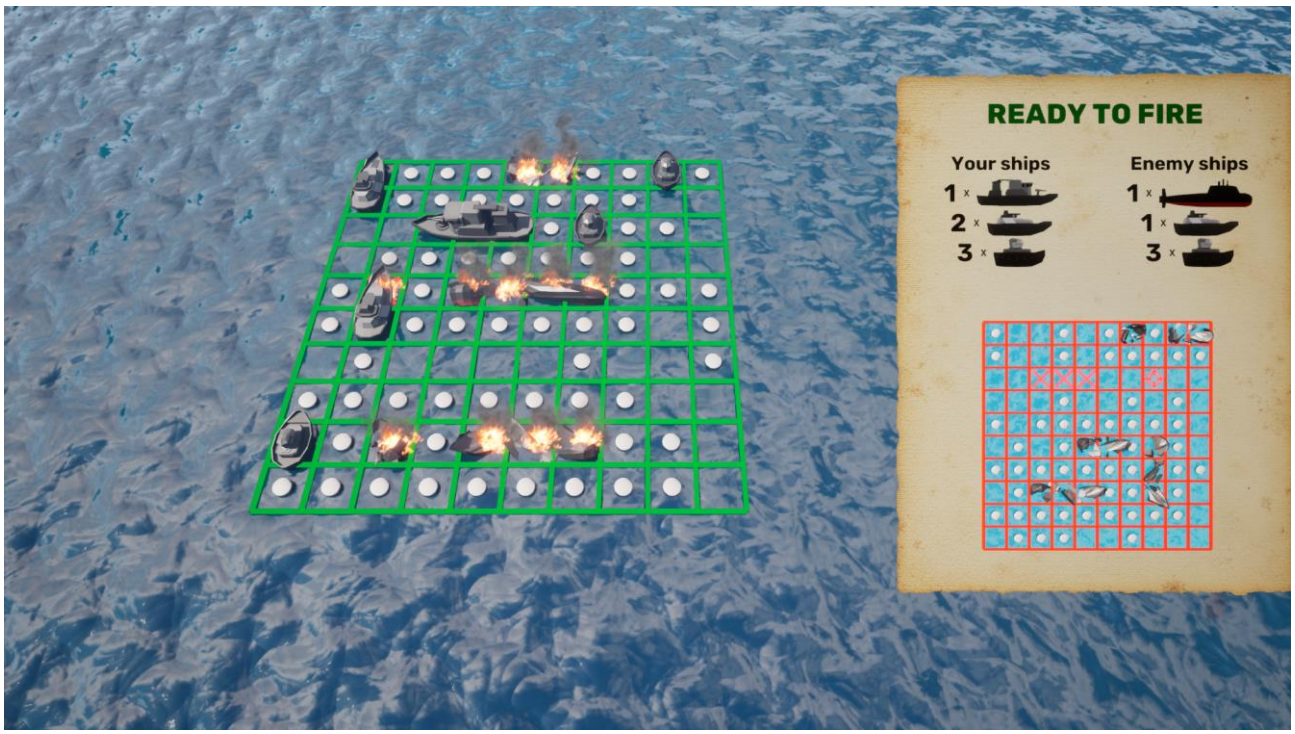

Рисунок 3.15.6 Етап битви.

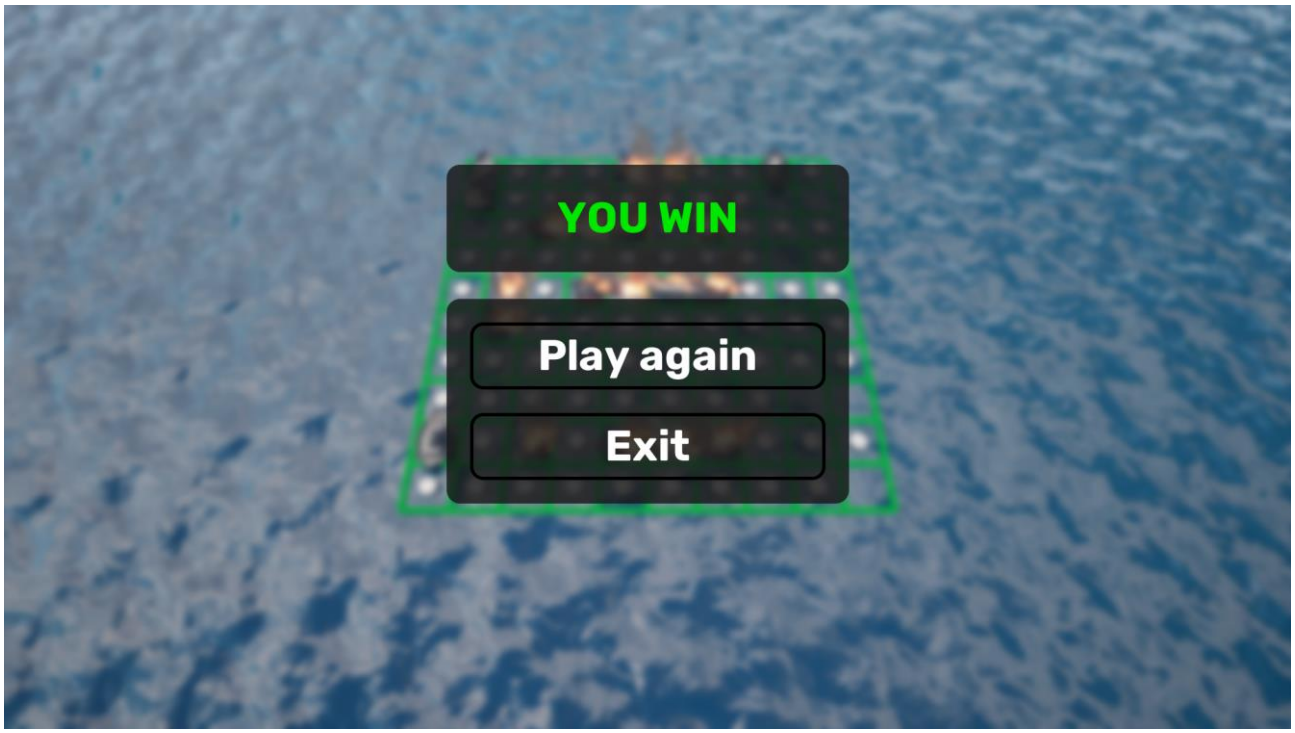

Рисунок 3.15.7 Кінець гри.

Висновок

В цій роботі було розглянуто загальну інформацію про Unreal Engine 5: історію, деякі інструменти і можливості. Наведено багато прикладів галузей, де Unreal використовується. Також розглянуті доступні платформи і ресурси для вивчення рушія. Було проведено порівняння з конкурентами ринку.

Аналіз ринку показав, що ігрова індустрія дуже популярна, а розробка ігор дуже перспектива галузь. Порівняння з іншими рушіями показало. що Unreal Engine 5 має одні з найкращих інструментів серед конкурентів. Багато як великих, так і маленьких ігор створюються на ньому кожного дня. Було розтягнуто приклад реалізації гри за допомогою системи нодів “Blueprint”.

Результатом курсової роботи стала гра “Морський бій”. Наведено детальний опис всіх класів та методів, які були використані при розробці гри, а також описано логіку гри та геймплей. Гра повністю готова і може бути використана як готовий продукт. Розробка гри дала можливість поглибити знання у галузі розробки ігор, ознайомитись з різними інструментами та можливостями Unreal Engine 5 та відчути процес розробки від початку до кінця.

Список використаної літератури

1. [Електронний ресурс] Newzoo Global Games Market Report 2022 | Free Version https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2022-free-version
2. [Електронний ресурс] Awesome Unreal Engine-powered games of 2022 https://www.unrealengine.com/en-US/blog/awesome-unreal-engine-powered-games-of-2022
3. [Електронний ресурс] History of the Unreal Engine https://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine
4. [Електронний ресурс] Film & Television | Unreal Engine https://www.unrealengine.com/en-US/solutions/film-television
5. [Електронний ресурс] Produce Broadcast & Live Events Mixed Reality Experiences - Unreal Engine https://www.unrealengine.com/en-US/solutions/broadcast-live-events
6. [Електронний ресурс] Advanced Automotive and Car Design Software and Visualization - Unreal Engine https://www.unrealengine.com/en-US/solutions/automotive-transportation
7. [Електронний ресурс] Cutting Edge Training & Simulation Software Platform - Unreal Engine https://www.unrealengine.com/en-US/solutions/simulation
8. [Електронний ресурс] Unreal Engine 5.1 Documentation https://docs.unrealengine.com/
9. [Електронний ресурс] Unreal Engine Forum https://forums.unrealengine.com/
10. [Електронний ресурс] Reddit Unreal Engine https://www.reddit.com/r/unrealengine/
11. [Електронний ресурс] UE Online Learning https://www.unrealengine.com/marketplace/en-US/content-cat/assets/onlinelearning
12. [Електронний ресурс] Awesome Unreal Engine-powered games of 2022 https://www.unrealengine.com/en-US/blog/awesome-unreal-engine-powered-games-of-2022

13. [Електронний ресурс] Game engines on Steam: The definitive breakdown https://www.gamedeveloper.com/business/game-engines-on-steam-the-definitive-breakdown
14. [Електронний ресурс] Behavior Tree https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/ArtificialIntelligence/BehaviorTrees/BehaviorTreesOverview/
15. [Електронний ресурс] Unreal Editor for Fortnite https://www.unrealengine.com/en-US/blog/unreal-editor-for-fortnite-is-now-available-in-beta
16. [Електронний ресурс] Unity vs Unreal, What Kind of Game Dev Are You? https://www.incredibuild.com/blog/unity-vs-unreal-what-kind-of-game-dev-are-you
17. [Електронний ресурс] Water Planes https://www.unrealengine.com/marketplace/en-US/product/water-planes
18. [Електронний ресурс] Realistic Starter VFX Pack Vol 2 https://www.unrealengine.com/marketplace/en-US/product/realistic-starter-vfx-pack-vol