МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра Комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
комп'ютерних систем та мереж

_____ Жуков І.А.

« ____ » _____ 2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
«БАКАЛАВР»

Тема: Система ігрового штучного інтелекту на базі *Unity*

Виконавець: Баборига Роман Сергійович

Керівник: Гузій Микола Миколайович

Нормоконтролер: Журавель Сергій Володимирович

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж
Напрям (спеціальність) 123 "Комп'ютерна інженерія"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем та мереж

_____ Жуков І.А.

« ____ » _____ 2021 р.

З А В Д А Н Н Я
на виконання дипломного проєкту

Бабориги Романа Сергійовича
(прізвище, м'я, по батькові випускника в родовому відмінку)

1. Тема проєкту: «Система ігрового штучного інтелекту на базі *Unity*» затверджена наказом ректора від «26» квітня 2021 року № 648/ст.

2. Термін виконання проєкту: з 24.05.2021 р. до 20.06.2021 р.

3. Вихідні данні до проєкту: розробка і створення системи ігрового штучного інтелекту з використанням технологій *Unity*.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): аналітичний огляд, дослідження ігрового штучного інтелекту, проєктування системи ігрового штучного інтелекту, створення системи ігрового штучного інтелекту в середовищі *Unity*, висновки.

5. Перелік обов'язкового графічного матеріалу: презентація *Power Point*.

6. Календарний план

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Проаналізувати літературу та джерела відповідно до теми проєкту	24.05.2021 26.05.2021	
2.	Розробити та затвердити план проєкту	27.05.2021 28.05.2021	
3.	Провести консультації з науковим керівником щодо створення першого розділу.	28.05.2021 29.05.2021	
4.	Ознайомитись із загальними положеннями ігрового штучного інтелекту та його особливостями	29.05.2021 30.05.2021	
5.	Виконати проєктування та визначення ключових компонентів розроблюваної системи	30.05.2021 01.06.2021	
6.	Виконати практичну реалізацію дипломного проєкту	02.06.2021 05.06.2021	
7.	Сформувати висновки та оформити пояснюальну записку	05.06.2021 07.06.2021	
8.	Підписати необхідні документи у встановленому порядку	07.06.2021 11.06.2021	
9.	Підготуватись до захисту та попереднього захисту дипломного проєкту на випусковій кафедрі дипломного проєкту	11.06.2021 12.06.2021	

7. Дата отримання завдання: «24» травня 2021р.

Керівник дипломного проєкту _____ Гузій М.М.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Баборига Р.С.
(підпис студенту) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Система ігрового штучного інтелекту на базі *Unity*»: 63 с., 41 рис., 22 літературних джерела, 2 додатки.

Мета дипломного проєкту – проєктування та розробка ігрового штучного інтелекту.

Об'єкт проєктування – ігровий штучний інтелект.

Предмет проєктування – архітектура та програмна реалізація системи ігрового штучного інтелекту.

Практична значимість – розроблена система ігрового штучного інтелекту може використовуватись в ігрових додатках в середовищі *Unity*.

Технічні та програмні засоби задіяні в проєкті – в даному проєкті використовувався ігровий рушій *Unity* - основне середовище, *Visual studio code* – текстовий редактор для написання коду, A* *Pathfinding Project* - бібліотека компонентів для організації роботи алгоритму A*.

Прогнози, припущення щодо розвитку об'єкта дослідження – отриману систему можна вдосконалювати шляхом оптимізації коду, котрий використовується для організації роботи системи, можлива імплементація нових систем сприйняття середовища та покращення візуальної складові.

ЗМІСТ

ВСТУП

На сьогоднішній день ми розуміємо, що живемо у матеріальному світі, основу якого складає глобальна економіка, а спеціалісти в галузі інформаційних технологій є ключовими елементами цієї складної системи. Їх ідеї та практична реалізація розумових задумів, спрямованих на задоволення нагальних потреб суспільства, визначають подальший тренд в усіх без винятку сферах.

Ігрова індустрія є великою частиною світової економіки. А вона, в свою чергу, поєднує в собі дві монументальні складові – це індустрія розваг та індустрія інформаційних технологій. Дана галузь економіки є однією з найдинамічніших на сьогоднішній день і цей факт є ключовим у визначенні ігрової індустрії як актуальної та перспективної галузі економіки в сучасних реаліях.

Продукти ігрової індустрії є досить складними, оскільки не складаються з матераільних об’єктів, а є винятково цифровими. Усе, що можна назвати відеогрою, створюється спеціалістами з різноманітних сфер. Велику частину роботи виконують саме спеціалісти інформаційних технологій, так як сучасна відеогра - це складна програмна система із обмеженим набором об’єктів та варіативним складом можливих їх станів. Невід’ємною частиною кожного продукту ігрової індустрії є ігровий штучний інтелект, який і забезпечує оригінальність розробок та урізномантнює варіанти прийняття можливих рішень . Він являє собою програмну складову продукту та призначений для виконання різноманітних задач в залежності від потреби та уподобань користувача. Прикладів застосування ігрового штучного інтелекту є велика кількість. Найчастіше дані системи використовують для заміни реальної людини в масштабах гри, тобто представлення комп’ютера як супротивника для гравця, або для виконання певних задач, необхідних для гравця в масштабах ігрового додатку та імітації поведінки реальної людини, істоти тощо. Загалом важко уявити сучасні відеоігри без використання даного потужного інструменту.

З розвитком інформаційних технологій невпинно розвивається й ігрова індустрія. З розвитком ігрової індустрії в цілому збільшується загальна складність

самих ігрових програм. Оскільки комп’ютери стають потужнішими, а це певним чином продиктоване потребами суспільства у всіх сферах життєдіяльності людини, котрі зав’язані на час реалізації ідей та обсяг потрібних для цього інформаційних ресурсів. Це відкриває можливість для розробників ігор реалізовувати набагато складніші проєкти з прицілом на урізноманітнення існуючих та генерування інноваційних сюжетів. Паритетно зростає і складність проєктування та реалізації ігрового ШІ, котрий в свою чергу є потужним інструментом з великою кількістю можливих моделей його практичної реалізації. Наприклад, в сучасних іграх для імітації поведінки живої людини комп’ютер повинен обробляти велику кількість інформації, яка доступна в ігровому середовищі; а зі збільшенням реалістичності продуктів ігрової індустрії кількість чинників, які безпосередньо впливають на систему, яка керована комп’ютером, та даних для обробки в цілому, постійно збільшується. Тому потреба в створенні систем ігрового штучного інтелекту є актуальною на даний час.

Оскільки тенденції на розвиток достатньо амбітні, і індустрія буде лише розвиватись з кожним роком, то і попит на створення програмних систем, які є суміжними з ігровою індустрією, буде лише збільшуватись. Потреба у створенні систем ігрового штучного інтелекту є потужним рушієм їх удосконалення як складних програмних систем.

Ці фактографічні дані і визначили мету даного дипломного проєкту - створення системи ігрового штучного інтелекту.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Ігрова індустрія

Ігрова індустрія є однією із найбільш технологічних та динамічних відгалужень світової економіки. Вона є досить великим та вагомим сегментом, адже знаходиться на перетині чисельних сфер: програмування, маркетингу, дизайну тощо. Особливо сьогодні ігрова індустрія набуває стрімкого розвитку за рахунок постійного притоку нових споживачів, що збільшує рівень зацікавленості, і це в свою чергу породжує ще більший попит.

На тлі глобальної економіки ігрова індустрія є активним та прибутковим сектором з тенденціями на постійний подальший розвиток, а самі ігри позиціонуються і сприймаються споживачами як високоякісні програмні продукти. Глобальний економічний обіг ігрової індустрії за останніх 5 років випередив музичну індустрію та кіноіндустрію, які до цього вважались основними та найбільш прибутковими (рис 1.1) [8].

Рис. 1.1. Динаміка світового ринку ігрової індустрії, млрд дол.

Згідно з доповіддю старшого аналітика ринку *Newzoo* Тома Віймана, світовий ринок ігор приніс в 2020 році 159,3 мільярда доларів прибутку. Це понад 9,3 відсотка зростання в порівнянні з минулим роком. Більше того, *Newzoo* запевняє, що галузь перевищить 200 мільярдів доларів у 2023 році [13].

На сьогоднійшній день продукти ігрової індустрії є невід'ємною складовою нашого життя, оскільки вони глибоко проникли у всі сфери людської діяльності. Сама ігрова індустрія стає глобальним провідником ідей сучасної творчості та новаторства. Також саме розвиток ігрового сегменту був одним із рушіїв для розвитку та модернізації апаратного забезпечення персональних комп'ютерів.

Ігрова індустрія є багаторівневою та складною структурою зі своїми особливостями. Кожен рівень є самостійною складною складовою індустрії. В структурі сучасної ігрової індустрії можна виділити декілька основних рівнів:

- Платформи
- Ігрові рушії
- Безпосередня розробка
- Видавництво та оперування
- Популяризація
- Споживання [15]

Кожен аспект індустрії включає в себе багато особливостей та потребує залучення великої кількості спеціалістів. Загалом у структурі ігрової індустрії працюють професіонали з різних сфер.

Платформа - це система, яка за своїх технічних характеристик дозволяє запустити інтерактивний ігровий додаток. Основні платформи:

- Персональні комп'ютери
- Ігрові консолі (спеціалізована ігрова платформа)
- Мобільні пристрої
- Універсальні *web* платформи
- Аркадні автомати
- Інноваційні платформи віртуальної реальності

Ігровий рушій – це базове програмне забезпечення, на основі якого будуються всі інші складові гри. Програмний код, який може використовуватися для створення варіацій гри, аддонів до неї або навіть зовсім нового ігрового світу. [22] Є проміжним елементом між платформою та безпосередньо кодом ігрового продукту.

Розробка ігрових програм - це досить складний та багаторівней процес, що потребує залучення спеціалістів різних аспектів ігрової індустрії в цілому. У процес розробки залучені: геймдизайнери, художники, композитори, аніматори, програмісти та багато інших спеціалістів.

Видавництвом та розповсюдження займаються як правило не самі розробники, а видавці. Вони також виконують важливу роль в оперуванні ігровими додатками. Видавці займаються локалізацією, взаємодіють з власниками платформ, забезпечують технічну підтримку продуктам ігрової індустрії.

Популяризація – важливий аспект в життєвому циклі будь-якого ігрового додатку. Над даним механізмом працює велика кількість спеціалістів з області маркетингу, адже він є ключовим аспектом у фінансовій сфері ігрового додатку. Також популяризацією займаються різноманітні ЗМІ, сайти, присвячені ігровій тематиці, інтернет-ресурси. Окремим рушієм у популяризації ігрових додатків є виставки та конференції, які відіграють важливу роль не тільки у фінансовому аспекті, але і в суспільному, оскільки зміцнюють стосунки між споживачем та розробником.

Споживання - це кінцевий пункт у структурі ігрової індустрії. Гравці - це основне джерело прибутку для ігрових продуктів. Саме гравці формують ігрові спільноти, які поглиблюють та урізноманітнюють культурну складову сучасного суспільства. А в сучасному світі найбільш активні гравці стали суттєвою рушійною силою в популяризації ігор та частково в розширенні контенту.

Ігрова індустрія є значною частиною культури сучасного суспільства. Продукти ігрової індустрії застосовуються не тільки для розваг. На ринку існує велика кількість додатків-симуляторів, створених для ознайомлювальних або для професійних цілей. Наприклад, сучасні армії використовують продукти ігрової

індустрії для практичної демонстрації основ тактики. Прикладом такої гри є військовий симулятор *ARMA. Microsoft Flight Simulator* – це ігровий симулятор польотів. Так в базі даних даного ігрового продукту містяться функціональні копії кабін справжніх літаків. Даний симулятор має безпосереднє практичне застосування в авіації, адже допомагає пілотам у процесі навчання без ризику для їхнього життя.

Психологами доведено, що інформація краще засвоюється, якщо її подавати у ігровій формі. Так базову інформацію доносять дітям саме у форматі гри. Це стало однією із причин використання продуктів ігрової індустрії в процесі навчання. Також це підвищує рівень зацікавленості студентів та поглиблює процес навчання.

Однією з іноваційних та перспективних відгалужень ігрової індустрії є *AR/VR*. Розвиток даної технології відкриває широкі перспективи. Так відкривається можливість створювати симуляції для тренувань спеціалістів різноманітних сфер, використовуючи ігрові додатки. Спеціалісти зможуть проводити практичні заняття у симуляціях без небезпеки для людського життя. Це значно покращить якість вмінь та поглибить навички спеціалістів різноманітних сфер.

Майже кожен продукт ігрової індустрії використовує ігровий штучний інтелект (ШІ) в тому чи іншому представленні. Він виконує різноманітні завдання, визначені програмістом на етапі проєктування. Специфікація завдання для ігрового ШІ покриває широкий спектр функцій, що забезпечує його актуальність. З розвитком ігрової індустрії в цілому зростає і загальна складність процесу проєктування та створення ігрових продуктів, і звідси ускладнюється процес проєктування та створення систем ігрового ШІ. Оскільки ігрова індустрія знаходиться у постійному розвитку, то і питання проєктування та реалізації систем ігрового ШІ завжди будуть актуальними та затребуваними.

1.2. Визначення ігрового штучного інтелекту

Ігровий штучний інтелект (ШІ) – це набір програмних методик, які використовуються у іграх та ігрових системах для відтворення ілюзії інтелекту в

поведінці ігрових одиниць та інших ігрових аспектів, керованих комп'ютером (рис.1.2) [17].

Ігровий ШІ є досить складною сукупною системою різноманітних компонентів, спроєктованих та побудованих для виконання конкретних задач, яка в залежності від специфікації завдання конкретного ШІ може варіюватись та складатись з багатьох складних взаємопов'язаних компонентів, які знаходяиться у постійному контакті. Так цими компонентами можуть виступати окремі інструменти середовищ проєктування ігрових прорграм. Ігровий ШІ, крім методів традиційного ШІ, включає також алгоритми теорії керування, робототехніки, комп'ютерної графіки та інформатики в цілому. Складність реалізації системи ігрового ШІ залежить від особливостей проєкту та може відрізнятись для конкретного завдання.

Рис 1.2. Приклад ігрового ШІ

Ігровий ШІ специфікується для виконання конкретного завдання або дії, яку повинен виконувати об'єкт, виходячи з персональних умов, приватних даних, стану оточення, в яких знаходиться одиниця. Зазвичай це називають управлінням «інтелектуальними агентами», де агент є ігровою одиницею, а іноді і чимось більш абстрактним: цілою групою сутностей, цивілізацією тощо.

Система ігрового ШІ повинна взаємодіяти з об'єктами середовища, повинна обробляти дані, які отримує із фіктивних органів чуття, приймати рішення і діяти

відповідно до приватних умов та інструкцій. Це називається циклом *Sense / Think / Act* (Відчувати / Мислити / Діяти). Даний цикл є основним етапом життєдіяльності агента.

- *Sense*: агент знаходится або отримує спеціалізовану інформацію в своєму середовищі, яка може впливати на приватні дані та на поведінку ШІ в цілому.
- *Think*: агент обробляє отриману інформацію. Тут виконується основний етап обчислень вхідних даних.
- *Act*: агент виконує дії для реалізації попереднього рішення.
- Повтор циклу, адже в результаті обрахунків змінились дані середовища.

Слід відмітити певну різницю між справжнім ШІ та ігровим ШІ. Так ігровий ШІ виконує тільки обмежений ряд задач, адже не потребує реалізації деяких програмних можливостей наукового ШІ за відсутністю нагальної потреби. Ігровий штучний інтелект відрізняється від справжнього ШІ, так як ігровий проєктується та будується для виконання необхідних завдань, обмежених ігровим середовищем. Ігровий ШІ є лише відгалуженням наукового, оскільки не потребує реалізації певних функціональних особливостей справжнього ШІ (почуття та самосвідомість). Дані обчислення та функціональні можливості не є необхідними для реалізації ігрового ШІ. Суть створення ШІ – це всебічна допомога людині при виконанні конкретних завдань. Суть ігрового ШІ полягає у виконанні різноманітних задач у ігровому середовищі, які могли би виконуватися людиною.

Основною задачею ігрового ШІ є прийняття рішень. Даний механізм є приватним для кожного агента, тобто в кожного ігрового об’єкта, який підпорядковується ШІ, є набір своїх правил та інструкцій, на основі яких агент приймає рішення. Умови прийняття рішень певного ігрового ШІ в залежності від складності виконання завдань можуть формувати складну структуру. Для проєктування даних агентів використовують дві загальні методики: *машина станів, ієрархічні дерева вибору рішень*. Перший спосіб використовується для проєктування більш складних систем ігрового ШІ, а другий для більш простих. Хоча на практиці використовують їх симбіоз.

1.3. Види ігрових ШІ та їх використання

Ігровий ШІ є досить складною системою, проте складність реалізації залежить від конкретно визначених на етапі проєктування технічних завдань певного ігрового агента. Ігрові агенти не мають конкретного представлення. Залежно від технічного завдання реалізація ігрового ШІ може приймати як конкретну так і абстрактну форму. Прикладом абстрактного ігрового ШІ є агент, який може балансувати та перерозраховувати характеристики ігрових одиниць в реальному часі. Ігрові ШІ, які приймають матеріальну форму, загалом розділяють на три типи в залежності від складності їх реалізації та специфікації завдань агенту:

- *NPC* (англ. *Non-player character — NPC*)
- Боти (англ. *Bot*)
- Моби (англ. *Mob*)

Неігровий персонаж (англ. *NPC*) – ігрова одиниця, яка не може бути керована людиною. Найчастіше *NPC* керований програмою. Даний вид ігрового агента залежно від специфікації проєкту та ігрового середовища може бути реалізованим як складна система у випадках з високим технічним рівнем проєкту, проте може поєднувати відносно невелику кількість механік та бути відносно простою системою у проєктах казуального характеру. Найчастіше неігрові персонажі, керовані ігровим агентом, слугують для виконання конкретних завдань ігрового характеру, специфічних для даного проєкту. Але в сучасних іграх можливості розширилися, і тепер у деяких персонажів є великі програми з прийняттям власних рішень і різними реакціями на ті чи інші дії гравця. *NPC* діють в залежності від поведінки та ігрових механік, яким підпорядковується агент. Так даний вид ШІ може приймати самостійні рішення, які будуть впливати безпосередньо на середовище в залежності від приватних даних агента.

Боти – це програми-роботи, які керуються комп’ютером та імітують поведінку людей в ігровому середовищі. Програма-бот заснована на модулі штучного інтелекту, який адаптований до особливостей конкретного ігрового продукту. Загалом система ігрового агенту типу “Бот” технічно має найскладнішу реалізацію.

Так ігровий ШІ має велику кількість приватних даних та інструкцій, які агент має виконувати в необхідні проміжки часу та при настанні необхідних умов. В деяких проєктах боти можуть виконувати велику частину ігрового процесу. Боти зазвичай утворюють складну систему поєднання різноманітних компонентів середовища в якому він реалізується. У процесі прийняття рішень бот ґрунтується на власних характеристиках і на характеристиках доступного йому ігрового середовища. Деякі параметри змінюються під час ігрового процесу, що безпосередньо впливає на поведінку бота.

Моби - це найпростіший вид ігрових агентів. Система даного ігрового ШІ є відносно простою в реалізації в порівнянні з видами, наведеними вище. Моби не відіграють велику роль у глобальному масштабі ігрового проєкту, проте займають безпосередньо важливе місце. Найчастіше моби використовуються для реалізації механізму отримання певних ігрових одиниць (наприклад, валюта). Структура даного виду ігрових агентів може варіюватись від складності проєкту. Так ігрова одиниця може являтись мобом та слугувати безпосередньо для отримання внутрішньоігрових очок, проте мати складну будову та виконувати велику кількість різноманітних функцій.

1.4. Аналіз існуючих рішень системи ігрового ШІ

Half-Life — відеогра, розроблена *Valve Software* для ПК. Даний ігровий продукт став революційним у ігровій індустрії і зробив неоціненний вклад в розвиток ігор та ігрового ШІ безпосередньо. До *Half-Life* вороги у ігрових продуктах мали примітивну структуру, а даний ігровий продукт став новатором у сфері ігрового ШІ. Агенти вирізнялися своєю реалістичністю та різноманіттям.

Ігровий ШІ в *Half-Life* працює на основі кінцевого автомату. Система являє собою набір спроєктованих та сформованих станів. Стан описує конкретний шаблон поведінки ігрового агента або системи у визначений період часу. Зміна станів відбувається при виконанні конкретних умов, які задаються на етапі проєктування.

В даному ігровому продукті програмна складова системи ігрового ШІ використовує механізм наслідування. “*Monster*” – це головний клас, від якого наслідуються інші класи, які представляють ігровий ШІ (рис. 1.3).

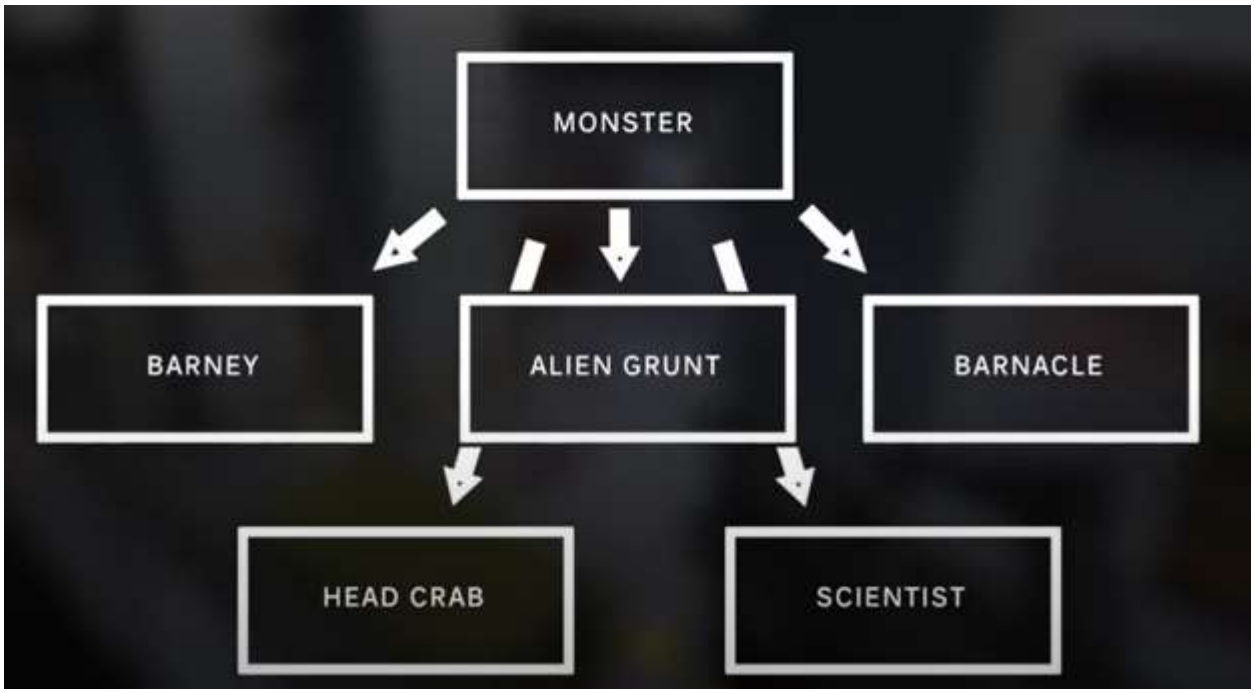

Рис. 1.3. Структура ігрових одиниць *Half-Life*

Класи, що наслідуються від основного, імплементували поведінку та дані базового класу та перевизначали їх у необхідних аспектах. Кожен клас, який реалізує певний вид ігрового агента, має поле типу “*State”*, яке використовується для оперування кінцевим автоматом, та “*Conditions”*, які відображають інформацію, що оперується ігровим агентом в конкретний момент часу.

Ігровий ШІ в *Half-Life* для сприйняття та осмислення навколишнього середовища використовує три механізми: зір, слух, нюх. Зір представлявся конусоподібним об’єктом і визначав область видимості ігрового агента (рис. 1.4). Інформація, яка поступає з фіктивного органу зору, обробляється ігровим агентом та використовується для зміни стану, задачі тощо.

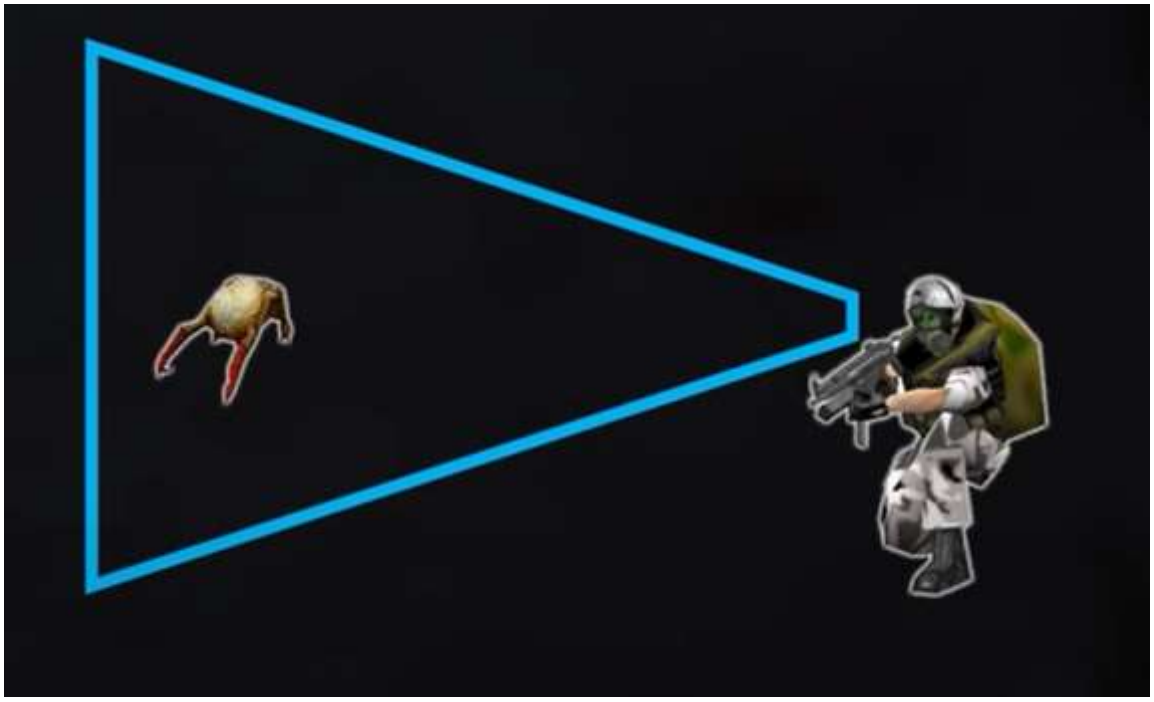

Рис. 1.4. Графічне представлення механізму “зору”

Слух представлений у примітивній формі і являє собою перевірку дистанції до ігрового об’єкту, оскільки саме середовище *Half-Life* не оперувало механічними коливаннями повітря. Так ігровий ШІ “чув” лише ті об’єкти, котрі попадали в радіус “видимості” агента (рис. 1.5).

Рис. 1.5. Графічне представлення механізму “слух”

Останній механізм сприйняття навколишнього середовища - це нюх. Загалом нюх працював як слух, а саме через перевірку відстані, проте мав і свої особливості.

Дані механізми фіксують та оновлюють приватні дані для ігрового агента, що в свою чергу забезпечує механізм життєдіяльності ігрової одиниці. Уся вхідна інформація з рецепторів обробляється ігровим ШІ та з відповідними бітами приватних даних (наприклад, кількість здоров'я) групуються в бінарні "*Conditions*" (рис. 1.6). Набір даних бітів слугує для забезпечення механізму життєдіяльності, оскільки використовуються в логічних переходах між станами кінцевого автомату. Одна така "*Conditions*" складається з 32-бітного цілочисельного типу даних і являє собою набір 1 та 0. Так кожен символ відповідає за певну умову, наприклад: чи бачить агент ворога, чи отримує пошкодження, чи наявні набої тощо. Такий спосіб представлення даних ігрового агента є досить компактним та швидким.

Рис. 1.6. Графічне представлення "*Conditions*"

Ігровий агент приймає рішення, яку дію виконати наступною, на основі наборів таких "*Conditions*". Даний механізм дозволяє використовувати "*Schedule*". *Schedule* – це лінійна серія "*Task*" для виконання ігровим агентом. "*Task*" – мінімальна неподільна одинична інструкція для конкретного ігрового ШІ. Тобто "*Schedule*" - це заданий в момент часу алгоритм дій для виконання ігровою одиницею (рис. 1.7). Кожен "*Task*" підпорядковується конкретному стану ігрового ШІ. Всього їх існує більше 80. Кожна мінімальна неподільна інструкція визначається для використання в конкретному стані кінцевого автомату ігровго ШІ. "*Task*" може бути перевизначеним в конкретному класі для виконання поведінки, притаманної тому чи іншому виду агентів. (Це означає, що класи можуть мати свої конфігурації

однакової поведінки, які відрізняються за порядком або характером виконання). Саме механізм “*Schedule*” дозволяє ігровим агентам мати таку складну поведінку та динамічно змінювати її в процесі життєвого циклу ігрового агента. Також “*Schedule*” містять біт інформації, такий як “*Conditions*”, який специфікує даний “*Schedule*” та при виконанні необхідних умов робить даний “*Schedule*” недійсним та сигналізує для перерозрахунку наступного “*Schedule*”.

Рис. 1.7. Графічне представлення “*Schedule*”

Це створює варіативність та динамічність поведінки ігрового агента. В деяких конкретних випадках ігровий ШІ має виконати деклілька “*Schedule*” послідовно для відтворення досить складної поведінки. Набір послідовно сформованих та відтворених “*Schedule*” називається “*Goal*”. Всього у ігровому продукті використовувались п’ять таких наборів.

Дана варіативність та складність структури ігрових агентів забезпечують розмаїття та динамічність ігрового ШІ у середовищі. А принцип побудови та сама система агентів стала прикладом для подальших реалізацій ігрових ШІ в індустрії в цілому.

1.5. Огляд *Unity* як середовища розробки проєкту

Unity — це промислова платформа розробки дво- та тривимірних додатків та ігрових продуктів.

Даний ігровий рушій використовує такі мови програмування як *C#, JavaScript* і *Boo*. На сьогоднішній день *Unity* є одним із найпопулярніших ігрових рушіїв та активно оновлюється. Створені за допомогою *Unity* додатки працюють на системах *Windows, OS X, Android, Apple iOS*, Linux, а також на гральних консолях *Wii, PlayStation 3 і XBox 360.*

Unity має низку переваг:

- Гнучкість – *Unity* надає легкий спосіб підтримки ігрових додатків та їх модифікації. Наявна потужна система скриптингу на мові програмування *C#*. Продукти створені за допомогою *Unity* можуть мати широкий спектр застосувань та підтримуються великою кількістю платформ / систем.
- Зручність – однією із суттєвих переваг над аналогами є інтуїтивно зрозумілий інтерфейс. Це значно прискорює процес створення ігрового прототипу уникаючи низькорівневого програмування. Доступна можливість використання інструментів візуального редактору логіки.
- Широкий спектр взаємодії – *Unity* надає велику кількість компонентів для реалізації виконання необхідних задач. Компоненти розділені на відповідні тематики. Наявний документований *API* з lоступом до усіх систем *Unity*.
- Доступний каталог ресурсів - *Unity Asset Store* надає доступ до бібліотеки готових ресурсів, активів та інструментів для підвищення продуктивності та якості ігрових додатків.

Загальна структура проєктів, реалізованих за допомогою рушія *Unity*, являє собою взаємодію великої кількості ігрових об’єктів, компонентів, скриптів, префабів на ігровій сцені.

Сцена представляє певне обмежене ігрове середовище. Усі об’єкти поміщаються на сцену та взаємодіють певним чином, який визначається розробниками продукту в процесі створення ігрового додатку.

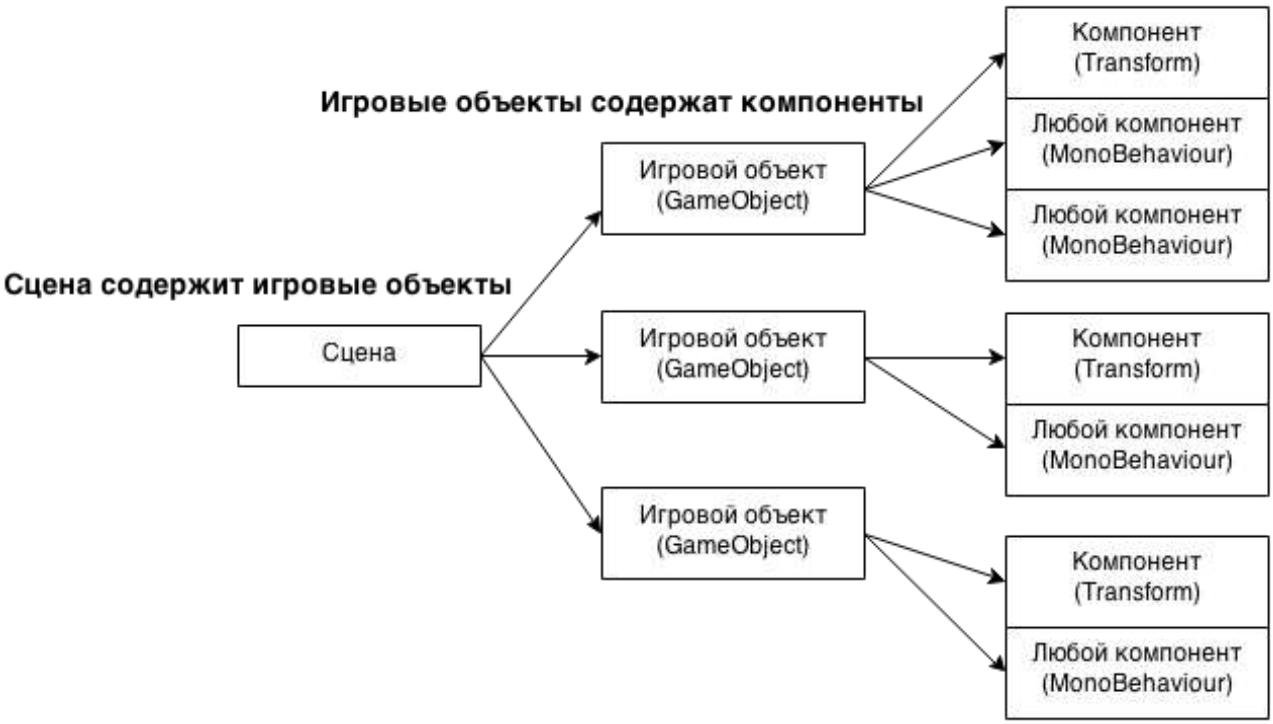

Рис. 1.8. Демонстрація ієрархії структури сцени

Ігрові Об'єкти – це базові функціональні одиниці ігрового середовища. В термінології *Unity* використовується термін "*GameObject*". Взаємодія ігрових одиниць на сцені і визначає сам ігровий процес. При створенні ігрового об’єкту останній має базовий компонент – *Transform*. Даний компонент описує позицію, поворот та масштаб ігрової одиниці в просторі ігрового середовища. Додаючи різні компоненти до об'єкта, ми можемо значно розширити його функціонал для забезпечення реалізації будь-якого бажаного сценарію гри.

Компоненти - це блоки даних, що визначають конкретні властивості будь-яких ігрових об'єктів або подій. Компоненти (*components*) мають різне призначення, вони можуть впливати на поведінку, зовнішній вигляд і багато інших функцій ігрових одиниць. *Unity* надає безліч компонентів різного призначення, які використовуються для реалізації необхідних для розробника задач.

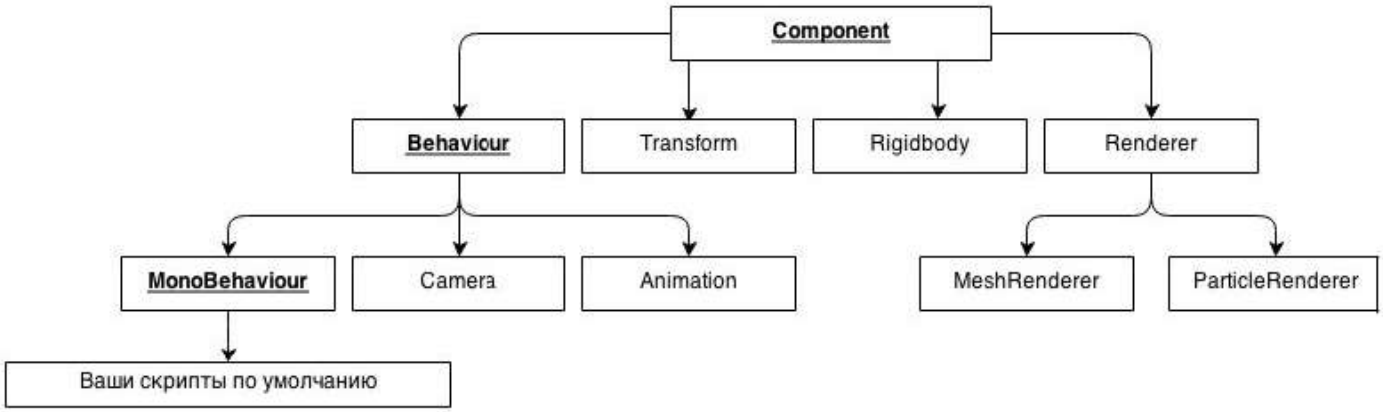

Рис. 1.9. Ієрархія наслідування ігрових компонентів

Для забезпечення інтерактивності різних ігрових елементів, в *Unity* використовуються скрипти. Скрипти є компонентами *Unity*, вони є важливою частиною розробки ігор. Скрипт визначає логіку та поведінку ігрової одиниці так як він являє собою набір інструкцій для виконання. Особливістю скриптів для *Unity* є їхня структура. Кожен скрипт, який прикріплюється до ігрової одиниці, повинен наслідуватись від класу *Monobehaviour*. Даний клас визначає базові функції (такі як *Start*(), *Update*(), *FixedUpdate*(), *Awake*()), які використовуються у ігровому рушії для забезпечення роботи ігрових одиниць.

Основною концепцією *Unity* є використання Ігрових Об'єктів (*Game Object*), проте при необхідності повторного використання даної ігрової одиниці прийнято попередньо зберігати об’єкт у форматі шаблону - префабу. При створенні складних ігрових одиниць з різними компонентами і налаштуваннями префаби значно полегшують процес формування нових ігрових одиниць. Кожен префаб може містити індивідуальні налаштування відносно базового об’єкту. Механізм використання префабів дозволяє багаторазово використовувати ігрову одиницю на сцені (або сценах) без потреби заново формувати даний об’єкт.

Взаємозв’язок та варіативність розглянутих компонентів ігрового рушія формує подальшу структуру ігрового додатку та забезпечує реалізацію необхідних сценаріїв. *Unity* надає велику кількість інструментів, які використовуються при побудові складних систем ігрових одиниць. Так даний ігровий рушій має 2 потоки. Основний потік, де відбувається зміна кадрів та життєвий цикл усіх скриптів

загалом та програми в цілому, і фізичний, який використовується для обробки об’єктів, які задіюють фізику в тій чи іншій мірі. *Unity* надає всі необхідні інструменти для реалізації ігрових продуктів із застосуванням штучного інтелекту, а тому був обраний як середовище для розробки проєкту.

Висновки до розділу

Ігрова індустрія є потужною складовою світової економіки, яка зберігає тенденції на постійний зріст. На даний час ця індустрія є однією з найперспективніших для фінансового та технологічного розвитку. Невід’ємною частиною кожного продукту даної індустрії є ігровий ШІ.

Ігровий ШІ – це складна система, яка складається з багатьох компонентів та призначена для виконання поставлених задач на етапі проєктуванні у ігровому середовищі. Усього виділяють три основних види ігрового ШІ :

- *NPC,*
- *Боти,*
- *Моби.*

Кожен вид має свої особливості та специфікації у будові самої системи, що забезпечує варіативність ігрового ШІ як компоненту відеогри. Основний життєвий цикл ігрового ШІ складається з трьох основних частин : *Sense, Think, Act.*

Unity – це ігровий рушій, який використовується для створення продуктів ігрової індустрії. Дане програмне забезпечення є доступним, легким в освоєнні та має

низку переваг поміж аналогами. Дане програмне забезпечення широко використовується для створення різноманітних ігрових систем, в тому числі і систем ігрового ШІ.

РОЗДІЛ 2
ПРОЄКТУВАННЯ СИСТЕМИ ІГРОВОГО ШІ

2.1. Програмне середовище для реалізації системи ШІ

Середовищем для життєвого циклу та роботи агентів є ігровий додаток власної розробки. Даний додаток представлений у вигляді симбіозу жанрів "*Tower Defense*" та "*RTS*".

Додаток представлений у вигляді інтерактивного середовища, з яким оператор (гравець) може взаємодіяти безпосередньо за допомогою елементів графічного інтерфейсу. Середовище предствлене у формі визначеного набору тайлів – клітинок шестикутної форми, які в сукупності створюють ігрову карту. Даний набір тайлів формує інтерактивний простір. Додаток представлений у форматі економічної стратегії. У процесі гри створюється унікальна економіка шляхом побудови різноманітних ігрових споруд та ігрових одиниць. В наявності є сім видів будівель з унікальним функціоналом та призначенням. Під час ігрової сесії доступний механізм збору ресурсів. За отримані ресурси в процесі розвитку можливе покращення наявного прогресу, що в свою чергу замикає цикл безкінечного розвитку в ігровому середовищі. Також за отримані ресурси можна створювати захисні одиниці, які використовуються для знищення ворожих ігрових одиниць. Загалом можна виділити основний ігровий цикл: *Розвиток економіки -> Побудова захисних споруд -> Захист від ворожого ШІ -> Повтор циклу*. Ціль – зберегти головну будівлю та пройти необхідну кількість ітерацій ігрового циклу.

У даному додатку використовується три види ігрових агентів:

- "Юніт"
- "Турель"
- "Ворог"

2.2. Формування завдань ігрового ШІ

2.2.1. “Юніт”

“Юніт” – це ігрова одиниця, яка використовується для забезпечення механізму збору ресурсів.

“Юніт” безпосередньо взаємодіє з ігровими одиницями, які відносяться до механізму збору ресурсів, для виконання поставлених задач. “Юніт” створюється у визначених будівлях. Після створення ігрового ШІ він переходить у режим очікування. Гравець може призначити даний вид ШІ на виконання колективної задачі по збору ігрових ресурсів. З режиму очікування “Юніт” переходить в режим переміщення і прямує до визначеного місця. По досягненню місця призначення агент розпочинає процес екстракції ігрового ресурсу. По завершенню даного процесу юніт прямує до сховища. По досягненні сховища “Юніт” інкрементує загальну кількість ресурсу визначеного типу. Так замикається цикл добування ігрових ресурсів.

Було визначено наступні вимоги до ігрового агента “Юніт”:

- “Юніт” повинен взаємодіяти із трьома ключовими одиницями:
  - Сховище ігрнових ресурсів “*Storage*”
  - Будівля, яка виробляє ресурс “*WorkPlace*”
  - Будівля, до якої відноситься дана одиниця “*Home*”
- “Юніт” повинен створюватись та відноситись до визначеної будівлі, з якою при створенні агента створюється зв’язок. При відсутності даного зв’язку даний агент повинен ставати неактивним.
- “Юніт” повинен забезпечувати повний цикл механізму збору ігрових ресурсів.
- “Юніт” повинен обробляти інформацію, яка стосується лише його області застосування. Агент не повинен змінювати або перевизначати дані інших ігрових об’єктів, які не задіяні в глобальному процесі життєдіяльності ігрового агента.

Було визначено наступні задачі агенту “Юніт”:

- Збір ігрових одиниць.
- Переміщення ігрових одиниць.
- Додавання ігрових ресурсів до сховища.
- Переміщення в ігровому просторі.
- Фіксування наближення до ігрових одиниць, з якими взаємодіє.
- Можливість переходу в неактивний стан.

2.2.2. “Турель”

“Турель” - це ігрова одиниця, яка є будівлею. Даний ШІ використовується для знищення ворожих ігрових агентів в процесі життєдіяльності додатку.

“Турель” виконує оборонну функцію в масштабах проєкту. Стан даної ігрової одиниці також залежить від зовнішнього чинника – кількості електроенергії. Дана змінна відповідає за можливість виконувати певні функції в даному додатку та регулює стан ігрового ШІ “Турель”. При досягненні певного значення даної змінної агент переходить в неактивний стан. Це значення є динамічним та змінюється під час ігрової сессії. Після створення “Турель” переходить в стан очікування. В даному стані агент виконує функцію патрулювання, тобто очікує появу ворожих ігрових ШІ в полі зору. При появі ворожих агентів ШІ переходить в стан атаки. При нанесенні ушкоджень ворожим ШІ їх одиниці здоров’я зменшується, а при досягнення мінімальної позначки ворог знищується, що і завершує оборонний механізм.

Було визначено наступні вимоги до ігрового агента “ Турель ”:

- “Турель” повинен забезпечувати оборонний механізм в ігровому середовищі.
- “Турель” повинен взаємодіяти з ворожими ігровими одиницями.
- “Турель” повинен фіксувати та реагувати на ворожих ігрових агентів.
- “Турель” повинен аналізувати сукупність ворогів, які потрапляють в область видимості агенту.

- “Турель” повинен опрацьовувати дані, які надходять лише з фіктивних органів чуття даного ШІ. Агент не повинен втручатись або змінювати дані, які не стосуються його області застосування.

Було визначено основні функції ігрового агента “Турель”:

- Фіксування переміщення ворожих одиниць в певному радіусі.
- Динамічна зміна положення в просторі (поворот).
- Аналіз даних.
- Знищення ворожих ігрових агентів.
- Можливість переходу в неактивний стан.

2.2.2. “Ворог”

“Ворог” - це ігрова одиниця, яка представляє протилежну для оператора сторону. Основна задача - бути перешкодою для гравця та завдавати шкоди для ігрової економіки оператора.

“Ворог” - це важливий вид ігрових одиниць, які забезпечують циклічність ігрового процесу. Даний вид ШІ створюється на окремих клітинках ігрової карти. Після появи агенти розпочинають побудову шляху до головної будівлі гравця або до будь-яких ігрових одиниць, які попадають в поле зору. В процесі переміщення по ігровому просторі в область видимості даного агента можуть потрапляти інші будівлі. В такому разі за особистим алгоритмом ворог створює можливі шляхи до будівель, які потрапляють в область видимості, та до головної будівлі і обирає оптимальний шлях. В процесі переміщення оператор може виконати певну функцію, яка відповідає за перехід агенту в неактивний стан. В такому разі ШІ перебуватиме в даному стані визначений час і по завершенню таймера повернеться до попереднього стану. При досягненні цілі агенти переходять в стан атаки та наносять пошкодження ігровим одиницям, які знаходяться під контролем оператора. При досягненні мінімального значення одиниць здоров’я одиниця буде знищена.

Було визначено наступні вимоги до ігрового агента “Ворог”:

- “Ворог” повинен створюватись на визначених клітинках ігрової карти.
- “Ворог” повинен під час виконання програми проводити аналіз навколишнього середовища, обмеженого полем видимості ігрового агента.
- “Ворог” повинен будувати динамічну кількість шляхів до ігрових одиниць, які знаходяться в полі зору агента, та проводити аналіз даних цих шляхів.
- “Ворог” повинен забезпечувати механізм завдання шкоди.
- “Ворог” повинен взаємодіяти з ігровими одиницями оператора.
- “Ворог” повинен забезпечувати механізм завдання шкоди та змінювати дані ігрових одиниць, з якими вони безпосередньо вступають у контакт в процесі механізму атаки.

Було визначено основні функції ігрового агента “Ворог”:

- Динамічна зміна положення в просторі.
- Знищення ігрових одиниць.
- Аналіз даних.
- Можливість переходу в неактивний стан.
- Забезпечення циклічності ігрового процесу.

2.3. Вибір методів реалізації системи ігрового ШІ

Для реалізації функціонування ігрового ШІ в рамках проєкту було визначено концепцію проєктування системи агентів через використання шаблону проєктування “Кінцевий автомат”. Даний метод реалізації ШІ є досить розповсюдженим та гнучким в конкретних специфікаціях.

Кінцевий автомат - особливий різновид абстракції, що використовується для опису шляху зміни стану об'єкта в залежності від поточного стану та інформації, отриманої ззовні [20].

Кінцевий автомат є способом моделювання і реалізації об'єкта, що володіє різними станами протягом свого життя. Основними компонентами кінцевого автомату є стани та переходи між ними (рис. 2.1).

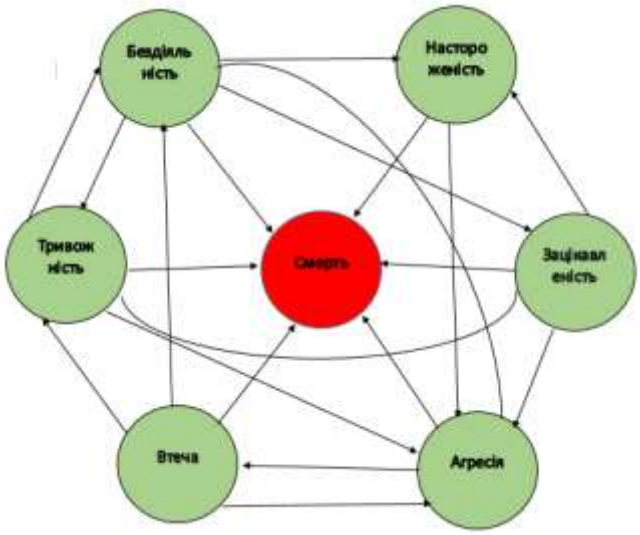

Рис. 2.1. Схема функціонування кінцевого автомату

Стан є складовою одиницею структури кінцевого автомату та представляє фізичні умови або відображає сукупність інструкцій, котрі ігровий агент повинен виконувати в даний момент часу.

Переходами між станами слугують конкретні умови, притаманні для конкретного ігрового агента. Структура та повнота умов може варіюватись в залежності від стану, в якому перебуває ігрова одиниця в момент часу.

Діаграма відображає суть прийняття рішень для цього агента на основі даних, якими оперує ігрова одиниця в конкретний момент часу. Перехід між станами кінцевого автомату забезпечується при виконанні логічних умов, заданих на етапі проєктування. Кожну ітерацію додатку ігровий ШІ опрацьовує інформацію, яка поступає з навколишнього середовища,, та при виконанні логічної умови переходу здійснює зміну поточного стану.

Для реалізації механізму зміни станів було прийнято рішення використовувати дерева рішень. Даний метод є досить варіативним та забезпечує гнучкість та швидкість модифікації логічних умов переходів станів кінцевого автомату.

Дерево рішень - математична модель, яка задає процес прийняття рішень так, що будуть відображені кожне можливе рішення, попередні та наступні за цим рішенням події або інші рішення і наслідки кожного кінцевого рішення [19].

Загальна структура моделі дерева рішень складається з наступних елементів:

- Дуги
- Вузли рішень
- Вузли подій
- Вузли результатів

Дана модель дозволяє забезпечити варіативність вибору рішення ігрового ШІ в конкретний момент часу (рис. 2.2).

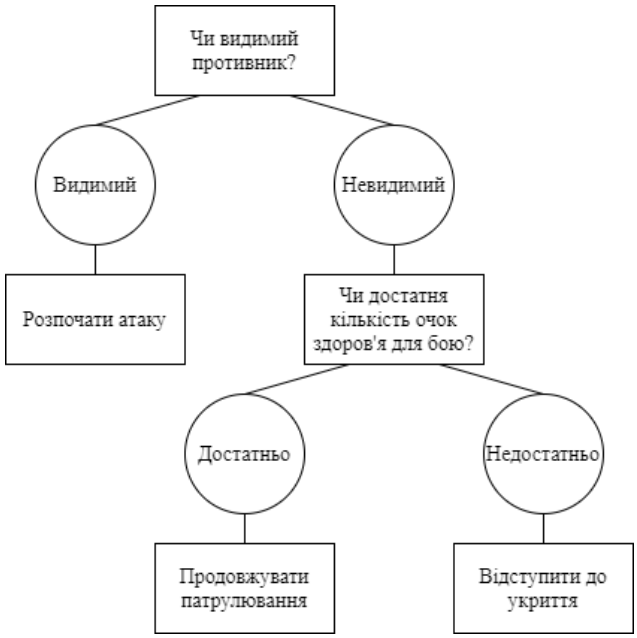

Рис. 2.2. Приклад схеми дерева рішень

У результаті було отримано симбіоз кінцевого автомату та дерев рішень. Отримана конфігурація цілком задовольняє структуру та потреби для організації роботи ігрового ШІ.

2.4. Проєктування кінцевого автомату ігрового ШІ

2.4.1. “Юніт”

Було виділено такі стани ігрового ШІ “Юніт”:

- Стан очікування
- Стан переміщення

- Стан екстракції ігрового ресурсу
- Стан інкрементації ігрового ресурсу
- Неактивний стан

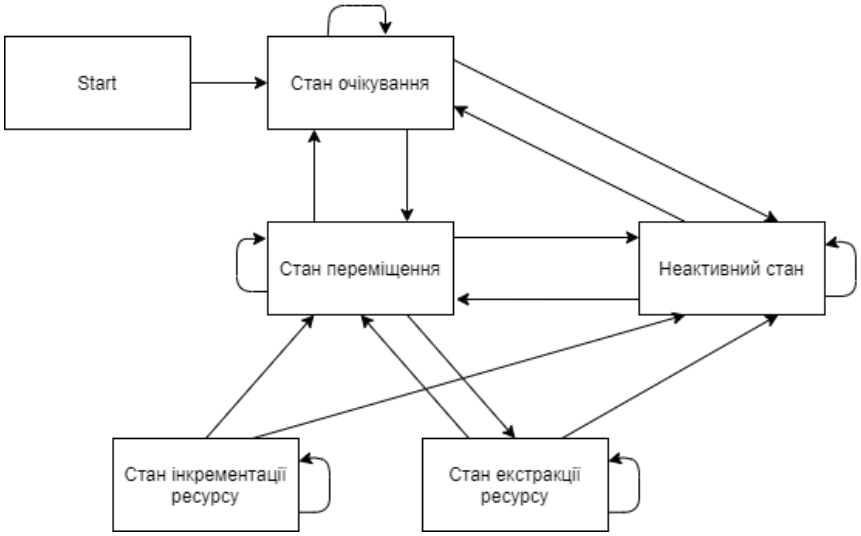

Рис. 2.3. Схема кінцевого автомату для ШІ “Юніт”

Дана схема відображає життєвий цикл та визначає набір можливих інструкцій для виконання в конкретний момент часу.

Стан очікування – початковий базовий стан ігрового агента “Юніт”. Він визначає алгоритм дій та інструкції, які “Юніт” повинен виконувати при відсутності конкретного завдання для виконання. В даному стані “Юніт” не виконує ніяких ігрових задач і перебуває в режимі очікування.

Стан переміщення – ключовий стан, який визначає переміщення агента у просторі. Відповідає за фундаментальний механізм зміни положення ігрової одиниці, який в свою чергу є рушієм для забезпечення циклічності ігрового агента. В даному стані містяться інструкції переміщення даного виду ШІ.

Стан екстракції ігрового ресурсу - це стан ігрового агента, який відповідає за функції добування ігрового ресурсу. Даний стан описує логіку створення та екстакції одиниці з відповідної будівлі “*WorkPlace*”. Завершення даного стану формує приватні змінні агента “Юніт”, які використовуються при інкрементуванні кількості ресурсів у оператора, та об’єкт, який у графічній формі представляє ресурс.

Стан інкрементації ігрових ресурсів – стан, який відповідає за механізм переміщення ігрового ресурсу до сховища “*Storage*”. Виконує функцію для зарахування очок ресурсів в масштабі проєкту. Містить набір інструкцій, які “Юніт” виконує при досягненні сховища.

Неактивний стан – стан ігрового ШІ, в якому ШІ не може виконувати свої безпосередні функції. Настає при відсутності зв’язку з будівлею “*Home*”. Даний стан використовується для забезпечення балансу та коректного функціонування логіки проєкту.

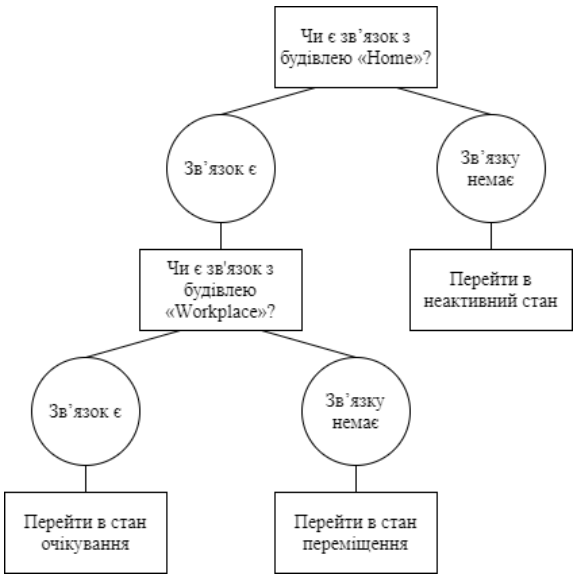

Рис. 2.4. Дерево вибору для стану очікування

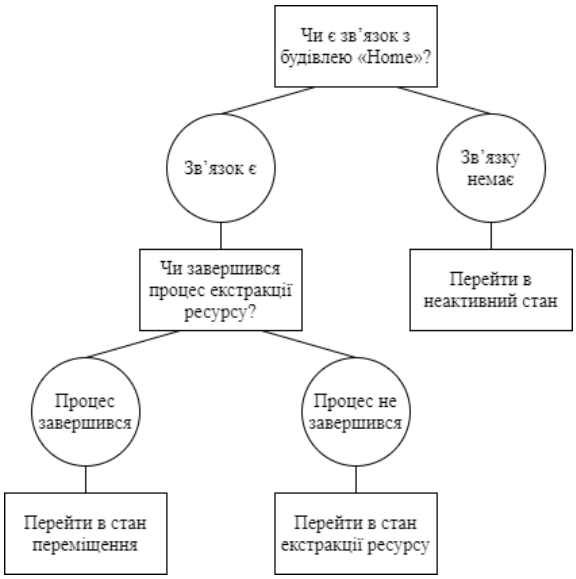

Рис. 2.5. Дерево вибору для стану екстракції ресурсу

Рис. 2.6. Дерево вибору для неактивного стану

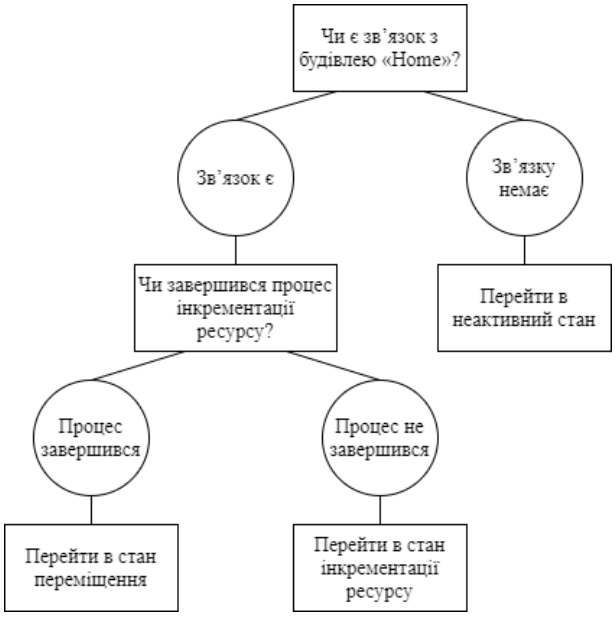

Рис. 2.7. Дерево вибору для стану інкрементації ресурсу

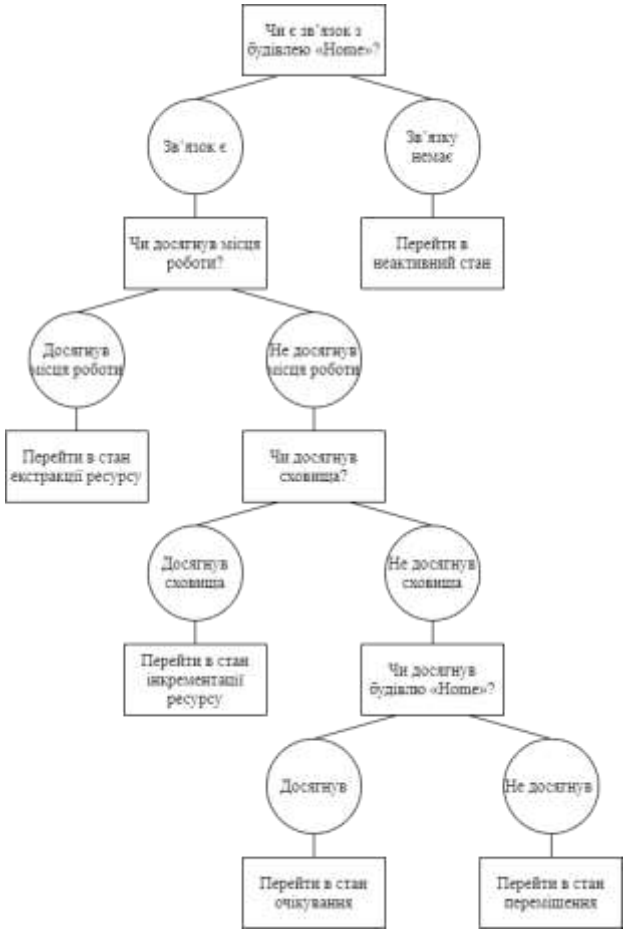

Рис. 2.8. Дерево вибору для стану переміщення

2.4.2. “Турель”

Було виділено такі стани ігрового ШІ:

- Стан очікування
- Стан атаки
- Неактивний стан

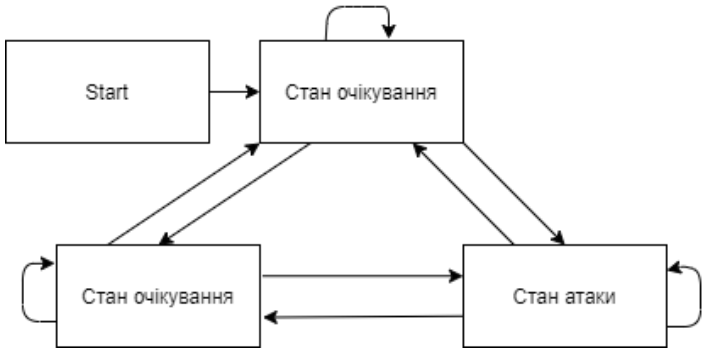

Рис. 2.9. Схема кінцевого автомату для агенту “Турель”

Стан очікування – початковий стан ігрового агента. Він означає, що ШІ знаходиться в режимі очікування. Даний стан визначає поведінку ігрової одиниці при відсутності в полі зору ворожих ігрових об’єктів та за умови наявності необхідної кількості електроенергії (електроенергія тут виступає в форматі змінної, котра регулює стан агента). В даному стані агент займається патрулюванням.

Стан атаки – стан ігрового агента, який визначає поведінку при виявленні в полі зору противника. Є основним станом для забезпечення механізму захисту ігрових одиниць оператора. В даному стані ігровий агент виконує алгоритм дій для знищення противників.

Неактивний стан – стан ігрового агента, в якому ШІ не може виконувати свої безпосередні функції. Настає при відсутності необхідної кількості очок електроенергії. Використовується для забезпечення різноманітності ігрового процесу та для забезпечення функціонування логіки проєкту в цілому. В даному стані ШІ не реагує на появу в полі зору ворожих одиниць.

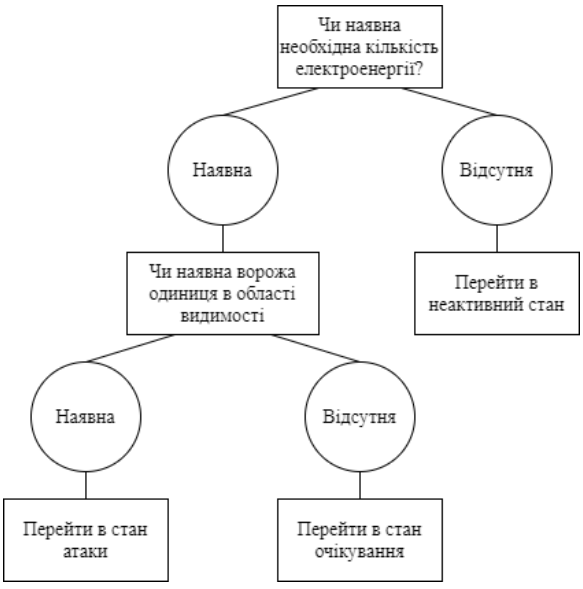

Рис. 2.10. Дерево вибору для усіх станів агенту “Турель”

Дана схема є однаковою для всіх трьох станів ігрового ШІ “Турель”, оскільки кінцевий автомат є повнозв’язним.

2.4.3. “Ворог”

Було виділено такі стани ігрового ШІ:

- Початковий стан
- Стан переміщення
- Неактивний стан
- Стан атаки

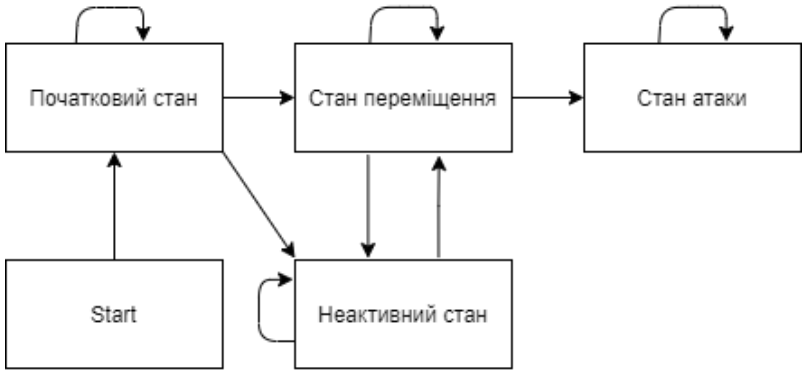

Рис. 2.11. Схема кінцевого автомату для ШІ “Ворог”

Початковий стан – це стан ігрової одиниці, який відповідає за організацію та побудову оптимального шляху для даного агента до головної будівлі оператора або до будівель, які попадають в поле зору даного ШІ.

Стан переміщення – стан ігрового ШІ, який відповідає за переміщення конкретної одиниці в просторі ігрової сцени. Відповідає за механізм зміни положення ігрової одиниці. В даному стані містяться інструкції переміщення.

Неактивний стан – стан одиниці, який відповідає за логіку поведінки агента при накладанні ефекту “оглушення". Даний ефект знаходиться під контролем оператора та вступає в дію при відповідних діях оператора. Даний стан визначає додаткові параметри балансу для нанесення додаткової шкоди ігровим агентам типу “Ворог”. Використовується для забезпечення різноманітності ігрового процесу та для забезпечення функціонування логіки проєкту в цілому. В даному стані ШІ не може змінювати своє положення у просторі та не реагує на появу нових ігрових одиниць оператора.

Стан атаки – ключовий стан агента. Даний стан відповідає за механізм нанесення пошкоджень прогресу оператора. При переході в стан атаки агент виконує персональний алгоритм дій, по завершенню яких ігрові одиниці, які знаходяться в розпорядженні оператора, отримують пошкодження.

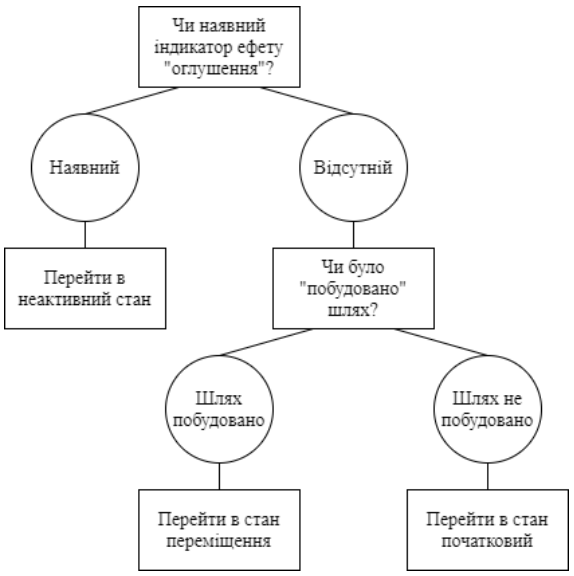

Рис. 2.12. Дерево вибору для стану початковго стану

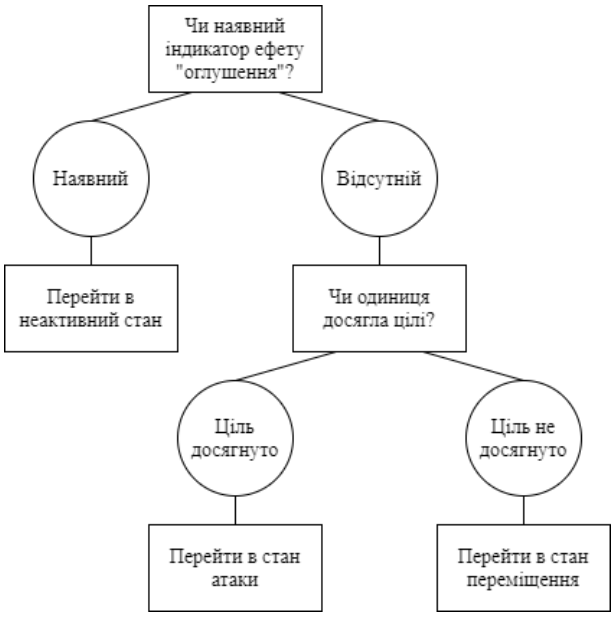

Рис. 2.13. Дерево вибору для стану переміщення

Рис. 2.14. Дерево вибору для неактивного стану

Стан атаки є кінцевим у життєвому циклі ігрового агента "Ворог" та не визначає переходи одиниці у будь-які інші стани.

Висновки до розділу

Проведено етап проєктування системи ігрового ШІ. Визначено ігровий простір для створення та імплементації ШІ. Виділено три існовні види агентів:

- “Юніт”
- “Турель”
- “Ворог”

Визначено перелік вимог, особливостей та функцій для реалізації системи агентів. Визначено метод реалізації системи ігрового ШІ. Спроєктовано основну концепцію функціонування агнетів - кінцевий автомат для всіх видів ШІ, надано специфікацію та детальний опис кожного стану відповідного виду ШІ. Визначені взаємозв’язки між станами кінцевих автоматів агентів.

Загалом була розроблена модель та конкретизовано усі ключові фактори для створення системи ігрового ШІ в додатку.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ІГРОВОГО ШІ

3.1. Структура скриптів ігового ШІ

“*Monobehaviour*” – це базовий клас в *Unity*, який визначає набір функцій життєвого циклу для коректної організації роботи ігрової одиниці або конкретного скрипту в масштабі ігрового додатку (рис. 3.1). Усі скрипти, які наслідуються від “*Monobehaviour*”, можуть бути прикріпленими до ігрових ассетів та переглядатись в інспекторі. Також наслідування від даного класу відкриває можливість для визначення набору функцій, які використовуються у середовищі *Unity* для організації роботи даного скрипту або ігрової одиниці у масштабі ігрового рушія. Так стають доступними певні функції, які кожен скрипт може перевизначати індивідуально.

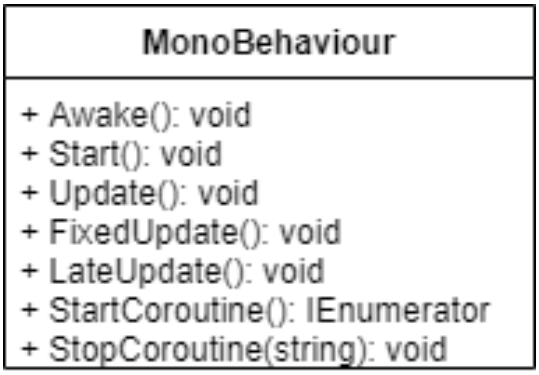

Рис. 3.1. Часткова *UML*-діаграма класу “*MonoBehaviour*”

Усі ігрові ШІ, представлені у даному проєкті, мають низку спільних характеристик. Кожен агент із представлених у даній роботі являє собою “живий” об’єкт, який має характеристику здоров’я та може знищуватись, тобто вмирати. При побудові ігрових ШІ було прийнято рішення використати механізм наслідування для забезпечення спільної логіки даних ігрових одиниць. Кожен із

агентів буде мати свою копію даних та отримає можливість виконання певного функціоналу, який притаманний даному абстрактному класу. “*AliveGameUnit*” - це клас, який відповідає за спільну логіку, що стосується усіх ігрових ШІ, та визначає спільний функціонал, який наявний для всіх “живих” об’єктів (рис. 3.2). Даний клас є абстрактним, адже не потребує реалізації та служить для побудови логіки та підтримки рівня абстракції даного проєкту.

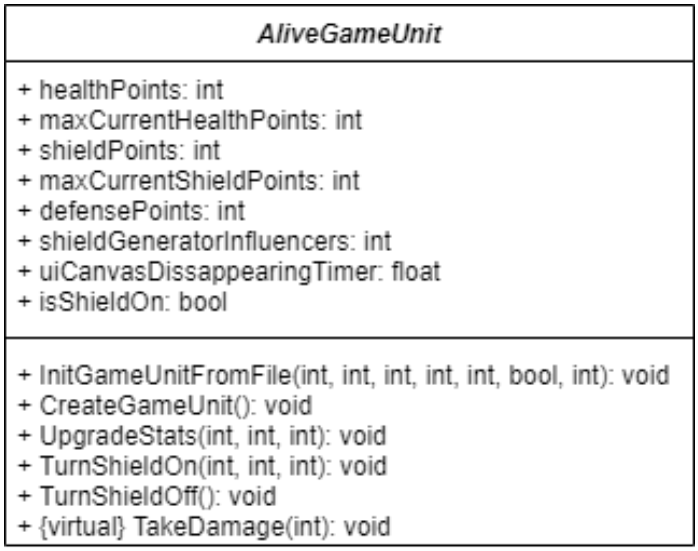

Рис. 3.2. *UML*-діаграма класу “*AliveGameUnit*”

Для отримання доступу скриптів, які будуть використовуватись для організації роботи ШІ, до функціоналу *Unity* було прийнято рішення зробити клас “*AliveGameUnit*” дочірнім до класу “*MonoBehaviour*”. Так усі скрипти, які будуть використовуватись для роботи агентів, зможуть бути прикріпленими до ігрових об’єктів та отримають функціонал *“MonoBehaviour”* для виконання своїх задач. У подальшому дана конструкція буде використовуватись як батьківський клас для подальших скриптів ігрових ШІ. Так кожен агент отримає набір даних та певний функціонал, притаманний “живим” ігровим одиницям, та імплементує необхідні функції “*MonoBehaviour*”, що забезпечить коректну логіку функціонування відповідного скрипту в масштабі проєкту та дасть змогу визначати необхідні інструкції та логіку у відповідних методах даного класу.

3.1.1. “Юніт”

Ієрархія скрипту, який використовується в ШІ “Юніт”, складається з трьох основних частин:

1. Спільна початкова модель ієрархії (“*AliveGameUnit*” -> ”*MonoBehaviour*”) - відповідає за можливість скрипту бути компонентом в середовищі *Unity* та використовувати необхідний функціонал рушія. Також визначає базовий функціонал всіх “живих” об’єктів.
2. Безпосередній клас функціоналу агента (“*Unit*”) - містить набір даних та методів, які використовуються для виконання агентом необхідних задач.
3. Сукупність класів (“*UnitResourceLeavingState*”, “*UnitIdleState*”, “*UnitNoSignalState*”, “*UnitMovingState*”, “*UnitExtractingState*”) – відповідають за стани ігрового ШІ, визначають інтерфейс “*IUnitState*”, що забезпечує належний рівень абстрактції для організації коректної роботи кінцевого автомату, інструкції та логіку у відповідних методах даного класу.

UML-діаграма скрипту наведена в додатку А.1.

3.1.2. “Турель”

Ієрархія скрипту, який використовується в ШІ “Турель” складається з шести основних частин:

1. Спільна початкова модель ієрархії (“*AliveGameUnit*” -> ”*MonoBehaviour*”) - відповідає за можливість скрипту бути компонентом в середовищі *Unity* та використовувати необхідний функціонал рушія. Також визначає базовий функціонал всіх “живих” об’єктів.
2. Безпосередній клас функціоналу агенту (“*Turret*”) – містить сукупність даних та полів, які вкористовуються в усіх класах-наслідниках для забезпечення коректного виконання задач агентом.
3. Сукупність класів (“*TurretPowerOffState*”, “*TurretIdleState*”, “*TurretCombatState*”) – відповідають за стани ігрового ШІ, визначають інтерфейс

“*ITurretState*”, що забезпечує належний рівень абстракції для організації коректної роботи кінцевого автомату.

4. Інтерфейс “*IBuilding*” – інтерфейс, який визначає ігрову одиницю як будівлю та надає необхідний функціонал.

5. Гілка класів “*Laser*” – сукупність класів, які відповідають певному виду даного ШІ.

6. Гілка класів “*Misile*” – сукупність класів, які відповідають певному виду даного ШІ.

UML-діаграма скрипту наведена в додатку А.2.

3.1.3. “Ворог”

Ієрархія скрипту, який використовується в ШІ, “Ворог” складається з трьох основних частин:

1. Спільна початкова модель ієрархії (“*AliveGameUnit*” -> ”*MonoBehaviour*”) - відповідає за можливість скрипту бути компонентом в середовищі *Unity* та використовувати необхідний функціонал рушія. Також визначає базовий функціонал всіх “живих” об’єктів.

2. Сукупність класів (“*Enemy*” -> “*EnemyBomber*”) – дана конструкція забезпечує належний рівень абстракції для функціонування логіки проєкту. Визначає набір даних та методів, які використовує ШІ “Ворог” для виконання безпосередніх задач.

4. Сукупність класів (“*BomberIdleState*”, “*BomberMovingState*”, “*BomberBashState*”, “*BomberAttackState*”) – відповідають за стани ігрового ШІ, визначають інтерфейс “*IBomberState*”, що забезпечує належний рівень абстракції для організації коректної роботи кінцевого автомату.

UML-діаграма скрипту наведена в додатку А.3.

3.2. Розробка префабів ігрового ШІ в *Unity*

Префаб - це особливий тип ресурсу, що дозволяє зберігати весь ігровий об’єкт з усіма компонентами і значеннями властивостей. Префаб виступає в ролі шаблону для створення екземплярів. Властивості, специфікацію та структуру конкретного префабу можна редагувати та налаштовувати в меню префабів середовища *Unity* (рис. 3.3).

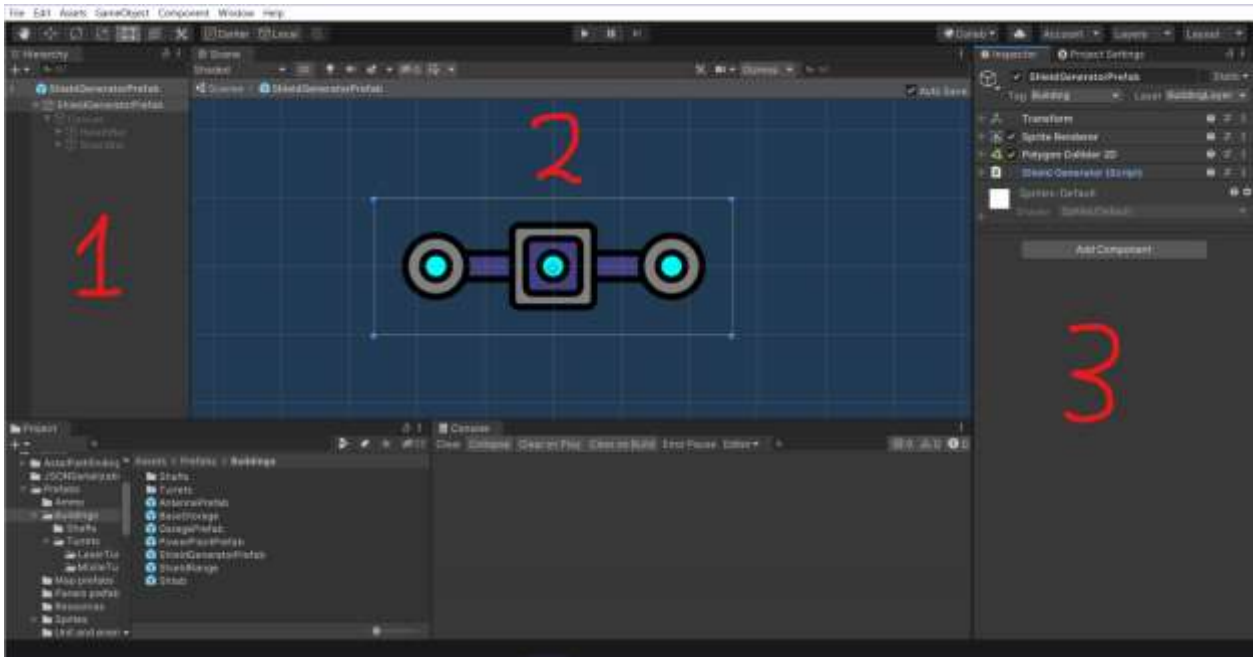

Рис. 3.3. Меню префабу в *Unity*

Меню префабу можна розділити на три головних частини:

1. Вікно ієрархії – відображає ієрархію ігрових об’єктів, які підпорядковуються даному префабу.

2. Сцена графічного представлення префабу – в даному вікні відображається графічна компонента.

3. Вікно інспектора – відображає структуру компонентів префабу.

3.2.1. Структура префабу агенту “Юніт”

Рис. 3.4. Вікно ієрархії об’єкту “*UnitPrefab*”

Префаб складається з двох частин:

- Об’єкт “*UnitPrefab*”
- Полотно “*Canvas*”

“*UnitPrefab*” містить усі компоненти, необхідні для графічного представлення даного агента та для виконання усіх поставлених задач.

“*Canvas*” використовується для графічного представлення додаткової інформації ігрового ШІ. Містить дочірні об’єкти, які використовуються для графічного відображення приватних даних.

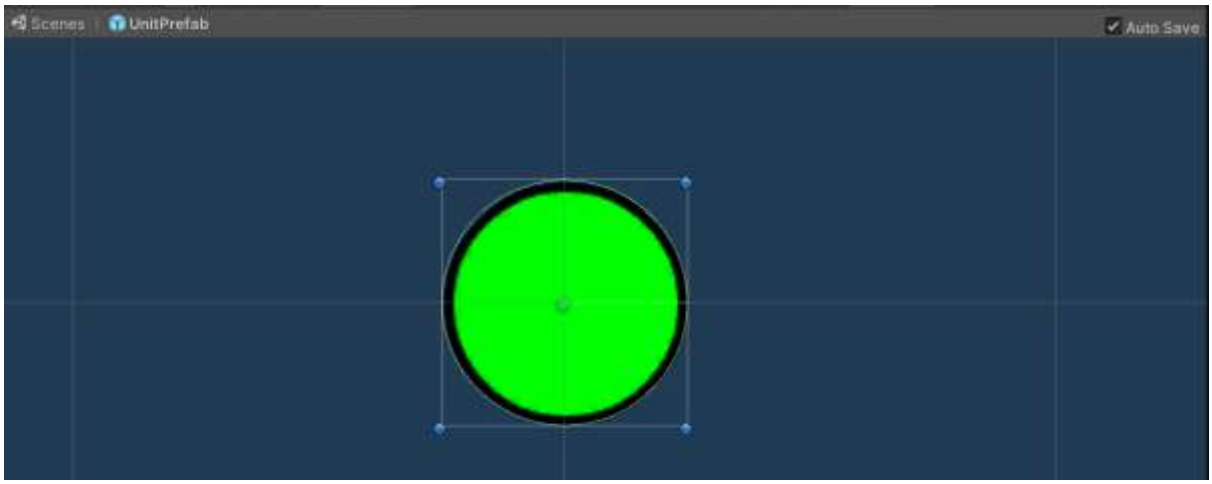

Рис. 3.5. Сцена графічного представлення префабу “*UnitPrefab*”

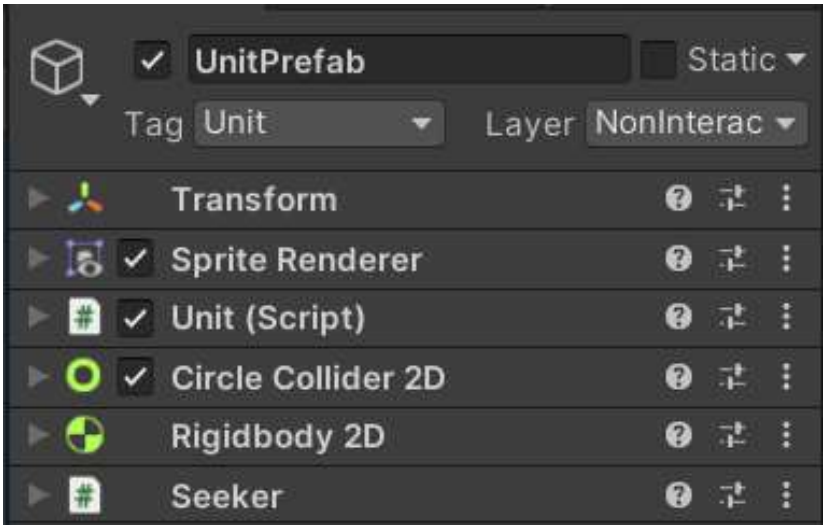

Рис. 3.6. Вікно інспектора об’єкту “*UnitPrefab*”

В даному вікні міститься список компонентів, які прикріплені до префабу та задіяні у різних аспектах конкретного ШІ.

Компоненти об’єкту “*UnitPrefab*”:

- ***Transform*** – компонент, який відповідає за положення ігрового об’єкту у просторі та його поворот.
- ***Sprite Renderer*** – компонент, який відповідає за графічне представлення даного об’єкту.
- ***Unit (Script)*** – скрипт, який містить набір інструкцій та даних для виконання даним ШІ.
- ***Circle Collider 2D*** – компонент, який використовується для колізії ігрових об’єктів.
- ***Rigidbody 2D*** – компонент, який використовується для використання фізики по відношенню до даного об’єкту.
- ***Seeker*** – компонент, який використовується для обробки викликів пошуку шляху.

3.2.2. Структура префабу агенту “Ворог”

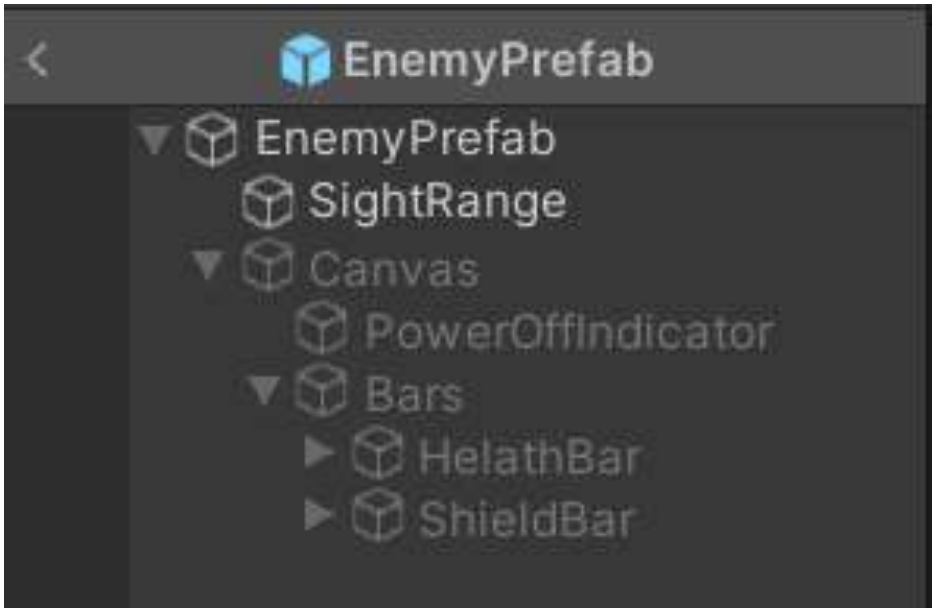

Рис. 3.7. Вікно ієрархії об’єкту “*EnemyPrefab*”

Префаб “*EnemyPrefab*” є шаблоном для створення агенту “Ворог”. Даний префаб складається з трьох основних частин:

- Об’єкт “*EnemyPrefab*”
- Об’єкт фіктивного органу зору “*SightRange*”
- Полотно “*Canvas*”

“*EnemyPrefab*” є основним та містить усі компоненти для виконання усіх поставлених задач даного агента.

“*SightRange*” використовується для сприйняття інформації з навколишнього середовища.

“*Canvas*” використовується для графічного представлення додаткової інформації ігрового ШІ. Містить дочірні об’єкти, які використовуються для графічного відображення приватних даних.

Рис. 3.8. Сцена графічного представлення префабу “*EnemyPrefab*”

На рис. 3.8 видно, що префаб містить два колайдери:

1. Колайдер об’єкту “*EnemyPrefab*”. Даний колайдер використовується для обробки фізичних взаємодій та перетинів безпосередньо з іншими об’єктами.

2. Колайдер об’єкту “*SightRange*”. Даний колайдер використовується для перетину з іншими ігровими одиницями та імітації роботи органу зору.

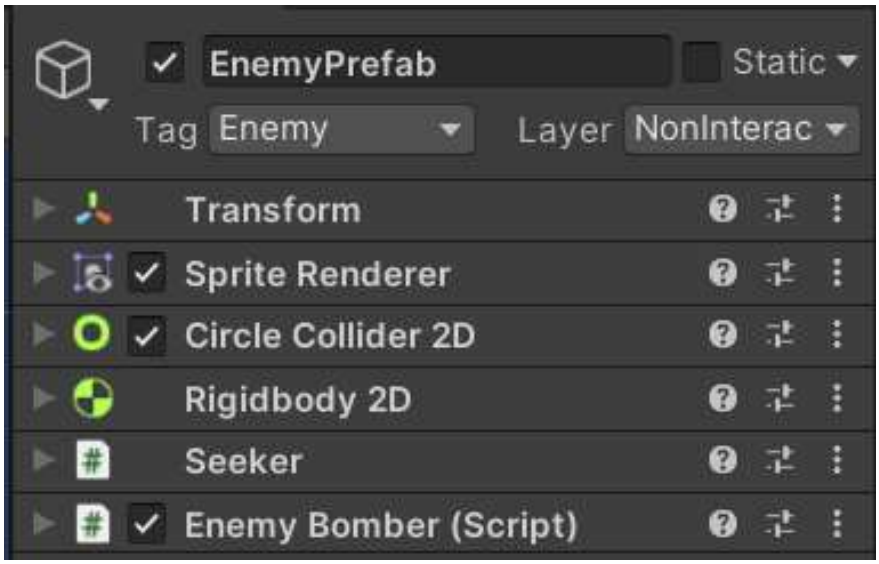

Рис. 3.9. Вікно інспектору об’єкту “*EnemyPrefab*”

Компоненти об’єкту “*EnemyPrefab*”:

- ***Transform*** – компонент який відповідає за положення ігрового об’єкта у просторі та його поворот.

- ***Sprite Renderer*** – компонент, який відповідає за графічне представлення даного об’єкту.
- ***Circle Collider 2D*** – компонент, який використовується для колізії ігрових об’єктів.
- ***Rigidbody 2D*** – компонент, який використовується для використання фізики по відношенню до даного об’єкту.
- ***Seeker*** – компонент, який використовується для обробки викликів пошуку шляху.
- ***EnemyBomber (Script)*** – скрипт, який містить набір інструкцій та даних для виконання даним ШІ.

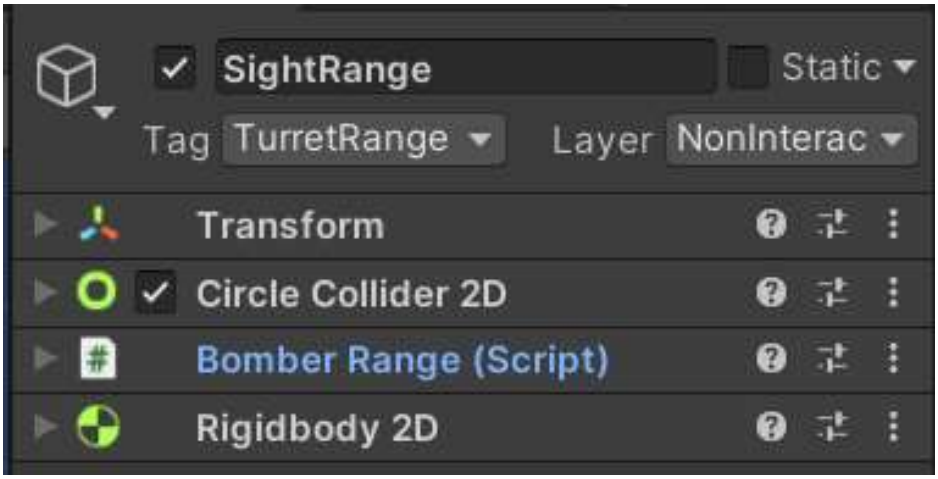

Рис. 3.10. Вікно інспектору об’єкту “*SightRange*”

Компоненти об’єкту “*SightRange*”:

- ***Transform*** – компонент, який відповідає за положення ігрового об’єкта у просторі та його поворот.
- ***Circle Collider 2D*** – компонент, який використовується для колізії ігрових об’єктів.
- ***BomberRange (Script)*** – скрипт, який містить інструкції для коректної роботи фіктивного органу чуття.
- ***Rigidbody 2D*** – компонент, який використовується для використання фізики по відношенню до даного об’єкту.

3.2.3. Структура префабу агенту “Турель”.

В даному пункті розглядається структура префабу агенту “Турель” лазерного типу, першого підтипу.

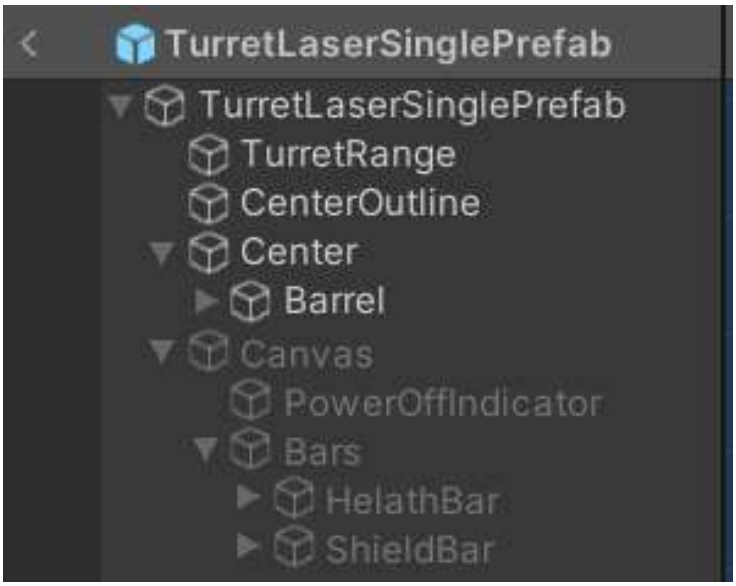

Рис. 3.11. Вікно ієрархії об’єкту “*TurretLaserSinglePrefab*”

Префаб “*TurretLaserSinglePrefab*” є шаблоном для створення агенту “Турель”. Даний префаб складається з трьох основних частин:

- Об’єкт “*TurretLaserSinglePrefab*”
- Об’єкт фіктивного органу зору “*TurretRange*”
- Полотно “*Canvas*”

“*TurretLaserSinglePrefab*” містить усі компоненти, необхідні для графічного представлення даного агенту та для виконання усіх поставлених задач.

“*TurretRange*” використовується для сприйняття інформації з навколишнього середовища. Обробляє перетини з іншими ігровими об’єктами.

“*Canvas*” використовується для графічного представлення додаткової інформації ігрового ШІ. Містить дочірні об’єкти, які використовуються для графічного відображення приватних даних.

Рис. 3.12. Сцена графічного представлення префабу "*TurretLaserSinglePrefab*"

На рис. 3.12 видно, що префаб містить два колайдери:

1. Колайдер власне самого об'єкту " *TurretLaserSinglePrefab* ". Даний колайдер містить форму шестикутника, оскільки даний вид ШІ є будівлею, а всі будівлі, в залежності від кількості зайнятих клітинок карти, мають форму шестикутників, оскільки ігрова карта складається із шестикутників.

2. Колайдер об'єкту " *TurretRange* ". Даний колайдер використовується для перетину з іншими ігровими одиницями та імітації роботи органу зору.

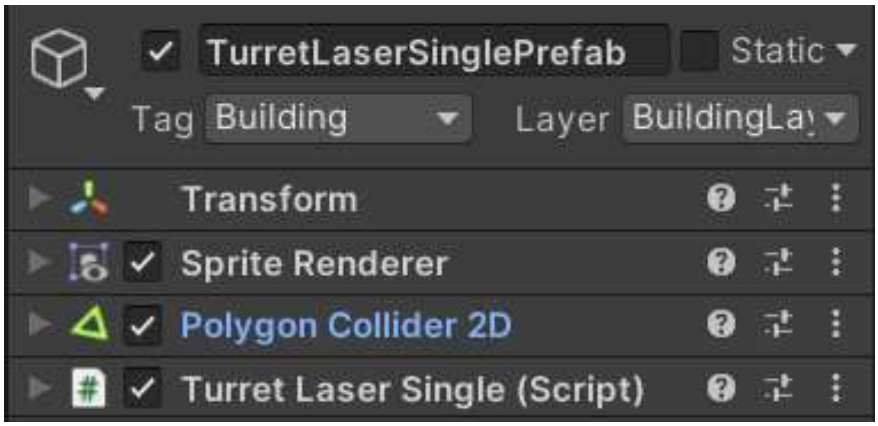

Рис. 3.13. Вікно інспектору об'єкту "*EnemyPrefab*"

Компоненти об'єкту "*EnemyPrefab*":

- ***Transform*** – компонент, який відповідає за положення ігрового об'єкта у просторі та його поворот.

- ***Sprite Renderer*** – компонент, який відповідає за графічне представлення даного об’єкту.
- ***Polygon Collider 2D*** – компонент, який використовується для взаємодії з дотиком оператора (гравця).
- ***Turret Laser Single (Script)*** – скрипт, який містить набір інструкцій та даних для виконання даним ШІ.

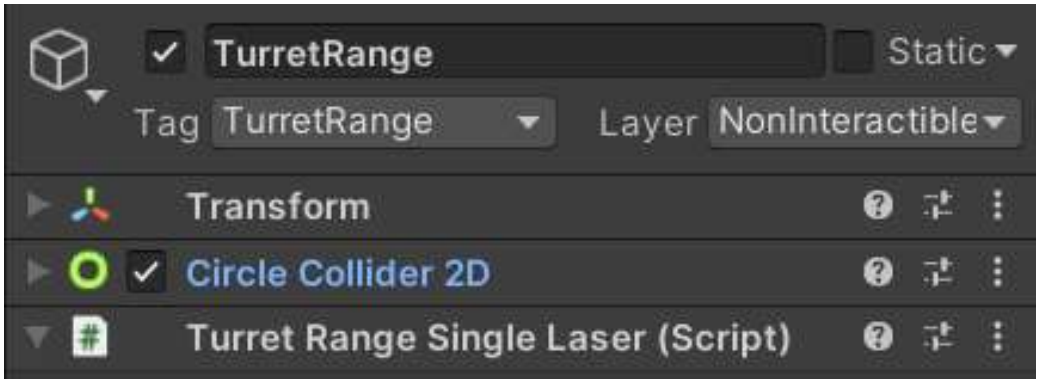

Рис. 3.14. Вікно інспектору об’єкту “*TurretRange*”

Компоненти об’єкту “ *TurretRange* ”:

- ***Transform*** – компонент, який відповідає за положення ігрового об’єкта у просторі та його поворот.
- ***Circle Collider 2D*** – компонент, який використовується для колізії ігрових об’єктів.
- ***Turret Range Single Laser (Script)*** – скрипт, який містить інструкції для коректної роботи фіктивного органу чуття.

3.3. Реалізація пошуку шляху ШІ

3.3.1. Алгоритм пошуку шляху A*

Алгоритм A* - це потужний алгоритм, який постійно використовується у IT сфері. Даний алгоритм має досить широкий спектр застосування. У сфері розробки ігор даний алгоритм має важливе значення - він використовується для вирішення

однієї із фундаментальних проблем, які виникають під час розробки майже всіх ігрових продуктів. Даний алгоритм допомагає вирішити проблему пошуку шляху. Алгоритм A * - один із найпопулярніших методів вирішення завдань на пошук найкоротшого маршруту. Даний алгоритм володіє двома ключовими характеристиками алгоритмів такого роду: оптимальність і повнота.

У алгоритмі A* виділяють основні складові:

- Вузол - всі потенційні унікальні позиції.
- Перехід - переміщення між вершинами або вузлами.
- Початковий вузол - той, від якого починається шлях.
- Кінцевий вузол – вузол, досягнення якого сигналізує про завершення алгоритму.
- Простір пошуку – набір усіх вузлів.
- Вартість - числове значення, що характеризує шлях, який потрібно подолати з даного вузла у наступний.
- *g (n)* - вартість шляху від початкової вершини до іншої.
- *h (n)* - евристичне наближення вартості шляху від вузла n до кінцевого вузла.
- *f (n)* - мінімальна вартість переходу в сусідній вузол.

Сітка, по якій переміщається ігрова одиниця, може мати різну форму. Для організації роботи даного алгоритму використовують “вузли”. Кожен раз при відвідуванні чергового вузла вираховується його вартість *f (n)*, де *n* – це сусідній вузол від даного. Для кожного сусіднього вузла вираховується його вартість, після вирахування вартості обираємо наступний такий, в якого найнижча вартість.

Формула виглядає наступним чином: $f(n) = g(n) + h(n)$

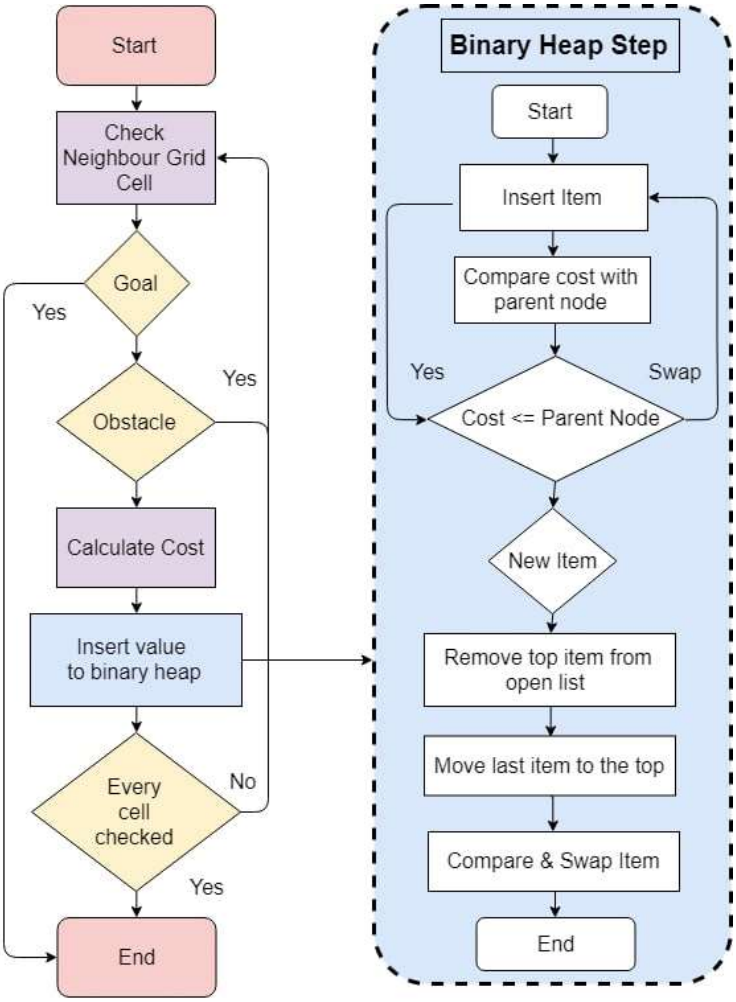

Рис. 3.15. Схема алгоритму A*

Виконавши вищеперераховані дії ми отримуємо список з вузлів, які потрібно пройти ігровій одиниці для досягнення цілі. Даний список і предствляє шлях. Даний алгоритм використовується в проєкті для вирішення проблеми пошуку шляху.

3.3.2. Реалізація пошуку шляху

Реалізація пошуку шляху ігрового ШІ забезпечується за допомогою компоненту “*Seeker*” [10]. Даний компонент забезпечує функціонал скрипту відповідного агента функцією “*StartPath*”. Дана функція формує сукупність координат, які в цілому являють собою шлях між двома точками ігрового простору. Дана функція використовує алгоритм пошуку A* для знаходження найкоротшого шляху між двома точками в ігровому просторі.

В межах додатку існує певна обмеженість на використання пошуку шляху між двома точками, оскільки ігрова одиниця оператора може займати декілька тайлів. В такому випадку виконується більш ресурсозатратний алгоритм знаходження оптимального шляху до ігрової одиниці.

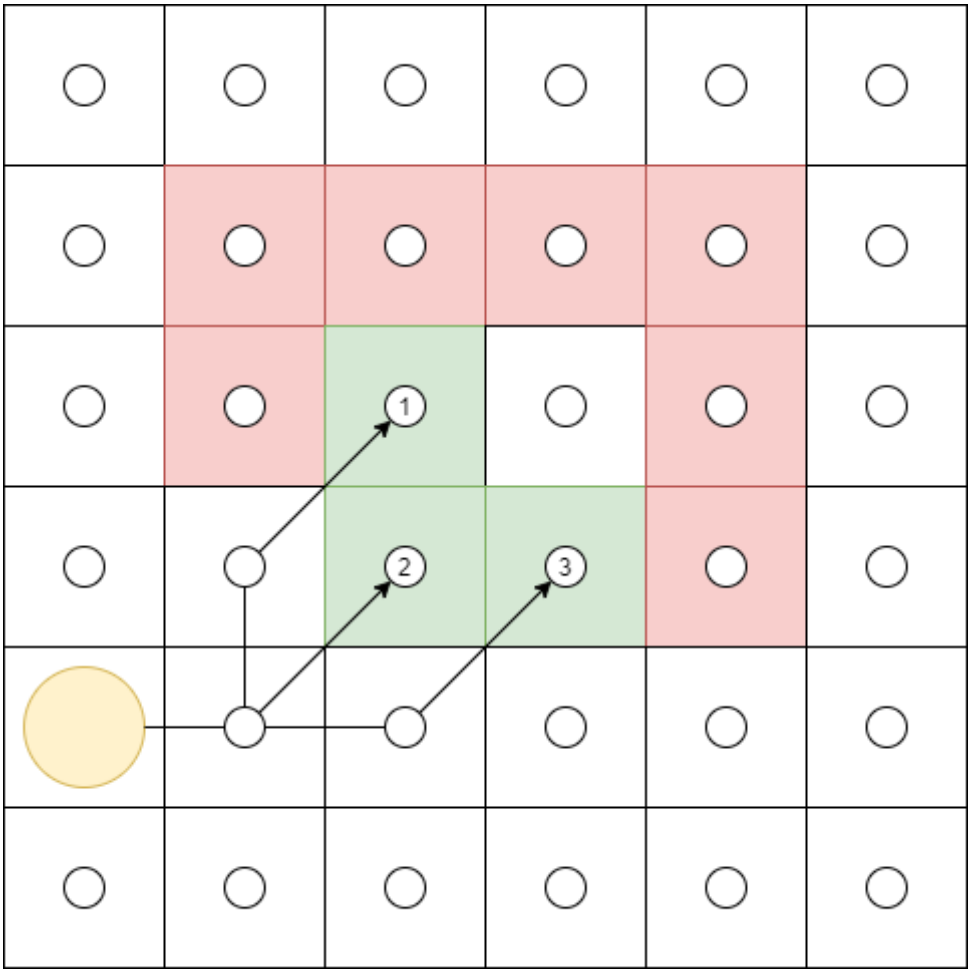

Рис. 3.16. Приклад побудови сукупності шляхів до тайлів ігрового об’єкту

На рис. 3.16 зображена сукупність шляхів до ігрової одиниці, яка займає три тайли. Для коректного визначення оптимального шляху до конкретної ігрової одиниці необхідне знаходження сукупності усіх можливих шляхів до тайлів ігрового об’єкту, визначення клітинок з відкритим доступом та знаходження найкоротшого шляху серед отриманої сукупності.

Кожна ігрова одиниця, яка задіяна в механізмі пошуку шляху, містить перелік координат усіх зайнятих тайлів. У пордальшом ці координати використовуються в функції “*StartPath*” для побудови шляху між двома точками простору.

Алгоритм знаходження оптимального шляху до ігрового об’єкту:

1. Будуємо усі можливі шляхи до всіх тайлів, які займає ігрова одиниця.
2. Визначаємо шляхи, які мають відкритий доступ.
3. Якщо шляхи з відкритим доступом до тайлу існують, формуємо з даних шляхів тимчасовий список.
4. Якщо шляхів з відкритим доступом немає, записуємо наявну сукупність шляхів в тимчасовий список.
5. Якщо тимчасовий список складається з одного елементу, то даний елемент і є найкоротшим шляхом.
6. Якщо тимчасовий список складається більше ніж з одного елементу, порівнюємо кількість елементів в масиві точок координат шляху. Шлях з найменшою кількістю точок проходження є найкоротшим.

Таким чином з усіх наявних умов ми отримуємо оптимальний шлях до ігрової будівлі.

3.3. Тестування системи ігрового ШІ

Тестування отриманої системи ігрового ШІ було проведено за допомгою двох технік: тестування "білої скриньки", тестування "чорної скриньки" [4].

Тестування "білої скриньки" засноване на аналізі керуючої структури програми. Тестуванням за принципом "білої скриньки" перевіряють логіку програми в цілому та окремих складових системи.

Використовуючи техніку тестування "біла скринька" було перевірено:

- Всі можливі випадки виконання програми для всіх логічних рішень.
- Коректність функціонування циклів, які використовуються в коді програми.
- Проаналізовано доцільність використання структур даних для виконання необхідних завдань
- Логічна складова скриптів в цілому та класів, які формують систему ігрового ШІ.

- Коректність побудови всіх елементів програми та правильність їх взаємодії між собою.

Тестування "чорної скриньки" полягає у перевірці роботи кожної функції на всій області визначення. При тестуванні "чорної скриньки" розглядаються системні характеристики програм та ігнорується їхня логічна складова. [4]

Використовуючи дану техніку, система розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням її вхідних та відповідних вихідних даних.

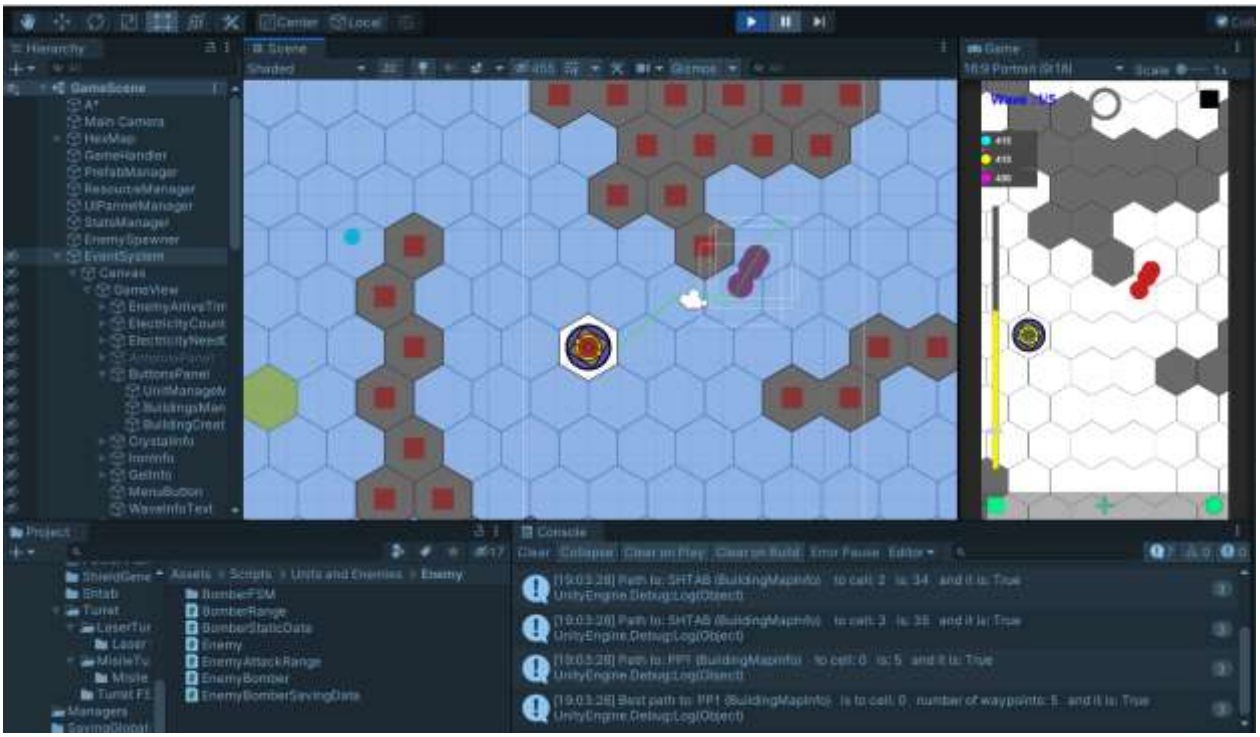

Рис. 3.17. Приклад проведення тестування

Використовуючи техніку тестування "чорна скринька" було перевірено:

- Вхідні дані функцій системи в конкретних умовах
- Вихідні дані функцій системи, які отримані у відповідь на набір вхідних параметрів
- Функціонування методів системи
- Наявність необхідних функцій обробки даних

Для перевікри коректного функціонування комплексних функцій було створено набори тестових випадків та перевірено результати тестів із реальними даними додатку (рис. 3.18).

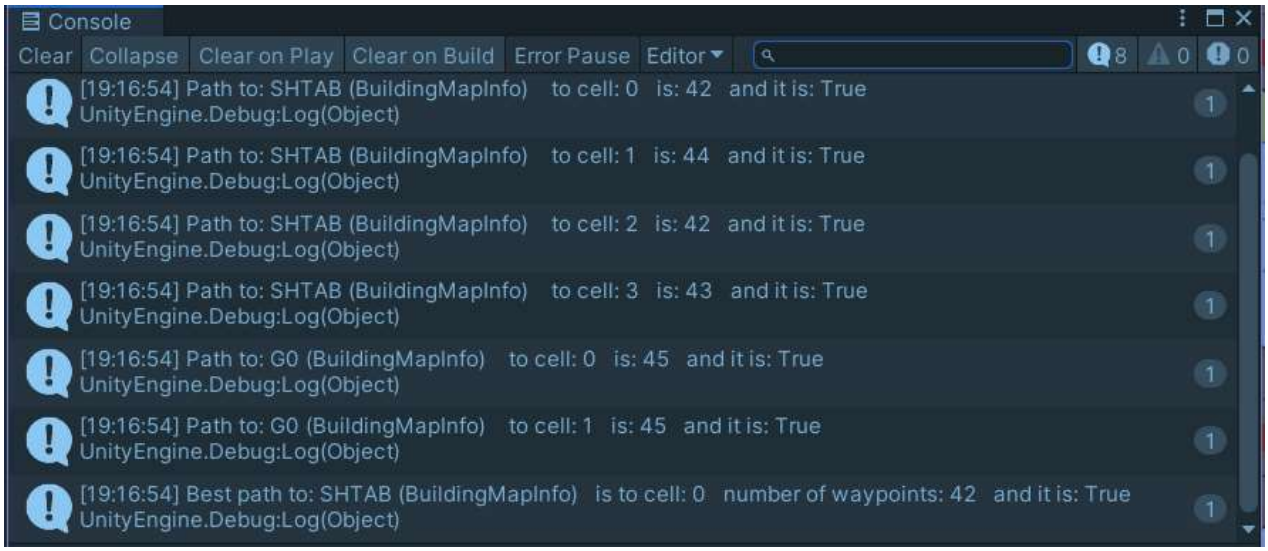

Рис. 3.18. Приклад коректної роботи функції визначення оптимального шляху

Рис. 3.18 наглядно демонструє інформацію про сукупність шляхів, які агент пробудовує до цілі. Кожен шлях із набору має необхідну інформацію:

- Назва цілі
- Номер клітинки із списку клітинок, які ігрова одиниця може займати
- Кількість одиниць карти, які потрібно пройти для досягнення цілі
- Інформація про доступність чи недоступність шляху до конкретної клітинки ігрової одиниці.

Після цього на рис. 3.18 виведена інформація про оптимальний шлях на основі попередніх даних. Дана інформація використовується для перевірки коректності генерації сукупності шляхів та вірного визначення оптимального.

Весь протестований та скомпонований програмний код системи ігрового штучного інтелекту можна переглянути за посиланням: *https://github.com/Nidzee/SpaceStrategyGame*

Висновки до розділу

В даному розділі продемонстрований етап реалізаці системи ігрового ШІ. Всі види агентів в середовищі *Unity* представлені у вигляді префабів – спеціальних шаблонів, які можуть бути легко модифікованими.

Основною частиною системи є ієрархія скрипту, який використовує агент. Наведено опис основних частин скриптів та представлена *UML*-діаграма, яка відображає усі ключові зв'язки та залежності між компонентами класів.

Для усіх скриптів, які керують роботою ігрового ШІ, сформовано ієрархічну конструкцію, яка відкриває можливості для скрипту виконувати необхідні для життєдіяльності об'єкту функції. Також було виділено основні спільні характеристики ігрових ШІ та винесено в окремий клас, який також є частиною ієрархічної конструкції. Так усі скрипти, які наслідуються від даної конструкції, відкривають можливість бути "живими" в ігровому середовищі.

Продемонстрована структура префабів, яка визначає весь можливий функціонал конкретного ШІ. Структура префабів агентів відрізняється в залежності від їх специфікації.

Однією з ключових функцій ігрового ШІ "Ворог" є пошук шляху. Пошук шляху реалізований за допомогою алгоритму A* з використанням додаткового особистого алгоритму визначення оптимального шляху. Дана специфічна функція є однією з найважливіших у системі ігрового ШІ.

ВИСНОВКИ

Проаналізувавши ігрову індустрію в цілому та ігровий штучний інтелект як невід'ємну частину ІТ-галузі економіки, ми з'ясували, що розробка систем ігрових ШІ на сьогоднішній день є актуальною та перспективною діяльністю. Досліджено поняття ігрового ШІ, його види та використання у продуктах ігрової індустрії. Досліджено основні принципи організації роботи ігрових агентів та їх специфікації. Виявлено основний цикл даних систем, кожен компонент якого відповідає за певний алгоритм дій, який повторюється під час ігрової сесії. Розглянуто приклад функціонування системи ігрового ШІ на прикладі гри *Half-Life*. Проаналізовано структуру та підхід до реалізації ігрових агентів та їх проєктування. Проаналізовано *Unity* як середовище розробки системи ігрового штучного інтелекту.

На етапі проєктування використовуючи проаналізовану інформацію визначено види ігрових ШІ для подальшої реалізації. Визначено задачі та обмеження, які накладаються на конкретний вид агента. Для реаліазції системи обрано модель кінцевого автомату. Дана модель відмінно підходить для організації роботи агентів. Для реалізації переходів між станами ігрового ШІ обрано модель дерева рішень. Дана модель забезпечує варіативність вибору рішення агентів у конкретний момент часу.

Проведене проєктування кінцевого автомату для кожного із видів ігрового ШІ, визначено набір станів, відповідних для виконання задач певним видом ШІ, описано всі стани та приведені схеми переходу між цими станами. У результаті після етапу проєктування отримали детальний опис усіх видів ШІ, які використовуються у проєкті, їх задачі та обмеження. Створено схему кінцевого автомату для кожного із видів агентів. Наведено схеми для опису переходів між станами ШІ.

Після формування детального проєкту здійснено реалізацію системи ШІ. Визначено спільний функціонал для агентів та винесено його в окремий модуль, який використовується скриптами ігрових ШІ для виконання поставлених задач.

Створено ієрархію скриптів, які використовуються певним агентом для виконання своїх задач. Кожен скрипт є складеним і формується із багатьох частин, які мають конкретне значення та певний функціонал. Представлена структура префабів кожного із видів створюваного ШІ. Показано унікальність кожного виду агента та його структури. Для організації механізму сприйняття середовища певних видів ігрового ШІ використано додатковий колайдер, який слугує органом зору. Такий спосіб є розповсюдженим та зручним при виконанні даної задачі.

Приведені *UML*-діаграми скриптів усіх агентів. Подано опис усіх полів та методів скриптів, які використовують агенти. Продемонстровано реалізацію власної системи визначення оптимального шляху для ігрового ШІ з використанням алгоритму A*. Дана система є ключовою в циклі життєдіяльності ігрового ШІ “Ворог”. Ігрове середовище має форму карти, яка складається з великої сукупності клітинок. Ігрові одиниці, які приймають участь у життєвому циклі агентів, можуть займати декілька клітинок ігрового простору. В такому разі алгоритм A* для визначення найкоротшого шляху між точками виявився недоречним. Тому для коректної роботи створено унікальну систему, яка містить інформацію про ігрові одиниці, які входять у розпорядження об’єкту, який взаємодіє з ШІ. Дана система генерує сукупність шляхів до всіх клітинок ігрового об’єкту та визначає оптимальний шлях.

В результаті виконання дипломного проєкту створено унікальну систему ігрового штучного інтелекту на базу *Unity*. У ході виконання дипломного проєкту всі поставлені задачі були виконані та досягнуто мету самого проєкту. Проведено аналіз ігрового штучного інтелекту в цілому, здійнено аналіз основних видів ігрових агентів та їх особливості. Визначено основні ключові аспекти системи ігрових ШІ, та на їх основі створено детальний проєкт та реалізовано кінцеву систему в середовищі *Unity*.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goldstone W. Unity 3 x Game Development Essntials / Will Goldstone. – Washington. Packt, 2011. – 488 с.
2. AI Game Programming Wisdom / Rabin Steve / упоряд. S. Rabin. - [Б. м. : б. в.], 2001. – 314 с.
3. Computing machinery and intelligence / упоряд. M. A. Turing. - [Б. м. : б. в.], 1950. – 460 с.
4. Програмна інженерія / упоряд. К. М. Лавріщева. - [Б. м. : б. в.], 2008. – 319 с.
5. Нікітіна Л.О. Моделі та методи штучного інтелекту у комп'ютерних іграх. / Л.О. Нікітіна, С. О. Нікітін. - Х.: Друкарня Мадрид, 2018. - 102 с.
6. Ножов И.М. Практическое применение искусственного интеллекта в компьютерных играх / Ножов И.М. – М. : РГГУ, 2008. – 140 с.
7. Савченко. А.С., Синельніков. О. О. Методи та системи штучного інтелекту / А.С. Савченко, О. О. Синельніков. – К. : НАУ, 2017. – 190 с.
8. Индустрия компьютерных игр / упоряд. И. А. Седых. - [Б. м. : б. в.], 2020. – 74 с.
9. Фримен Эр., Фримен Эл., Сьерра К., Бейтс Б. Паттерны проектирования / Эр. Фримен., Эл. Фримен., К. Сьерра., Б. Бейтс. – СПБ. Питер, 2011. – 647 с.
10. A* Pathfinding Project. Arongranberg.com – Game Development with Unity 3D. [Електронний ресурс]. Інтернет ресурс Arongranberg, 2017. - Режим доступу: https://arongranberg.com/astar/
11. A* Pathfinding (E01: algorithm explanation). [Електронний ресурс]. Sebastian Lague, 2020. - Режим доступу: https://www.youtube.com/watch?v=-L-WgKMFuhE
12. AI: Основы игрового AI - Библиотека программиста. Библиотека программиста. [Електронний ресурс]. Інтернет ресурс Библиотека программиста, 2021. - Режим доступу: https://masandilov.ru/ai/basics_of_game_ai

13. Newzoo games market revenues and audience2020-2023 [Електронний ресурс]. Інтернет платформа NewZoo Platform, 2020. - Режим доступу: https://newzoo.com/insights/articles/newzoo-games-market-numbers-revenues-and-audience-2020-2023/

14. Глобальный обзор игровой индустрии: тренды, инсайты и прогнозы на 2021 год. [Електронний ресурс]. AdIndex, 2020. - Режим доступу: https://adindex.ru/news/researches/2019/01/25/230750.phtml

15. Игровая индустрия: геймдев (gamedev). [Електронний ресурс]. Центр развития компетенций в бизнес информатике Высшей школы бизнеса НИУ ВШЭ., 2020. - Режим доступу: https://hsbi.hse.ru/articles/igrovaya-industriya-geymdev/

16. Индустрия компьютерных игр. Будущее вики. [Електронний ресурс]. Інтернет ресурс Fandom, 2019. - Режим доступу: https://future.fandom.com/ru/wiki/Индустрия_компьютерных_игр

17. Ігровий штучний інтелект. [Електронний ресурс]. Інтернет ресурс Znaimo, 2020. - Режим доступу: https://znaimo.com.ua/Ігровий_штучний_інтелект

18. Как работает искусственный интеллект в играх? [Електронний ресурс]. Інтернет ресурс itProger, 2019. - Режим доступу: https://itproger.com/news/kak-rabotaet-iskusstvenniy-intellekt-v-igrah

19. Метод дерева решений и другие методы на основе графов. [Електронний ресурс]. Інтернет ресурс Function-x, 2017. - Режим доступу: https://function-x.ru/graphs4_modeling_decision_tree_game_tree.html#paragraph1

20. Паттерны/шаблоны проектирования. [Електронний ресурс]. Інтернет ресурс Refactoring guru, 2020. - Режим доступу: https://refactoring.guru/ru/design-patterns/

21. Создание искусственного интеллекта для игр – от проектирования до оптимизации. Все публикации подряд. [Електронний ресурс]. Інтернет ресурс Хабр, 2019. - Режим доступу: https://habr.com/ru/company/intel/blog/265679/

22. Что такое игровой движок? [Електронний ресурс]. Інтернет платформа Funduk, 2018. - Режим доступу: https://funduk.ua/technoblog/gaming-raznoe/chto-takoe-igrovoy-dvizhok/