Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія інтелектуальної поведінки штучних суперників у комп’ютерній грі»

Виконав: студент 2 курсу,
групи 1КН-18 м
спеціальності 122 «Комп'ютерні науки»
Ільченко М. О.
Керівник: д.т.н., проф. Яровий А.А.
Рецензент: к.т.н., доц. Майданюк В.П

Вінниця - 2019 року

ЗАТВЕРДЖУЮ

Завідувач кафедри____КН____

д.т.н., проф.. Яровий А.А.

_____

(підпис)

“____” _____2019 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.006.18.109.ПЗ

Магістранта групи 1КН-18м Ільченка Михайла Олександровича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія інтелектуальної поведінки штучних суперників у комп'ютерній грі»

Вхідні дані: Кількість типів штучної поведінки у грі – не менше 3; кількість вузлів ігрового дерева – не менше 5; підтримка інтегрування технології у двовимірні та трьовимірні ігрові середовища; розмірність ігрового середовища – не менше 10х10 тайлів.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: Схема роботи алгоритму інтелектуальної поведінки штучних суперників, проектування модулю прийняття рішень за методологією IDEF0, проектування модулю пошуку шляху за методологією IDEF0, діаграма класів модулю прийняття рішень, результати тестування інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі.

2. Текстова (пояснювальна записка): вступ, обґрунтування доцільності розробки інтелектуальної поведінки штучних суперників у комп’ютерній грі, аналіз методів інтелектуальної поведінки штучних суперників у комп’ютерній грі, програмна реалізація інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Обґрунтування доцільності розробки інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Аналіз методів інтелектуальної поведінки штучних суперників у комп'ютерній грі			Математичні моделі, розділ 2
3	Програмна реалізація інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі			розділ 3
4	Підготовка економічної частини			розділ 4
5	Апробація та/або впровадження результатів дослідження			тези доповідей
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ д.т.н., професор
(підпис) наук. ступінь, вчене звання (посада)
"____" _____20___ р. _____ А. А. Яровий
ініціали та прізвище

2. Економічна частина _____ к.е.н, доцент кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
Бальзан М.В
ініціали та прізвище

"____" _____20___ р.

Дата попереднього захисту роботи "____" _____20___ р.

Рецензент _____ к.т.н, доц. кафедри ПЗ
(підпис) наук. ступінь, вчене звання (посада)
В.П Майданюк
ініціали та прізвище

Завдання видав
науковий керівник _____ д.т.н., професор
(підпис) наук. ступінь, вчене звання (посада)
А. А. Яровий
ініціали та прізвище

"____" _____20___ р.

Завдання отримав магістрант _____ М.О. Ільченко
(підпис) ініціали та прізвище
"____" _____20___ р.

АНОТАЦІЯ

У даній магістерській кваліфікаційній роботі реалізується інформаційна технологія інтелектуальної поведінки штучних суперників у комп’ютерній грі. Її призначенням є підвищення рівня інтелектуальності поведінки штучних суперників у комп’ютерних іграх.

В ході роботи проведено аналіз предметної області інтелектуальної поведінки штучних суперників у комп’ютерній грі та систем-аналогів. Розглянуто існуючі методи та алгоритми, що можуть бути використані у створенні технології інтелектуальної поведінки штучних суперників у комп’ютерній грі. Спроектовано модулі технології інтелектуальної поведінки штучних суперників.

Здійснено програмну реалізацію технології інтелектуальної поведінки штучних суперників у комп’ютерній грі за допомогою мови програмування Java з використанням фреймворку libGDX.

ABSTRACT

This master's qualification work implements the information technology of the intellectual behavior of artificial adversaries in the computer game. It’s purpose is to increase the level of intelligence of the behavior of artificial opponents in computer games.

During the work, the subject area of intellectual behavior of artificial opponents in the computer game and analog systems were analyzed. Existing methods and algorithms that can be used to create the technology of artificial behavior of artificial adversaries in a computer game were considered. Modules of technology of intellectual behavior of artificial adversaries is designed.

With the usage of Java programming language along with libGDX framework a software implementation of the technology of artificial behavior of artificial adversaries in a computer game was done.

ЗМІСТ

ВСТУП

Актуальність. Від початку 90-х років комп’ютерні ігри із розвитком комп’ютерної техніки стрімко почали займати рівноправне місце у сфері дозвілля поряд із кіно та телебаченням. Із ростом потужності обчислювальних машин та появою нових екранів із більшою роздільною здатністю комп’ютерні ігри набували більш якісну графіку та більш опрацьовані світи. Водночас з'являлись нові жанри, новий геймплей. Важливою складовою кожної гри є штучний інтелект, що із розвитком техніки також прогресував, адже якісний штучний інтелект є досить вимогливим до апаратної складової обчислювальних систем. Не дивлячись на бурхливий розвиток протягом майже 30 років усі аспекти сучасної гри не є ідеальними, ще є безліч шляхів для розвитку. Ідеалом можливо можна буде назвати гру, яку неможливо буде відрізнити від реальності. Тобто таку, де рівень поведінки ігрового світу буде відповідати поведінці реального світу, а візуальна частина буде такою, що мозок люди сприйматиме її за реальність.

Штучний інтелект є однією із найголовніших програмних проблем для створення такого ігрового середовища, адже у створенні ідентичного за поведінкою ігрового світу до реального виникає ряд проблем, які складно вирішити на поточному рівні розвитку області штучного інтелекту.

Залежно від жанру комп’ютерної гри задачі штучного інтелекту ігрових об’єктів та суб’єктів різняться. Але усі вони мають на меті ефективне орієнтування та діяльність у ігровому середовищі. Для суб’єктів це відбувається за допомогою алгоритмів, що імітують природні відчуття живих істот як-то зір, слух, тактильні відчуття, тощо, а також алгоритми що мають на меті імітацію мислення, пам’яті й прийняття рішень відповідно до поточної ситуації у ігровому середовищі. Для об’єктів це алгоритми, що дозволяють імітувати явища неживої природи із урахуванням фізики цих явищ та фізики їх взаємодії між собою.

Отже штучний інтелект застосовується у широкому спектрі задач що

постають у комп’ютерних іграх для створення інтелектуальної поведінки внутрішньоігрових об’єктів.

Зв’язок роботи з науковими програмами, планами, темами.

Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка спеціалізованих засобів штучного інтелекту на основі інтелектуального аналізу даних та машинного навчання».

Мета і завдання досліджень. Метою магістерської кваліфікаційної роботи є підвищення рівня інтелектуальності поведінки штучних суперників у комп’ютерній грі.

Для досягнення мети розробки необхідно виконати наступні задачі:

- здійснити обґрунтування доцільності розробки інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі;
- здійснити аналіз методів та алгоритмів, що можуть бути застосовані у створенні інтелектуальної поведінки штучних суперників у комп’ютерній грі;
- здійснити проектування програмного забезпечення інтелектуальної поведінки штучних суперників;
- обґрунтувати вибір програмного інструментарію для реалізації інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі;
- здійснити програмну реалізацію та тестування інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі.

Об’єктом дослідження є процес інтелектуальної поведінки штучних суперників у ігровому середовищі комп’ютерної гри.

Предметом дослідження є інформаційна технологія та програмні засоби інтелектуальної поведінки штучних суперників у ігровому середовищі.

Методи дослідження. Для досягнення мети дослідження застосовувалися методи та моделі подання знань про внутрішньоігрове середовище, дерева рішень, метод Монте-Карло для пошуку рішень у дереві, метод пошуку шляху у графі, метод кидання променів (ray casting), методи розпаралелювання обчислень, методи ООП.

Наукова новизна одержаних результатів.

1. Удосконалено існуючу модель інтелектуальної поведінки штучних суперників у комп'ютерних іграх, що відрізняється від відомих використанням комплексної моделі щодо прийняття рішень, пошуку шляху, та імітації зору штучних суперників з одночасним розпаралелюванням обчислень, що забезпечило підвищення рівня інтелектуальності штучних супротивників у комп'ютерній грі.

2. Розроблено інформаційну технологію інтелектуальної поведінки штучних суперників у комп'ютерній грі на основі удосконаленої моделі, що забезпечило підвищення рівня інтелектуальності та підвищення швидкодії при прийнятті рішень.

Практичне значення одержаних результатів полягає у розробленому на основі проведених досліджень програмному забезпеченні, що реалізує технологію інтелектуальної поведінки штучних суперників у комп'ютерній грі.

Розроблена інформаційна технологія підвищує рівень інтелектуальності поведінки штучних суперників у комп'ютерній грі завдяки наступним аспектам:

– розроблено алгоритми для підвищення рівня інтелектуальності штучних суперників у прийнятті рішень;

– програмно реалізовано систему інтелектуальної поведінки штучних суперників у комп'ютерній грі;

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації

інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Результати даної магістерської кваліфікаційної роботи отримані самостійно. В публікації у співавторстві здобувачу належить аналіз особливостей застосування моделі скінченного автомата при розробці комп’ютерних ігор [1].

Апробація результатів роботи. Результати досліджень було апробовано на XLVII науково-технічній конференції професорсько-викладацького складу, співробітників та студентів Вінницького національного технічного університету у 2019р.

Публікації. За основними результатами досліджень опубліковано одну публікацію та подано заявку про реєстрацію авторского права на твір (комп’ютерну програму) [1].

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОЇ ПОВЕДІНКИ ШТУЧНИХ СУПЕРНИКІВ У КОМП'ЮТЕРНІЙ ГРІ

1.1 Аналіз предметної області інтелектуальної поведінки штучних суперників у комп'ютерній грі

Інтелектуальна поведінка штучних суперників у комп'ютерній грі – це набір дій штучних суперників у ігровому середовищі, метою яких є створення ілюзії розумності цих самих штучних суперників. Це відбувається через реалістичну відповідність реакцій штучного суперника на зміни у ігровому середовищі. За інтелектуальну поведінку суперників відповідають алгоритми штучного інтелекту (ШІ). Їх реалізація має значний вплив на геймплей, апаратні вимоги й бюджет гри. Тому існує необхідність балансу у розробці такого штучного інтелекту між його потужністю, ресурсами для його роботи та ціною розробки. Задачею ігрового штучного інтелекту є вдала імітація інтелектуальності та створення цікавого геймплею у рамках ігрового середовища.

На даний момент лише невелика частка ігрових ШІ базується на машинному навчанні. Серед них можна виділити ШІ, що реалізований у програмі AlphaGo, що у 2016 році перемогла гравця у Го вищого рангу – Лі Седоля. AlphaGo тренувалася через спостереження багатьох історичних матчів з Го. Однак термін «штучний інтелект» у комп'ютерних іграх не обмежений лише самонавчальним ШІ.

Замість того щоб навчатись тому, як найкраще перемогти гравців-людей, ШІ у комп'ютерних іграх призначений для покращення ігрового процесу людей. Це є причиною відмінності підходу до створення ігрового штучного інтелекту від підходу створення традиційного штучного інтелекту. Ігровий штучний інтелект часто представляє собою не «істинний штучний інтелект», що означає

що він не базується на алгоритмах машинного навчання. В основному це попередньо визначений та лімітований набір відповідей на попередньо визначений набір вхідних даних [2].

Штучний інтелект у комп'ютерних іграх використовується для створення чуйної, адаптивної або інтелектуальної поведінки, схожої на людську для неігрових персонажів та штучних суперників. Неігрові персонажі – це будь-який персонаж гри, що не контролюється гравцем та/або не є ворожим по відношенню до гравця [3]. Роль штучного інтелекту у відеоіграх значно виросла з того часу як його почали використовувати у виробництві комп'ютерних ігор. Сучасні відеоігри часто включають у себе технології штучного інтелекту, такі як пошук шляху та дерева рішень для керування діями неігрових персонажів. Окрім цього штучний інтелект часто використовується у прихованих для гравця механізмах, таких як процедурна генерація ігрового вмісту [4].

Термін «ігровий штучний інтелект» використовується для позначення широкого набору алгоритмів, тому ігровий штучний інтелект може являти собою не «істинний штучний інтелект», так як ці алгоритми можуть не вирішувати задачі машинного навчання, а лише виконувати «автоматизовані обчислення» або давати заздалегідь визначений набір відповідей на визначений набір вхідних даних [5]. Так як ігровий штучний інтелект для неігрових персонажів фокусується на вигляді інтелектуальної поведінки та якісного ігрового досвіду у рамках ігрового середовища, підходи до його створення відрізняються від підходів до створення традиційного штучного інтелекту.

Алгоритми ігрового штучного інтелекту використовуються для різних задач всередині гри. Найбільш частими є задачі прийняття рішень неігровими персонажами для створення сценаріїв дій яких створюються скрипти, що реалізують дерева рішень. Часто результатом таких дерев рішень, що пишуться програмістом вручну, є неякісний штучний інтелект, поведінка якого повторюється, що негативно впливає на ігровий процес. Іншим недоліком може бути аномальна поведінка штучного персонажа у ситуаціях, що не були

передбачені програмістом [6]. Такі проблеми можна вирішити більш складними деревами рішень, та станів, перехід між якими відбувається не прямо, а за певними алгоритмами, що оцінюють загальний стан ігрового середовища, і роблять оптимальний вибір.

Іншою задачею, що часто постає у розробці штучного інтелекту неігрових персонажів для ігор є пошук шляху у ігровому середовищі. Пошук шляху – це метод визначення яким чином перемістити неігрового персонажа з однієї точки ігрового середовища в іншу, враховуючи ландшафт, перешкоди, та можливо «туман війни» [7]. Туман війни – це невизначеність у знаннях про поточну ситуацію у грі, а саме про свої можливості, можливості супротивника, а також знання про поточний стан ігрового середовища [8]. Комерційні відеоігри у переважній більшості використовують швидкі та відносно прості алгоритми пошуку шляху, що базуються на сітці координат. Ігрове середовище ділиться сіткою координат на квадрати, з яким працюють алгоритми пошуку шляху для графів, такі як A* [9]. Деякі ігри використовують для даної задачі не жорстку сітку, а нерівні полігони, з яких збирається навігаційна сітка областей ігрового середовища, куди може потрапити неігровий персонаж [10]. Третім методом може слугувати створення точок шляху між якими переміщуються ігрові об’єкти. Але рух персонажів, що використовують такий метод виглядає ненатурально, тому такий підхід слід використовувати для руху «неживих» ігрових об’єктів, наприклад платформ, що циклічно рухаються по чітко заданому шляху. Окрім статичного пошуку шляху, навігація неігрових персонажів передбачає динамічний пошук шляху у ігровому середовищі, що постійно змінюється. Це ускладнює пошук шляху постійною зміною напрямку, з необхідністю уникати перешкод, що рухаються, та можливою задачею кооперації з іншими персонажами для групової навігації.

В контексті комп’ютерних ігор штучний інтелект може включати у себе механіки омани гравця, тобто програміст може надати неігровим персонажам інформацію та можливість робити певні дії, що були б недоступні для гравця-

людини у такій же ігровій ситуації [11]. Наприклад штучний супротивник хоче дізнатись чи гравець знаходиться поряд із ним. Цю задачу можна вирішити створенням складної системи сенсорів, відповідних природнім, як-то зір або слух. Або штучний супротивник може просто «запросити» у ігрового рушія поточну позицію гравця. Розповсюдженими варіаціями такої поведінки можуть бути підвищення швидкості штучних суперників у гоночних іграх щоб наздогнати гравця, або надання штучним суперникам більш вигідну стартову позицію у іграх жанру стратегія або шутер. Необхідність використання штучного інтелекту з обманною механікою відображає складність створення істинної інтелектуальної поведінки. Загалом, у іграх де важливе креативне стратегічне мислення, люди з легкістю можуть перемогти штучного суперника при мінімальній кількості спроб та помилок, у разі якщо штучний інтелект не користується обманними механіками. Такі механіки часто реалізуються у ігровому рушії через проблеми із продуктивністю. При цьому важливо щоб ефект обману не був помітним для гравця. Хоча обманні механіки надають деяку перевагу штучним супротивникам, вони не повинні володіти неприродними для людини швидкістю та точністю, притаманними комп’ютеру. Штучні супротивники мають помилятися як це роблять люди для створення ілюзії натурального штучного інтелекту, дії якого виглядають схожими на людські. В іншому випадку гравець почне вважати, що комп’ютер грає нечесно.

1.2 Аналіз об’єкту проектування

Інформаційна технологія інтелектуальної поведінки штучних суперників у комп’ютерній грі повинна керувати поведінкою штучних суперників у деякому ігровому середовищі. Для спрощення процесу тестування та більшої наочності результатів роботи технології буде доцільно реалізувати її у складі гри жанру покрокової стратегії.

Відповідно ігрове середовище має являти собою карту, що складається із квадратних полів за принципом шахової дошки. Кожне поле – це певна місцевість, що може містити певні ігрові об’єкти, як-то скала, озеро, шипи, фортецю, аптечка, зброя та ін. Кожна місцевість впливає на певні характеристики персонажу що там знаходиться, та має власний рівень прохідності.

Гравець та штучні супротивники мають певні характеристики, здатності та типи атаки. Мета кожної із сторін полягає у винищенні супротивника.

У складі інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі мають бути присутні модуль прийняття оптимального рішення для кожного кроку із урахуванням того, як буде розвиватись ситуація у майбутньому та модуль пошуку шляху на місцевості.

Обчислення модулю прийняття рішень слід розпаралелити для підвищення швидкодії модулю.

Вхідними даними для інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі є доступні ходи для кожного кроку гри та сітка ігрового середовища для пошуку шляху. Вихідними даними є оптимальне рішення для кожного кроку гри та найкоротший шлях до певної точки ігрового середовища.

Для коректної роботи комп’ютерної гри, у склад якої входить інформаційна технологія інтелектуальної поведінки штучних суперників система повинна відповідати наступним вимогам:

- центральний процесор – двох’ядерний або більше, від 2ГГц ;
- об’єм оперативної пам’яті – від 1Гб;
- відеокарта з об’ємом пам’яті – від 1Гб та шиною від 128 біт ;
- операційна система сімейства Windows 7\8\10, Linux або macOS;
- встановлена програмна платформа JRE;

1.3 Характеристика та аналіз аналогів

Оскільки проектована інформаційна технологія інтелектуальної поведінки штучних суперників буде реалізовуватись у двовимірній комп'ютерній грі, доречно буде провести аналіз декількох двовимірних ігор, що також містять у собі систему інтелектуальної поведінки штучних суперників. Механіка штучного інтелекту для ігрових суперників має подібні задачі у ряді різних жанрів, тому до аналізу буде залучено декілька ігор різних жанрів.

Enter the Gungeon – гра у жанрі top-down shooter, рис.1.1. Вона складається з п'яти рівнів, що генеруються випадковим чином. Суть гри полягає у тому, що гравець має пройти усі рівні, що представляють собою лабіринти з кімнатами. У кожній кімнаті гравця зустрічають штучні суперники, поведінка яких керується штучним інтелектом. Вони переміщуються по кімнаті перманентно переслідуючи гравця із метою його ураження.

Рисунок 1.1 – Ігровий процес Enter the Gungeon

Це є прикладом штучного інтелекту, що постійно звертається до ігрового рушія для отримання поточної позиції гравця. У штучних суперників відсутні

штучні модулі сенсорів зору та слуху. Для того щоб вибратися із кімнати гравець має убити усіх штучних суперників за допомогою різноманітної зброї, яку можна знайти мандруючи лабіринтом. Під час бою штучні суперники переслідують гравця та стріляють у нього. Інколи вони ховаються за інші ігрові об’єкти, як-то столи, діжки, стіни. Кожен ворог має лише один унікальний паттерн атаки, що ніяк не змінюється від рівня до рівня і не комбінується з іншими здатностями.

Гра є динамічною та цікавою, деякий час вона тримає гравця в азарті, протие наведений приклад поведінки штучних суперників є простим та недосконалим. Його можна покращити уведенням більшої кількості можливих дій, типів атак, й оцінюванням перспективності кожної із доступних сукупності дій. Оптимальний набір дій для кожного моменту ігрового процесу має віднаходитись у дереві поточно доступних рішень. Наприклад якщо у штучного ворога залишилось замало «здоров’я» він почне тікати від гравця й шукати аптечку. Як тільки він віднайде її та відновить рівень «здоров’я», він знову почне атакувати гравця. Але якщо і у гравця низький рівень «здоров’я», штучний суперник може вирішити провести контратаку, жертвуючи собою, але й забираючи з собою гравця, тим самим поліпшуючи свій ігровий рахунок. Іншим прикладом інтелектуальної поведінки може бути пам’ять штучного супротивника. Наприклад ворог помітив діжку із вибухівкою. Він може запам’ятати цей факт, і потім повернутися до неї, знайшовши сірники. Коли гравець наблизиться до ворога біля діжки, ворог підпалить її задля ураження гравця. Ще одним прикладом можливого покращення інтелектуальності штучних ворогів може бути кооперація, у процесі якої вони перекривають декілька доступних виходів із кімнати, таким чином блокуючи та оточуючи гравця задля створення останньому труднощів у проходженні кімнати.

Dwarf Fortress – це комп’ютерна гра, що поєднує у собі елементи симулятора будування та управління і roguelike. Суть її основного режиму полягає у тому що гравець непрямим чином керує групою гномів щоб збудувати підземну фортецю у процедурно згенерованому світі, що має згенерований

ландшафт, об’єкти та тисячолітню історію. Гравець встановлює завдання, які необхідно виконати, а гноми, керовані штучним інтелектом самі вирішують порядок та час виконання робіт, поєднуючи роботу із персональними потребами, імітуючи реальне життя та побут середньовічних людей. На їхнє життя впливають погодні умови, історично складені умови, їх особиста персоналізація та різноманітні вороги, що на дії яких також вливає безліч факторів. Кожної миті стан світу змінюється, і відповідно до цього гноми приймають рішення про наступний крок, що не завжди є оптимальним, так як рушій штучного інтелекту у грі значним чином покладається на обрання випадкових дій, із простору доступних варіантів. При цьому початкові наміри гнома можуть бути оптимальними для певної ситуації, але у результаті дії певних факторів ігрового середовища, гном не зможе вчинити як планувалося, що веде до невизначених наслідків. Різноманіття ситуацій гри та непередбачуваність майбутнього є основною її перевагою, це робить процес гри дуже захоплюючим.

Гра використовує мінімалістичний дизайн, задля спрощення розробки та підвищення продуктивності ігрового рушія, який щосекунди обробляє численну кількість об’єктів світу, та їх взаємодію під впливом сотень факторів. Приклад ігрового процесу зображено на рис. 1.2. Уся інформація про події у світі та причини цих подій виводиться у вигляді історії у спеціальному вікні.

Рисунок 1.2 – Ігровий процес Dwarf Fortress

Суттєвим недоліком як ігрового рушія так і системи штучного інтелекту даної гри є його однопоточність, тобто усі обчислення досить складного ігрового середовища проходять на одному ядрі процесора незалежно від їх фізичної кількості. Із ростом населення фортеці кожен крок обчислення займає все більше часу, що врешті решт робить ігровий процес занадто повільним та нецікавим.

Отже при розробці інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі слід реалізувати багатопоточне обчислення для уникнення проблеми, притаманної Dwarf Fortress. Це, у разі довготривалої підтримки та модифікації технології інтелектуальної поведінки штучних суперників, що проектується, не дозволить ігровому процесу деградувати.

1.4 Прикладне значення та застосування інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі

Штучний інтелект неігрових персонажів є однією із ключових систем у переважній більшості сучасних відеоігор. Незалежно від жанру, до якої належить гра, методи та алгоритми, що використовуються для розробки інтелектуальної поведінки неігрових персонажів, знаходять своє втілення не тільки в ній, а ще у багатьох інших ігор інших жанрів. Тому розвиток та поліпшення підходів до розробки методів та алгоритмів штучного інтелекту в одній певній грі сприяє прогресу усієї галузі. Значну роль у цьому відіграє відкрите програмне забезпечення. За допомогою нього світова спільнота розробників ігор ділиться між собою напрацюваннями у галузі. Зокрема для програмної реалізації проектованої технології інтелектуальної поведінки штучних суперників у комп'ютерній грі планується використати фреймворк із відкритим програмним кодом libGDX. Результат напрацювань проектованої технології також буде відкрито для спільноти, що внесе свій внесок у розвиток штучного інтелекту для комп'ютерних ігор.

Технологія, що проектується буде реалізована як загальне універсальне програмне забезпечення, що можна буде підключити у вигляді додаткової бібліотеки у будь-який ігровий проект, що потребуватиме її методи штучного інтелекту для прийняття супротивниками та неігровими персонажами рішень у ігровому середовищі, координації та пересування. Інтегрування бібліотеки у новий ігровий проект потребуватиме мінімальну кількість змін коду, для адаптації вхідних та вихідних даних гри до даних технології штучного інтелекту.

Це, а також легкість її розширення роблять проектовану технологію інтелектуальної поведінки штучних суперників у комп'ютерній грі зручною для застосування у багатьох ігрових проектах.

1.5 Висновок

Під час роботи над розділом обґрунтування доцільності розробки інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі проаналізовано предметну область інтелектуальної поведінки штучних суперників у комп'ютерній грі. Визначено сутність ігрового штучного інтелекту та основних підходів до його створення.

В ході аналізу об'єкту проектування визначено: основні вимоги до інформаційної технології, вхідні і вихідні дані для системи штучного інтелекту та вимоги до програмно-апаратного забезпечення системи.

В результаті аналізу систем-аналогів, що входять до складу ігор: Enter the Gungeon та Dwarf Fortress визначено їх характеристики, переваги та недоліки.

Під час аналізу прикладного значення та застосування визначено що штучний інтелект є важливою складовою відеоігор, а поліпшення методів та підходів до його створення сприятимуть розвитку усієї галузі.

Отже в даному розділі магістерської кваліфікаційної роботи визначено сутність та задачі штучного інтелекту неігрових персонажів та шляхи їх вирішення, що дало основні дані для подальшого дослідження.

2 АНАЛІЗ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОЇ ПОВЕДІНКИ ШТУЧНИХ СУПЕРНИКІВ У КОМП'ЮТЕРНІЙ ГРІ

2.1 Аналіз методів алгоритмів прийняття рішень

Найбільш розповсюджена роль ШІ у комп'ютерних іграх – це керування неігровими персонажами, тобто персонажами, що не керуються гравцем. Одним із найбільш використовуваним алгоритмом для придання «розумності» діям неігрових персонажів є алгоритм скінченного автомату.

Скінченний автомат, є обчислювальною моделлю, яка може бути використана для імітації послідовної логіки, або, іншими словами, для представлення і управління потоком виконання дій. Скінченні автомати можуть бути використані для підвищення інтелектуальності у багатьох областях, включаючи комп'ютерні ігри [12]. У комп'ютерних іграх він дозволяє створювати дерево рішень для штучного інтелекту неігрових персонажів.

Скінченний автомат є моделлю обчислення, що має один або декілька станів. Одночасно може бути активним лише одне стан. Це означає, що модель повинна переходити з одного стану в інший, щоб виконувати різні дії. Скінченним автоматом гіпотетично може бути будь-який об'єкт, що знаходиться у певному стані у певний час. Стан змінюватиметься на основі вхідних даних, використовуючи отриманий результат для впровадження подальших змін [12].

Основними ознаками скінченного автомата є:

- існує фіксований набір станів, в яких може бути модель;
- модель може перебувати тільки в одному стані одночасно;
- модель отримує послідовність вхідних даних;
- кожен стан має набір переходів, кожен перехід пов'язаний з вхідними даними і вказує на стан.

Скінченні автомати поділяються на детерміновані і недетерміновані [13]. Детермінованим скінченним автоматом (ДСА) є такий автомат, у якому для кожного вхідного символу в кожному стані повинна бути одна перехідна функція. Недетермінований скінченний автомат (НСА) є узагальненням детермінованого. На відміну від ДСА, НСА не обов'язково повинні мати перехідних функцій для кожного символу, і можуть мати декілька функцій переходу в одному і тому ж стані для одного символу. Крім того, НСА можуть використовувати нульові переходи. Нульові переходи дозволяють моделі переходити з одного стану в інший без необхідності зчитувати символ [13].

Найбільш популярним способом модифікації скінченного автомату є скінченний автомат на основі стеку. Без нього скінченний автомат не має концепції історії. Він знає свій поточний стан, але не може повернутися до попереднього стану. Для вирішення цієї проблеми, використовується стек, який зберігає елементи в стилі LIFO (Last In, First Out), щоб зберегти різні стани. Це означає, що поточний стан знаходиться на вершині стека. Потім, коли необхідно перейти до нового стану, ми додаємо новий стан до стеку, і цей стан стає поточним станом. Після відпрацювання дій стану, він вилучається із стеку і попередній стан стає поточним станом [13]. Існує декілька варіантів зміни стану з використанням стеку, а саме:

- вилучення із стеку поточного стану та додавання нового;
- вилучення із стеку поточного стану;
- додавання нового стану до стеку.

Багато відомих комп'ютерних ігор, таких як BattleField, Call of Duty та Serious Sam містять у собі успішні приклади використання моделі скінченного автомату у створенні внутрішньоігрового штучного інтелекту для неігрових персонажів.

Очевидним недоліком алгоритму скінченного автомату є його передбачуваність. Уся поведінка неігрових персонажів, побудована на моделі скінченного автомату є незмінною, тому після декількох партій у гру, ШІ якої

побудований лише на моделі скінченного автомату, гравець втратить інтерес до гри. Більш просунутим алгоритмом, що дозволяє створити більш різноманітний ігровий досвід є алгоритм Монте-Карло для пошуку у дереві (Monte Carlo Tree Search або MCTS).

Алгоритм MCTS – це евристичний алгоритм пошуку для процесів прийняття рішень, що втілює стратегію використання випадкових випробувань для вирішення проблеми. Алгоритм аналізує всі можливі рухи у дереві можливих рішень обчислюючи найбільш вигідну гілку дій на даний момент. Після здійснення відповідного попереднім обчисленням кроку, ШІ повторить етап пошуку по дереву можливих варіантів дій. У комп'ютерних іграх ШІ на базі даного алгоритму може обчислити тисячі можливих дій і вибрати ті, що принесуть найбільше вигоди (наприклад, більше золота). Основна увага у MCTS приділяється аналізу найбільш перспективних кроків, розширенню дерева пошуку на основі випадкової вибірки простору пошуку. Застосування MCTS в іграх базується на низці програшів гри. На кожному програші гри програються усі варіанти можливих дій до перемоги, вибираючи ходи навмання. Кінцевий результат гри кожного програшу використовується для зважування вузлів в ігровому дереві. Найкращі вузли отримують пріоритет бути зіграними у майбутніх програшах.

Основний спосіб використання програшів – застосувати однакову кількість програшів після кожного ходу штучного суперника, а потім вибрати хід, який призвів до найкращих результатів [14]. Ефективність цього методу часто покращується з часом, оскільки більше програшів надається діям, які часто призводили до перемоги штучного суперника у попередніх програшах. Кожен раунд пошуку дерев Монте-Карло складається з чотирьох кроків: [15]

– Вибір: починаючи з кореня R обираються послідовні дочірні вузли, поки не буде досягнуто вузла L. Корінь R – це поточний стан гри, а вузол L – це будь-який вузол, з якого ще не було ініційовано моделювання (програш).

– Розширення: допоки L остаточно не завершить гру (тобто виграш/програш/нічия) для будь-якого гравця, створюється один (або більше) дочірніх вузлів і серед них обирається вузол C. Дочірні вузли – це будь-які можливі дії від ігрової позиції L.
– Симуляція: грається одна випадкова гра з вузла C. Цей крок також називають програшем або моделюванням. Симуляція може зводитись до вибору випадкових ходів, поки гра не буде завершена (тобто виграш/програш/нічия).
– Зворотне розповсюдження: оновлення даних кожного вузла від C до R.

На рисунку 1.1 зображено кроки, що беруть участь в одному рішенні, причому кожен вузол показує відношення виграшів до загального числа розіграшів починаючи з цього вузла в ігровому дереві для гравця, що представлений вузлом [16].

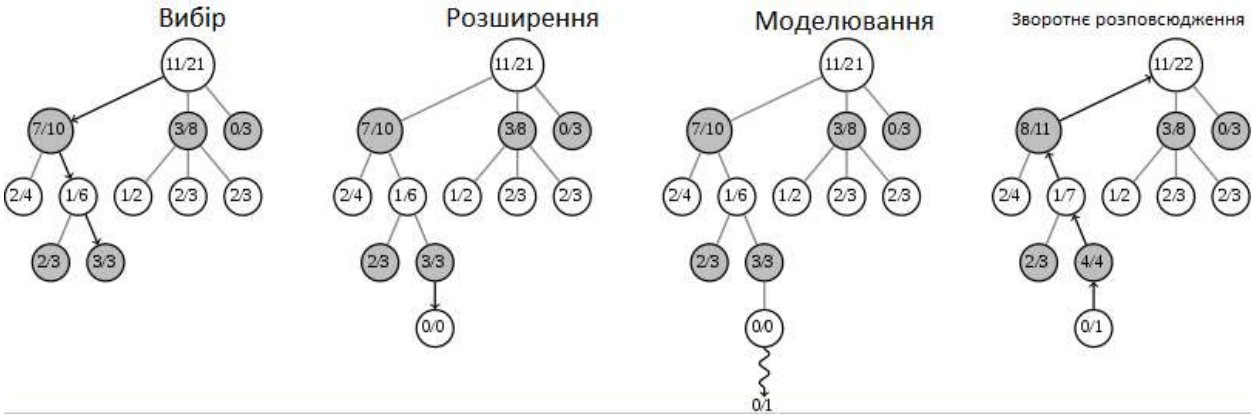

Рисунок 1.1 – Кроки алгоритму MCTS

На діаграмі Вибору показано хід чорних. Кореневий вузол показує, що з цієї позиції на даний момент є 11 перемог з 21 розіграшів для білих. Нижче показано три чорні вузли кожен з яких являє собою можливий чорний хід. Чорні мають загалом 10/21 виграшів. Якщо білий програє у моделюванні, під час Вибору усі вузли збільшують кількість їх моделювання (знаменник), але серед них лише чорні вузли збільшують лічильник виграшу (чисельник). Якщо

перемогли білі, під час Вибору усі вузли збільшують кількість їх моделювання, але серед них лише білі збільшують лічильник виграшу. У іграх, де можливі нічиї, нічия призводить до збільшення чисельника чорних і білих на 0,5, а знаменника на 1. Раунди пошуку повторюються до тих пір, поки залишається час, відведений на хід. У якості кінцевого вибору ходу є хід з найбільшою кількістю зроблених моделювань (тобто найвищим знаменником). Основна складність у виборі дочірніх вузлів полягає у підтримці певного балансу між експлуатацією глибоких варіантів після ходів з високим середнім коефіцієнтом виграшу та дослідженням ходів за допомогою декількох моделювань. Цю проблему можна вирішити за формулою Upper Confidence bound applied to Trees (UCT), що обирає у кожному вузлі ігрового дерева дію, для якої вираз [17]:

$$\frac{w_i}{n_i} + c\sqrt{\frac{lnN_i}{n_i}} \tag{1.1}$$

має найвище значення. У цій формулі:

- w_i означає кількість виграшів для вузла після і-го ходу;
- n_i означає кількість моделювань для вузла після і-го ходу;
- N_i означає загальну кількість моделювань після і-го ходу батьківського вузла відносно поточного;
- c означає параметр дослідження, що теоретично дорівнює $\sqrt{2}$; на практиці значення обирається емпірично.

Перший компонент вищезгаданої формули відповідає експлуатації; він відповідає ходам з високим середнім коефіцієнтом виграшу. Другий компонент відповідає дослідженню; він відповідає ходам з невеликим числом моделюванням [17].

2.2 Аналіз модифікацій методу Монте-Карло для пошуку у дереві рішень

Існує декілька модифікацій алгоритму Монте-Карло для пошуку у дереві, що направлені на скорочення часу пошуку. Деякі з них використовують експертні знання з гри. Алгоритм Монте-Карло може використовувати так звані легкі та важкі симуляції. Легка симуляція складається з випадкових ходів, а важка симуляція використовує деяку евристичну функцію для того, щоб впливати на вибір ходу [18]. Евристична функція може використовувати результати попередніх симуляцій або експертні знання з гри. Наприклад у багатьох ШІ для гри в Го певні шаблони позицій каменів у певному секторі ігрової дошки впливають на можливість ходу у тому секторі [19]. Парадоксально, але іноді неоптимальні ходи у симуляції роблять гру ШІ із використанням алгоритму Монте-Карло загалом сильнішим. Експертні знання можуть бути використані при побудові дерева рішень, щоб допомогти ШІ обрати деякі варіанти ходу. Одна із моделей, що використовує експертні знання призначає ненульові початкові значення перемог та кількості симуляцій дочірнім ходам, що призводить до штучно підвищеного або заниженого середнього коефіцієнту перемог, що впливає на шанс обрання цього ходу під час етапу Вибір [20]. Споріднений метод, що називається «прогресивне зміщення» полягяє у додаванні до формули UCT вираз $\frac{b_i}{n_i}$, де b_i – це евристична оцінка і-го ходу, а n_i означає кількість моделювань для вузла після і-го ходу [21].

Базовий алгоритм Монте-Карло збирає достатньо інформації для знаходження оптимального ходу після багатьох симуляцій. До певного моменту його ходи по суті є випадковими. Ця дослідницька фаза може бути суттєво скорочена у певному класі ігор із використанням швидкої оцінки вартості дій або RAVE (Rapid Action Value Estimation) [20]. Це ігри, у яких перестановки послідовності ходів призводять до однакової позиції. Зазвичай це настільні ігри у яких хід передбачає розташування камінця на дошку; кожен хід мінімально

впливає на інші ходи.

При використанні RAVE у ігровому дереві дочірні вузли деякого вузла *N* зберігають не тільки статистику перемог під час симуляцій, розпочатих з *N*, але й статистику перемог в усіх симуляціях від *N* і далі, у випадку якщо вони містять хід *i* (при умові що хід зіграно між *N* та кінцем симуляції). Таким чином, на вміст деревних вузлів впливають не тільки ходи, що відтворюються негайно в заданій позиції, але й ті ж самі ходи, які відтворюються пізніше. При використанні RAVE етап Вибір обирає вузол, для якого формула [20]:

$$\left(1 - \beta(n_i, \tilde{n}_i)\right)\frac{w_i}{n_i} + \beta(n_i, \tilde{n}_i)\frac{\tilde{w}_i}{\tilde{n}_i} + c\sqrt{\frac{lnt}{n_i}} \tag{1.2}$$

має найвище значення. У цій формулі $\tilde{w}_i$ та $\tilde{n}_i$ означають кількість виграних симуляцій що включають хід i та загальну кількість симуляцій, що включають хід i. Результат виразу $\beta(n_i, \tilde{n}_i)$ має бути близьким до одиниці або до нуля для відносно малого або відносно великого значення n_i та $\tilde{n}_i$ відповідно. Одна з багатьох формул для $\beta(n_i, \tilde{n}_i)$ для збалансованих позицій може мати вигляд [22]:

$$\beta(n_i, \tilde{n}_i) = \frac{\tilde{n}_i}{n_i + \tilde{n}_i + 4b^2 n_i \tilde{n}_i} \tag{1.3}$$

де b є емпірично обраною константою.

Пошук рішень алгоритмом Монте-Карло може виконуватись одночасно декількома потоками. Існує декілька фундаментально різних методів для розпаралелювання його виконання, а саме:

– Вузлова паралелізація
– Коренева паралелізація
– Паралелізація дерева

Вузлова паралелізація передбачає паралельний старт декількох симуляцій з одного вузла ігрового дерева.

Коренева паралелізація передбачає паралельну побудову декількох незалежних ігрових дерев, кожне з яких бере свій початок від спільного для всіх

них вузла.

Паралелізація дерева – це спосіб паралельної побудови одного дерева із захистом даних вузлів від одночасного запису.

2.3 Аналіз методів і алгоритмів пошуку шляху

Найбільш поширеними методами для пошуку шляху, що застосовуються у розробці комп’ютерних ігор є ті, що базуються на наступних алгоритмах:

- Алгоритм Дейкстри;
- Алгоритм A*;
- Алгоритм Лі.

Алгоритм Дейкстри знаходить найкоротші шляхи від однієї з вершин графа до всіх інших. Алгоритм працює тільки для графів без ребер негативного ваги. Кожній вершині з V він зіставляє мітку - мінімальне відоме відстань від цієї вершини до A. Алгоритм працює покроково - на кожному кроці він «відвідує» одну вершину і намагається зменшувати мітки. Робота алгоритму завершується, коли всі вершини відвідані. Мітка самої вершини A покладається рівною 0, мітки інших вершин - нескінченності. Це відображає те, що відстані від a до інших вершин поки невідомі. Всі вершини графа позначаються як невідвідані. Якщо все вершини відвідані, алгоритм завершується. В іншому випадку, з ще не відвіданих вершин вибирається вершина u, що має мінімальну позначку. Розглядяються різні маршрути, в яких u є передостаннім пунктом. Вершини, в які ведуть ребра з u, називаються сусідами цієї вершини. Для кожного сусіда вершини u, крім позначених як відвідані, розглядається нова довжина шляху, що дорівнює сумі значень поточної мітки u і довжини ребра, що з'єднує u з цим сусідом. Якщо отримане значення довжини менше ніж значення мітки сусіда, значення мітки заміняється отриманим значенням довжини. Після розглянення всіх сусідів, вершина u позначається як відвідана і крок алгоритму повторюється. Якщо отримане значення довжини менше ніж значення мітки сусіда, значення мітки

заміняється отриманим значенням довжини. Після розглянення всіх сусідів, вершина u позначається як відвідана і крок алгоритму повторюється [23].

Алгоритм A* покроково переглядає всі шляхи, що ведуть від початкової вершини в кінцеву, поки не знайде мінімальний. Як і всі поінформовані алгоритми пошуку, він переглядає спочатку ті маршрути, які «здаються» ведучими до мети. Алгоритм враховує весь пройдений шлях. Складова $g(x)$- це вартість шляху від початкової вершини. На початку роботи проглядаються вузли, суміжні з початковим; вибирається той з них, який має мінімальне значення $f(x)$, після чого цей вузол розкривається. На кожному етапі алгоритм оперує з безліччю шляхів з початкової точки до всіх ще не розкритих (листових) вершин графа - безліччю можливих рішень, - які розміщуються в черзі за пріоритетом. Пріоритет шляху визначається за значенням функції $f(x) = g(x) + h(x)$. Алгоритм продовжує свою роботу до тих пір, поки значення f(x) цільової вершини не виявиться меншим, ніж будь-яке значення в черзі, або поки все дерево не буде переглянуто. З безлічі рішень вибирається рішення з найменшою вартістю [24].

Алгоритм A* і допустимий, і обходить при цьому мінімальну кількість вершин, завдяки тому, що він працює з «оптимістичною» оцінкою шляху через вершину. Оптимістичною в тому сенсі, що, якщо він піде через цю вершину, у алгоритма «є шанс», що реальна вартість результату буде дорівнювати цій оцінці, але ніяк не менше. Але, оскільки A* є поінформованим алгоритмом, така рівність може бути цілком можливою.

Коли A* завершує пошук, він, згідно з визначенням, знайшов шлях, справжня вартість якого менше, ніж оцінка вартості будь-якого шляху через будь-який відкритий вузол. Але оскільки ці оцінки є оптимістичними, відповідні вузли можна без сумнівів відкинути. Інакше кажучи, A* ніколи не упустить можливості мінімізувати довжину шляху, і тому є допустимим.

Припустимо тепер, що якийсь алгоритм B повернув в якості результату шлях, довжина якого більше оцінки вартості шляху через деяку вершину. На

підставі евристичної інформації, для алгоритму В можна виключити можливість, що цей шлях мав і меншу реальну довжину, ніж результат. Відповідно, поки алгоритм В переглянув менше вершин, ніж А*, він не буде допустимим. Отже, А* проходить найменшу кількість вершин графа серед допустимих алгоритмів, що використовують таку ж точну (або менш точну) евристику.

Алгоритм Лі працює на дискретному робочому полі (ДРП), що представляє собою обмежену замкнутою лінією фігуру, не обов'язково прямокутну, розбиту на прямокутні комірки, в окремому випадку - квадратні. Комірки ДРП розбиваються на підмножини: «прохідні» (вільні), тобто при пошуку шляху їх можна проходити, «непрохідні» (перешкоди), тобто шлях через цю комірку заборонений, стартова комірка (джерело) і фінішна комірка (приймач). Робота алгоритму включає в себе три етапи: ініціалізацію, поширення хвилі і відновлення шляху. Під час ініціалізації будується образ множини комірок оброблюваного поля, кожній комірці приписуються атрибути прохідності або непрохідності, запам'ятовуються стартова і фінішна комірки. Далі, від стартової комірки породжується крок до сусідньої комірки, при цьому перевіряється, прохідна вона, і чи не належить раніше поміченій у шляху комірці. Сусідніми осередками вважаються 4 осередки, тобто по одній з кожної сторони. При виконанні умов прохідності і неналежності її до раніше позначеним у шляху коміркам, в атрибут комірки записується число, що дорівнює кількості кроків від стартового осередку. На першому кроці це буде число 1. Кожна комірка, помічена числом кроків від стартової комірки стає стартовою і з неї породжуються чергові кроки в сусідні комірки. Очевидно, що при такому переборі буде знайдено шлях від початкової комірки до кінцевої, або черговий крок з будь-якої породженої в дорозі комірки буде неможливим. Відновлення найкоротшого шляху відбувається в зворотному напрямку: при виборі комірки від фінішної комірки до стартової на кожному кроці вибирається комірка, що має атрибут відстані від стартової на одиницю менше поточної комірки. Очевидно,

що таким чином знаходиться найкоротший шлях між парою заданих комірок [25]. На рисунку 2.1 зображений приклад роботи алгоритму Лі.

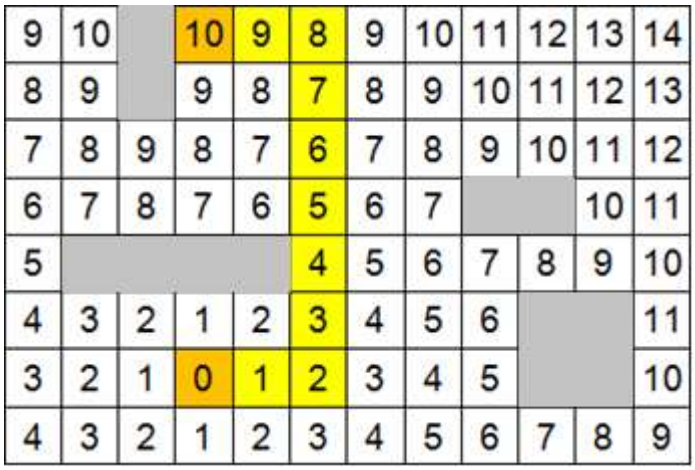

Рисунок 2.1 – Робота алгоритму Лі

У проектованій системі необхідно знаходження шляху між двома відомими точками ігрового світу, а отже алгоритм Дейкстри, що знаходить найкоротший шлях між однією точкою та усіма іншими у графі не є корисним. Алгоритм Лі заснований на алгоритмі пошуку в ширину (BFS), і його складність складає $O(b^d)$, де b- фактор розгалуження і d- глибина цілі. Складність алгоритму A * залежить від евристики. У гіршому випадку, число вершин, досліджуваних алгоритмом, зростає експоненціально в порівнянні з довжиною оптимального шляху. У найгіршому випадку A* деградує до пошуку в ширину (BFS) зі складністю $O(b^d)$. Для розроблюваної системи складність алгоритму є дуже важливою характеристикою алгоритму, адже при динамічному оновленні шляху алгоритм має обраховуватись щонайменше 10 разів на секунду. Отже правильним рішенням є обрання алгоритму A* для реалізації пошуку шляху.

Важливою складовою алгоритму A* є евристична функція, адже вона впливає на складність алгоритму та вірність його роботи. Найбільш розповсюдженими евристичними функціями для алгоритму A* є наступні:

– Манхеттенська відстань – це відстань між двома точками в виміряними вздовж осей під прямим кутом. У площині з точкою p_1 з координатами (x_1, y_1)

та точкою p_2 з координатами (x_2, y_2) манхеттенська відстань дорівнює $| x_1 - x_2 | + | y_1 - y_2 |$ [26]. У більшості випадків ця міра відстані приводить до таких же результатів, як і для відстані Евкліда.

– Евклідова відстань – відстань між двома точками евклідового простору, що обчислюється за теоремою Піфагора [27].

– Відстань Чебишева – відстань між двома точками, що обчислюється за формулою $p(x, x') = max(|x_i - x_i|)$ [28]. Алгоритм її знаходження близький до алгоритму Лі.

Відстань Чебишева не підходить через складність обчислення, продиктованою складністю алгоритму її знаходження. Евклідова відстань є нескладним в обчисленні методом знаходження відстані між двома точками, проте найкращим рішенням буде використати манхеттенську відстань через її заточеність під граф-сітку, що і використовується у проектованій системі для створення карти.

На основі об'єктивних переваг для пошуку шляху у графі був обраний алгоритм A*, що у якості евристичної функції буде використовувати манхеттенську відстань.

Принцип алгоритму полягає у послідовному «розкритті» вершин графу та постійній перевірці значення ціни уже пройденого шляху складеної із значенням евристики та вибору мінімального з можливих таких значень. Значення ціни обраховується за формулою$f(x) = g(x) + h(x)$, де $g(x)$ дорівнює ціні пройденого шляху, а $h(x)$дорівнює значенню евристичної функції. У нашому випадку значення евристичної функції дорівнює манхеттенскій відстані між поточною та цільовою вершинами.

Роботу алгоритму можна розділити на такі кроки:

– Сортування відкритих вершин;

– Видалення першої вершини x (з мінімальним $f(x)$) з відкритих;

– Розширення першої вершини x її сусідами;

– Перевірка чи ціль досягнена;

– Закриття вершини x;
– Обрахування значення $f(x) = g(x) + h(x)$ для сусідніх вершин;
– Відкриття сусідніх вершин.

Приклад роботи алгоритму A* зображений на рисунку 2.2.

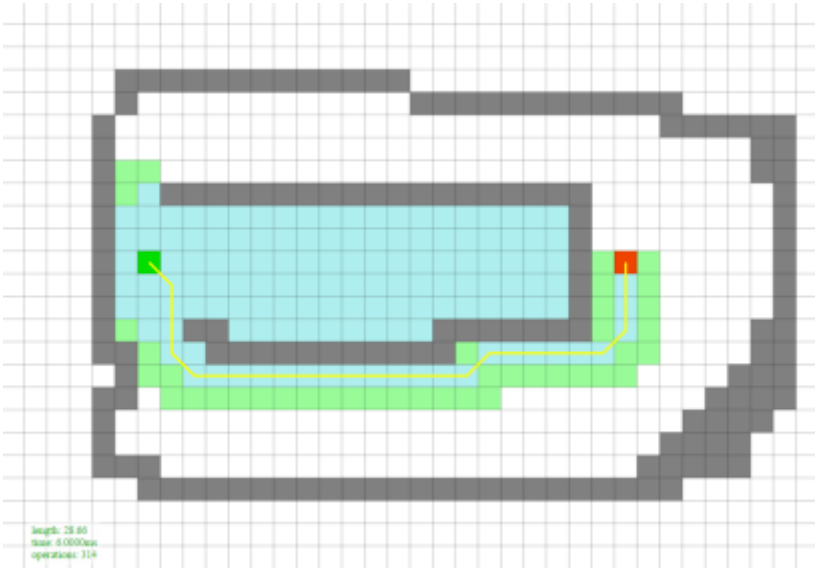

Рисунок 2.2 – Приклад роботи алгоритму A* [24]

2.4 Аналіз методів для імітування зору штучних суперників

На відміну від наявності рекомендацій від спільноти спеціалістів у галузі розробки комп'ютерних ігор стосовно пошуку шляху у комп'ютерних іграх, немає загальних трендів щодо вибору методів імітування зору штучних суперників. У ході дослідження було виявлено два способи реалізації цієї задачі.

Перший метод полягає у створенні радіусу погляду об'єкта-ворога. У разі потрапляння у радіус погляду деякого типу об'єкту відбувається реакція ворога у вигляді ураження об'єкта у зоні. Для перевірки чи є хоча б один об'єкт у зоні необхідно кожен крок рендеру перевіряти координати усіх ігрових об'єктів, що можуть рухатися на предмет співпадання їх із координатами зони. Це досить ресурсоємна робота, особливо при великій кількості об'єктів, що перевіряється. Іншим недоліком цього методу є важкість визначення наявності перешкоди між

об’єктом спостереження та штучним супротивником. Адже супротивник не має «бачити» крізь стіну, тому що за нею є об’єктом спостереження, що потрапив у радіус зони погляду.

Другим методом є метод кидання променів або рейкастинг. Сутність рейкастингу полягає у випусканні променів від деякої точки А до точки Б. Під час проходження шляху промінь зтикається з ігровими об’єктами, що лежать у нього на шляху, та отримує деяку інформацію про них. Це можна використати для вирішення задачі штучного зору суперників. Кожен крок рендеру від об’єкта супротивника посилаються промені в усі сторони і якщо перший об’єкт на шляху променя не є гравцем, промінь зупиняє рух, повідомляючи ворога про те, що гравця не видно. У разі, якщо промінь зіткається із гравцем, штучний супротивник «бачить» гравця, отримуючи певну інформацію від ігрового рушія [29]. Приклад випускання променів у різні сторони ігрового середовища зображений на рисунку 2.4.

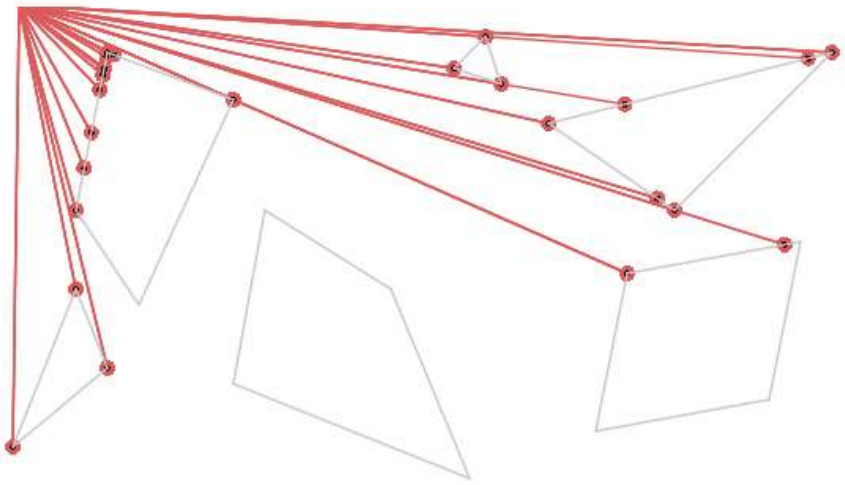

Рисунок 2.4 – Приклад рейкастингу

Рейкастинг не потребує багато обчислювальних ресурсів для своєї роботи, адже він не перевіряє координат усіх об’єктів на карті. Існує можливість

перевіряти лише об’єкти, що знаходяться на шляху променя. Цього достатньо для створення штучного зору супротивника.

2.5 Висновок

В даному розділі магістерської кваліфікаційної роботи описано та проаналізовано методи та алгоритми прийняття рішень штучними суперниками та неігровими персонажами у комп’ютерній грі, серед яких: скінченний автомат, його різновиди та модифікації, а також алгоритм Монте-Карло для пошуку у дереві рішень й можливі варіанти його модифікації. Описано переваги та недоліки поданих алгоритмів та методів.

При аналізі алгоритмів пошуку шляху у ігровому середовищі описано такі алгоритми як: Алгоритм Дейкстри, Алгоритм A*, Алгоритм Лі. Визначено їх переваги та недоліки. Описано та проаналізовано найбільш розповсюджені евристичні функції для роботи алгоритму A*.

Проаналізовано можливі варіанти реалізації штучного зору штучних суперників у ігровому середовищі.

Обрано оптимальні алгоритми, що будуть використанні при проектуванні та програмній реалізації інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі, що дало необхідні дані для етапу практичної реалізації інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі. Для задачі прийняття рішень обрано алгоритм Монте-Карло для пошуку у дереві рішень. Для задачі пошуку шляху обрано алгоритм A*. Для задачі імітації зору штучних суперників обрано метод кидання променів.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОЇ ПОВЕДІНКИ ШТУЧНИХ СУПЕРНИКІВ У КОМП'ЮТЕРНІЙ ГРІ

3.1 Проектування технології інтелектуальної поведінки штучних суперників у комп'ютерній грі

Проектована інформаційна технологія інтелектуальної поведінки штучних суперників у комп'ютерній грі складається із декількох модулів, що можуть бути інтегровані у ігровий рушій незалежно один від одного та виконуватись різними потоками.

3.1.1 Проектування модулю прийняття рішень

IDEF0 – це методологія функціонального моделювання і графічного опису процесів систем, що базується на ієрархічному представленні об'єктів. Вона дозволяє представляти логічні відношення між процесами. Спочатку створюється загальна контекстна діаграма, що описує взаємодію системи із зовнішніми даними, над якою потім проводиться процес функціональної декомпозиції під час якої система розбивається на підсистеми, для їх детального опису. Процес декомпозиції проводиться до необхідного рівня декомпозиції [30].

Основною задачею модулю прийняття рішень є пошук оптимальних ходів у дереві рішень. Вхідними даними для цього модулю є кореневий вузол з якого починається дерево рішень (Nodes). Кореневий вузол має інформацію про можливі наступні ходи у вигляді дочірніх вузлів. Управління процесу пошуку шляху відбувається алгоритмом Монте-Карло для пошуку у дереві рішень (Monte Carlo Tree Search Algorithm). Механізмом є гравець, що визначає кореневий вузол, роблячи перший хід у грі (Player). Вихідними даними є побудоване дерево рішень, що містить у собі вузли із інформацією, що вказує на оптимальність ходів у кожній можливій ігровій ситуації (Search tree).

На рисунку 3.1 наведено контекстну IDEF0 діаграму модуля прийняття рішень штучними супротивниками у комп’ютерній грі.

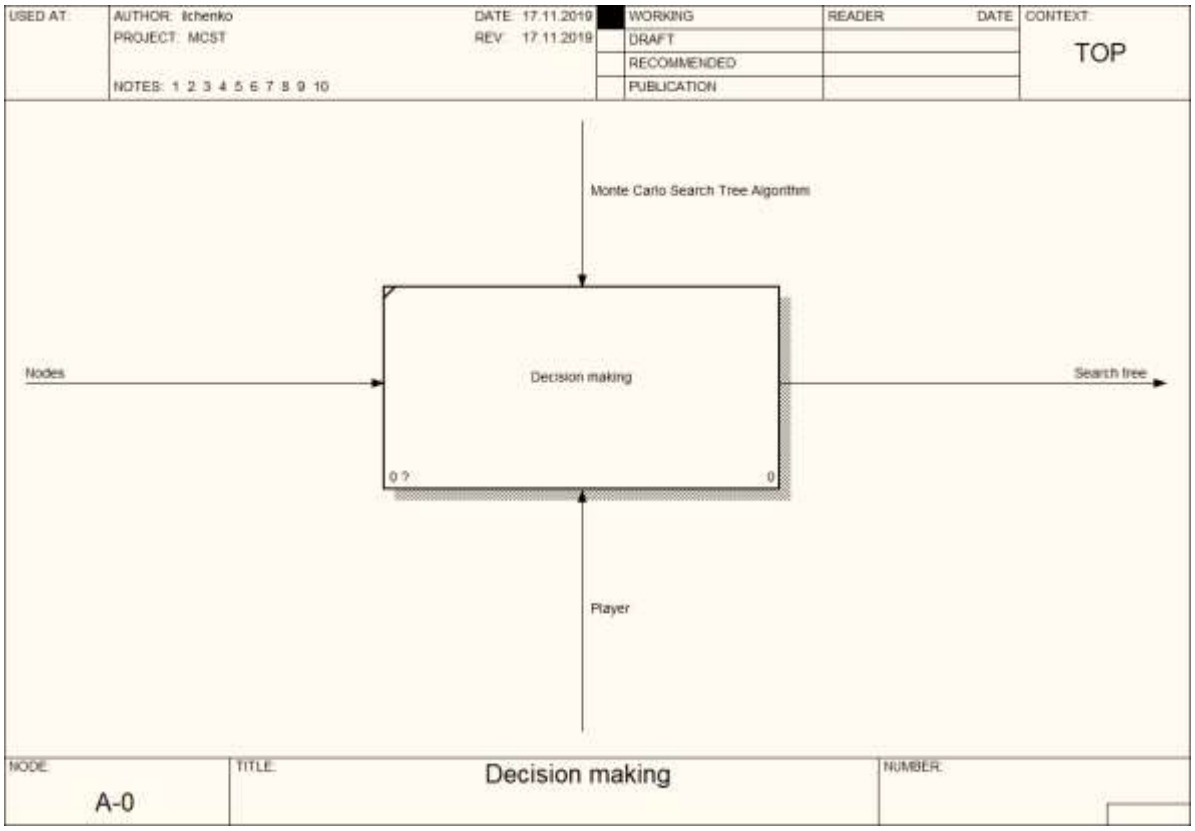

Рисунок 3.1 – Контекстна діаграма модулю прийняття рішень

На рис. 3.2 наведено діаграму декомпозиції модулю прийняття рішень. Модуль прийняття рішень формує дерево рішень, що містить інформацію про оптимальні ходи для кожної ігрової ситуації. Процес формування дерева рішень складається з вибору оптимальних вузлів (Select) доки не буде досягнуто останній вузол гілки, потім відбувається розширення (Expand) дерева новим вузлом, далі відбувається Симуляція (Simulation), під час якої випадковим чином симулюється гра від нового вузла. Останнім етапом є зворотне розповсюдження результату симуляції (Backpropagate results), що оновлює інформацію дерева. Цей ряд дій рекурсивно повторюється поки усі можливі стани гри не будуть занесені у дерево, або іншу встановлену кількість часу, що залежить від об’єму можливих ігрових станів певного ігрового середовища. Алгоритм MCTS модифіковано

вузловим паралельним обчисленням. Вузлова паралелізація передбачає паралельний старт декількох симуляцій з одного вузла ігрового дерева.

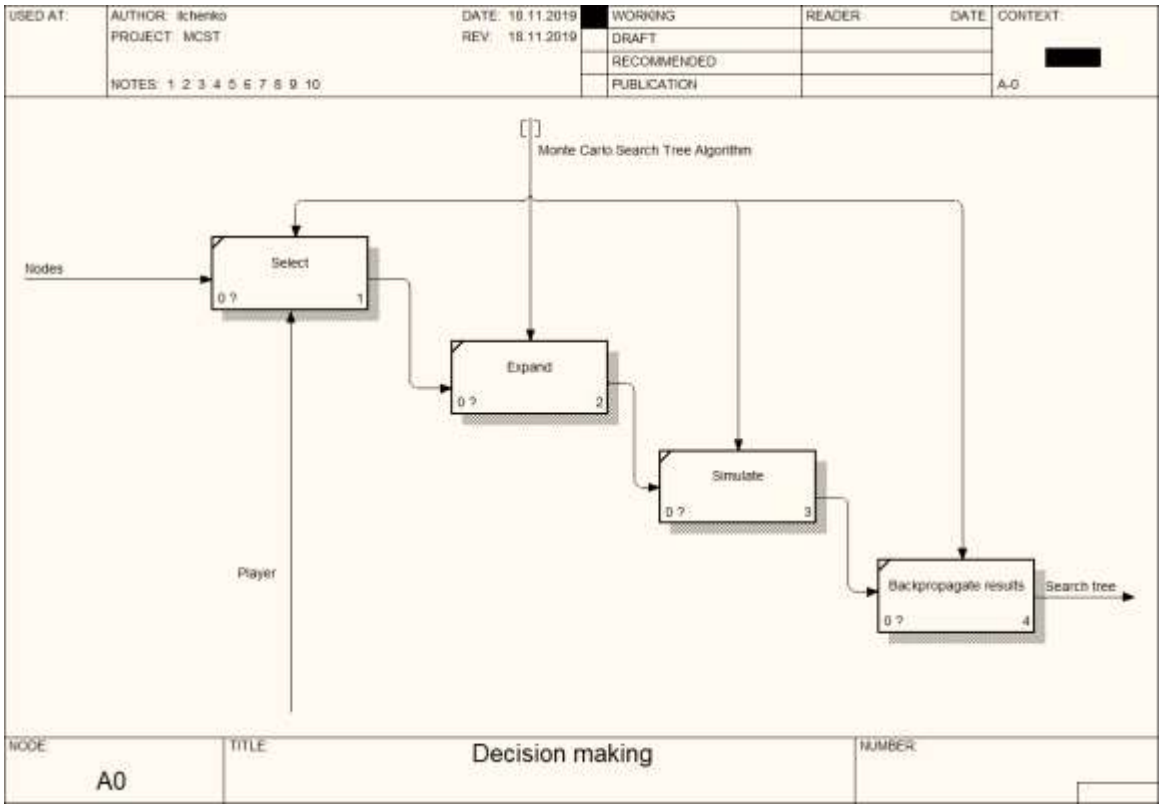

Рисунок 3.2 – Діаграма декомпозиції модулю прийняття рішень

3.1.2 Проектування модулю пошуку шляху

Задачею модулю пошуку шляху є пошук найкоротшого шляху у між двома точками у графі. Вхідними даними є граф ігрового середовища (Graph). Керування процесом виконується алгоритмом пошуку шляху A* (Algorithm A*) та певною евристичною функцією (Heuristic Function), що необхідна для роботи алгоритму A*. Механізмом є ігровий рушій (Game Engine), що визначає дві точки для пошуку шляху між ними та запускає роботу алгоритму пошуку шляху. Вихідними даними є граф найкоротшого шляху (Graph Path), що складається із послідовності вузлів графу ігрового середовища. На рис. 3.3 наведено

контекстну діаграму модулю пошуку шляху. Діаграма декомпозиції модулю пошуку шляху наведено на рис. 3.4.

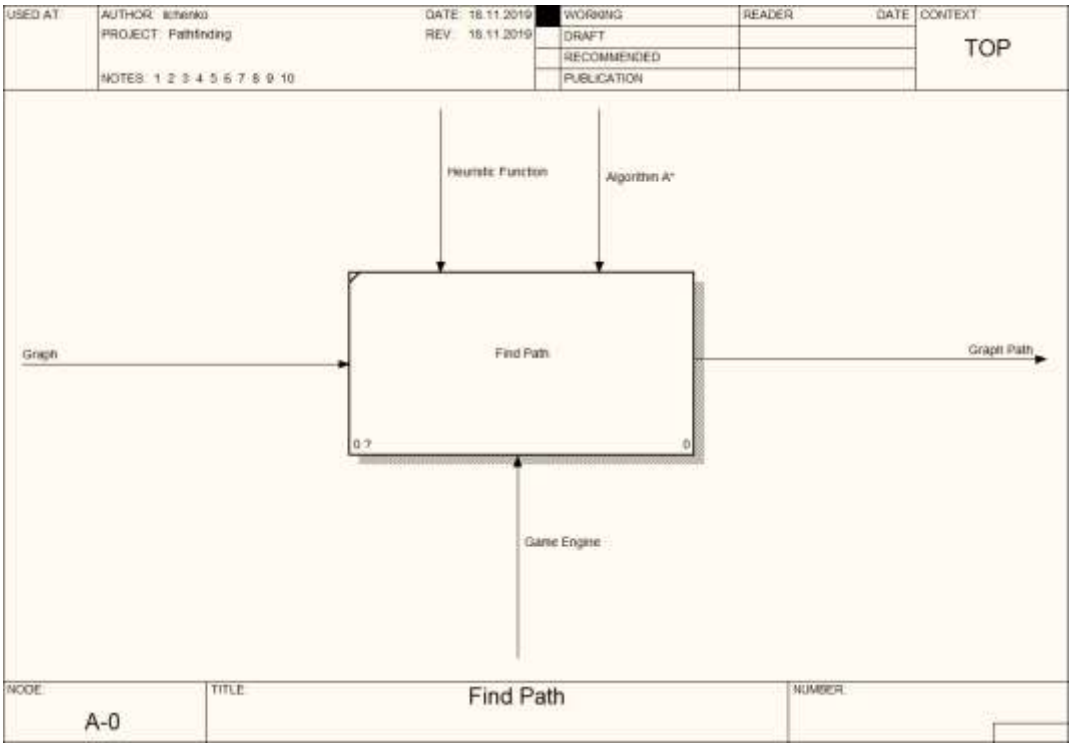

Рисунок 3.3 – Контекстна діаграма модулю пошуку шляху

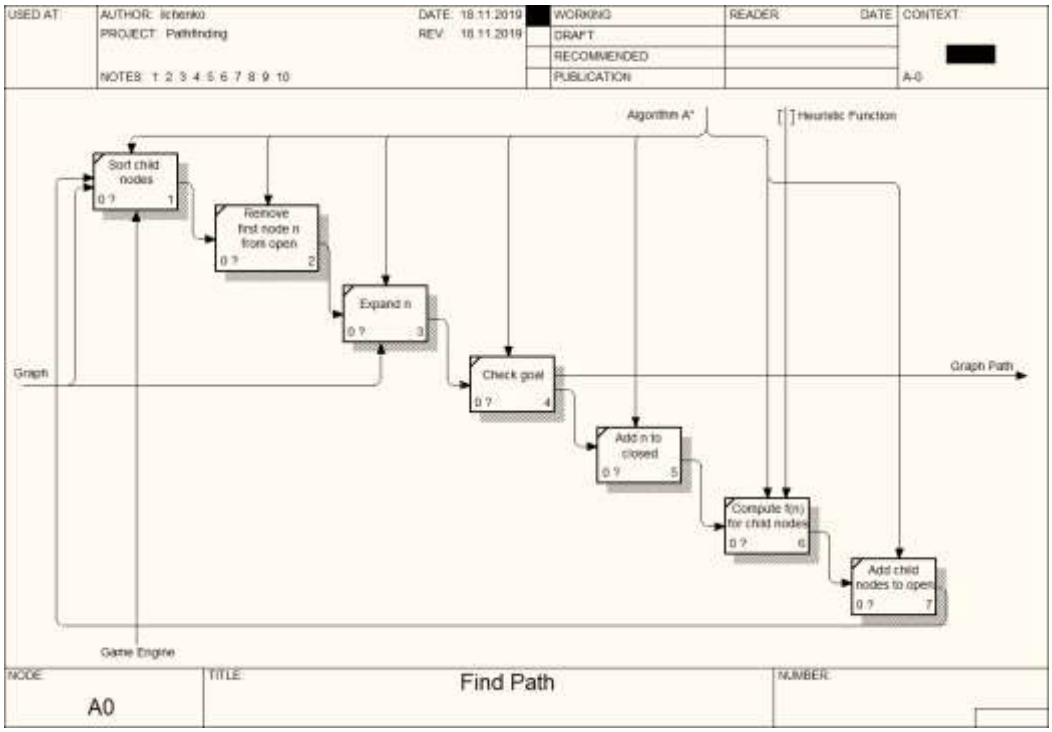

Рисунок 3.4 – Діаграма декомпозиції модулю пошуку шляху

Процес пошуку шляху складається з сортування доступних вузлів за дальністю до цілі (Sort child nodes), видалення найближчого вузла до цілі з відкритих вузлів (Remove first node *n* from open), розширення вузла *n* генеруванням його дочірніх вузлів (Expand *n*), перевірки, чи досягнуто кінцевого вузла (Check goal). Якщо так – формується вихідний граф шляху (Graph Path). Якщо ні – процес продовжується додаванням *n* до закритих вузлів (Add *n* to closed). Далі обчислюється ціна шляху через кожен дочірній вузол (Compute f(n) for child nodes), й змінюється статус усіх дочірніх вузлів на відкритий (Add child nodes to open). Усі кроки рекурсивно повторюються до того моменту, доки найкоротший шлях не буде знайдено, якщо він існує.

3.1.3 Проектування модулю імітування зору штучних суперників

Задачею модуля імітування зору штучних суперників є перевірка чи певний об’єкт знаходиться у полі зору штучного супротивника. Для виконання цієї задачі модуль використовує модель кидання променів. Вхідними даними є координати штучного супротивника та об’єкту спостереження. Управлінням процесу керує алгоритм кидання променів. Механізмом є ігровий рушій, що ініціює процес. Вихідними даними є тип першого об’єкту, на який натрапив промінь. На рис. 3.5 наведено контекстну діаграму модулю імітування зору штучних суперників. Діаграма декомпозиції модулю імітування зору штучних суперників наведено на рис. 3.6. Процес роботи модулю складається із кидання променя (Cast Ray) для кожного об’єкту, що спостерігається, та визначення типу об’єкту першої перешкоди на шляху променя (Determine First Obstacle Type). Тип об’єкту повертається до ігрового рушія, і у випадку збігу типу об’єкту із тим, що спостерігається, виконується певна функція, визначена ігровим рушієм.

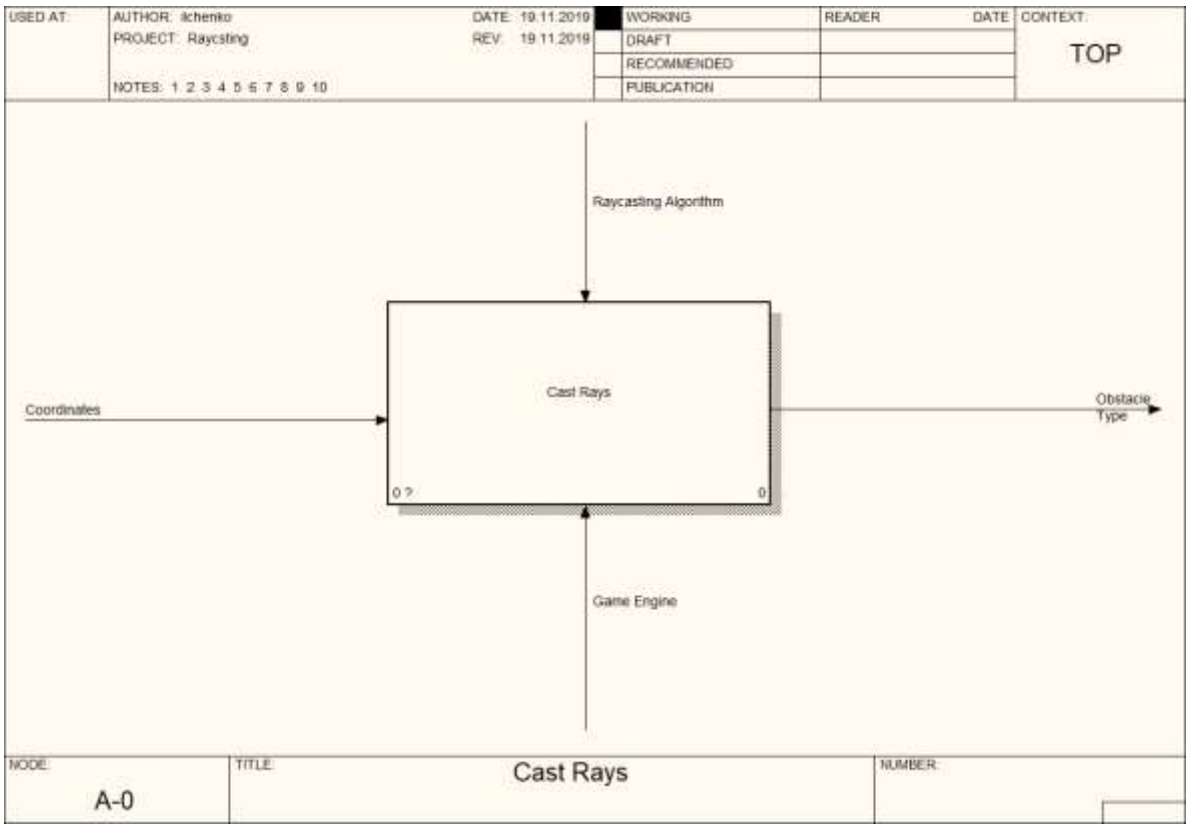

Рисунок 3.5 – Контекстна діаграма модулю імітування зору

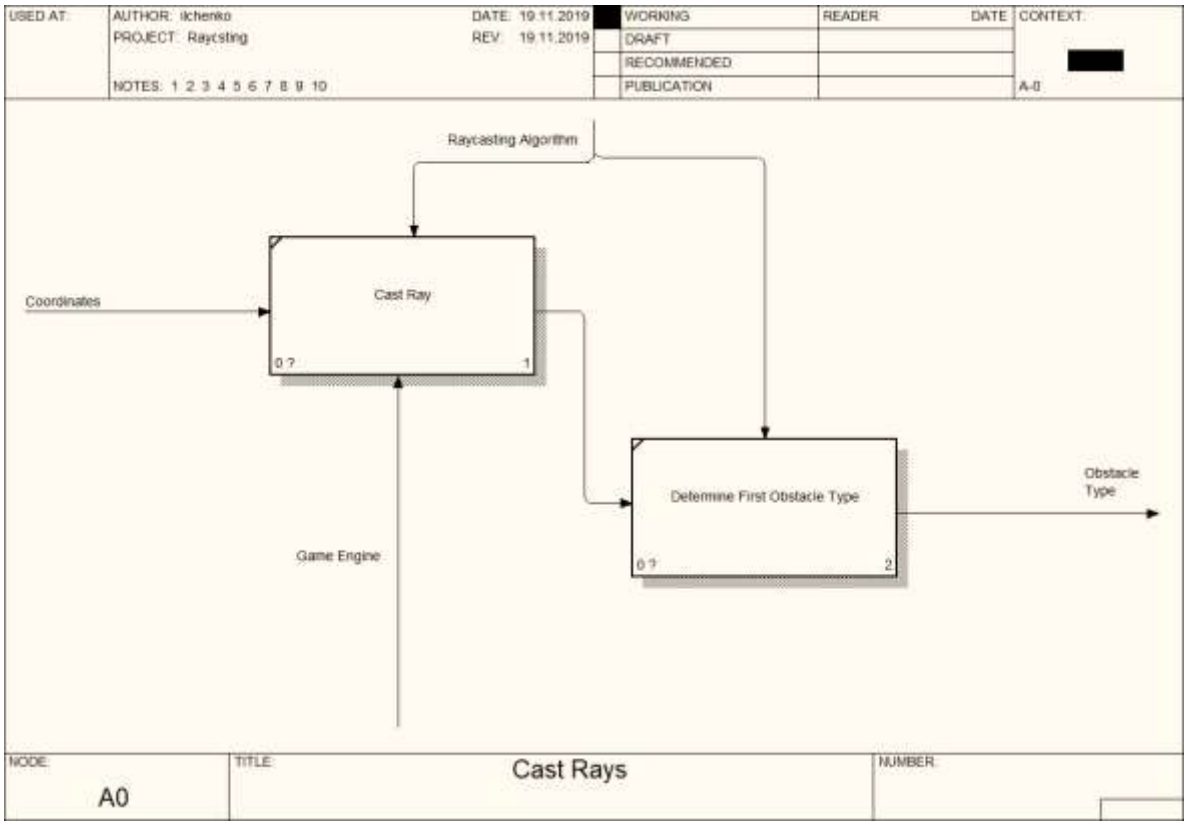

Рисунок 3.6 – Діаграма декомпозиції модулю імітування зору

3.2 Структурна організація технології інтелектуальної поведінки штучних суперників у комп’ютерній грі

Відповідно до поставленої мети було розроблено структурну схему технології інтелектуальної поведінки штучних суперників у комп’ютерній грі, що наведено на рис. 3.7.

Інформаційна технологія інтелектуальної поведінки складається із трьох основних модулів: модуль прийняття рішень, модуль пошуку шляху та модуль імітації зору штучних суперників. Відповідно до цього розроблено діаграми класів, що демонструють загальну структуру ієрархії класів кожного модуля, їх кооперацій, атрибутів, полів, методів, інтерфейсів та взаємозв’язків між ними [31].

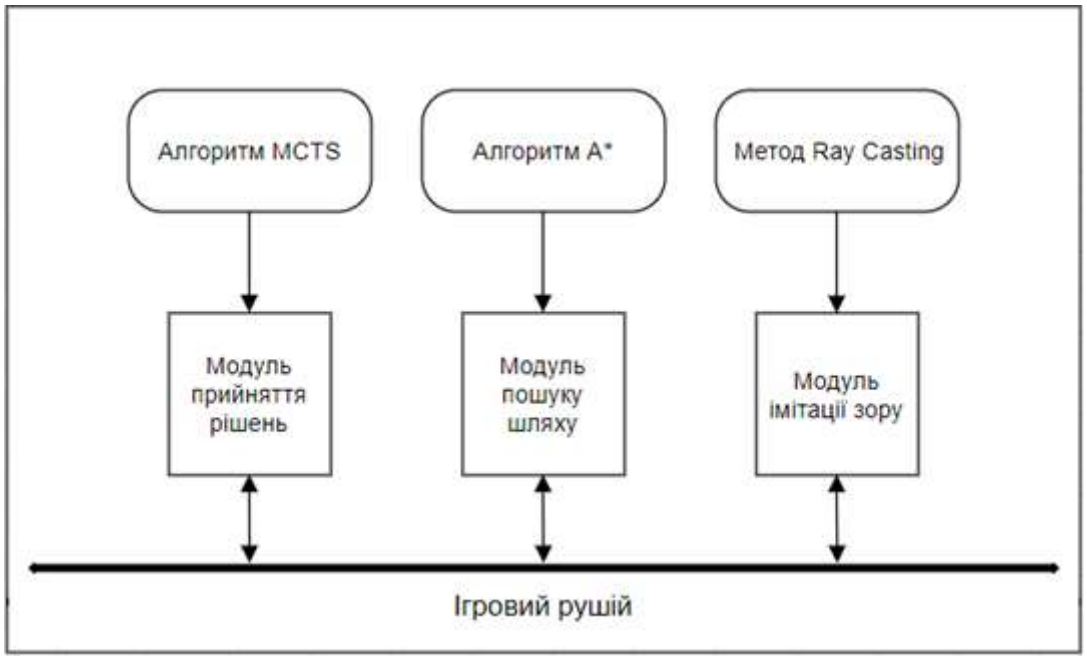

Рисунок 3.7 – Структурна організація технології інтелектуальної поведінки штучних суперників

На рис. 3.8 наведено діаграму класів модулю прийняття рішень, що складається з чотирьох класів: MCSTAlgorithm, Node, MinimalNode, Tree.

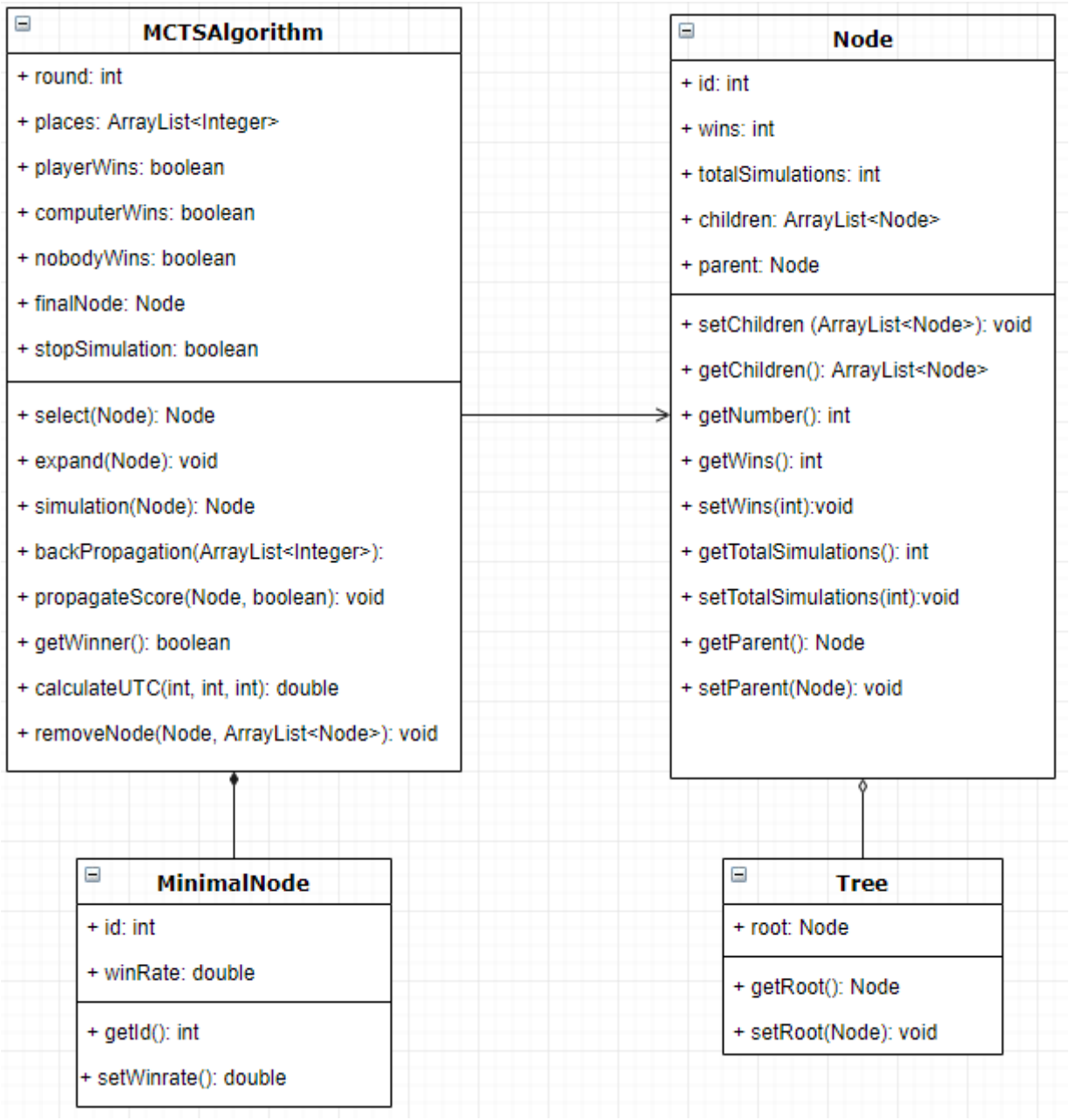

Рисунок 3.8 – Діаграма класів модулю прийняття рішень

MCSTAlgorithm – класс, що реалізує у собі роботу алгоритму Монте-Карло для пошуку у дереві. Перелік та опис членів класу MCSTAlgorithm:

Поля:

– round – лічильник ходів у кожній симуляції;

– places – масив зіграних ходів;

– playerWins – змінна виграшу гравця;

– computerWins – змінна виграшу штучних суперників;

– nobodyWins – змінна нічиї;

– finalNode – останній вузол у симуляції;

– stopSimulation – індикатор необхідності продовження симуляції.

Методи:

– select () – виконує етап вибору алгоритму Монте-Карло;

– expand() – виконує етап розширення алгоритму Монте-Карло;

– simulation() – виконує етап симуляції алгоритму Монте-Карло;

– backpropagation() – виконує етап зворотного розповсюдження алгоритму Монте-Карло;

– propagateScore() – оновлює дані кожного вузла після симуляції;

– getWinner() – визначає переможця у симуляції;

– calculateUTC() – обчислює значення UTC;

– removeNode() – видаляє вузол з масиву вузлів.

Клас Node реалізує вузол ігрового дерева. Він містить такі поля та методи:

Поля:

– id – змінна, що характеризує вузол як хід відносно ігрового середовища;

– wins – кількість виграшів даного вузла;

– totalSimulations – кількість симуляцій даного вузла;

– children – масив дочірніх вузлів даного вузла;

– parent – батьківський вузол даного вузла;

Методи:

– setChildren() – встановити масив дочірніх вузлів даного вузла;

– getChildren() – отримати масив дочірніх вузлів даного вузла;

– getId() – отримати id даного вузла;

– getWins() – отримати кількість перемог даного вузла;

– setWins() – встановити кількість перемог даного вузла;

– getTotalSimulations() – отримати кількість симуляцій даного вузла;

– setTotalSimulations() – встановити кількість симуляцій даного вузла;

– getParent() – отримати батьківський вузол даного вузла;

– setParent() – встановити батьківський вузол даного вузла.

Клас Minimal Node є внутрішнім класом, реалізує мінімальний інтерфейс звичайного вузла і використовується лише для етапу вибору алгоритму Монте-Карло. Він містить такі поля та методи:

Поля:

– id – змінна, що характеризує вузол як хід відносно ігрового середовища;

– winRate – рейтинг вузла, що використовується для етапу вибору алгоритму Монте-Карло для пошуку у дереві;

Методи:

– getId() – отримати id вузла;

– setWinrate() – встановити рейтинг вузла;

Клас Tree є класом-обгорткою, кореневого вузла, що містить усе ігрове дерево. Він містить такі поля та методи:

Поля:

– root – кореневий вузол дерева;

Методи:

– getRoot() – отримати кореневий вузол дерева;

– setRoot() – встановити кореневий вузол дерева.

На рис. 3.9 наведено діаграму класів модулю пошуку шляху. Він складається з класів Pathfinder, HeuristicImp, GraphImp, Node, ConnectionImp, GraphPathImp, GraphGenerator.

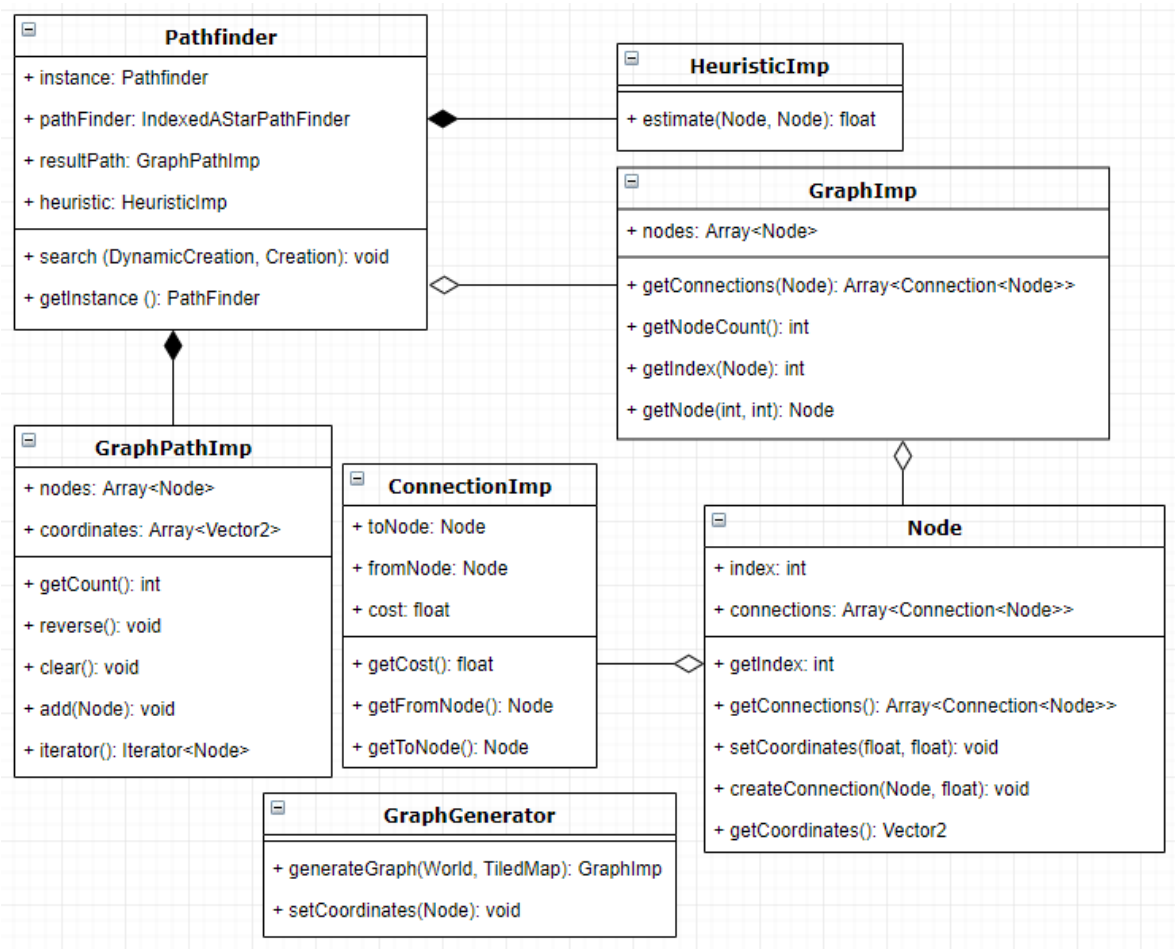

Рисунок 3.9 – Діаграма класів модулю пошуку шляху

Клас Pathfinder є основним класом, що об’єднує інші класи модулю. Він містить такі поля та методи:

Поля:

– instance – єдиний об’єкт класу Pathfinder;

– pathFinder – об’єкт, що реалізує алгоритм пошуку шляху A*;

– resultPath – результат роботи алгоритму A* – найкоротший шлях;

– heuristic – евристична функція для алгоритму A*.

Методи:

– search() – метод, що стартує пошук шляху;

– getInstance() – метод, що повертає єдиний екземпляр Pathfinder.

Клас HeuristicImp має один метод estimate(), що обчислює результат евристичної функції для алгоритму A*.

Клас GraphImp представляє ігрове середовище у вигляді, з яким може працювати алгоритм A*. Він містить такі поля та методи:

Поля:

– nodes – вузли ігрового графу.

Методи:

– getConnections() – отримати сусідні вузли деякого вузла;

– getNodeCount() – отримати кількість вузлів графу;

– getIndex() – отримати індекс деякого вузла графу;

– getNode() – отримати вузол за координатами.

Клас Node описує вузол ігрового графу. Він містить такі поля та методи:

Поля:

– index – індекс вузла у графі;

– connections – масив сусідніх вузлів даного вузла.

Методи:

– getIndex – отримати індекс вузла;

– getConnections – отримати масив сусідніх вузлів даного вузла;

– setCoordinates – встановити координати;

– createConnection – додати сусідній вузол;

– getCoordinates – отримати координати вузла.

Клас ConnectionImp описує зв’язки між сусідніми вузлами графу та ціну проходження між ними. Він містить такі поля та методи:

Поля:

– toNode – кінцевий вузол зв’язку;

– fromNode – початковий вузол зв’язку;

– cost – ціна проходження шляху між вузлами.

Методи:

– getCost() – отримати ціну проходження шляху між вузлами;

– getFromNode() – отримати початковий вузол зв’язку;

– getToNode() – отримати кінцевий вузол зв’язку.

Клас GraphPathImp описує послідовність вузлів у ігровому графі, що являє собою найкоротший шлях між двома вузлами графу. Він містить такі поля та методи:

Поля:

– nodes – масив вузлів найкоротшого шляху;

– coordinates – масив координат вузлів найкоротшого шляху.

Методи:

– getCount() – отримати кількість вузів шляху;

– reverse() – змінити порядок вузлів шляху у масиві;

– clear() – очистити масив вузлів;

– add() – додати вузол до шляху;

– iterator() – отримати об’єкт, що надає функціонал для роботи з масивом вузлів найкоротшого шляху.

Клас GraphGenerator створює ігровий граф з карти ігрового середовища, що поділяється на тайли. Він має два методи:

– generateGraph() – генерує граф із тайлів ігрового середовища;

– setCoordinates() – встановлює координати вузлам графу.

На рис. 3.10 наведено діаграму класів модулю імітації зору штучних суперників, що складається із класів RayCasting, RayCastCallBack, World.

Клас RayCasting містить метод rayCast(), що ініціює кидання променів від координат суперника до координат гравця у ігровому середовищі. Поля класу:

– rayCastCallBack – екземпляр класу RayCastCallBack;

– raycast – змінна, що вказує чи є прямий зоровий контакт;

– world – екземпляр класу World.

Клас RayCastCallBack має один метод reportRayFixture(), що встановлює поведінку модулю для зіткнень проміня з різними об’єктами.

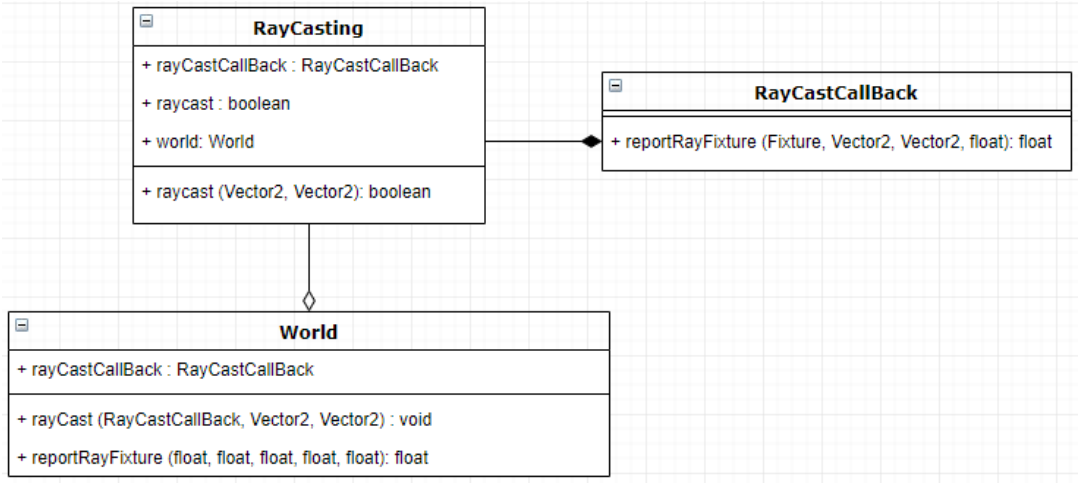

Рисунок 3.10 – Діаграма класів модулю імітації зору

Клас World містить поле rayCastCallBack, що відповідає необхідний екземпляр класу RayCastCallBack для поточного кидання променів, так як поведінка RayCastCallBack може бути різною в залежності від об’єктів процесу.

Клас World також містить метод rayCast(), що запускає процес кидання променів, та метод reportRayFixture(), що повертає тип об’єкту з яким зіткнувся промінь.

3.3 Програмна реалізація інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі

3.3.1 Обґрунтування вибору мови програмування

Java – це об'єктно-орієнтована мова програмування. Java є сильно типізованою мовою, що працює поверх JVM. Програми на Java компілюються в байт-код, який потім інтерпретується для кожної платформи індивідуально та виконується JVM. Перевага такого способу полягає у повній незалежності байт-коду від операційної системи та апаратної складової, що дозволяє виконувати програми на Java на будь-якому пристрої, для якого існує відповідна віртуальна машина [32]. Серед недоліків концепції віртуальної машини є зниження продуктивності, що частково вирішується використанням платформенно-

орієнтованого коду в стандартних бібліотеках, а також використання технології трансляції байт-коду в машинний код безпосередньо під час роботи програми з можливістю збереження версій класу в машинному коді.

Синтаксис Java має багато спільного із синтаксисом C++ та C. Зокрема, у ній за основу взято об'єктну модель C++, що була модифікована.

Основні можливості мови програмування Java:

– Автоматичне керування пам’яті;
– Розширені можливості обробки виключень;
– Широкий набір засобів фільтрації вводу-виводу;
– Великий набір стандартних колекцій;
– Вбудовані у мову засоби створення багатопоточних додатків;
– Підтримка узагальнень;
– Підтримка лямбд, замикань [32].

Автоматичне керування пам’яті – це процес, що періодично автоматично вивільнює пам’ять, видаляючи об’єкти, що не будуть використовуватися програмою.

Java є однією із найпопулярніших мов програмування. Для Java існує велика кількість бібліотек та готових рішень стандартних задач, багато з яких знаходиться у відкритому доступі, що відкриває безмежні можливості програмістам, адже використання готових та оптимізованих спільнотою рішень економить час та покращує якість коду.

В таблиці 3.1 наведено порівняння мов програмування Java і C#. Враховуючи всі переваги і недоліки, була обрана мова Java, бо вона дозволяє розроблювати додатки під будь-яку платформу і безкоштовна для використання, а також має багато зручних відкритих бібліотек для створення ігор та ігрового штучного інтелекту.

Таблиця 3.1 – Порівняння мов програмування Java і C#

Ознака порівняння	Java	C#
Підтримка ООП	Так	Так
Кроссплатформеність	Так	Ні, тільки з додатковим ПЗ
Багато open-source бібліотек	Так	Ні
Сумісність різних версій	Так	Ні
Підтримка функціонального програмування	Так	Ні
Збір сміття	Так	Так

3.3.2 Обґрунтування вибору і аналіз фреймворку libGDX

libGDX – це фреймворк із відкритим вихідним кодом для написання комп'ютерних ігор, що забезпечує уніфікований інтерфейс прикладного програмування, що працює на усіх платформах, де можливо розвернути JVM [33]. До його складу входять декілька потужних інструментів. Для розроблюваної системи використовується ядро libGDX, scene2d, box2d та gdxAI. Ядро фреймворку є зв'язуючим елементом усіх інших компонентів у єдину систему. Scene2d слугує для графічного відображення ігрового середовища. Вона надає інструментарій для створення ігрових сцен, графічного інтерфейсу користувача, та HUD'у. HUD (head-up display) – це метод відображення ігрової інформації так, що вона завжди знаходиться перед очима гравця. Прикладами можуть слугувати показники кількості патронів чи здоров'я гравця, або поточний рахунок гри [33].

Box2d – це вільна бібліотека з відкритим вихідним кодом, що містить у собі багато функцій, що емулюють фізику реального світу. Вона надає можливість

наділяти ігрові об’єкти фізичними властивостями, такими як вага, сила тертя поверхні, пружність та інші.

gdxAI – це бібліотека, що містить у собі реалізації багатьох алгоритмів штучного інтелекту, що широко використовуються у створенні комп’ютерних ігор, зокрема алгоритм пошуку шляху A* та алгоритми керованого руху.

Бібліотеки libGDX вирішують багато задач розробки ігор. Розвинена спільнота фреймворку, наявність книг та інших матеріалів з особливостей розробки ігрових середовищ за допомогою libGDX роблять його потужним інструментом.

3.3.3 Розробка алгоритму інтелектуальної поведінки штучних суперників у комп’ютерній грі

На рис. 3.11 наведено загальний алгоритм роботи інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі. Алгоритм складається з наступних кроків:

1 – на даному кроці алгоритму відбувається ініціалізація усіх модулів.

2 – технологія отримує запит від ігрового рушія на початок роботи одного із модулів із деякими початковими параметрами;

3– початок роботи модулю прийняття рішень, що залучає алгоритм Монте-Карло для пошуку у дереві рішень;

4 – початок роботи модулю шляху, що отримує ігрове середовище, у якому шукає шлях за допомогою алгоритму A*;

5 – початок роботи модулю імітації зору штучних суперників, що використовує модель кидання променів;

6 – перевірка чи ігровий рушій активний та можливо буде надсилати нові команди. Якщо так – відбувається перехід на крок 2, якщо ні – технологія завершує роботу.

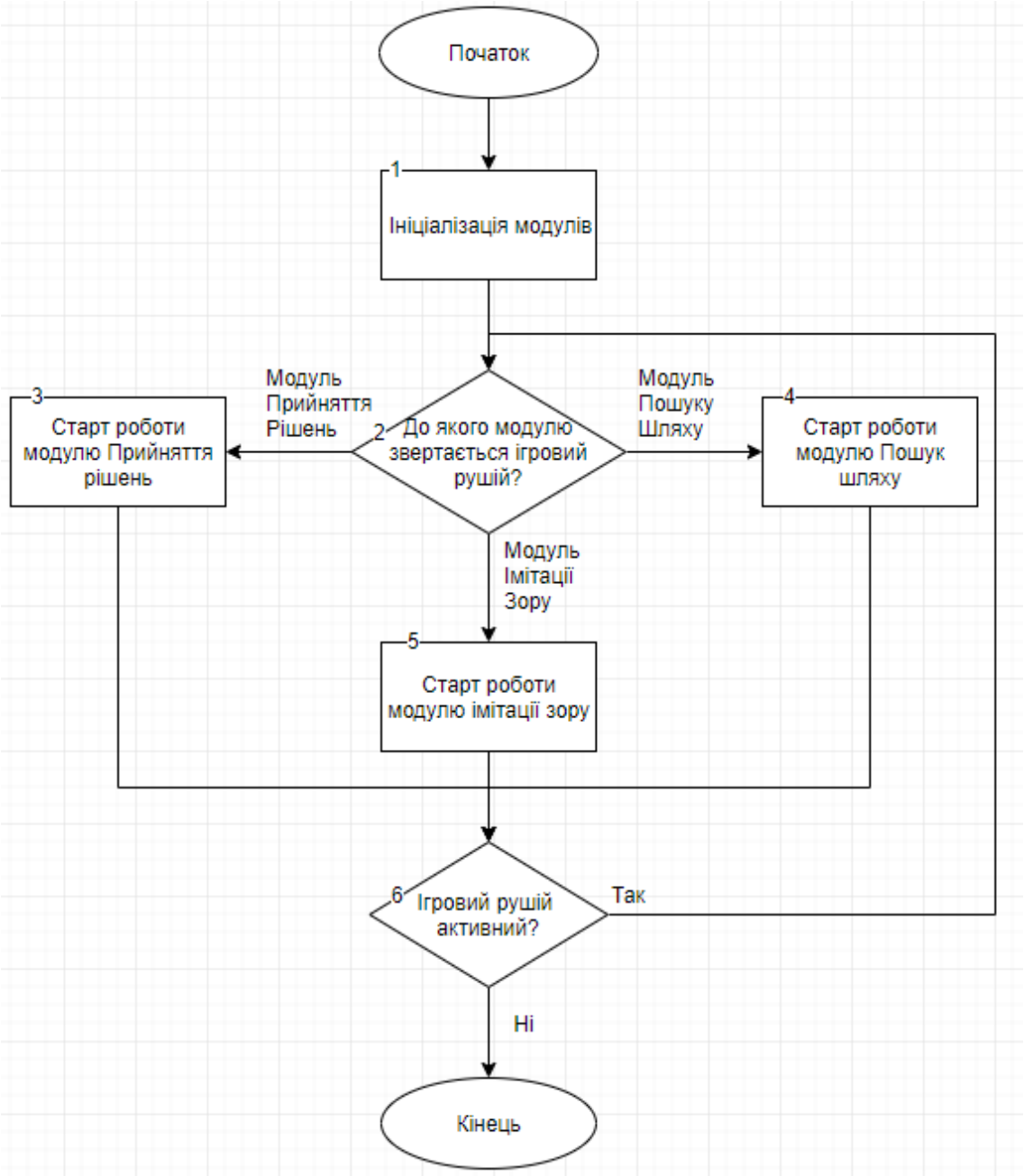

Рисунок 3.11 – Схема роботи алгоритму технології

3.3.4 Тестування інформаційної технології та аналіз результатів

Тестування розробленої програмної реалізації інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі проводилось на двовимірному ігровому середовищі розмірністю 10x10 тайлів.

На рис. 3.12 наведено ігрову ситуацію у покроковій грі, що є вдалою для тестування модулю прийняття рішень. Поточний хід за штучним суперником (червоний круг). Об’єкт гравця виглядає як лицар. Поточний рівень здоров’я гравця дорівнює 2/6, ворога – 2/6. У суперника в наявності здатність стрибка, що дозволяє ходити через одне поле. Гравець використав аналогічну здатність раніше, тому вона не доступна для гравця. Суперник не має куль, а у гравця є одна куля для стрільби. За один хід можна перемістити персонажа у ігровому середовищі та атакувати врукопашну, або тільки вистрілити кулею, без переміщення. У штучного суперника є вибір:

– використати стрибок на північ під прикриття стіни та підібрати аптечку, що відновить його рівень здоров’я;

– використати здатність стрибка, перестрибнути лаву, втративши 1/6 здоров’я, та атакувати гравця врукопашну, що приведе до перемоги супротивника;

– зробити один крок у будь-яку сторону, після чого гравець ймовірно вистрелить кулею та переможе.

Під керуванням алгоритму Монте-Карло для пошуку у дереві рішень штучний супротивник обрав оптимальний у даній ігровій ситуації варіант – другий серед наведених вище.

На рис. 3.12 наведено ігрове середовище після того як штучний супротивник зробив оптимальний хід. Штучний супротивник може продовжити гру рухаючись до аптечки використовуючи модуль пошуку шляху.

На рис. 3.13 зображено результат роботи модулю пошуку шляху – супротивник знайшов найкоротший шлях до аптечки.

Рисунок 3.12 – Ігрова ситуація

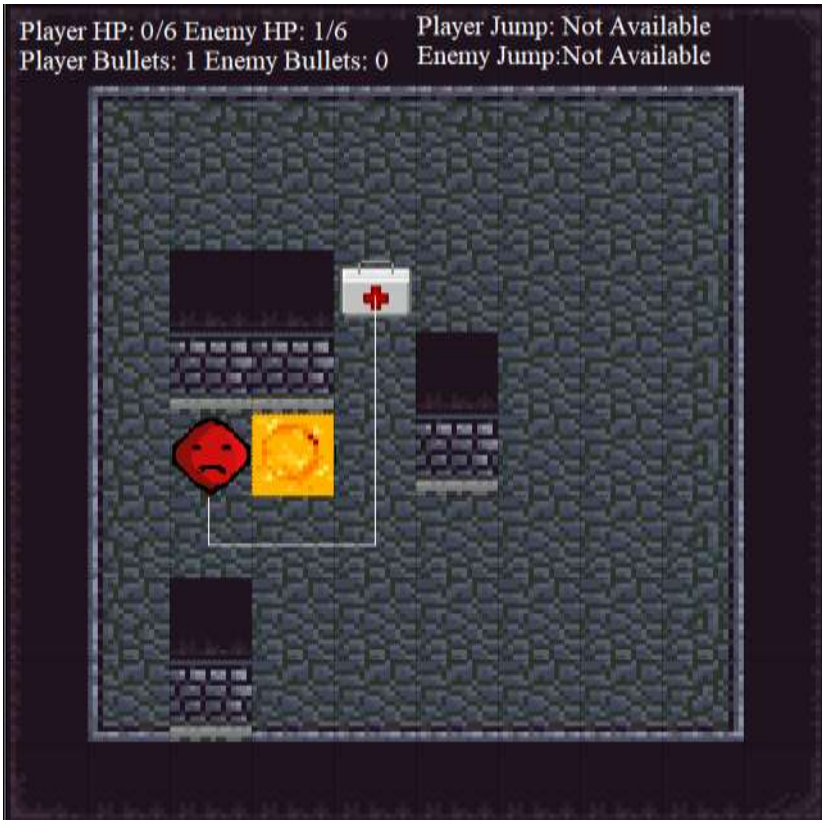

Рисунок 3.13 – Оновлена ігрова ситуація

Під час тестування модулю імітації зору штучних суперників модуль відпрацював правильно. Кожен хід модуль посилав промінь від точки координат супротивника до точки координат гравця й у разі якщо перша перешкода на його шляху була гравцем, ворог «бачив» гравця, що давало йому можливість стріляти кулею у гравця.

При тестуванні модулю прийняття рішень при встановленні часу створення дерева можливих рішень у 10 секунд та вузловому розпаралелюванні обчислень алгоритму MCTS загальна кількість симуляцій, що оновлювала дані кореневого вузла перевищувала кількість таких же симуляцій при однопоточному виконанні на 21%, що демонструє ефективність модифікації за розпаралелювання обчислень алгоритму MCTS.

Для оцінювання рівня інтелектуальності розробленої інформаційної технології було залучено п’ять експертів, що оцінили роботу технології та систем-аналогів за 10-бальною шкалою. Результат оцінювання наведено у таблиці 3.2.

Таблиця 3.2 – Експертна оцінка розробленої ІТ та аналогів

Критерій	Розроблена ІТ	ШІ Enter the Gungeon	ШІ Dwarf Fortress
Прийняття рішень	7, 8, 9, 7, 8	3, 4, 3, 4, 2	6, 6, 7, 5, 6
Пошук шляху	9, 8, 7, 7, 8	8, 9, 8, 7, 7	7, 8, 9, 8, 8
Штучний зір	8, 7, 7, 6, 7	8, 9, 8, 7, 7	8, 8, 7, 8, 8
Швидкодія	7, 8, 7, 7, 8	9, 10, 9, 9, 9	5, 4, 4, 5, 5

Оцінка критерію «Прийняття рішень» для розробленої ІТ складає 78%, для ШІ Enter the Gungeon – 32%, для ШІ Dwarf Fortress – 60%.

Оцінка критерію «Пошук шляху» для розробленої ІТ складає 78%, для ШІ Enter the Gungeon – 78%, для ШІ Dwarf Fortress – 80%.

Оцінка критерію «Штучний зір» для розробленої ІТ складає 70%, для ШІ Enter the Gungeon – 78%, для ШІ Dwarf Fortress – 78%.

Оцінка критерію «Швидкодія» для розробленої ІТ складає 74%, для ШІ Enter the Gungeon – 92%, для ШІ Dwarf Fortress – 46%.

Отже загальна оцінка розробленої ІТ дорівнює 75%, ШІ Enter the Gungeon – 70%, ШІ Dwarf Fortress – 66%. За результатами експертної оцінки можна зробити висновок, що розроблена інформаційна технологія працює на 5% ефективніше ніж наступна за рейтингом система-аналог.

3.4 Висновок

Під час практичної реалізації інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі проведено проектування модулів інформаційної технології, зокрема модуль прийняття рішень, модуль пошуку шляху та модуль імітації зору штучних суперників. Розроблено IDEF0 діаграми, що описують функціональність кожного модуля.

Описано структуру інформаційної технології та кожного модуля інформаційної технології, створено діаграми класів та подано опис їх полів та методів. Визначено схему загального алгоритму інформаційної технології.

Обґрунтовано вибір мови програмування Java для програмної реалізації інформаційної технології. Java є універсальною мовою програмування, програми написані на Java можуть бути запущені з-під будь-якої операційної системи.

В якості інструменту для створення ігрових середовищ для тестування інформаційної технології обрано фреймворк libGDX, що надає потужні інструменти для створення ігор.

Тестування програмної реалізації інформаційної технології інтелектуальної поведінки штучних суперників у комп'ютерній грі проведено на декількох ігрових ситуаціях, у яких розроблена ІТ діяла оптимально та відповідно поставленим перед нею задачам. Проведено експертну оцінку розробленої ІТ, що показала 5% приріст ефективності розробленої ІТ відносно систем-аналогів.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

На сьогоднішній день комп’ютерні ігри є достатньо популярним видом дозвілля. Індустрія розробки комп’ютерних ігор є більш масштабною ніж кіноіндустрія. Важливою складовою більшості комп’ютерних ігор є штучний інтелект для керування внутрішньоігровими об’єктами. Рівень інтелектуальності поведінки штучних суперників та неігрових персонажів у комп’ютерних іграх сильно впливає на ігровий процес та рівень занурення гравця у ігрове середовище. Розвиток галузі ігрового штучного інтелекту є перспективним, глобальною ціллю галузі є створення штучного інтелекту, що буде грати з реальними людьми на рівні людини, та цим самим створювати унікальний ігровий досвід для гравця. Саме тому при розробці комп’ютерних ігор системам штучного інтелекту приділяється багато уваги.

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності. Магістерська кваліфікаційна робота за темою “Інформаційна технологія інтелектуальної поведінки штучних суперників у комп’ютерній грі” передбачає розробку технології, що надає розробникам комп’ютерних ігор доступ до загальної реалізації ряду методів та алгоритмів для створення ігрового штучного інтелекту що слугує для керування штучних суперників у комп’ютерній грі.

Проведемо оцінювання комерційного потенціалу даної розробки. Для проведення технологічного аудиту було залучено 3-х незалежних експертів: Яровий Андрій Анатолійович – керівник магістерської кваліфікаційної роботи, доктор технічних наук, професор кафедри комп’ютерних наук; Арсенюк Ігор Ростиславович – кандидат технічних наук, доцент кафедри комп’ютерних наук, Сілагін Олексій Віталійович – кандидат технічних наук, доцент кафедри комп’ютерних наук.

В таблиці 4.1 наведено критерії оцінювання комерційного потенціалу розробки та їх оцінки в балах.

Таблиця 4.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер	0	1	2	3	4
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкуренті в немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або навчати наявних	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер	0	1	2	3	4
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер	0	1	2	3	4
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки потрібно звести в таблицю за зразком таблиці 4.2.

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Яровий	2 – Арсенюк	3 – Сілагін
	Бали, виставлені експертами:		
1	3	3	3
2	2	2	2
3	4	3	3
4	3	3	2
5	3	4	3
6	4	3	3
7	2	3	3
8	4	4	3
9	4	3	3
10	4	4	4
11	4	4	4
12	4	4	4
Сума балів	$СБ_1$=41	$СБ_2$=40	$СБ_1$=37
Середньоарифметична сума балів СБ	$\overline{СБ}\frac{\sum_1^3 СБ_i}{3} = \frac{115}{3} = 39.33$		

За даними таблиці 4.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 4.3.

Таблиця 4.3 – Рівні комерційного потенціалу розробки.

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 –10	Низький
11–20	Нижче середнього
21–30	Середній
31–40	Вище середнього
41–48	Високий

Рівень комерційного потенціалу розробки, становить 39,33 балів, що відповідає рівню «вище середнього».

Розробка має якісні переваги відносно систем-аналогів, що застосовуються у схожому класі комп'ютерних ігор через підвищення рівня інтелектуальності поведінки штучних суперників, що дозволить покращити ігровий процес для гравців, що підвищує якість самої гри, у яку буде інтегровано розроблену інформаційну технологію. Розробка є перспективною через те, що вона реалізує у собі загальну систему алгоритмів штучного інтелекту, що можуть бути використані у безлічі ігрових проектів багатьох жанрів із мінімальною кількістю роботи по адаптації розробленої інформаційної технології до особливостей ігрового рушія певного ігрового проекту. Отже розробка може принести користь у розробці штучного інтелекту штучних суперників та неігрових персонажів ряді відеоігор. Розроблена інформаційна технологія є конкурентоспроможною, адже вона містить основні необхідні методи та алгоритми для прийняття рішень у ігровому середовищі, що продукує підвищений рівень інтелектуальності поведінки штучних суперників. Окрім того розробка використовує модель паралельних обчислень для підвищення швидкодії усієї технології. Алгоритм прийняття рішень реалізований у розробці є ефективнішим за ті, що реалізовані у багатьох існуючих іграх.

Система штучного інтелекту гри Enter the Gungeon для прийняття рішень опирається на модель простого переходу між станами ігрового середовища, не зважаючи на нюанси кожної окремої ситуації, та рівень оптимальності тих чи інших дій у кожній ситуації. Інформаційна технологія, що розробляється пропонує метод, що враховує оптимальність кожного ходу аналізуючи поточну ситуацію в ігровому середовищі. Система штучного інтелекту гри Dwarf Fortress є достатньо сильною, вона детально враховує нюанси ігрових ситуацій. Це займає багато обчислювальних ресурсів. Через відсутність реалізації паралельних обчислень, гра через деякий час, коли ігрових об'єктів стає все більше, починає занадто довго обраховувати кожен хід штучних неігрових

персонажів. Інформаційна технологія, що розробляється реалізує модель паралельних обчислень, що у перспективі розширення як технології так і ігрового рушія гри, у який технологія буде інтегрована, мінімізує виникнення відчутної затримки у обчисленнях.

Повний аналіз систем-аналогів, поданий у першому розділі магістерської кваліфікаційної роботи показує їх недоліки, що вирішені у інформаційній технології, що розробляється. Розробка є перспективною з точки зору її здійсненності. Наявні фахівці із достатнім рівнем теоретичної та практичної підготовки. Існує робочий прототип, що показав свою ефективність у виконанні поставлених перед ним задач. В наявності необхідний інструментарій, а також фінансові ресурси. Розробка може бути інтегрована у комп'ютерну гру, що може продаватись на відповідних відкритих майданчиках. Існує можливість переговорів із ігровими виданнями, які на даний момент не проводились.

Розробка спроектована таким чином, щоб її було легко розширювати у майбутньому, модифікувати та покращувати. Саму технологію, разом із її можливими модифікаціями можна інтегрувати у якості модуля ігрового штучного інтелекту у безліч ігрових проектів, що є основним напрямком її подальшого впровадження. Завершений ігровий проект, в залежності від якості ігрового процесу, що залежить у тому числі від інших складових гри, може показати як високий комерційний потенціал, так і низький. У першому випадку існує можливість дистриб'юції відеогри за допомогою ігрового видання, що значно підвищує кількість реалізованих копій гри.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження результатів

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення, яке складається з таких етапів:

1-й етап: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

2-й етап: розрахунок загальних витрат на виконання даної роботи;

3-й етап: прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

1. Виконаємо розрахунок витрат враховуючи те, що для розробки інформаційної технології було залучено одного розробника програмного забезпечення. Основна заробітна плата кожного із працівників $З_о$, якщо вони працюють в наукових установах бюджетної сфери:

$$З_о = \frac{М}{T_p} \cdot t \text{ [грн]}, \quad (4.1)$$

де М – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн;

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні;

t – число робочих днів роботи розробника (дослідника), розробка програмного забезпечення триває 80 днів.

Зроблені розрахунки внесені до таблиці 4.5:

Таблиця 4.5 – Основна заробітна плата розробників.

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.	Примітка
Програміст	4000	173.39	70	12000	
Науковець	6500	282.60	70	19500	
Всього				$\sum З_о$	31500

2. Додаткова заробітна плата $З_р$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12%) від суми основної заробітної плати розробників та робітників розраховується за формулою:

$$З_р = 0.11 \cdot 31500 = 3465(грн).$$

3. Нарахування на заробітну плату $Н_{ЗП}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується за формулою:

$$Н_{зп} = (З_0 \cdot З_д) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (4.2)$$

де $З_0$ – основна заробітна плата розробника, грн.;

$З_Д$ – додаткова заробітна плата розробника, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування – 37.3%.

$$Н_{ЗП} = (31500 + 3465) \cdot 0{,}37 = 12937(грн)$$

4. Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час (чи для) виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц \cdot Т}{12 \cdot Т_В}[грн], \quad (4.3)$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

Т – фактична тривалість використання, міс;

$Т_В$– термін, використання обладнання, приміщень тощо, місяці, роки.

Зроблені розрахунки наведено в таблиці 4.6

Таблиця 4.6 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, роки	Фактична тривалість використання, міс.	Величина амортизаційних відрахувань, грн
Офісне приміщення	10000	1	4	2500
ПК	25000	1	4	2200
Всього				4700

5. Інформацію про матеріали, що використовуються при виготовленні даного інноваційного продукту внесено до таблиці 4.7.

Таблиця 4.7 – Матеріали, що використовуються при виготовленні даного продукту

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу з урахуванням доставки, грн
Папір (пачка)	100,00	1	110,00
Канцтовари	40,00	1	46,00
Всього			156,00

Під час розробки програмного продукту використовувалось безкоштовне відкрите програмне забезпечення.

6. Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}, \quad (4.4)$$

де В – вартість 1 кВт-год. електроенергії, 1.90 грн/кВт;

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи обладнання, годин;

$К_п$ – коефіцієнт використання потужності.

Потужність використовуваного комп'ютера становить П=0.6 кВт.

Фактична кількість годин роботи обладнання – 552 год (70 робочих днів по 8 годин на день).

$$В_е = 1.90 \cdot 0{,}6 \cdot 552 \cdot 0{,}6 = 377.57 \text{ (грн)}.$$

7. Сума всіх попередніх статей витрат дає витрати на виконання даної частини розділу роботи В.

$$В = 31500 + 3465 + 12937 + 4700 + 156 + 377.57 = 53135.13$$

2-й етап: розрахунок загальних витрат на виконання даної роботи.

$$В_{заг} = \frac{В}{\alpha} [грн], \quad (4.5)$$

$$В_{заг} = \frac{53135.13}{1} = 53135.13 [грн]$$

3-й етап. Прогнозування загальних витрат на виконання та впровадження результатів виконаної роботи. Прогнозування витрат ЗВ на виконання та впровадження виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} [грн], \quad (4.6)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Так, як розробка знаходиться:

- на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$
- на стадії технічного проектування , то $\beta \approx 0{,}2$
- на стадії розробки конструкторської документації то $\beta \approx 0{,}3$
- на стадії розробки технології, то $\beta \approx 0{,}4$

- на стадії науково-дослідних робіт, то β ≈0,1
- на стадії промислового зразка, β ≈0,7
- на стадії впровадження, то β ≈0,9

$$ЗВ = \frac{53135.13}{0,9} = 59039.03 \text{ (грн).}$$

Отже, прогноз загальних витрат на виконання та впровадження результатів становить 59039.03 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У даному підрозділі проведемо кількісне прогнозування, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Основним мірилом вигоди від впровадження результатів наукової роботи є чистий прибуток, що дозволить підприємству продовжити свою діяльність, покращуючи свої фінансові показники.

Виконання даної наукової роботи та впровадження її результатів складає приблизно 1 рік.

Позитивні результати від впровадження розробки очікуються вже в перший рік впровадження.

Проведемо детальніше прогнозування позитивних результатів та кількісне їх оцінювання по роках.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \cdot \Delta N)_i \text{ [грн]}, \quad (4.7)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Припустимо, що внаслідок впровадження результатів наукової розробки покращується якість кожної копії гри, що містить у собі розробку, що дозволяє підвищити ціну реалізації гри на 25 грн, а реалізована кількість копій гри збільшиться: протягом першого року – на 5000 од., протягом другого року – ще на 3000 од., протягом третього року – ще на 2000 од.

Орієнтовно: реалізація гри до впровадження результатів наукової розробки складала 100 шт., а її ціна – 100грн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_1$ протягом першого року складе:

$$\Delta П_1 = 100 \cdot 100 + (100+25) \cdot 5000 = 635000 \text{ (грн)}.$$

Обчислимо збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року:

$$\Delta П_2 = 100 \cdot 100 + (100+25) \cdot (5000+3000) = 1010000 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_3 = 100 \cdot 100 + (100+25) \cdot (5000+3000+2000) = 1260000 \text{ (грн)}.$$

Отже, відповідно до проведених розрахунків, прогнозований комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає:

1-й крок. Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, тобто ЗВ = PV= 59039.03 (грн).

2-й крок. Розрахунок очікуваного збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження проведено вище.

3-й крок. Будуємо вісь часу, на якій відображаємо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів. Платежі показуємо у ті терміни, коли вони здійснюються.

Припустимо, що загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнює 59039.03 грн. Результати вкладених у наукову розробку інвестицій почнуть з’являтися протягом трьох років. Ці результати виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на 635000 грн відносно базового року, у другому році – збільшення чистого прибутку на

1010000 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 1260000 грн (відносно базового року).

Тоді рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 4.1.

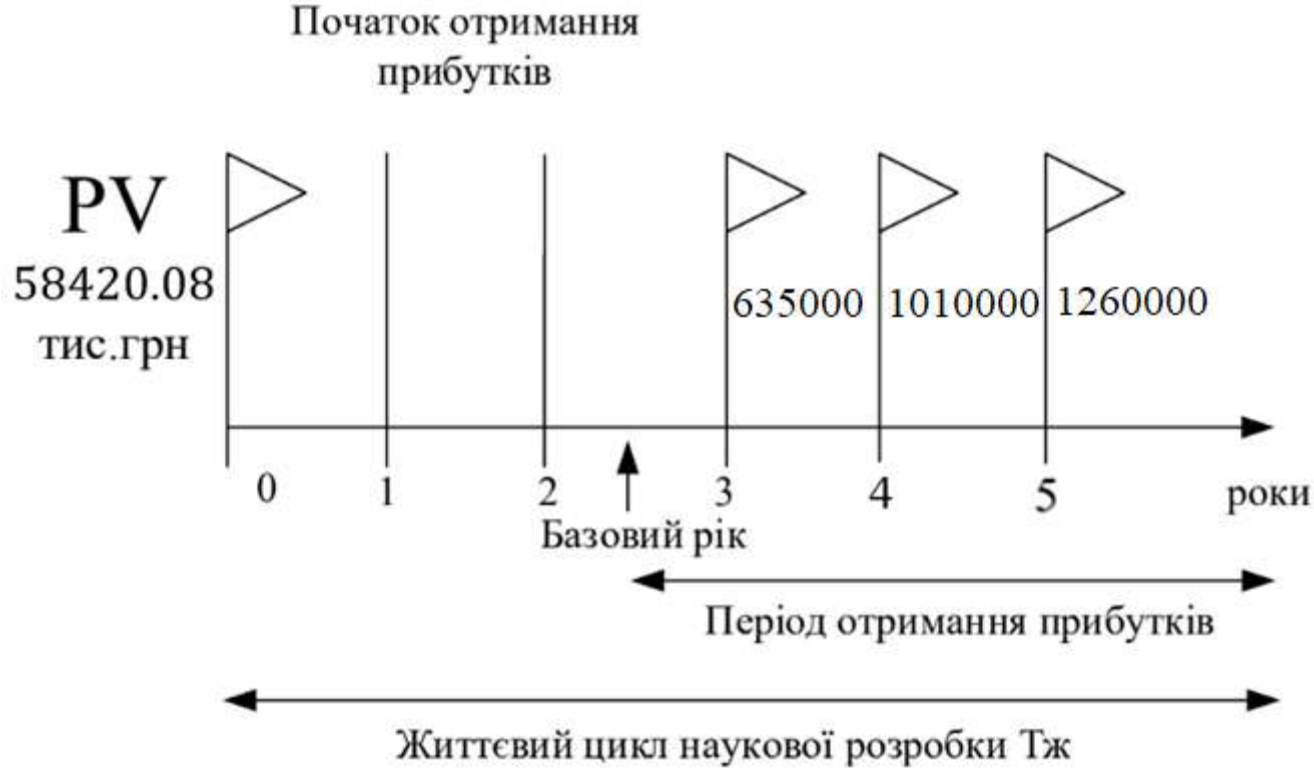

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.8)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні - 0,2;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{59039.03}{(1+0,2)^0} + \frac{635000}{(1+0,2)^3} + \frac{1010000}{(1+0,2)^4} + \frac{1260000}{(1+0,2)^5} = 1090475.6 \text{ (грн)}.$$

4-й крок. Розрахунок абсолютної ефективності вкладених інвестицій $Е_{абс}$.

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$, що обчислюється за формулою:

$$E_{абс} = (ПП - PV) \quad (4.9)$$

$$Е_{абс} = 1090475.6 - 59039.03 = 1031436.57$$

Оскільки $Е_{абс} > 0$, результат від проведення наукових досліджень щодо розробки програмного продукту та його впровадження принесе прибуток, тобто є доцільним.

5-й крок. Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1, \quad (4.10)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{1031436.57}{59039.03}} - 1 = \sqrt[3]{17.47} - 1 = 1.59 \text{ або } 159\%$$

Порівняємо $Е_в$ з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

Спрогнозуємо величину $\tau_{\text{мін.}}$ У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{\text{мін}}$ визначається за формулою:

$$\tau_{\text{мін}} = d + f, \quad (4.11)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,14;

f – показник, що характеризує ризикованість вкладень; величина f = 0,1.

$$\tau = 0,14 + 0,1 = 0,24$$

Оскільки $E_в = 159\% > \tau_{\text{мін}} = 0,24 = 24\%$, то у інвестора буде зацікавленість вкладати гроші в дану наукову розробку, так як він отримає значні прибутки.

6-й крок. Розраховують термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ за формулою:

$$T_{ок} = \frac{1}{E_в} \text{ [грн].} \quad (4.12)$$

$$T_{ok} = \frac{1}{1.59} = 0.62 \text{ (роки).}$$

Оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій менше трьох років ($T_{ок} < 3$ років), то фінансування нової розробки є доцільним.

4.5 Висновок

В даному розділі було проведено оцінювання комерційного потенціалу розробки інформаційної технології інтелектуальної поведінки штучних суперників у комп’ютерній грі.

Здійснено технологічний аудит із залученням трьох незалежних експертів, під час якого визначено, що рівень комерційного потенціалу розробки вище середнього.

Проведено аналіз існуючих систем-аналогів у складі розроблених комп’ютерних ігор. Визначено, що система прийняття рішень штучного інтелекту гри Enter the Gungeon не шукає оптимальних ходів для штучних суперників, а використовує модель лімітованих станів ігрового середовища та прямого переходу між ними. Гра Dwarf Fortress не використовує моделі паралельних обчислень, і тому при збільшенні кількості неігрових персонажів, для яких обчислюються оптимальні ходи, ігровий рушій витрачає більше часу на обчислення. Запропонована розробка вирішує обидва недоліка наведених систем-аналогів.

Крім того проведено розрахунки витрат на заробітну платню, витрати на амортизаційні потреби, витрати на комплектуючі та електроенергію. Загальна вартість реалізації становить 59039 грн.

Проведено прогнозування комерційних ефектів від впровадження результатів розробки. Розраховано, що основний прибуток від гри, що буде використовувати розробку у якості модулю ігрового штучного інтелекту буде отримуватись протягом трьох років. Чистий прибуток підприємства після впровадження розробки за три роки становить 1260000 грн.

Також здійснено обрахунок ефективності інвестицій та часу їх окупності. Життєвий цикл наукової розробки становить 3 роки. Розраховано абсолютну ефективність вкладених інвестицій, що становить 1031436.57 грн.

Визначено щорічну ефективність вкладених в наукову розробку інвестицій, що складає 159%. При мінімальному порозі 25% отримано дуже високий рівень ефективності вкладених інвестицій. Термін окупності розробки у складі комп’ютерної гри складає 0.62 року. Тому інвестування в інформаційну технологію інтелектуальної поведінки штучних суперників у комп’ютерній грі є доцільним для інвесторів.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію інтелектуальної поведінки штучних суперників у комп’ютерній грі. Під час аналізу предметної області виявлено основні задачі та проблеми, що постають перед ігровим штучним інтелектом. Проаналізовано системи-аналоги і визначено їх переваги та недоліки. Проведено оцінку прикладного значення ІТ та область її застосування.

В другому розділі магістерської кваліфікаційної роботи описано та проаналізовано методи та алгоритми й їх модифікації для вирішення задач ігрового штучного інтелекту, таких як прийняття рішень, пошук шляху та імітація зору штучних суперників. Серед них обрано оптимальні на основі їх переваг та недоліків.

Третій розділ описує процес практичної реалізації інформаційної технології, під час якого проведено проектування модулів ІТ із використанням методології функціонального моделювання і графічного опису процесів систем IDEF0. Визначено структурну організацію ІТ, розроблено діаграми класів її модулів, наведено опис полів та методів класів модулів, розроблено алгоритм роботи ІТ.

Описано особливості мови програмування Java, наведено проведено порівняння мов програмування Java та C#, обґрунтовано вибір Java для практичної реалізації ІТ. Описано відкритий ігровий фреймворк libGDX, та основні його модулі gdxAI, box2d та scene2d, що використовувались при програмній реалізації ІТ.

Здійснено тестування програмної реалізації ІТ, під час якого розроблена ІТ працювала відповідно до поставлених перед нею задач. Виявлено, що розпаралелювання обчислень модулю прийняття рішень із використанням модифікації вузлового розпаралелювання алгоритму Монте-Карло для пошуку у дереві підвищило його швидкодію на 21%. Проведено експертну оцінку ІТ, під

час якої виявлено, що на основі чотирьох критеріїв (ефективність роботи модулів прийняття рішень, пошуку шляху, імітації зору штучних супротивників та загальної швидкодії ІТ) розроблена ІТ працює на 5% ефективніше за системи-аналоги.

Під час економічного обґрунтування розробки проведено оцінювання економічного потенціалу розробки, проведений прогноз щодо витрат на розробку ІТ, виконання науково-дослідної, дослідно-конструкторської, конструкторсько-технологічних робіт та комерційного ефекту від реалізації результатів розробки, розраховано ефективність вкладених інвестицій та періоду їх окупності.

Результати досліджень було апробовано на XLVII науково-технічній конференції професорсько-викладацького складу, співробітників та студентів Вінницького національного технічного університету у 2019р.

За основними результатами досліджень опубліковано одну публікацію та подано заявку про реєстрацію авторского права на твір (комп’ютерну програму) [1].

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ільченко М.О. Особливості застосування моделі скінченного автомата при розробці комп'ютерних ігор / Ільченко М.О., Яровий А.А.: Збірник матеріалів XLVIII науково-технічної конференції Вінницького національного технічного університету, (Вінниця, 13-15 березня 2019 р.). – В.: ВНТУ, 2019 – С. 769-770. [Електронний ресурс] – Режим доступу:https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2019/paper/download/7329/6270
2. Artificial Intelligence in Games – [Електронний ресурс]. – Режим доступу: https://medium.com/aifrontiers/an-overview-of-artificial-intelligence-for-video-games-f491229c0e7d
3. The Next Generation 1996 Lexicon A to Z: NPC (Nonplayer Character)". Next Generation. No. 15. Imagine Media. March 1996. p. 38.
4. Yannakakis, Geogios N (2012). "Game AI revisited". Proceedings of the 9th Conference on Computing Frontiers: 285–292.
5. 'Artificial Intelligence' Has Become Meaningless – [Електронний ресурс]. – Режим доступу: https://www.theatlantic.com/technology/archive/2017/03/what-is-artificial-intelligence/518547/
6. Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C., & Fernández-Leiva, A. J. (2015). Game artificial intelligence: challenges for the scientific community.
7. Yannakakis, G. N. (2012, May). Game AI revisited. In Proceedings of the 9th conference on Computing Frontiers (pp. 285–292). ACM.
8. Hagelback, Johan, and Stefan J. Johansson. "Dealing with fog of war in a real time strategy game environment." In Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On, pp. 55-62. IEEE, 2008.
9. Grid-based path-finding." In Conference of the Canadian Society for Computational Studies of Intelligence, pp. 44-55. Springer, Berlin, Heidelberg, 2002.

10. Goodwin, S. D., Menon, S., & Price, R. G. (2006). Pathfinding in open terrain. In Proceedings of International Academic Conference on the Future of Game Design and Technology.
11. Scott, Bob (2002). "The Illusion of Intelligence". In Rabin, Steve (ed.). AI Game Programming Wisdom. Charles River Media. pp. 16–20.
12. What is Finite State Machine – Medium [Електронний ресурс]. https://medium.com/@mlbors/what-is-a-finite-state-machine-6d8dec727e2c
13. Finite-State Machines: Theory and Implementation – Envatotuts+ [Електронний ресурс]. https://gamedevelopment.tutsplus.com/tutorials/ finite-state-machines-theory-and-implementation--gamedev-11867
14. Brügmann, Bernd (1993). «Monte Carlo Go». Technical report, Department of Physics, Syracuse University.
15. G.M.J.B. Chaslot; M.H.M. Winands; J.W.H.M. Uiterwijk; H.J. van den Herik; B. Bouzy (2008). "Progressive Strategies for Monte-Carlo Tree Search". *New Mathematics and Natural Computation*. 4 (3): 343–359.
16. Bradberry, Jeff – "Introduction to Monte Carlo Tree Search". – [Електронний ресурс]. – Режим доступу: http://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/
17. Kocsis, Levente; Szepesvári, Csaba (2006). "Bandit based Monte-Carlo Planning". In Fürnkranz, Johannes; Scheffer, Tobias; Spiliopoulou, Myra (eds.). Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18–22, 2006, Proceedings. Lecture Notes in Computer Science. 4212. Springer. pp. 282–293.
18. Swiechowski, M.; Mandziuk, J., "Self-Adaptation of Playing Strategies in General Game Playing" (2010), IEEE Transactions on Computational Intelligence and AI in Games, vol: 6(4), pp. 367-381
19. Sylvain Gelly; Yizao Wang; Rémi Munos; Olivier Teytaud (November 2006). «Modification of UCT with Patterns in Monte-Carlo Go». Technical report, INRIA

20. Sylvain Gelly; David Silver (2007). "Combining Online and Offline Knowledge in UCT" . Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20–24, 2007. Zoubin Ghahramani (ed.). ACM. pp. 273–280. ISBN 978-1-59593-793-3
21. G.M.J.B. Chaslot; M.H.M. Winands; J.W.H.M. Uiterwijk; H.J. van den Herik; B. Bouzy (2008). "Progressive Strategies for Monte-Carlo Tree Search". New Mathematics and Natural Computation. 4 (3): 343–359
22. David Silver (2009). Reinforcement Learning and Simulation-Based Search in Computer Go. PhD thesis, University of Alberta
23. Dijkstra E. W. A note on two problems in connexion with graphs // *Numer. Math* — Springer Science+Business Media, 1959. — Vol. 1, Iss. 1. — P. 269–271. — ISSN 0029-599X
24. A* – GitHub [Електронний ресурс]. – Режим доступу: https://github.com/libgdx/gdx-ai/wiki/A*
25. Lee, C.Y., «An Algorithm for Path Connections and Its Applications», IRE Transactions on Electronic Computers, vol. EC-10, number 2, pp. 364—365, 1961
26. National Institute of Standarts and Technology – "Manhattan distance". – [Електронний ресурс]. – Режим доступу: https://xlinux.nist.gov/dads/HTML/manhattanDistance.html
27. Anton, Howard (1994), Elementary Linear Algebra (7th ed.), John Wiley & Sons, pp. 170–171, ISBN 978-0-471-58742-2
28. James M. Abello, Panos M. Pardalos, and Mauricio G. C. Resende (editors) (2002). Handbook of Massive Data Sets. Springer. ISBN 1-4020-0489-3
29. Box2D C++ tutorials – Ray casting [Електронний ресурс]: http://www.iforce2d.net/b2dtut/raycasting
30. IDEF0 – Habr [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/trinion/blog/322832/
31. Буч, Рамбо, Якобсон, 2006, Диаграмма классов, с. 120

32. Брюс Эккель. Философия Java Thinking in Java. – 4-е изд. –СПб.: Питер, 2018. –1168 с. – ISBN 978-5-496-01127-3.

33. Suryakumar Balakrishnan Nair Learning LibGDX Game Development 2015 // Birmingham-Mumbai: Packt Publishing, 2014 500 p.: ISBN-13: 978-1783554775