Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Кафедра _____Програмної інженерії_____

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти - другий (магістерський)

Дослідження методів формалізації для ігрових ситуацій
(тема)

Виконав: студент 2 курсу, групи ПЗСм-18-1___
_____Шутєєв І.В._____
(прізвище, ініціали)

спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-професійої програми_____
(тип програми)

Програмне забезпечення систем___
(повна назва освітньої програми)

Керівник _____к.т.н., доц. Каук І.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет_Комп'ютерних наук_____

Кафедра_Програмної інженерії_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _121-Інженерія програмного забезпечення_____

Тип програми освітньо-професійна програма

Освітня програма _Програмне забезпечення систем

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2019 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ
студентові _Шутєєву Іллі Вячеславовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів формалізації для ігрових ситуацій
затверджена наказом по університету від "___" _____20 ___ р № _____

2. Термін подання студентом роботи до екзаменаційної комісії _____2019 р.

3. Вихідні дані до роботи *методи формалізації для ігрових ситуацій, алгоритми кодування, пояснювальна записка, ОС Windows, середовище об'єктно-орієнтованого проектування*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд методів моделювання, пошук датасетів для ігрових ситуацій, порівняння метрік програмних реалізацій методів моделювання, практична користь отриманих результатів аналізу.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1.	Аналіз предметної галузі	10 вересня 2019р.	
2.	Огляд існуючих методів	20 вересня 2019р.	
3.	Методи ком’ютерного моделювання	15 жовтня 2019р.	
4.	Підготовка пояснювальної записки	30 жовтня 2019р.	
5.	Спецчастина	1 листопада 2019р.	
6.	Підготовка презентації та доповіді	15 листопада 2019р.	
7.	Попередній захист	8 грудня 2019р.	
8.	Нормоконтроль, рецензування	12 грудня 2019р.	
9.	Занесення диплома в електронний архів		
10.	Допуск до захисту у зав. кафедри		
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____ _____2019 р.

Студент _____Шутєєв І.В.
(підпис)

Керівник роботи _____ __к.т.н., доц. Каук В.І._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Атестаційна робота магістра містить: 96 с., 24 рис., 2 табл., 4 додатки, 23 джерела.

C#, VISUAL STUDIO 2017, АТЕСТАЦІЙНА РОБОТА, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, СТАТИСТИЧНЕ МОДЕЛЮВАННЯ, КОНСОЛЬНЕ ЗАСТОСУВАННЯ, ЛОГАРІФМІЧНА РЕГРЕСІЯ, СИСТЕМА, КОМПОНЕНТ.

Об’єктом дослідження є ігровий процес. Предметом дослідження є методи моделювання ігрових ситуацій.

Метою роботи є порівняння методів формалізації для ігрових ситуацій.

Методи розробки базуються на технології .NET та мові програмування C#. У результаті роботи здійснена програмна реалізація імітаційного та статистичного методів ком’ютерного моделювання. Було проведене порівняння цих моделей за такими параметрами як точність та швидкість роботи, складність додавання нових параметрів до моделі.

ASP .NET, C#, VISUAL STUDIO 2017, ATTESTATION WORK, SIMULATION MODELING, STATISTICAL MODELING, CONSOLE APPLICATION, LOGARIFM REGRESSION, SYSTEM, COMPONENT.

The object of the study is the game process. The subject of the study is the comparison of methods of game situations modeling.

The purpose of the work is to compare the formalization methods for game situations.

Development methods are based on .NET platform and C# programming language. As a result, computer simulation and statistical methods software was implemented. These models were compared in terms of accuracy and speed, the difficulty of adding new parameters to the model.

ЗМІСТ

ВСТУП

Гра – це моделювання людиною іншого виду діяльності з метою розважання чи навчання. У грі обов'язково присутні певні правила, що обмежують дії гравців [1].

Для гри необхідні один або кілька гравців. У грі, що розрахована на одну людину, гравець, дотримуючись встановленої для себе тактики та правил, намагається вирішити складне завдання. Ігри для декількох гравців часто мають на меті виграш, перемогу над іншими, тобто гра є змаганням. В іграх з елементами змагання гравці можуть складати команди – колективи, що намагаються досягти однієї мети. Відповідні ігри мають назву командних [2].

У математиці також є поняття гри. Це клас математичних задач, що займаються прийняттям рішень. Розділ математики, що присвячений вивченню ігор, називається «Теорія Ігор». Вибір стратегії є характерним завданням для цього класу задач. Правильна стратегія забезпечує найкращий результат для гравців. Теорія ігор, попри слово «гра» в назві, займається серйозними задачами, часто економічного характеру. Математична теорія ігор не має на меті пошук виграшних стратегій у простих іграх. У них матриця виграшів-програшів вважається заданою. Але на основі цієї матриці можливо провести аналіз ситуації, в якій ігровий процес повторюється багато разів, особливо коли випадкові елементи є частиною матриці виграшів-програшів.

Метою моделювання, у тому числі і ігор, є здобуття, обробка, представлення і використання інформації про об'єкти, які взаємодіють між собою і зовнішнім середовищем [3]. Завдяки моделюванню ігор та методів прийняття рішень в них можна зрозуміти яка тактика є виграшною або програшною.

Моделювання – це процес створення моделі для подальшого дослідження об'єкта (процесу, явища). Інформаційна модель – це опис важливих для певного дослідження властивостей об'єкта (явища, процесу). Комп'ютерна модель – це інформаційна модель, реалізована за допомогою програмного середовища

(текстового або графічного редактора, редактора презентацій, середовища програмування тощо).

Об’єктом дослідження даної роботи є ігровий процес. Предметом дослідження є моделювання ігрових ситуацій.

Створення різних програмних моделей ігрових процесів допоможе наглядно оцінити та порівняти характеристики ігор серед яких і ймовірність перемоги або програшу від використовування певної стратегії. Така програма допоможе обирати користувачу необхідну тактику залежно від ігрової ситуації.

Задачами дослідження є:

- порівняння моделей формалізації ігрових ситуацій;
- обрання найбільш відповідних моделей;
- отримання необхідних вхідних даних для генерування моделей;
- вибір метрик для порівняння моделей;
- програмна реалізація обраних моделей;
- порівняння отриманих результатів.

Наукова новизна отриманих результатів дослідження полягає в порівнянні точності, швидкості роботи та практичності імітаційної та статистичної моделі, порівнянні їх застосування.

Актуальність даної роботи полягає в одержані метрик цих моделей для баскетбольних ігрових ситуацій, отримані кореляції між характеристиками гравців та результатом гри, що може бути використано тренером для корегування тренувального процесу для поліпшення результатів команди. Використовуючи результати дослідження можна завершити розробку повноцінного додатка здатного передбачати результати ігрових ситуацій. Це допоможе знаходити найоптимальніші характеристики для досягнення найвищого результату в обраній предметній області за допомогою симуляції ігрового процесу. Програма може бути використана в різноманітних індустріях: економіка, політика, спорт, та інші.

Результати дослідження опубліковані на 15-ій міжнародній науковій конференції «Science And Society» від 8-го листопада 2019-го року [4].

1 АНАЛІЗ СТАНУ РОЗВ'ЯЗАННЯ ПРОБЛЕМИ

1.1 Комп'ютерне моделювання

Ефективним способом вивчення явищ навколишньої дійсності є науковий експеримент, що складається у відтворенні досліджуваного явища природи в керованих і контрольованих умовах. Однак часто проведення експерименту неможливо або вимагає занадто великих економічних витрат і може привести до небажаних наслідків. У цьому випадку досліджуваний об'єкт замінюють комп'ютерною моделлю і досліджують її поведінку при різних зовнішніх впливах. Повсюдне поширення персональних комп'ютерів, інформаційних технологій, створення потужних суперЕОМ зробило комп'ютерне моделювання одним з результативних методів вивчення фізичних, технічних, біологічних, економічних і інших систем. Часто комп'ютерні моделі простіше і зручніше досліджувати, вони дозволяють проводити обчислювальні експерименти, реальна постановка яких ускладнена або може дати непередбачуваний результат. Логічність і формальність комп'ютерних моделей дозволяє виявити основні фактори, що визначають властивості досліджуваних об'єктів, досліджувати відгук фізичної системи на зміни її параметрів і початкових умов.

Комп'ютерне моделювання вимагає абстрагування від конкретної природи явищ, побудови спочатку якісної, а потім і кількісної моделі. Після цього відбувається проведення серії обчислювальних експериментів на комп'ютері, інтерпретація результатів, зіставлення результатів моделювання з поведінкою досліджуваного об'єкта, подальше уточнення моделі і т.д. Обчислювальний експеримент фактично є експериментом над математичною моделлю досліджуваного об'єкта, що проводиться за допомогою ЕОМ [5]. Часто він значно дешевше і доступніше натурного експерименту, його виконання вимагає меншого часу, він дає більш детальну інформацію про величини, що характеризують стан системи.

Сутність комп'ютерного моделювання системи полягає в створенні комп'ютерної програми (пакета програм), яка описує поведінку елементів

досліджуваної системи в процесі її функціонування, що враховує їх взаємодію між собою і зовнішнім середовищем, і проведенні на ЕОМ серії обчислювальних експериментів. Це робиться з метою вивчення природи і поведінки об’єкта, його оптимізації та структурного розвитку, прогнозування нових явищ. Перерахуємо вимоги, яким повинна задовольняти модель досліджуваної системи [6]:

- повнота моделі, тобто можливість обчислення всіх характеристик системи з необхідною точністю і достовірністю;
- гнучкість моделі, що дозволяє відтворювати і програвати різні ситуації і процеси, змінювати структуру, алгоритми і параметри досліджуваної системи;
- загальна тривалість розробки і реалізації, що характеризує тимчасові витрати на створення моделі;
- блочність структури, яка припускає додавання, виняток і заміну де-яких частин (блоків) моделі.

Крім того, інформаційне забезпечення, програмні та технічні засоби повинні дозволяти моделі обмінюватися інформацією з відповідною базою даних і забезпечувати ефективну машинну реалізацію і зручну роботу користувача.

До основних етапів комп’ютерного моделювання відносяться:

- постановка завдання, опис досліджуваної системи і виявлення її компонентів і елементарних актів взаємодії;
- формалізація, тобто створення математичної моделі, що представляє собою систему рівнянь і відбиває сутність досліджуваного об’єкта;
- розробка алгоритму, реалізація якого дозволить вирішити поставлене завдання;
- написання програми на конкретній мові програмування;
- планування та виконання обчислень на ЕОМ, доробка програми і отримання результатів;
- аналіз і інтерпретація результатів, їх зіставлення з емпіричними даними. Потім все це повторюється на наступному рівні.

Розробка комп'ютерної моделі об'єкта є послідовність ітерацій: спочатку на основі наявної інформації про систему S будується модель M_1, проводиться серія обчислювальних експериментів, результати аналізуються. При отриманні нової інформації про об'єкт S враховуються додаткові чинники, виходить модель M_2, поведінка якої теж досліджується на ЕОМ. Після цього створюються моделі M_3, M_4, і т.д. до тих пір, поки не вийде модель, що з необхідною точністю відповідає системі S.

У загальному випадку поведінка досліджуваної системи S описується законом функціонування $Y(t) = F(X, V, H, t)$, де $X = (x_1, x_2, \dots, x_n)$, – вектор вхідних впливів (стимулів), $Y = (y_1, y_2, \dots, y_m)$ – вектор вихідних сигналів (відгуків, реакцій), $V = (v_1, v_2, \dots, v_k)$ – вектор впливів зовнішнього середовища, $H = (h_1, h_2, \dots, h_l)$ – вектор власних параметрів системи [6]. Закон функціонування може мати вигляд словесного правила, таблиці, алгоритму, функції, сукупності логічних умов і т.д. У разі, коли закон функціонування містить час, говорять про динамічні моделі та системи. Наприклад, розгін і гальмування асинхронного двигуна, функціонування обчислювальної мережі, системи масового обслуговування. У всіх цих випадках стан системи, а значить і її моделі, змінюється з плином часу.

Якщо поведінка системи описується законом $Y = F(X, V, H)$, що не містить час t явно, то мова йде про статичні моделі та системи, вирішення стаціонарних задач і т.д. Наведемо кілька прикладів: розрахунок нелінійного ланцюга постійного струму, знаходження стаціонарного розподілу температури в стержні при постійних температурах його кінців, форми пружною плівки, натягнутої на каркас, профілю швидкостей в сталому перебігу в'язкої рідини і т.д.

Функціонування системи можна розглядати як послідовну зміну станів $q_1(t), q_2(t), \dots, q_r(t)$, яким відповідають деякі точки в багатовимірному фазовому просторі. Безліч всіх точок $A = \{a_1, a_2, \dots, a_s\}$, що відповідають всіляким станам системи, називають простором станів об'єкта (або моделі). Кожній реалізації процесу відповідає одна фазова траєкторія, що проходить через деякі точки з

множини A. Якщо математична модель містить елемент випадковості, то виходить стохастична комп'ютерна модель. В окремому випадку, коли параметри системи і зовнішні впливи однозначно визначають вихідні сигнали, говорять про детерміновану моделі.

1.2 Принципи комп'ютерного моделювання

Модель – це об'єкт, який замінює досліджувану систему, і імітує її структуру і поведінку. Моделлю може бути матеріальний об'єкт, сукупність особливим чином впорядкованих даних, система математичних рівнянь або комп'ютерна програма. Під моделюванням розуміють представлення основних характеристик об'єкта дослідження за допомогою іншої системи (матеріального об'єкта, сукупності рівнянь, комп'ютерної програми). Перерахуємо принципи моделювання [7]:

- принцип адекватності. Модель повинна враховувати найбільш суттєві сторони досліджуваного об'єкта і відображати його властивості з прийнятною точністю. Тільки в цьому випадку результати моделювання можна поширити на об'єкт дослідження;
- принцип простоти і економічності. Модель повинна бути досить простою для того щоб її використання було ефективно і економічно вигідно. Вона не повинна бути складнішою, ніж це потрібно для дослідника;
- принцип інформаційної достатності. При повній відсутності інформації про об'єкт побудувати модель неможливо. При наявності повної інформації моделювання позбавлене сенсу. Існує рівень інформаційної достатності, при досягненні якого може бути побудована модель системи;
- принцип здійсненності. Створювана модель повинна забезпечувати досягнення поставленої мети дослідження за кінцевий час;
- принцип множинності та єдності моделей. Будь-яка конкретна модель відображає лише деякі сторони реальної системи. Для повного дослідження необхідно побудувати ряд моделей, що відображають

найбільш суттєві сторони досліджуваного процесу і мають щось спільне. Кожна наступна модель повинна доповнювати та уточнювати попередню;

- принцип системності. Досліджувана система подана в вигляді сукупності взаємодіючих між собою підсистем, які моделюються стандартними математичними методами. При цьому властивості системи не є сумою властивостей її елементів;
- принцип параметризації. Деякі підсистеми модельованої системи можуть бути охарактеризовані єдиним параметром (вектором, матрицею, графіком, формулою).

Модель повинна відповідати таким вимогам:

- бути адекватною, тобто відображати найбільш істотні сторони досліджуваного об’єкта з необхідною точністю;
- сприяти вирішенню певного класу задач;
- бути простою і зрозумілою, ґрунтуватися на мінімальній кількості припущень;
- дозволяти модифікувати і доповнювати себе, переходити до інших даних;
- бути зручною у використанні.

Об’єкт пізнання досліджується емпіричними методами (спостереження, експеримент). Встановлені факти є основою для побудови математичної моделі. Отримана система математичних рівнянь може досліджуватися аналітичними методами або за допомогою ЕОМ, – в цьому випадку мова йде про створення комп’ютерної моделі досліджуваного явища. Проводиться серія обчислювальних експериментів або комп’ютерних імітацій, отримані результати зіставляються з результатами аналітичного дослідження математичної моделі та експериментальними даними. Висновки враховуються для поліпшення методики експериментального вивчення об’єкта дослідження, розвитку математичної моделі та вдосконалення комп’ютерної моделі. Дослідження соціальних і економічних процесів відрізняється лише неможливістю в повній мірі використовувати експериментальні методи.

1.3 Види комп’ютерних моделей

Під комп’ютерним моделюванням в найширшому сенсі будемо розуміти процес створення і дослідження моделей за допомогою комп’ютера. Виділяють наступні види моделювання:

- фізичне моделювання [6]. Комп’ютер – частина експериментальної установки або тренажера, він сприймає зовнішні сигнали, здійснює відповідні розрахунки і видає сигнали, що керують різними маніпуляторами. Наприклад, навчальна модель літака, що представляє собою кабіну, встановлену на відповідних маніпуляторах, з’єднаних з комп’ютером, який реагує на дії пілота, імітуючи політ реального літака;
- динамічне або чисельне моделювання, що припускає чисельне рішення системи алгебраїчних і диференціальних рівнянь способами обчислювальної математики і проведення обчислювального експерименту при різних параметрах системи, початкових умовах і зовнішніх впливах [8]. Використовується для моделювання різних фізичних, біологічних, соціальних та інших явищ: коливання маятника, поширення хвилі, зміна чисельності населення, популяції цього виду тварин і т.д.;
- імітаційне моделювання полягає в створенні комп’ютерної програми (або пакета програм), що імітує поведінку складної технічної, економічної чи іншої системи на ЕОМ з необхідною точністю [9]. Імітаційне моделювання передбачає формальний опис логіки функціонування досліджуваної системи з плином часу, який враховує суттєві взаємодії її компонентів і забезпечує проведення статистичних експериментів. Об’єктно-орієнтовані комп’ютерні симуляції використовуються для дослідження поведінки економічних, біологічних, соціальних та інших систем, для створення комп’ютерних ігор, так званого «віртуального світу», навчальних програм і анімацій. Наприклад, модель технологічного процесу, аеродрому, деякою галузі виробництва і т.д.;

– статистичне моделювання використовується для вивчення стохастичних систем і складається в багаторазовому проведенні випробувань з подальшою статистичною обробкою отриманих результатів [10]. Подібні моделі дозволяють вивчати поведінку всіляких систем масового обслуговування, багатопроцесорних систем, інформаційно-обчислювальних мереж, різних динамічних систем, на які впливають випадкові чинники. Статистичні моделі застосовуються при вирішенні імовірнісних задач, а також при обробці великих масивів даних (інтерполяція, екстраполяція, регресія, кореляція, розрахунок параметрів розподілу і т.д.). Вони відрізняються від детермінованих моделей, використання яких передбачає чисельне рішення систем алгебраїчних або диференціальних рівнянь, або заміну досліджуваного об’єкта детермінованим автоматом;
– інформаційне моделювання полягає в створенні інформаційної моделі, тобто сукупності спеціальним чином організованих даних (знаків, сигналів), що відображають найбільш суттєві сторони досліджуваного об’єкта. Розрізняють наочні, графічні, анімаційні, текстові, табличні інформаційні моделі. До них відносяться всілякі схеми, графи, графіки, таблиці, діаграми, малюнки, анімації, виконані на ЕОМ, в тому числі цифрова карта зоряного неба, комп’ютерна модель земної поверхні і т.д;
– моделювання знань передбачає побудову системи штучного інтелекту, в основі якої лежить база знань деякої предметної області (частини реального світу). Бази знань складаються з фактів (даних) і правил. Наприклад, комп’ютерна програма, яка вміє грати в шахи, повинна оперувати інформацією про «здібності» різних шахових фігур і «знати» правила гри. До даного виду моделей відносять семантичні мережі, логічних моделі знань, експертні системи, логічні ігри і т.д.

Виходячи з цілей моделювання, комп’ютерні моделі поділяють на групи:

- дескриптивні моделі, використовувані для розуміння природи досліджуваного об’єкта, виявлення найбільш істотних факторів, що впливають на його поведінку;
- оптимізаційні моделі, що дозволяють обрати оптимальний спосіб управління технічною, соціально-економічною чи іншою системою;
- прогностичні моделі, які допомагають прогнозувати стан об’єкта в наступні моменти часу (модель земної атмосфери, що дозволяє передбачити погоду);
- навчальні моделі, що застосовуються для навчання, тренінгу і тестування учнів, студентів, майбутніх фахівців;
- ігрові моделі, що дозволяють створити ігрову ситуацію, яка імітує управління армією, державою, підприємством, людиною, літаком і т.д., або грають в шахи, шашки та інші логічні ігри.

Виходячи з теми дослідження, в роботі повинні використовуватися прогностично-ігрові моделі.

1.4 Класифікація комп’ютерних моделей за типом математичної схеми

Облік способу представлення стану системи і ступеня випадковості модельованих процесів дозволяє побудувати таблицю 1.1.

Таблиця 1.1 – Класифікація комп’ютерних моделей за станом системи і ступенем випадковості модельованих процесів

	Дискретні	Безперервні
Детерміновані	Дискретно-детерміновані моделі	Безперервно-детерміновані моделі
Стохастичні	Дискретно-стохастичні моделі	Безперервно-стохастичні моделі

В теорії моделювання систем комп’ютерні моделі підрозділяються на чисельні, імітаційні, статистичні і логічні. При комп’ютерному моделюванні, як

правило, використовують одну з типових математичних схем: диференціальні рівняння, детерміновані і імовірнісні автомати, системи масового обслуговування, мережі Петрі і т.д.

За типом математичної схеми розрізняють:

– безперервно-детерміновані моделі, які використовуються для моделювання динамічних систем і передбачають вирішення системи диференціальних рівнянь. Математичні схеми цього виду називаються D-схемами (від англ. Dynamic);

– дискретно-детерміновані моделі використовуються для дослідження дискретних систем, які можуть перебувати в одному з безлічі внутрішніх станів. Вони моделюються абстрактним кінцевим автоматом, що задається F-схемою (від англ. Finite automata): $F =< Q, X, Y, \phi, \Psi >$. Тут $X, Y,$ – множини вхідних і вихідних сигналів, Q – безліччю внутрішніх станів, ϕ – функція переходів, Ψ – функція виходів;

– дискретно-стохастичні моделі передбачають використання схеми імовірнісних автоматів, функціонування яких містить елемент випадковості. Вони також називаються P-схемами (від англ. Probabilistic automat). Переходи такого автомата з одного стану в інший визначається відповідною матрицею ймовірностей;

– безперервно-стохастичні моделі як правило застосовуються для вивчення систем масового обслуговування і називаються Q-схемами (від англ. Queueing system). Для функціонування деяких економічних, виробничих, технічних систем притаманне випадкове поява вимог (заявок) на обслуговування і випадкове час обслуговування;

– мережеві моделі використовуються для аналізу складних систем, в яких одночасно протікає кілька процесів. У цьому випадку говорять про мережі Петрі і N-схеми (від англ. Petri Nets). Мережа Петрі задається четвіркою $N =< B, D, I, O >$, де B – безліч позицій, D – безліч переходів,

I – вхідна функція, O – вихідна функція. Маркована N-схема дозволяє промоделювати паралельні і конкуруючі процеси в різних системах;

– комбіновані схеми ґрунтуються на понятті агрегативной системи і називаються А-схемами (від англ. Aggregate system). Цей універсальних підхід, розроблений Бусленко Н.П. [11], дозволяє досліджувати всілякі системи, які розглядаються як сукупність взаємопов'язаних між собою агрегатів. Кожен агрегат характеризується векторами станів, параметрів, впливу зовнішнього середовища, вхідних впливів (керуючих сигналів), початкових станів, вихідних сигналів, оператором переходів, оператором виходів.

Дослідження імітаційної моделі проводиться на цифрових і аналогових обчислювальних машинах. Використовувана імітаційна система включає в себе математичне, програмне, інформаційне, технічне та ергономічне забезпечення. Ефективність імітаційного моделювання характеризується точністю і достовірністю отриманих результатів, вартістю і часом створення моделі і роботи з нею, витратами машинних ресурсів. Для оцінки ефективності моделі необхідно отримані результати порівняти з результатами натурного експерименту, а також результатами аналітичного моделювання.

У де-яких випадках доводиться об'єднувати чисельні рішення диференціальних рівнянь і імітацію функціонування тієї чи іншої досить складної системи. У цьому випадку говорять про комбінований або аналітико-імітаційному моделюванні. Його основна перевага полягає в можливості дослідження складних систем, обліку дискретних і безперервних елементів, нелінійності різних характеристик, випадкові чинники. Аналітичне моделювання дозволяє проаналізувати тільки досить прості системи.

Одним з ефективних методів дослідження імітаційних моделей є метод статистичних випробувань. Він передбачає багаторазове відтворення того чи іншого процесу при різних параметрах, що змінюються випадковим чином по заданому закону. ЕОМ може провести 1000 випробувань і зареєструвати основні характеристики поведінки системи, її вихідні сигнали, а потім визначити їх

математичне сподівання, дисперсію, закон розподілу. Недолік використання машинної реалізації імітаційної моделі полягає в тому, що отримане з її допомогою рішення має приватний характер і відповідає конкретним параметрам системи, її початкового стану і зовнішніх впливів. Перевага полягає в можливості дослідження складних систем.

1.5 Області застосування комп'ютерних моделей

Удосконалення інформаційних технологій зумовило використання комп'ютерів практично у всіх сферах діяльності людини. Розвиток наукових теорій припускає висунення основних принципів, побудову математичної моделі об'єкта пізнання, отримання з неї наслідків, які можуть бути співставленні з результатами експерименту. Використання ЕОМ дозволяє, виходячи з математичних рівнянь, розрахувати поведінку досліджуваної системи в тих чи інших умовах. Часто це єдиний спосіб отримання наслідків з математичної моделі. Наприклад, розглянемо задачу про рух трьох або більше частинок, що взаємодіють один з одним, яка актуальна при дослідженні руху планет, астероїдів та інших небесних тіл. У загальному випадку вона складна і не має аналітичного рішення, і лише використання методу комп'ютерного моделювання дозволяє розрахувати стан системи в наступні моменти часу.

Удосконалення обчислювальної техніки, поява ЕОМ, що дозволяє швидко і досить точно здійснювати обчислення за заданою програмою, ознаменувало якісний стрибок на шляху розвитку науки. На перший погляд здається, що винахід обчислювальних машин не може безпосередньо впливати на процес пізнання навколишнього світу. Однак це не так: рішення сучасних завдань вимагає створення комп'ютерних моделей, проведення величезної кількості обчислень, що стало можливим лише після появи електронно-обчислювальних машин, здатних виконувати мільйони операцій в секунду. Обчислення здійснюються автоматично, відповідно до заданого алгоритму і не вимагають втручання людини. Якщо ЕОМ відноситься до технічної бази проведення

обчислювального експерименту, то її теоретичну основу складають прикладна математика, чисельні методи рішення систем рівнянь.

Із сучасних вчених вагомий внесок у розвиток комп'ютерного моделювання зробив академік Самарскій А.А. [12], основоположник методології обчислювального експерименту в фізиці. Саме їм була запропонована знаменита тріада «модель – алгоритм – програма» і розроблена технологія комп'ютерного моделювання, що успішно використовується для вивчення фізичних явищ. Одним з перших видатних результатів комп'ютерного експерименту в фізиці є відкриття в 1968 році температурного струмового шару в плазмі, що створюється в МГД-генераторах (ефект Т-шару). В даний час обчислювальний експеримент використовується для виконання досліджень в наступних напрямках [13]:

- розрахунок ядерних реакцій;
- рішення задач небесної механіки, астрономії і космонавтики;
- вивчення глобальних явищ на Землі, моделювання погоди, клімату, дослідження екологічних проблем, глобального потепління, наслідків ядерного конфлікту і т.д .;
- рішення задач механіки суцільних середовищ, зокрема, гідродинаміки;
- комп'ютерне моделювання різних технологічних процесів;
- розрахунок хімічних реакцій і біологічних процесів, розвиток хімічної і біологічної технології;
- соціологічні дослідження, зокрема, моделювання виборів, голосування, поширення відомостей, зміна громадської думки, військових дій;
- розрахунок і прогнозування демографічної ситуації в країні і світі;
- імітаційне моделювання роботи різних технічних, зокрема, електронних пристроїв;
- економічні дослідження розвитку підприємства, галузі, країни.

Отже, комп'ютерне моделювання є надзвичайно важливим та потрібним у наш час у багатьох галузях та напрямках.

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

Як було наведено раніше, ігрові моделі є однією з груп комп'ютерного моделювання, згрупованих за цілями моделювання. Спочатку теорія ігор використовувалася для опису і моделювання поведінки людських популяцій. Деякі дослідники вважають, що за допомогою визначення рівноваги в відповідних іграх вони можуть передбачити поведінку людських популяцій в ситуації реальної конфронтації. Такий підхід до теорії ігор останнім часом піддається критиці з кількох причин [14].

Припущення, що використовуються при моделюванні, найчастіше порушуються в реальному житті. Дослідники можуть припускати, що гравці обирають поведінки, максимізуючи їх сумарну вигоду (модель економічної людини), однак на практиці людську поведінку часто не відповідає цій передумові. Існує безліч пояснень цього феномена – нераціональність, моделювання обговорення, і навіть різні мотиви гравців (включаючи альтруїзм). Автори теоретико-ігрових моделей заперечують це, кажучи, що їх припущення аналогічні подібним припущенням у фізиці. Тому навіть якщо їх припущення не завжди виконуються, теорія ігор може використовуватися як розумна ідеальна модель, за аналогією з такими ж моделями у фізиці.

В теорії ігор рівновагою Неша у грі з двома чи більше гравцями називається сукупність стратегій або дій, згідно з якими кожен учасник реалізує оптимальну стратегію, передбачаючи дії суперників [15]. Це така сукупність стратегій та виграшів, при якій жоден із учасників не може збільшити виграш, змінивши вибір стратегії в односторонньому порядку, коли інші учасники не змінюють свого вибору [16].

Однак питання про те, як ці популяції приходять до рівноваги Неша, залишається відкритим. Деякі дослідники в пошуках відповіді на це питання переключилися на вивчення еволюційної теорії ігор. Моделі еволюційної теорії ігор припускають обмежену раціональність або нераціональність гравців. Незважаючи на назву, еволюційна теорія ігор займається не тільки і не стільки

питаннями природного відбору біологічних видів. Цей розділ теорії ігор вивчає моделі біологічної та культурної еволюції, а також моделі процесу навчання.

З іншого боку, багато дослідників розглядають теорію ігор не як інструмент передбачення поведінки, але як інструмент аналізу ситуацій з метою виявлення найкращого поведінки для раціонального гравця [17]. Оскільки рівновага Неша включає стратегії, що є найкращим відгуком на поведінку іншого гравця, використання концепції рівноваги Неша для вибору поведінки виглядає цілком обґрунтованим. Однак, і таке використання теоретико-ігрових моделей зазнало критики. В деяких випадках гравцеві вигідно обрати стратегію, що не входить в рівновагу, якщо він очікує, що інші гравці також не дотримуватимуться рівноважним стратегіям.

Отже, питання про вибір стратегії в ігровому процесі залишається відкритим на даний час. Як зазначалося раніше, комп'ютерне моделювання в контексті теорії ігор може допомогти людям обрати правильну стратегію в будь-якій справі: військові дії, політика, економіка, спортивні та комп'ютерні ігри і так далі. Взагалі, можна перелічити наступні ознаки гри:

– учасники та їх стратегії;
– логіка поведінки учасників;
– невідповідність інтересів учасників;
– взаємопов'язаність поведінки учасників;
– наявність правил гри, що знають усі учасники.

Дуже корисним міг би бути інструмент, що допоміг би моделювати будь-яку ситуацію, задавати правила гри та логіку учасників. Основною задачею, що необхідно виконати перед розробкою такого інструменту є порівняння способів комп'ютерного моделювання та обрання найкращого з них залежно від цілей моделювання. Було виділено такі основні види комп'ютерного моделювання:

– фізичне моделювання;
– динамічне або чисельне моделювання;
– імітаційне моделювання;

– статистичне моделювання;

– інформаційне моделювання;

– моделювання знань.

Фізичне моделювання допомагає перетворити фізичні сигнали на комп’ютерні і передати їх в існуючу модель з логікою емульованого явища. Цей спосіб моделювання надто залежить від предметної області і не дозволить створити єдиний узагальнений інструмент моделювання ігрової ситуації. Отже, цей спосіб не задовольняє поставленій меті.

Динамічне моделювання добре підходить для систем, де чітко простежуються закони, що можна описати математичними формулами та рівняннями. Програма, що емулюватиме чисельну модель, буде вимагати від користувача вводу математичних рівнянь, що описують систему. Після цього вона повинна буде вивести результат розв’язання рівняння. Така система потребуватиме первинної обробки користувачем, тобто створення системи рівнянь перед внесенням їх у програму. Цей підхід потребує додаткових наукових знань у користувача, отже цільова аудиторія такого додатку значно зменшується. Окрім того, не кожний ігровий процес можна легко завдати за допомогою системи рівнянь. Вона може бути дуже великою і громіздкою, отже потребуватиме великої кількості часу. Плюсом такої моделі є точність, мінусами є необхідність первинної обробки людиною та складність системи. Програма, що реалізує таку модель повинна лише вирішувати системи рівнянь. Найпоширенішою програмою, що задовольняє таким цілям є Wolfram.

Імітаційне моделювання полягає в описі правил та логіки дій гравців під час ігрового процесу. Отже, користувачу необхідно буде завдати в програмі логіку ігрового процесу за допомогою візуального конструктора. Людина не повинна буде мати знання в додаткових сферах. Вона лише матиме знати правила гри та матиме завдати основну логіку дій гравців. Після запуску емуляції процесу можна буде побачити результат вибору тієї чи іншої тактики. Мінусом цього методу є велика громіздкість логіки, а отже великий час на введення даних від користувача. Іншим мінусом є складність анімації логіки гри. Розробити

універсальний спосіб описання анімації логіки користувачем є дуже складною задачею, бо існує надто велика кількість дій, що відображається по-різному. Отже, анімація повинна буде вже запрограмована в системі, що накладає обмеження на предметну область використання програми.

Задачею статистичного моделювання є збір і порівняння результатів емуляції в залежності від вхідних даних. Маючи велику базу знань, програми, що реалізують цю модель можуть повернути вірогідність різноманітних результатів. Використовуючи ці результати можна створити статистику по різним параметрам системи. Головним мінусом цього підходу є потреба в великій базі знань. Також користувач повинен розуміти, які ознаки він може використовувати задля досягнення найточнішого результату. Плюсом є проста робота з програмою, що потребує лише запуску.

Інформаційна модель лише дозволяє за допомогою комп’ютерних засобів наглядно змоделювати предмет чи явище. Вона не містить в собі ніякої логіки, тобто не змінюється з часом. Отже вона не підходить для моделювання ігрових процесів.

Модель моделювання знань схожа на імітаційну модель. Вона також моделює логіку гравців та правила гри. З іншого боку рішення цієї моделі можуть залежати не від запрограмованих правил а від навчання на таких же ситуаціях, що мали місце в минулому. Ця частина моделювання знань схожа на статистичну модель.

3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

Враховуючи аналіз існуючих способів моделювання комп'ютерних систем в контексті створення узагальнюючого конструктора логіки ігор можна виділити дві основні моделі, що необхідно порівняти: імітаційне моделювання та статистичне моделювання.

Закон функціонування має наступний вигляд:

$$Y(t) = F(X, V, H, t) \quad (1)$$

де $X = (x_1, x_2, \dots, x_n)$, – вектор вхідних впливів (стимулів);

$Y = (y_1, y_2, \dots, y_m)$ – вектор вихідних сигналів (відгуків, реакцій);

$V = (v_1, v_2, \dots, v_k)$ – вектор впливів зовнішнього середовища;

$H = (h_1, h_2, \dots, h_l)$ – вектор власних параметрів системи, t – час.

При моделюванні ігрових ситуацій не будемо враховувати вплив зовнішнього середовища. Якщо будуть якісь зовнішні параметри, що впливають на хід гри, будемо вважати їх вхідними параметрами. Задачею роботи є створення узагальненого конструктора ігрових ситуацій, тому система не може мати ніяких специфічних для предметної області параметрів. Окрім того, статистичний спосіб моделювання не залежить від часу. Від залежить лише від вхідних даних та історичних даних.

Отже, для імітаційної моделі формула закону функціонування має наступний вигляд: $Y(t) = F(X, t)$, а для статистичної моделі – $Y = F(X)$, де F – функція, що генерується на основі великої кількості історичних даних.

Задачею даного дослідження є порівняння цих функцій за допомогою програмного забезпечення для моделювання ігрових ситуацій. Основними характеристиками для порівняння є:

- швидкість роботи;
- складність зміни моделі;
- точність моделі.

Статистична модель дозволяє передбачати результат ігрової ситуації на основі великої кількості історичних даних. Отже, в формулі $Y = F(X)$ для статистичного методу моделювання F – модель, що була згенерована на цих даних. Для кожної ситуації у нас є набір параметрів, що характеризують систему. Позначимо їх вектором $X = (x_1, x_2, \dots, x_n)$, де n – це кількість параметрів моделі. Кожному набору параметрів відповідає значення Y. Позначимо набір історичних даних як $X_h = (X_1, X_2, \dots, X_m)$, а набір значень $Y_h = (Y_1, Y_2, \dots, Y_m)$, де m – кількість наборів даних. Використовуючи ці набори ми можемо побудувати функцію F, що буде максимально близько задовільняти відношенню $Y_i \approx F(X_i)$. Отримавши таку функцію ми зможемо прогнозувати значення моделі для нових вхідних даних. Цей аналіз називається регресійним [18].

Існують різноманітні регресійні методи. Об’єктом дослідження є ігрові ситуації, отже найчастішими результатами є виграш чи програш гравця. Тобто залежна змінна Y для формули моделі $Y = F(X)$ є категорійною. В таких випадках застосовують логістичну регресію [19]. Логістична регресія має наступний вигляд:

$$F(X) = \frac{1}{1 + e^{-\theta X}} \tag{2}$$

де $\theta = (\theta_0, \theta_1, \dots, \theta_n)$ – коефіцієнти моделі;

$$\theta X = \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n.$$

Тоді значення залежної змінної можна записати наступним чином:

$$Y = \begin{cases} 1, & F(X) \geq 0.5 \\ 0, & F(X) < 0.5 \end{cases} \tag{3}$$

де $F(X)$ – значення логістичної регресії.

Для отримання функції $F(X)$ необхідно мінімізувати функцію:

$$J(\theta) = \sum_{i=1}^{m} (Y_i \cdot \log F(X_i) + \ (1 - Y_i) \cdot \log(1 - F(X_i))) \qquad (4)$$

де $F(X)$ – значення логістичної регресії.

Для мінімізації функції необхідно, щоб похідна від цієї функції по кожній з залежних змінних дорівнювала 0, тобто для кожного $j \in [0, n]$ виконуватиметися рівняння $\frac{\partial}{\partial \theta_j} J(\theta) = 0$.

Програмно таке рівняння можна вирішити двома способами. Точне вирішення рівняння має складну формулу та займає багато часу і ресурсів. Тому програмно використовують інший спосіб.

Він полягає в ініціалізації $\theta_0, \theta_1, \dots, \theta_n$ випадковими числами, та задання випадкового числа α. Після цього для кожного θ_j, де $j \in [0, n]$ виконують наступну операцію:

$$\theta_j = \theta_j - \alpha \sum_{i=1}^{m} (F(X_i) - Y_i) \cdot x_j^i, \qquad (5)$$

де x_j^i – параметр під номером j з набору X_i.

Ця операція повторюється для кожного θ_j доки кожний з них не буде змінюватися.

Зміна даної моделі полягає в зміні усього набору історичних даних. Тобто для додавання нового параметру моделі необхідно його додати в кожний набір X_i, де $i \in [0, m]$. Після цього модель необхідно заново перерахувати. Тобто для зміни моделі необхідно повністю повторити процес створення моделі. Швидкість та точність можливо проаналізувати тільки після програмного впровадження моделі.

Для статистичного методу необхідно розробити настільний додаток, що реалізуватиме алгоритм логістичної регресії та буде виконувати наступні функції:

– навчання моделі на датасеті;
– отримання передбачення навченої моделі для нового набору даних;
– отримання точності навченої моделі;
– отримання часу навчання та часу передбачення моделі.

Основною думкою імітаційної моделі є завдання користувачем логіки гри, що дозволяє симулювати ігровий процес та приймати комп’ютером рішення. Отже, необхідно розробити програмне забезпечення, що матиме наступні функції:

– отримання передбачення моделі виходячи з вхідних даних користувача та правил моделі;
– отримання точності та часу передбачення моделі;
– можливість багаторазової симуляції ігрового процесу;
– візуальне представлення дій об’єктів гри.

Необхідно пам’ятати, що необхідно мати можливість порівняти обидва методи на однакових вхідних даних.

Найпоширенішим способом симуляції ігрового процесу є підхід ECS (Entity-Component-System) [20]. Під час створення гри в програмі є дуже велика кількість типів сутностей, що мають складну ієрархію. Щоб не дублювати логіку часто за допомогою успадкування класів спільну логіку виносять у базовий клас. Але дуже часто така логіка може виникати у класах що зовсім не пов’язані один з одним. Саме в таких випадках допомагає ECS. Уся програма поділяється на три частини: сутності, компоненти та системи.

Компоненти – це об’єкти-контейнери, що не володіють властивостями, а виступають сховищами для компонентів. Компоненти – це блоки даних, що визначають всілякі властивості будь-яких ігрових об’єктів або подій. Всі ці дані, згруповані в контейнери, обробляються логікою, яка існує виключно у вигляді систем – чистих класів з певними методами для виконання.

4 ОПИС РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ

4.1 Опис програмної системи для статистичного методу моделювання

Для розробки програмної реалізації статистичного методу моделювання було використано платформу .NET [21], а саме технологію WPF. Вона є універсальною платформою розробки з відкритим кодом, яку підтримує корпорація Майкрософт і товариство .NET на сайті GitHub.

Середою розробки була обрана Microsoft Visual Studio 2017 як найпоширеніша та найбільш зручна середа розробки для цієї платформи.

Зазвичай для вирішення проблем пов’язаних з машинним навчанням використовується язик програмування Python. На ньому вже існує велика кількість різноманітних фреймворків. Але через свою швидкість була обрана саме платформа .NET та язик програмування C#.

Існує багато реалізацій для логістичної регресії. Було вирішено порівняти власну реалізацію логістичної регресії з найпоширенішою реалізацією на платформі .NET, що є у відкритому доступі. Вона міститься у пакеті Accord.NET [22].

Отже, архітектура програмної реалізації вигляд зображений на рисунку 4.1. Як вказано на рисунку, програма поділена на 6 модулів:

- GameSituationsStatistics – модуль, що є точкою входу в програму;
- GameSituationsStatistics.Core – модуль з основну бізнес-логікою програми;
- GameSituationsStatistics.DataLoader – модуль для завантаження даних з файлів різних типів у пам’ять програми;
- GameSituationsStatistics.IoC – модуль інверсії залежностей програми;
- GameSituationsStatistics.Models – модуль з основними моделями програми, що використовуються на усіх шарах програми;
- GameSituationsStatistics.Prediction – модуль, що зберігає у собі усю математичну логіку програми.

Розберемо більш детально кожний з модулей. Розпочнемо з GameSituationsStatistics.IoC. Інверсія управління [23] – важливий принцип

програмування, що використовується для зменшення зчеплення в комп'ютерних програмах.

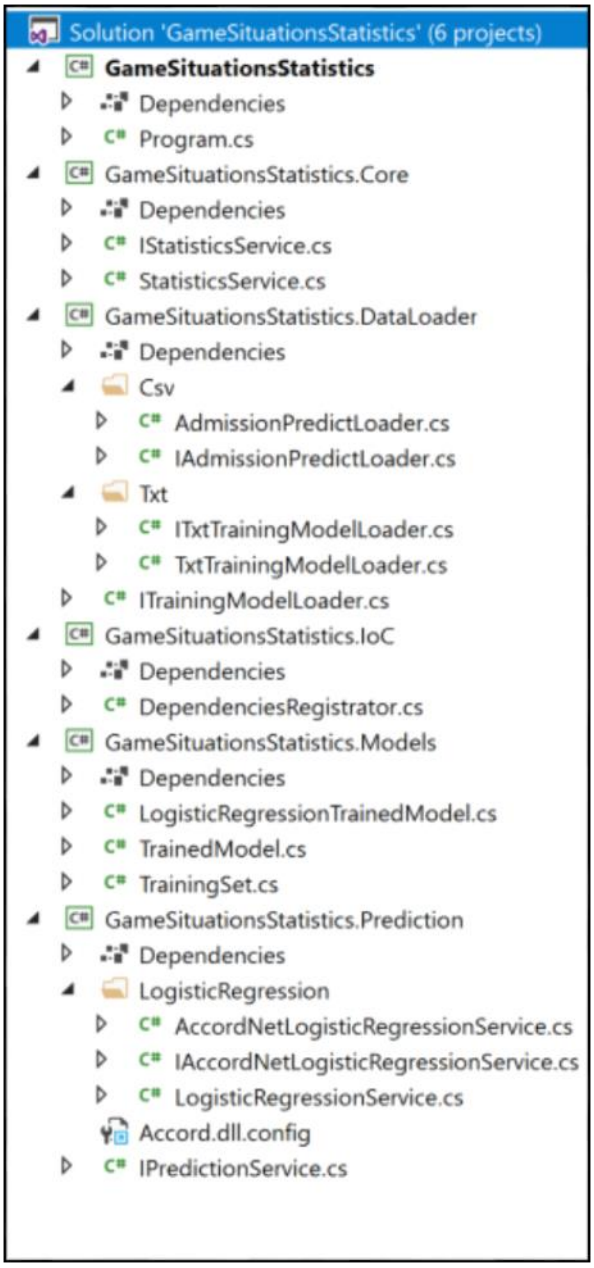

Рисунок 4.1 – Загальна архітектура програмної реалізації статистичного методу моделювання

Однією з реалізацій інверсії управління є впровадження залежностей. Впровадження залежностей використовується в багатьох фреймворках, які називаються IoC-контейнерами. Якщо порівняти з більш низькорівневими

технологіями, IoC-контейнер – це компоновщик, який збирає не об'єктні файли, а екземпляри класу під час виконання програми.

GameSituationsStatistics.IoC має лише один файл, що регіструє в собі усі залежності. Код цього файлу наведений на рисунку 4.2. Реєстрація залежностей реалізована за допомогою бібліотеки Autofac.

```
1 reference
public class DependenciesRegistrator
{
    1 reference
    public static IContainer Configure()
    {
        var builder = new ContainerBuilder();

        builder.RegisterType<AdmissionPredictLoader>().As<IAdmissionPredictLoader>();
        builder.RegisterType<TxtTrainingModelLoader>().As<ITxtTrainingModelLoader>();
        builder.RegisterType<AccordNetLogisticRegressionService>().As<IAccordNetLogisticRegressionService>();
        builder.RegisterType<LogisticRegressionService>().As<IPredictionService>();
        builder.RegisterType<StatisticsService>().As<IStatisticsService>();

        return builder.Build();
    }
}
```

Рисунок 4.2 – Інверсія залежностей в програмі

Розглянемо модуль GameSituationsStatistics.Models. В ньому зберігаються 3 моделі що використовуються програмою:

– TrainingSet – навчальна модель, що складається з набору числових параметрів системи та набору значень, що відповідають цим наборам;
– TrainedModel – базовий клас навченої моделі, що має код для коефіцієнтів вихідної функції, точності моделі у відсотках та функції нормалізації даних перед передбаченням значення для нового набору даних;
– LogisticRegressionTrainedModel – клас для логістичної регресії, що успадковує клас TrainedModel.

Перед навчанням та передбаченням нового значення набір вхідних параметрів необхідно нормалізувати, тобто привести до стану, що легше обробляти. Найчастіше це зміна всіх значень так, щоб вони булі в діапазоні $[0; 1)$ або $[-1; 1)$. Для нормалізації даних в цій програмі використовується підхід minmax. Для кожного параметру набору навчальних даних ми підраховуємо

мінімальне, середнє та максимальне значення. Після цього кожен параметр в кожному наборі навчального датасету перетворюється наступним способом:

$$x_j^i = \frac{x_j^i - avg_j}{max_j - min_j} \tag{6}$$

де x_j^i – параметр під номером j з набору X_i;

avg_j – середнє значення j-го параметру;

max_j – максимальне значення j-го параметру;

min_j – мінімальне значення j-го параметру.

Програмна реалізація підходу зображена на рисунку 4.3.

```
1 reference
private double[] Normalize(double[] x)
{
    if (Averages != null && Ranges != null)
    {
        return x.Select((feature, i) => (feature - Averages[i]) / Ranges[i]).ToArray();
    }

    return x;
}
```

Рисунок 4.3 – Програмна реалізація minmax

Модуль GameSituationsStatistics.DataLoader складається з класів, що завантажують дані з файлів різних типів до пам’яті програми у вигляді моделі TrainingSet, що наведена в модулі GameSituationsStatistics.Model. Окрім цього модуль може перемішувати рядки даних, для більш точного результату. Датасети можуть мати різні формати зберігання даних. Наприклад, це можуть бути текстові файли, кожен набір даних якого починається з наступної строки, а кожен параметр відділяється комою чи крапкою с комою. Також це можуть бути .xls, .csv, файли баз даних та інші.

Модуль GameSituationsStatistics.Prediction складається з набору класів, що генерують модель. LogisticRegressionService генерує власну модель, AccordNetLogisticRegressionService використовує фреймворк Accord.NET для генерації. Код для другої реалізації зображений на рисунку 4.4. Можна побачити, що ми навчаємо модель на 70% усього датасету, після чого перевіряємо точність отриманої моделі на 30%.

```
4 references
public TrainedModel TrainModel(TrainingSet set)
{
    var trainSet = set.X.Take(set.X.Length * 70 / 100).ToArray();
    var testSet = set.X.Skip(trainSet.Length).ToArray();

    var trainSetY = set.Y.Take(trainSet.Length).ToArray();
    var testSetY = set.Y.Skip(trainSet.Length).ToArray();

    var learner = new IterativeReweightedLeastSquares<Accord.Statistics.Models.Regression.LogisticRegression>()
    {
        Tolerance = 1e-4,  // Let's set some convergence parameters
        Iterations = 100,  // maximum number of iterations to perform
        Regularization = 0
    };

    var model = learner.Learn(trainSet, trainSetY);

    var testSetAccuracy = testSet
        .Select(s => GetFunctionResult(model.Coefficients, s) < 0.5 ? 0 : 1)
        .Zip(testSetY, (trainedResult, actualResult) => trainedResult == actualResult)
        .Where(isEqual => isEqual)
        .Count() / (double)testSetY.Length * 100;

    return new LogisticRegressionTrainedModel(model.Coefficients, testSetAccuracy);
}
```

Рисунок 4.4 – Реалізація логістичної регресії використовуючи фреймворк Accord.NET

Розглянемо власну реалізацію логістичної регресії. Спочатку проводиться minmax нормалізація даних. Код для нормалізації даних зображений на рисунку 4.5. Ми паралельно сумуємо усі числа характеристики та знаходимо мінімальне та максимальне значення з набору значень кожної характеристики моделі. Після цього ми маємо можливість отримати середні значення та діапазон значень кожної характеристики. Після нормалізації, як і в випадку з готовою реалізацією логістичної регресії, ми використовуємо лише певний відсоток даних для навчання, а усі інші дані використовуємо для перевірки точності моделі.

Далі необхідно в циклі підраховувати значення вартості функції, послідовно зменшуючи похідні використовуючи випадково обране значення α. На початковій стадії ми встановлюємо коефіцієнти у випадкове значення, наприклад, привласнюємо їм значення одиниць. Після цього описується впроваджується логіка логістичної регресії.

```
for (var j = 0; j < n; j++)
{
    var max = set.X[0][j];
    var min = set.X[0][j];

    for (var i = 0; i < m; i++)
    {
        averages[j] += set.X[i][j];

        if (set.X[i][j] > max)
        {
            max = set.X[i][j];
        }

        if (set.X[i][j] < min)
        {
            min = set.X[i][j];
        }
    }

    averages[j] /= m;
    ranges[j] = max - min;
}

for (var i = 0; i < m; i++)
{
    x[i] = new double[n];

    for (var j = 0; j < n; j++)
    {
        x[i][j] = (set.X[i][j] - averages[j]) / ranges[j];
    }
}
```

Рисунок 4.5 – Minmax нормалізація датасету

Під час реалізація виникла проблема з підрахунком чисел с плаваючою точкою. Як можна побачити на рисунку 4.6, в умові головного циклу стоїть не

строге порівняння старої та нової вартостей функції, а порівняння їх різниці з якимось маленьким числом дуже близьким до нуля. Цього достатньо для отримання точного результату.

Модуль GameSituationsStatistics.Core включає всю бізнес-логіку програми. Він працює з GameSituationsStatistics.DataLoader, LogisticRegressionTrainedModel та GameSituationsStatistics.Prediction, компанує їх та повертає необхідний результат.

```
do
{
    cost = newCost;

    for (var i = 0; i < nextThetas.Length; i++)
    {
        thetas[i] = nextThetas[i];
    }

    var functionDifferences = GetNewFunctionDifferences(thetas, trainSet, trainSetY);

    double sumTheta0 = 0;

    for (var i = 0; i < m; i++)
    {
        sumTheta0 += functionDifferences[i];
    }

    nextThetas[0] = thetas[0] - alpha * sumTheta0;

    for (int j = 0; j < n; j++)
    {
        double sum = 0;

        for (var i = 0; i < m; i++)
        {
            sum += functionDifferences[i] * trainSet[i][j];
        }

        nextThetas[j + 1] = thetas[j + 1] - alpha * sum;
    }

    newCost = GetCostFunctionValue(nextThetas, trainSet, trainSetY);

} while (Math.Abs(cost - newCost) > tolerance);
```

Рисунок 4.6 – Підрахунок коефіцієнтів моделі

Головний модуль програми вирішує усі залежності за допомогою GameSituationsStatistics.IoC та визиває логіку з GameSituationsStatistics.Core.

Також він вимірює час затрачений на логіку та виводить результати моделей (рисунок 4.7). Результати запуску додатку можна побачити на рисунку 4.8.

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        var builder = DependenciesRegistrator.Configure();
        var service = builder.Resolve<IStatisticsService>();

        var trainingSet = service.LoadTrainingSet("Admission_Predict.txt");

        var date1Ticks = DateTime.Now.Ticks;
        var model = service.TrainCustomModel(trainingSet);
        var date2Ticks = DateTime.Now.Ticks;
        var model2 = service.TrainAccordModel(trainingSet);
        var date3Ticks = DateTime.Now.Ticks;

        Console.WriteLine($"Custom model accuracy: {model.Accuracy}, work time ticks: {date2Ticks - date1Ticks}");
        Console.WriteLine($"Accord.NET model accuracy: {model2.Accuracy}, work time ticks: {date3Ticks - date2Ticks}");
    }
}
```

Рисунок 4.7 – Код головного класу програми

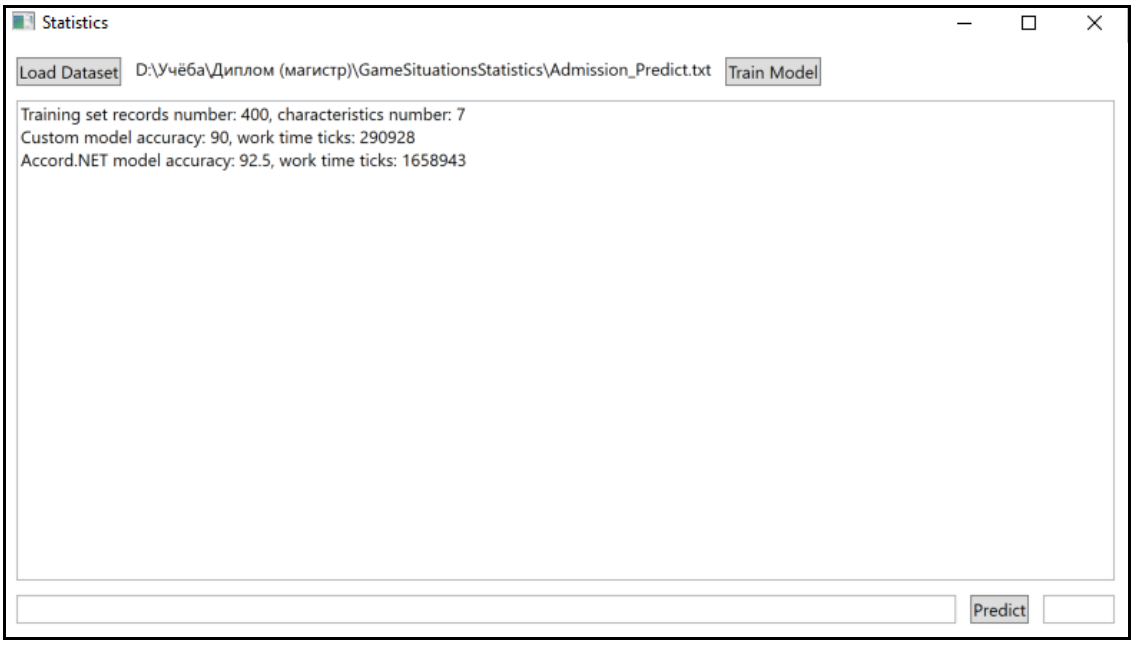

Рисунок 4.8 – Результат роботи програмної реалізації статистичного методу моделювання

У даному випадку вирішувалася задача створення моделі для передбачення вступу до університету. Вхідними параметрами були результати іспитів, сердній бал в школі, різноманітні додаткові заслуги. Вихідним параметром був результат: вступ або відмова у вступі в університет. З рисунку 4.8 можна побачити що

обидві моделі дають більш менш однакову точність, але власна модель працює швидше. Це можливо через наявність якихось оптимізацій у фреймворці, що надають більшої точності.

На рисунку 4.8 можна побачити, що є кнопка для завантаження датасету, кнопка для тренування моделі та вікно для виводу інформації. Після тренування моделі можна передбачати результати нових ситуацій (рисунок 4.9).

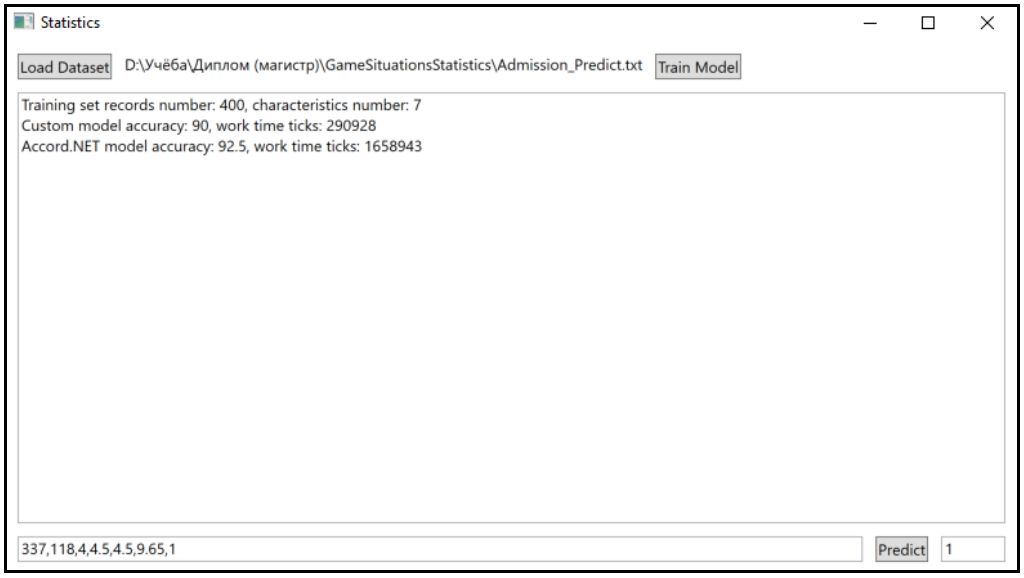

Рисунок 4.9 – Передбачення результату нової ситуації

Для передбачення нової ситуації необхідно ввести усі числові параметри розділені комою та натиснути кнопку Predict. Після цього в текстовому полі в правому нижньому куті програми з’явиться вірогідність успіху чи невдачі для введених вхідних даних.

4.2 Опис програмної системи для імітаційного методу моделювання

Для реалізації програмної системи для імітаційного методу моделювання була також обрана платформа .NET та середа розробки Microsoft Visual Studio 2017.

Задачею для реалізації цього методу моделювання було створення загального конструктора для моделювання ігрових ситуацій. Програмна система для імітаційного методу моделювання являє собою конструктор для ECS системи.

Тобто першою задачею користувача є створення компонентів. Компонентом є звичайна характеристика будь-якого об’єкту, наприклад: швидкість, зріст, колір. Компонент може мати один з наступних примітивних типів: число, рядок, логічне значення (так чи ні).

З цих компонентів користувач повинен завдавати сутності системи. Після цього користувач завдає умову кінця гри. Умова може складатися рекурсивно з багатьох інших умов. Тобто умова складає собою дерево листям якого є прості умови на компоненти сутностей. Для числа це можуть бути наступні умови:

– більше;

– менше;

– дорівнює.

Складні умови являють собою дві умови (складні чи прості) пов’язані логічним оператором. Після завдання умови кінця гри користувач завдає систему, тобто набір команд, що повинні виконуватися доки не спрацює умова кінця гри. В системі було створено два типи команд: команда зміни значення та команда умови. Команда зміни значення змінює значення компонента на інше, що задане користувачем.

Роботу такого загального додатка було вирішено протестувати на грі «хрестики-нулики». Загальна логіка зображена на рисунку 4.10.

Спочатку користувач повинен створити компоненти. В даній предметній області були створені наступні компоненти:

– номер клітинки по горизонталі;

– номер клітинки по вертикалі;

– значення, що каже, чи пуста клітинка;

– значення, що каже, чи стоїть в клітинки хрестик;

– значення, що каже, чи стоїть в клітинки нулик.

Після цього необхідно створити дев’ять пустих клітин з правильними номерами по вертикалі та горизонталі. Також для простоти роботи аналогічно створюються сутності ліній та компонент стану лінії.

Наступним кроком є створення систем команд для дій хрестиків та нуликів. Команди будуть дзеркальні для обох систем, тому розглянемо логіку на прикладі дій хрестиків.

```
0 references
static void Main(string[] args)
{
    Create Components

    Create Cell Entities

    Create Line Entities

    var crossSystem = new GameSystem(new List<Command>()...);

    var circleSystem = new GameSystem(new List<Command>());

    var systems = new List<GameSystem>() { crossSystem, circleSystem };

    BaseCondition endGameCondition = new NumberComponentToNumberEqualCondition(
        GetCellComponent(cellsArray[0], isEmptyComponent),
        0);

    for (var i = 1; i < 9; i++)
    {
        endGameCondition = new AndCondition(endGameCondition, new NumberComponentToNumberEqualCondition(
            GetCellComponent(cellsArray[0], isEmptyComponent),
            0));
    }

    var game = new Game(systems, endGameCondition);
}
```

Рисунок 4.10 – Загальна логіка імітаційної моделі для гри «хрестики-нулики»

Першою командою є перевірка, чи можливо наступним ходом закінчити гру. Тобто необхідно перевірити усі лінії на стан, що відповідає стану з двома хрестиками та пустою клітинкою. Якщо така лінія знайшлася, то необхідно знайти серед трьох клітинок лінії пусту та поставити в неї хрестик. Якщо гру закінчити неможливо, потрібно перевірити, чи може опонент завершити гру наступним ходом. Тобто таким саме чином перевірити усі лінії на наявність лінії зі станом з двома нуликами та пустою клітинкою. Після цього потрібно знайти пусту клітинку в лінії та поставити в неї хрестик. Якщо ці умови не правдиві, то необхідно перевірити, чи є центральна клітинка пустою. Якщо так, то поставити в неї хрестик. Після цього перевіряється, чи є пуста кутова клітинка, що не належить лінії станом із хрестиком, нуликом та пустою клітинкою. Якщо така є,

то ставим хрестик туди. Якщо і ця умова не виконалася, то необхідно знайти лінію станом з одним хрестиком та двома пустими клітинками та поставити хрестик туди. При невиконанні усіх умов ставимо хрестик в першу пусту клітинку.

Як ми бачимо, отримана логіка надто велика. Описана раніше логіка займає 1000 строк коду. Отже, на більш складних іграх, що необхідно буде імітувати, програма буде ще складнішою і складною у використанні. Тому було вирішено створити більш предметний симулятор гри, що матиме логіку вшиту в пам’ять програми і не буде потребувати вводу усіх правил, команд, сутностей та компонентів від користувача.

Для предметно-орієнтованої програми була обрана тема баскетболу, а саме ситуація баскетбольної атаки.

Архітектура нової програми зображена на рисунку 4.11. Вона складається з трьох основних модулів:

– BasketballGameSimulator – UI-частина програми;
– BasketballGameSimulator.BaseEcsLogic – базова логіка ECS;
– BasketballGameSimulator.Core – логіка баскетбольної гри.

Логіка ECS була винесена в окремий модуль щоб абстрагувати її від предметної області баскетболу. Модуль включає:

– Component.cs – клас компоненту, що є абстрактним класом;
– Entity.cs – клас сутності, що складається з набору компонентів;
– System.cs – клас системи, що включає лише один абстрактний метод обробки;
– Game.cs – клас гри, що складається з набору систем, таймера та методів Start, Stop, Run і абстрактного методу IsGameEnded.

Метод Start запускає таймер, Stop його зупиняє. Метод Run запускає одну за одною системи гри з інтервалом таймеру.

Розберемо детальніше логіку баскетбольної гри. Клас баскетбольної гри зображений на рисунку 4.12.

Гра вважається завершеною коли м’яч в стані кидка та його координати дорівнюють координатам центру кільця. Цієї умови достатньо щоб проаналізувати одну баскетбольну атаку.

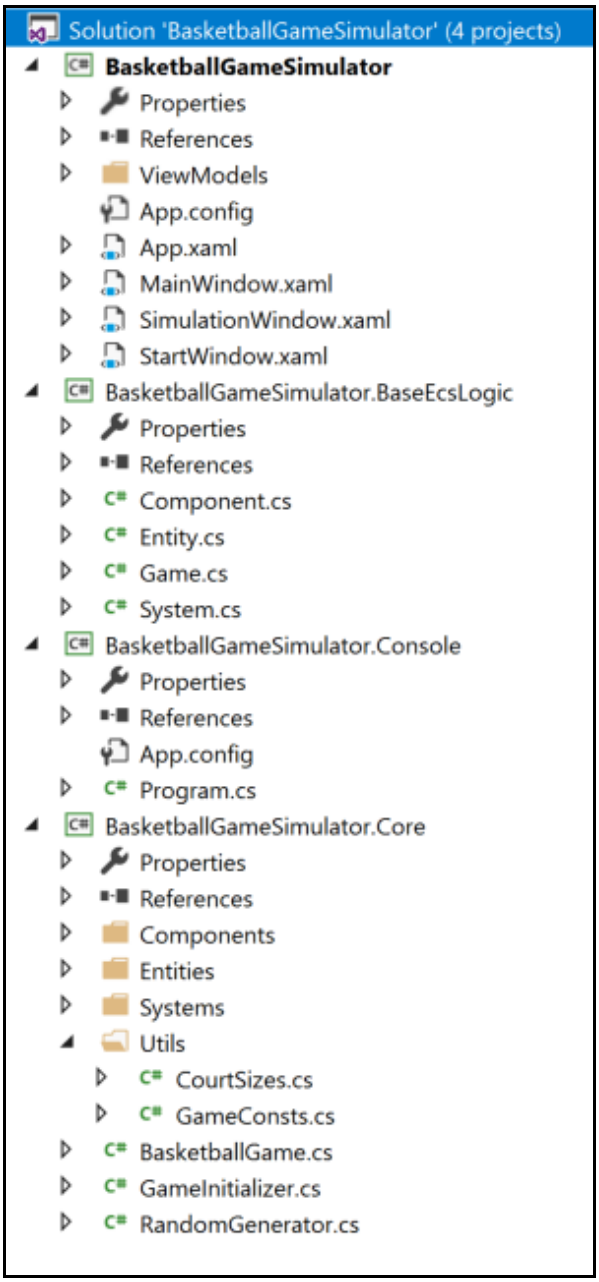

Рисунок 4.11 – Модулі програмного забезпечення з імітації баскетбольної гри

Баскетбольна гра складається з таких сутностей: м’яч, атакуючі гравці, захисники. Усі сутності можуть рухатися з певною швидкістю, отже було вирішено створити компонент для руху, що складається з координат об’єкта, швидкості та координат кінцевої точки.

М’яч може мати один з наступних станів: пас, кидок, дриблінг. Цей стан змінюється системами гри. Захисник має компонент захисту, що складається з посилання на гравця атаки. Атакуючий гравець має компонент кидку. Він включає відсотки влучання з різних дистанцій.

```
11 references
public class BasketballGame : Game
{
    private Ball ball;

    5 references
    public BasketballGame(List<GameSystem> systems, int interval, Ball ball, bool isWpfApp)
        : base(systems, interval, isWpfApp)
    {
        this.ball = ball;
    }

    3 references
    protected override bool IsGameEnded()
    {
        return ball.MovementComponent.X == CourtSizes.RIM_CENTER_X
            && ball.MovementComponent.Y == CourtSizes.RIM_CENTER_Y
            && ball.BucketComponent.IsShotSuccessful.HasValue;
    }
}
```

Рисунок 4.12 – Клас баскетбольної гри імітаційного методу

Уся логіка розташована в системах, що обробляють та змінюють ці сутності та компоненти. Основними системами програми є:

– система захисників;

– система атакуючих гравців;

– система малювання гри;

– система руху.

Програма є WPF-приладом, отже система малювання пов’язана з WPF-елементами. На початку роботи малюється розмітка баскетбольної площадки та баскетбольне кільце. Після цього кожна сутність видаляється та створюється знову для кожної ітерації програми. Зовнішній вигляд графічної частини програми зображений на рисунку 4.13.

Атакуючи гравці зображені колами синього кольору, захисники – колами красного кольору, м’яч – колом помаранчевого кольору.

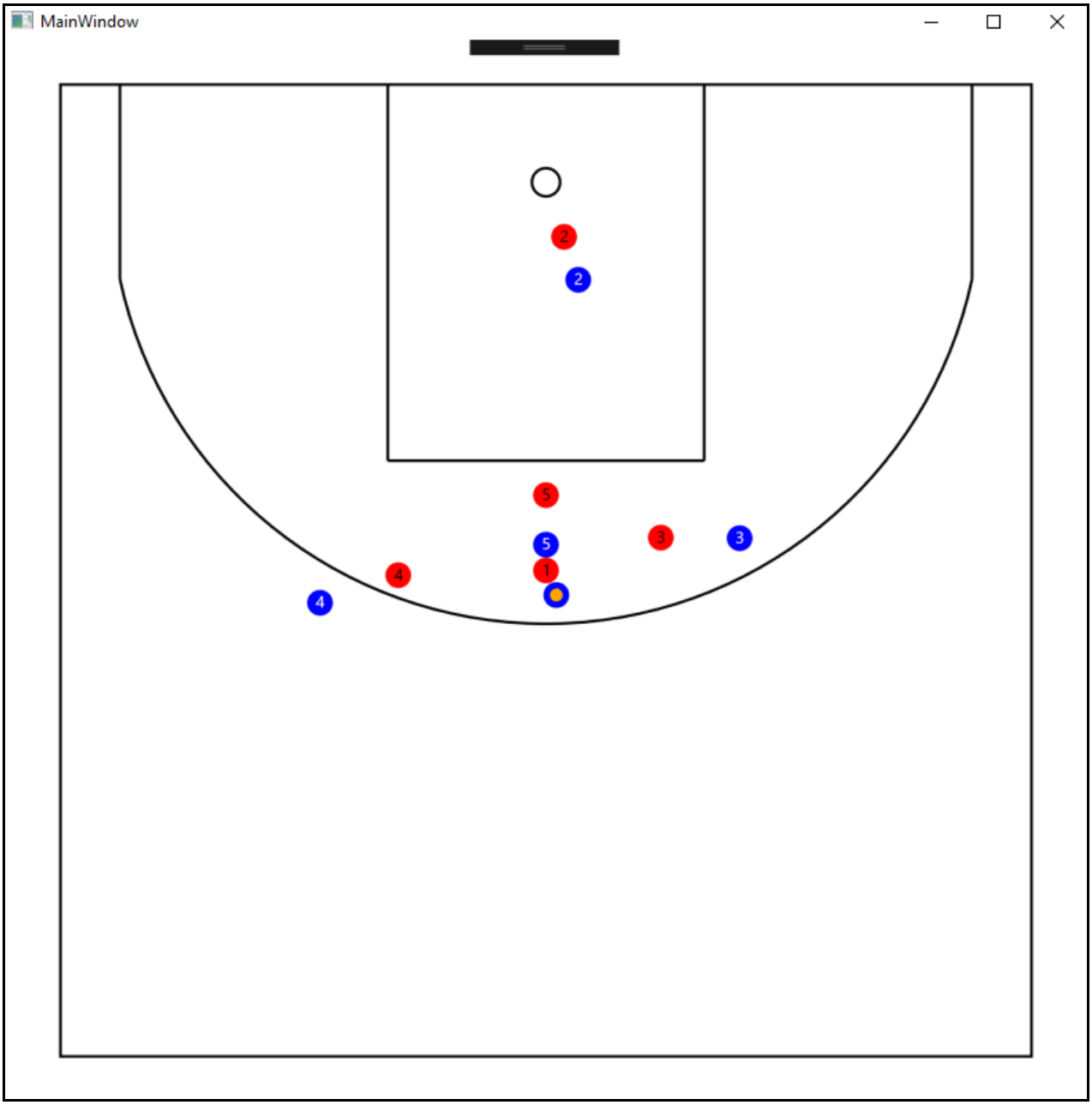

Рисунок 4.13 – Графічна частина імітаційного метода моделювання

Розберемо систему руху. Кожна сутність програми має компонент руху, що складається з поточних координат, кінцевих координат та швидкості. Для розрахунку наступної позиції сутності складається рівняння прямої, що проходить через поточну та кінцеву точку, та рівняння кругу, що має центр в поточній точці та радіус, що дорівнює швидкості сутності.

Тобто вирішується система рівнянь:

$$\begin{cases} y = \dfrac{y_2 - y_1}{x_2 - x_1} \cdot x + y_1 - \dfrac{y_2 - y_1}{x_2 - x_1} \cdot x_1 \\ (y_1 - y)^2 + (x_1 - x)^2 = v^2 \end{cases} \quad (7)$$

де (x_1, y_1) – поточні координати сутності;

(x_2, y_2) – кінцеві координати сутності, v – її швидкість.

Помітимо, що система рівнянь матиме два кореня, але нам потрібна точка, що знаходиться між поточною точкою та кінцевою. Якщо дистанція між поточною та кінцевою точкою менше або дорівнює швидкості сутності, то поточна точка переміщується в кінцеву.

Для руху м’яча немає перешкод, він рухається завжди прямими відрізками. Кожен гравець не може проходити крізь іншого гравця, тому перед кожним рухом перевіряється, чи нема перетину у гравця з іншим гравцем. Якщо є, то перебираються усі можливі напрями переміщення гравця, що не перетинаються з іншими гравцями та обирається найближчий до кінцевої точки.

Логіка захисників знаходиться в в системі DefenseSystem. Існує два основних типи захисту в баскетболі: особистий та зонний захист. Зонний захист, в свою чергу, поділяється ще не декілька підтипів. Розберемо логіку особистого захисту як найпоширеніший тип захисту в баскетболі.

Серед захисників виділяється захисник гравця з м'ячом. Він повинен знаходитися на лінії гравця та кільця на дистанції, що є достатньою, щоб заважати гравцеві з м'ячом здійснити кидок. Отже, для нього вирішується система рівнянь така ж сама (7), що і для наступної точки переміщення об’єкту. Кінцеві координати захисника встановлюються в координати, що є вирішенням системи рівнянь.

Кожен захисник без м’яча повинен знаходитися так, щоб бачити і свого гравця, і м’яч. Окрім того він повинен бути на такої дистанції, щоб змогти допомогти іншому гравцю та у випадку паса встигнути повернутися до свого

гравця. Отже, він, гравець, з яким він захищається, та м’яч повинні створювати трикутник, де кут зору захисника дорівнює середньому боковому зору людини, тобто 150^{o} (рисунок 4.14).

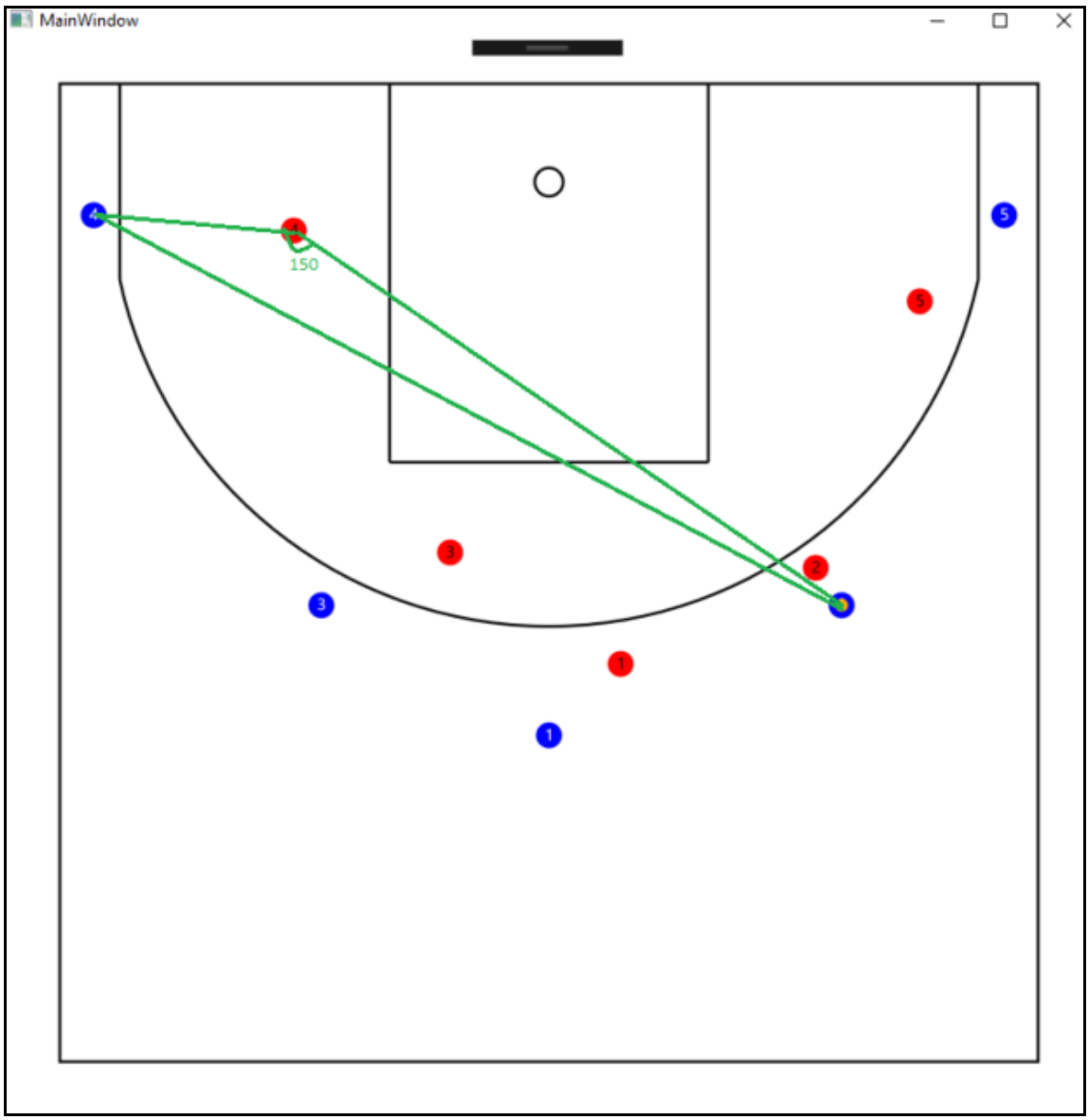

Рисунок 4.14 – Трикутник захисника

Отже, ми маємо координати м’яча, координати гравця, з яким захищається захисник, дозволена дистанція до атакуючого гравця та кут трикутника. Отже в нас є довжина двох сторін та кут трикутника. Використовуючи теорему косинусів можна вирахувати третю сторону. Маючи координати двох точок трикутника та усі довжини сторін можна розрахувати координати третьої точки. Для цього

необхідно скласти систему рівнянь двох кіл на знайти їх перекреслення. Перетин кіл зображений на рисунку 4.15. Помітимо, що у двох кіл є дві точки перетину, нам необхідна та точка, що знаходиться ближче до кільця.

Цей розрахунок координат гравців повинен проходити для кожного захисника, що захищається з гравцем без м’яча.

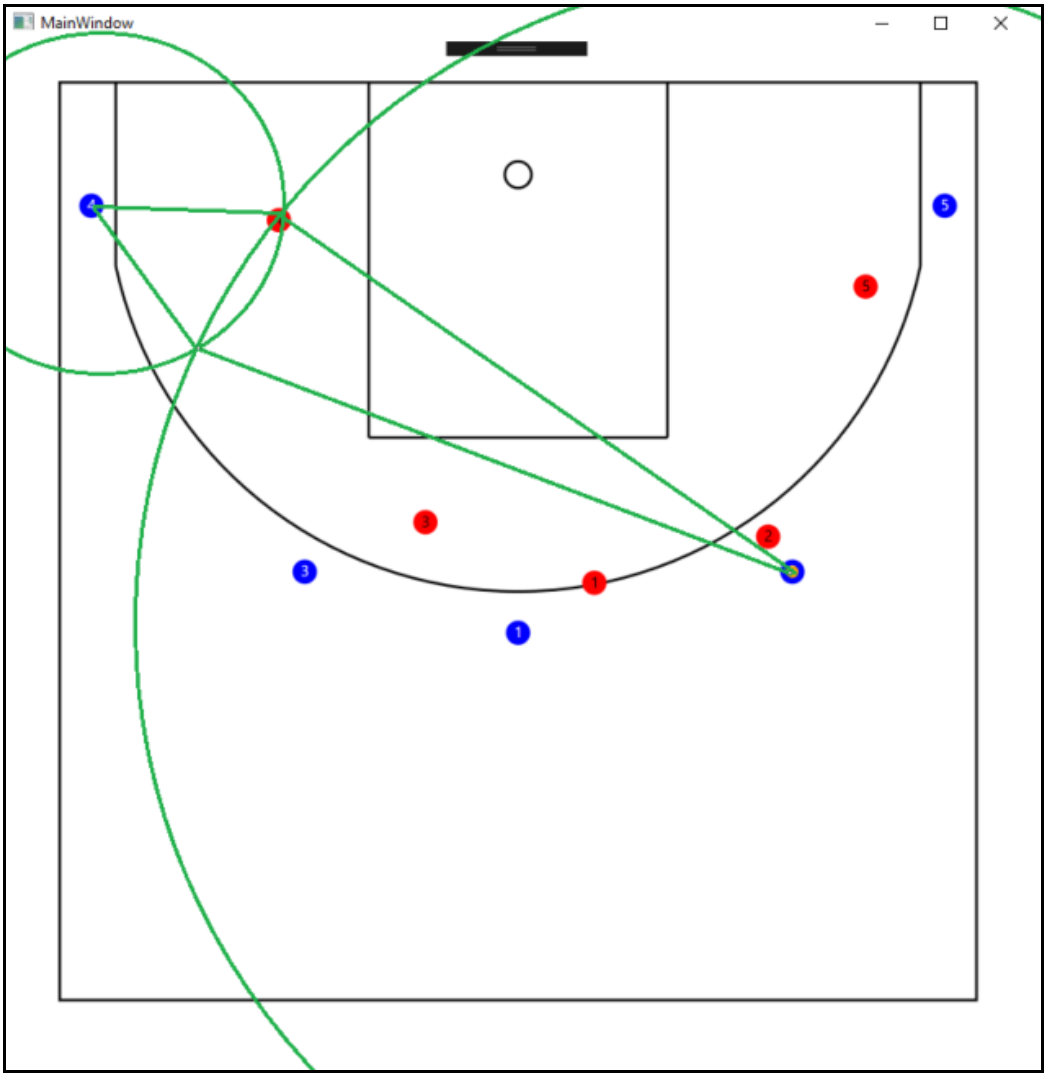

Рисунок 4.15 – Розрахунок координат захисника

Логіка атакуючих гравців знаходиться в системі OffenseSystem. Її можна описати наступним набором правил:

– якщо гравець з м’ячем вільний та його відсоток з поточної точки достатньо високий, виконується кидок;

– якщо інший гравець вільний та відстань від нього до захисника достатньо велика, пас гравцю;
– якщо захисник не заважає гравцю з м’ячем, виконується дриблінг до кошика;
– якщо нічого з переліченого не звершилося, грається комбінація, в програмі вже є логіка для основних баскетбольних комбінацій.

Кожна комбінація складається з кроків. У кожного кроку є логіка та умова завершення кроку. Якщо крок завершується, то система переходить до наступного кроку.

Розберемо UI програми. Початкове вікно програми зображене на рисунку 4.16.

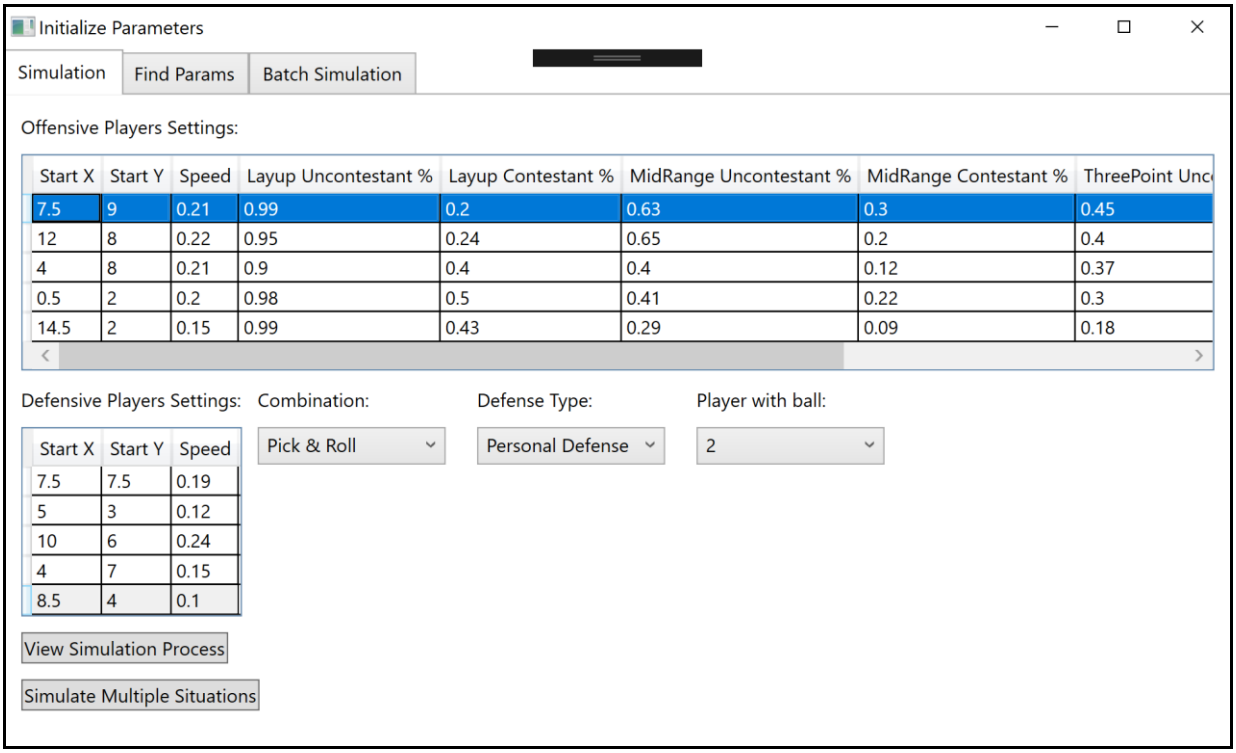

Рисунок 4.16 – Початкове вікно програми імітаційного методу моделювання

На стартовому вікні програми ми можемо побачити три основні розділи: Simulation, Find Params та Batch Simulation. В розділі Simulation користувач може завдати основні вхідні параметри системи, тобто значення усіх компонентів для п’яти гравців нападу та п’яти гравці захисту. Також користувач має можливість

обрати комбінацію для нападу, тип захисту та гравця, що розташовується з м’ячем. Є два типи симуляції процесу: візуальний та багаторазовий. Після натискання на кнопку «View Simulation Process» з’являється вікно симуляції. Після завершення симуляції з’являється вікно, що говорить чи був кидок завершений успішно (рисунок 4.17).

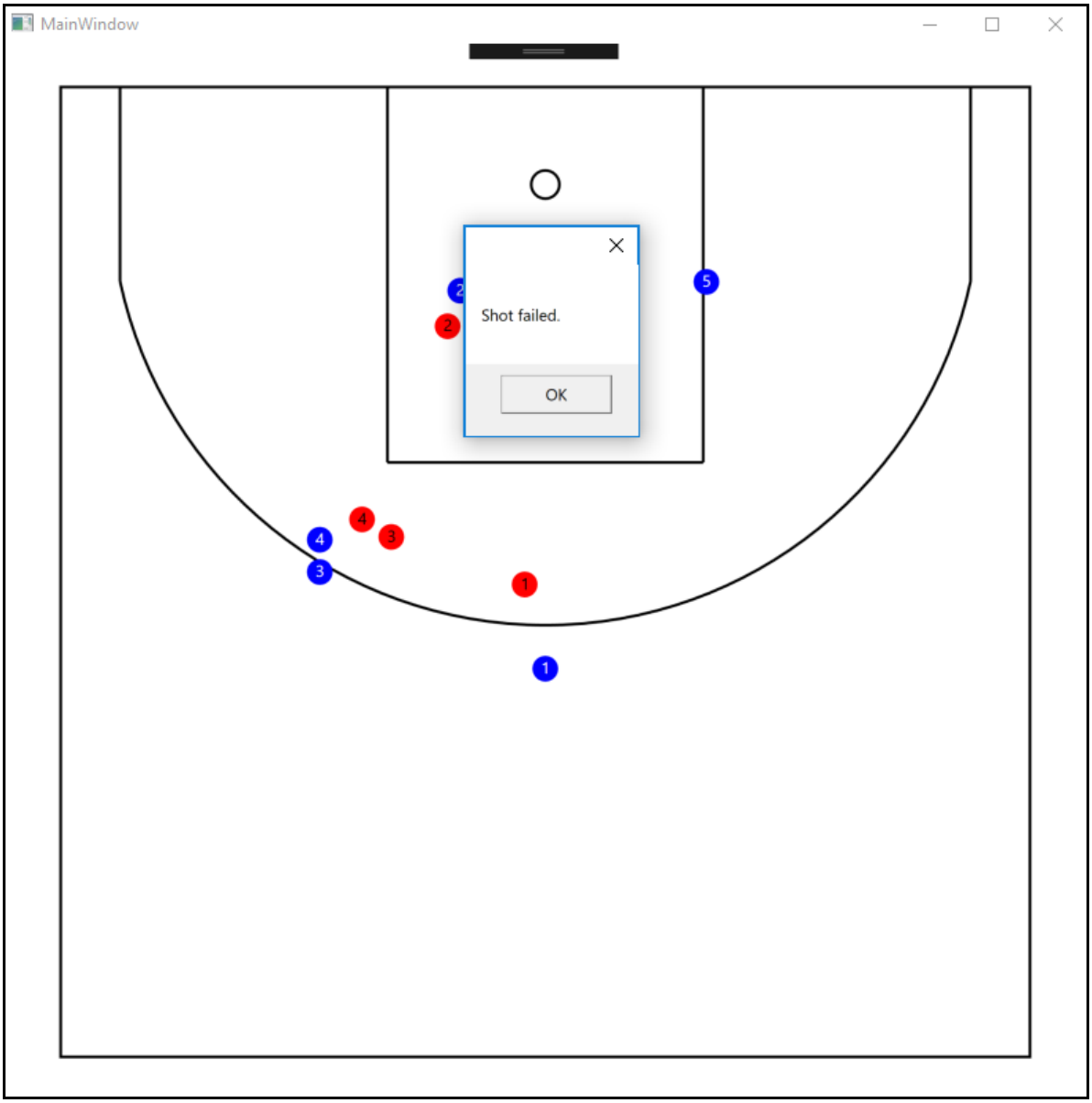

Рисунок 4.17 – Повідомлення про завершення гри

Багаторазовий процес симуляції можна запустити якщо натиснути кнопку «Simulate Multiple Situations». Після цього з’являється вікно, в якому можна

вказати кількість ітерацій для повторної симуляції процесу гри (рисунок 4.18). Процес симуляції запускається після натискання кнопки «Start Simulation».

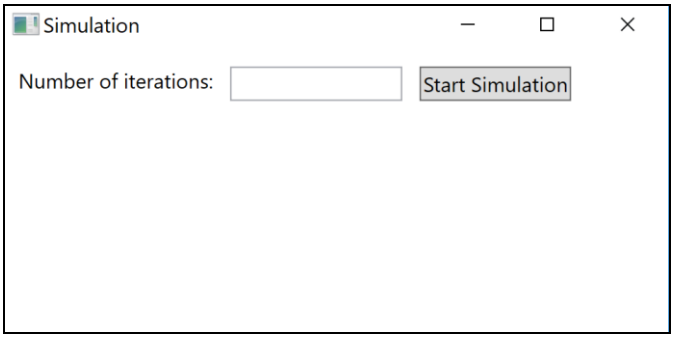

Рисунок 4.18 – Вікно для вводу кількості ітерацій симуляції

Результат багаторазової симуляції зображений на рисунку 4.19.

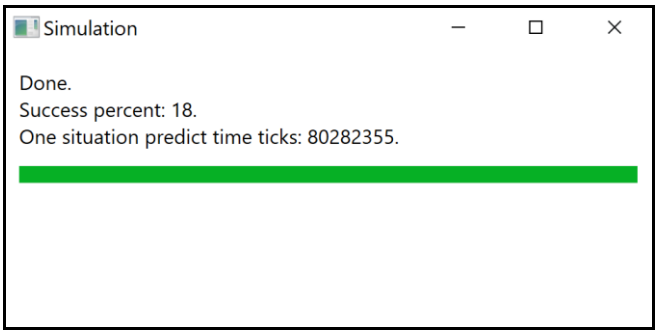

Рисунок 4.19 – Вікно результату багаторазової симуляції

Програма симуляції баскетбольної ситуації має елемент випадковості під час кидка. Тому необхідно запустити програму велику кількість разів, щоб отримати більш-менш точний відсоток успіху.

Розглянемо другу вкладинку «Find Params» на рисунку 4.20. Користувач задає початкові координати, комбінацію в нападі, тип захисту та гравця з м’ячем. Також він має можливість завдати діапазон значень для інших характеристик системи. Окрім цього важливо завдати кількість ітерацій для кожної ситуації та відсоток успіху, що вважається задовільним. Після успішного перебору усіх варіантів звіт з усіма вдалими ситуаціями, сутностями та значеннями їх компонентів зберігаються в файлі. Розглянемо третю вкладку на головній сторінці

програми, «Batch Simulation». Вона має такий самий інтерфейс, як і програма для статистичного методу моделювання (рисунок 4.21).

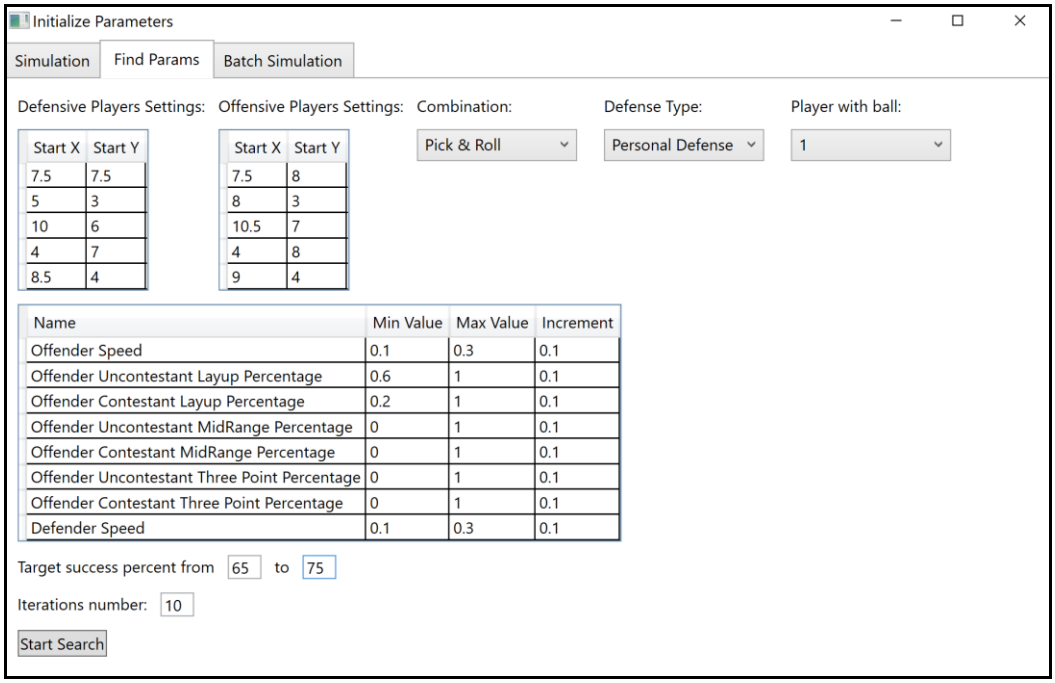

Рисунок 4.20 – Інтерфейс симуляції усіх можливих ситуацій

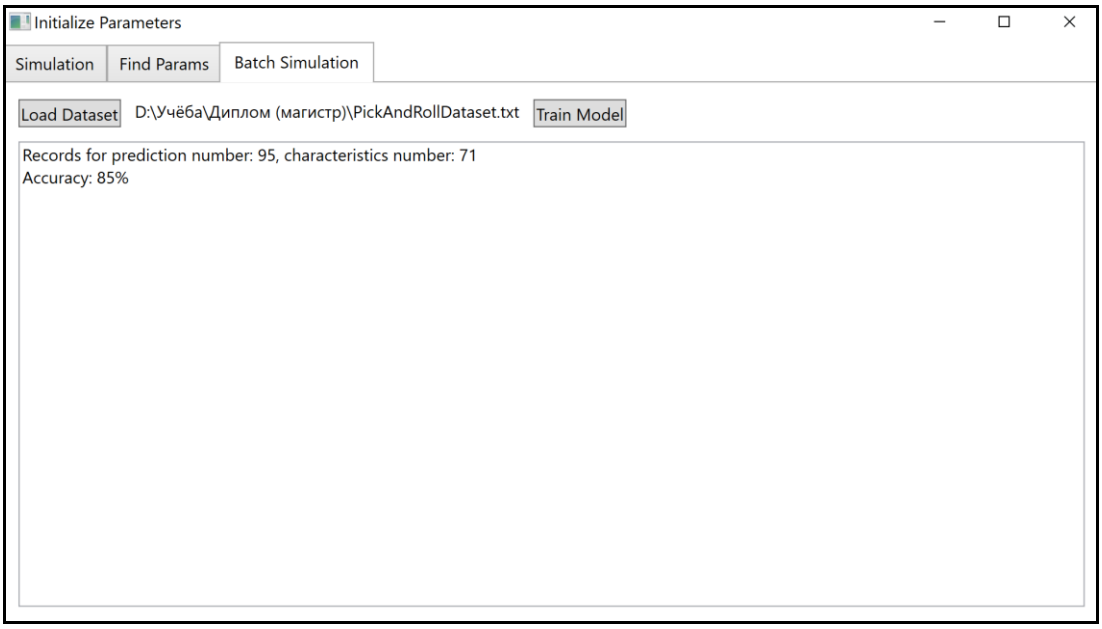

Рисунок 4.21 – Інтерфейс симуляції ситуацій з файлу

Після завантаження даних з текстового файлу та натискання кнопки «Train Model» можна побачити кількість записів в датасеті, кількість характеристик та

шлях до завантаженого файлу. Після моделювання усіх ситуацій можна побачити відсоток точності моделі.

4.3 Порівняння отриманих результатів

Для порівняння імітаційної та статистичної моделі необхідно мати однакові вхідні дані, щоб порівняти точність та швидкість моделей. Розроблена статистична модель ніяк не залежить від предметної області, тому вона не має вимог до датасету. Обов’язковими вхідними даними для розробленої імітаційної моделі є ситуації, що складаються з різних характеристик. Для кожного гравця нападу це:

- швидкість
- стартова позиція;
- відсоток влучання з близької дистанції без перешкод;
- відсоток влучання з близької дистанції з перешкодою;
- відсоток влучання з середньої дистанції без перешкод;
- відсоток влучання з середньої дистанції з перешкодою;
- відсоток влучання з дальньої дистанції без перешкод;
- відсоток влучання з дальньої дистанції з перешкодою.

Кожен гравець захисту має такі характеристики:

- швидкість;
- стартова позиція гравця.

Окрім цього перед створення моделі гри необхідно знати:

- стратегію захисту;
- стратегію нападу;
- номер гравця нападу з м’ячем.

У відкритому доступі не вдалося знайти датасетів, що були схожими на той, що необхідний цій реалізації імітаційного методу. Тому було вирішено створити свій набор даних шляхом просмотру ігор аматорський та професіональних команд. Предметом порівняння стала дуже поширена в баскетболі комбінація «заслін» та її ефективність проти особистого типу захисту. Отже стратегія захисту

та нападу вважається заданою. Таким чином необхідно зібрати датасет, що буде складатися з усіх перелічених характеристик, окрім від стратегій нападу та захисту. Враховуючи, що стартова позиція складається з двох координат, кожен гравець нападу має 9 характеристик, кожен гравець захисту – 3 характеристики. Тобто загалом необхідно 60 характеристик пов’язаних з гравцями та характеристика, що зберігає в собі номер гравця з м’ячем.

Після перегляду великої кількості баскетбольних ситуацій був складений датасет на 89 записів. Його зовнішній вигляд зображений на рисунку 4.22.

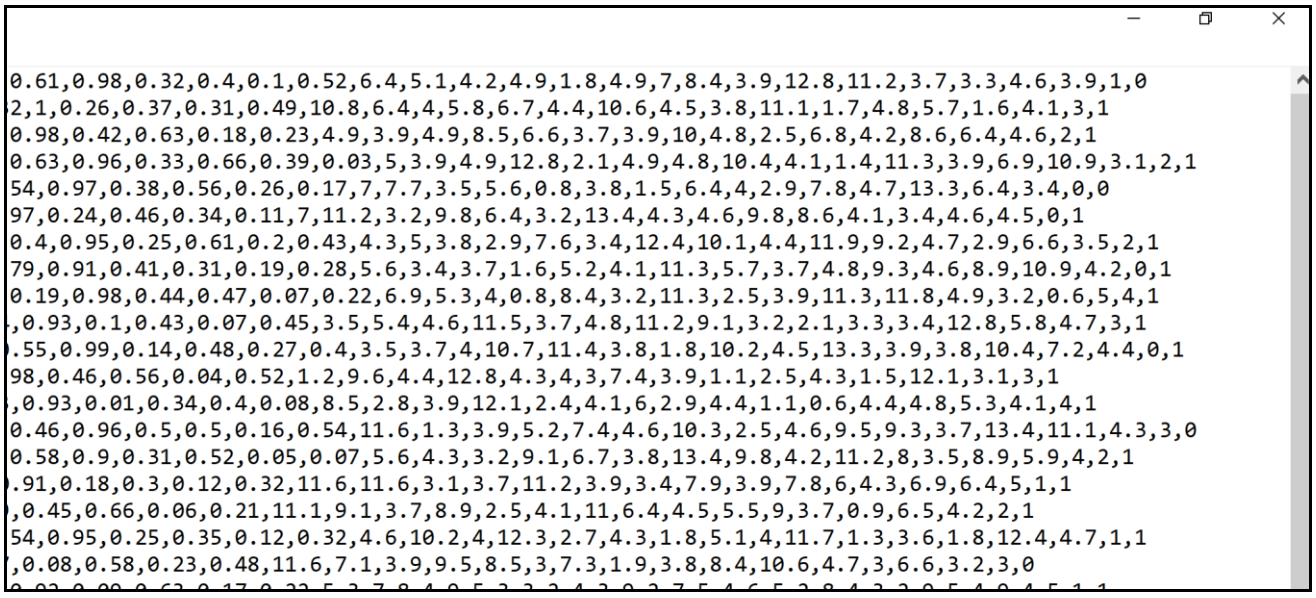

Рисунок 4.22 – Датасет для порівняння моделей

Останнім числом кожного датасету є значення того, чи була атака успішною або ні. Усі характеристики розділені комою. Кожен запис починається з нової строки.

Результати запуску програмної реалізації статистичного методу зображені на рисунку 4.23. Спочатку завантажується датасет, перший рядок виводу інформує про кількість характеристик та наборів даних. Після цього запускається підрахунок двох моделей: самостійної реалізації та реалізації Accord.NET. Як можна побачити з рисунку 4.23, Точність самостійно розробленої моделі дорівнює 74.1%, точність моделі Accord.NET дорівнює 55.6%.

Тренування самостійної реалізації статистичної моделі займає 251055 тіки процесора, тобто 25 мілісекунд. Навчання моделі Accord.NET займає 731543 тіки, тобто 73 мілісекунд.

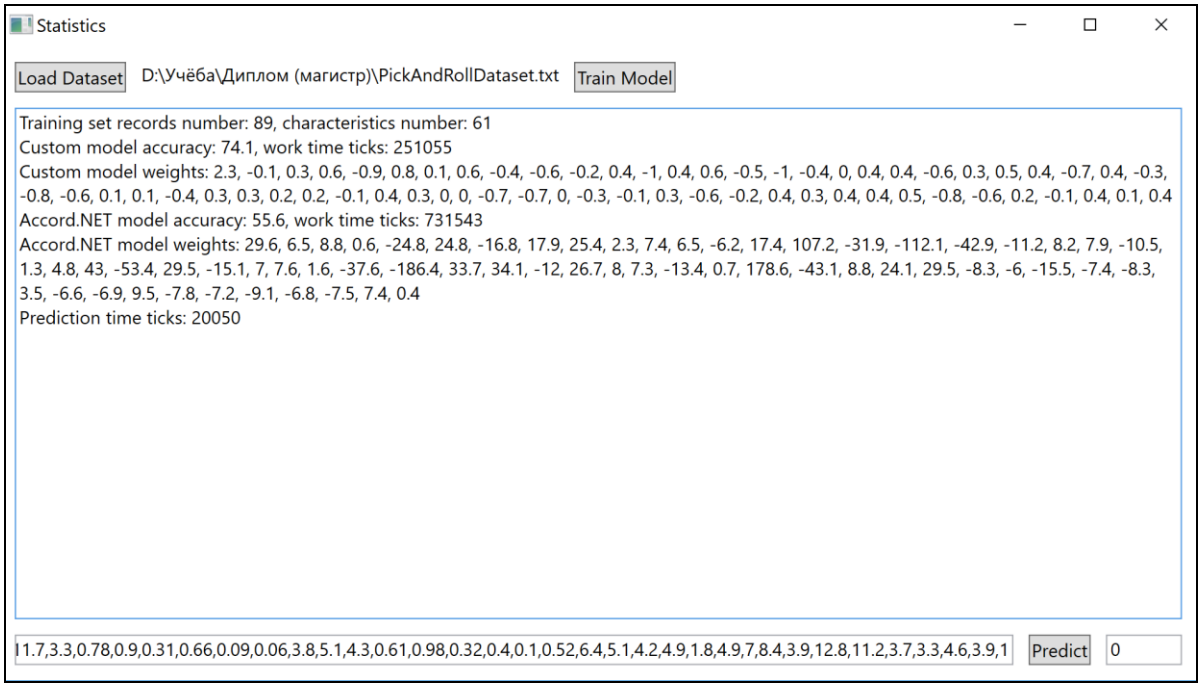

Рисунок 4.23 – Результат роботи програмної реалізації статистичного методу

Різниця в часі не складає великої різниці у відносному порівнянні, тобто різниця не порядкова. Вона може бути зумовлена більш складною логікою для більшої точності моделі. Окрім цих метрік програма також рахує вагу кожної з характеристик моделі. Тобто наша модель повертає 61 коефіцієнт. Кожен з них необхідно брати за модулем для порівняння. Чим більше значення за модулем, тим вагоміша характеристика. В самостійній реалізації найвагомішою характеристикою стала перша характеристика, тобто швидкість першого гравця. Модель Accord.NET вказала, що найвагомішою є 33-я характеристика, тобто відсоток влучання без перешкод четвертого атакуючого гравця з середньої дистанції.

Після навчання моделі було запущене передбачення на новому наборі даних. Передбачення зайняло 20050 тіки, тобто 2 мілісекунди. В результаті був

отриманий результат 0, тобто для нової ситуації модель передбачила негативний результат.

Під час запуску статистичного методу моделювання модель навчалась на 70% даних, тобто на 62 записах. Тестування точності моделі здійснювалося на залишкових 27 записах. Імітаційна модель не потребує записів для навчання. Отже її порівняння необхідно запустити на таких самих 27 записах. Запуск програмної реалізації імітаційного методу моделювання зображений на рисунку 4.24. Як ми бачимо з рисунку, передбачення було зроблено на тих самих 27 записах, що і перевірка статистичного методу.

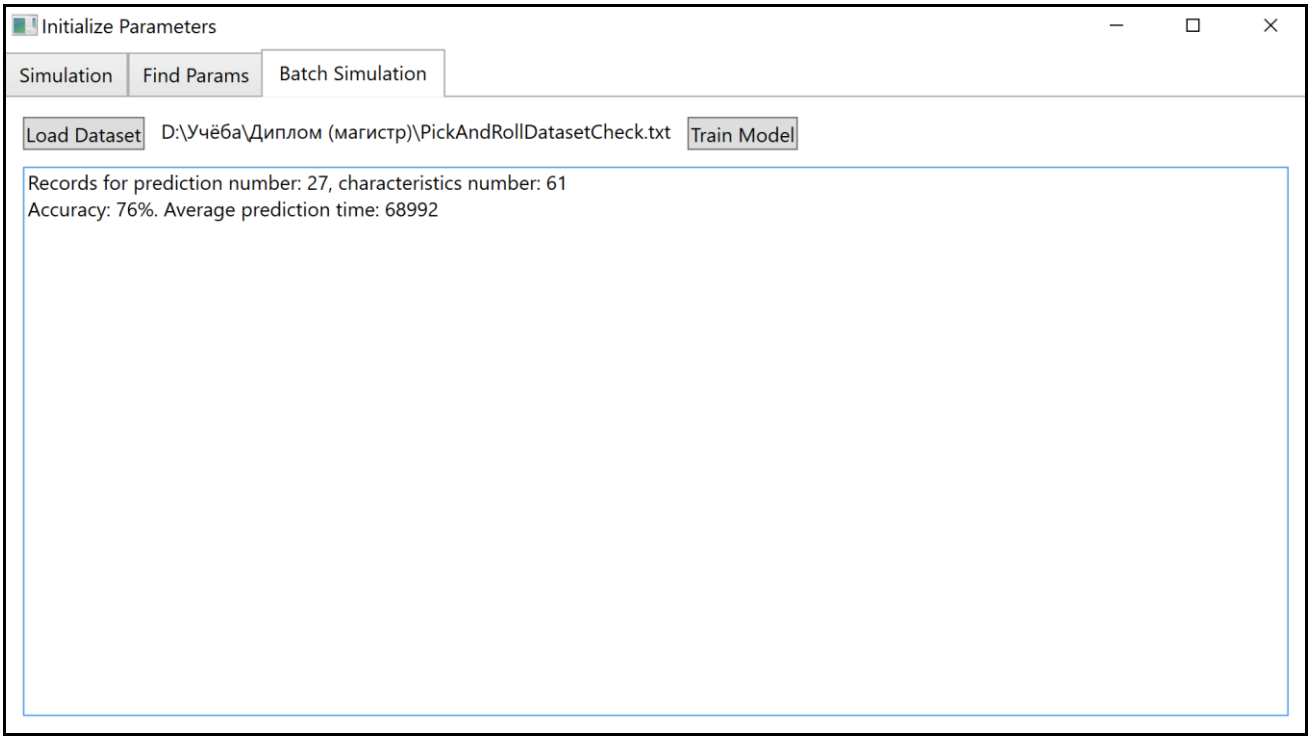

Рисунок 4.24 – Результат роботи програмної реалізації імітаційного методу

Точність моделі складає 76%, середній час передбачення складає 68992 тиків процесора, тобто 7 мілісекунд. Помітимо, що точність обох методів на цьому наборі даних більш-менш однакова. Швидкість передбачення менша у статистичного методу. Це можна пояснити відсутністю циклів та складних операцій в системі передбачення статистичної моделі.

Складність навчання статистичної моделі можна оцінити як $O(mnk)$. Тут m – кількість наборів даних в датасеті, n – кількість характеристик набору, k – випадкова величина, що залежить від правильного вибору коефіцієнта α та початкових $\theta = (\theta_0, \theta_1, \dots, \theta_n)$. Тоді складність передбачення значення для нового набору оцінюється як $O(n)$.

Складність імітаційної моделі залежить від її реалізації. Для поточної реалізації можна окремо підрахувати складність кожної системи. Для системи руху складність складає $O(pl)$, де p – кількість гравців, l – кількість напрямів для руху. Системи захисту та нападу залежать лише від кількості гравців та набору правил. Отже, складності обох систем можна оцінити як $O(pR)$, де p – кількість гравців, R – набір правил.

Порівняння моделей можна побачити у таблиці 4.1.

Таблиця 4.1 – Порівняння методів формалізації баскетбольної ситуації

	Імітаційний метод	Статистичний метод
Необхідність навчання	ні	так
Необхідність датасету	ні	так
Необхідність ретельного підбору характеристик моделі	ні	так
Необхідність завдання правил поведінки об’єктів	так	ні
Швидкість передбачення	$O(R)$	$O(n)$
Залежність алгоритму від предметної області	так	ні
Точність (для обраної предметної області)	76%	74%

Великою перевагою статистичного методу є його незалежність від предметної області. Недоліком є необхідність навчання, необхідність у великому

датасеті для точного результату та необхідність в ретельному підборі характеристик моделі. Імітаційний метод не потребує датасету для навчання. Його недоліком є необхідність в точно описаних правилах та залежність правил від обраної предметної області.

З цього можна зробити висновок, що імітаційний метод є сенс використовувати коли можна чітко та легко сформувати правила поведінки сутностей гри. Тоді він буде давати точніший результат. Тобто він підходить для простих ігор. Статистичний метод є сенс використовувати коли є гарний датасет для навчання моделі. Але це навчання займає де-який час. Його є сенс використовувати для передбачення результатів ігор зі складнішою логікою.

5 ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ У НАУКОВІЙ І ПРАКТИЧНІЙ ДІЯЛЬНОСТІ

Статистичний метод може використовуватися в багатьох галузях, наприклад, в економічній, політичній, військовій, спортивній та багатьох інших галузях. Завдяки навчанню статистичної моделі можна передбачати результати для нових вхідних даних та отримувати значущість кожної характеристики для вихідного значення. Наприклад, маючи достатньо великий датасет, що описує публікації в Інтернеті, можна навчити модель, що буде фільтрувати вигадані новини. Можливими характеристиками такого датасету можуть бути наявність якихось слів в назві або тексті статті, кількість позначок «сбодобалося», зміст коментарів. Для вирішення такої ж задачі надто важко буде придумати логіку та реалізації імітаційного методу. Існує багато способів реалізації статистичного методу. Найпоширенішими методами є лінійна регресія, логістична регресія та різноманітні нейронні мережі.

Імітаційний метод можливо використати коли є точний алгоритм для передбачення результатів. Імітаційний підхід використовується в багатьох симуляторах спортивних ігор, в розрахунках природних, хімічних та фізичних явищ.

Проведене дослідження показує, що імітаційний метод дає точніший результат але більш предметно-орієнтований та складніший для змін. Швидкість передбачення для однієї ситуації достатньо мала для обох методів моделювання.

Розроблений додаток для імітаційного методу моделювання може бути використаний баскетбольними тренерами та аналітиками для:

- передбачення результатів певної стратегії;
- візуального перегляду ходу гри як в нападі, так і в захисті;
- порівняння впливу різноманітних характеристик на кінцевий результат;
- перебору усіх можливих ситуацій та знаходження найкращих характеристик для найбільш успішного результату.

Такий саме додаток може бути розроблений для будь-якої галузі. Усі компоненти програми слабо пов’язані тому будь-яка частина може бути поліпшена без руйнівного впливу на програму. Так можуть бути змінені:

- модуль малювання програми на більш швидкий двигун з більшою кількістю можливостей;
- модуль логіки може бути покращений або навіть змінена предметна галузь програми;
- логіка перебору усіх можливих ситуацій та вибір найбільш влучних.

Подальшим розвитком дослідження може стати синтез двох методів та їх компонування для створення програми, що за складністю та логікою буде нагадувати штучний інтелект. Тобто частину рішень вона зможе приймати за допомогою імітаційного алгоритму, іншу частину – за допомогою статистичного методу.

ВИСНОВКИ

Під час дослідження були проаналізовані головні способи комп’ютерного моделювання, а саме:

– фізичне моделювання;

– динамічне або чисельне моделювання;

– імітаційне моделювання;

– статистичне моделювання;

– інформаційне моделювання;

– моделювання знань.

Найбільш відповідними серед них для вирішення задачі формалізації ігрових процесів були обрані імітаційне моделювання та статистичне моделювання.

У звіті були описані деталі кожного з методів та способи реалізації програмного продукту для досягнення мети дослідження.

Достатньо точне моделювання ігрового процесу дозволяє користувачам обирати виграшну тактику залежно від дій опонента. Програма, що моделюватиме ігровий процес, матиме великий вплив на спортивну індустрію. Аналітика ігрових ситуацій буде значно спрощена.

Окрім цього, результати роботи можуть бути використані у багатьох інших індустріях, таких як політична, економічна, юридична. Усі життєві процеси, що мають набір правил та конфліктуючі інтереси, можуть бути розглянуті як ігрові.

Для статистичного методу необхідно обрати параметри, що впливають на результат. Для імітаційного методу потрібно задати сутності, їх властивості та системи, що змінюють сутності.

Під час дослідження більшу точність показав імітаційний метод моделювання. Головною проблемою дослідження стала відсутність датасету та необхідність його створювати власноруч. Великою перевагою імітаційного методу моделювання є відсутність необхідності даних для навчання. З іншого боку, алгоритмічно скласти правила для ігрового процесу не завжди є можливим.

Тому статистичний метод є більш різнобічним. Він дуже гарно підходить для ігрових процесів зі складною логікою. Швидкість обох методів залежить від різних параметрів, тому її порівняння не має великого внеску. Зміна предметної галузі імітаційної моделі можлива лише завдяки програмному втручанню, статистична модель, в свою чергу, не потребує програмних змін.

Обидва методи можливо використати, щоб знайти «ідеальні» параметри, тобто значення параметрів, що будуть давати достатньо високий результат. Наприклад, тренер може використовувати результати роботи програми щоб зрозуміти, яким показникам треба наділити більше уваги під час тренувального процесу.

Результати дослідження опубліковані на 15-ій міжнародній науковій конференції «Science And Society» від 8-го листопада 2019-го року [4].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гра [Електронний ресурс] / Wikipedia, the free encyclopedia. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0 – 21.10.2019 р. – Назва з екрана.
2. Йоган Гейзинга. Досвід визначення ігрового елемента культури., навч. посіб. – Київ: Основи, 1994. – 20 с.
3. Мороз О. С., Шинкарук В. І.; Л. В. Озадовська. Моделювання – Київ: Абрис, 2002. – С. 392. – 742.
4. Шутєєв І.В. Дослідження методів формалізації для ігрових ситуацій // Premier Publishing. 2019. № 16. С. 123.
5. Федоренко Р.П. Введення в обчислювану фізику, навч. посіб. – Москва: видат. физ.–техн. ин–та, 1994. — 528 с.
6. Советов Б.Я., Яковлев С.А. Моделювання систем: підр. для вузів – Москва: Висш. Шк., 2001. – 343 с.
7. Кунин С. Обчислювальна фізика. – Москва: Світ, 1992. – 518 с.
8. Боев В.Д., Сипченко Р.П., Комп’ютерне моделювання. – Москва: Інтуіт, 2010. – 349 с.
9. Дворецький С.І., Муромцев Ю.Л., Погонин В.А. Моделювання систем. – Москва: Академія, 2009. – 320 с.
10. Паничев В.В., Солов’їв Н.А. Комп’ютерне моделювання: навч. посіб. – Оренбург: ГОУ ОГУ, 2008. – 130 с.
11. Бусленко Н.П. Моделювання складних систем. – Москва: Наука, 1968. – 356 с.
12. Рубанов В.Г., Філатов А.Г. Моделювання систем, навч. посіб. – Бєлгород: Видатництво БГТУ, 2006. – 349 с.
13. Булавін Л.А., Вигорніцький Н.В., Лебовка Н.І. Комп’ютерне моделювання фізичних систем. – Долгопрудний: Видатницький Дом «Інтелект», 2011. – 352 с.
14. Нейман Дж., Моргенштерн О. Теорія ігор та економічна поведінка – Москва: Наука, 1970.

15. Шеллінг, Т. Стратегія конфлікту – Москва: IPICEH, 2007.
16. Рівновага Неша [Електронний ресурс] / Wikipedia, the free encyclopedia. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D1%96%D0%B2%D0%BD%D0%BE%D0%B2%D0%B0%D0%B3%D0%B0_%D0%9D%D0%B5%D1%88%D0%B0 – 21.10.2019 р. – Назва з екрана.
17. Новіков Ю.С., Підоренко Д.А. Моделювання процесів побудови оптимальної стратегії гри за допомогою матричних ігор // ХНУРЕ, 2017 с. 150.
18. Фрідман Д. Статистичні моделі: теорія та практика. Кембрідж: Видатництво кембріджського університету, 2009. – 128 с.
19. Машинне навчання [Електронний ресурс] / Coursera. – Режим доступу: https://www.coursera.org/learn/machine-learning/home/info – 21.10.2019 р. – Назва з екрана.
20. Entity Systems [Електронний ресурс] / T-machine.org. – Режим доступу: http://t-machine.org/index.php/2007/09/03/entity- systems-are-the-future-of-mmog-development-part-1/ – 21.10.2019 р. – Назва з екрана.
21. Марк П. C # 7 і .NET Core. Крос-платформна розробка для професіоналів – Санкт-Петербург: Пітер, 2016.
22. Accord.NET Machine Learning Framework [Електронний ресурс] / Accord.NET – Режим доступу: http://accord-framework.net/ – 11.01.2019 р. – Назва з екрана.
23. InversionOfControl [Електронний ресурс] / Мартін Фаулер – Режим доступу: https://martinfowler.com/bliki/InversionOfControl.html – 11.01.2019 р. – Назва з екрана.