МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра Інформаційних технологій

Кваліфікаційна робота магістра

на тему: Розробка імітаційної моделі взаємодії інтелектуальних агентів

Виконав студент групи МІС-22
спеціальності 122 Комп'ютерні науки
Сербін Сергій Валерійович

Керівник к.т.н., доцент
Гнатовська Ганна Арнольдівна

Рецензент к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Одеса 2023

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ТЕРМІНІВ І ПОЗНАЧЕНЬ

БАС – багатоагентна система
БЗ – база знань
ГА – генетичний алгоритм
ІНМ – інтелектуальна нейронна мережа
МАС – мультиагентна система
НМ – нейронна мережа
ПА – програмні агенти
ШІ – штучний інтелект
ШНМ – штучна нейронна мережа
ABM – (agent-basedmodel) – агентне моделювання
BDI (Belief – Desire – Intention) – архітектура інтелектуального агента
MLP (multilayer perceptron) – багатошаровий персептрон

ВСТУП

Завдяки інтенсивному розвитку інформатики та комп'ютерних технологій стало набагато простіше вирішувати складні завдання, що вимагають великих витрат часу та фінансових витрат. Спростити та прискорити рішення можливо з використанням імітаційного комп'ютерного моделювання об'єктів та процесів реального світу, що на сьогодні є одним з найбільш поширених та зручних способів моделювання складних систем [1].

В основу моделювання закладено процедуру формалізації, що дозволяє здійснювати переклад властивостей об'єкта на мову понять предметної галузі, алгоритмів та математики. Одним з найпоширеніших методів імітаційного моделювання є агентний метод, який дозволяє уявити складну систему у вигляді процесу, тобто послідовності операцій, що виконуються з агентами. Агентне моделювання (agent-basedmodel (ABM)) є різновидом імітаційного моделювання, що дозволяє досліджувати роботу децентралізованих агентів і визначати діяльність усієї системи в цілому в залежності від поведінки кожного з агентів.

При використанні агентного підходу завданням імітаційного моделювання є визначення характеристик стану агентів і середовища, вивчення поведінки агентів при різних ситуаціях взаємодії, а також при різних змінах стану середовища. Агентний підхід дозволяє досліджувати завдання колективної взаємодії, ефективно вирішувати задачі прогнозування. Агентні системи дозволяють досліджувати процеси самоорганізації, дають можливість природного опису складних систем, мають високу гнучкість. Агентів моделі може бути як багато, так і мало. При цьому агенти можуть бути як одного типу, так і різних. Застосування систем з багатьма агентами обумовлено все більш зростаючою складністю сучасних інформаційних систем, об’єктів дослідження [2].

Багатоагентні системи мають властивості здійснення самоорганизаці, що дозволяє здійснювати пошук найкращих рішень без зовнішнього втру-

чання та витрат найменшої кількості енергії за умов обмежених ресурсів. Перевагою застосування багатоагентної системи є гнучкість, що зумовлена властивістю самовідновлення, стійкістю до збоїв завдяки достатньому запасу компонентів і самоорганізації. Такі системи можуть бути використані для вирішення таких проблем, які складно чи неможливо вирішити за допомогою одного агента чи монолітної системи. Прикладами таких завдань є онлайн-торгівля, ліквідація надзвичайних ситуацій, моделювання соціальних структур, тощо. Інтелектуальні мультиагентні системи – один із нових перспективних напрямів штучного інтелекту, який сформувався на основі результатів досліджень у галузі розподілених комп'ютерних систем, мережевих технологій вирішення проблем та паралельних обчислень. У мультиагентних технологіях закладено принцип автономності окремих частин програми, які здатні спільно функціонувати у розподіленій системі, де одночасно протікає безліч взаємопов'язаних процесів. Такі програми називають агентами [3].

В даний час у розробці ігор все більше застосовуються методи класичного штучного інтелекту. Актуальним є завдання створення імітаційної моделі поведінки невласних персонажів, керування якими здійснює комп’ютер. Ігровий штучний інтелект, крім методів традиційного штучного інтелекту, включає також алгоритми теорії управління, робототехніки, комп'ютерної графіки та інформатики загалом. Алгоритми інтелектуальної обробки даних застосовуються в тих випадках, коли точне вирішення задачі аналітичними методами невідоме або має велику обчислювальну складність, тому для досягнення більш реалістичної поведінки використовуються методики штучного інтелекту: нейронні мережі, продукційні правила, кінцеві автомати. У комп’ютерних іграх та завдання анімації актуальною є задача моделювання поведінки агентів, а саме їх пересування та алгоритмів орієнтації в навколишньому світі, імітуючи поведінку в реальному світі. Завдання полягає в вирішенні стратегії поведінки, що описує правила вибору шляху пересування об’єктів, а також для імітації самого процесу руху. Обираючи ту чи іншу стратегію руху або їх комбінації інтелектуальні агенти виконують завдання пере-

сування з одного місця в інше, не уможливлення зіткнень з іншими об’єктами, здійснення цілеспрямованого руху з групою агентів, приєднуючись до цих груп.

Інтелектуальний агент уявляється як автономний штучний об'єкт, що має активну мотивовану поведінку і здатний до взаємодії з іншими об'єктами в динамічних віртуальних середовищах. Саме від середовища агент отримує дані, які зумовлюють події, що відбуваються в середовищі, виконує команди, інтерпретуючи отримані данні.

Мета роботи – розробка імітаційної моделі взаємодії інтелектуальних агентів, що дозволяє досліджувати і аналізувати різні стратегії поведінки та використовуючи засоби нейронних мереж здійснювати навчання агентів.

Ця імітаційна модель може бути використана як для вирішення завдань комп’ютерної анімації, графічних додатків, ігор.

1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ МЕТОДІВ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

1.1 Методологічні засади імітаційного моделювання

Стрімкий розвиток сучасних інформаційних та комп'ютерних технологій зумовив розширення можливостей моделювання реальних об'єктів, з'явилися нові методи та технології, що дозволяють моделювати складні об'єкти та процеси в промисловості, охороні здоров'я, в економічних та соціальних системах, у науці та інших сферах. А поява нових систем (середовищ) моделювання призвело до появи нового типу комп'ютерних моделей – «імітаційних моделей» [2].

Під імітаційним моделюванням розуміється розробка моделі системи у вигляді програми для комп'ютера та проведення експериментів із програмою замість проведення експериментів з реальною системою чи об'єктом. Імітаційне моделювання застосовується, коли неможливо побудувати аналітичну модель системи, що враховує причинні зв'язки, наслідки, не лінійності, стохастичні змінні, коли необхідно імітувати поведінку системи в часі, розглядаючи різні можливі сценарії її розвитку при зміні зовнішніх та внутрішніх умов. Таким чином, імітаційне моделювання – це високорівнева інформаційна технологія із застосуванням комп'ютерів, яка найчастіше використовувана при моделюванні складних систем [1].

Імітаційне моделювання умовно може бути представлено різними різновидами або напрямками, відповідно які мають свої методології. Визначені наступні різновиди (напрями) імітаційного моделювання [1]:

- статистичне (чисельне) моделювання;
- моделювання динамічних систем;
- системна динаміка;
- експертне моделювання;
- когнітивне моделювання;

– агентне моделювання;

– ситуаційне моделювання;

– дискретно-подійне моделювання.

Імітаційне моделювання довело свою успішність у багатьох сфер застосування. Поява нових методів моделювання та зростання обчислювальної потужності комп'ютерів дозволяє стверджувати, що кількість цих областей тільки зростатиме.

В імітаційному моделюванні під методом розуміється деяка основа, яку використовують, щоб «перевести» систему з реального світу у світ моделей. Метод передбачає певну мову, «положення та умови» для розробки моделі. На даний момент існує три методи:

– системна динаміка;

– дискретно-подійне моделювання;

– агентне моделювання.

Кожен метод застосовується у певному діапазоні рівнів абстракції. Системна динаміка передбачає дуже високий рівень абстракції і зазвичай використовується для стратегічного моделювання. Дискретно-подійне моделювання підтримує середній та низький рівні абстракції. Між ними знаходяться агентні моделі, які можуть бути дуже деталізованими, коли агенти представляють фізичні об'єкти, так і гранично абстрактними, коли за допомогою агентів моделюються конкуруючі компанії або уряди держав.

Перш ніж вибрати метод моделювання, слід ретельно дослідити систему, модель якої створюється та цілі моделювання.

1.2 Метод імітаційного моделювання

Агентне моделювання (agent-basedmodel (ABM)) є одним з різновидів імітаційного моделювання, сучасний метод, що дозволяє досліджувати роботу децентралізованих агентів і те, як така поведінка визначає діяльність усієї системи в цілому. На відміну від методу системної динаміки, аналітик визна-

чає поведінку агентів на індивідуальному рівні, а глобальна поведінка виникає як результат діяльності безлічі агентів (моделювання «знизу вгору»). Агентне моделювання – відносно новий метод моделювання. Починаючи з 2000-х років розробники імітаційних моделей почали використовувати метод агентного моделювання на практиці.

Перехід до агентного моделювання було зумовлено наступними причинами [3]:

– бажанням глибше вивчити системи, які важко описати традиційними методами моделювання;
– розвитком технології агентного моделювання (об'єктно-орієнтоване моделювання, діаграми станів);
– швидким зростанням потужності процесорів та обсягу оперативної пам'яті комп'ютерів.

Агентні моделі більш вимогливі до ресурсів, ніж моделі системної динаміки чи дискретно-подійні моделі. Агентне моделювання пропонує розробнику моделей альтернативний погляд на поведінку системи.

Можливо не знати ні поведінки системи в цілому, ні її головних змінних і залежностей між ними або не бачити чіткої схеми процесів, але при цьому розуміти, як поводяться окремі елементи системи. У такому випадку можливо почати створення моделі з ідентифікації об'єктів (агентів), що моделюються, і завдання їх поведінки. Іноді може знадобитися об'єднати агентів у мережу і дозволити їм взаємодіяти один з одним або помістити агентів у середу, яка має власну динаміку. Таким чином, глобальна поведінка системи формується з багатьох десятків (тисяч, мільйонів) процесів, що паралельно протікають.

На даний момент не існує стандартної мови агентного моделювання. Структура агентної моделі може бути визначена як графічно, і за допомогою сценаріїв. Поведінка агента може бути поставлено різними способами. Якщо агент має стан, від якого залежать його дії і реакції, то його поведінка найкраще ставити за допомогою діаграми станів. Іноді поведінка агента задаєть-

ся діями, що виконуються при настанні певних подій. Іноді внутрішня динаміка агента найкраще задається за допомогою дискретних подій чи системної динаміки. Також і динаміка середовища, в якому живуть агенти, може моделюватися з допомогою традиційних методологій. З цієї причини багатоагентні моделі поєднують у собі кілька підходів до моделювання [3].

Агентами можуть бути різні об'єкти: транспортні засоби, обладнання, проекти, організації, земельні ділянки, люди і т.п.

1.3 Об’єкти, засоби та цілі агентного моделювання

Завданням імітаційного моделювання при агентному підході полягає у визначенні характеристик стану агентів і середовища, вивчення поведінки агентів при різних ситуаціях взаємодії – і станах середовища, що змінюються. Через вивчення поведінки безлічі агентів у деякому просторі згідно з деякими правилами взаємодії можливо прогнозування поведінки системи загалом. Агент – це деяка сутність, що володіє активністю, автономною поведінкою, може приймати рішення відповідно до деякого набору правил, взаємодіяти з оточенням, а також самостійно змінюватись. Агентами може бути люди (споживачі, жителі, працівники, пацієнти, лікарі, клієнти, солдати та інші); транспорт, обладнання (автомобілі, крани, літаки, вагони, верстати); нематеріальні речі (проекти, продукти, інновації, ідеї, інвестиції); організації (підприємства, політичні партії, країни) [3].

Середовище – деякий простір, в якому знаходяться агенти, що характеризується своїми станами та факторами, агенти перебувають у певному місці цього простору, з можливістю орієнтування та пересування в даному просторі. Правила взаємодії – закони взаємодії агентів у навколишньому середовищі, з процедурами прийняття рішення та вибору стратегії при черговому етапі взаємодії.

Мета створення агентних моделей – отримати уявлення про ці глобальні правила, загальну поведінку системи виходячи з припущень про індивіду-

альну, приватну поведінку її окремих активних об'єктів та взаємодію цих об'єктів у системі.

Агентний підхід дозволяє досліджувати завдання колективної взаємодії, ефективно вирішувати задачі прогнозування. Агентні системи дозволяють досліджувати процеси самоорганізації, дають можливість природного опису складних систем, мають високу гнучкість.

Аналіз особливостей формального опису та дослідження складних об'єктів та систем показує, що при моделюванні та управлінні об'єктами та системами слід базуватися на концепціях та принципах, покладених в основу сучасних технологій системного (комплексного) моделювання.

Зазначимо основні переваги імітаційного моделювання [1]:

– імітаційні моделі дозволяють аналізувати системи і знаходити рішення у тих випадках, коли такі методи, як аналітичні обчислення та лінійне програмування не справляються із завданням;
– після того, як визначено з рівнем абстракції, розробляти імітаційну модель буде набагато простіше, ніж аналітичну, оскільки процес створення моделі буде інкрементальним та модульним;
– структура імітаційної моделі природним чином відображає структуру системи, що моделюється.
– імітаційна модель дозволяє відстежувати всі об'єкти системи, враховані у вибраному рівні абстракції, додавати метрики та проводити статистичний аналіз;
– однією з головних переваг імітаційного моделювання є можливість програвати модель у часі та анімувати її поведінку. Анімація буде незаперечною перевагою при демонстрації моделі і може виявитися корисною для верифікації моделі та знаходження помилок.
– імітаційні моделі набагато переконливіші за електронні таблиці. Використовуючи імітаційне моделювання, презентація проекту має явну перевагу перед тими, у кого на руках лише цифри та рішення.

2 ВИЗНАЧЕННЯ ПОВЕДІНКИ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ ТА СТАНІВ СЕРЕДОВИЩА

2.1 Основні засади агентно-орієнтованого підходу

Штучний інтелект має багато додатків у різних галузях. Комп'ютерні ігри та анімаційна графіка стали об'єктом дослідження, у тому числі й у контексті штучного інтелекту, ще на початку 50-х років. Ігровий штучний інтелект використовує багато технік класичного штучного інтелекту, які також застосовуються і в інших областях: пошук у просторі станів, нейронні мережі, нечітка логіка та інші. В даний час у розробці ігор, анімації все більше застосовуються методи класичного штучного інтелекту, зростає інтерес академічної спільноти до технологій, що використовуються в комп'ютерні ігри. Домінуючим підходом до розробки інтелектуальних систем є агентно-орієнтований підхід. Основними його концепціями є поняття агента та середовища, в якому агент знаходиться. Агентом є все, що може розглядатися як таке, що сприймає своє середовище за допомогою датчиків та впливає на це середовище за допомогою виконавчих механізмів. Ігровий агент взаємодіє з середовищем згідно з правилами гри, які мають різну складність, при цьому властивості середовища також є визначальними для поведінки агента. Описуючи ігри з різними правилами та різними властивостями ігрового світу, створюються моделі світів довільної складності, для яких необхідно створити ефективного агента з можливістю навчання з часом. Саме тому ігри представляють прекрасний полігон для різних апробацій методів штучного інтелекту [2].

У мультиагентних іграх беруть участь два або більше гравців підпрограм, які конкурують за деякий ресурс в ігровому полі. Ігрове поле є деяким обмеженим простором, яким певним чином розподілені перешкоди, загрози у вигляді різних елементів, що завдають гравцям шкоди, а також самі ресурси. Завдання гравців полягає в тому, щоб випередивши суперника отримати ре-

сурси і при цьому уникнути загроз. Як правило, гравцям доступні різні дії, вибір наступної дії здійснюють на підставі доступної інформації про середовище та конкурентів, а також на підставі власної історії актів сприйняття. Середовище, в якому існують ігрові агенти, може бути повністю або частково-контролюємою, коли агент бачить все ігрове поле або не бачить якихось його областей через перешкоди або бачить обмежений набір квадратів біля себе. Середовище може бути стохастичним, а може таким здаватися через те, що середовище є частково контролюємим [4].

Практично всі ігрові середовища є конкурентними, послідовними, при цьому можуть бути дискретними, так і безперервними. Частково спостерігаються, стохастичні, динамічні середовища є особливо складними, оскільки в умовах нестачі інформації про світ, що змінюється, агенту доводиться підтримувати деяку внутрішню модель світу, щоб мати уявлення про його стан та зміни. Реалізація агентів для таких ігор є складним завданням.

2.2 Визначення інтелектуальних агентів

У комп'ютерних науках інтелектуальний агент визначається, як програма, яка самостійно виконує завдання, які надані користувачем комп'ютера протягом тривалих проміжків часу. Основним завданням для інтелектуального агента є сприяння оператору чи збору інформації. Комп'ютерні віруси, роботи, пошукові системи – все це можна віднести до «інтелектуальних» агентів, які мають складний алгоритм функціонування, для чого доцільно використовувати нейронні мережі, які мають властивість навчання з часом. У цьому контексті «інтелектуальність» визначається як спроможність отримання зворотного відгуку за результатами аналізу попередніх пошукових запитів [5].

В теорії штучного інтелекту поняття «інтелектуальний агент» визначається у вигляді сутності, яка здатна отримувати інформацію через групу сенсорів про стан керованих ними процесів, і здійснюють вплив на них через

систему актуаторів. Наприклад, у навколишньому середовищі це може бути примітивна інстинктивна поведінка комах.

При цьому, термін «інтелектуальний» не відображає наявності будь-якого інтелекту, але визначає вищий рівень технології управління в порівнянні з примітивними тригерними системами автоматичного управління. Такий агент може бути як програмною системою, так і складною автоматизованою системою, наприклад, верстатом з ЧПУ або комплексом управління технологічними, логістичними, фінансовими чи будь-якими іншими процесами. Про «інтелектуальність» агента можна говорити, якщо його взаємодія з навколишнім середовищем є адекватною тій чи іншій системі вимог. Жодного відношення навіть до інтелекту вищих тварин і вже тим більше людини подібна функціональність не має [4].

Сприйняття інтелектуальним агентом навколишнього світу не може бути пасивним, т. к. його завдання полягає в активному та цілеспрямованому зборі інформації щодо стану навколишнього середовища. Інтелектуальний агент повинен не тільки активно сприймати навколишнє середовище, але також модифікувати та поповнювати свої знання про навколишнє середовище на основі зроблених сприйняттів. Здатність модифікувати та поповнювати знання на основі сприйняттів називається здатністю вчитися.

Початковий вміст пам'яті агента може відображати деякі вбудовані знання агента про навколишнє середовище, адекватні середовищі в момент створення агента. Однак, якщо середовище монотонно змінюється, то ці знання все більшою мірою застарівають і перестають бути адекватними. Якщо агент використовує неадекватні знання, для формування своєї поведінки, він робить помилки і може не досягти мети. Агент, який має здатність, навчаться, у міру зміни середовища модифікує та поповнює вбудовані знання, набуваючи «життєвого досвіду». Навчання забезпечує агенту володіння адекватними знаннями про навколишнє середовище, що змінюється, на будь-якому етапі існування агента. Проте можна припустити випадок, коли середовище залишається незмінним протягом усього «життя» агента. Тоді агенту

достатньо вбудованих знань для успішної цілеспрямованої поведінки. В цьому випадку йому не потрібно вчитися. Він завжди діє правильно і не робить помилок [6].

Тому, поведінку інтелектуального агента зумовлюють, як знання, які є початковими на етапі створення агента, так і знання, які набуті в процесі навчання протягом часу існування (рис. 1.1).

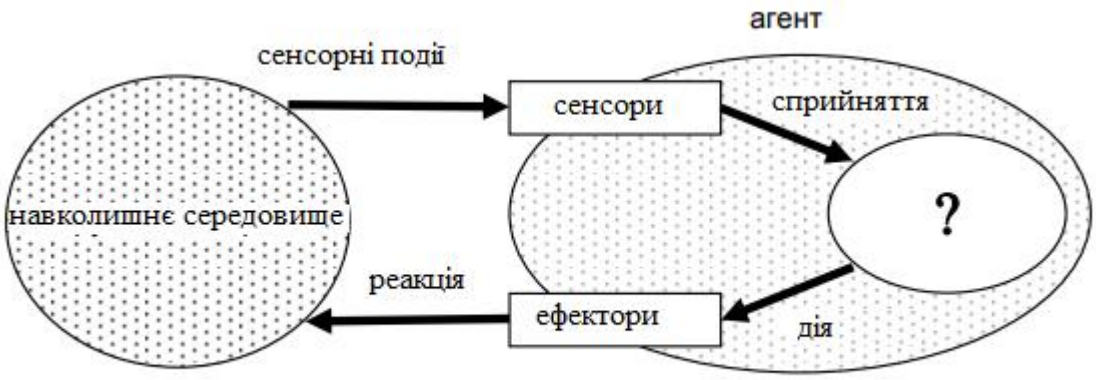

Рисунок 2.1 – Взаємодія інтелектуального агента з середовищем

Агент називається автономним у тому випадку, коли він формує свою поведінку, спираючись і на вбудовані знання та знання, отримані в процесі навчання. Автономний агент набуває досвіду у процесі навчання, а початковий стан такого агента зумовлено вбудованими знаннями.

2.3 Властивості інтелектуальних агентів

Стрімкий розвиток розподілених інформаційних систем зумовив також і розвиток мультиагентних технологій, які отримали появу нових моделей агентів, способів їх реалізації, середовищ розробки мультиагентних систем, алгоритмів та принципів взаємодії агентів у різних середовищах.

Можливо визначити набір базових характеристик довільного агента, серед яких зазначені наступні [6]:

- активність – здатність до організації та реалізації дій;
- автономність – відносна незалежність від навколишнього середовища чи наявність певної «свободи волі», пов'язана з хорошим ресурсним забезпеченням його поведінки;
- комунікабельність, що випливає із необхідності вирішувати свої завдання спільно з іншими об'єктами та забезпечується розвиненими протоколами комунікації;
- цілеспрямованість, що передбачає наявність власних джерел мотивації;
- реактивність – адекватне сприйняття стану середовища і реакція на його зміну.

Вирішення завдання одним агентом на основі інженерії знань являє собою точку зору класичного штучного інтелекту, згідно з яким агент, володіючи глобальним баченням проблеми, має всі необхідні здібності, знання та ресурси для її вирішення.

Агенти – це програми, які мають взаємні зобов'язання, що визначаються у процесі діалогу, ведуть переговори та координують передачу інформації. Процес передбачає наявність сприйняття і дії, а діалог немислимий без засобів комунікації. Агенти мають наполегливість, що пов'язано з наявністю своїх власних уявлень про те, як виконувати завдання, або своїх особистих програм дій.

Агент називається автономним у тому випадку, коли він формує свою поведінку, спираючись і на вбудовані знання та знання, отримані в процесі навчання. Автономний агент набуває досвіду у процесі навчання. І поки такого досвіду немає, агент будує свою поведінку, спираючись лише на вбудовані знання.

Еволюція забезпечує живих істот, на початковому етапі їх існування, деякою кількістю вбудованих знань у вигляді рефлексів, що дозволяють живій істоті існувати до тих пір, поки не набутий необхідний життєвий досвід.

Неавтономний агент, або агент, який будує свою поведінку тільки на основі вбудованих знань, не має достатньої гнучкості, оскільки вбудовані знання відображають деякі припущення про середовище в момент створення агента. Неавтономний агент таким чином, функціонує успішно доти, доки незмінним залишається середовище.

Програма інтелектуального агента має працювати у обчислювальній системі, забезпеченою сенсорами та ефекторами. Цю систему називають архітектурою інтелектуального агента. Архітектура інтелектуального агента забезпечує формування та передачу в програму сприйняттів, виконання програми, а також формування дій та передачу їх у ефектори. Очевидно, що програма агента та його архітектура повинні відповідати один одному. Наприклад, якщо програма формує дію Walk, то архітектура агента має включати опорно-руховий апарат [6].

Для здійснення класифікації агентний програм використовуються наступні дві основні ознаки: степінь розвитку внутрішньої уяви про зовнішній середовище та спосіб поведінки.

За першою ознакою наведеної класифікації виділяють інтелектуальні і реактивні агенти, які мають здатність розмірковувати. Таких інтелектуальних агентів можливо охарактеризувати, як агентів з розвинутою і поповнюваною символьною моделлю зовнішнього оточення, яка забезпечена наявною базою знань, механізмів здійснення міркувань та аналізу дій.

Поведінка реактивних агентів визначається метою, для досягнення якої відповідно формуються реакції на пропоновані ситуації. Такі агенти не мають уявлення про зовнішнє середовище, не використовують міркувань та не мають власних ресурсів. Саме реактивні агенти не мають внутрішніх джерел мотивації та не можуть планувати свої дії.

Навколишнє середовище є джерелом проблем чи завдань, які вирішує інтелектуальний агент. Тому для у інтелектуальних агентів, навколишнє середовище називають проблемним середовищем. По суті, поведінка інтелектуального агента – це зовнішній прояв процесу вирішення ним того чи іншо-

го завдання. Інтуїтивно ясно, що чим складніше навколишнє середовище, тим складніші завдання воно ставить перед агентом і тим складніше має бути сам агент (рис. 2.2).

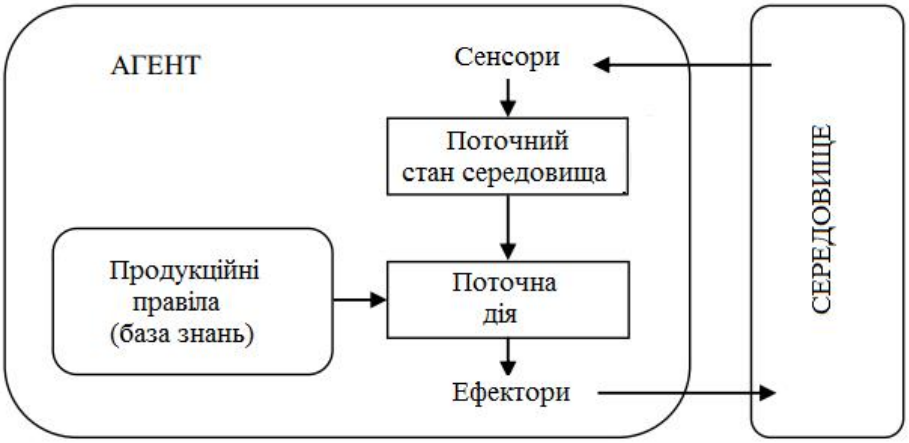

Рисунок 2.2 – Структура простого реактивного агента

Інтелектуальна поведінка агента забезпечується здатністю приймати рішення, а реактивна – системою контролю за вмістом робочої пам'яті, що функціонує за принципом глобальної дошки оголошень.

2.4 Аналіз архітектурних рішень мультиагентних систем

Архітектура мультиагентних систем визначає організацію та взаємодію агентів усередині системи та має різні компоненти і принципи, які спрямовані на забезпечення ефективного функціонування системи. Всі агенти в такій системі є автономними та мають можливість приймати рішення і виконувати дії. Кожен з агентів має знання та змогу взаємодіяти між іншими агентами, має свої цілі [7].

Основним аспектом мультиагентних систем є здатність взаємодії між агентами. Комунікація між агентами дозволяє здійснювати обмін інформацією, узгоджувати свої дії та передавати повідомлення. Для здійснення процесу

комунікації в мультиагентній системі застосовують різні протоколи та механізми взаємодії.

Організація агентів у мультиагентній системі визначає, як вони співпрацюють та координують свої дії. Ієрархічна структура забезпечує агентам різні рівні влади та відповідальності, а децентралізована структура забезпечує агентам рівні права та можливість приймати рішення на основі погодження.

Управління мультиагентною системою включає контроль і координацію дій агентів. Це може бути централізоване управління, де існує один агент або система, яка приймає рішення та керує всіма агентами, або децентралізоване управління, де кожен агент приймає рішення самостійно, але з урахуванням загальної мети системи. Принципи роботи мультиагентних систем ґрунтуються на ідеях розподіленого інтелекту, де кожен агент виконує своє завдання, але взаємодіє з іншими агентами для досягнення спільної мети.

Архітектура мультиагентних систем може бути різною, включаючи централізовану, децентралізовану або гібридну структуру.

При розробці мультиагентної системи можливо використовувати різні варіанти її архітектури, які залежать від концепції. Виділяють наступні базові три типи архітектур [7]:

– архітектури, засновані на методах роботи зі знаннями;
– архітектури, у яких використовуються поведінкові моделі «стимул – реакція»;
– гібридні архітектури.

Інтелектуальна мультиагентна система є множиною інтелектуальних агентів, які мігрують по ній у пошуках релевантних даних, знань, процедур і обов’язково об’єднуються для досягнення поставленої мети.

В архітектурі, яка заснована на методах роботи зі знаннями для подання і обробки знань використовуються традиційні моделі, методи і засоби штучного інтелекту, а прийняття рішень здійснюється на основі механізмів формальних міркувань. У найперших системах такого типу для подання і об-

робки знань використовувалася логіка предикатів першого порядку. Розвиток досліджень у цій області привів до появи спеціальних розширень логічних вираховувань, орієнтованих на урахування таких властивостей агентів, як переконання, бажання, наміри і зобов'язання. Основний недолік використання такої архітектури – це складність або принципова неможливість побудови досить повних баз знань, які є необхідною частиною таких систем. Зокрема, інтелектуальний агент може мати архітектуру типової продукційної системи, що здатна сприймати інформацію із зовнішнього середовища і здійснювати ті або інші дії в результаті обробки цієї інформації. Головні відмінності агентної програми від звичайної продукційної пов'язані з наявністю механізму формування цілей і модуля комунікації, що забезпечує взаємодію з іншими агентами. Агент із такою архітектурою здатний до міркувань, але не здатний до навчання. Адаптивна поведінка агента дозволяє реалізувати архітектуру на основі систем, що класифікує Дж.Холланд. Найважливішими відмінностями систем, які класифікують, від продукційних є [8]:

- можливість формування нових правил із застосуванням генетичного алгоритму;
- наявність механізму заохочень.

Архітектурі, у яких використовуються поведінкові моделі «стимул – реакція» називають реактивними. Такі архітектури не використовуються традиційні для штучного інтелекту символьні моделі подання знань. Моделі поведінки агентів представлені або наборами правил, які дозволяють вибрати дії, що відповідають ситуації, або кінцевими автоматами, або іншими засобами, що забезпечують формування адекватних реакцій агента на виникаючі в системі стимули. Такі системи мають високу степінь спеціалізації і строгі обмеження на складність задач, які підлягають вирішенню.

Гібридні інтелектуальні мультиагентні системи мають використовувати властивості як інтелектуальних так і реактивних архітектурних рішень. Наприклад, архітектура з ієрархічною базою знань повинна містити структуровану базу знань, робочу пам'ять, модуль керування комунікацією і людино-

машинний інтерфейс. Агенти при такій архітектурі системи мають можливість здійснення міркувань. База знань інтелектуального агента в гібридній інтелектуальній мультиагентній системі має три рівні: знання предметної області; знання про взаємодії, які дозволяють приймати рішення в умовах невизначеності; керуючі знання.

Гібридні архітектурні рішення є багаторівневими і відрізняються друг від друга структурою і змістом рівнів, які можуть відповідати різним рівням керування, абстракції або окремим функціональним властивостям агента. Зручним інструментом для реалізації багатоагентної системи є нейронні мережі. Саме завдяки архітектурі, яка заснована на інтелектуальної нейронної мережі з’являється можливість створення самонавчающихся агентів, знання яких формуються в процесі розв’язання практичних завдань. Тому актуальним завданням є створення агентів, що самі навчаються, а також мають мережі зі зворотними зв'язками.

2.5 Організація взаємодій між агентами

Колективна взаємодія агентів має особливі риси функціонування, яка характеризується наявністю взаємодії агентів при вирішенні як приватних завдань так і одного загального завдання. При моделюванні поведінки агентів в таких системах використовуються наступні правила координації [2]:

- принцип координації засновано на відмові від пошуку найкращого рішення на користь «хорошого», що забезпечує виконання пошуку компромісного рішення, а не виконання оптимізації пошуку;
- стійкість механізму формування колективної поведінки забезпечується дотриманням принципів самоорганізації;
- при вирішенні конфліктних ситуацій використовується випадково-ймовірнісний спосіб вибору рішень;

– застосування рефлексивного управління, яке передбачає формування у суб’єкта бажання і намірів діяти у зв’язці, свідомо підкоряючись впливу ззовні.

Визначено наступну низьку умов взаємодії між агентами: кооперація, конкуренція, компроміс, конформізм, відхилення від взаємодії.

У мультиагентних системах МАС завдання розподілені між агентами, кожен із яких сприймається як член групи чи організації. Розподіл завдань передбачає призначення ролей кожному з членів групи, визначення мети, його відповідальності та вимог до досвіду.

Основою формою організації взаємодії між агентами, що характеризується об'єднанням їх зусиль для досягнення спільної мети при одночасному поділі між ними функцій, ролей та обов'язків є кооперація. Загалом це поняття можна визначити формулою: кооперація дорівнює сумі трьох складових: співпраця, координація дій та вирішення конфліктів. Під координацією зазвичай розуміється управління залежностями між діями. Комунікація між агентами залежить від обраного протоколу, який є безліччю правил, визначальних, як синтезувати значні та правильні повідомлення. Фундаментальними особливостями групи, складеної з агентів, які співпрацюють для досягнення спільної цілі, є соціальна структура та розподіл ролей між агентами.

Для будь-якої мультиагентної системи можливо визначити основні риси взаємодії [4]:

– спрямованість − це кооперація або конкуренція; співробітництво або конфронтація; координація або субординація; може мати значення позитивна або негативна.
– вибірковість − взаємодія між агентами здійснюється за рахунок відповіді один одному і єдиній меті у поставленій задачі, з урахуванням, що агенти можуть бути зв'язані в одному відношенні і незалежні – в іншому;
– інтенсивність − ця взаємодія передбачає наявність певної сили між агентами та не зводиться до наявності або відсутності;

– динамічність − це характеристика, що відображає наявність, силу і спрямованість взаємодій, які здатні змінюватися з часом.

Аналіз взаємодії між агентами у мультиагентних системах передбачає вирішення наступних задач: визначення ролей агентів та їх розподіл між ними; визначення кількості агентів та типів їх взаємодії; визначення стратегій поведінки агентів; побудову формальної моделі взаємодії агентів у системі; визначення множини комунікативних дій; визначення ситуації взаємодії агентів.

Відмінною особливістю мультиагентних систем є наявність взаємодії між агентами, яка передбачає встановлення двосторонніх і багатобічних динамічних відношень між суб'єктами. Взаємодія забезпечує у такій системі не тільки зв'язок між співіснуючими агентами, але і створює передумови для взаємних перетворень як самих агентів так і відношень між ними.

Інтелектуальні агенти мають можливість співпраці з іншими агентами, маючи при цьому певні цілі. Кооперація серед реактивних агентів базується на природних реакціях окремих агентів та є ненавмисною, що має на меті підтримку виживання виду. Виживання передбачає наявність у агента або групи агентів здатності зберігати свою цілісність, незважаючи на фактори, які можуть їх зруйнувати. Кооперація агентів може бути директивною, яка виникає на примусових умовах, або ситуативна, яка заснована на добровільних відношеннях агентів. Такі види називаються контрактною формою кооперації, та передбачають взаємодію агентів засновану на існуванні формальних або неформальних угод між ними.

Визначимо причини взаємодія агентів в мультиагентних системах [5].

Перша причини, єдина мета. Саме сумісність цілей передбачає взаємодію по типу кооперації або співпраці. Але така причина може приводити до зниження життєздатності окремих агентів. Стимулюванню процесів розвитку може погрожувати несумісність цілей або переконань, що приводить до конфліктів. Прикладом такої одночасної взаємодії є модель «хижак-жертва», яка має взаємодію по двох типах кооперація-конфронтація.

Наступна причина – відношення до ресурсів. В ролі ресурсів виступають будь-які засоби, які використовують агенти для досягнення своєї мети. Обмеження кількості ресурсів у мультиагентних системах є приводом наражденння конфліктів між агентами. Рішенням такої конфліктної ситуації є розподіл ресурсів між агентами за правилом: найсильніший агент відбирає ресурс у слабшого. Інші алгоритми розв’язання конфліктів є складним шляхом знаходження компромісного рішення з урахуванням інтересів агентів засобами переговорів.

Третя причина: необхідність залучення відсутнього досвіду. Обмежений набір знань кожного з агентів змушує взаємодіяти з іншими агентами для досягнення як власних та і спільних цілей. Але саме тут з’являється множина різних ситуацій. Іноді агент здатний виконати завдання самостійно, або є здатність вирішення завдання самостійно, але залучення допомоги у вигляді кооперації дозволить вирішити завдання більш ефективно. А іноді, агент не може впоратися з завданням поодинці і вимушений залучати на допомогу інших агентів. В кожній конкретній ситуації агенти повинні обирати той чи інший тип взаємодії та проявляти зацікавленість в співпраці для досягнення спільної мети.

Наступна причина – взаємні зобов'язання. Саме вони дозволяють упорядкувати взаємодію агентів в мультиагентній системі. Зобов’язання надають можливість планування та прогнозування дій агентів у системі. Можливо сформулювати поведінку агентів, виділивши наступні групи: зобов'язання перед іншими агентами; зобов'язання агента перед групою; зобов'язання групи перед агентом; зобов'язання агента перед самим собою.

Саме формалізація у моделі цілей, зобов'язань, намірів, та інших важливих характеристик і є каркасом ментальної моделі кожного інтелектуального агента. Така формальна модель забезпечує мотивовану поведінку агента в автономному режимі.

Множена наданих причин взаємодії агентів та сполучення цих причин може визначати форми взаємодії між агентами в мультиагентній системі.

Інтеграція досвіду окремих агентів без залучання будь-яких механізмів координації є формою простого співробітництва. Форма співробітництва, що координується, передбачає що агенти змушені погоджувати свої дії для виконання завдань ефективного використання ресурсів і власного досвіду. А непродуктивне співробітництво передбачає коли агенти спільно використовують ресурси або розв'язують загальну проблему, не обмінюючись досвідом і заважаючи один одному.

1.1 2.6 Постановка завдання

Метою кваліфікаційної роботи є розробка, дослідження та аналіз імітаційної моделі взаємодії інтелектуальних агентів, що дозволяє здійснювати навчання інтелектуальних агентів засобами нейронної мережі. Мультиагентна система повинна мати власне середовище і модель взаємодії інтелектуальних агентів. Середовище є замкнутим та має обмеження за розмірами. Агенти не можуть вийти за межі середовища. Для підтримки свого існування кожен агент в системі повинен виконувати завдання пошуку їжі. Частинки їжі характеризуються координатами (x, y). Кожен з агентів характеризується положенням (x, y) і вектором напрямку руху у середовищі, розміри якого обмежені. Завданням інтелектуальних агентів є задача пошуку та збору їжі. Ефективність функціонування визначається кількістю зібраної їжі.

Імітаційна модель управління агента об'єднує в собі дві моделі:

- модель поведінки інтелектуального агента, яка визначає алгоритми дії агентів;
- модель оцінки дій інтелектуального агента, що визначається цілями поведінки агентів.

Модель поведінки інтелектуального агента повинна бути заснована на нейронні мережі, виходи якої визначають дії агента. Навчання нейронної мережі здійснювати за допомого генетичного алгоритму.

Модель оцінки дій інтелектуального агента визначає навчальний сигнал, який повинен надавати інформацію щодо стану агента.

При здійсненні графічної візуалізації запропонованої імітаційної моделі використовувати динамічну зміну часу (у тактах). Визначення що до стану агенту здійснювати в порівнянні зі станом на попередньому такті.

3 МОДЕЛЮВАННЯ ВЗАЄМОДІЇ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ

3.1 Моделювання стратегії руху агента

При здійсненні моделювання взаємодії інтелектуальних агентів в мультиагентній системі, перш за все, необхідно визначити стратегію поведінки агентів у системі, яку можливо уявити у вигляді декомпозиції стратегії руху на декілька рівнів. Стратегію руху можливо поділити на шари, які передбачають наявність ієрархії трьох рівнів (рис. 3.1).

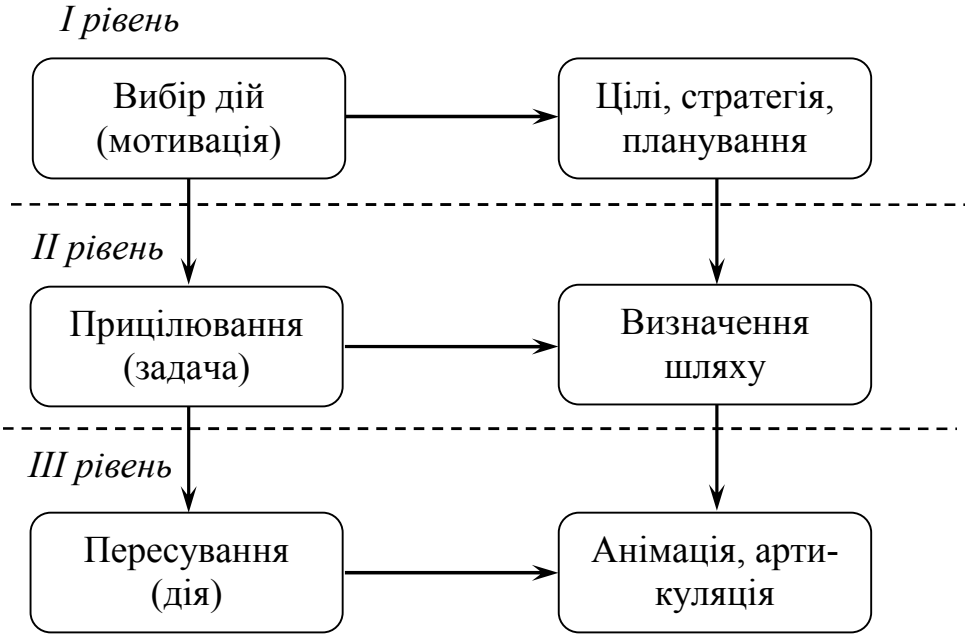

Рисунок 3.1 – Ієрархічна структура стратегії руху агентів

Стратегія руху агентів поділена на три рівні: вибір дій, прицілювання, пересування, але можливо і використання інших стратегій руху об'єктів. Наприклад, автори Бламберг і Галейан [8], поділяють стратегію руху об’єктів на наступні шари: мотивація, задача і дія. Наданий поділ стратегії саме на такі шари використовується лише для реалізації стратегії пересування, але не доцільний для реалізації таких стратегій, як ведення розмови, реалізації чатбота. Множина інших задач потребує зовсім іншої розбивки на шари.

У ієрархічній структурі, що описує поведінку агента, пересування є нижчим рівнем. Поняття шару пересування можливо трактувати як фізичне втілення інтелектуального агента. Шар пересування перетворює сигнали з шару прицілювання у рух тіла агента. Фізичною моделлю тіла об’єкта накладаються деякі обмеження рухів. Наприклад, взаємодія імпульсу і сили мають обмеження, які враховують межу всіх сил, накладених на це тіло. Стратегії прицілювання можливо реалізувати таким чином, що вони стають зовсім незалежними від особливостей схеми пересування, але в реальних ситуаціях необхідно організувати компенсацію різниці між спритністю і індивідуальними характеристиками об'єкта пересування. Організація такої компенсації в стратегіях прицілювання досягається введенням певних «підгінних» параметрів на поточну модель або з застосуванням механізмів адаптації та самокалібрування. Прикладом може стати ситуація, коли досвідчений водій швидко адаптується до особливостей керування новим автомобілем, до його габаритних розмірів. У першому випадку, стратегії прицілювання можливо задати швидкість руху за замовчуванням, яку у певній ситуації об’єкт не може перевищувати. Другий підхід, який можливо реалізувати, пропонує об’єкту пригальмувати, доки не буде одержано результат аналогічний першому.

Моделювання агента може мати дуже просту стратегію пересування: до моделі пересування прикріплено статичний об’єкт, або попередньо анімований, що здатен циклічно переміщуватись у просторі. З іншого боку, агент може бути представлений як фізична, динамічно балансуюча модель об'єкта, реалізуючи пересування та рух, як реалістичну анімацію. Але можливо об’єднати ці два підходи та отримати агента, який має просту модель руху і адаптивну анімаційну модель. Пересування агента може бути організовано з застосуванням моделі, що має набір анімованих частин, що імітують пересування. Наприклад: рухатись, стояти, бігти, повернути ліворуч, повернути праворуч, що дозволить обирати ці частини дискретно, або змішувати їх [8].

3.2 Моделювання руху засобів пересування агента

Розглянемо модель пересування агента на примітивному засобі пересування, де перш за все треба визначити об’єкт, що і буде засобом пересування. Засіб пересування є ідеалізованим і охоплює весь спектр об'єктів, що рухаються: автомобілі, літаки, кораблі, коні і т.п. Також в шар пересування слід включити частину агента, наприклад ноги. В цій моделі нехтуємо розмірами агента, визначаємо, що наш агент є матеріальною точкою. Це значно спрощує фізичну модель, що надає можливість легко обчислювати наприклад швидкість такої точки, бо не враховується інерція. Але з іншого боку, така модель втрачає реалістичність: матеріальні точки у реальному світі не існують. Фізичний об’єкт повинен мати масу, ненульовий радіус, момент інерції. Але для зручності дослідження поведінки і стратегій було обрано саме таку спрощену модель.

Припустимо, що фізично матеріальна точка, запропонована у моделі, визначається своїм положенням та масою, а також має параметр швидкості. Зміна значення параметра швидкості відбувається при застосуванні до точки, що і є агентом, деяких сил. При цьому, якщо розглядати засіб пересування, ці сили обмежені і впливають самі на себе. Наприклад, сила, яка відповідна за швидкість пересування об’єкта машина – це «тяга». Така сила створюється двигуном «машини» і обмежена величиною його елементів живлення. В моделі, що є примітивною, визначимо максимальну силу агента. Засоби пересування характеризуються їхньою максимальною швидкістю, що і є обмеженням, яке виникає в силу взаємодії прискорення і гальмування. Прискорення обмежене величиною тяги машини, а гальмування виникає за рахунок тертя. Засіб пересування у моделі повинен мати параметр орієнтації, що визначається поточною позицією об'єкта, вирівняного по швидкості локальної системи координат, до якої геометрична модель засобу пересування приєднана [7].

Визначимо характеристики, які повинна мати модель агента засобу пересування: маса – (mas – скаляр); позиція – (position – вектор); швидкість – (speed – вектор); максимальна швидкість – (max_speed – скаляр); максимальна сила – (max_force – скаляр); орієнтація – (N базисних векторів).

Для побудови тривимірної моделі значення векторів position і speed складаються із трьох компонентів. В тривимірної моделі значення параметра орієнтації має три вектори (матриця 3x3). При моделюванні двомірного засобу пересування кожний із цих векторів містить дві компоненти, і значення параметра орієнтації складається із двох двомірних базисних векторів, або ж може бути представлений як одиничний скалярний кут напрямку моделі.

Фізичну основу примітивної моделі агента засновано на прямому методі Ейлера. На кожному кроці симуляції сили прицілювання, визначені алгоритмами поведінки і обмежені параметром max_force, застосовуються до матеріальної точки. У результаті одержуємо прискорення, еквівалентне відношенню величини сили прицілювання до маси «машини». Отримане прискорення додаємо до старого значення швидкості, при цьому обмежуючи її значенням max_speed. Далі, додаємо значення швидкості до старої позиції об'єкта: Steering force = truncate (steering direction, max_force); Steering force = truncate (steering direction, max_force); Acceleration = steering force / mas; Speed = truncate (speed + acceleration, max_speed); Position = position + speed.

Спрощена модель агента засобу пересування у локальному просторі має власне вирівнювання по швидкості, яке здійснюється за допомогою інкрементного регулювання параметрів, щодо попереднього тимчасового кроку. При визначені локальної системи координат у моделі прийнято наступні позиції: вектор позиції задає початок координат, а три інших вектори є базисними векторами зазначеного простору.

Для кожного із трьох взаємно перпендикулярних напрямків осей за визначення напрямку і довжини кроку координат відповідальними є саме базисні вектори. Визначення для осей простору: X – пряма (forward), Y – верхня (up), Z – бічна (side). Для вирішення завдання вирівнювання швидкості на

кожному кроці імітації рухів агента та засобу пересування, необхідно забезпечити постійний оберт в новому напрямку базисних векторів. Але можливо вирішити це завдання по-іншому. Замість реалізації обертів, можливо організувати перебудову простору, використовуючи заміщення і апроксимацію. На початку кроку одержуємо нову швидкість, а потім апроксимуємо її до нового прямого базису. Наприклад, старий базис *X* може бути використаний як наближення до нового базису *X*. Для цього будемо використовувати вирази векторного добутку для знаходження нових базисних векторів: New_forward = normalize (speed); Approximate_up = normalize (approximate_up) // необов'язкова умова; New_side = cross (new_forward, approximate_up); New_up = cross(new_forward, new_side). В подальшому виконуємо апроксимацію напрямку *Y* найближчими перпендикулярами до нового напрямку *X,* що забезпечується наявністю дуже малих змін від кадру до кадру в орієнтації об’єкта у моделі. Виходячи з визначення векторного добутку, новий отриманий бічний напрямок *Z* буде перпендикулярним новому напрямку *X*. А отриманий новий верхній напрямок *Y* буде векторним добутком перпендикулярів *X* і *Z*. Таким чином, всі вектори будуть перпендикулярні між собою [9].

Алгоритм вирівнювання по швидкості об’єкта та засобу пересування визначається не тільки параметром орієнтації, а ще і параметром ступінь свободи, що відповідає обертанню навколо осі *X*. Для спрощення моделі, створюючи новий простір, припустимо, що величина вільного обертання об'єкта, залишається постійною. Визначення «вірної» величини такого обертання вимагає створення додаткових евристик, заснованих на конкретному застосуванні конкретної моделі транспортного засобу.

Для здійснення реалізації більш реалістичних моделей пересування потрібні більш складні набори подібних сигналів. Наприклад, автомобіль має керуючі колесо, газ, гальма, а кожна з величин, що характеризує ці сигнали, може бути представлена як скаляр. У цьому випадку стає можлива генералізація сили, яка прицілює, на ці три компоненти: ось *Z* стає величиною прицілювання, а *X* – сигналом прискорення, якщо величина додатна, і сигналом

гальмування, якщо величина від’ємна. Останні відображення осі *X*, у загальному випадку можуть бути не симетричними. Наприклад, звичайний автомобіль може гальмувати тільки у випадку, якщо сила гальмування буде сильніше за силу тяги його двигуна.

3.3. Визначення стратегій поведінки агентів

Агенти з поведінкою, яка заснована на моделі, можуть оперувати із середовищем, яке лише частково піддається спостереженню. Усередині агента зберігається уявлення про ту частину, що знаходиться поза межами огляду. Щоб мати таке уявлення, агенту необхідно знати, як виглядає навколишній світ, як він влаштований. Ця додаткова інформація доповнює «картину світу». Агенти, що мають здатність до навчання (англ. autonomous intelligent agents), мають незалежність і здатність пристосовування до обставин, що можуть змінюватися. Мультиагентна система, де існують інтелектуальні агенти повинна виявляти такі здібності [6]:

- навчання і розвиток досягається у процесі взаємодії з навколишнім середовищем;
- пристосовуватись до змін навколо у режимі реального часу;
- використовувати алгоритми навчання, які здатні швидко навчати на основі великого обсягу даних;
- мати базу прикладів з можливістю її поповнення і подальшого використання;
- мати параметри для моделювання швидкої та довгої пам'яті, віку тощо.
- аналізувати себе в термінах поведінки, помилки та успіху.

Імітаційна модель взаємодії інтелектуальних агентів передбачає використання різних стратегій поведінки та їх комбінацій. Розглянемо основні стратегії поведінки для примітивної моделі агентів, де агент є матеріальною точкою.

Стратегія «Пошук» передбачає, що агент буде направлений у точки у просторі. Ця точка може бути конкретною або випадковою. За цією стратегією, поведінка агента передбачає, що вектор швидкості агента стає спрямованим убік цілі. А сила не є силою тяжіння (гравітації), що не дозволяє створювати орбітальний шлях навколо цілі. Стратегія передбачає наявність бажаної швидкості, яку можливо надати як вектор, що спрямований від агента до його цілі, довжина якого визначається максимальною або поточною швидкістю, що залежить від конкретної потреби. Тоді вектор прицілювання визначається як різниця між бажаною швидкістю і поточною швидкістю агента: Desired_velocity = normalize (position - target) * max_speed ; Steereing = desired_velocity – velocity (рис. 3.2).

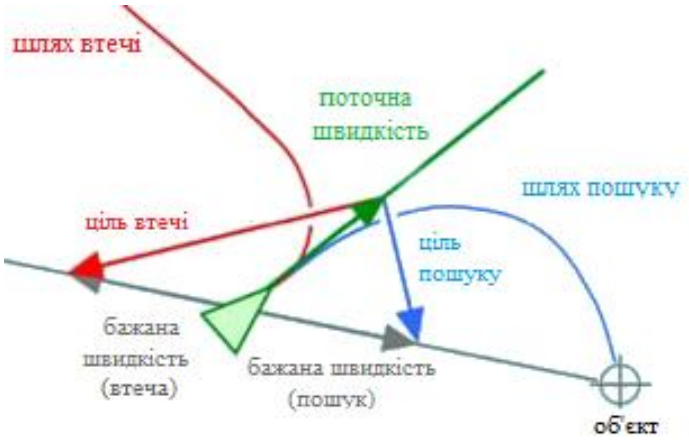

Рисунок 3.2 – Графічне уявлення стратегії втечі та пошуку

Зворотною стратегією є стратегія «Втеча», яка примушує агента здійснювати рух від цілі, змінюючи свою швидкість. Вектор бажаної швидкості спрямований у протилежну сторону (рис. 3.2).

Стратегію «Прицілювання» для примітивної моделі агентів, де агент є матеріальною точкою, можливо визначити засобами геометричних обчислень над вектором бажаної сили прицілювання. Ця стратегія визначає шлях агента, який залежить від його цілі і може бути пошуком, втечею і т.д.

Стратегія «Переслідування» повторює основні засади стратегії пошуку, за винятком того, де в якості цілі переслідування виступає інший агент, який виконує свій рух у системі. Такий алгоритм дій передбачає, що необхідно реалізувати прогнозування майбутнього стану цілі. Треба застосувати метод, який має так званого передвісника, що на кожному кроці імітації виконує перезапуск для обчислення нових координат.

Майбутня позиція агента у час T може бути обчислена шляхом масштабування його швидкості по проміжку часу T і додавання цього відхилення до його поточної швидкості. В такому випадку, стратегія переслідування зводиться до стратегії пошуку, в якій при визначенні цілі буде зазначена передбачувана позиція мети (рис. 3.3).

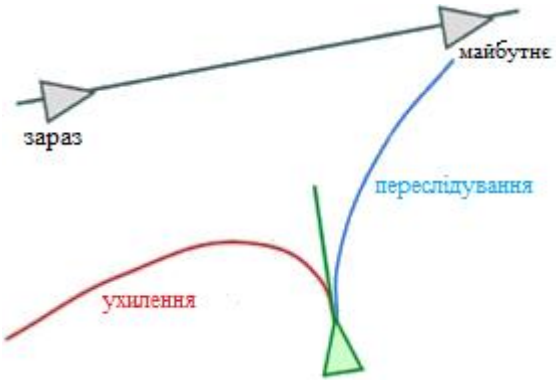

Рисунок 3.3 – Графічне уявлення стратегії переслідування

Для реалізації стратегії переслідування необхідно визначити метод для проведення оцінки інтервалу T. Коли час T є часом до моменту зіткнення об'єктів – все дуже просто. Але на практиці, цей час T неможливо точно передбачити, тому що ціль здійснює непередбачувані рухи, що не дає змоги розрахувати час, коли ж саме станеться зіткнення. Якщо час T уявити у вигляді константи, що не зовсім відповідає реальної ситуації, тоді стратегія переслідування є більш реалістичною в порівнянні з простою стратегію пошуку в якій передбачається, що час T дорівнює нулю. Примітивна оцінка параметра часу T може бути представлена як $T = Dc$, де D – це відстань між агентом і

ціллю переслідування, а *c* –деякий параметр, що настроюється. Більш складною оцінкою інтервалу *T* може бути застосування методу, який здійснює оцінку шляхом додавання до попереднього приклада додаткової умови, що дозволяє визначати положення агента відносно цілі. Де перебуває агент: попереду, з боків – ліворуч або праворуч. Опис цього положення можливо здійснити за допомогою скалярних добутків векторів (між вектором *X* переслідувача, вектором *У* переслідуваного і відступом у наступну позицію переслідуваного). Слід стежити, щоб час *T* не дорівнював нулю. В цьому випадку маємо ситуацію, коли переслідувач буде знаходитись прямо перед своєю ціллю [6].

Стратегію переслідування та пошуку можливо описати і іншим алгоритмом дій. Коли агент знаходиться на прямому курсі зіткнення з ціллю, маємо ситуацію, вектор напрямку агента стає постійним усередині його локального простору, що дозволяє агенту здійснювати рух прямо до цілі, просто зберігаючи заданий напрямок.

Стратегія «Відхилення» використовує алгоритм стратегії переслідування, за винятком того, що замість пошуку агент використовує втечу, для здійснення перенаправлення агента в протилежну сторону від прогнозуємої позиції цілі. Для реалізації алгоритмів стратегії переслідування та відхилення застосовують методи керування. Методи, описані в цій роботі, не оптимальні. У дійсних системах, відхилення дуже часто навмисне неоптимальне. Це робиться для того, щоб зробити їх більше непередбаченими, заважаючи класичним стратегіям переслідування [10].

Стратегія «Переслідування з відступом» передбачає використання методу, який прокладає шлях агента до місця поруч із ціллю. Прикладом реальних життєвих подій може бути рух космічного корабля, який виконує стикування. Алгоритм такої стратегії передбачає вираховування величини відступу від майбутнього положення цілі, яке постійно змінюється та задана радіусом R. Для досягнення заданої оцінки виконується алгоритм стратегії пошуку (рис. 3.4).

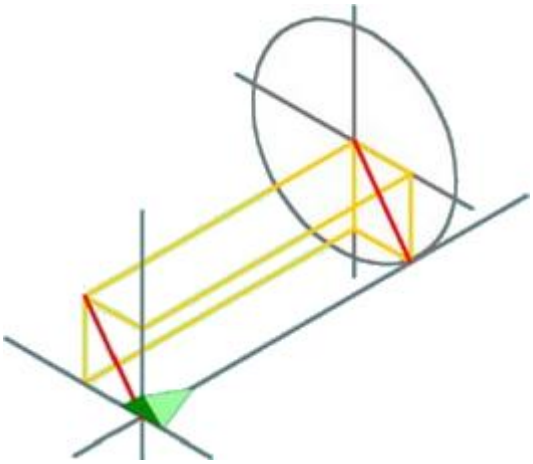

Рисунок 3.4 – Графічне уявлення стратегії переслідування з відступом

Для організації пошук точки відступу, по-перше необхідно знайти локальну позицію цілі для агента в майбутньому. Подальшим кроком буде проекція цілі на площину *ZY* цього агента. Нормалізуючи цей відступ, треба здійснити масштабування його на параметр *R,* та здійснити додавання цього вектору у поточне положення цілі.

Якщо розглядати стратегію «Прибуття», то можливим стає застосування стратегії пошуку, але лише для випадку, коли агент знаходиться досить далеко від цілі пошуку(рис. 3.5).

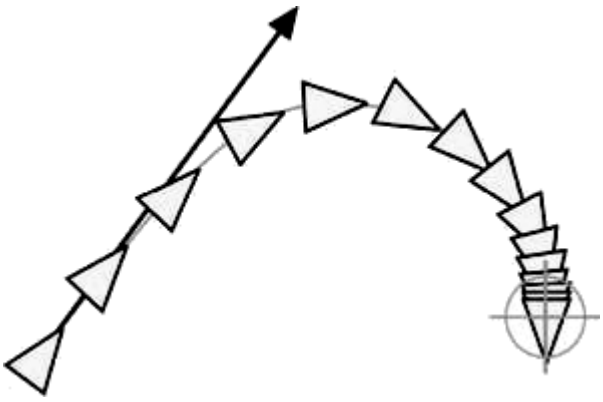

Рисунок 3.5 – Графічне уявлення стратегії прибуття

Але згадуючи, що в стратегії пошуку при досягненні цілі агент проходить скрізь неї на максимальній швидкості, то для стратегії прибуття така по-

ведінка недоцільна. Тому для цієї стратегії, агент повинен сповільнюватися в міру наближення до цілі, з подальшим гальмуванням до повної зупинки і виконати поєднання з потрібною ціллю. Значущим параметром для даної поведінки є значення відстані з якої починається гальмування. Алгоритм такої поведінки схожий з алгоритмом пошуку, де бажаний вектор швидкості визначається напрямком руху агента до цілі [6].

Якщо розглядати окружність, де відбувається гальмування, то з зовнішньої сторони швидкість обмежена значенням максимальної швидкості, а в середині – швидкість зменшується лінійно, наближаючись до нуля.

Target_offset = target – position; Distance = length (target_offset); Ramped_speed = max_speed * (distance / slowing_distance); Clipped_speed = minimum (ramped_speed, max_speed); Desired_velocity = (clipped_speed / distance) * target_offset; Steering = desired_speed – speed. Наприклад, машина прямує до деякого перехрестя доріг, а побачивши червоне світло поступово зупиняється.

Для здійснення маневрування в просторі з деякою кількістю перешкод використовують стратегію «Відхилення від перешкод», яка забезпечує пересування агента у середовищі уникаючи зіткнень з цими перешкодами. Стратегія відхилення від перешкод та стратегія втечі не є аналогами. Реалізуючи алгоритм стратегії втечі, агент завжди віддаляється від цілі, а алгоритм стратегії відхилення від перешкод змушує агента віддалятися лишу у випадку, коли перешкода з’являється перед агентом.

Щоб визначити алгоритм для стратегії відхилення від перешкод введемо наступні припущення: агент і перешкоди у просторі будуть мати геометрію сфер, що надасть змогу спростити алгоритм. Крім того, відхилення від перешкод у середовищі не обов'язково повинно бути пов'язано з наявністю колізій між об'єктами. Наприклад, літак здійснює маневр, що забезпечує ухилення від зіткнення з горою. Якщо геометрію літака та гори уявити у вигляді сфери відповідних розмірів, цього буде достатньо для уникнення зіткнення цих об’єктів. Далі геометрична побудова алгоритму стратегії відхилення від

перешкод використовує основу від стратегії переслідування з відступом, де використовується система координат агента. Уздовж осі *X* агента визначимо деякий циліндр, діаметр якого дорівнює діаметру сфери. А довжина циліндра визначається як сукупність його швидкості та швидкості повороту уздовж осі *Y*. Для агента не становлять загрози перешкоди які знаходяться поза циліндром. А стратегія відхилення визначає будь-яку загрозу (перешкоду) на тимчасово визначеному кроці, здійснює перевірку можливості перетину агента і циліндра. Перевірка перетинання циліндра з агентом досить швидко розраховується: для цього досить визначити центр кожної сфери перешкоди. Локальний центр сфер проектується на площину *XZ*. Зіткнення не станеться у випадку, коли двомірна відстань від цієї точки до центра координат агента більше, ніж сума радіусів агента і перешкоди [6]. Не розглядаються перешкоди, які мають досить значну відстань від циліндру. Для всіх значимих перешкод проводиться перевірка можливості перетину їх сфер з лінією руху агента. Найнебезпечнішою вважається перешкода, що перетинає ось *X* агента і знаходиться найближче. Сила прицілювання, що буде уникати зіткнення, буде дорівнювати від'ємному значенню проекції на осі *XZ* центра сфери перешкоди (рис. 3.6).

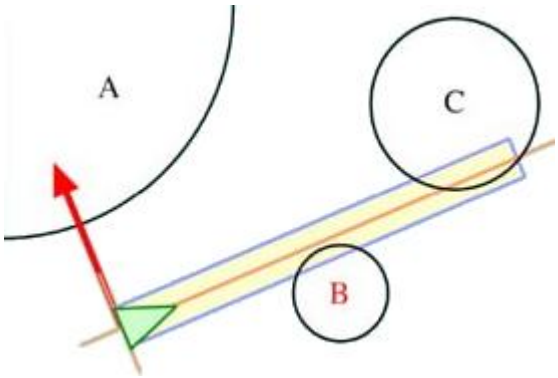

Рисунок 3.6 – Графічне уявлення стратегії відхилення від перешкод

На рисунку перешкода *A* не перетинає циліндр агента, а перешкоди *B* і *C* перетинають. Так як перешкода *B* знаходиться ближче до агента, використання сили прицілювання направить агента уліво від цієї перешкоди. Значен-

ня, що повертає цей аналіз поведінки дозволяє уникнути «поточних погроз» або визначає, що у даних час погрози зіткнення немає і коригувати рух агента непотрібно. Наступний крок повинен дати відповідь на питання: чи існують перешкоди на шляху руху агента до цілі. Наприклад, гора, яка розташована за аеропортом, – диспетчера не хвилює, але гора, яка розташована між аеропортом і літаком, вельми важлива.

Стратегія «Блукання» реалізує поведінку випадкового прицілювання, забезпечуючи нестійкий рух. Додавання сили прицілювання на кожному кроці рухів агента в імітації і забезпечить функціонування такої стратегії поведінки агента. При цьому підході необхідно реалізувати можливість збереження напрямку прицілювання, намагаючись при здійсненні кожного кроку робити лише невеликі випадкові відступи від визначеного напряму. При виконанні одного з кроків агент може повернути вгору і праворуч, другий крок – зберігаючи приблизно той же напрямок, здійснює оберт [6]. Сила прицілювання буде мігрувати випадково від одного напрямку до іншого без значних коливань значень. При геометричному трактуванні цієї стратегії необхідно зафіксувати силу прицілювання на поверхні сфери, розташованої прямо перед агентом, яка буде спроектована на площину *XZ* (рис. 3.7).

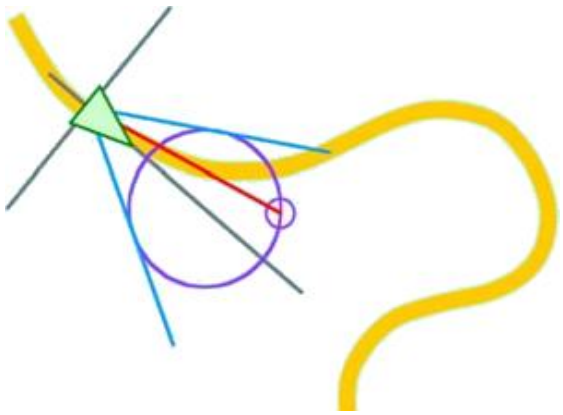

Рисунок 3.7 – Графічне уявлення стратегії блукання

Для розрахунку сили на наступному кроці, необхідно додати випадковий зсув сили до попереднього значення. Ця сума буде обмежена поверхнею сфери, а радіус цієї сфери буде мати максимальне значення сили блукання, а значення зсуву буде дорівнювати величині блукання.

Стратегія «Проходження по шляху» забезпечує агенту координацію руху уздовж визначеного шляху, наприклад по шосе, в коридорі або тунелі. Такий рух схожий на рух людей по коридору: шлях агента проходить поруч, а найчастіше паралельно центральної лінії тунелю, але агент вільний відхилитися від цієї лінії. Шлях є узагальненим такими поняттями як хребет шляху і радіус шляху. Хребет шляху може бути представлений набором з'єднаних між собою сегментів ліній [6].

Таким чином, шлях представлений як циліндр певного радіуса, прокладеного уздовж заданого «хребта». Метою даної поведінки буде рух агента уздовж шляху. При цьому він повинен буде залишатися усередині заданого радіуса хребта. Якщо агент яким-небудь образом покине зазначений шлях, він повинен буде спочатку повернутися на нього, а потім продовжити рух уздовж нього.

Для того, щоб здійснювати визначення сили прицілювання в стратегії проходження шляху треба ввести передвісник, який надасть значення майбутнього положення агента у просторі, який буде залежати від швидкості руху агента. Прогнозована позиція агента проектується на найближчу опорну точку хребта и якщо відстань менша ніж радіус шляху, то дії агента та шлях вважаються не потребуючими коригування. Якщо ця умова має негативне значення, то вважається що агент відхилився від заданого шляху і для його повернення використовується алгоритм стратегії пошуку.

В даному випадку ціллю стає проекція майбутньої найближчої опорної точки хребта, а заданий шлях може бути як орієнтованим, так і неорієнтованим (рис. 3.8).

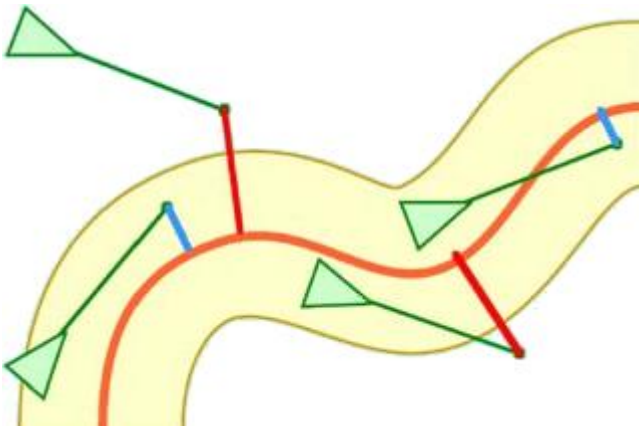

Рисунок 3.8 – Графічне уявлення стратегії проходження по шляху

Окремими випадками стратегії проходження шляху є поведінки «проходження по стіні» та «стримування». Алгоритм «проходження по стіні» реалізує алгоритм наближення до зазначеної стіни, поверхні, шляху, і подальше проходження уздовж неї на певної відстані. Алгоритм реалізації руху на відстані спирається на алгоритм стратегії «переслідування з відступом». Алгоритм стратегії «стримування» реалізує можливість агента здійснювати рух в визначеної обмеженої області. Стратегія «проходження шляху» має аналогічний алгоритм, що і стратегія «стримування», де область визначається циліндром навколо хребта шляху (рис. 3.9).

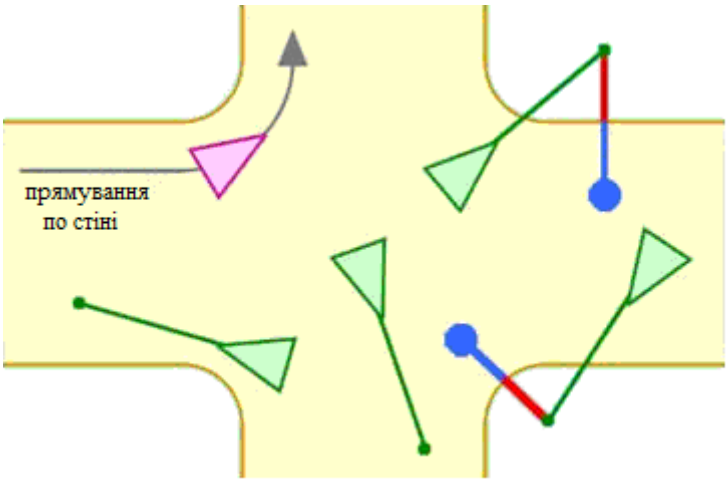

Рисунок 3.9 – Графічне уявлення стратегії проходження по стіні

Наприклад, фігуристи катаються на льодовій арені, яка обмежена бортами і має визначений розмір. Реалізація такої поведінки передбачає визначення майбутньої позиції агента. Якщо агент знаходиться усередині заданої області, то корегування руху і області не треба. В іншому випадку – агент рухається усередину цієї області, виконуючи алгоритм стратегії «пошуку», де ціллю визначається будь-яка точка усередині області.

Застосування стратегії «Проходження потоку» визначає алгоритм поведінки агента при ситуації знаходження агента усередині оточення. Ця стратегія найбільш затребувана при реалізації анімації агентів, яка позбавляє розробників у програмуванні анімації [6]. В ігровій індустрії такі стратегії називаються дизайнерами рівнів, а в комп'ютерній анімації їх називають планувальниками сцен. Алгоритм стратегії «проходження потоку»передбачає, що агент хоче вирівняти напрямок свого руху за рахунок векторного поля або поля сили. Поле сили визначає відображення положення у просторі об'єкта і вектор потоку (рис. 3.10).

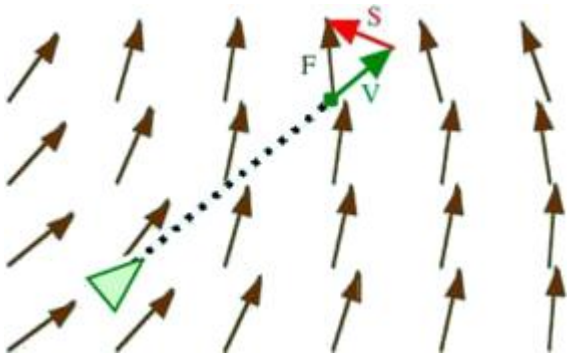

Рисунок 3.10 – Графічне уявлення стратегії проходження потоку

Наприклад, є деяка підлога, на яку нанесено множена стрілок Така намальована карта зі стрілками є планом руху агента по поверхні. Алгоритм проходження потоку передбачає визначення майбутньої позиції агента і здійснює перевірку стану векторного поля в цій точці. Напрямок потоку, що визначається вектором *F,i* є «бажана швидкість», а напрямок прицілювання, що

визначається вектором S – це різниця між поточною швидкістю (вектор V) і бажаною швидкістю.

Алгоритм стратегії «Відхилення від колізій у натовпі» передбачає вирішення ситуації, коли агенти, які рухаються в схожих напрямках, можуть перетинатись один з одним. Рухаючись серед значної кількості агентів (наче серед натовп людей на площі), для запобіганню зіткнень, агенти вимушені передбачати можливі колізій і змінювати як напрям руху так і швидкість руху. Для реалізації такої стратегій, дії агента повинні враховувати поведінку всіх найближчих агентів навколо: з якою швидкістю, в якому напрямку вони рухаються, через який час можливе зіткнення, якщо відстань один від одного дуже мала і саме де можливе зіткнення [6].

По-перше, визначається найближча з можливих колізій. Потім, коригується сила прицілювання для уникнення колізії (повернення агента у бік, гальмування або навпаки – прискорення). Головна мета – потрапити у точку колізії до або після того, як вона повинна по прогнозам відбутися. На рисунку 3.11 представлено ситуацію, коли агент, що рухається праворуч, вирішив загальмувати і повернути ліворуч, для уникнення колізії. А в цей самий час інший агент вирішує прискоритись і потім здійснити поворот ліворуч.

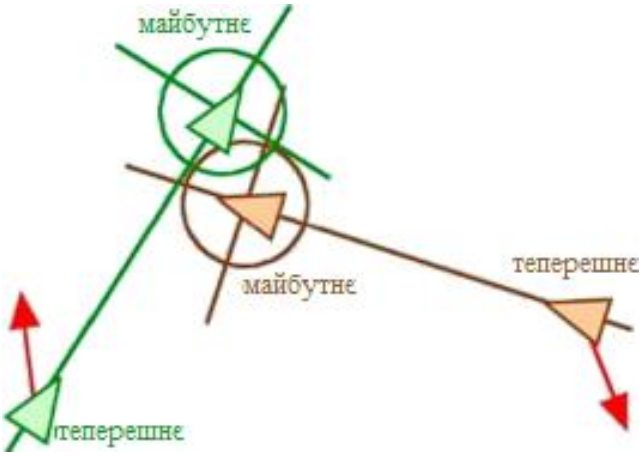

Рисунок 3.11 – Графічне уявлення стратегії відхилення від колізій у натовпі

Розглянемо також стратегії поведінки агентів, що передбачають визначення алгоритмів дій у групі. До таких стратегій відносяться стратегія «розподіл», стратегія «зчеплення», стратегія «вирівнювання». В цих стратегіях визначаються дії агентів по відношенню до своїх локальних сусідив, ігноруючи всіх інших агентів у системі. Сусідами визначаються агенти, які визначаються відстанню, що входить в «кут огляду» агента.

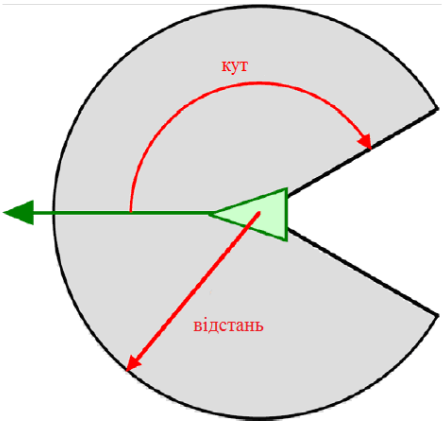

Рисунок 3.12 – Графічне уявлення визначення сусідів

Алгоритм стратегії «Розподіл» передбачає, що агент може мати певну дистанцію зі своїми сусідами і дозволяє уникнути скуплення агентів. Для визначення сили розподілу, необхідно вичислити всіх можливих сусідів. Це можливо реалізувати перебором всіх можливих варіантів сусідив навколо або використовуючи механізм «кешування», що прискорить пошук [6].

Для кожного визначеного сусіда відштовхуюча сила обчислюється як різниця між позицією нашого агента та позицією сусіда та подальшою нормалізацією отриманого значення. І так для кожного з випадків. Потім до отриманого значення застосовується значення ваг $1/r$, яке отримано експериментальним шляхом, і не є фундаментально доведеною величиною. Отримані сили відштовхування підсумовуються і визначаються як значення сили прицілювання (рис. 3.13).

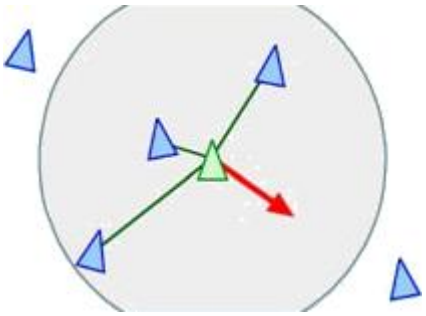

Рисунок 3.13 – Графічне уявлення стратегії розподілу

Реалізація стратегії «Зчеплення» передбачає надання можливості агентам долучатись до сусідів агентів та формувати групи. Для визначення сили зчеплення розраховують середню позицію всіх сусідніх агентів (визначають так званий цент мас), потім направляють силу в напрямку розрахованого центр мас. Для цього знаходять різницю між позицією зазначеного агента та середньою позицією. Точка центра мас і буде ціллю поведінки (рис. 3.14).

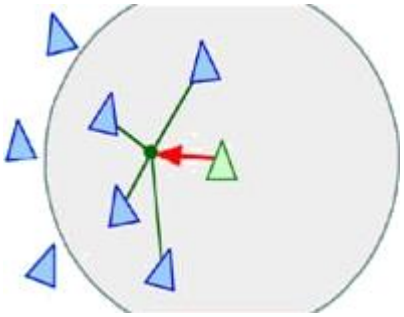

Рисунок 3.14 – Графічне уявлення стратегії зчеплення

Алгоритм стратегії «Вирівнювання» передбачає, що агенти здатні вирівнювати вектор руху швидкості в тім же напрямку з іншими агентами. Розрахунок цієї сили виконується, як усереднення всіх векторів швидкостей сусідніх агентів, що дозволяє отримати так звану бажану швидкість. А вектор прицілювання буде дорівнювати різниці між цим усередненням і швидкістю визначеного агента. Такий алгоритм передбачає вирівнювання напряму руху (повертання) таким чином, щоб бути вирівняним відносно сусідів (рис.3.15).

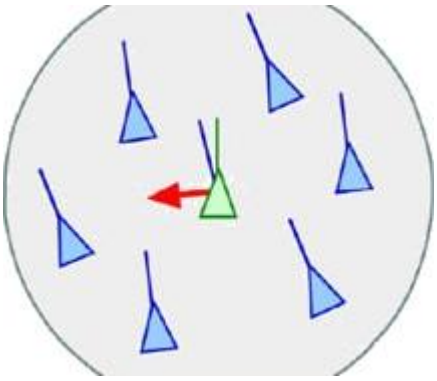

Рисунок 3.15 – Графічне уявлення стратегії вирівнювання

Ці розглянуті три групові стратегій руху агентів в мільтиагентній системі можуть поєднуватись в стратегії «Зграя» для здійснення імітації руху стад, груп, зграй та інших групових моделей.

Інколи в імітаційних моделях використовують не кожну стратегію, як окрему, а їх поєднання в одну єдину силу прицілювання. Для організації контролю руху агентів в таких імітаційних моделях спочатку здійснюють нормалізацію цих алгоритмів, масштабування до певних ваг і наприкінці виконують сумування їх між собою. Результатом є групова модель, що характеризується наступними параметрами: вага (для комбінування векторів), відстань і кут для кожного з агентів в мультиагентній системі [7].

Алгоритм стратегії «Слідування за лідером» передбачає організацію руху агента за об’єктом, який є обраним як лідер.

Всі агенти, що хочуть бути поруч із лідером, який не є агентом із групи, повинні стежити за тим, щоб не з’явитися на його шляху. Якщо число агентів слідування більше одного, то йому необхідно уникати зіткнень із сусідами. Алгоритм цієї стратегії використовує поведінку агентів стратегії «Прибуття», намагаючись досягти певної точки, сповільнюючи рух при наближенні до неї. Точкою «прибуття» є точка, що знаходиться на малій відстані за лідером. Якщо агент слідування перебуває в певному квадраті перед лідером, то він повинен буде спробувати втекти від нього, а потім поверну-

тися до поведінки «прибуття». Для уникнення зіткнень між собою ця стратегія передбачає застосування стратегії поділу (рис. 3.16).

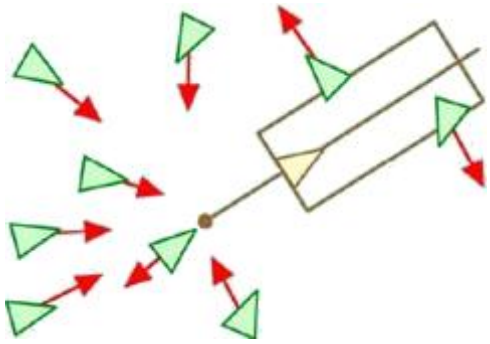

Рисунок 3.16 – Графічне уявлення стратегії проходження за лідером

Алгоритм стратегії «Клинчик» дозволяє агентам здійснювати рух схожий на прохід гравці у футболі при блокуванні передачі м’ячу іншим гравцям команди. Для реалізації цього алгоритму прогнозуємо майбутні позиції двох агентів, розраховуємо точку перетинання між цим двома позиціями, і застосовуємо стратегію «пошук», для того, щоб досягти цієї точки [6].

Стратегія «Укриття» передбачає реалізацію алгоритму стратегії «пошук» після того, як визначено цільову позицію, що знаходиться із протилежної сторони будь-якої перешкоди.

В сучасних мультиагентних системах використовуються стратегії руху агентів, що засновані на ройових методах, що передбачають наявність моделей поведінки рою (моделі ройвого руху). Будь-які складні дії пропонується виконувати за допомогою різних комбінацій і маніпуляцій з базовою стратегією «пошук», яка наявна у всіх моделях.

Однак, якщо розглядають поведінки інтелектуальних агентів, які обучаються і мають мету, то краще моделювати їх поведінки за допомогою нейронних мереж (НМ).

4 МОДЕЛЮВАННЯ ВЗАЄМОДІЇ АГЕНТІВ ЗА ДОПОМОГОЮ НЕЙРОННОЇ МЕРЕЖІ

Метою кваліфікаційної роботи є побудова імітаційної моделі багатоагентної системи у якій наявні інтелектуальні агенти та середовище у якому вони діють. Інтелектуальний агент виконує низьку дій, які приводять до досягнення поставленої мети. Саме весь набір дій, що призводить інтелектуального агента до цілі і формує поведінку агента, а правила взаємодії агентів між собою у середовищі формують набір стратегій для досягнення цілі. Вважається, що цілеспрямована інтелектуальна поведінка агента характеризується наступними чотирма чинниками [11]:

– критерієм успішності поведінки;
– знаннями агента про середовище, набуті раніше.
– дії, які можуть бути виконані агентом;
– послідовність сприйняттів або історія сприйняттів.

З урахуванням цих чинників, для кожної можливої послідовності сприйняттів інтелектуальний агент повинен вибрати дію, яка максимізує його показники поведінки, з врахуванням інформації, наданої послідовністю сприйняттів та знаннями про середовищі, якими володіє агент.

Середовище існування агентів характеризується розмірами (шириною і висотою), а також кількістю агентів та часток їжі. Середовище є замкнуним, що не дозволяє агентам вийти за межі середовища. Наявні у середовищі частинки їжі мають координати (x, y).

Кожен агент характеризується становищем (x, y) і вектором напрямку руху. Задача групи агентів – збирати їжу. Ефективність розглядається виходячи з сумарної кількості зібраної їжі.

В імітаційній моделі управління агентами здійснюється засобами двох модулів:

– модуль поведінки, який визначає дії агента;
– модуль оцінки дій, який визначає цілі поведінки агентів.

Імітаційна модель передбачає, що модуль поведінки застосовує засоби нейронної мережі, виходи якої визначають дії агента. Поведінка агентів визначається нейронною мережею.

Модуль оцінки дій визначає покращився або погіршився стан агента на даному такті часу в порівнянні з попереднім тактом при виконанні деяких передбачуваних дій.

4.1 Математична модель нейрона

Штучна нейронна мережа (ШНМ) є спрощеною моделлю мозку людини і являє набір елементів, з'єднаних між собою певним чином, так щоб між ними забезпечувалася взаємодія. Ці елементи також називаються нейронами або вузлами. Вони являють собою примітивні процесори, обчислювальні можливості яких зазвичай обмежуються деяким правилом комбінування вхідних сигналів і правилом активації, що дозволяє обчислювати вихідний сигнал за сукупністю вхідних сигналів. Вихідний сигнал може надсилатися іншим елементам по зважених зв'язках, з кожним з яких пов'язаний ваговий коефіцієнт або вага. Залежно від вагового коефіцієнта переданий сигнал або посилюється або пригнічується. Перевага ШНМ полягає в тому, що вона може автоматично здобувати знання в процесі навчання і має здатність узагальнення отриманих знань. Однак, навченість одночасно є і недоліком, так як аналіз навченої нейронної мережі досить складний і має свої нюанси. Але одна із самих позитивних якостей використання ШНМ полягає в тому, що хоча елементи такої мережі мають обмежені обчислювальні можливості, але вся мережа в цілому, що складається як правило, з великого числа таких елементів, здатна виконувати досить складні завдання [12].

Одиницю обробки інформації в штучних нейронних мережах являє собою нейрон, а в основі штучних нейронних мереж лежить модель нейрона (рис. 4.1).

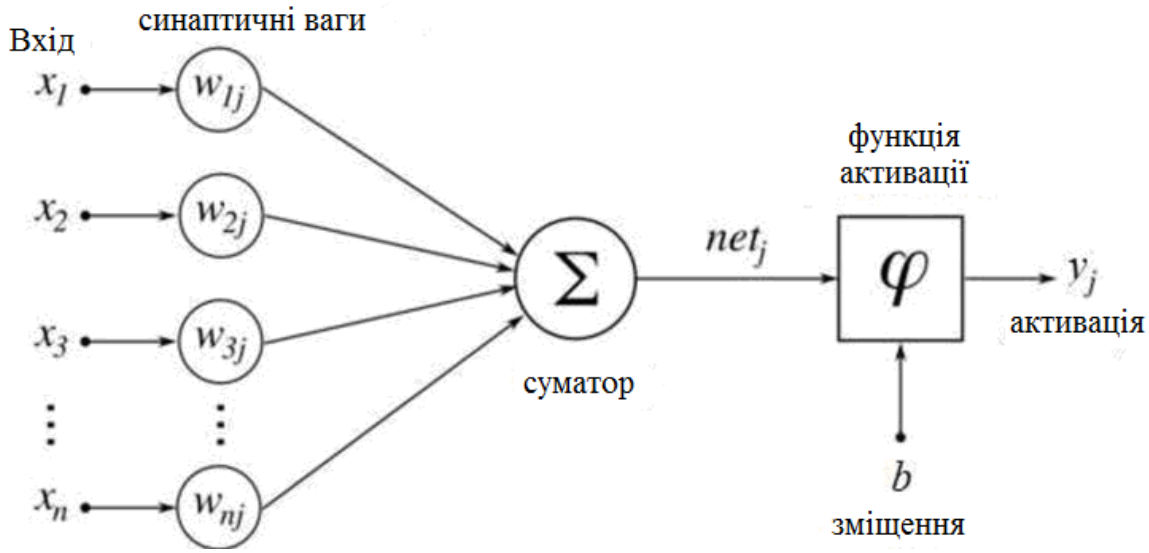

Рисунок 4.1 – Модель нейрона

Модель функціонування нейрона передбачає, що на вхід нейрон отримує вектор *x*. Його компоненти помножуються на відповідні ваги та додаються, а наприкінці, ще додається зміщення *b*. До зваженої суми застосовується функція активації. У моделі можливо визначити наступні головні елементи [12]:

– набір синапсів (зв'язків), де кожен має власну вагу. Сигнал x j, отриманий на вході синапсу j, що є пов'язаним з нейроном k, помножується на вагу w k j.

– суматор додає вхідні сигнали, зважені щодо відповідних синапсів нейрона.

– функція активації виконує обмеження амплітуди вихідного сигналу нейрона, діапазон амплітуд виходу нейрона яких знаходиться в інтервалі [0,1] або [–1, 1] і є нормалізованим (рис. 4.2).

Функціонування деякого нейрона *k* у математичному вигляді можливо визначити наступними рівняннями:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{4.1}$$

$$y_k = \varphi(u_k + b_k), \tag{4.2}$$

де $x_1, x_2, \ldots, x_m$ – вхідні сигнали;

$w_{k1}, w_{k2}, \ldots, w_{km}$ – синаптичні ваги нейрона k;

u_k – лінійна комбінація вхідних впливів;

b_k – поріг;

φ(*) – функція активації;

y_k – вихідний сигнал нейрона.

Після синаптичний потенціал використання порога b_k в моделі, можливо розрахувати за наступним рівнянням:

$$v_k = u_k + b_k. \tag{4.3}$$

Зовнішнім параметром штучного нейрона k є поріг b_k. З виразу (4.3) та формули (4.2) можливо отримати:

$$y_k = \varphi(v_k). \tag{4.4}$$

Представлена у формулах (4.2, 4.4) функція активації φ(v) здійснює визначення вихідного сигналу нейрона та може мати різний вигляд (рис.4.2).

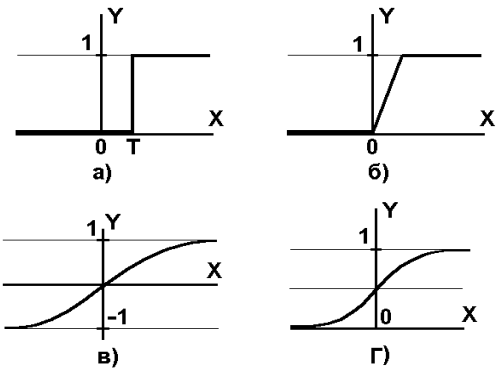

Рисунок 4.2 – Графічне подання функції активації: а) функція одиничного стрибка; б) лінійний поріг (гістерезис); в) сигмоїдальна функція; г) гіперболічний тангенс.

Можливо визначити сигмоїдальну функцію:

$$\varphi(v) = \frac{1}{1 + \exp(-av)}, \tag{4.5}$$

де a – є параметр нахилу сигмоїдальної функції, засобами якого здійснюється керування крутизною функції. Перевагою використання цієї функції полягає в тому, що функція має просту похідну та є диференційована на всій осі абсцис:

$$\varphi'(v) = a\varphi(v) \cdot (1 - \varphi(v)). \tag{4.6}$$

Для здійснення визначення топологій нейроних мереж необхідно представити до якого ж с класів відноситься ця нейрона мережа (рис. 4.3).

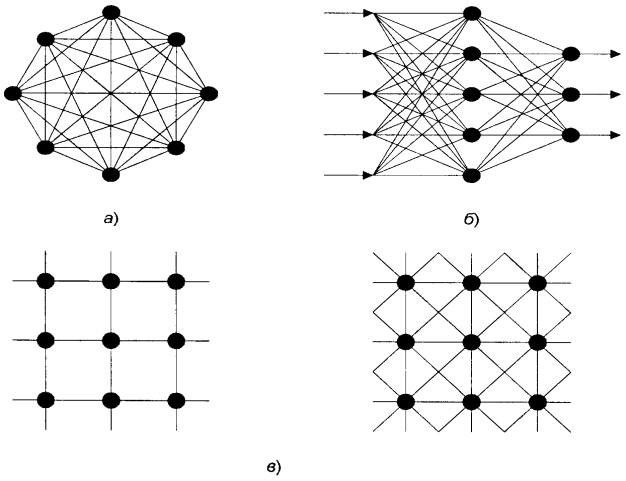

Рисунок 4.3 – Класифікація нейронних мереж за топологією
а) повнозв'язні мережі; б) багатошарова мережа з послідовними зв'язками;
в) слабкозв'язана мережа

За топологією виділяють наступні класи нейроних мереж [12]:

– повнозв'язні мережі передбачають, що кожен з нейронів надає свій вхідний сигнал всім нейронам множини і самому собі;
– багатошарові мережі, передбачають, що всі нейрони об'єднуються в шари, які характеризуються наявністю єдиного вхідного сигналу у кожному з шарів. При розгляді багатошарової мережі входом є вхід нульового шару, а вихід – це вихід останнього шару. Проміжні шари

називають прихованими. Багатошарова нейронна мережа може містити один або кілька таких шарів;

– слабкозв'язані мережі.

Якщо здійснювати класифікацію багатошарові мережі за наявністю зворотних сигналів, то можливо виділити наступні класи:

– мережі прямого поширення або мережі без зворотних зв'язків – це нейронні мережі, в яких сигнали передаються від шару до шару у напрямку від входу до виходу, при умові, що не обумовлено протилежне, то вихідний сигнал шару N подається на входи шару N + 1;
– рекурентні мережі або мережі зі зворотними зв'язками, де сигнал від наступних шарів передається на попередні.

Якщо здійснювати класифікацію багатошарові мережі за типом функції активації нейронів, нейронну мережу можливо поділяти на:

– гомогенну мережу, де у всіх нейронів однакова функція активації;
– гетерогенну мережу, де використано безліч різних функцій активації.

Безліч моделей штучних нейронних мереж, які не можна віднести до жодної з класів, називаються неструктурованими.

4.2. Багатошаровий персептрон

Розглянемо багатошарову штучну нейронну мережу прямого поширення, яка використана при побудові імітаційної моделі взаємодії інтелектуальних агентів. Зазначимо, що багатошаровий персептрон (MLP) є синхронним, що передбачає зміну стану відразу у цілої групи нейронів всього шару. Хід часу в такий ШНМ задається ітераційним виконанням однотипних дій над нейронами.

MLP – це багатошаровий персептрон, який є звичайною багатошаровою ШНМ прямого поширення та характеризується наступними відмінними ознаками [12]:

– кожен нейрон мережі має нелінійну функцію активації. Дана нелінійна функція є гладкою (тобто усюди диференційованою). Найпопулярнішою функцією, що задовольняє цій вимозі, є сигмоїдальна функція (3.5).
– мережа містить один або декілька шарів прихованих нейронів, які не є частиною входу або виходу мережі. Ці нейрони дозволяють мережі навчатися вирішенню складних задач, послідовно витягуючи найбільш важливі ознаки з вхідного образу (вектора).
– мережа має високий ступінь зв'язності, що реалізовується за допомогою синаптичних з'єднань. Зміна рівня зв'язності мережі вимагає зміни безлічі синоптичних з'єднань або їх вагових коефіцієнтів.

Багатошарова мережа представлена тришаровим персептроном з повністю пов'язаними шарами [12] (рис. 4.4).

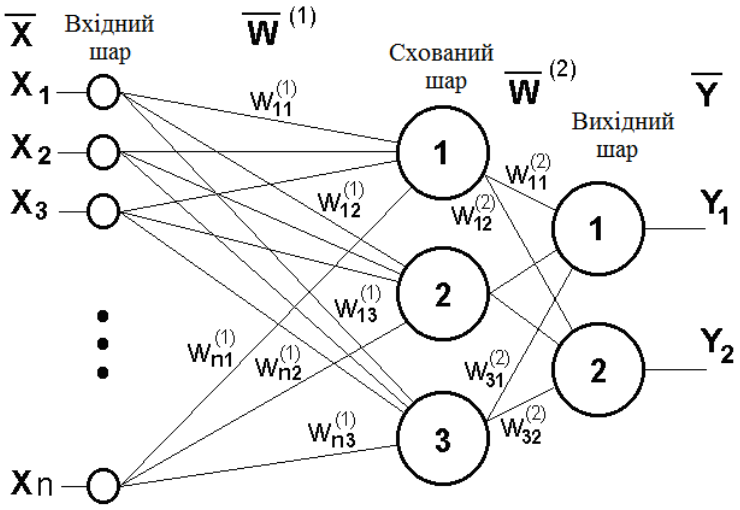

Рисунок 4.4 – Структурна схема багатошарового персептрона

Функція активації є нелінійною та включена до алгоритму роботи кожного нейрона, що дозволяє обійти умову функціонування будь-якої *m*-шарової ШНМ з ваговими матрицями $W^{(i)}$, $i=1,2,...m$ для кожного шару і уникнути розмноження вхідного вектора сигналів X на матрицю [12].

$$W^{(\Sigma)}=W^{(1)}\cdot W^{(2)}\cdot\ldots\cdot W^{(m)} \quad (4.7)$$

тобто фактично така *m*–шарова НМ еквівалентна одношаровій НМ з ваговою матрицею єдиного шару $W^{(\Sigma)}$:

$$Y=XW^{(\Sigma)} \quad (4.8)$$

Залучання умови нелінійності в MLP мережах, збільшує обчислювальну потужність мережі. Таким чином, маючи менше число нейронів з «нелінійними» синапсами сконструювати ШНМ, яка виконує роботу звичайної ШНМ з великим числом стандартних нейронів більш складної конфігурації.

4.3 Алгоритми навчання нейронної мережі

Фундаментальною властивістю мозку є здатність до навчання. Найбільш характерною і привабливою рисою штучних нейронних мереж є навчання, що значною мірою виділяє технології ШНМ, від багатьох інших обчислювальних систем.

Навчання в ШНМ розглядається як здатність налаштування параметрів мережі для вирішення поставленого завдання. Такими параметрами можуть бути синаптичні коефіцієнти, так звані ваги зв'язків або пороги (зсуви). Основною метою здійснення навчання ШНМ є отримання бажаної вихідної реакції мережі на деяку множину вхідних сигналів, яка називається навчальною вибіркою. Для зручності, вхідні і вихідні множини сигналів надають як вектора. У процесі навчання ШНМ виконується послідовне надання вхідних векторів деякої навчальної вибірки з одночасним налаштуванням параметрів мережі відповідно до обраної процедури. Такий процес називають алгоритмом навчання, який виконується до тих пір, поки не буде досягнута бажана вихідна реакція ШНМ для всієї навчальної вибірки [13].

Навчання ШНМ є ітераційним процесом, який налаштовано на таке підстроювання параметрів мережі, щоб деякий функціонал якості звертався в оптимум для всієї навчальної вибірки. Таким функціоналом є використання функція помилки, що характеризує ступінь близькості відображення вхідного вектора в бажаний вихідний. У загальному випадку функціонал якості (функція помилки) може мати довільний вигляд, тому навчання ШНМ перетворюється в задачу багатоекстримальної неопуклої багатомірної оптимізації [6,10]. Для здійснення процесу навчання ШНМ необхідно створити модель зовнішнього середовища та визначити доступну для цієї мережі інформацію. Така модель виступає в якості парадигми навчання. Маючи парадигму навчання, далі формулюються правила підстроювання параметрів, які і є алгоритм навчання. Розділяють три парадигми навчання: «з вчителем», «без вчителя» (самонавчання) і змішана [13].

Коли в мережі тільки один шар, алгоритм її навчання із учителем досить очевидний, тому що правильні вихідні стани нейронів єдиного шару свідомо відомі, і підстроювання синаптичних зв'язків іде в напрямку мінімізації помилки на виході мережі. По цьому принципу будується, наприклад, алгоритм навчання одношарового персептрона [13]. У багатошарових мережах оптимальні вихідні значення нейронів усіх шарів, крім останнього, як правило, не відомі, і двох або більш шаровий персептрон уже неможливо навчити, керуючись тільки величинами помилок на виходах ШНМ. Самий прийнятний варіант вирішення цієї проблеми – поширення сигналів помилки від виходів ШНМ до її входів, у напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи. Цей алгоритм навчання ШНМ називається «алгоритм зворотного поширення помилки».

4.4 Застосування генетичного алгоритму навчання

ШНМ може здійснювати навчатися генетичним алгоритмом, який був розроблений Джоном Холландом і є модифікацією так званого «еволюційно-

го програмування». Основна мета генетичного алгоритму полягає в створенні алгоритму пошуку, який засновано на ідеї відомого з біології механізму природного відбору або так званого алгоритму «спрямованого» випадкового пошуку. На початковому етапі ініціалізується популяція можливих рішень. На основі цієї початкової популяції виводиться нове покоління рішень, яке в подальшому є «вихідним матеріалом» для наступного покоління і т.д. В циклі генетичного алгоритму передбачено застосування стадії відбору, схрещування і мутації. Якщо дотримуватись виконання правила, що кожне нове покоління повинно містити кращі рішення, ніж попереднє, то отримаємо можливість відбору найкращих представників покоління для відтворення нової популяції. Популяція складається з набору бінарних рядків, де кожен з рядків надає вирішення проблеми і називається хромосомою [9].

На першому етапі генетичного алгоритму виконуємо відбір, який дозволяє визначити рядки для створення нового покоління. Так звані «батьки» вибираються довільно, але «найкращі особини» популяції мають всі шанси бути обраними. Саме засобами наведеного алгоритму пошуку здійснюється просування в самому перспективному напрямку.

Наступним етапом є схрещування, що передбачає для пари відібраних рядків довжини r вибір довільним чином числа $s \in \{1,...,r\}$.«Батьки» обмінюються бітами від $s+1$ до r; що дозволяє отримувати хромосоми нащадків.

Останнім етапом є мутація, яка дозволяє при ініціалізації алгоритму встановити фіксовану невелику ймовірність мутації до якої здатні новоутворені хромосоми. Блок-схема генетичного алгоритму де передбачено застосування етапів відбору, схрещування і мутації наведено на рис. 4.5. Таким чином, алгоритм просувається в самому перспективному напрямку пошуку. Наступна стадія – схрещування. Схрещування полягає в тому, що для пари відібраних рядків довжини r кожна вибирається довільним чином число $s \in \{1,...,r\}$. «Батьки» обмінюються бітами від $s+1$ до r; таким чином виходять хромосоми нащадків. Заключна стадія – мутація. При ініціалізації алго-

ритму встановлюється фіксована маленька ймовірність мутації, якій піддаються новоутворені хромосоми (рис. 4.5).

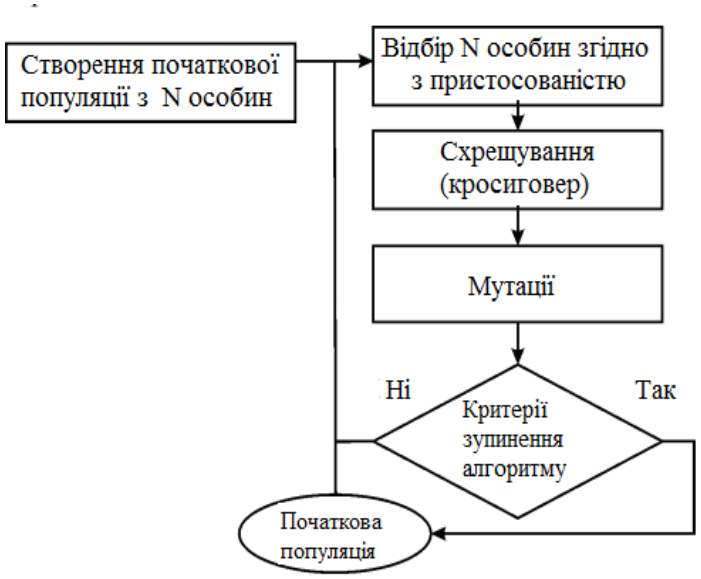

Рисунок 4.5 – Блок-схема функціонування генетичного алгоритму

4.5 Навчання нейронної мережі генетичним алгоритмом

Ітераційний принцип генетичного алгоритму дозволяє отримати лише приблизне рішення, але дозволяє використовувати цей алгоритм у різноманітних областях, отримуючи значну швидкість обчислень при обмежених обчислювальних ресурсах. Він може застосовуватися для підстроювання ваг схованих і вихідних шарів при фіксованому наборі зв'язків і широко використовується в задачах оптимізації і навчання нейронних мереж. З точки зору формальної мови математики, генетичний алгоритм – це алгоритм знаходження глобального екстремуму багатоекстремальної функції, що полягає в паралельній обробці множини альтернативних рішень. При цьому пошук концентрується на найбільш перспективних із них. В алгоритмі використано наступні визначення [14]:

– ген – ваговий коефіцієнт нейронної мережі;
– хромосома – набір генів, тобто вагових коефіцієнтів нейронної мережі, зчитуваних у певному порядку зверху – вниз, справа – наліво);

кожна хромосома є можливим рішенням, тобто таким набором вагових коефіцієнтів, які краще підходять для поведінки агента;

– популяція – безліч хромосом, варіантів наборів вагових коефіцієнтів;
– епоха – ітерація, відповідна до створення нового покоління хромосом;

Основними сутностями є хромосоми над якими в певному порядку в межах однієї епохи проводяться наступні операції [14]:

– схрещування – створення з певним ступенем імовірності (pc) нової хромосоми з генів двох інших і додавання її в популяцію;
– мутація – зміна з певним ступенем імовірності (pm) значення довільного гена будь-якої хромосоми і додавання її в популяцію;
– пристосування – видалення з популяції хромосом, тобто наборів вагових коефіцієнтів, що показали гірший результат.

Мутація вирішує одну із проблем, яка присутня у навчанні багатошарової нейронної мережі методом зворотного поширення помилки. Вона необхідна для виходу популяції з локального екстремуму і сприяє захисту від передчасної збіжності (рис.4.6).

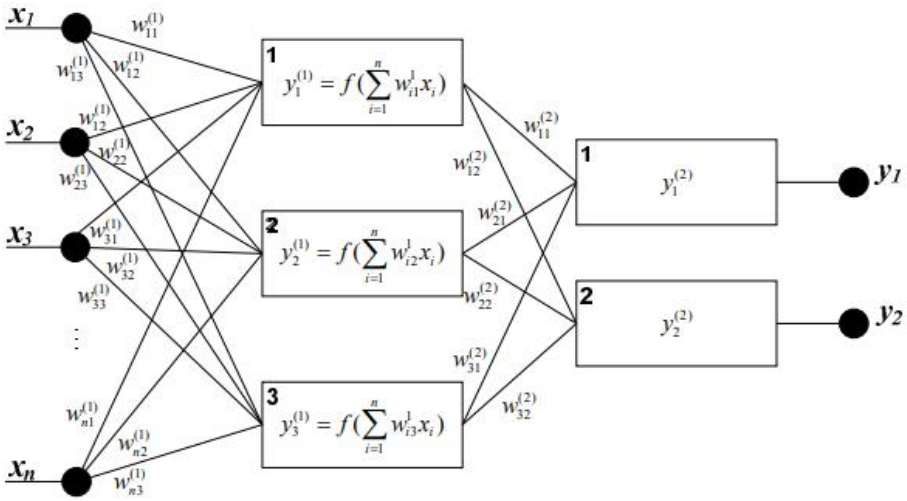

Рисунок 4.6 – Двошарова нейронна мережа з генами хромосом

На рисунку 4.6 наведено нейронну мережу, вагові коефіцієнти якої при виконанні обходу верху – вниз, справа – наліво, заповнюються хромосоми. Таким чином, хромосома являє собою набір генів – вагових коефіцієнтів [10].

При створенні нової популяції, успішне функціонування алгоритму забезпечується ще на етапі відбору батьківських хромосом, де використовується так званий метод відбору – рулетка. При використанні генетичного алгоритму працюють два основних генетичних оператора: оператор схрещування, який забезпечує високу імовірність схрещування ($0{,}5 \le p_c \le 1$) та оператор мутації, який забезпечує досить малу імовірність мутації ($0 \le p_m \le 0{,}1$). Таке співвідношення ймовірностей забезпечує в генетичному алгоритмі виконання схрещування практично завжди, тоді як мутації виконуються досить рідко.

Визначимо порядок дій оператора схрещування [10]:

– з ймовірністю pc з популяції вибираються дві особини, які включаються до складу тимчасової батьківської популяції;
– випадковим чином визначається точка схрещування lk;
– виконується операція конкатенації з частинами першого і другого з батьків.

З ймовірністю p_m оператор мутації змінює значення гена в хромосомі на протилежне. Імовірність мутації може обчислюватися випадковим вибором числа з інтервалу [0, 1] для кожного гена і відбором для виконання цієї операції тих генів, для яких розігране число виявляється менше або рівним значенню p_m.

Хромосома являє собою набір генів – вагових коефіцієнтів. В склад нової популяції включаються хромосоми, які отримані при застосуванні генетичних операторів до хромосом тимчасової батьківської популяції. Така популяція стає поточної при виконанні даної ітерації генетичного алгоритму. Кожна наступна ітерація передбачає виконання розрахунку значення функції пристосованості для всіх хромосом цієї популяції, після чого перевіряється умова зупинки алгоритму. Умовою зупинки алгоритму може буди деяке обмеження на максимальне число епох функціонування популяції, або умовою

стає можливість збіжності алгоритму шляхом порівняння значень функції пристосованості популяції на кількох епохах. Умовою зупинки алгоритму є наявність стабілізації цього параметра [10].

Навчання нейронної мережі із застосуванням генетичного алгоритму наведено на рисунку 4.7.

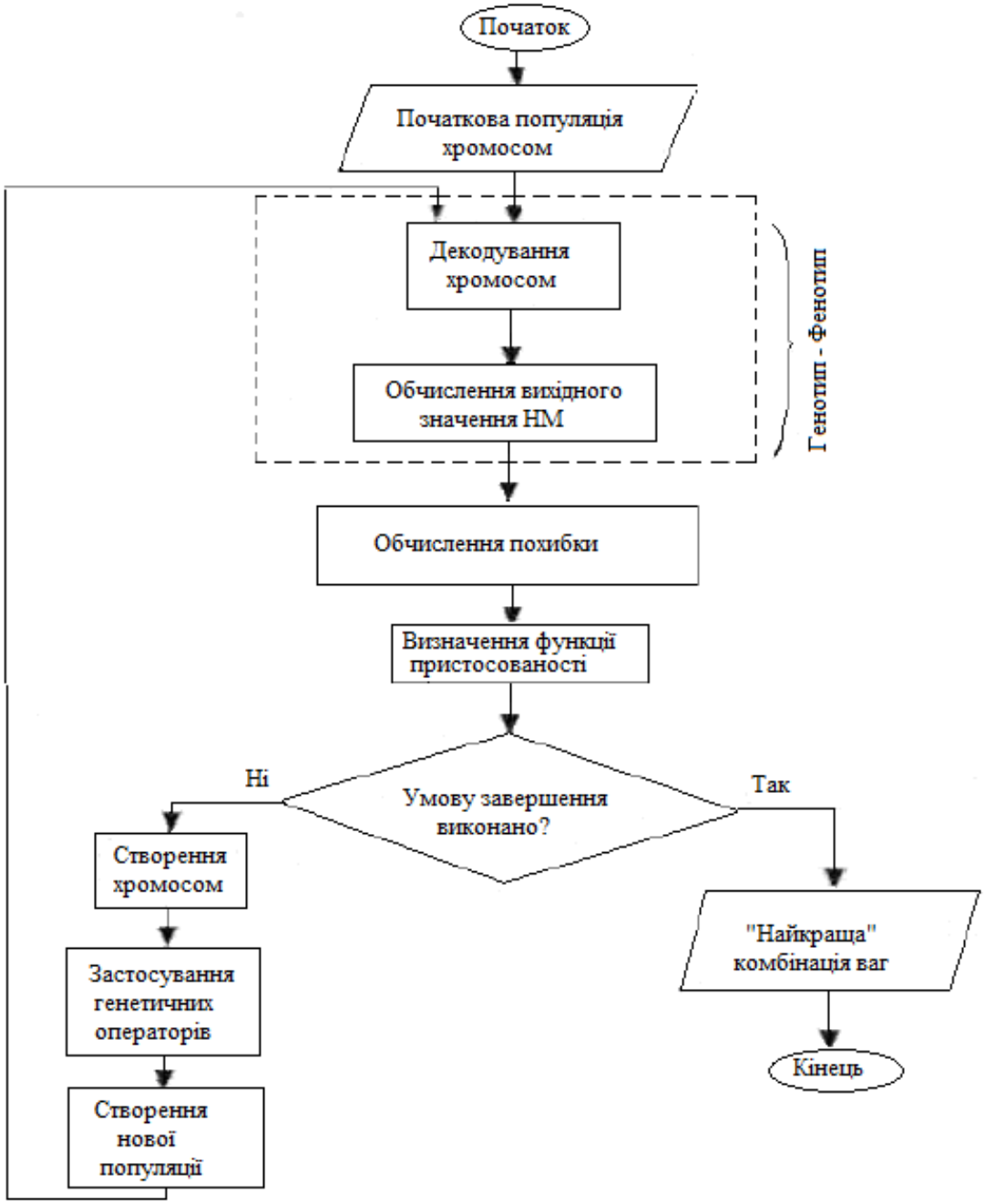

Рисунок 4.7 – Блок-схема навчання нейронної мережі генетичним алгоритмом

4.6 Моделювання взаємодії агентів на основі нейронної мережі

Для здійснення управління агентом на основі нейронної мережі використовуються рецепторні (вхідні) і ефекторні (вихідні) модулі. В якості входів нейронної мережі використовується видима картина світу та внутрішній стан агентів. Агенти отримають сенсорами з середовища наступні дані:

– дані про наявність їжі у найближчому оточенні;
– значення відстані до частки їжі;
– значення косинусу кута між вектором напрямку агента і вектором, спрямованим на їжу;
– дані про наявність інших агентів поблизу (і відповідно розраховується – відстань і косинус кута між напрямком нашого агента і вектором спрямованим до іншого агенту).

В моделі реалізована здатність агента отримувати інформацію про середовище, в якому агент функціонує. Ці дані є так званою «зоною видимості агента», яка дозволяє агенту в цій моделі «бачити» тільки перед собою (рис. 4.8).

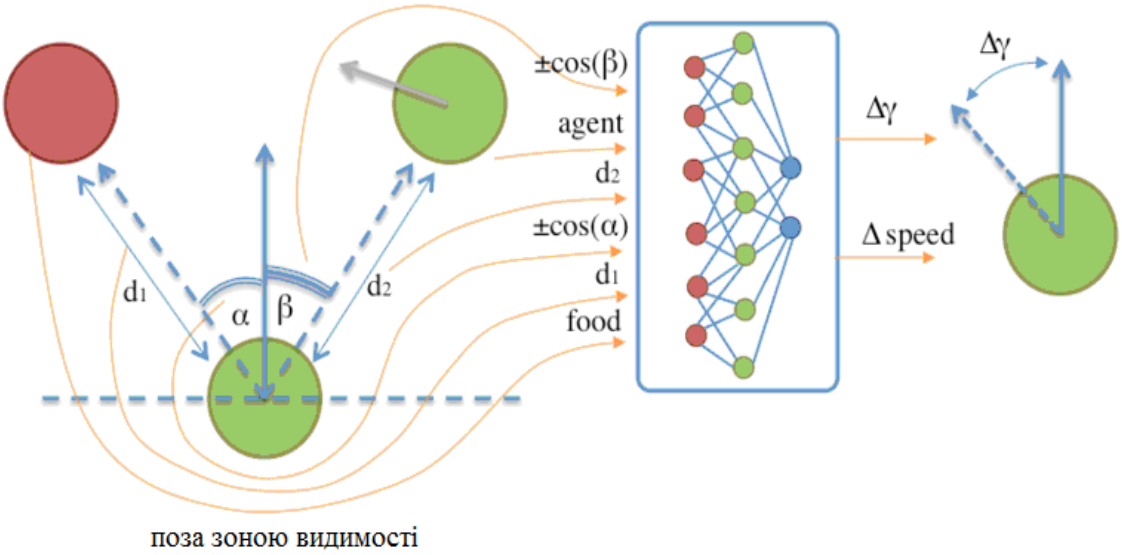

Рисунок 4.8 – Модель нейронної мережі

Взаємодія агента з середовищем забезпечена механізмом зміни свого становища і напрямку, тобто, на вході в нейронну мережу подаються значення сенсорів, а на виході з мережі зчитуємо значення кута, на який потрібно здійснити поворот, а також значення величини для зміни швидкості руху агента.

Завданням множини агентів є збирання їжі в середовищі. Якщо відстань між агентом та часткою їжі досить мала, то їжа вважається отриманою (з'їденою) і зникає з полю зору, а в той самий момент у середовищі у випадковому місті з’являється новий екземпляр їжі. Ефективність функціонування агентів оцінюється як значення кількості зібраної їжі.

Виходячи з наведеного функціонування агентів у середовищі, завдання навчання нейронної мережі передбачає налагодження вагових коефіцієнтів, де дуже добре справляється саме генетичний алгоритм.

Визначимо кроки алгоритму навчання нейронної мережі генетичним алгоритмом :

- перший крок: створити популяцію з набором хромосом, що складається із генів, рівних у кожній хромосомі числу вагових коефіцієнтів нейронної мережі;
- другий крок: заповнити кожну хромосому популяції генами з випадковими значеннями, що не суперечать умові задачі.
- третій крок: перевірити, якщо операції зроблені для всіх хромосом популяції, то перейти до восьмого кроку.
- четвертий крок: якщо операції зроблені не для всіх хромосом популяції, то вибрати з популяції пари хромосом з імовірністю pc і виконати схрещування; отриману нову результуючу хромосому додати в популяцію;
- п’ятий крок: вибрати з популяції хромосому з імовірністю pm і виконати мутацію генів, додавши нову результуючу хромосому до популяції;

– шостий крок: здійснити перевірку кожної хромосоми на відповідність вирішення завдання, та за отриманим результатом перевірки розташувати хромосому в порядку убування відповідності;
– сьомий крок: вилучити з популяції хромосому, порядковий номер якої перевищує початкове число хромосом у популяції та перейти до третього кроку;
– восьмий крок: перевірити відповідність першої хромосоми у популяції на відповідність вирішення завдання, якщо отримано рішення, то завдання вирішене.

Слід враховувати, що популяція хромосом еволюціонує, при цьому, виконуючи схрещування, відбувається народження нащадків хромосом, які відрізняються від своїх батьків. Механізм зміни генома виконується від батька до нащадка, а кожен ген w_i нащадка визначається як випадкова величина x, яка рівномірно розподілена на інтервалі $[w_{p1}, w_{p2}]$, де w_{p1} – відповідний ген батька 1 а w_{p2} – ген батька 2.

Геном агента S складається з хромосом $S = (w_b, w_l)$. Перша хромосома w_b містить ваги синапсів нейронної мережі блоку поведінки w_{ij}, а друга хромосома w_l містить ваги синапсів нейронної мережі блоку оцінки дій. При схрещуванні змінюються тільки хромосоми блоку поведінки, так як мається на увазі, що система цінностей кожного агента незмінна.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ ІМІТАЦІЙНОЇ МОДЕЛІ

5.1 Розробка класів імітаційної моделі

Розроблений програмний додаток надає можливість візуалізації імітаційної моделі взаємодії інтелектуальних агентів, а саме різних алгоритмів, побудованих на основі стратегій поведінки агентів в мультиагентній системі для подальшої оцінки та використання в ігрових застосунках. Даний додаток являє собою імітаційну модель, яка дозволяє встановлювати кількість агентів, виконувати ініціалізацію інтелектуальних агентів, обираючи стратегії взаємодії агентів в мультиагентній системі, здійснювати ініціалізацію агентів на основі нейронної мережі, виконувати навчання нейронної мережі генетичним алгоритмом.

Для програмної реалізації додатку, що здійснює візуалізацію отриманої імітаційної моделі було обране середовище розробки Microsoft Visual Studio та мова програмування C# [15]. Діаграма основних класів створеного додатку наведена на рисунку 5.1.

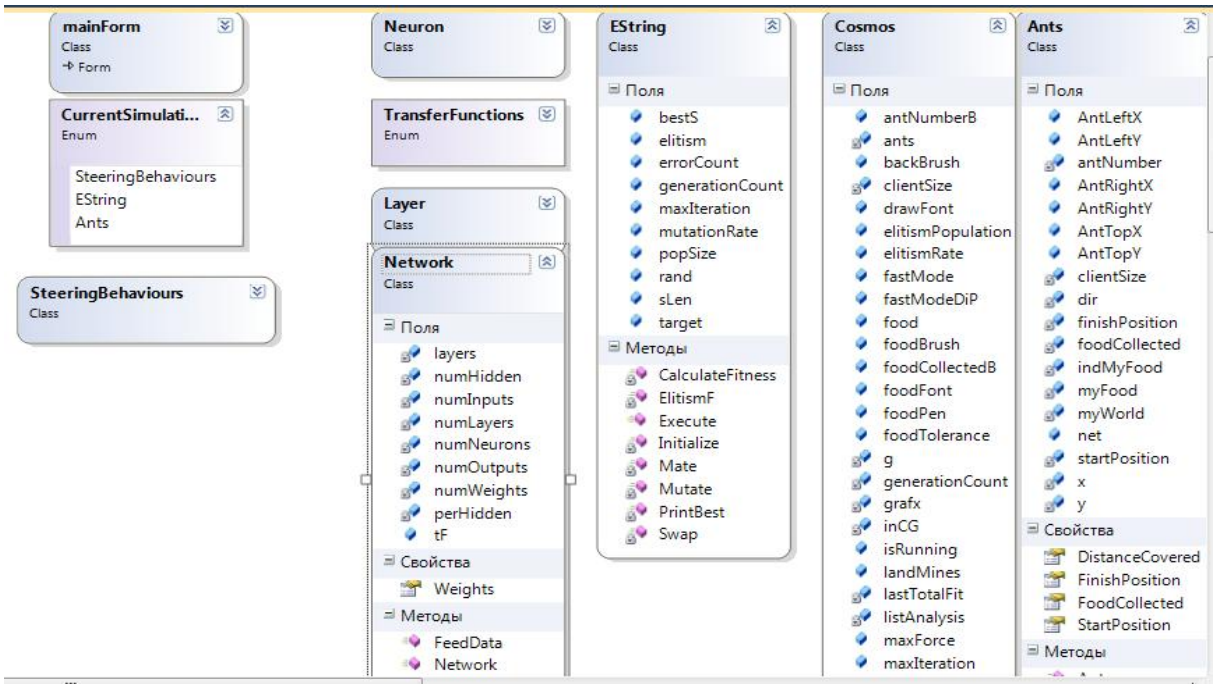

Рисунок 5.1 – Діаграма класів імітаційної моделі

Проведемо опис основних класів та їх функцій. Головний клас MainForm розміщає всю інформацію щодо функціонування додатку. Клас MainForm відповідає за відображення всіх дій (візуалізацію), які надає додаток та володіє об'єктами SBC (Steering Behavior Controller), Cosmos і EStrings (Evolving Strings). Засобами цих об’єктів MainForm виконується управління програмою. SBC створює об'єкти класу Vehicle, де кожен агент має доступ до класу SteeringBehaviours, який він використовує для визначення сили (напрямку, а також швидкості). Силу напрямку та швидкість інтелектуальні агенти застосовують до себе відповідно до стратегії взаємодії (поведінки). SteeringBehaviours і Vehicle використовують Vector для векторних обчислень.

За створення об'єктів агентів відповідає клас Cosmos, а агенти, в свою чергу, створюють об'єкти мережі (так званий «мозок»), який і визначає положення, швидкості і обертання. Клас EStrings відповідальний за реалізацію генетичного алгоритму, засобами якого здійснюється навчання нейронної мережі для вирішення завдання.

Генерація довільної сукупності ваг (рядків) відбувається на першому кроці. Кожен з отриманих рядків називається хромосомою, а кожен з символів являє собою характеристику організму. Нижче наведено ілюстрацію фрагмента програмного коду:

```
static void Initialize(ref List<OneString> ones, ref
List<OneString> two,ref  List<OneString> temp)
{
    string rString = "";
    for (int i = 0; i < popSize; i++)
    {
        rString = "";
        for (int j = 0; j < sLen; j++)
        {
            rString += (char)rand.Next(32, 142);
        }
        ones.Add(new OneString(rString, 0));
        two.Add(new OneString(rString, 0));
        temp.Add(new OneString(rString, 0));
    }
```

Для здійснення оцінка похибки навчання ваг використовується наступне програмне рішення:

```
private static void CalculateFitness(ref List<OneString> ones)
{
    int fitness;
    for (int i = 0; i < popSize; i++)
    {
        fitness = 0;
        for (int j = 0; j < sLen; j++)
        {
            fitness += Math.Abs(ones[i].name[j] - target[j]);
        }
        ones[i].fitness = fitness;
    }}
```

Навчання штучної нейронної мережі виконується генетичним алгоритмом, що завершується як тільки помилка дорівнює нулю, що є ознакою знайденого рішення. Якщо помилка має не нульове значення, то здійснюється генерація нового покоління, використовуючи для цього мутацію, елітарність, кросовер тощо. Надалі повторюємо знову цей алгоритм навчання з новим поколінням (рис. 5.2).

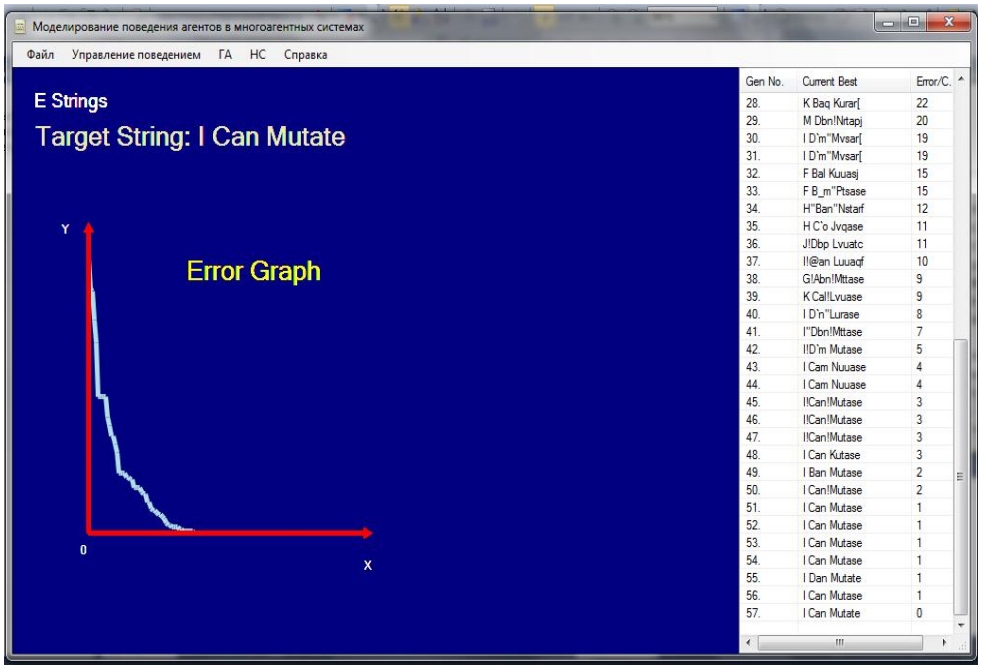

Рисунок 5.2 – Результат використання класу EStrings для навчання ШНМ

Призначенням класу Ants є поєднання генетичного алгоритму з нейронною мережею. Клас Ants відповідає за створення множини агентів, використовуючи типовий мурашиний, які здатні до навчання, здатні до пошуку їжі завдяки інтелектуальним властивостям, що забезпечує нейронна мережа. Набір сенсорів агента передають дані у мережу що до поточного руху агентів, а отримуючи виходи – здійснюють переміщення агентів для збору їжі згідно з алгоритмом пошуку. Дії виконують продовж деякого часу, який називається епохою. Оцінити ефективність роботи кожного з агентів можливо по кількості зібраної їжі, але ж всі дії агента зумовлюють використання алгоритму на основі нейронної мережі, що надає агенту інтелекту. Після завершення епохи, зважуючи кількості зібраної їжі, сортируємо агентів, обираючи найкращих для створення нової популяції і продовжуємо навчання, поки агенти не навчаться. у певному порядку, вибираємо найкращих, нехай вони будуть батьками та мати дітей. Далі продовжуємо, поки агенти не навчаться.

Всі обчислення, необхідні для «інтелектуальної» роботи агентів знаходяться на рівні класів симуляції, а алгоритм взаємодії агентів (поведінки) забезпечується функціонуванням класу CurrentSimulation. Керування всіма агентами здійснює клас SBC. Засобами класу SteeringBehaviour реалізовано алгоритм стратегії прицілювання при взаємодії агентів. Кожен з агентів у середовищі намагається здійснити пересування у певну обрану точку, яка може бути як нерухомою, так і може бути іншим агентом, що рухається.

5.2 Опис інтерфейсу додатку для управляння моделлю

Програмний додаток імітаційної моделі реалізовано у середовищі Microsoft Visual Studio, класи додатка реалізовані мовою програмування C# [15]. Додаток «Моделювання поведінки агентів в мультиагентній системі» має головне меню з розділами: Файл, Управління поведінкою, Довідка (рис. 5.3).

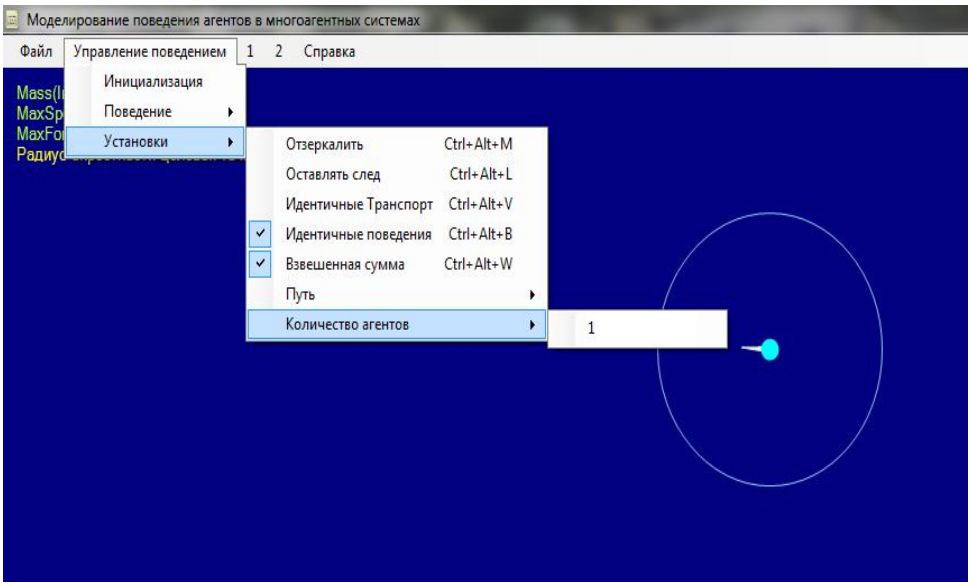

Рисунок 5.3 – Інтерфейс програмного додатка

Після запуску додатка, в меню «Управління поведінкою» користувач повинен обрати кількість агентів після чого додаток потребує виконання ініціалізації. Наступним кроком є вибір моделі взаємодії агентів, що передбачає вибір у меню «Управління поведінкою» необхідної моделі (рис. 5.4).

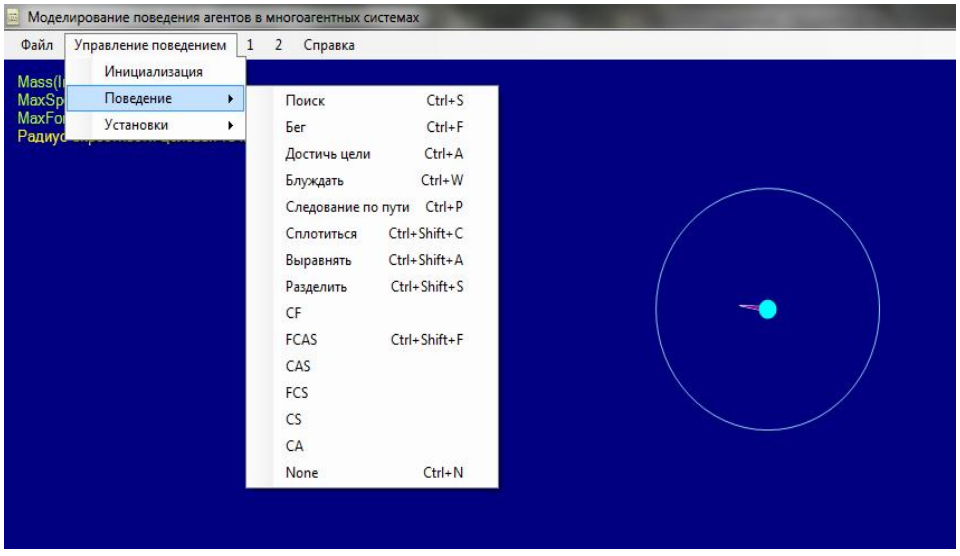

Рисунок 5.4 – Вибір у меню «Управління поведінкою» стратегії

Додаток має можливість відображення у вікні (у лівому верхньому куті) координат агентів і цільової точки пошуку (рис. 5.5).

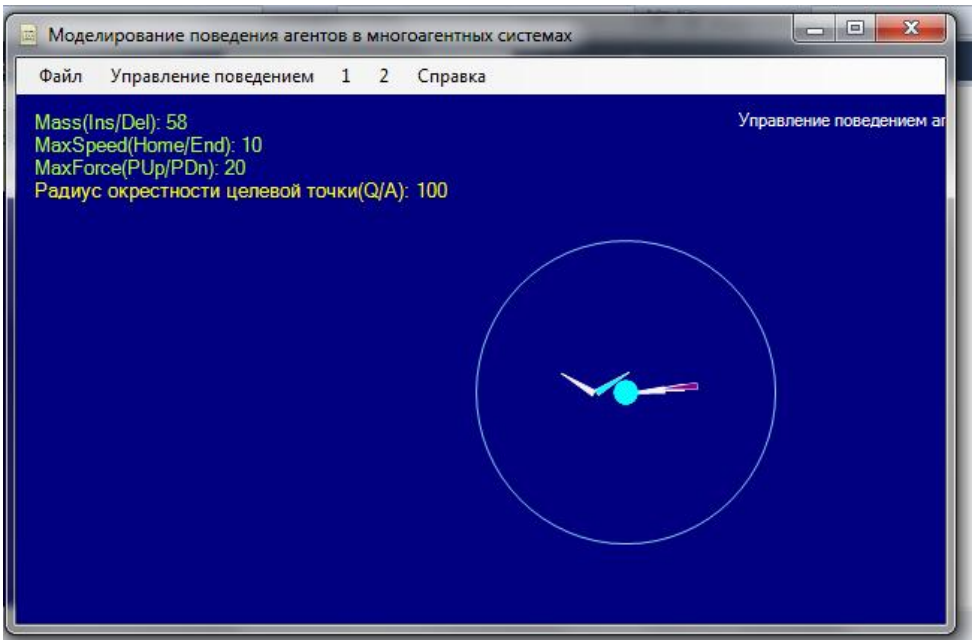

Рисунок 5.5 – Статистика при виборі стратегії «Досягнення цілі»

Вибір стратегії взаємодії агентів дозволяє користувачам візуалізувати в додатку алгоритм функціонування та оцінити доцільність використання різних стратегій для вирішення конкретних завдань. Стратегія «Проходження по шляху» передбачає генерацію вершин графа (відзначено червоним) випадковим образом, між якими є лінії з'єднання. Якщо агент відвідав вершину графа, то ця вершина відображається блакитним кольором (рис. 5.6).

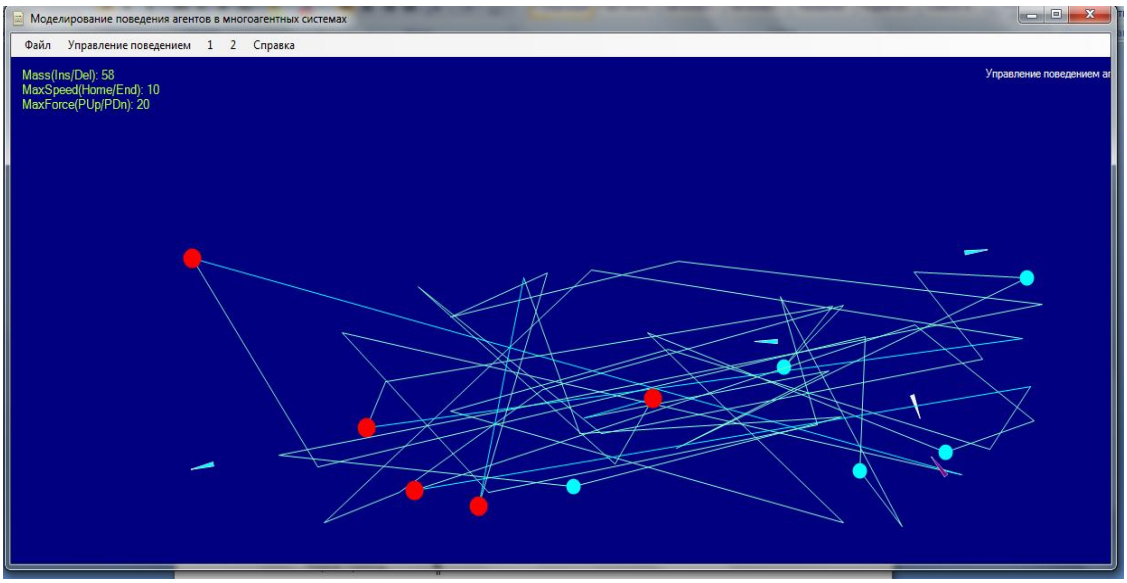

Рисунок 5.6 – Візуалізація стратегії «Проходження по шляху»

Вікно додатку відображає наступну обрану користувачем візуалізацію стратегії поведінки агентів «Блукання» (рис. 5.7).

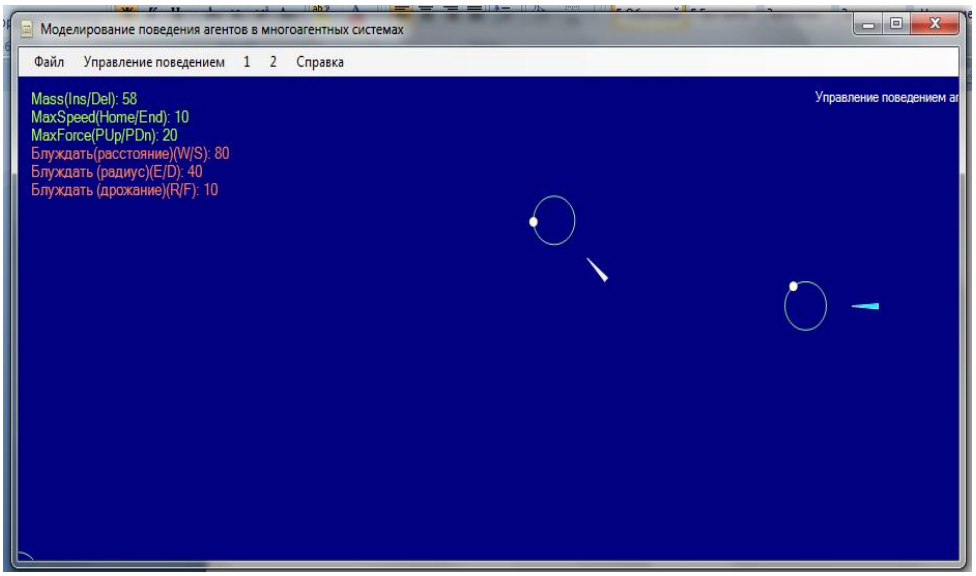

Рисунок 5.7 – Візуалізація стратегії «Блукання»

Для демонстрування виконання стратегії взаємодії агентів «Біг», користувач повинен у вікні додатка курсором мишки встановити коло, яке за алгоритмом стратегії агенти повинні обходити та тікати від перешкоди (рис. 5.8).

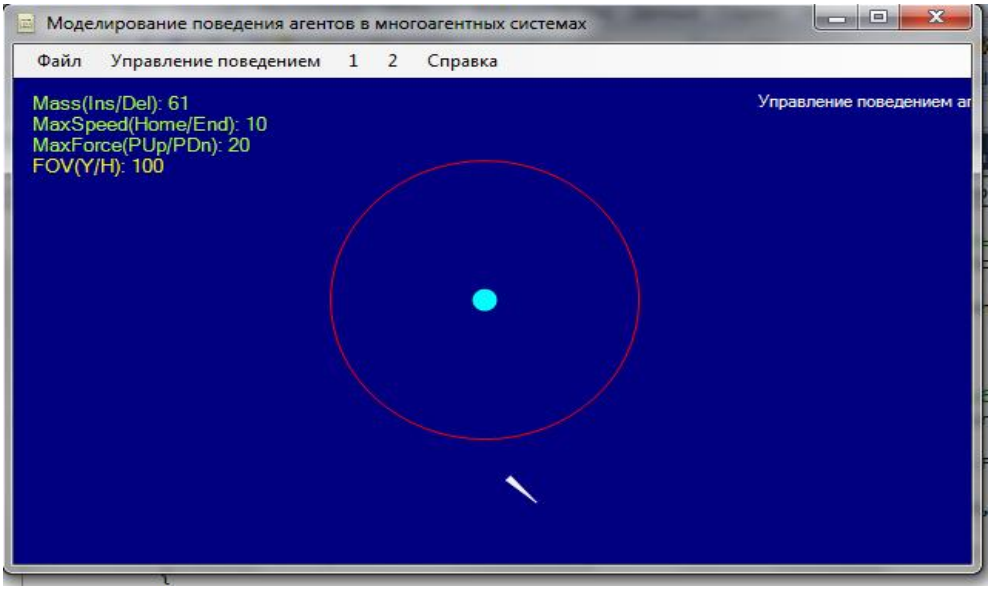

Рисунок 5.8 – Візуалізація стратегії «Біг»

При виборі у додатку ініціалізації агентів на основі штучної нейронної мережі, треба обрати функцію активації, елітарності агентів, функцію швидкого режиму і функцію відображення значущих при моделюванні значень у вікні додатка (рис. 5.9).

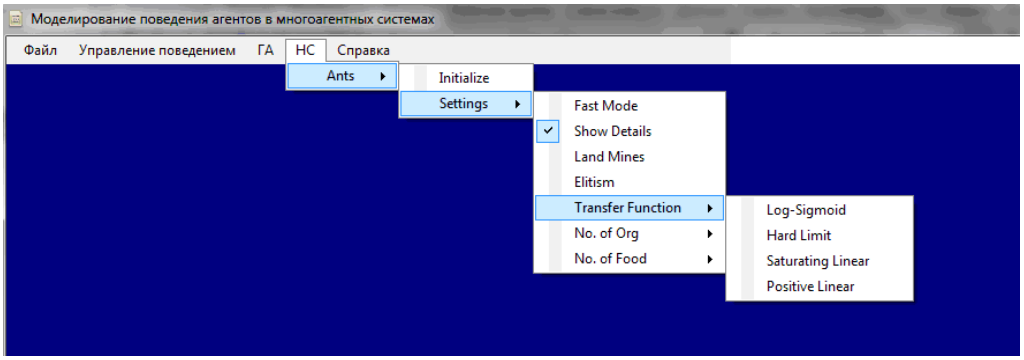

Рисунок 5.9 – Меню вибору параметрів для агентів на основі ШНМ

Після налаштування всіх необхідних параметрів функціонування нейронної мережі, здійснюється ініціалізація агентів, а додаток відображає візуалізацію моделі взаємодії інтелектуальних агентів у середовищі (рис. 5.10).

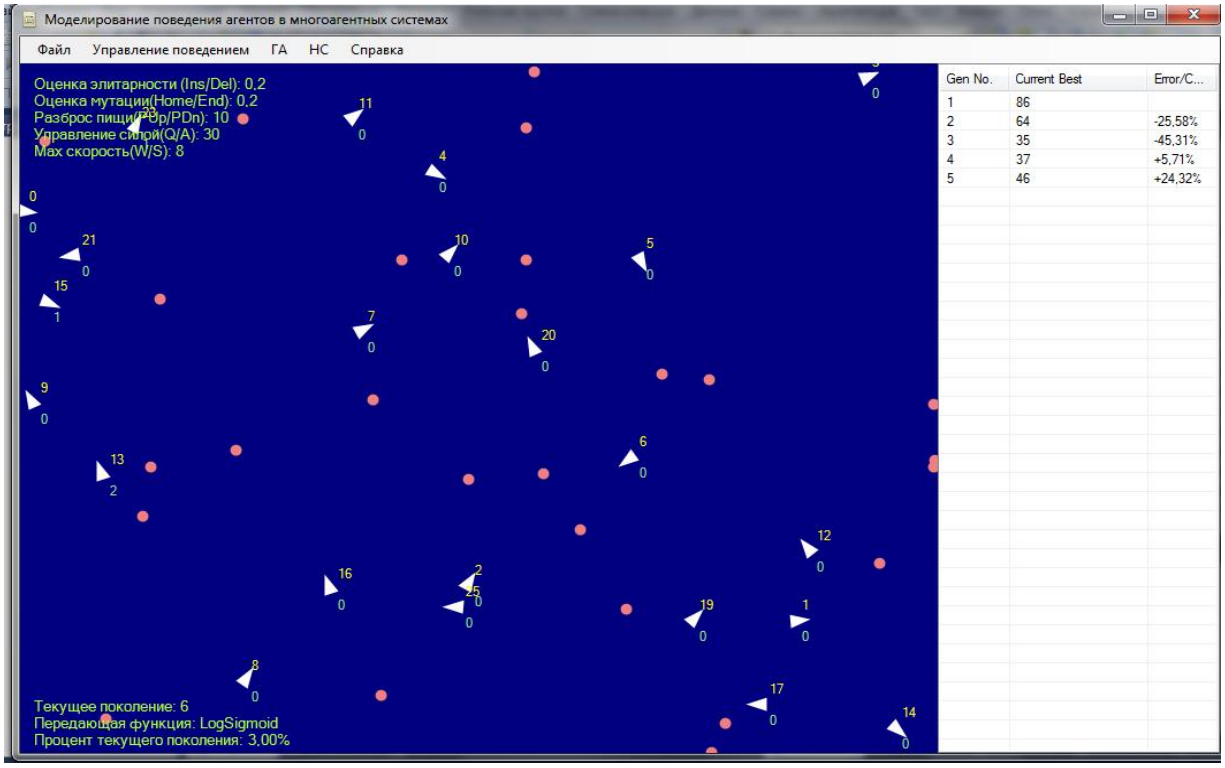

Рисунок 5.10 – Імітаційна модель взаємодії інтелектуальних агентів

В відображеній у вікні додатку імітаційній моделі взаємодії інтелектуальних агентів на основі нейронної мережі червоними точками позначаються частки їжі, агенти позначені геометричними фігурами трикутників. Поруч з кожним агентом (трикутником) зелений номер позначає кількість зібраної їжі; жовтий номер – число агента. У вікні, що розташоване праворуч відображається інформація що до існування попередніх поколінь.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи було здійснено розробку імітаційної моделі взаємодії інтелектуальних агентів в мультиагентній системі, що дозволяє візуалізувати та досліджувати поведінку агентів, використовуючи різні стратегії та їх комбінації.

В побудованій моделі мультиагентної системи функціонують інтелектуальні агенти на основі нейронної мережі, що дозволяє в процесі штучної еволюції популяції на основі навчання нейронної мережі здійснювати моделювання їх поведінки та взаємодії. Головною властивістю інтелектуального агента в моделі можливо вважати поведінку, яка спрямована на пристосування до навколишнього середовища.

Розроблена імітаційна модель дозволяє виконувати навчання нейронної мережі генетичним алгоритмом та функціонує у вигляді двошарового персептрона.

Дослідження створеної імітаційної моделі надало можливість встановити вплив неоднорідності розподілу їжі на поведінку, взаємодію та видоутворення в популяції. Якщо розподіл є неоднорідним, то зростає число підвидів. Крім того, запропонована модель дозволяє дослідити, механізми еволюційного навчання і збереження знань на рівні популяції в нестаціонарних умовах, коли змінюється у часі значення кількості їжі.

Модель дозволяє зробити висновки, що наявність лише одної еволюції без залучення навчання, забезпечує погану адаптацію популяції агентів. Застосування навчання дозволяє отримувати кращу адаптацію. Найкраща і більш ефективна адаптація отримана, коли в моделі залучені разом і еволюція і навчання. При застосуванні алгоритмів еволюції та навчання спостерігалась поведінка, яка дозволяє існувати агентам декілька десятків тисяч тактів.

Створений програмний додаток у середовищі Microsoft Visual Studio засобами мови програмування C# реалізує різні стратегії взаємодії інтелектуальних агентів і дозволяє досліджувати модель поведінки агентів при виборі

різних умов функціонування. Практична цінність роботи полягає в можливості застосування розробленої імітаційної моделі в сучасні анімаційні та ігрові додатки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Кравець І.О. Імітаційне моделювання: Навч. Посібник – ЧДУ ім. Петра Могили, 2010.– 107 с.
2. В. Ф.Ситник, Н.С. Орленко. Імітаційне моделювання: Навч. посібник. – К.: КНЕУ, 2010. – 232 с.
3. Томашевський В.М. Моделювання систем. Підручник. – К .: Видавнича група BHV, 2015. – 352 с.
4. Олійник А.О., Субботін С.О., Олійник О.О. Інтелектуальний аналіз даних: навч. посіб. – Запоріжжя: ЗНТУ, 2011. – 271 с.
5. Фратавчан В.Г., Фратавчан Т.М., Лукашів Т.О., Літвінчук Ю.А. Методи та системи штучного інтелекту: навчальний посібник. Чернівці: ЧНУ, 2023. – 114 с.
6. Плескач В.Л., Рогушина Ю.В. Агентні технології: Монографія. – К.: Київ. нац. торг.-екон. ун–т, 2005. – 344 с.
7. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: Монографія під заг. ред. С.О.Субботіна. – Запоріжжя: ЗНТУ, 2009. – 375с.
8. В. Д.. Дмитрієнко, О.Ю. Заковоротний, В.І. Носков, М.В. Мезенцев. Основи нейрокомп'ютингу: навчально-методичний посібник до практичних занять – Х.: НТМТ, 2014. – 140 с.
9. Вороновский Г.К., Махотило К.В., Петрашев С.Н., Сергеев С.А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. Харьков: Основа, 1997. – 112 с.
10. Руденко О.Г., Бодянський Є.В. Штучні нейронні мережі – Харків: Компанія СМІТ, 2006. – 404 с.
11. Чмир І.О. Методы и системы искусственного интеллекта: конспект лекций и упражнения для практических занятий для студентов 4 курса. ОДЕКУ, Одеса. URL: http://eprints.library.odeku.edu.ua/id/eprint/131/1/ChimirIA_Metody_ta_systemy_shtuchnoho_intelektu_2018.pdf (дата звернення: 01.05.2023).
12. Субботін С.О. Нейронні мережі: теорія та практика: навч. посіб. – Житомир: Вид. О.О. Євенок, 2020. – 184 с. ISBN 978-966-995-189-2.
13. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: монографія під заг. ред. С.О. Субботіна. – Запоріжжя: ЗНТУ, 2009. – 375с.

14. Л.М. Добровська, І.А. Добровська. Теорія та практика нейронних мереж: навч. посіб. – К.: НТУУ «КПІ» Вид-во «Політехніка», 2015. – 396 с. ISBN 978-966-622-691-7.
15. Andrew Troelsen, Philip Japikse. Pro C# 7: With. NET and .NET Core/ – Apress, 2017. – 1372p.