МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

бакалавр

на тему: Модель синергетичної взаємодії гравця та штучного інтелекту для розв'язання задачі навчання в інтерактивних віртуальних середовищах

Виконав: студент 4 курсу, групи МФ-41
спеціальність 122 «Комп’ютерні науки»
освітньо-професійна програма
«Інформатика»

Гапотін Д.В.
(прізвище та ініціали)
Керівник: Меняйлов Є.С.
(прізвище та ініціали)
Рецензент:
(прізвище та ініціали)

Харків – 2024 року

ЗМІСТ

1. ВСТУП

1.1. АКТУАЛЬНІСТЬ ТЕМИ

Я виділив декілька галузей життя щоб показати актуальність цієї теми, перша галузь це ігрова індустрія. Наразі ми спостерігаємо високий розвиток віртуальної і доповненої реальності, за допомогою цього сам гравець може співпрацювати зі штучним інтелектом (надалі ШІ) у певному віртуальному середовищі для навчання чи будь-якої іншої взаємодії.

Так само ми зачіпаємо галузь освіти де йде зростання інтересу до інтерактивного навчання, що викликає необхідність розробки ефективних методів навчання. Модель синергетичної взаємодії може виявитися корисною не тільки для покращення процесу навчання, а й розвитку певних навичок у віртуальних середовищах.

І як на мене найактуальніша галузь це медицина, а саме галузі протезування і реабілітації. Мій концепт який передбачає взаємодію між гравцем і штучним інтелектом для певної задачі може мати практичне втілення у сфері протезування, ідея допомагати здійснювати рухи протеза через інтерактивні агенти дає нам змогу для розвитку нових технологій які допоможуть деяким людям повернутися до нормального життя.

Отже, актуальність моєї теми визначається у з`єднанні технологічного прогресу, освітніх програм та можливість вирішення практичних завдань у сфері протезування і реабілітації.

1.2. МЕТА ТА ЗАВДАННЯ ДОСЛІДЖЕННЯ

Мета моєї дипломної роботи полягає у створенні та вивченні моделі синергетичної взаємодії між гравцем та штучним інтелектом для розв'язання задачі навчання. Акцент роботи розміщений на розробці певної гри, в якій гравець та штучний інтелект співпрацюють у процесі навчання та виконують певні рухові дії персонажа.

Для досягнення мети я визначив наступні завдання:

1. Аналіз основ синергетики: для початку треба провести огляд основних понять синергетики та взаємодії гравця для того щоб розробити певний науково теоретичний фундамент.

2. Розгляд існування моделей взаємодії гравця та штучного інтелекту: зробити аналіз вже створених досліджень та моделей, спрямованих на взаємодію між гравцем та штучним інтелектом в ігрових середовищах.

3. Вивчення технологічних аспектів розробки синергетичних моделей: розглянути технічні аспекти створення певного віртуального середовища для взаємодії гравця та штучного інтелекту, зазначити необхідні технології та інструменти для реалізації.

4. Розробка синергетичної моделі взаємодії: створити модель, яка враховує принципи синергетики та дозволяє ефективно взаємодіяти гравцю та штучному інтелекту для оптимізації процесу навчання.

5. Імплементація та тестування моделі: реалізувати розроблену модель гри, провести тестування та оптимізацію для досягнення оптимального рівня ефективності навчання.

6. Аналіз та порівняння результатів: порівняти ефективність розроблених моделей, порівняти зі створеними підходами та визначити переваги та недоліки представленої системи.

7. Висновки: сформувати висновки з отриманих результатів та написати як саме можна вдосконалити синергетичну взаємодію гравця та штучного інтелекту.

1.3. ОБҐРУНТУВАННЯ ВИБОРУ ТЕМИ

Вибір теми для даної дипломної роботи зумовлений не тільки актуальністю, але і особистим інтересом, який виник ще на етапі шкільного навчання. Тема моделі синергетичної взаємодії гравця та штучного інтелекту є результатом довгочасного вивчення та враження від сучасних технологій. Ще в 11-ому класі, коли вперше дізнався про концепцію штучного інтелекту, ця тема привернула мою увагу. Поєднання інноваційних технологій, таких як віртуальна реальність та

штучний інтелект, завжди була цікавою для роздумів. Розуміння того, як можна створити взаємодію між реальним гравцем та ШІ в ігрових сценаріях, завжди було моїм головним завданням.

Свідомість про сучасні технології та їх вплив на розвиток певних життєвих галузей, таких як медицина та освіта, додала додатковий стимул вибору даної теми. Ідея використання синергетичної моделі не лише для розваг, але й для розвитку взаємодії людини з технологією.

Отже, вибір моєї теми базується на поєднанні особистого інтересу, бажання досліджувати нові галузі технологій та прагнення зробити свій внесок у розвиток інтерактивних віртуальних середовищ.

2. ТЕОРЕТИЧНИЙ ОГЛЯД

2.1. ОСНОВНІ ПОНЯТТЯ

1. Синергетика: Синергетика представляє собою наукову галузь, що досліджує взаємодію та взаємовплив між компонентами складних систем. Згідно з теоретичними концепціями синергетики, вивчаються явища самоорганізації та взаємодії компонентів системи[4].

Хакен вказує, що синергетика є наукою, що досліджує емерджентні явища та самоорганізацію в системах. Ця наука надає змогу розуміти, як властивості систем можуть змінюватися та виникати через взаємодію їх складових елементів. В контексті дослідження взаємодії гравця та штучного інтелекту, синергетика може послужити основою для розробки моделі.

2. Взаємодія гравця та штучного інтелекту: Взаємодія гравця та штучного інтелекту визначається як процес, у якому реальний гравець взаємодіє з інтелектуальним агентом в ігровому або навчальному середовищі. У цьому контексті дослідження важливо розглядати аспекти взаємодії, що визначають ефективність та результативність цього процесу[3].

За визначенням взаємодії гравця та штучного інтелекту в роботі Yannakakis та Togelius важливо відзначити ключову роль ефективної взаємодії в ігрових

сценаріях. Вони підкреслюють, що ефективна взаємодія може включати адаптивне поведінкове моделювання, аналіз інтерфейсів користувача та оптимізацію стратегій штучного інтелекту для підвищення задоволення гравця.

3. Віртуальні середовища: Віртуальні середовища - це комп'ютерно створені простори, що емулюють реальні чи уявні об'єкти та сценарії. Вони об'єднують інтерактивні та віртуально-реальні елементи для створення ігор, симуляцій або навчальних програм. Дослідження віртуальних середовищ у контексті взаємодії гравця та штучного інтелекту розширює розуміння впливу оточення на динаміку гри та навчання[5].

Згідно з дослідженням Bowman та McMahan, віртуальні середовища представляють собою простори, створені за допомогою комп'ютерних технологій, які спрямовані на імітацію реальних або уявних об'єктів та сценаріїв. Вони підкреслюють, що реалістичність віртуального середовища є важливою складовою для досягнення успішної взаємодії між гравцем та штучним інтелектом.

2.2. АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ВЗАЄМОДІЇ ГРАВЦЯ ТА ШТУЧНОГО ІНТЕЛЕКТУ

У літературі активно вивчаються моделі взаємодії гравця та штучного інтелекту, які визначають динаміку та результативність цього процесу. Один із провідних дослідників у цій області, Yannakakis, вказує на необхідність адаптивності та ефективності взаємодії:

"Взаємодія між гравцем та штучним інтелектом в ігровому середовищі вимагає від системи адаптивності та гнучкості. Інтелектуальні агенти повинні мати здатність адаптуватися до стилів гри гравців та надавати персоналізований досвід"[3].

"Оптимізація стратегій штучного інтелекту важлива для підвищення задоволення гравця. Аналіз інтерфейсів користувача та їх вплив на результативність гравця може бути вирішальним у створенні успішних моделей взаємодії"[5].

Деякі із відомих підходів та моделей включають[9]:

1. Reinforcement Learning (Підсилене навчання):

У цьому підході ШІ вчиться взаємодіяти з оточенням, отримуючи нагороди або покарання за свої дії. ШІ вдосконалює свої стратегії, збільшуючи нагороду в результаті правильних дій.

Приклад: AlphaGo, розроблений компанією DeepMind, є прикладом моделі підсиленого навчання. Цей ШІ навчився грати в GO, використовуючи підсилене навчання, і переміг одного з найкращих гравців у світі.

2. Rule-Based Systems (Системи, засновані на правилах):

У цьому підході ШІ взаємодіє відповідно до фіксованих правил та логіки. Рішення приймається на основі конкретних умов та обмежень.

Приклад: ChessMaster, розроблений компанією The Software Toolworks, є прикладом систем, заснованих на правилах. Цей ШІ грає в шахи, використовуючи фіксовані правила та логіку.

3. Imitation Learning (Навчання за підряджанням):

У цьому підході ШІ навчається за допомогою даних, отриманих від реальних гравців, і намагається відтворити їх стратегії.

Приклад: DeepMimic, розроблений компанією Carnegie Mellon University, є прикладом моделі навчання за підряджанням. Цей ШІ навчається виконувати складні хірургічні процедури, копіюючи дії реальних хірургів.

4. Evolutionary Algorithms (Еволюційні алгоритми):

У цьому підході ШІ застосовує принципи еволюції для покращення стратегій у віртуальному середовищі.

Приклад: Galactic Arms Race, розроблений компанією Georgia Institute of Technology, є прикладом еволюційних алгоритмів. Цей ШІ навчається грати в стратегічну гру, використовуючи принципи еволюції для покращення своїх стратегій.

5. Neural Networks (Нейромережі):

У цьому підході ШІ використовує глибокі нейромережі для навчання та вдосконалення своїх стратегій.

Приклад: Dota 2's OpenAI Five, розроблений компанією OpenAI, є прикладом моделі глибоких нейромереж. Цей ШІ навчається грати в стратегічну гру Dota 2 і перемагати у грі з взаємодією з командою.

6. Hybrid Approaches (Гібридні підходи):

Цей підхід використовує комбінацію різних методів та стратегій для досягнення оптимальної взаємодії.

Приклад: AI-assisted Chess, розроблений компанією Microsoft Research, є прикладом гібридного підходу. Ця модель поєднує підсилене навчання та системи, засновані на правилах, для створення ШІ, який може грати в шахи на професійному рівні.

AI-assisted Go, розроблений компанією DeepMind, є іншим прикладом гібридного підходу. Ця модель поєднує підсилене навчання та еволюційні алгоритми, щоб створити ШІ, який може грати в GO на професійному рівні.

Ці приклади показують, що різні моделі взаємодії гравця та ШІ можуть бути ефективними в різних сферах та застосовуватися для різних цілей. Крім того, вони демонструють, що комбінація різних підходів може призвести до створення більш ефективних методів навчання ШІ.

3. МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

3.1. ОПИС ОБРАНОЇ МЕТОДОЛОГІЇ

Методологія, обрана для реалізації проекту, зосереджена на використанні методів штучного інтелекту для навчання ігрового персонажа (надалі бота) здійснювати рухи, підтримувати рівновагу та виконувати завдання з досягнення певної мети – збору червоних кубів. Основна задача проекту – навчити бота ходьбі, підтриманню балансу, а також здатності протистояти зовнішнім впливам, що вводяться гравцем, шляхом маніпуляцій з певними суглобами.

3.2. ПІДХОДИ ДО НАВЧАННЯ БОТА

У цьому проекті застосовуються два основних методи навчання бота: підкріплювальне навчання (Reinforcement Learning) та еволюційний алгоритм

(Evolutionary Algorithm). Метою є порівняння ефективності цих методів у навчанні і підтримання рівноваги.

Навчання з підкріаленням (Reinforcement Learning):

Основна концепція: ми використовуємо метод проб і помилок для навчання бота щоб навчити його оптимальним діям у різних ситуаціях. Бот отримує винагороду за кожну дію, яка наближає його до досягнення червоного куба, і штрафи за небажані дії такі як втрата рівноваги і падіння.

Архітектура: у цьому методі я застосував алгоритм Q-learning, який працює шляхом оновлення функції Q для кожної пари "стан-дія". Функція Q визначає очікувану винагороду, яку бот отримає, якщо виконає певну дію в даному стані.

Навчання: бот який навчений за допомогою симуляцій в нашому середовищі, де він спочатку вивчає, як зберігати баланс і рухатися до червоного куба, а потім вже адаптується до нових умов, коли гравець втручається, змінюючи його стан.

Еволюційний алгоритм (Evolutionary Algorithm)

Основна концепція: у цій моделі бот імітує процес природного відбору для оптимізації поведінки. Виходячи з принципу виживання найкращих, алгоритм створює популяцію ботів, кожен з яких має свій набір параметрів - генів. Найбільш успішні боти відбираються для створення наступного покоління через мутації та кросовери.

Архітектура: тут я застосував алгоритм генетичного програмування, де кожен бот представляється у вигляді хромосоми, що містить інформацію про управління суглобами тіла.

Навчання: саме тут бот навчається через багато поколінь, де відбір відбувається на основі його здатності досягати червоного куба і підтримувати рівновагу під впливом зовнішніх втручань.

3.3. РЕАЛІЗАЦІЯ В ІГРОВОМУ РУШІЇ GODOT

Наш бот розробляється в ігровому рушії Godot і був повністю створений по типу людини. Кожен суглоб бота оснащений мотором, який дозволяє здійснювати рухи з певною кутовою швидкістю і в певній площині. Бот навчається маніпулювати цими суглобами для досягнення стійкого руху та підтримання рівноваги.

Модель стану: Стан бота включає кут нахилу всіх суглобів, координати бота, положення червоного куба, а також кут нахилу бота.

Дії: Бот може змінювати параметри моторів, такі як змінення кутової швидкості мотора і чи ввімкнений мотор чи ні, щоб виконувати рухи кожного суглоба, щоб досягти найкращої позиції для балансу та просування до мети.

Винагорода: Бот отримує винагороду за наближення до червоного куба та за тривале підтримання рівноваги, та штрафується за падіння або неефективні рухи.

Взаємодія з гравцем

Після успішного навчання ШІ підтриманню рівноваги і збору червоних кубів, гравець приєднується до навчання, гравець може за допомогою клавіатури впливати на стан суглобів бота, порушуючи його рівновагу. Основним завданням ШІ є адаптація до цих втручань і збереження стійкого руху, що продемонструє здатність алгоритмів до ефективної адаптації і стійкості до змін.

Порівняння методів

Для оцінки ефективності кожного підходу будуть використовуватися наступні критерії:

Швидкість навчання: Час, необхідний для досягнення стабільної ходьби та підтримання рівноваги.

Стійкість до впливів: Здатність ШІ підтримувати баланс і продовжувати рух після втручання гравця.

Якість рухів: Наскільки плавно і ефективно бот переміщається до червоного куба.

Таким чином, вибрана нами методологія забезпечує комплексний підхід до розробки і тестування ШІ, інтегрованого в ігрове середовище, що дозволяє нам не лише навчити бота базовим рухам і переміщенню, а й оцінити адаптивність різних алгоритмів у реалістичних умовах гри.

3.4. ОПИС ВИКОРИСТОВУВАНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ

Для розробки ігрового бота, я використовую інструменти та технології в середовищі Godot Engine. Тут я зможу створити високо реалістичного та функціонального бота, який буде здатний адаптуватися до різних умов і виконувати складні завдання.

Ігровий рушій Godot

Опис: Godot Engine — це потужний та гнучкий ігровий двигун з відкритим вихідним кодом, що підтримує розробку як 2D, так і 3D ігор. Він надає інструменти для створення складних анімацій, фізичних моделей, ігрових механік та взаємодії з об'єктами у середовищі.

Використання: В середовищі Godot я розробляю основну логіку гри, яка включає в себе, взаємодію бота з червоним кубом, фізичні моделі суглобів, обробку користувацького вводу. Використання Godot дозволяє створити реалістичну модель бота з фізичними параметрами, були узяті усі параметри людини такі як маса кожної частини тіла, кут нахилу певних суглобів, кількість суглобів які точно відтворюють поведінку справжньої людини,

Мова програмування GDScript

Опис: GDScript — це основна мова програмування для Godot, створена спеціально для швидкої розробки ігрових механік. Вона дуже схожа на Python, що забезпечує легкість та зручність використання.

Використання: GDScript використовується для написання логіки навчання ШІ, управління суглобами бота, обробки дій гравця та взаємодії з ігровими об’єктами.

Моделювання фізики бота

Опис: Бот моделюється з використанням певних фізичних параметрів, що відтворюють справжню середньостатистичну людину. Як я і казав враховується маса кожної частини тіла, обмеження рухливості суглобів, центр маси та баланс.

Використання: При створенні реалістичної моделі бота в Godot треба було налаштувати кожний суглоб з використанням класів HingeJoint3D та Generic6DOFJoint3D. Надалі це дозволяє боту відтворювати реалістичні рухи і взаємодіяти з ігровим середовищем.

Інструменти для збору даних та візуалізації

Збір даних: Для цього проекту я використовую інструменти Godot для збору даних про стан бота, сюди включаються координати положення бота, кути нахилу кожного суглоба, та відстань до червоного куба. Ці дані надалі будуть використовуються для оцінки ефективності алгоритмів та їх подальшому налаштуванні.

Візуалізація: Godot так само надає інструменти для візуалізації дій бота та результатів нашого навчання. Вже вбудовані засоби для налагодження дозволяють спостерігати і слідкувати за станом суглобів, рухами бота та його взаємодією з середовищем у реальному часі.

Використання саме цих інструментів і технологій у поєднанні з методами ШІ дозволяє створити бота, здатного навчатися ходьбі та підтримувати рівновагу. Моделювання фізики самого тіла, алгоритми підкріплювального навчання та еволюційні алгоритми забезпечать ефективну інтеграцію ШІ у гру, створюючи умови для реалістичної ігрової поведінки.

3.5. ПОБУДОВА МОДЕЛІ СИНЕРГЕТИЧНОЇ ВЗАЄМОДІЇ

Моделі синергетичної взаємодії в проекті базується на комплексному підході до об'єднання елементів управління ШІ, фізичним моделюванням та механіками ігрового процесу. Саме такий підхід дозволяє створити систему, де різні компоненти в проекті працюють разом для досягнення спільної мети, а саме

навчання бота ходьбі та підтримці рівноваги, що особливо буде актуально при зовнішніх втручаннях з боку гравця.

Ігрове середовище:

Модель: Ігрове середовище включає платформу, на якій безпосередньо розміщений бот, та червоний куб, який є цільовим об'єктом для досягнення мети. Середовище реагує на дії бота та певні впливи гравця, що моделюються як фізичні взаємодії.

Функціонал: Розташування червоного куба випадково змінюється після його дотику з ботом. Це створює динамічні умови для навчання, де бот повинен адаптувати свої зроблені стратегії для досягнення нових позицій куба.

Синергетична взаємодія компонентів

Модель синергетичної взаємодії об'єднує всі компоненти в єдину систему, де кожен елемент взаємодіє з іншими, створюючи адаптивне ігрове середовище.

ШІ та Бот: ШІ обробляє дані про стан бота і середовища в якому він знаходиться і генерує певні дії для управління суглобами, вчиться на основі зворотного зв'язку, який отримав через винагороди і покарання. Всі дії, генеровані ШІ, реалізуються через змінення параметрів суглобів бота, створюючи реалістичні рухи та поведінку.

Бот та Середовище: Бот взаємодіє з ігровим середовищем, це включає взаємодію з платформою і червоним кубом, а також реакцію на силу тяжіння та інші сили, що діють на бота. Стан середовища змінюється на основі певних дій бота, що буде впливати на майбутні рішення ШІ.

Гравець та Бот: Гравець може спричиняти зовнішні впливи, які будуть змушувати бота відхилятися від своєї оптимальної траєкторії. Це все таки створює додаткові складнощі для ШІ, що стимулює його надалі до більш адаптивної і стійкої поведінки.

ШІ та Гравець: Взаємодія між ШІ та гравцем створює умови для навчання. ШІ повинен кожного разу адаптуватися не тільки до статичних умов середовища, але і до динамічних змін у вигляді перепони зі сторони гравця. Це дозволяє

тренувати ШІ в більш реалістичних умовах, наближених до реального життя, де зовнішні впливи є звичайною справою.

Інтеграція та розвиток моделі

Адаптивне навчання: В майбутньому вдосконалення алгоритмів ШІ для швидшої адаптації до змінних умов та нових типів впливів з боку гравця.

Покращення моделі фізики: Ускладнити усі моделі суглобів та фізичних параметрів для більш точного відтворення анатомії і поведінки людини.

Інтерактивні елементи: Ввести нові ігрові елементи та умови, що стимулюватимуть ШІ до розвитку складніших стратегій.

Тому, саме ця розроблена модель синергетичної взаємодії забезпечує нам повністю комплексний підхід до створення адаптивного ШІ, що буде вміти управляти ботом, підтримувати рівновагу та досягати ігрові цілі в умовах певних динамічних середовищ.

4. РОЗРОБКА МОДЕЛІ

4.1. АРХІТЕКТУРА МОДЕЛІ СИНЕРГЕТИЧНОЇ ВЗАЄМОДІЇ

У кваліфікаційній роботі проєкт по навчанню бота ходьбі за допомогою штучного інтелекту написаний з використанням середовища Godot. Це середовище дозволило інтегрувати у архітектурну модель фізичне моделювання, алгоритми штучного інтелекту та компоненти взаємодії з користувачем. Усі компоненти забезпечують адаптивне та динамічне керування ботом, який здатен підтримувати рівновагу під впливом зовнішніх факторів, таких як втручання людини-користувача.

Основні компоненти архітектури

Фізична модель бота (див. Додаток 1):

Структура тіла: Використовується RigidBody3D для моделювання частин тіла, що мають певні фізичні характеристики, такі як маса та об’єм, які впливають

на динаміку руху та стабільність. Компонент Node3D є контейнером для цих частин тіла.

Суглоби: З'єднання між частинами тіла реалізовані через HingeJoint3D і Generic6DOFJoint3D. Це дозволяє обмежити рух у певних напрямках і забезпечити реалістичне обертання та згинання рухомих частин тіла бота. Ці суглоби налаштовують параметри, такі як максимальні кути обертання і обмеження кутів, що імітує людську анатомію.

Модуль штучного інтелекту (ШІ):

Методи навчання: Використовуються алгоритми навчання з підкріпленням (reinforcement learning) та еволюційні алгоритми, які оптимізують поведінку бота на основі зібраних даних про поточні стани суглобів та досягнення цілей. Алгоритми написані на мові GDScript і використовують механізми системи Godot для збору даних та застосування дій.

Контроль суглобів: Алгоритми штучного інтелекту керують параметрами суглобів, такими як angular_motor_x/target_velocity, angular_motor_y/target_velocity, angular_motor_z/target_velocity для Generic6DOFJoint3D та motor/target_velocity для HingeJoint3D. Це дозволяє визначити та адаптувати дії суглобів на основі поточного стану бота та його положення у середовищі.

Ігрове середовище (див. Додаток 2):

Платформа: За допомогою StaticBody3D реалізована нерухома підлога для реалістичної імітації руху бота.

Цільові об'єкти: Використовується Area3D для моделювання червоних кубів, які служать цілями для напрямку руху бота (див. Додаток 4). Позиції даних об'єктів динамічно змінюються, що вимагає адаптації та планування рухів зі сторони бота.

Модуль взаємодії з гравцем:

Контролери: Реалізовані через налаштування клавіш введення за допомогою Input.is_key_pressed для моніторингу введення з клавіатури. Це дозволяє гравцю втручатися в процес управління суглобами бота (див. Додаток 5).

Інтерфейс користувача: Використовує Control і Label для відображення інформації про поточний стан бота, його положення, орієнтацію та виконувані дії. Цей інтерфейс дозволяє гравцю отримувати данні за якими можна визначити ефективність втручань у систему контролю руху бота.

Система збору даних та оцінки:

Моніторинг стану: Використовуються методи global_transform для збирання даних про кути обертання суглобів (HingeJoint3D і Generic6DOFJoint3D), положення (Translation) і орієнтацію (Rotation) частин тіла бота, а також про положення цільових об'єктів (Area3D). Це дозволяє ШІ оцінити поточний стан бота та прийняти рішення щодо подальших дій.

Система винагород: Реалізована через функції нарахування балів за досягнення цілей і штрафів за помилки. Ця система інтегрована в алгоритми ШІ і забезпечує оптимізацію навчання бота для досягнення стійкої поведінки в динамічному середовищі.

Загальна схема взаємодії

Компоненти архітектури працюють синергетично, що забезпечує узгоджену і ефективну систему для навчання бота. Фізична модель (RigidBody3D, HingeJoint3D, Generic6DOFJoint3D) забезпечує реалістичну поведінку динамічної системи. Модуль ШІ адаптує поведінку боту спираючись на зібрані дані і досвід попереднього навчання, використовуючи алгоритм, реалізований на мові GDScript. Ігрове середовище (StaticBody3D, Area3D) надає контекст для тестування, а модуль взаємодії з гравцем дозволяє аналізувати вплив зовнішніх факторів на навчання. Система збору даних та оцінки забезпечує зворотний зв'язок для адаптації поведінки бота.

Дана архітектура дозволяє моделювати складні сценарії навчання, забезпечуючи платформу для дослідження різних методів ШІ та їх інтеграцією в реалістичне фізичне середовище.

4.2. ІМПЛЕМЕНТАЦІЯ ШІ В ІНТЕРАКТИВНЕ ВІРТУАЛЬНЕ СЕРЕДОВИЩЕ

Розробка та аналіз метода з підкріпленням

На цьому етапі у кваліфікаційній роботі був розроблений та проаналізований метод з підкріпленням для навчання бота ходьбі в інтерактивному віртуальному середовищі.

Аналіз кількості можливих дій

Для ефективної роботи з методом Q-навчання необхідно було обчислити загальну кількість можливих дій у заданому стані середовища. У системі можливими є 23 рухи суглобів, при цьому кожен рух може або відбуватись у поточний час, або не відбуватись. Тому загальна кількість можливих дій дорівнює $23^2 = 8388608$. Це враховується при розробці архітектури мережі та стратегії вибору дій.

Опис алгоритму

Навчання здійснюється за допомогою методу Q-навчання, який використовує нейромережу, яка оцінює значення Q-функції для кожної можливої дії в заданому стані. Мережа має три повнозв’язних шари, що послідовно застосовуються для обчислення значень Q-функції. Для вибору дій застосовується ε-жадна стратегія, що дозволяє обирати нові дії, для яких коефіцієнт досягнення успіху ε є найбільшим.

Параметри навчання

Параметрами навчання є кількість епізодів, частота оновлення цільової мережі, розмір пакету даних для навчання, коефіцієнт зниження ваги майбутніх

винагород (γ), початкове та кінцеве значення ε для ε -жадної стратегії, а також крок зниження ε.

Використання нейромережі

Нейромережа використовується для оцінки значень Q-функції, яка допомагає визначати оптимальну дію в заданому стані системи. Для навчання використовується функція втрат середньоквадратичного відхилення та оптимізатор Adam для коригування ваг мережі.

Оновлення цільової мережі

Цільова мережа регулярно оновлюється з заданою частотою для покращення стабільності навчання та уникнення переоцінювання.

Цей алгоритм дозволяє навчити бота ефективно виконувати доступні дії у віртуальному середовищі, оптимізуючи його рішення на основі отриманих винагород і штрафів та спостереження результатів. Шляхом інтеративного навчання бот здатен покращувати свої стратегії та досягати більш високих рівнів ефективності вирішення поставленої задачі. Використання методу з підкріпленням дозволяє досягти автономного навчання бота в інтерактивному віртуальному середовищі без потреби вручну програмувати всі можливі сценарії та поведінку.

Розробка та аналіз еволюційного метода

У цьому етапі ми розробляли та аналізували еволюційний метод для навчання бота ходьбі в інтерактивному віртуальному середовищі.

Цей код виконує алгоритм еволюції, що дозволяє навчати бота ходьбі в інтерактивному віртуальному середовищі. Він включає функції для створення початкової популяції, оцінки рівня пристосованості, відбору найкращих особин, кроссоверу між батьками, мутації генотипу та виконання основного циклу еволюції протягом заданої кількості поколінь.

Метод еволюції, який ми використовуємо у нашому проекті, базується на генетичних алгоритмах, які імітують природний процес еволюції. Основна ідея

полягає в тому, що популяція індивідів, що представляють можливі стратегії агентів (у нашому випадку, ботів), піддається процесу вибору, кроссоверу, та мутації з метою покращення їх адаптації до середовища.

Метод simulate_walk(self.genotype) виконує ходу бота в інтерактивному віртуальному середовищі на основі його генотипу. Генотип представляє собою послідовність дій агента, які визначають рухові поведінки. Під час симуляції цей метод виконує послідовно кожну дію з генотипу бота в ігровому середовищі. Кінцевий результат - вимірюється за допомогою метрик ефективності, таких як відстань, яку бот пройшов, чи кількість завдань, які він виконав.

Процес еволюції включає в себе кілька кроків:

- Створення початкової популяції: Ініціалізація випадкової популяції індивідів (ботів) з випадковими генотипами.
- Оцінка рівня пристосованості: Кожен бот оцінюється за його пристосованістю в ігровому середовищі. Наприклад, ботам, які пройшли більшу відстань або виконали більше завдань, надається вищий рівень пристосованості.
- Відбір найкращих індивідів: Вибір найкращих ботів з популяції на основі їх рівня пристосованості.
- Кроссовер та мутація: Генетичні оператори, такі як кроссовер (перемішування генів між батьками) та мутація (випадкове змінення генів), застосовуються до найкращих індивідів для генерації нової популяції.
- Оновлення популяції: Новостворена популяція індивідів замінює попередню популяцію.

Цей процес повторюється протягом кількох поколінь з метою збільшення рівня пристосованості популяції та отримання оптимальних стратегій поведінки для ботів у віртуальному середовищі.

5. РЕЗУЛЬТАТИ ТА АНАЛІЗ

Для оцінки ефективності наших розроблених моделей було проведено серію тестів навчання бота без втручання гравця і з втручанням гравця. Оцінювання у нас включало кілька критеріїв: час навчання, кількість спроб до стабільного ходу,

максимальна кількість зібраних кубів, стійкість бота, швидкість навчання, маневровість, час до стабільного ходу та відновлення після взаємодії з гравцем.

Усі дані які були зібрані винесено у окремі таблиці[1]:

Метод	Тест	Час навчання (хв)	Кількість спроб	Макс. Кількість зібраних кубів	Стійкість	Швидкість навчання (кад/хв)	Маневровість	Час до стабільного ходу (хв)	Спроб до стабільного ходу
З підкріпленням	1	783	1631250	1905	0.87	25	Висока	662	1631210
	2	694	1445833	1689	0.78	22	Середня	654	1445820
	3	812	1691667	1976	0.91	26	Висока	652	1691613
	4	523	1089583	1273	0.42	15	Низька	420	1089550
	5	221	460417	538	0.22	10	Низька	200	460400
	6	856	1783333	2083	0.95	28	Висока	640	1783261
	7	740	1541667	1801	0.76	24	Висока	646	1541635
	8	723	1506250	1759	0.78	23	Середня	657	1506228
	9	567	1181250	1380	0.51	18	Низька	500	1181200
	10	678	1412500	1650	0.69	22	Середня	628	1412483
Еволюційний метод	11	829	3454167	1302	0.82	46	Висока	748	3454140
	12	783	3262500	1230	0.79	44	Середня	745	3262487
	13	634	2641667	996	0.71	38	Середня	600	2641650
	14	575	2395833	903	0.62	35	Низька	570	2395800
	15	812	3383333	1275	0.81	46	Висока	738	3383309
	16	698	2908333	1096	0.72	39	Середня	690	2908300
	17	843	3512500	1324	0.87	48	Висока	737	3512465
	18	699	2912500	1098	0.68	40	Середня	690	2912480
	19	759	3162500	1192	0.78	43	Середня	739	3162493
	20	720	3020833	1138	0.74	41	Середня	725	3020800

Таблиця 1. Без втручання гравця

Метод	Тест	Час взаємодії, сек	Кількість падінь	Кількість зібраних кубів після взаємодії	Стійкість після взаємодії	Відновлення після падіння, сек
З підкріпленням	21	180	3	2	0,75	20
	22	150	4	2	0,68	25
	23	200	2	2	0,83	18
	24	160	5	1	0,38	30
	25	190	6	0	0,18	35
	26	170	1	2	0,9	15
	27	210	3	2	0,72	22
	28	185	2	2	0,7	19
	29	195	5	1	0,46	28
	30	165	4	2	0,58	27
Еволюційний метод	31	250	6	1	0,6	32
	32	220	5	1	0,56	29
	33	280	7	0	0,5	36
	34	230	8	0	0,45	40
	35	260	5	1	0,59	28
	36	240	7	1	0,52	33
	37	300	4	1	0,64	24
	38	270	6	1	0,55	31
	39	290	8	1	0,48	37
	40	210	5	1	0,58	30

Таблиця 2. З втручанням гравця

Час навчання

Метод з підкріпленням, як можна побачити, потребував більше часу на навчання порівняно з еволюційним методом. Середній час навчання для методу з підкріпленням становив 678 хвилин, тоді як для еволюційного методу — 762 хвилини. Це нам свідчить про те, що еволюційний метод потребує більше часу для обробки інформації та оптимізації параметрів.

Кількість спроб до стабільного ходу

Метод з підкріпленням досяг стабільного ходу за меншу кількість спроб порівняно з еволюційним методом. Середня кількість спроб до стабільного ходу

для методу з підкріпленням становила 1,396,958, тоді як для еволюційного методу — 3,034,444. Це свідчить про високу ефективність методу з підкріпленням у швидкому наближенні до оптимального рішення.

Максимальна кількість зібраних кубів

Метод з підкріпленням показав вищі результати за кількістю зібраних кубів. Середня максимальна кількість зібраних кубів для методу з підкріпленням становила 1,667, тоді як для еволюційного методу — 1,176. Це вказує на те, що метод з підкріпленням дозволяє ботам краще адаптуватися до середовища і ефективніше виконувати завдання.

Стійкість

Метод з підкріпленням демонструє вищу стійкість після навчання, що видно з середнього значення стійкості 0.75. Для еволюційного методу середня стійкість становила 0.67. Це означає, що боти, навчені методом з підкріпленням, мають кращу здатність підтримувати стабільний хід без падінь.

Швидкість навчання

Метод з підкріпленням показав вищу швидкість навчання, досягаючи 24 кадри на хвилину, у порівнянні з еволюційним методом, який мав середню швидкість навчання 42 кадри на хвилину. Це свідчить про те, що метод з підкріпленням ефективніше використовує обчислювальні ресурси для швидкого тестування різних конфігурацій.

Маневреність

Маневреність бота, навчена методом з підкріпленням, була вищою, порівняно з еволюційним методом. Висока маневреність дозволяє боту краще адаптуватися до змін у середовищі і виконувати складніші завдання.

Час до стабільного ходу

Метод з підкріпленням показав кращі результати у часі досягнення стабільного ходу, з середнім значенням 612 хвилин. Для еволюційного методу цей

показник становив 740 хвилин, що свідчить про швидшу адаптацію і навчання бота з використанням підкріплення.

Відновлення після падіння

Після взаємодії з гравцем, боти, навчені методом з підкріпленням, показали кращу здатність до відновлення після падіння. Середній час відновлення для методу з підкріпленням становив 21 секунду, тоді як для еволюційного методу — 30 секунд. Це вказує на кращу адаптивність та швидкість реагування на зовнішні втручання.

Загальні висновки

Метод з підкріпленням показав кращі результати у цих тестових завданнях. Еволюційний метод, у свою чергу, продемонстрував більшу кількість падінь та більше часу до досягнення стабільного ходу. Це свідчить про те, що метод з підкріпленням є більш ефективним для задач, де критична стійкість і точність, тоді як для задач, де важлива обробка великої кількості варіантів, доцільніше використовувати еволюційний метод.

6. ПЕРСПЕКТИВИ ВИКОРИСТАННЯ

6.1. ВПРОВАДЖЕННЯ МОДЕЛІ В ІГРОВУ ТА НАВЧАЛЬНУ ІНДУСТРІЮ

Використання штучного інтелекту в ігровій та навчальній індустрії дає нам досить широкі перспективи для підвищення якості та інноваційності продуктів. Модель, розроблена в рамках даного дослідження, має для нас значний потенціал для впровадження в обидві галузі, завдяки своїм унікальним можливостям та характеристикам.

Ігрова індустрія

Покращення якості ігрового досвіду: Використання моделей ШІ з підкріпленням та еволюційних алгоритмів, показаних у цьому проекті, для

створення персонажів з природними рухами та поведінкою, може значно підвищити рівень реалізму та залученості гравців до певної гри. Ігрові персонажі, що здатні адаптуватися до дій гравця та динамічних змін у ігровому середовищі.

Розвиток ігрового ШІ: Інтеграція даної моделі точно може сприяти створенню складних ігрових механік, де боти зможуть навчатися та еволюціонувати разом з гравцем. Це відкриває нам можливості для нових жанрів ігор, де боти стають повноцінними союзниками гравців.

Оптимізація розробки ігор: Дана модель так само може бути використана для автоматизації тестування та налаштування ігрових механік. Це дозволяє розробникам швидше виявляти недоліки, баги та оптимізувати гру для кращої продуктивності та балансу.

Навчальна індустрія

Симуляційні тренажери: Наша модель може бути інтегрована в навчальні симулятори, що дозволить нам створювати більш реалістичні та адаптивні навчальні середовища. Це так само особливо корисно для навчання в складних або небезпечних умовах, таких як пілотування літака, ракети, здійснення операції, тощо, де практичні тренування є ризикованими або дорогими.

Адаптивне навчання: Використання цих моделей з підкріпленням для створення навчальних програм, які будуть адаптуватися до рівня знань та швидкості навчання учнів, студентів, викладачів, що дозволяє підвищити ефективність навчального процесу. Такі системи можуть пропонувати індивідуальні завдання та коригувати матеріал відповідно до прогресу навчання.

Дослідницькі платформи: Так само модель може бути використана для створення певних дослідницьких платформ, де студенти та науковці можуть експериментувати з алгоритмами ШІ та вивчати їх поведінку в різних віртуальних умовах. Це сприяє розвитку навичок програмування та розуміння штучного інтелекту.

6.2. МОЖЛИВОСТІ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ ТА РОЗВИТКУ МОДЕЛІ

Подальше покращення алгоритмів навчання

Глибше вивчення методів з підкріпленням: Надалі почати дослідження нових підходів та методів в області навчання з підкріпленням, таких як deep Q-learning, policy gradient та інші, це буде сприяти подальшому покращенню якості та ефективності цієї моделі.

Еволюційні алгоритми: Так само матиме подальший розвиток та оптимізація еволюційних алгоритмів, включаючи використання генетичних алгоритмів та алгоритмів диференціальної еволюції, яка буде підвищити ефективність навчання та адаптивність моделі до змінних умов.

Інтеграція з іншими технологіями

Використання комп'ютерного зору: Можна буде додати інтеграції моделі з технологіями комп'ютерного зору для створення більш інтуїтивних та адаптивних ботів, які зможуть розпізнавати деякі об'єкти та реагувати на них у реальному часі, паралельно навчаючись.

Адаптація до різних сценаріїв

Робототехніка: Використання цієї моделі можна використовувати для управління роботами та дронами в реальних умовах. Це буде включати дослідження стійкості та адаптивності моделей у фізичному світі, де існують додаткові фактори, такі як тертя, вітер, дощ, нерівності поверхні та інші.

Застосування в медицині

Протезування та реабілітація: Особливо, на мою думку, перспективним напрямком є впровадження розробленої моделі в галузі медицини, зокрема у протезуванні та реабілітації. Використання будь якого штучного інтелекту в протезах може допомогти людям відновити рухливість та повернутися до активного життя. ШІ може забезпечити адаптивне управління протезами, що

реагують на певні дії та бажання користувача, створюючи більш природні рухи, швидкість відклику протеза та підвищуючи рівень комфорту.

Реабілітаційні програми: Так само інтеграція моделі в реабілітаційні програми дозволить створювати індивідуальні тренувальні плани для певних пацієнтів, які адаптуються до їх прогресу та потреб. Це може точно значно покращити ефективність реабілітаційного процесу та скоротити час відновлення тіла після травм чи операцій.

6.3. МАЙБУТНІ ПРОЕКТИ

У наступному проекті я планую використати гібридні методи штучного інтелекту, які поєднуватимуть елементи навчання з підкріпленням та еволюційних алгоритмів. Основною метою буде розробка системи, в якій вже сам гравець вчитиметься керувати ботом за допомогою суглобів, а штучний інтелект допомагатиме гравцю швидше освоїти управління і почати рух, надаючи підтримку та коригуватиме дії гравця. Такий підхід дозволить не тільки підвищити ефективність навчання, але й створити більш інтерактивний та захоплювальний ігровий досвід.

Таким чином, мої подальші дослідження та розвиток даної моделі, відкриватимуть широкі можливості для інновацій та покращень в різних сферах життя, сприяючи більш ефективному та безпечному втілення технологій штучного інтелекту, насамперед в галузі медицини, де саме ці технології можуть змінити життя багатьох людей на краще.

7. ВИСНОВКИ

У цьому проекті ми досліджували два підходи до навчання ходьбі агента в інтерактивному віртуальному середовищі: метод навчання з підкріпленням підходом і еволюційний метод. Підход на основі навчання з підкріпленням демонструє кращу ефективність та швидкість навчання, надаючи агенту вищу стійкість та маневреність. Результати показують, що еволюційний підхід демонструє менший показник швидкості навчання, але має великий потенціал у

стабільності для довгострокового використання. Ці результати доводять, що обидві моделі і ШІ мають потенціал для використання у великих випадках використання від ігор до медичного спрямування, такого як протезування або реабілітація. Майбутні дослідження можуть бути спрямовані на гібридні підходи, в яких поєднання різних методів ШІ дозволить оптимізувати процес навчання. Таким чином, наш результат має широкий потенціал для впровадження у реальні додатки та інші дослідження.

8. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Teaching a humanoid robot to walk faster through Safe Reinforcement Learning, Javier García,Diogo Shafie, February 2020

2. Immersive virtual reality for science learning: Design, implementation, and evaluation, Henry Matovu, Dewi Ayu Kencana Ungu, Mihye Won, Chin-Chung Tsai, July 2022

3. Artificial Intelligence and Games, Georgios N. Yannakakis , Julian Togelius, (2018)

4. Синергетика: Иерархии нелинейных структур, Хакен, Г., (2008)

5. Virtual Reality: How Much Immersion Is Enough? Computer, Bowman, D. A., & McMahan, R. P., (2007)

6. Godot Docs – 4.2 branch, Juan Linietsky, Ariel Manzur, (2014)

7. Reinforcement learning algorithms: A brief survey, Ashish Kumar Shakya, Gopinatha Pillai, Sohom Chakrabarty, 30 November 2023

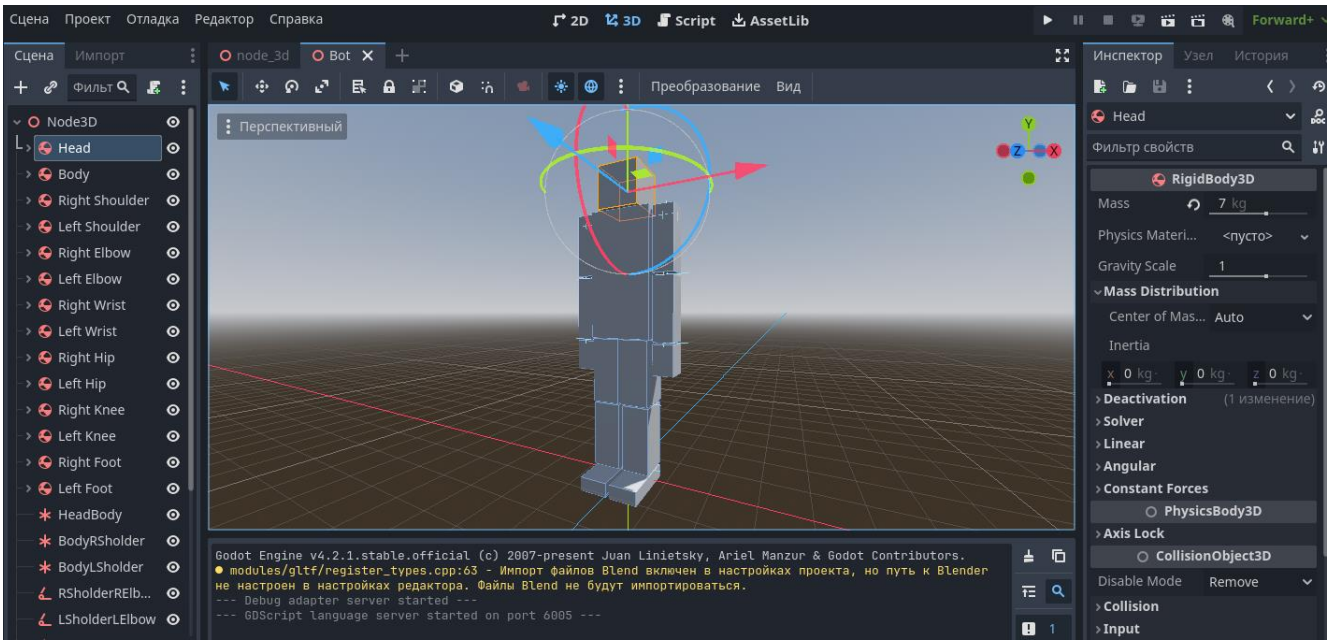
8. Evolutionary Algorithms, Thomas Bartz-Beielstein, Juergen Branke, Jorn Mehnen, May 2014

9. Transforming the future: a review of artificial intelligence models, Andrei A. Pugachev, Alina V. Kharchenko, Nikolai A. Sleptsov, December 2023

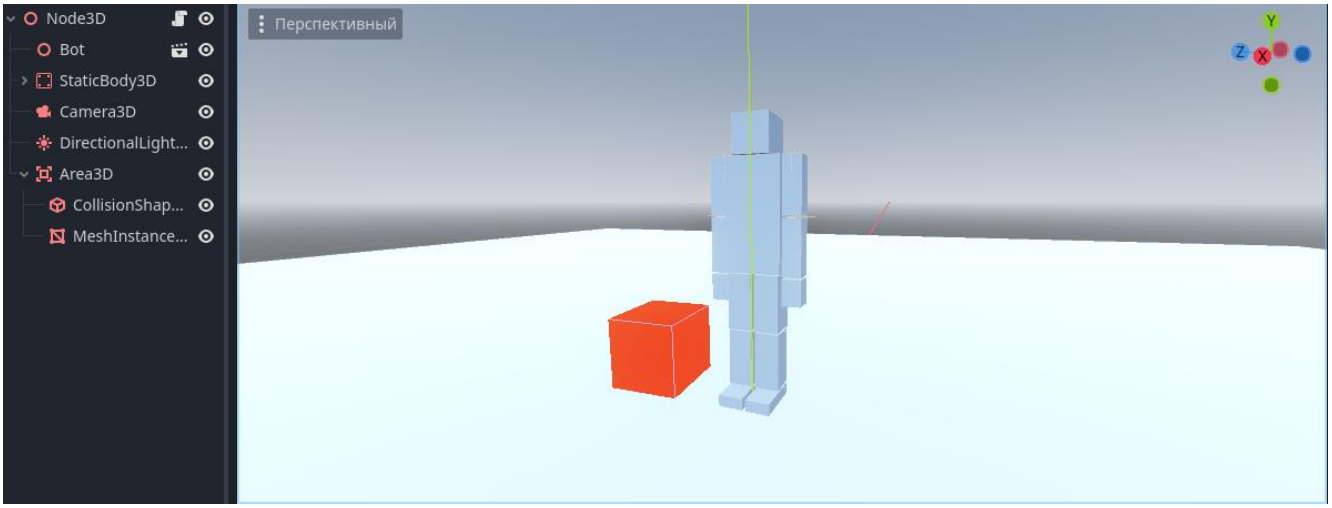
10. Exploring the Synergy of Artificial Intelligence and Robotics in Industry 4.0 Applications Introduction, Jeff Shuford, February 2024

9. ДОДАТКИ

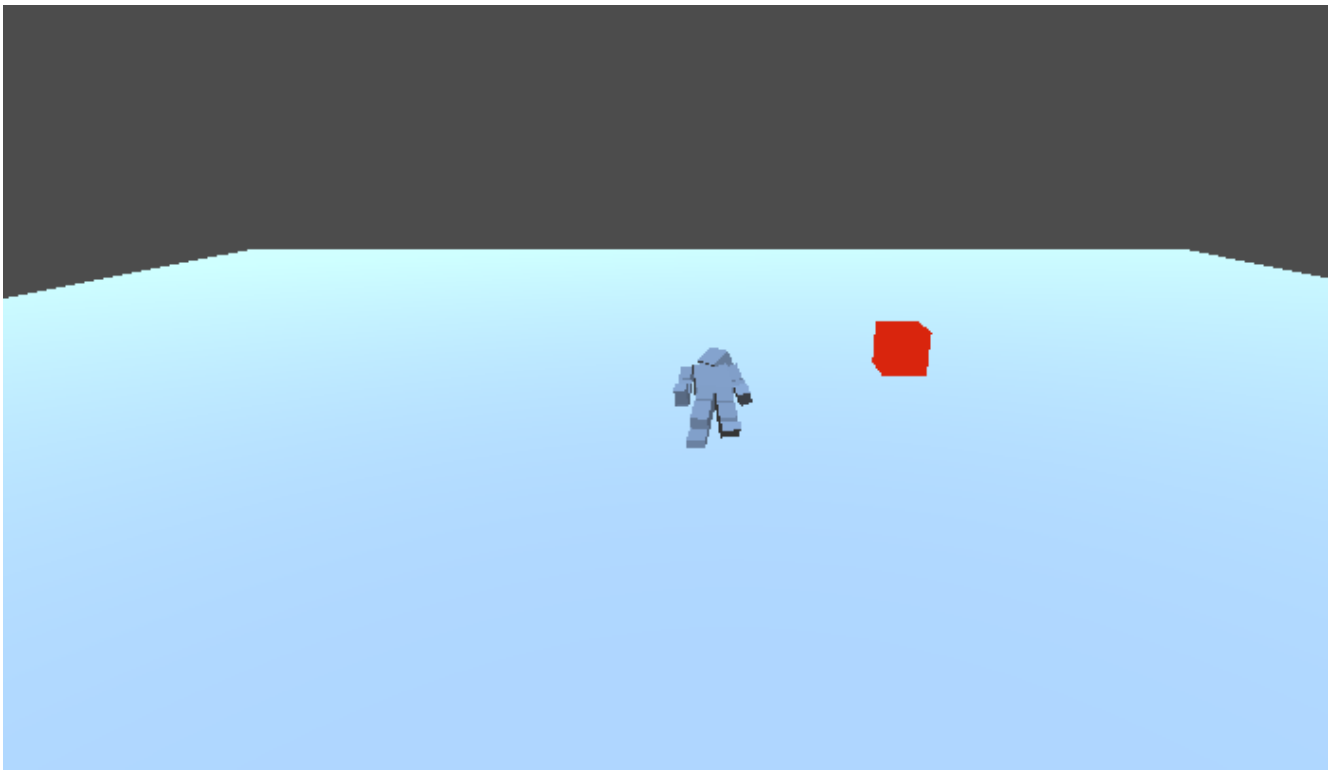
Додаток 1. Реалізація та фізичні властивості бота:



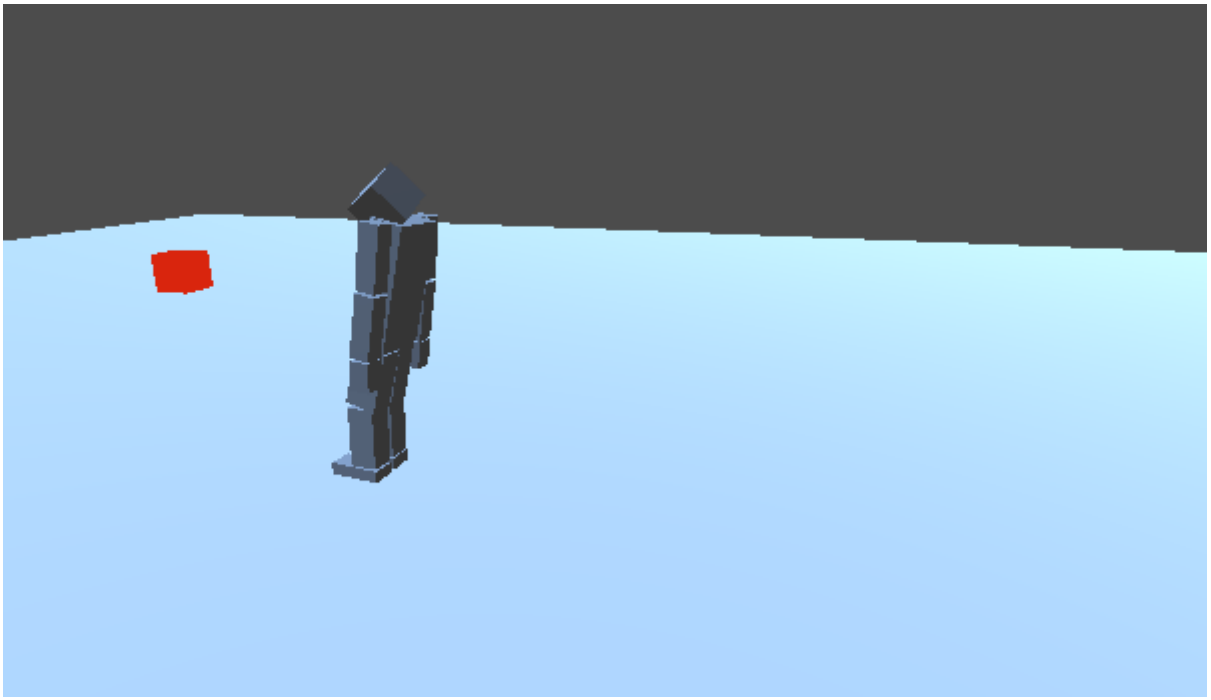
Додаток 2. Ігрова сцена:



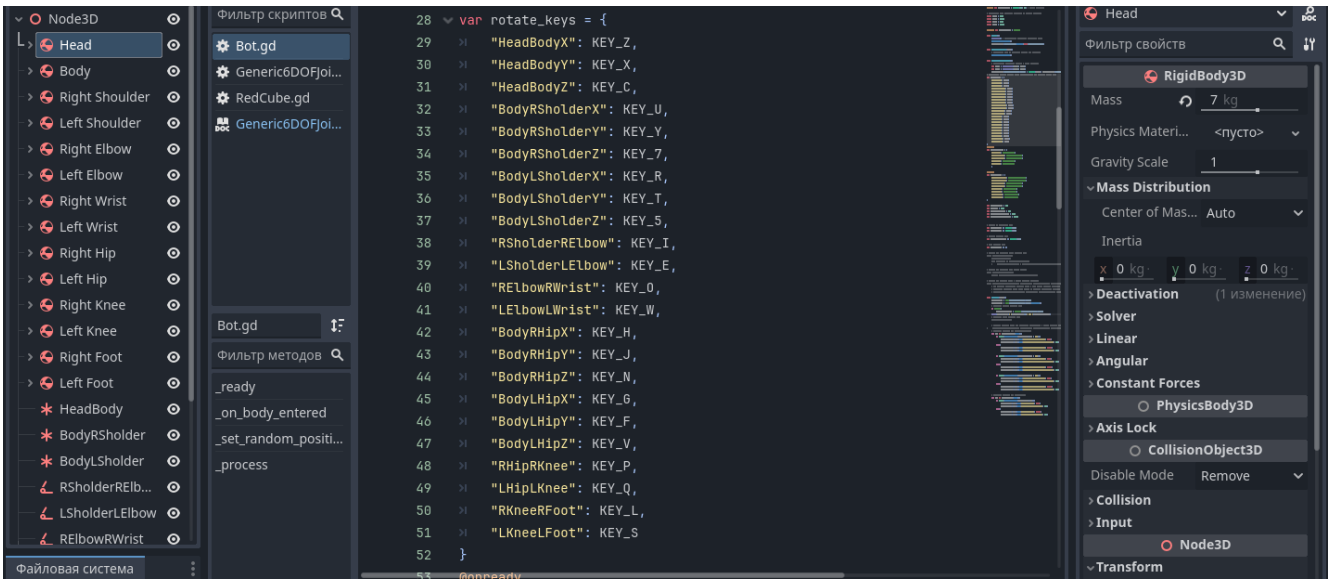
Додаток 3. Перщі кроки ШІ:



Додаток 4. Досягнення ШІ мети навчання:



Додаток 5. Клавіші певного руху суглобів гравця:



Додаток 6. Реалізація червоного куба:

```
3   @export var platform_size = Vector2(25, 25) # Розмір платформи
4
5   # Максимальні межі платформи (замініть на реальні значення)
6   const MAX_X = 25.0
7   const MIN_X = -25.0
8   const MAX_Y = 25.0
9   const MIN_Y = -25.0
10
11  @onready var red_cube_area = $Area3D
12
13  func _ready():
14      randomize() # Ініціалізація генератора випадкових чисел
15      _set_random_position()
16      red_cube_area.body_entered.connect(self._on_body_entered)
17
18  func _on_body_entered(body):
19      # Переміщення куба на випадкову позицію при зіткненні
20      _set_random_position()
21
22  func _set_random_position():
23      # Встановлення випадковоі позиції в межах платформи
24      var new_x = randf_range(MIN_X, MAX_X)
25      var new_y = randf_range(MIN_Y, MAX_Y)
26      red_cube_area.global_transform.origin = Vector3(new_x, red_cube_
```

Додаток 7. Змінення параметрів гравцем або ШІ моторів суглобів:

```
key in generic_joints.keys():
if Input.is_key_pressed(rotate_keys[key + "X"]):
    generic_joints[key].set("angular_motor_x/enable", true)
    generic_joints[key].set("angular_motor_x/target_velocity", 4.0
else:
    generic_joints[key].set("angular_motor_x/enable", false)
    generic_joints[key].set("angular_motor_x/target_velocity", 0.0

if Input.is_key_pressed(rotate_keys[key + "Y"]):
    generic_joints[key].set("angular_motor_y/enable", true)
    generic_joints[key].set("angular_motor_y/target_velocity", 4.0
else:
    generic_joints[key].set("angular_motor_y/enable", false)
    generic_joints[key].set("angular_motor_y/target_velocity", 0.0

if Input.is_key_pressed(rotate_keys[key + "Z"]):
    generic_joints[key].set("angular_motor_z/enable", true)
    generic_joints[key].set("angular_motor_z/target_velocity", 4.0
else:
    generic_joints[key].set("angular_motor_z/enable", false)
    generic_joints[key].set("angular_motor_z/target_velocity", 0.0
```

Додаток 8. Реалізація методу ШІ з підкріпленням:

```
# Оголошення класу для мережі Q
class_name QNetwork
# Визначення конструктора класу
func _init(state_dim, action_dim):
    # Ініціалізація шарів нейромережі
    fc1 = Linear.new()  # Перший повнозв'язний шар
    fc1.set("input_size", state_dim)  # Встановлення розміру вхідного шару
    fc1.set("output_size", 64)  # Встановлення розміру вихідного шару
    fc2 = Linear.new()  # Другий повнозв'язний шар
    fc2.set("input_size", 64)  # Встановлення розміру вхідного шару
    fc2.set("output_size", 64)  # Встановлення розміру вихідного шару
    fc3 = Linear.new()  # Третій повнозв'язний шар
    fc3.set("input_size", 64)  # Встановлення розміру вхідного шару
    fc3.set("output_size", action_dim)   # Встановлення розміру вихідного
шару
    # Додавання шарів до контейнера
    add_child(fc1)
    add_child(fc2)
    add_child(fc3)
# Функція для прямого поширення вперед через нейромережу
func forward(x):
    x = torch.relu(fc1.forward(x))   # Пряме поширення через перший шар з
ReLU-активацією
    x = torch.relu(fc2.forward(x))   # Пряме поширення через другий шар з
ReLU-активацією
    x = fc3.forward(x)  # Пряме поширення через третій шар
    return x
# Функція для вибору дії за ε-жадною стратегією
func epsilon_greedy_action(state, epsilon):
    if randf() < epsilon:
        return randi() % action_dim  # Випадкова дія
    else:
        var q_values = q_network.forward(state)
        return q_values.argmax()  # Вибір найкращої дії згідно з оцінкою Q-
функції
# Параметри навчання
var num_episodes = 10000  # Кількість епізодів навчання
```

```
var target_update = 10  # Частота оновлення цільової мережі
var batch_size = 32  # Розмір пакету даних для навчання
var gamma = 0.99  # Коефіцієнт зниження ваги майбутніх винагород
var epsilon_start = 1.0  # Початкове значення ε для ε-жадної стратегії
var epsilon_end = 0.01  # Кінцеве значення ε для ε-жадної стратегії
var epsilon_decay = 0.99  # Коефіцієнт зниження ε для ε-жадної стратегії
# Ініціалізація нейромережі та оптимізатора
var state_dim =  8,388,608 # розмір вектора стану
var action_dim =  8,388,608 # кількість можливих дій
var q_network = QNetwork.new(state_dim, action_dim)  # Ініціалізація мережі Q
var target_network = QNetwork.new(state_dim, action_dim)  # Ініціалізація цільової мережі Q
target_network.load_state_dict(q_network.get_state())  # Завантаження параметрів навчання з основної мережі в цільову мережу
var optimizer = optim.Adam(q_network.parameters(), lr=0.001)  # Ініціалізація оптимізатора Adam
# Основний цикл навчання
for episode in range(num_episodes):
    var state =  # початковий стан середовища
    var total_reward = 0  # Ініціалізація загальної винагороди для поточного епізоду
    while not done:
        var epsilon = max(epsilon_end, epsilon_start * epsilon_decay ** episode)  # Обчислення поточного значення ε
        var action = epsilon_greedy_action(state, epsilon)  # Вибір дії за ε-жадною стратегією
        var next_state, reward, done = perform_action(action)  # Виконання дії в середовищі та отримання наступного стану, винагороди та ознаки закінчення епізоду
        total_reward += reward  # Додавання винагороди до загальної винагороди
        var q_values = q_network.forward(state)  # Оцінка Q-функції для поточного стану
        var next_q_values = target_network.forward(next_state)  # Оцінка Q-функції для наступного стану з цільовою мережею
        var max_next_q_values = torch.max(next_q_values, dim=1)[0]  # Обчислення максимальних значень Q-функції для наступного стану
    var target_q_values = q_values.clone()  # Копіювання значень Q-функції
```

```
    target_q_values[range(batch_size), action] = reward + gamma * max_next_q_values * (1 - done)  # Обчислення цільових значень Q-функції
    var loss = nn.MSELoss()(q_values, target_q_values.detach())     # Розрахунок втрат
    optimizer.zero_grad()  # Обнулення градієнтів
    loss.backward()  # Обчислення градієнтів втрат
    optimizer.step()  # Оновлення параметрів нейромережі
    state = next_state  # Оновлення поточного стану
# Оновлення цільової мережі
if episode % target_update == 0:
    target_network.load_state_dict(q_network.get_state())     # Копіювання параметрів основної мережі в цільову мережу
# Вивід інформації про хід навчання
print("Episode", episode + 1, "/", num_episodes, ", Total Reward:", total_reward)
```

Додаток 9. Реалізація методу ШІ з еволюційного алгоритму:

```
# Оголошення класу для особини в популяції
class_name Individual:
    var genotype: Array  # Генотип особини
    var fitness: float  # Рівень пристосованості

    # Конструктор класу
    func _init(genotype: Array):
        self.genotype = genotype
        self.fitness = 0.0

    # Функція для обчислення рівня пристосованості
    func evaluate():
        # Оцінка пристосованості шляхом імітації ходьби бота в ігровому середовищі
        self.fitness = simulate_walk(self.genotype)

# Функція для створення початкової популяції
func create_initial_population(population_size: int, genotype_length: int) -> Array:
    var population = []
    for _ in range(population_size):
        var genotype = generate_random_genotype(genotype_length)
        var individual = Individual.new(genotype)
        population.append(individual)
    return population

# Функція для обчислення рівня пристосованості всієї популяції
func evaluate_population(population: Array):
    for individual in population:
        individual.evaluate()

# Функція для відбору найкращих особин у популяції
func select_best_individuals(population: Array, num_best: int) -> Array:
    population.sort_custom(lambda x, y: x.fitness > y.fitness)
    return population[0:num_best]

# Функція для здійснення кроссоверу між двома батьками
```

```
func crossover(parent1: Individual, parent2: Individual) -> Array:
    var crossover_point = randi() % parent1.genotype.size()
    var child1_genotype = parent1.genotype[0:crossover_point] +
parent2.genotype[crossover_point:]
    var child2_genotype = parent2.genotype[0:crossover_point] +
parent1.genotype[crossover_point:]
    return [Individual.new(child1_genotype),
Individual.new(child2_genotype)]

# Функція для проведення мутації у генотипі
func mutate(genotype: Array, mutation_rate: float) -> Array:
    for i in range(genotype.size()):
        if randf() < mutation_rate:
            genotype[i] = randomize_gene()
    return genotype

# Функція для запуску алгоритму еволюції
func evolutionary_algorithm(population_size: int, genotype_length: int,
num_generations: int, mutation_rate: float):
    # Створення початкової популяції
    var population = create_initial_population(population_size,
genotype_length)

    # Основний цикл еволюції
    for generation in range(num_generations):
        # Обчислення рівня пристосованості популяції
        evaluate_population(population)

        # Відбір найкращих особин
        var best_individuals = select_best_individuals(population,
population_size // 2)

        # Створення нової популяції за допомогою кроссоверу та мутації
        var new_population = []
        while new_population.size() < population_size:
            var parent1 = best_individuals[randi() %
best_individuals.size()]
            var parent2 = best_individuals[randi() %
best_individuals.size()]
```

```
            var children = crossover(parent1, parent2)
            for child in children:
                child.genotype = mutate(child.genotype, mutation_rate:
mutation_rate)
new_population.append(child)
    # Заміна старої популяції новою
    population = new_population

    # Вивід інформації про поточне покоління
    print("Generation:", generation + 1, ", Best Fitness:",
best_individuals[0].fitness)

# Повернення найкращої знайденої особини
return best_individuals[0]
```

Робота складається зі вступу, 5-ти глав, загальних висновків, списку використаної літератури (10), додатків (9). Зміст роботи висвітлено на 38 сторінках основного тексту і містить 2 таблиці.