МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІГРОВИЙ ЗАСТОСУНОК З ВИКОРИСТАННЯМ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ НА ІГРОВОМУ РУШІЇ 3D UNITY

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 402.22010325

Виконав студент 4-го курсу, групи 402
_____ О. Д. Черкас
«17» червня 2024 р.

Керівник: д-р техн. наук, проф.
_____І. М. Журавська
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.

_____ Ю. П. Кондратенко

«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Черкасу Олександру Дмитровичу

1. Тема кваліфікаційної роботи «Ігровий застосунок з використанням методів штучного інтелекту на ігровому рушії 3D Unity».

Керівник роботи Журавська Ірина Миколаївна, д-р техн. наук, професор.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: ринок ігрової індустрії, підходи та методи до створення 3D відеогри, підходи та методи створення та навчання ШІ до відеогри.

Очікуваний результат: 3D-гра в жанрі спортивної гри, розроблена шляхом використання середовища розробки Unity зі штучним інтелектом.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- розкрити теоретичні засади створення 3D-відеогри;
- огляд існуючих аналогів;
- розкрити теоретичні засади навчання нейронних мереж (НМ) для 3D-гри;
- обґрунтувати вибір інструментальних засобів розробки гри;
- розробити та здійснити програмну реалізацію 3D-гри.

5. Перелік графічного матеріалу: презентація, 24 картинки.

6. Завдання до спеціальної частини: «Охорона праці при дистанційній роботі»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __д-р техн. наук, проф. Журавська І. М._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Черкас О. Д._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Ігровий застосунок з використанням методів штучного інтелекту на ігровому рушії 3D Unity

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз існуючих аналогів, написання вступу, завершення розробки застосунку, написання розділів.	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	18.06.2024	
12	Захист БКР перед екзаменаційною комісією (ЕК)	24.06.2024	24.06.2024	

Розробив студент __Черкас О. Д._____ _____

(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ д-р техн. наук, проф. Журавська І. М. _____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи студента групи 402 ЧНУ ім. Петра Могили

Черкас Олександра Дмитровича

Тема: «Ігровий застосунок з використанням методів штучного інтелекту на ігровому рушії 3d unity»

Керівник: зав. каф. комп'ютерної інженерії, д-р техн. наук, проф. Журавська І. М.

Кваліфікаційна робота присвячена розробці програмної реалізації 3D-гри з інтеграцією штучного інтелекту на базі ігрового рушія Unity.

Об’єктом дослідження: процес створення комп'ютерних ігор у середовищі розробки 3D Unity з використанням методів штучного інтелекту.

Предмет дослідження: сучасні методи та технології створення і навчання нейронних мереж для ігрових застосунків.

Метою дипломної роботи є створення більш реалістичних ігрових персонажів і сценаріїв для використання ігри як середовища для тестування і вдосконалення алгоритмів штучного інтелекту.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка містить вступ, три основні розділи, висновки, список використаних джерел, додатки.

У першому розділі розглядається сучасний вплив відеоігр на світову економіку, використання нейронних мереж у сучасному геймдеві, огляд особливостей і аналогів ігор у обраному жанрі та сучасні ігрові двигуни.

У другому розділі наведено інструменти для розробки 3D-гри та обґрунтовано їх вибір.

Третій розділ містить опис процесу розробки ігрового застосунку та процесу навчання нейронної мережі.

В результаті розроблено 3D-гру з навченою нейронною мережею в жанрі спортивної гри.

Бакалаврська кваліфікаційна робота містить 90 сторінок, 24 рисунків, 15 використаних джерел та 3 додатка.

***Ключові слова**: комп’ютерна гра, Unity, Blender, 3D-гра, спортивний жанр, нейронна мережа, штучний інтелект.*

ABSTRACT

qualification work of a student of group 402 of the Petro Mohyla Black Sea National University

Cherkas Oleksandr Dmytrovych

Theme: "Game application using artificial intelligence methods on the 3d unity game engine"

The qualification work is devoted to the development of a software implementation of a 3D game with the integration of artificial intelligence based on the Unity game engine.

Object of research: the process of creating computer games in the 3D Unity development environment using artificial intelligence methods.

Subject of research: modern methods and technologies for creating and training neural networks for gaming applications.

The purpose of the thesis is to create more realistic game characters and scenarios for using games as an environment for testing and improving artificial intelligence algorithms.

The paper consists of a professional section and a special section on labor protection. The explanatory note contains an introduction, three main chapters, conclusions, a list of references, and appendices.

The first chapter discusses the current impact of video games on the global economy, the use of neural networks in modern game development, an overview of the features and analogues of games in the chosen genre, and modern game engines.

The second section describes tools for 3D game development and justifies their choice.

The third section describes the process of developing a game application and the process of training a neural network.

As a result, a 3D game with a trained neural network in the genre of a sports game was developed.

The bachelor's thesis contains 90 pages, 24 figures, 15 references and 3 appendices.

Keywords: *computer game, Unity, Blender, 3D game, sports genre, neural network, artificial intelligence.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

МЛБ – головна ліга бейсболу

НБА – національна баскетбольна асоціація

НМ – нейронна мережа

НФЛ – національна футбольна ліга

ПЗ – програмне забезпечення

НХЛ – національна хокейна ліга

ПК – персональний комп’ютер

ШІ – штучний інтелект

CIL – Common Intermediate Language

DXR – DirectX Raytracing

ESA – Entertainment Software Association

GameDev – Game Development

IDE – Integrated Development Environment

IT – Information Technology

NPC – Non-Player Character

PPO – Proximal Policy Optimization

ВСТУП

Розваги можуть бути не тільки джерелом тимчасового задоволення, але й стати ключем до саморозвитку. Ігри – один з прикладів розвитку людини через розваги. Сьогодні ігрові застосунки дуже поширені і є невід'ємною частиною людського дозвілля, вони мають величезний вплив на культуру, світогляд і масову свідомість.

Ігрові застосунки з'явилися приблизно в другій половині минулого століття і являли собою скоріше науковими роботами або виробами для виставок, ніж розвагою для мас. З розвитком технологій і ростом потужністю комп'ютерів, ігри розвивалися і ставали більш популярними.

Один з найбільш відомих ігрових рушіїв є Unity; він дозволяє розробляти різні за жанром ігрові застосунки, ігри для різних платформ, таких як Android, iOS, ПК або консоль. Він здобув своє місце серед ігрових рушіїв завдяки безкоштовному розповсюдженню і дуже простому освоєнню його механік та інструментів.

Наряду з великими IT-компаніями існують великі GameDev-студії, які продають мільйони копій своїх ігор по всьому світу. Команди для створення ігор включають людей з різних сфер: маркетологи, менеджери, 3D-художники, художники концепт-артів, програмісти, музиканти і т. п.

На сьогодні одна з найцікавіших та набираючих популярність сфера – це сфера штучного інтелекту (ШІ). Останнім часом з'явилося багато різних продуктів, які допомагають виконувати різноманітну роботу, наприклад: створення зображень, переклад текстів та відеороликів в реальному часі. Розробники застосовують ШІ, щоб ефективніше виконувати задачі, які до цього доводилось виконувати вручну.

Отже, розробка ігор на основі ігрового рушія Unity з використанням штучного інтелекту є актуальною та має науково-практичний вплив на сучасний світ.

Предметом кваліфікаційної роботи є інструменти та технології розроби гри зі штучним інтелектом. Дослідження включає аналіз принципів розробки ігор на

основі ігрового рушія Unity, навчання нейронної мережі для інтеграції в ігровий застосунок, налагодження і тестування гри. В результаті дослідження буде створена гра з інтегрованою і навченою нейронною мережею (НМ).

Метою дипломної роботи є створення більш реалістичних ігрових персонажів і сценаріїв для використання ігри як середовища для тестування і вдосконалення алгоритмів штучного інтелекту.

Завдання, які потрібно виконати для досягнення мети:

- аналіз аналогів та механік притаманних обраному жанру;
- розробка основних механік гри;
- аналіз створення оточення для навчання НМ;
- навчання НМ у створеному оточенні;
- створення візуальної частини застосунку;
- розробка графічного інтерфейсу гри.

Об’єктом роботи є процес створення комп'ютерних ігор у середовищі розробки 3D Unity з використанням методів штучного інтелекту.

Предметом роботи є сучасні методи та технології створення і навчання нейронних мереж для ігрових застосунків.

Характеристики розроблювального проєкту:

- мова програмування – C#, мова для роботи з рушієм Unity;
- ігровий рушій Unity;
- гра призначена для одного гравця;
- для навчання використовується мова Python та бібліотеки для роботи застосунку.

1 ВПЛИВ ІГРОВОЇ ІНДУСТРІЇ НА СУЧАСНУ ЕКОНОМІКУ ТА ЗАСОБИ СТВОРЕННЯ КОМП'ЮТЕРНИХ 3D-ІГОР

1.1 Огляд предметної області

Ігрова індустрія поступово стає найбільш значущою у глобальній економіці. Так виходячи зі звіту **Newzoo**, стало відомо, що обсяг доходів ігрової індустрії за 2017 рік склав 116 млрд доларів США (рис. 1). Це на 10,7 % більше, ніж до цього. Також згідно зі звітом Асоціації розважального програмного забезпечення (англ. Entertainment Software Association, ESA), 2019 року індустрія відеоігор принесла економіці США 90,3 млрд доларів [1].

Індустрія відеоігор охоплює компанії, що займаються розробкою, виробництвом і розповсюдженням відеоігор на різних платформах, таких як консолі, ПК і мобільні пристрої. Крім цих компаній, вигоду від індустрії отримують і багато інших підприємств, наприклад маркетингові агентства та виробники товарів (рис. 1.1).

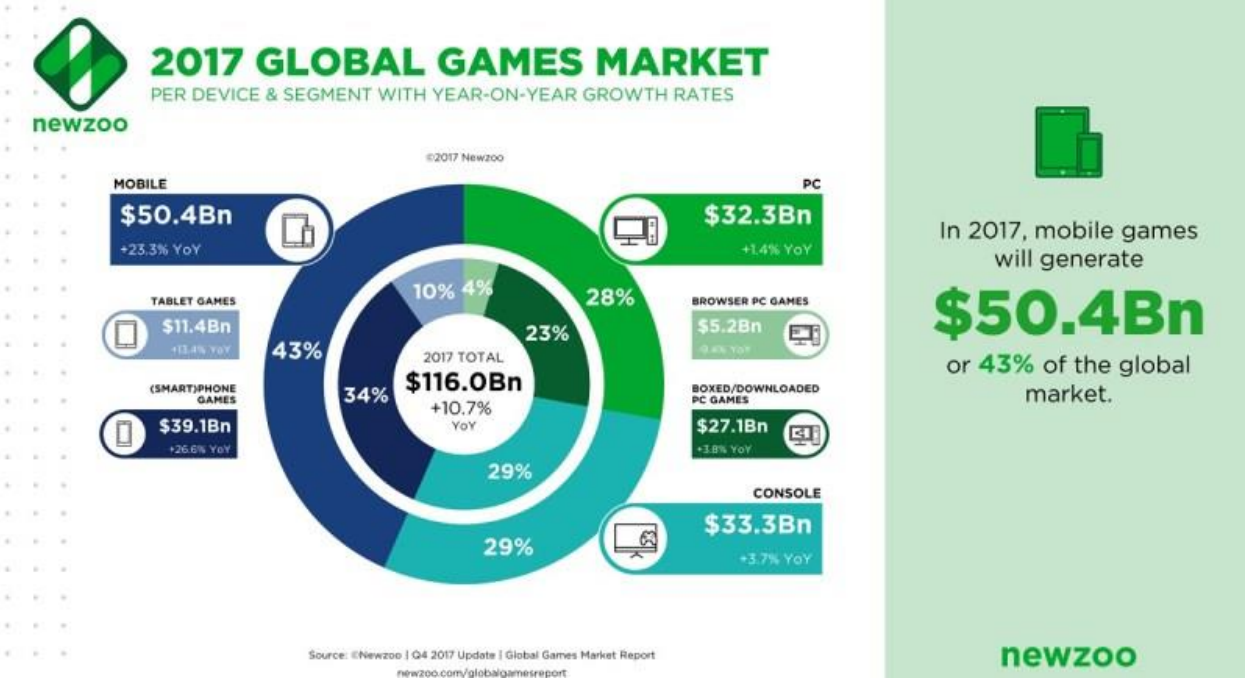

Рисунок 1.1 – Фінансовий звіт від Newzoo про ринок ігрової індустрії за 2017 рік [1]

Індустрія відеоігор також сприяла зростанню суміжних галузей, таких як технології та роздрібна торгівля. Для запуску відеоігор потрібні передові технології, тому індустрія стимулює розвиток нових та інноваційних технологій. Це призвело до зростання технологічної галузі: компанії розробляють нове обладнання та програмне забезпечення для підтримки відеоігор [2].

Індустрія відеоігор також мала значний вплив на роздрібну торгівлю. Відеоігри продаються в роздрібних магазинах, що призвело до зростання роздрібної торгівлі. Такі роздрібні мережі, як GameStop і Best Buy, мають спеціальні відділи для відеоігор, і продаж відеоігор значно збільшує їхні доходи.

Крім технологій і роздрібної торгівлі, індустрія відеоігор також сприяла розвитку інших галузей, таких як музика і кіно. Багато відеоігор містять оригінальну музику і озвучку, що призвело до зростання музичної та акторської індустрії. Деякі популярні відеоігри також були адаптовані у фільми, що призвело до зростання кіноіндустрії.

Крім економічного впливу, індустрія відеоігор справила значний соціальний і культурний вплив. Відеоігри стали популярною формою соціальної взаємодії, і багатьох людей об'єднує спільний ігровий досвід. Онлайн-ігри також дали змогу людям спілкуватися з іншими людьми по всьому світу, руйнуючи культурні бар'єри і заохочуючи різноманітність.

Відеоігри також використовують як інструмент для освіти та соціальних змін. Багато навчальних закладів і некомерційних організацій використовують відеоігри для навчання школярів таких предметів, як історія, наука і соціальні проблеми. Відеоігри також використовують для підвищення обізнаності про такі соціальні проблеми, як бідність, голод і збереження навколишнього середовища.

На закінчення слід зазначити, що відеоігри мають значний вплив на економіку, а також на суспільство і культуру. Щорічно індустрія відеоігор

приносить мільярди доларів в економіку, і в ній зайняті тисячі людей на різних посадах. Відеоігри також сприяють збільшенню споживчих витрат і зростанню суміжних галузей, таких як технології та роздрібна торгівля.

Соціальний і культурний вплив відеоігор також значний, оскільки вони стали популярною формою соціальної взаємодії і використовуються для просування різноманітності та соціальних змін. У міру того як індустрія відеоігор зростатиме і розвиватиметься, її вплив на економіку і суспільство ставатиме дедалі значнішим.

1.2 Штучний інтелект в сучасному GameDev

Штучний інтелект (ШІ) здатний повністю змінити індустрію відеоігор – від того, як розробляються ігри, до того, як у них грають. ШІ обіцяє відкрити нові горизонти в плані масштабу, реалізму, інтерактивності та багато чого іншого, що може докорінно змінити ігри, якими ми їх знаємо [3].

ШІ може значно поліпшити і прискорити багато аспектів створення і виробництва ігор.

1.2.1 Автоматизоване створення світу і генерація активів

Ручне створення великих 3D-середовищ, детальних активів і складних ігрових елементів вимагає величезних витрат часу і ресурсів. Інструменти штучного інтелекту дають змогу автоматизувати більшу частину цього процесу, алгоритмічно генеруючи світи, текстури, моделі, об'єкти та інші активи. Це дає змогу розробникам швидко створювати насичені, яскраві ігрові простори, створення яких вручну було б неправдоподібно трудомістким.

Наприклад, система штучного інтелекту може аналізувати реальні архітектурні дані та інформацію про місцевість, щоб побудувати реалістичні 3D-будівлі та природні ландшафти для ігрового середовища з відкритим світом так наприклад в грі Infection Free Zone (рис. 1.2) використовуються алгоритми ШІ для генерації карт но основі реальної місцевості. ШІ може враховувати такі фактори, як оптимальне використання простору, лінії огляду, можливості укриття і доступність,

щоб генерувати структури, які підходять для захопливого ігрового процесу. Він також може імітувати реальні естетичні дизайни і планування, щоб зробити оточення візуально достовірним.

Рисунок 1.2 – Генерація карти з реальної місцевості у грі Infection Free Zone

Переваги також поширюються на дрібні ігрові активи: за допомогою ШІ можна створювати нескінченну кількість невеликих моделей 3D-об'єктів, як-от меблі, предмети безладу, рослини та реквізит, щоб щільно заселити ігрові світи, що зазвичай неможливо зробити вручну. Заощаджений час розробки дає змогу творцям зосередитися на вищому рівні дизайну та творчості.

1.2.2 Аналіз та оптимізація ігрових даних

Алгоритми штучного інтелекту можуть працювати з такими ігровими даними, як 3D-сітки, текстури, аудіофайли, геометрія оточення і багато іншого, щоб стиснути їх без негативного впливу на візуальне оформлення, якість звуку і враження гравця. Стиснення розмірів файлів даних дає змогу значно підвищити

загальну продуктивність гри, прискорити завантаження і зробити ігровий процес більш плавним.

ШІ також може динамічно регулювати розподіл внутрішньоігрових ресурсів на льоту, проводячи аналіз продуктивності в режимі реального часу і надаючи ресурси ігровим елементам у міру необхідності. Завдяки такому балансуванню навантаження ігри завжди використовують доступні обчислювальні потужності найбільш ефективним чином для оптимальної роботи.

1.2.3 Автоматизоване тестування ігор та ітерації поліпшень

Широке ігрове тестування дуже важливе для виявлення проблем з ігровою механікою, балансом, складністю та іншим перед випуском. Однак всебічне ігрове тестування складних ігор вимагає величезних людських зусиль. ШІ-інструменти для моделювання на основі алгоритмів машинного навчання можуть проходити ігри набагато швидше, ніж людина, при цьому точно моделюючи поведінку людини. Це дає змогу проводити всебічне тестування ігор за лічені дні, а не за тижні та місяці.

Гранульовані дані, одержувані під час ігрового тестування за допомогою ШІ, також дають повнішу інформацію порівняно з відгуками людей. Розробники можуть точно налаштовувати і доопрацьовувати ігри, ґрунтуючись на конкретних показниках і візуалізаціях, наданих ШІ, що тестує, про те, що працює, а що ні. У підсумку ігри більш високої якості створюються в більш короткі терміни.

Впровадження ШІ в ігрові системи та механіки може зробити оточення відеоігор живішим, реактивнішим і таким, що відповідає реальності:

1.2.4 Фотореалістична графіка та оточення

Сьогодні навіть графічно складні ігри мають помітні обмеження на рендеринг текстур та об'єктів у великих оточеннях. Інструменти штучного інтелекту, як-от GauGAN від Nvidia, можуть аналізувати дані ландшафтних знімків для створення практично фотореалістичної графіки та рендерингу оточення. Ігри, що використовують подібні системи, можуть дозволити гравцям створювати ігрові

світи з надзвичайною візуальною точністю на величезних просторах без помітного повторення текстур або активів. Такі ефекти, як погодні умови, рух листя і поширення вогню, також можуть поводитися реалістично, а не здаватися повторюваними або програмними.

1.2.5 NPC зі справжніми емоціями, особистістю та діапазоном

Поведінка неігрових персонажів (англ. Non-Player Character, NPC) у сучасних відеоіграх, як правило, доволі спрощена, повторювана і невиразна. ШІ може наділити NPC набагато більшою емоційною глибиною і мінливістю в їхніх реакціях на ігрові події та дії гравця. Їхній діалог міг би коригуватися на льоту, щоб посилатися на спільний досвід, який вони пережили з гравцем, сприяючи встановленню більш значущих зв'язків. Поведінка NPC може істотно змінюватися, водночас зберігаючи автентичність їхньої особистості та передісторії. Відносини між NPC можуть динамічно розвиватися залежно від взаємодії з гравцем, що загалом призведе до появи NPC, які будуть більше схожі на переконливих, багатовимірних персонажів, ніж на роботів, які видають квести.

NPC у «The Witcher 3» (рис. 1.3) продемонстрували складну поведінку, включно зі щоденними рутинами, реакціями на погоду та взаємодією з іншими персонажами. Це створювало переконливий, живий світ, у якому дії та рішення гравця впливали на навколишнє середовище і стосунки з NPC [20].

Рисунок 1.3 – Кадр з гри «The Witcher 3»

1.2.6 Ігрові світи, які реагують на гравця

Навіть у найбільш наративно розгалужених сучасних відеоіграх діапазон способів, якими ігрові світи можуть реагувати на вибір гравця, за своєю суттю обмежений складністю розробки. ШІ може концептуалізувати й актуалізувати ігрові простори, які змінюють свою форму у відповідь на поведінку користувача, практично необмеженою мірою в межах обмежених параметрів. Якщо гравець ухвалює рішення, яке змінює траєкторію руху певних фракцій або персонажів у світі, ШІ може згенерувати логічні наслідки, які відповідним чином змінять локації, взаємодії та квести, створюючи величезний потенціал для нелінійного розвитку сюжету під керівництвом гравця. Таким чином, можливість повторного проходження значно зростає. Так, наприклад, у грі No Man’s Sky світи генеруються у необмеженої кількості, що дозволяє зімітувати розмір галактики, чи навіть більше (рис. 1.4).

Рисунок 1.4 – Генерація світу в грі No Man’s Sky

1.2.7 Персоналізована подорож

Подібно до того, як розумні ШІ-помічники з часом вивчають уподобання своїх користувачів, щоб краще передбачати їхні потреби, ігровий ШІ може спостерігати за поведінкою гравця впродовж десятків або сотень годин, щоб вибудовувати сюжетні гілки, завдання та нагороди, адаптовані виключно під конкретного гравця. У грі можуть з'явитися компаньйони, які доповнюватимуть або суперечитимуть вашому ігровому стилю і характеру. Оточення може робити акцент на дослідженні, а не на дії, залежно від того, чому ви віддаєте перевагу – головоломкам чи боям. Кожен гравець може відчути, що його гру створено спеціально для нього, що призведе до сильніших емоційних вкладень і задоволення.

1.2.8 Анімація та фізична достовірність

Сучасні анімації в іграх, як правило, мають невловимо штучну якість, оскільки вони являють собою захоплені рухами акторів, які потім змішуються разом. Аналіз величезної кількості відеозаписів, на яких зафіксовано, як люди орієнтуються в навколишньому середовищі та фізично реагують на перешкоди в незліченних реальних умовах, за допомогою штучного інтелекту дасть змогу створити гіперреалістичну анімацію. Персонажі могли б рухатися і реагувати з плавністю і динамічністю справжніх людей. Фізика також буде менше схожа на апроксимацію і більше на реальність – об'єкти розлітатимуться на шматки, вітер дутиме, частинки розсіюватимуться і т. д., і все це поводитиметься саме так, як це природно, завдяки симуляції ШІ. Така точність фізики значно посилює занурення.

1.2.9 Динамічна складність і темп

Сьогодні в більшості ігор складно збалансувати складність залежно від рівня майстерності гравця. ШІ-режисер, що стежить за діями гравця в режимі реального часу, міг би динамічно і плавно посилювати або зменшувати небезпеку, щоб забезпечити ідеально збалансовані рівні складності для індивідуального зростання здібностей і майстерності. Він також міг би регулювати темп розкриття сюжету, головоломок, бойових зіткнень тощо, щоб елегантно відповідати вподобанням гравця, не допускаючи нудьги. Таким чином, ігри будуть постійно захоплювати гравця.

Таким чином, ШІ відіграє все більш важливу роль у розробці відеоігор. Його використовують для створення реалістичних та складних неігрових персонажів (NPC), динамічних ігрових світів, адаптивних рівнів складності та багато іншого. Застосування ШІ у відеоіграх дозволяє зробити їх більш захоплюючими, динамічними та цікавими для гравців.

У разі вмілого використання ШІ призведе до зміни парадигми відеоігор, починаючи з моменту їхнього розроблення і закінчуючи вісцеральними відчуттями від гри. Незабаром можуть бути реалізовані величезні інтерактивні світи з безпрецедентним рівнем деталізації, реактивності та адаптації. І це лише мала

дещиця можливого: у міру розвитку технологій ШІ зростатиме і потенціал їх застосування в іграх.

Однак настільки широка інтеграція ШІ пов'язана з певними ризиками і проблемами, які вимагають відповідального підходу до її реалізації. Але за розумного підходу ШІ може значно посилити емоційну взаємодію з віртуальними світами і кардинально змінити ігровий процес.

1.3 Проєкти в жанрі спортивних ігор

Спортивні ігри – це жанр відеоігор, що імітують спорт. Зазвичай вони засновані на реальних видах спорту, але можуть бути й вигаданими або перебільшеними. Зазвичай у таких іграх гравець керує одним або кількома спортсменами під час змагань. Гравці повинні слідувати правилам виду спорту і грати або проти керованих комп'ютером супротивників, або проти інших гравців.

Існують спортивні відеоігри практично для всіх видів спорту, включно з бейсболом, баскетболом, футболом, гольфом, хокеєм, перегонами та іншими. Багато спортивних відеоігор засновані на реальних спортивних лігах, таких як НФЛ, НБА, МЛБ і НХЛ. Інші засновані на олімпійському або аматорському спорті.

1.3.1 Проєкти, засновані на футболі

Футбол, як один з найпопулярніших видів спорту у світі, не оминув й увагу розробників відеоігор. Існує безліч проєктів, присвячених цій грі, які охоплюють різні жанри та платформи.

Найпопулярнішим жанром у цій категорії безперечно є симулятори футболу. Ці ігри прагнуть максимально реалістично відтворити атмосферу та динаміку футбольного матчу. До найвідоміших симуляторів належать серії FIFA від EA Sports та Pro Evolution Soccer від Konami (рис. 1.5). Ці ігри пропонують гравцям можливість керувати командами та гравцями з реального світу, брати участь у чемпіонатах та кубках, а також створювати власні команди.

Рисунок 1.5 – Геймплей гри FIFA

1.3.2 Проєкти, засновані на перегонах

Перегони – це ще один популярний жанр спортивних ігор, який пропонує гравцям випробувати свої навички водіння та змагатися з іншими на різноманітних треках. Існує безліч піджанрів ігор про перегони, які здатні захопити будь-якого любителя швидкості та азарту.

Одним з найпоширеніших піджанрів перегонів є аркадні перегони. Ці ігри пропонують швидке та динамічне ігрове досвідчення з акцентом на розваги. Вони часто мають мультяшну графіку та прості правила. Прикладами аркадних перегонів є Mario Kart (рис. 1.6) та Crash Team Racing.

Рисунок 1.6 – Геймплей гри Mario kart

1.3.3 Баскетбол у світі відеоігор

Баскетбол, один з найпопулярніших видів спорту у світі, не оминув й увагу розробників відеоігор. Існує безліч проєктів, присвячених цій грі, які охоплюють різні жанри та платформи.

Найпопулярнішим жанром у цій категорії безперечно є симулятори баскетболу. Ці ігри прагнуть максимально реалістично відтворити атмосферу та динаміку баскетбольного матчу. До найвідоміших симуляторів належать серії NBA 2K від 2K Sports та NBA Live від EA Sports (рис. 1.7). Ці ігри пропонують гравцям можливість керувати легендами НБА, грати за команди з реального світу, а також створювати власних гравців.

Рисунок 1.7 – Геймплей гри NBA 2K

1.4 Ігрові рушії

Ігровий рушій – це середовище розроблення програмного забезпечення, яке також називають «ігровою архітектурою» або «ігровим фреймворком», з налаштуваннями та конфігураціями, що оптимізують і спрощують розроблення відеоігор на різних мовах програмування. Ігровий рушій може містити рушій рендерингу 2D- або 3D-графіки, сумісний із різними форматами імпорту, фізичний рушій, що моделює реальні дії, штучний інтелект (ШІ), який автоматично реагує на дії гравця, звуковий рушій, що керує звуковими ефектами, анімаційний рушій і безліч інших функцій.

Перші відеоігри розроблялися на власних рушіях рендерінгу, кожен з яких був призначений спеціально для однієї гри. Згодом ігрові рушії еволюціонували від пропрієтарних рушіїв власного виробництва до комерційних рушіїв, які широко доступні сьогодні. Розробники ігор, попит на які є надзвичайно високим, можуть спростити та прискорити процес розробки ігор, використовуючи комерційно розроблені ігрові рушії для створення нових ігор або розширення наявних ігор на додаткові платформи [6].

Деякі популярні ігрові рушії:

– Unity: один з найпопулярніших ігрових рушіїв, який використовується для створення ігор для ПК, консолей, мобільних пристроїв та браузерів;

– Unreal Engine: потужний ігровий рушій, який використовується для створення AAA-ігор з високоякісною графікою;

– Godot: безкоштовний і відкритий ігровий рушій, який підтримує 2D- та 3D-розробку.

Вибір ігрового движка залежить від потреб проєкту, досвіду розробників та бюджету. Важливо ретельно вивчити доступні варіанти та вибрати рушій, який найкраще відповідає поставленим потребам.

1.4.1 Ігровий рушій Unity

Unity – це багатоплатформний ігровий двигун, розроблений компанією Unity Technologies. Він широко використовується для створення 2D- та 3D-ігор для ПК, консолей, мобільних пристроїв та браузерів [18]. Unity відомий своєю простотою використання, гнучкістю та потужністю, що робить його популярним вибором як для початківців, так і для досвідчених розробників. Вікно інтерфейсу представлене нижче (рис. 1.8).

Основні можливості Unity:

– **візуальне програмування:** Unity використовує систему візуального програмування під назвою C#, яка дозволяє розробникам створювати ігрову логіку без глибоких знань програмування;

– **2D- та 3D-розробка:** Unity підтримує розробку як 2D-, так і 3D-ігор. Він пропонує широкий спектр інструментів та функцій для створення персонажів, об'єктів, декорацій та інших ігрових елементів;

– **фізика:** Unity має вбудований фізичний рушій, який дозволяє реалістично моделювати фізичні явища, такі як падіння предметів, зіткнення та рух транспортних засобів;

– **штучний інтелект:** Unity підтримує розробку ігор з використанням штучного інтелекту (ШІ). Розробники можуть створювати NPC, які реагують на дії гравця та самостійно приймають рішення;

– **звук:** Unity має вбудований звуковий рушій, який дозволяє відтворювати звукові ефекти, музику та озвучку в грі;

– **мережеві можливості:** Unity підтримує розробку багатокористувацьких ігор. Розробники можуть створювати сервери та клієнти, які дозволяють гравцям взаємодіяти один з одним;

– **підтримка VR/AR:** Unity підтримує розробку ігор для віртуальної та доповненої реальності (VR/AR);

– **спільнота та ресурси:** Unity має велику та активну спільноту розробників. Доступно безліч навчальних посібників, документації та ресурсів, які допоможуть розпочати роботу з Unity.

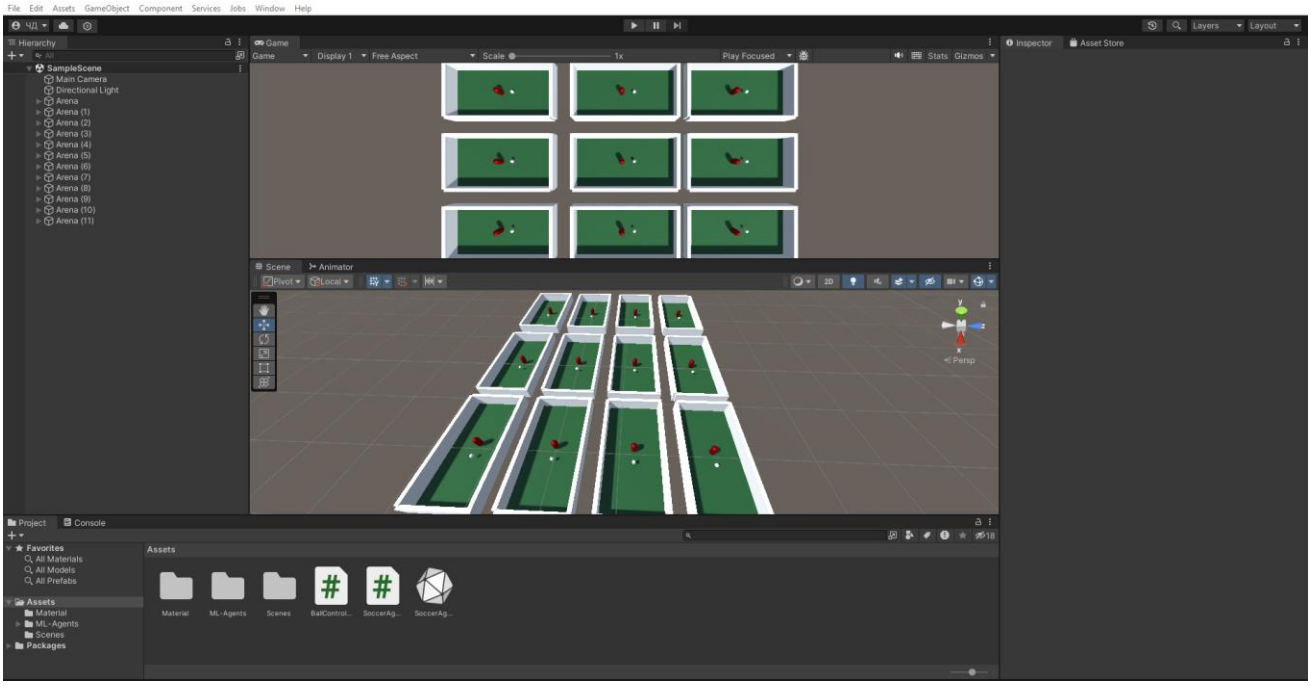

Рисунок 1.8 – Вікно інтерфейсу ігрового рушія Unity

Unity – це чудовий вибір для розробників, які хочуть створювати 2D- та 3D-ігри для різних платформ. Його перевагами є простота використання, гнучкість та

потужність, що роблять його популярним вибором як для початківців, так і для досвідчених розробників.

1.4.2 Ігровий рушій Unreal Engine

Unreal Engine – це система рендерингу, що дає змогу створювати цілі віртуальні світи за допомогою найкращих в індустрії інтерактивних інструментів у режимі реального часу. Програмне забезпечення для рендерингу розробив 1995 року засновник Epic Games Тім Суїні, розробник-самоук, разом із Джеймсом Шмальцем і Кліффом Блезінскі, які є легендами в галузі ігор і візуальних ефектів [8].

Спочатку Unreal була створена для створення гри, яка згодом стане «Unreal», – шутера від першої особи. Вона розроблялася протягом кількох років, поки не дебютувала в 1998 році. Суїні написав 90 % коду рушія, «включно з графікою, інструментами та мережевою взаємодією».

Unreal складається з функцій, концепцій та інструментів, які використовуються для створення будь-яких візуальних проєктів, включно з кінематографічними фільмами, телесеріалами, мобільними іграми та архітектурними твореннями з приголомшливою роздільною здатністю і якістю (рис. 1.9). Він використовує міць графіки NVIDIA RTX і Microsoft DirectX Raytracing (DXR) для створення візуально вражаючих реалістичних ефектів, включно з фізично точними віддзеркаленнями, м'якими тінями від плоского освітлення, ambient occlusion, напівпрозорістю, освітленням Sky Light і освітленням на основі зображень.

Рисунок 1.9 – Вікно інтерфейсу ігрового рушія Unreal Engine [8]

Основні можливості Unreal Engine:

– **візуальне програмування:** Unreal Engine використовує систему візуального програмування під назвою Blueprints, яка дозволяє розробникам створювати ігрову логіку без глибоких знань програмування C++;

– **2D та 3D розробка:** Unreal Engine підтримує розробку як 2D-, так і 3D-ігор. Він пропонує широкий спектр інструментів та функцій для створення персонажів, об'єктів, декорацій та інших ігрових елементів;

– **фізика:** Unreal Engine має вбудований фізичний рушій NVIDIA PhysX, який забезпечує реалістичне моделювання фізичних явищ;

– **штучний інтелект:** Unreal Engine підтримує розробку ігор з використанням штучного інтелекту (ШІ). Розробники можуть створювати NPC, які реагують на дії гравця та самостійно приймають рішення;

– **звук:** Unreal Engine має вбудований звуковий рушій, який дозволяє відтворювати звукові ефекти, музику та озвучку в грі;

– **мережеві можливості:** Unreal Engine підтримує розробку багатокористувацьких ігор. Розробники можуть створювати сервери та клієнти, які дозволяють гравцям взаємодіяти один з одним;

– **підтримка VR/AR:** Unreal Engine підтримує розробку ігор для віртуальної та доповненої реальності (VR/AR);

– **C++ програмування:** Unreal Engine також підтримує розробку ігор на C++ для досвідчених розробників, які хочуть мати більше контролю над кодом;

– **спільнота та ресурси:** Unreal Engine має велику та активну спільноту розробників. Доступно безліч навчальних посібників, документації та ресурсів, які допоможуть розпочати роботу з Unreal Engine.

1.4.3 Ігровий рушій Gobot

Godot Engine – це кросплатформний, безкоштовний ігровий рушій з відкритим вихідним кодом, який використовується для створення 2D- і 3D-ігор і застосунків (рис. 1.10). Відповідно до дозвільної ліцензії MIT розробники ігор на рушії Godot володіють усією грою до останнього рядка коду рушія, без будь-яких роялті або зобов'язань. Godot відомий своїм унікальним підходом до архітектури вузлів і сцен для представлення специфічних ігрових функцій.

Вперше розроблений аргентинською ігровою студією в 2001 році, Godot був випущений з відкритим вихідним кодом у 2014 році. Ігри, створені на Godot, як правило, мають простішу графіку порівняно з Unity і Unreal Engine, але інді-розробники працюють у цих рамках, створюючи широкий спектр ігор, серед яких найвідомішими є Ex Zodiac і Helms of Fury [17].

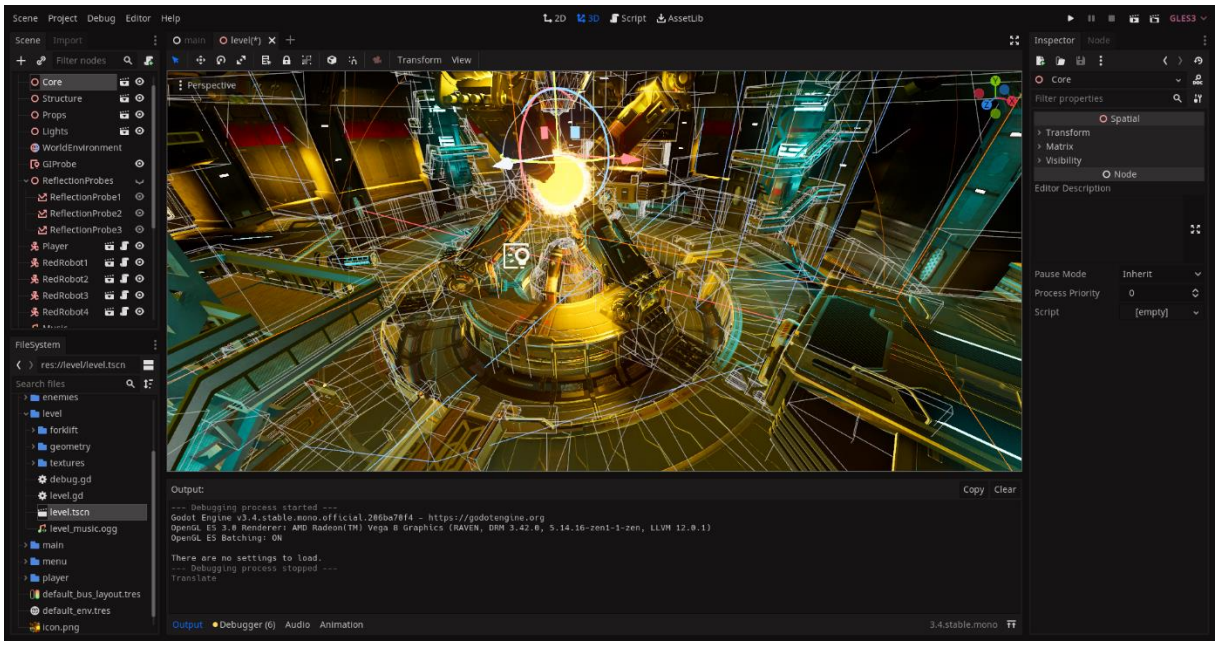

Рисунок 1.10 – Вікно інтерфейсу ігрового рушія Gobot

Основні можливості Godot:

– **візуальне програмування:** Godot використовує систему візуального програмування під назвою GDScript, яка дозволяє розробникам створювати ігрову логіку без глибоких знань програмування;

– **2D та 3D розробка:** Godot підтримує розробку як 2D-, так і 3D-ігор. Він пропонує широкий спектр інструментів та функцій для створення персонажів, об'єктів, декорацій та інших ігрових елементів;

– **фізика:** Godot має вбудований фізичний рушій, який дозволяє реалістично моделювати фізичні явища, такі як падіння предметів, зіткнення та рух транспортних засобів;

– **штучний інтелект:** Godot підтримує розробку ігор з використанням штучного інтелекту (ШІ). Розробники можуть створювати NPC, які реагують на дії гравця та самостійно приймають рішення;

– **звук:** Godot має вбудований звуковий рушій, який дозволяє відтворювати звукові ефекти, музику та озвучку в грі;

– **мережеві можливості:** Godot підтримує розробку багатокористувацьких ігор. Розробники можуть створювати сервери та клієнти, які дозволяють гравцям взаємодіяти один з одним;

– **підтримка VR/AR:** Godot підтримує розробку ігор для віртуальної та доповненої реальності (VR/AR);

– **спільнота та ресурси:** Godot має активну та зростаючу спільноту розробників. Доступно безліч навчальних посібників, документації та ресурсів, які допоможуть вам розпочати роботу з Godot.

Висновки до розділу 1

Ігрові застосунки мають велике значення в житті сучасної людини і великий вплив на різні сфери життя, включаючи економіку, культуру, науку. Для штучного інтелекту існує багато сфер використання – від генерації ігрових світів до «живого» спілкування з ігровими NPC.

Найбільш активну та зростаючу спільноту мають 3D-ігри у спортивному жанрі, як такі, що максимально реалістично можуть відтворити атмосферу та динаміку реального життя.

Найпопулярнішими ігровими рушіями, які використовується на теперішній час у сучасному GameDev, є Unity, Unreal Engine та Unity. Серед них платформа Unity відрізняється найбільшою універсальністю для створення ігор для ПК, консолей, мобільних пристроїв та браузерів у реальному часі.

Застосування ШІ обіцяє відкрити нові горизонти в плані масштабу, реалізму, інтерактивності створюваних ігор.

2 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ІГРОВОГО ЗАСТОСУНКУ

2.1 Мова програмування C# та платформа .NET

На сьогодні мова програмування C# – одна з найпотужніших, таких, що швидко розвиваються і користуються попитом в IT-галузі. Наразі на ній пишуть найрізноманітніші застосунки: від невеликих десктопних програм до великих веб-порталів і вебсервісів, які обслуговують щодня мільйони користувачів.

C# вже не молода мова і як і вся платформа .NET вже пройшла великий шлях. Перша версія мови вийшла разом із релізом Microsoft Visual Studio .NET у лютому 2002 року. Поточною версією мови є версія C# 12, яка вийшла 14 листопада 2023 року разом із релізом .NET 8.

C# є мовою з Сі-подібним синтаксисом і близька в цьому відношенні до C++ і Java. Тому, якщо ви знайомі з однією з цих мов, то опанувати C# буде легше.

C# є об'єктно-орієнтованою і в цьому плані багато перейняла у Java та C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дає змогу розв'язати завдання з побудови великих, але водночас гнучких, масштабованих і розширюваних застосунків. І C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональностей [9].

2.1.1 Роль платформи .NET

За висловлюванням Білла Гейтса платформа .NET – це найкраще, що створила компанія Microsoft. Можливо, він мав рацію. Фреймворк .NET представляє потужну платформу для створення застосунків. Можна виділити такі її основні риси:

– підтримка кількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд із C# це також VB.NET, C++, F#, а також різні діалекти інших мов,

прив'язані до .NET, наприклад, Delphi.NET. Під час компіляції код будь-якою з цих мов компілюється в збірку загальною мовою CIL (Common Intermediate Language) – свого роду асемблер платформи .NET. Тому за певних умов ми можемо зробити окремі модулі одного застосунку окремими мовами;

– кросплатформність. .NET є переносною платформою (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент – .NET 8 підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти застосунки мовою C# для найрізноманітніших платформ – Windows, MacOS, Linux, Android, iOS, Tizen;

– потужна бібліотека класів. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І який би застосунок не були наміри писати на C# – текстовий редактор, чат або складний вебсайт – так чи інакше буде задіяно бібліотеку класів .NET;

– різноманітність технологій. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти під час побудови тих чи інших застосунків. Наприклад, для роботи з базами даних у цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core. Для побудови графічних застосунків із багатим насиченим інтерфейсом – технологія WPF і WinUI, для створення простіших графічних застосунків – Windows Forms. Для розробки кросплатформних мобільних і десктопних застосунків – Xamarin/MAUI. Для створення вебсайтів і вебзастосунків – ASP.NET тощо;

– продуктивність. Згідно з низкою тестів вебзастосунки на .NET у низці категорій сильно випереджають вебзастосунки, побудовані за допомогою інших технологій. Застосунки на .NET в принципі відрізняються високою продуктивністю.

Окрім вищезазначених переваг, C# також має ряд перспективних напрямків розвитку, які роблять її ще більш привабливою мовою програмування для майбутнього:

– **хмарні технології:** C# широко використовується для розробки хмарних рішень на платформі Microsoft Azure. Зростання популярності хмарних сервісів відкриває нові можливості для C#-розробників;

– **штучний інтелект та машинне навчання:** C# підтримує інтеграцію з популярними фреймворками машинного навчання та штучного інтелекту, такими як TensorFlow і PyTorch. Це робить C# цінним інструментом для розробки інтелектуальних застосунків;

– **інтернет речей (IoT):** C# використовується для розробки програмного забезпечення для IoT-пристроїв. Зростання популярності IoT створює попит на C#-розробників з досвідом роботи в цій галузі;

– **ігри:** Unity, один з найпопулярніших ігрових движків, використовує C# як основну мову програмування. Це робить C# чудовим вибором для розробників ігор.

C# – це потужна, гнучка та перспективна мова програмування, яка користується великим попитом в IT-галузі. Її простота використання, широкі можливості та активна спільнота роблять її чудовим вибором як для початківців, так і для досвідчених розробників.

2.2 Середовище розробки Visual Studio

Visual Studio – це інтегроване середовище розроблення (англ. Integrated Development Environment, IDE), розроблене компанією Microsoft для розроблення застосунків для настільних комп'ютерів, графічних інтерфейсів користувача, консолей, вебзастосунків, мобільних застосунків, хмарних і вебсервісів тощо (рис. 2.1). За допомогою цього середовища розробки можна створювати як керований, так і нативний код. Воно використовує різні платформи для розробки програмного забезпечення Microsoft, такі як Windows Store, Microsoft Silverlight, Windows API

тощо. Це не мовна IDE, оскільки з її допомогою можна писати код на C#, C++, VB (Visual Basic), Python, JavaScript і багатьох інших мовах. Вона підтримує 36 різних мов програмування. Вона доступна як для Windows, так і для macOS [10].

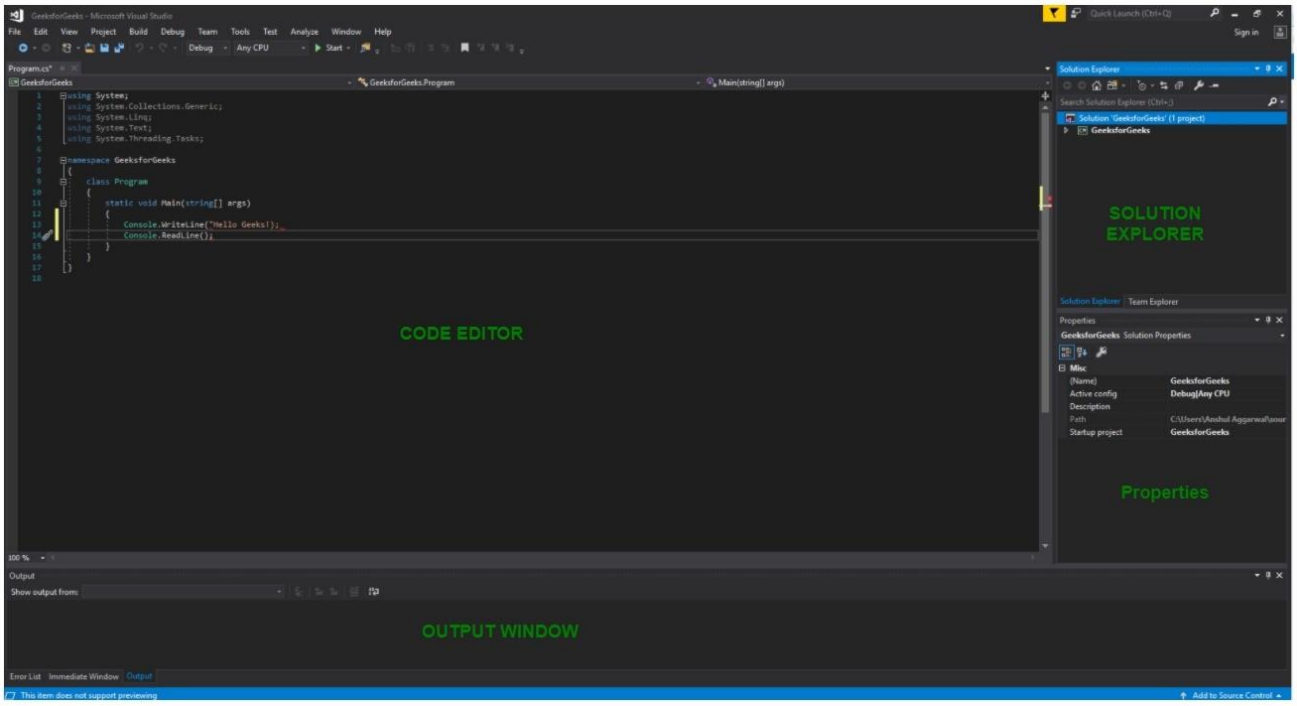

Рисунок 2.1 – Робоче вікно Visual Studio [10]

Visual Studio пропонує широкий спектр функцій, які роблять його потужним інструментом для розробників програмного забезпечення:

– **редактор коду:** Visual Studio має потужний редактор коду з підсвіткою синтаксису, автозаповненням коду, рефакторингом та іншими функціями, які допомагають розробникам писати код швидше та ефективніше;

– **відладчик:** Visual Studio має вбудований відладчик, який дозволяє розробникам покроково виконувати код, встановлювати точки зупинки, переглядати значення змінних та інше;

– **інструменти для тестування:** Visual Studio включає в себе інструменти для тестування, які дозволяють розробникам писати та виконувати тестові сценарії для своїх програм;

– **інструменти для дизайну:** Visual Studio має інструменти для дизайну, які дозволяють розробникам створювати графічні інтерфейси користувача, веб-сторінки та інші візуальні елементи;

– **інтеграція з системами керування версіями:** Visual Studio інтегрується з популярними системами керування версіями, такими як Git і Subversion, що дозволяє розробникам легко відстежувати зміни в коді та співпрацювати з іншими розробниками;

– **розширення:** Visual Studio підтримує розширення, які дозволяють розширювати його функціональність та додавати нові функції.

Visual Studio дозволяє створювати різні проєкти, такі як: настільні застосунки, графічні інтерфейси користувача, вебзастосунки, мобільні застосунки та хмарні і вебсервіси.

Visual Studio – це чудовий вибір для розробників програмного забезпечення, які шукають потужне та багатофункціональне середовище розробки. Його простота використання, широкий спектр функцій та активна спільнота роблять його популярним вибором як для початківців, так і для досвідчених розробників.

2.3 Ігровий рушій для розробки відеоігр Unity

Unity – це ігровий рушій для 2D- і 3D-ігор, який існує з 2005 року. Розроблений компанією Unity Technologies, він був створений для того, щоб надати більшій кількості розробників доступ до інструментів розробки ігор, що в ті часи було в новинку. За своє довге життя рушій сильно змінився і розширився, зумівши не відстати від новітніх практик і технологій.

І сьогодні основним завданням ігрового рушія є надання найнадійнішого набору інструментів для індустрії розроблення ігор, а також максимальне спрощення використання рушія розробниками будь-якого рівня підготовки (так, включно з початківцями-розробниками). Вони також розширили сферу свого

впливу на інші галузі, приділяючи велику увагу розробці 3D у реальному часі, що робить цей рушій одним із найпотужніших серед існуючих.

Варто зазначити, що Unity також пропонує сертифікацію з використання рушія в таких галузях, як розробка ігор або програмування. Це один із небагатьох визнаних в індустрії сертифікатів, доступних для розроблення ігор загалом, що вигідно вирізняє його з-поміж інших ігрових рушіїв [18].

Розглянемо ключові особливості ігрового рушія

2.3.1 Підтримка 3D- і 2D-графіки

Як уже йшлося на початку, Unity підтримує як 3D-, так і 2D-графіку, що дає змогу вам вільно обирати художній стиль для своїх проєктів. Кожен тип графіки поставляється з власним спеціалізованим набором інструментів (наприклад, вирізання спрайтових аркушів для 2D-графіки) і навіть має власні API сценаріїв, які можна використовувати для різних варіантів фізики, що підходять для кожного стилю.

3D-графіка також пропонує надзвичайно багатий набір інструментів, що дають змогу створювати користувацькі матеріали, будувати шейдери за допомогою Shader Graph, налаштовувати освітлення, використовувати ефекти постоброблення та багато іншого. Ви навіть можете генерувати 3D-територію або створювати 2D-тайлмапи прямо в рушії, так що для будь-якої графіки, яку ви використовуєте, є великий набір інструментів.

2.3.2 Проста для розуміння архітектура

Unity пропонує дуже прозорий метод створення архітектури гри. Кожен «рівень» в ігровому проєкті Unity ділиться на сцени, і кожна сцена містить усі ігрові об'єкти, потрібні гравцеві для проходження рівня.

Unity також пропонує можливість встановлювати відносини «батько-дитина» між об'єктами в ієрархії, що дає змогу легко додавати кілька об'єктів (наприклад, екіпіровку, зброю або колайдер для виявлення зіткнень) до одного батьківського

об'єкта персонажа гравця. Крім того, в Unity є інструмент «Інспектор», який дає швидкий доступ до всіх властивостей об'єкта, а отже, ви можете швидко змінювати щось на льоту, не занурюючись у код.

2.3.3 Unity Scripting API

Unity постачається з потужним скриптовим API, який пропонує швидкий доступ до найпоширеніших функцій. Сюди входять як загальні ігрові функції, так і специфічні виклики API, що дають змогу отримати доступ до конкретних функцій і нюансів рушія. Можна стверджувати, що цей API – велика частина сутності Unity, оскільки без нього рушій – ніщо.

Наприклад, хоча можна налаштовувати елементи користувацького інтерфейсу з самого рушія, такі як колір тексту, Scripting API також розкриває ці елементи, щоб можна було налаштовувати їх і через код. Це стосується всього, що доступно в інспекторі Unity, включаючи положення, обертання, матеріали, відтворення звуку та багато іншого. Крім того, тут є багато документації, яка допоможе розібратися з цим.

2.3.4 Підтримка кросплатформних збірок

Unity – це потужний ігровий двигун, який славиться своєю гнучкістю та широкими можливостями кросплатформної розробки. Unity дає змогу створювати ігри, які без проблем працюватимуть на безлічі пристроїв та операційних систем, заощаджуючи час та ресурси.

Unity підтримує широкий спектр платформ: Android, iOS, Windows, macOS, Linux, PlayStation 4, Xbox One, Nintendo Switch, HTML5, Oculus Rift, HTC Vive, Google Cardboard, tvOS, watchOS, PlayStation Vita, Xbox 360, Wii U.

2.3.5 Можливості віртуальної та доповненої реальності

Для VR існує безліч пакетів, що підтримують майже всі доступні VR-гарнітури, які постійно оновлюються і підтримують гнучкість цієї мінливої технології.

AR теж не залишився осторонь: є безліч пакетів для ARCore і ARKit. Unity також пропонує AR Foundation, який Unity створила для того, щоб розробники Unity могли одночасно створювати AR-застосунки для Android і iOS, позбавляючись необхідності створювати окремі проєкти.

Крім того, в Unity з'явився інструментарій XR Interaction Toolkit, який ще більше спрощує розробку VR- та AR-ігор. Отже, достатньо сказати, що Unity є одним із найбільших прихильників технологій XR.

2.3.6 Великий магазин асетів

Unity Aset store надає безліч платних і безкоштовних ресурсів, які можна використовувати для будь-якого ігрового проєкту. Це також відіграє ключову роль у розумінні того, що таке Unity, оскільки в деяких випадках люди можуть мати на увазі тільки магазин активів.

Хоча Unity розробляє деякі з них, багато з них також створюються спільнотою, а отже, магазин має величезний вибір. Крім того, Unity дає змогу легко додавати активи до колекції та встановлювати їх у проєкт за допомогою менеджера пакетів, що означає, що не потрібно возитися з файлами вручну.

2.3.7 Параметри конвеєра рендерингу

Виведення графіки на екран – непросте завдання для комп'ютера, і те, як він це робить, може суттєво вплинути на продуктивність розроблювальних ігор. Саме тому Unity запропонувала кілька вбудованих варіантів конвеєрів рендерингу, які можна використовувати для переведення гри зі сцени на екран. Це дає змогу розробникам обирати конвеєр візуалізації, який найкраще підходить для їхніх проєктів та їхніх графічних потреб.

Крім того, Unity пропонує API Scriptable Render Pipeline, який дає змогу розробникам створювати власні конвеєри, якщо вони цього хочуть. Ці конвеєри також є основною частиною розуміння того, що таке Unity, оскільки вони визначають багато чого з того, як використовується Unity.

2.3.8 Інструменти анімації

Unity пропонує потужний набір інструментів для анімації, які працюють як з 3D-, так і з 2D-графікою. Хоча є можливість імпортувати анімацію з іншої програми, наприклад Blender або Autodesk Maya, Unity пропонує можливість анімувати проєкти прямо в самому рушії (рис. 2.2). Це включає в себе регулювання положення і обертання всього об'єкта, а також фізичне маніпулювання кістками 3D-моделі. Unity навіть пропонує можливість додавати кісткові ригіди до 2D-зображень.

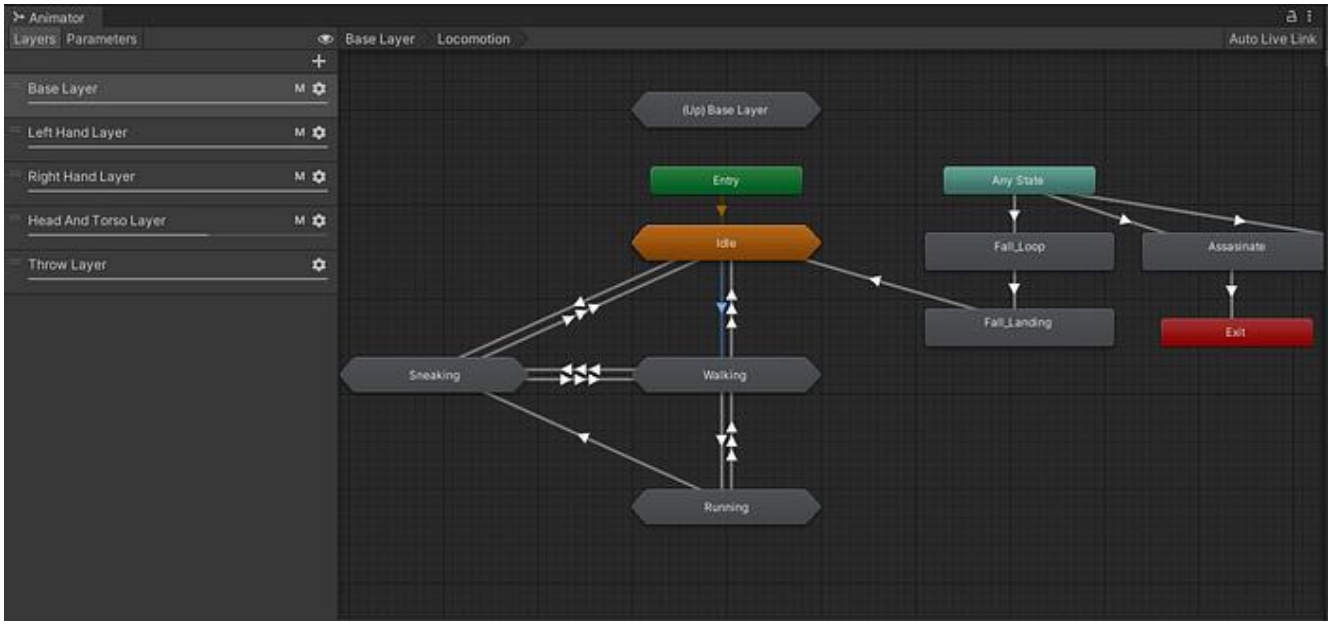

Рисунок 2.2 – Вікно аніматора Unity [11]

Усі ці функції, зрозуміло, доступні і через API сценаріїв, що дає безпрецедентний контроль над тим, як працює анімація.

Крім того, система Animator дає змогу легко створити машину станів анімації. Це означає, що можна не тільки відтворювати анімації залежно від того, що робить гравець (наприклад, стрибає), а й переходити від однієї анімації до іншої відповідним чином і плавно. Крім того, оскільки Animator представлений у стилі наочних графіків, легко зрозуміти, як усе взаємопов'язано.

2.3.9 Інструменти аналітики

У міру набуття навичок розробника ігор стає дедалі необхіднішим мати у своєму розпорядженні різні інструменти аналітики. Unity пропонує кілька з них, включно з інструментами для відстеження проблем із продуктивністю та інструментами для простого спостереження за тим, як гравці взаємодіють із ігровим проєктом.

Крім того, Unity пропонує безліч способів поліпшити налагодження за допомогою цих інструментів, забезпечуючи надійний спосіб зрозуміти кожен аспект створюваної гри.

2.4 Плагін ML-Agents Toolkit

Unity Machine Learning Agents Toolkit (ML-Agents Toolkit) – це проєкт з відкритим вихідним кодом, який дозволяє використовувати ігри та симулятори як середовище для навчання інтелектуальних агентів. Ці агенти можуть навчатися за допомогою різних методів машинного навчання, таких як навчання з підкріпленням, імітаційне навчання, нейроеволюція та інші методи, використовуючи зручний у користуванні Python API. Інструментарій надає реалізацію найсучасніших алгоритмів на основі PyTorch, що дозволяє розробникам ігор та ентузіастам легко навчати інтелектуальних агентів для 2D-, 3D- та VR/AR-ігор [16].

Навчені агенти можуть бути застосовані для різних завдань, включаючи керування поведінкою NPC у різних умовах, таких як мультиагентні та змагальні сценарії. Вони також можуть використовуватися для автоматичного тестування

ігрових збірок та оцінки різних рішень щодо дизайну гри перед її випуском. Інструментарій ML-Agents Toolkit є корисним як для розробників ігор, так і для дослідників штучного інтелекту, оскільки він надає центральну платформу, де досягнення у галузі ШІ можуть бути перевірені в багатому середовищі Unity та зроблені доступними для широкої спільноти дослідників і розробників. Спрощена блох–схема, яка ілюструє процес навчання ML–Agents (рис. 2.3) [19]:

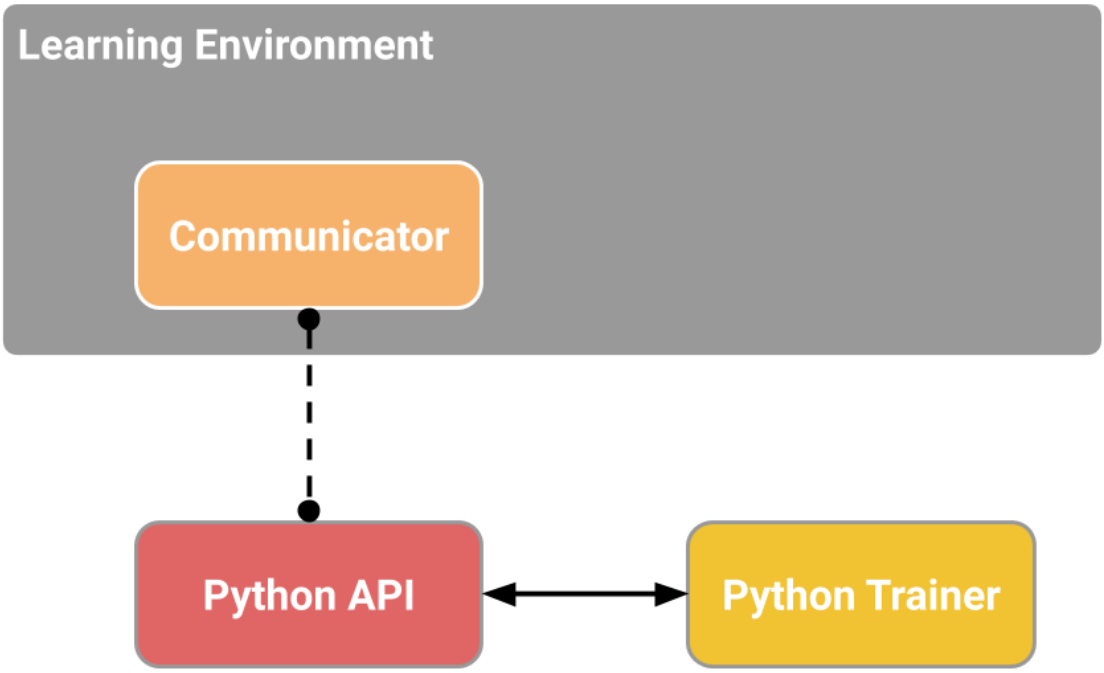

Рисунок 2.3 – Спрощена блок-схема ML-агентів [16]

Окрім цього, ML-Agents Toolkit пропонує велику кількість довідкових сторінок, які містять огляди та корисні ресурси щодо використання рушія Unity, принципів машинного навчання та роботи з PyTorch. Ці ресурси допомагають користувачам швидко орієнтуватися у можливостях інструментарію та ефективно використовувати його для своїх потреб, сприяючи розвитку інновацій у галузі штучного інтелекту та ігрової індустрії.

Висновки до розділу 2

Використання мови програмування C# у поєднанні з платформою .NET забезпечує потужний і гнучкий інструментарій для розробки різноманітних ігрових механік та функціональностей. Ця мова програмування, завдяки своїй простоті і водночас великій функціональності, є ідеальним вибором для розробників, що працюють у середовищі Unity.

Середовище розробки Visual Studio, з його багатим набором інструментів для написання, налагодження та тестування коду, є незамінним помічником для розробників. Воно забезпечує зручний інтерфейс, підтримку різних мов програмування та інтеграцію з іншими інструментами, що значно спрощує процес розробки і підвищує продуктивність.

Ігровий рушій Unity, завдяки своїм широким можливостям та підтримці різноманітних платформ, є провідним інструментом для створення відеоігор. Його зручний інтерфейс, багатий набір функцій і підтримка кросплатформності дозволяють розробникам створювати високоякісні ігри з мінімальними затратами часу і ресурсів.

Додатково плагін ML-Agents Toolkit для Unity відкриває нові горизонти у розвитку штучного інтелекту в іграх. Використання цього інструментарію дозволяє легко інтегрувати передові методи машинного навчання в ігровий процес, що сприяє створенню більш реалістичних ігрових персонажів і сценаріїв. Це не тільки покращує ігровий досвід, але й дає змогу використовувати ігри як середовище для тестування і вдосконалення алгоритмів штучного інтелекту.

Отже, комбінація мови програмування C# і платформи .NET, середовища розробки Visual Studio, ігрового рушія Unity та плагіна ML-Agents Toolkit створює потужний набір інструментів, які дозволяють ефективно розробляти, тестувати та вдосконалювати відеоігри, інтегруючи найсучасніші досягнення в галузі програмування і штучного інтелекту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ

3.1 Створення оточення для навчання ШІ

Для того щоб штучний інтелект міг навчатися, йому треба створити оточення, в якому він буде це робити. Оточення має бути в міру простим і мати все необхідне для якісного навчання. Також для поставленої задачі навчання треба визначити мету, яку буде присліджувати ШІ та визначити винагороди за досягання мети або за зроблені помилки.

Для поставленої задачі, створення спортивної гри футбол, ШІ інтелекту потрібно створити сцену (рис. 3.1), що буде містити ворота опонента, свої ворота, м’яч та стіни, які будуть обмежувати гравця та нейронну мережу в пересуванні.

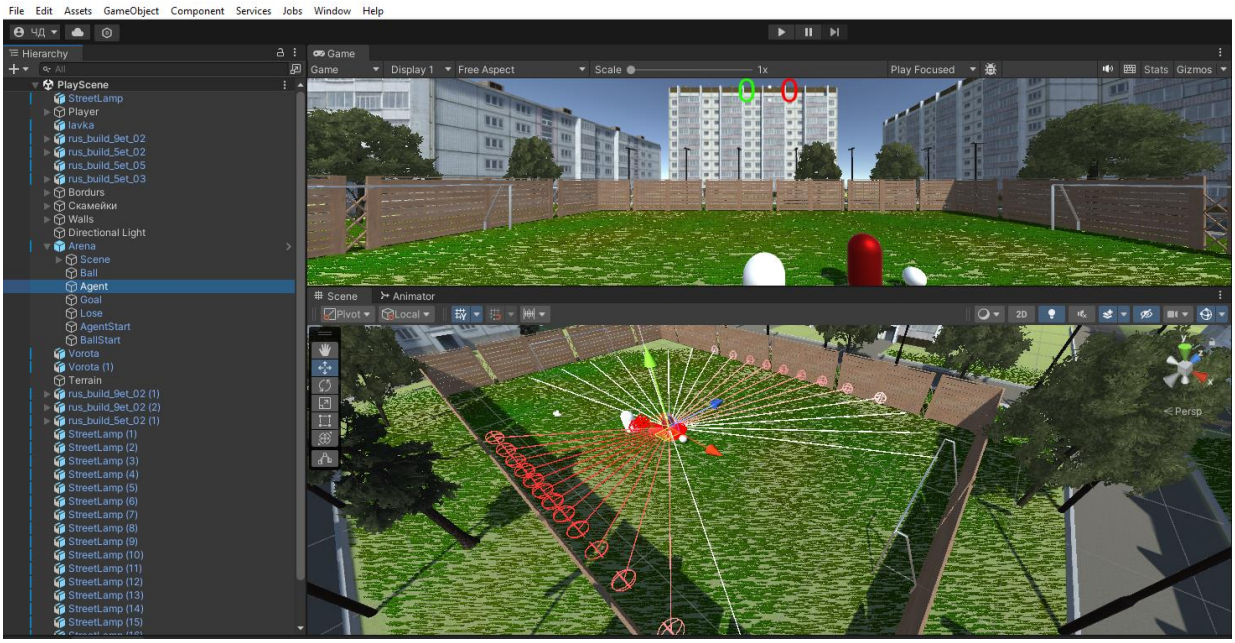

Рисунок 3.1 – Робоче вікно Unity зі створеною сценою

3.2 Створення логіки агента для навчання НМ

Для навчання агента потрібно додати до нього 3 основні компоненти з бібліотеки ML-Agents: *Decision Requester, Ray Perception Sensor* та *Behavior Parameters*.

Decision Requester – це компонент, який робить запит на нові дії у моделі ML-Agents, через задані інтервали часу та інформацію стосовно оточення. Іншими

словами, через кожний заданий проміжок часу, агент робить запит «що йому робити?».

Ray Perception Sensor – це компонент, який дозволяє агенту «бачити» оточення за допомогою промінів (*raycasts*). Він посилає проміні в різні направлення і збирає інформацію о перетині з іншими об’єктами. Ця інформація використовується агентом як частина спостережень для прийняття рішень.

Behavior Parameters – компонент, який визначає основні параметри поведінки агента, включаючи його спостереження, дії та тип моделі. Цей компонент необхідний для налаштування і навчання моделі.

ML-Agents дозволяє створювати та навчати агентів з мінімальним написанням коду, але все ж таки в деяких випадках написання логіки через скрипти необхідне. З початку написання застосунку було прийнято рішення прописувати логіку через скрипти на мові C#, але в деяких випадках це викликає помилки. Наприклад, спостереження агентом навколишнього оточення можна зробити за допомогою перевизначення методу *collectObservations* (рис. 3.2):

```
/// [GameObject]: https://docs.unity3d.com/Manual/GameObjects.html
/// [Observations and Sensors]: https://github.com/Unity-Technologies/ml-agents/blob/release_17_docs/docs/Learning-Environment-Design-Agents.md#observations-and-sensors
/// </remarks>
public virtual void CollectObservations(VectorSensor sensor)
{
}
```

Рисунок 3.2 – Метод класу *Agent collectObservations*

При перевизначені цього метода, ми можемо задати об’єкти за якими повинен спостерігати агент, дані передаються у вигляді векторів.

При написанні визначені цього метода і подальшому тестуванні було виявлено одну особливість: якщо задати якісь об’єкти, навчити модель, а потім додати нові об’єкти, – це викличе помилки. Тож додавання можливості агенту «бачити» було прийнято зробити через компонент *Ray Perception Sensor.*

Для навчання агента потрібно додати винагороди за потрібні дії (рис. 3.3). Наприклад, за забивання м’яча в ворота опонента агент отримує винагороду в 5*f*, а за забивання в свої ворота –5*f*, Це додає можливості агенту розуміти за які дії він

отримує винагороду і на основі цих даних навчатися робити «правильні» дії все краще і краще.

```
Ссылок: 1
public void Lose()
{
    SetReward(-5f);
    EndEpisode();
    playerScoreNumber += 1;
}

Ссылок: 1
public void ScoreGoal()
{
    SetReward(5f);
    EndEpisode();
    enemyScoreNumber += 1;
}
```

Рисунок 3.3 – Визначення винагороди за забивання м’яча у ворота опонента

Також для пришвидшення якості і часу навчання було додану систему винагород за бездіяльність та наближення до м’ячу (рис. 3.4).

```
Ссылок: 0
public override void OnActionReceived(ActionBuffers actions)
{
    float moveHorizontal = actions.ContinuousActions[0];
    float moveVertical = actions.ContinuousActions[1];

    Vector3 movement = new Vector3(moveHorizontal, 0f, moveVertical).normalized;

    rb.velocity = movement * speed;

    float distanceToBall = Vector3.Distance(transform.position, ball.position);
    if (distanceToBall < 1f && ball.GetComponent<Rigidbody>().velocity.magnitude < 0.1f)
    {
        AddReward(rewardForBallClose);
    }

    if (Vector3.Distance(transform.position, lastPosition) < 0.01f)
    {
        AddReward(-0.01f);
    }
    float distanceToLastPosition = Vector3.Distance(transform.position, lastPosition);
    if (distanceToLastPosition < 0.01f)
    {
        AddReward(-0.001f);
    }
    float distanceBallToGoal = Vector3.Distance(ball.position, goal.position);
    float ballMovement = Vector3.Distance(ball.position, lastBallPosition);



    lastPosition = transform.position;
    lastBallPosition = ball.position;
}
```

Рисунок 3.4 – Визначення винагороди за бездіяльність та наближення до м’ячу

3.3 Налаштування параметрів навчання

Коли сцена створена і налаштовані всі її параметри, потрібно налаштувати параметри навчання і нейронної мережі. В ML-agents конфігурація навчання прописана в файлі *«назва_файлу».yaml* (рис. 3.5).

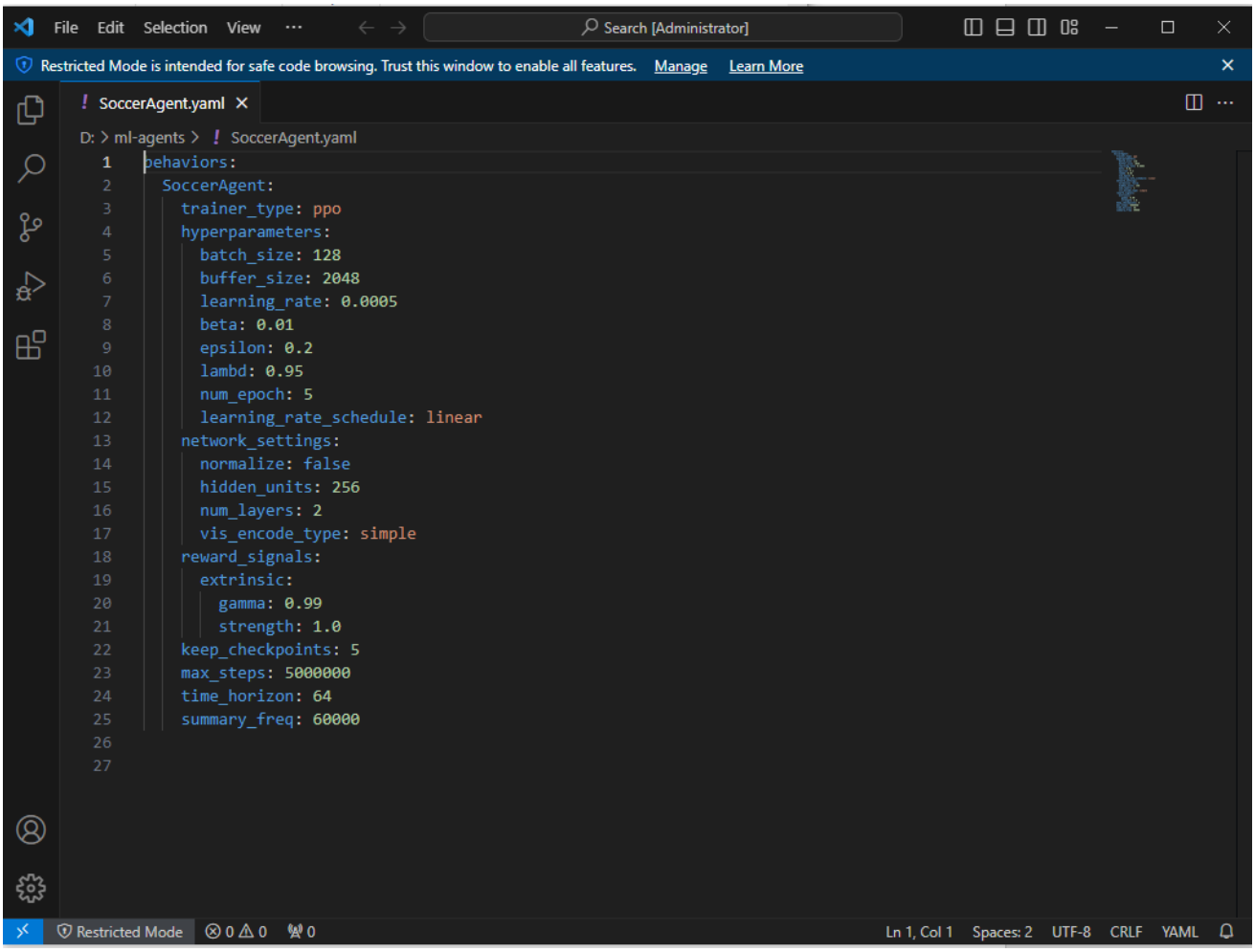

Рисунок 3.5 – Файл конфігурації навчання ШІ

Конфігураційний файл для навчання агента в Unity ML-Agents містить різні змінні, які визначають параметри навчання та налаштування мережі. Ось детальний опис усіх змінних у наведеному файлі:

– trainer_type: ppo – тип тренера (алгоритму навчання), який використовується для навчання агента. У даному випадку це Proximal Policy Optimization (PPO), популярний алгоритм навчання з підкріпленням;

– batch_size: розмір міні-вибірки (batch) для кожного оновлення градієнта під час навчання. У цьому випадку 128;

– buffer_size: розмір буфера досвіду, який використовується для збору даних перед оновленням моделі. У цьому випадку 2048;

– learning_rate: швидкість навчання, яка визначає, як сильно оновлюються параметри моделі на кожному кроці навчання. Тут значення 0,0005;

– beta: параметр регуляризації ентропії, який контролює дослідження простору дій. Значення 0,01;

– epsilon: параметр, що визначає радіус «проксимальності» для алгоритму PPO. Значення 0,2;

– lambd: параметр гейтування (lambda) для обчислення обумовлених градієнтів (GAE). Значення 0,95;

– num_epoch: кількість епох, протягом яких проходять дані з буфера під час кожного оновлення політики. У цьому випадку 5;

– learning_rate_schedule: графік зміни швидкості навчання. Вказано «linear», що означає лінійне зменшення швидкості навчання;

– normalize: визначає, чи нормалізувати спостереження агента. У цьому випадку false (не нормалізувати);

– hidden_units: кількість прихованих нейронів у кожному шарі нейронної мережі. Тут значення 256;

– num_layers: кількість прихованих шарів у нейронній мережі. У цьому випадку 2;

– vis_encode_type: тип кодувальника для візуальних спостережень. Значення «simple» вказує на простий кодувальник;

– extrinsic: зовнішні сигнали винагороди;

– gamma: коефіцієнт зниження (discount factor) для майбутніх винагород. Значення 0,99;

– strength: сила зовнішнього сигналу винагороди. Значення 1,0;

– keep_checkpoints: кількість контрольних точок (checkpoint), які зберігаються під час навчання. У цьому випадку 5;

– max_steps: максимальна кількість кроків навчання. У цьому випадку 5’000’000;

– time_horizon: горизонт часу для навчання. Визначає кількість кроків, після яких буфер скидається. Значення 64;

– summary_freq: частота запису зведених даних (summary) для TensorBoard. У цьому випадку 60’000 кроків.

Цей файл конфігурації визначає, як агент SoccerAgent буде навчатися, включаючи параметри навчання, структуру нейронної мережі та налаштування винагород. Важливо зазначити, що немає готових наборів параметрів для навчання власної нейронної мережі і всі значення в конфігураційному файлі змінюються методом проб і помилок, для покращення процесу навчання.

Алгоритм навчання

Для навчання НМ було обрано алгоритм PPO – це алгоритм в області навчання з підкріпленням, який тренує функцію прийняття рішень комп’ютерного агента для виконання складних завдань [15]. Алгоритм PPO був розроблений Джоном Шульманом в 2017 році і став алгоритмом навчання з підкріпленням американської компанії OpenAI.

Цей алгоритм навчання було обрано по наступним критеріям:

– стабільність і надійність: PPO забезпечує стабільне та надійне навчання, знижуючи ризик розриву градієнта, що часто зустрічається в інших методах навчання з підкріпленням;

– ефективність: PPO дозволяє ефективно використовувати обчислювальні ресурси, знижуючи необхідність у великій кількості взаємодій з середовищем;

– адаптивність: алгоритм добре працює у різних середовищах і з різними типами задач, що робить його універсальним інструментом для навчання агентів;

– простота реалізації: відносно проста реалізація PPO робить його доступним для впровадження та налаштування навіть для складних задач, що вимагають специфічних налаштувань;

– гнучкість: PPO дозволяє легко налаштовувати гіперпараметри, що забезпечує можливість оптимізації процесу навчання для конкретних умов завдання.

Політика, в термінології навчання з підкріпленням, - це відображення з простору дій у простір станів. Її можна уявити як інструкції для агента, які вказують, які дії він повинен виконувати залежно від того, в якому стані середовища він зараз перебуває (рис. 3.6).

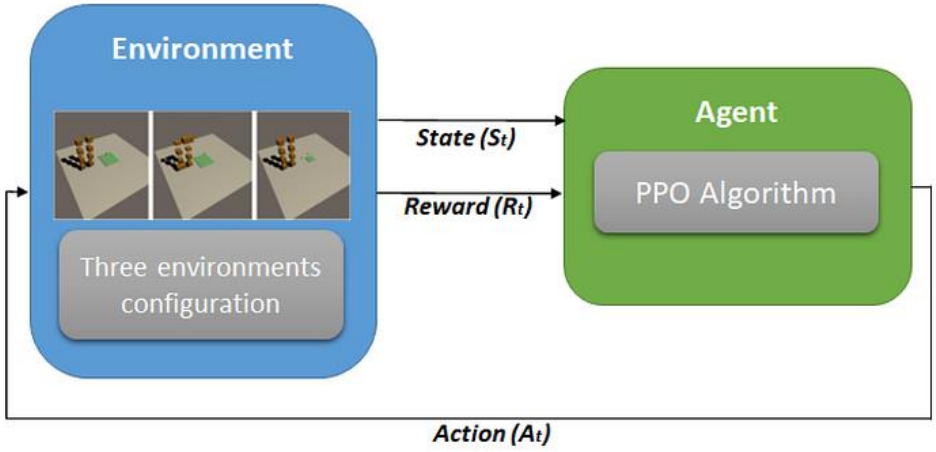

Рисунок 3.6 – Діаграма конфігурації трьох середовищ алгоритму PPO [26]

3.3 Запуск процесу навчання

Коли всі параметри налаштовані, можна запускати процес навчання. Для цього у терміналі потрібно перейти у папку зі встановленим ML-Agents та запустити процес навчання(рис. 3.7).

Рисунок 3.7 – Запуск процесу навчання у терміналі Anaconda Prompt

Відстеження процесу навчання в ML-Agents відбувається за допомогою робочого вікна Unity або за допомогою графіків навчання (рис. 3.8).

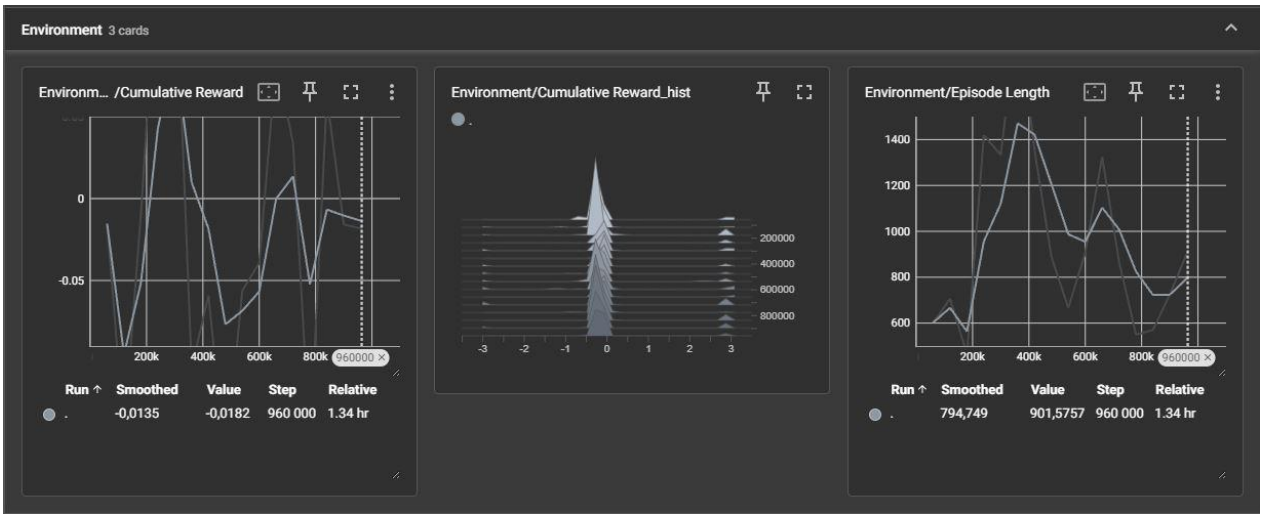

Рисунок 3.8 – Графіки результатів першого циклу навчання НМ

В основному для відстеження якості навчання на рис. 3.8 використовуються два графіки: Cumulative Reward та Episode Length.

Графік Cumulative Reward показує накоплення винагороди агентом за кожен епізод навчання. З цього графіку видно, що винагорода зростає, що свідчить проте, що агент за 2 млн епізодів вчиться краще заробляти винагороду.

Графік Episode Length відображає кількість часу, на досягнення мети, тобто на завершення епізоду. Цей графік свідчить проте, що агенту з часом потрібно все більше часу на завершення епізоду.

Аналіз поведінки агента на основі отриманого графіку показує, що НМ зменшує час на закінчення епізоду, а накопичення винагороди з кожним етапом зростає. Це свідчить про те, що НМ навчається у правильному напрямку. Другий етап навчання відображено на рис. 3.9.

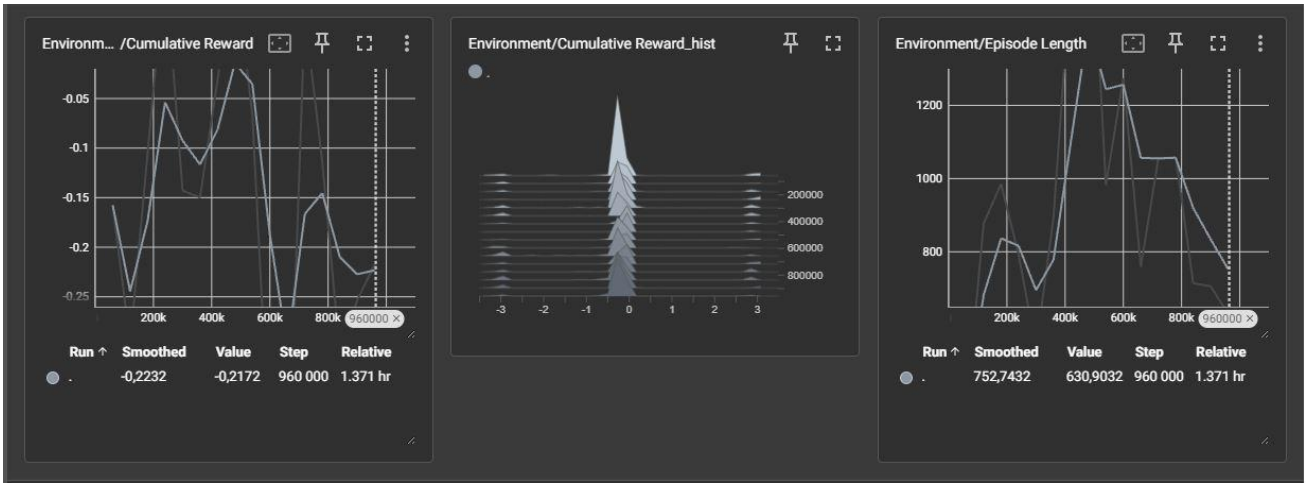

Рисунок 3.9 – Графіки результатів другого циклу навчання НМ

Аналіз цього графіку показує, що є тенденція зменшення накопичення винагороди НМ з кожним епізодом, а кількість часу, затраченого на завершення епізоду, зменшується. У порівнянні з першим циклом, на другому НМ зробила більше дій, 134 проти 91, що може свідчити про те що, ШІ краще став розуміти свої цілі, але навчання потрібно продовжувати, доки агент не навчиться виконувати потрібні від нього дії.

3.4 Створення візуальної частини для сцени

Деякі моделі для наповнення візуальної частини гри було розроблено власноруч інструментом 3D-моделювання Blender. Однією з основних моделей для гри є ворота, в які відбувається забивання гола (рис. 3.10):

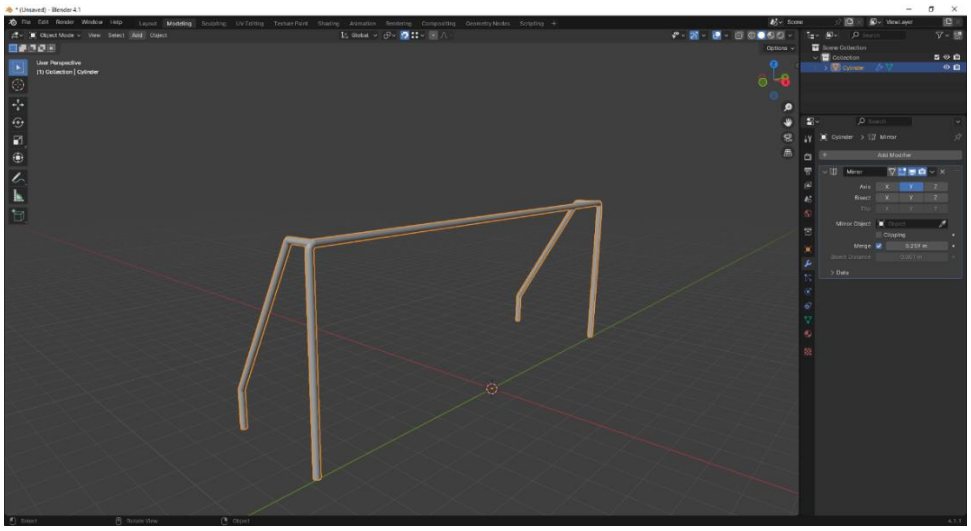

Рисунок 3.10 – Моделювання воріт в застосунку Blender

Більшість моделей було взято з відкритого доступу в інтернеті. Зовнішній вигляд рівня було рішено зробити у вигляді типового двора кінця 20-го століття; такий вигляд повинен викликати у гравця відчуття ностальгії і приємні спогади з дитинства (рис. 3.11).

Рисунок 3.11 – Зовнішній вигляд рівня в створеній грі

3.5 Створення головного меню гри

Меню було прийнято рішення зробити простим з декількома кнопками (початок гри, налаштування та вихід з гри) та сценою з гри на задньому фоні (рис. 3.12).

Рисунок 3.12 – Зовнішній вигляд головного меню

Висновки до розділу 3

У цьому розділі було детально описано основні етапи розробки ігрового застосунку, включаючи створення середовища для навчання, налаштування параметрів для навчання нейронної мережі, розробку логіки агента та створення графічної частини проєкту.

Створення оточення для навчання: розроблено віртуальне середовище, в якому агент має взаємодіяти з м’ячем та іншими елементами гри. Налаштовано стартові позиції для агента та м'яча, а також визначено умови завершення епізодів.

Налаштування параметрів для навчання нейронної мережі: обрано алгоритм навчання з підкріпленням Proximal Policy Optimization (PPO) через його

ефективність та стабільність. Встановлено ключові параметри для тренування, такі як розмір пакету, швидкість навчання, кількість прихованих шарів та інші гіперпараметри.

Створення логіки агента: розроблено скрипт для управління агентом, включаючи методи Initialize, CollectObservations, OnActionReceived та інші, які визначають поведінку агента під час навчання. Налаштовано систему винагород та штрафів для оптимізації навчання агента.

Створення графічної частини проєкту: розроблено візуальні елементи гри, включаючи моделі агента, м’яча та інших об’єктів. Використано Unity для інтеграції графіки з логікою гри, забезпечуючи візуальний зворотний зв'язок під час навчання та тестування агента.

В результаті виконаної роботи було створено функціональний ігровий застосунок, в якому агент, керований навченою нейронною мережею, може грати в спортивну гру футбол зі штучним інтелектом. Це досягнення демонструє ефективність підходу навчання з підкріпленням у створенні складних ігрових агентів, здатних до автономного прийняття рішень та взаємодії в динамічних середовищах.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра було створено ігровий застосунок з використанням методів штучного інтелекту на ігровому рушії 3D Unity.

Для досягання поставленої мети було вирішено наступні задачі:

– проведено детальний аналіз існуючих ігор у жанрі спортивних симуляторів, що дозволило визначити ключові механіки та особливості, які повинні бути реалізовані в розробленому застосунку. Зібрано та систематизовано інформацію про найкращі практики у сфері ігрового дизайну для створення більш реалістичного та цікавого геймплею;

– створено та налаштовано основні ігрові механіки, такі як управління персонажем, фізика м'яча, правила гри та система рахунку. Забезпечено логіку взаємодії між гравцем та штучним інтелектом, що керує суперником;

– зроблено аналіз створення оточення для навчання НМ. Визначено оптимальні параметри середовища, що сприяють ефективному та швидкому навчанню агента;

– проведено навчання НМ у створеному оточенні. Використано алгоритм навчання з підкріпленням Proximal Policy Optimization (PPO) для тренування агента. Проведено серію навчальних епізодів, під час яких агент навчався досягати поставлених цілей та взаємодіяти з навколишнім середовищем;

– створено візуальну частину застосунку: розроблено та інтегровано візуальні моделі ігрових об'єктів і оточення. Забезпечено відповідний рівень деталізації та естетичної привабливості візуальних елементів, що підвищує загальну якість гри;

– розроблено графічний інтерфейс гри – простий, зрозумілий для користувача.

Завдяки виконанню цих завдань, було створено функціональний ігровий застосунок, що демонструє можливості штучного інтелекту в контексті спортивних

симуляторів. Цей проєкт підтверджує ефективність використання методів навчання з підкріпленням для розробки складних ігрових агентів, здатних автономно приймати рішення та взаємодіяти з динамічним ігровим середовищем.

Практичне значення роботи полягає в створення більш реалістичних ігрових персонажів і сценаріїв для використання ігри як середовища для тестування і вдосконалення алгоритмів штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Game industry growing faster than expected, up 10.7% to $116 billion 2017. URL: https://venturebeat.com/esports/newzoo-game-industry-growing-faster-than-expected-up-10-7-to-116-billion-2017/ (Last accessed: 07.05.2024).

2. The impact of video games on the economy. URL: https://chiefcxofficer.com/the-impact-of-video-games-on-the-economy (Last accessed: 07.05.2024).

3. The Future of gaming is AI: How artificial intelligence is changing everything. URL: https://elearningindustry.com/the-future-of-gaming-is-ai-how-artificial-intelligence-is-changing-everything (Last accessed: 07.05.2024).

4. Strobach T., Schubert T. Video game training and effects on executive functions. *In book: Cognitive Training, An Overview of Features and Applications*. Springer Nature, 2020. P. 229–241. DOI: 10.1007/978-3-030-39292-5_16.

5. Hoseini F. S., Khodadadi M., Khodadadi A. The effect of 2D and 3D action video game interventions on executive functions in male students. *Simulation & Gaming.* June 2022. Vol. 53(1). P. 1–18. DOI: 10.1177/10468781221110577.

6. Gaming Engines. URL: https://www.arm.com/glossary/gaming-engines (Last accessed: 08.05.2024).

7. Huang V., Young M., Fiocco A. J. The association between video game play and cognitive function: Does gaming platform matter? *Cyberpsychology, Behavior, and Social Networking.* Oct. 20, 2017. Vol. 20, No. 11. DOI: 10.1089/cyber.2017.0241.

8. How to Learn Unreal Engine (2024). URL: https://academyofanimatedart.com/learn-unreal-engine/ (Last accessed: 08.05.2024).

9. Мова C# та платформа .NET URL: https://metanit.com/sharp/tutorial/1.1.php (дата звернення: 09.05.2024).

10. Introduction to Visual Studio. Publ. Sep. 22, 2023. URL: https://www.geeksforgeeks.org/introduction-to-visual-studio/ (Last accessed: 08.05.2024).

11. Introduction to Unity’s Animator. Publ. Sep. 22, 2023. URL: https://medium.com/geekculture/introduction-to-unitys-animator-baac57eccc8a (Last accessed: 08.05.2024).

12. Паласиос Х. UNITY 5.X. Програмування штучного інтелекту в іграх. Print2print, 2017. 272 с.

13. Камінський Б. Unity випустив набір інструментів штучного інтелекту Muse для розробників відеоігор. Опубл. 17 листопада 2023 р. URL: https://gagadget.com/uk/ai/356120-unity-vipustiv-nabir-instrumentiv-shtuchnogo-intelektu-muse-dlia-rozrobnikiv-videoigor/ (дата звернення: 09.05.2024).

14. Yannakakis G., Togelius J. Artificial Intelligence and Games. Springer, 2018. URL: http://gameaibook.org (Last accessed: 09.05.2024).

15. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms. *arXiv.org/abs/1707.06347v2 [cs.LG].* DOI: 10.48550/arXiv.1707.06347 (Last accessed: 09.06.2024).

16. ML-Agents Toolkit Overview. URL: https://unity-technologies.github.io/ml-agents/ML-Agents-Overview/ (Last accessed: 08.06.2024).

17. Learn the 5 most important features of Godot engine with us! URL: https://www.logiscool.com/en/blog/coding-tech/learn-the-5-most-important-features-of-godot-engine-with-us (Last accesed: 08.06.2024).

18. What Makes It the Best Game Engine? URL: https://kevurugames.com/blog/unity-what-makes-it-the-best-game-engine/#:~:text=Simple%20and%20Intuitive%20Interface,are%20created%20by%20combining%20nodes (Last accessed: 08.06.2024).

19. How do Unity ML-Agents work? URL: https://huggingface.co/learn/deep-rl-course/unit5/how-mlagents-works (Last accesed: 08.06.2024).

20. Evolution of npcs in gaming: ai impact URL: https://whimsygames.co/blog/evolution-of-npcs-in-gaming-ai-impact/ (Last accessed: 08.06.2024).

21. The algorithms of No Man’s Sky. URL: https://www.rambus.com/blogs/the-algorithms-of-no-mans-sky-2/ (Last accessed: 09.06.2024).

22. The best football games dominating the digital dugout. URL https://www.gamesradar.com/best-football-games/ (Last accesed: 09.06.2024).

23. What is AI for Gaming Industry. URL: https://www.engati.com/blog/ai-for-gaming (Last accessed: 09.06.2024).

24. How To Choose The Best Gaming Engine For Your Game? URL: https://pinglestudio.com/blog/co-development/how-to-choose-the-best-gaming-engine-for-your-game (Last accessed: 09.06.2024).

25. Introducing Unity Machine Learning Agents Toolkit. URL: https://blog.unity.com/engine-platform/introducing-unity-machine-learning-agents (Last accessed: 09.06.2024).

26. PPO Algorithm. URL: https://medium.com/@danushidk507/ppo-algorithm-3b33195de14a (Last accessed: 09.06.2024).

ДОДАТОК А

Код агента ML–Agents на мові C#

SoccerAgent.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Sensors;
using Unity.MLAgents.Actuators;

public class SoccerAgent : Agent
{
    public Transform ball;
    public Transform goal;
    public Transform lose;
    public Transform spawn;
    Rigidbody rb;
    public float speed = 20f;
    public int enemyScoreNumber = 0;
    public int playerScoreNumber = 0;
    private Vector3 lastPosition;
    private Vector3 lastBallPosition;
    private float rewardForBallClose = 0.1f;
    private float rewardForGoalClose = 0.05f;
    private float distanceThreshold = 0.5f;

    private RayPerceptionSensorComponent3D raySensor;

    public override void Initialize()
    {
        rb = GetComponent<Rigidbody>();
        lastPosition = transform.position;
        lastBallPosition = ball.position;

        // Инициализируем сенсор
        raySensor = GetComponent<RayPerceptionSensorComponent3D>();

        // Загружаем начальные позиции из объекта "AgentStart"
        if (GameObject.Find("AgentStart") != null)
        {
            transform.position = GameObject.Find("AgentStart").transform.position;
        }
        // Загружаем начальные позиции мяча из объекта "BallStart"
        if (GameObject.Find("BallStart") != null)
        {
            ball.position = GameObject.Find("BallStart").transform.position;
        }
    }
    public override void Heuristic(in ActionBuffers actionsOut)
    {
        var continuousActionsOut = actionsOut.ContinuousActions;
        continuousActionsOut[0] = Input.GetAxisRaw("Horizontal");
        continuousActionsOut[1] = Input.GetAxisRaw("Vertical");
    }
```

```
    public void Lose()
    {
        SetReward(-5f);
        EndEpisode();
        playerScoreNumber += 1;
    }

    public void ScoreGoal()
    {
        SetReward(5f);
        EndEpisode();
        enemyScoreNumber += 1;
    }

    public override void OnActionReceived(ActionBuffers actions)
    {
        float moveHorizontal = actions.ContinuousActions[0];
        float moveVertical = actions.ContinuousActions[1];

        Vector3 movement = new Vector3(moveHorizontal, 0f, moveVertical).normalized;

        rb.velocity = movement * speed;

        float distanceToBall = Vector3.Distance(transform.position, ball.position);
        if (distanceToBall < 1f && ball.GetComponent<Rigidbody>().velocity.magnitude <
0.1f)
        {
            AddReward(rewardForBallClose);
        }

        if (Vector3.Distance(transform.position, lastPosition) < 0.01f)
        {
            AddReward(-0.01f);
        }
        float distanceToLastPosition = Vector3.Distance(transform.position, lastPosition);
        if (distanceToLastPosition < 0.01f)
        {
            AddReward(-0.001f);
        }
        float distanceBallToGoal = Vector3.Distance(ball.position, goal.position);
        float ballMovement = Vector3.Distance(ball.position, lastBallPosition);



        lastPosition = transform.position;
        lastBallPosition = ball.position;
    }

    public override void OnEpisodeBegin()
    {
        Vector3 agentOffset = new Vector3(Random.Range(-2f, 2f), 0.5f, Random.Range(-2f,
2f));
        Vector3 ballOffset = new Vector3(Random.Range(-2f, 2f), 0f, Random.Range(-2f,
2f));

        transform.position = spawn.position + agentOffset;
        ball.position = spawn.position + ballOffset;
```

```
        rb.velocity = Vector3.zero;
        rb.angularVelocity = Vector3.zero;
        ball.GetComponent<Rigidbody>().velocity = Vector3.zero;
        ball.GetComponent<Rigidbody>().angularVelocity = Vector3.zero;

        SetReward(0f);

        lastPosition = transform.position;
        lastBallPosition = ball.position;
    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Wall"))
        {
            SetReward(-0.1f);
            EndEpisode();
        }
    }
}
```

ДОДАТОК Б

Код управління персонажем на мові C#

PlayerController.cs

```
using UnityEngine;

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    public float speed = 10f;
    public float mouseSensitivity = 100f;
    public Transform playerBody;

    private Rigidbody rb;
    private float xRotation = 0f;

    void Start()
    {
        rb = GetComponent<Rigidbody>();

        // Скрыть и зафиксировать курсор мыши
        Cursor.lockState = CursorLockMode.Locked;
    }

    void Update()
    {
        // Обработка ввода мыши
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

        // Обработка поворота камеры мышью
        xRotation -= mouseY;
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);

        Camera.main.transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
        playerBody.Rotate(Vector3.up * mouseX);
    }

    void FixedUpdate()
    {
        // Обработка ввода пользователя для движения
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        // Вычисляем направление движения
        Vector3 movement = playerBody.transform.forward * moveVertical +
playerBody.transform.right * moveHorizontal;
        movement = movement.normalized;

        rb.velocity = movement * speed;
    }
}
```

ДОДАТОК В

Код меню на мові C#

MenuScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Rendering;
using UnityEngine.SceneManagement;

public class MenuScript : MonoBehaviour
{
    public GameObject settingPanel;
    public GameObject menuPanel;
    public void StartGame()
    {
        SceneManager.LoadScene("PlayScene");
    }
    public void ShowSettings()
    {
        settingPanel.SetActive(true);
        menuPanel.SetActive(false);
    }
    public void ExitGame()
    {
        Application.Quit();
    }
    public void HideSettings()
    {
        settingPanel.SetActive(false);
        menuPanel.SetActive(true);
    }
}
```