Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка ігрового додатку з використанням нейронної мережі на рушії Unity»

Виконав: студент групи К20-3
Спеціальність 122 «Комп'ютерні науки»

Головко Микита Вячеславович
(прізвище та ініціали)

Керівник к.т.н., доц. Мала Ю.А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Університет митної справи та фінансів
(місце роботи)

доцент кафедри кібербезпеки та інформаційних
(посада)

технологій, к.т.н., доц. Клим В.Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Головко М.В. Розробка ігрового додатку з використанням нейронної мережі на рушії Unity

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2024.

Дана кваліфікаційна робота присвячена розробці ігрового додатку з використанням нейронної мережі на рушії Unity. Розглянуті методи та технічні засоби, необхідні для реалізації додатку, сформульовані вимоги до програмного забезпечення. У результаті було проведено проектування та розробку гри, створено датасет, розроблено та натрененовано нейронну мережу з метою досягнення високої точності та швидкості класифікації класу зображення. Також була здійснена інтеграція навченої моделі нейронної мережі в гру та реалізована взаємодія з нею у вигляді головоломок. Розроблений програмний додаток є надійним та стабільним, використовує передові технології, такі як згорткова нейронна мережа для класифікації зображень.

Розроблено ігровий додаток та натреновано згорткову нейронну мережу, яка класифікує малюнки, створені гравцем у додатку, в один з класів та створює перед ним відповідний об'єкт.

Ключові слова: Unity, Tensorflow, Barracuda, нейронна мережа, модель, скрипт, тренування, валідація, класифікація, розробка, датасет.

ABSTRACT

Holovko M.V. Development of a game application using a neural network on the Unity engine

Qualification work for a bachelor's degree in speciality 122 «Computer Science». - University of Customs and Finance, Dnipro, 2024.

This bachelor's thesis is dedicated to the development of a game application using a neural network on the Unity engine. The methods and technical means necessary for the implementation of the system are considered, the requirements for the software are formulated. As a result, the game was designed and developed, a dataset was created, and a neural network was developed and trained to achieve high accuracy and speed of image classification. The trained neural network model was also integrated into the game and interacted with in the form of puzzles. The developed software application is reliable and stable, using advanced technologies such as convolutional neural network for image classification.

A gaming application was developed and a convolutional neural network was trained, which classifies drawings created by the player in the application into one of the classes and creates a corresponding object in front of it.

Keywords: Unity, Tensorflow, neural network, model, script, training, validation, classification, development, dataset.

ЗМІСТ

ВСТУП

У сучасному світі обчислювальні технології знаходять широке застосування у різних сферах діяльності людини. Основною метою цього напрямку є створення інтелектуальних автоматизованих систем, здатних виконувати функції, аналогічні до людських. Одним із ключових інструментів штучного інтелекту є нейронні мережі, які здатні навчатися на великих обсягах даних, розпізнавати патерни та приймати рішення на основі цієї інформації.

В останні роки нейронні мережі отримали значний імпульс росту завдяки своїй здатності до глибокого навчання. Цей підхід дозволяє нейронним мережам автоматично визначати корисні ознаки з вхідних даних і використовувати їх для досягнення високої точності в різноманітних задачах, таких як розпізнавання образів, обробка природної мови та багатьох інших проблемах.

Важливим аспектом розвитку нейронних мереж є їхня здатність до постійного удосконалення через навчання на нових даних. Це дозволяє системам штучного інтелекту адаптуватися до змінних умов і забезпечує їхню ефективність в різних областях. Завдяки поєднанню розуміння людських функцій та здатності до аналізу великих обсягів інформації, нейронні мережі стають все більш потужним інструментом для вирішення складних завдань в нашому світі.

Однією з захоплюючих областей застосування нейронних мереж є їх використання для розпізнавання об'єктів у відеоіграх. Це дозволяє створювати реалістичні та інтелектуальні ігрові середовища, де реакція на дії гравця може змінюватися в залежності від зміни ситуації у грі. Використання нейронних мереж у ігорвих додатках дозволяє підвищити рівень автентичності та іммерсивності геймплею, що робить ігровий досвід більш захоплюючим для гравців.

У галузі розробки відеоігор на рушії Unity, застосування нейронних мереж для розпізнавання об'єктів стає все більш поширеним. Unity надає зручні

інструменти для інтеграції нейронних мереж у геймплей, що дозволяє розробникам створювати реалістичні та інтерактивні ігрові світи.

Сучасні інструменти для розробки нейронних мереж, такі як TensorFlow, значно спрощують процес створення та впровадження складних моделей глибокого навчання. Вони надають розробникам потужні засоби для роботи з даними, конструювання моделей і оптимізації процесу навчання. У поєднанні з Unity, ці технології відкривають нові можливості для створення ігор, де штучний інтелект грає ключову роль та створюють новий рівень взаємодії та глибини у відеоіграх, надаючи гравцям більш динамічний та інтуїтивний геймплей.

Мета дослідження полягає в розробці ігрового додатку жанру головоломка на рушії Unity та тренуванні згорткової нейронної мережі для розпізнавання класу об’єкта, намальованого гравцем. Дослідження спрямоване на розробку та впровадження методів взаємодії в гральному процесі за допомогою нейронних мереж.

Методи дослідження включають: теорію інформації, обробку та аналіз інформації, методи проектування та розробки програмного забезпечення, методи штучного інтелекту.

Завдання дослідження: відповідно до поставленої мети, в кваліфікаційній роботі ставились та вирішувались наступні:

1. Тренування нейронної мережі з метою досягнення високої точності та швидкості розпізнавання класу зображення.
2. Створення прототипу гри на рушії Unity, що зможе вразити та зацікавити користувача.
3. Інтеграція навченої моделі нейронної мережі в гру та реалізацію взаємодії з нею у вигляді головоломок

Об’єктом дослідження процес створення гри з інтегруванням у процес гри нейронної мережі.

Предметом дослідження є розробка гри у жанрі головоломка, з інтегруванням у процес гри нейронної мережі для класифікації об’єктів , з використанням рушія Unity.

Структура роботи:

Розділ 1 Аналіз предметної області. Постановка задачі. У цьому розділі буде виконано пошук та аналіз публікацій стосовно теми ігрових додатків у жанрі головоломка на рушії Unity та методи для розробки методу класифікації об’єктів за допомогою Tensorflow.

Розділ 2. Проектування ігрового додатку з використанням нейронної мережі. У цьому розділі буде проаналізовано та обрано технології для розробки ігрового додатку та спроектовано архітектуру нейронної мережі.

Розділ 3. Практична реалізація додатку. У цьому розділі буде натреновано спроектовану нейронну мережу, розроблено графіку, механіки та метод для розпізнавання об'єктів у ігровому додатку на платформі Unity розроблено тестовий варіант ігрового додатку та викладено його в мережу.

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел з 20 найменувань. Обсяг роботи становить 53 сторінки, 37 рисунків, 1 таблиці.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд сфери створення ігрових додатків у жанрі головоломка

За останні роки індустрія створення ігрових додатків у жанрі головоломок стрімко розвивається. Сучасні ігрові рушії, такі як Unity, надають розробникам потужні інструменти для реалізації складних механік, графіки та інтерактивного контенту. Unity є одним з найпопулярніших рушіїв для створення ігор завдяки своїй доступності, потужності та активній спільноті користувачів.

Однією з ключових особливостей сучасних ігор-головоломок є їхній інтерактивний та захоплюючий геймплей, що стимулює логічне мислення та вирішення завдань. Такі ігри часто використовують різноманітні механіки, які варіюються від простих головоломок до складних логічних задач.

Для створення таких ігрових додатків на базі Unity часто використовуються мови програмування C# та візуальне програмування за допомогою Unity. Інструменти цього рушія дозволяють розробникам легко розробляти та тестувати ігрові механіки, а також швидко вносити зміни та вдосконалювати гру.

Unity — це багатоплатформенний ігровий рушій, розроблений компанією Unity Technologies. Цей рушій дозволяє створювати додатки, які можуть функціонувати в двовимірному та тривимірному просторах, а також підтримує роботу з пристроями віртуальної реальності. [1]

Основні характеристики Unity:

− Мова програмування: Для розробки ігор на Unity використовується мова C#. Раніше рушій підтримував також мови Boo та JavaScript, але з часом розробники відмовилися від їх підтримки.

− Інтерфейс: Інтерфейс Unity складається з різних вікон. За замовчуванням, на головному екрані присутні вікна оглядача ресурсів, інспектора для маніпуляцій з об'єктами, вікно сцени з різними режимами та оглядач ієрархії проекту.

− Організація проекту: Проект в Unity поділений на сцени, які представляють собою окремі файли з усією інформацією про певний рівень. Сцени можуть містити 3D моделі чи 2D спрайти, а також об'єкти з прикріпленими скриптами для виконання ролі контролерів, тригерів, точок збереження тощо.

− Маніпуляції з об'єктами: Кожен об'єкт можна переміщувати, змінювати його масштаб та обертати за допомогою спеціальних інструментів або компонентів у вікні інспектора. Об'єктам можна присвоїти теги або додати до шарів для групування, створюваних користувачем.

− Фізика: Рушій підтримує фізику твердих тіл та тканин, що дозволяє змінювати сценарії взаємодії тіл або силу гравітації в налаштуваннях проекту.

− Рендеринг: Рендеринг об'єктів здійснюється за допомогою віртуальної камери, яка може працювати в перспективному або ортографічному режимах. Об'єкти можна переміщувати вглиб сцени навіть в ортографічному режимі.

− Графічні можливості: Unity підтримує DirectX (Windows), OpenGL (Windows, Mac, Linux), OpenGL ES (мобільні платформи). Для роботи з шейдерами використовується ShaderLab, який підтримує програми, написані на GLSL та Cg.

− Скрипти: Рушій дозволяє створювати користувацькі скрипти, які можуть бути додані до об'єктів у вигляді компонентів.

Переваги та недоліки Unity:

Переваги:

- Наявність візуального середовища розробки.
- Мультиплатформенна підтримка, яка включає інструменти для розробки під різні платформи.
- Велика кількість навчальних матеріалів та документації. Наприклад курси від самого Unity, такі як Create with code [2].

Недоліки:

- Складність роботи в складних сценах через обмеження візуального редактора.
- Деякі проблеми при роботі з зовнішніми бібліотеками.

1.2 Аналіз ігрових додатків – аналогів

Щоб краще зрозуміти сучасні тенденції та можливості рушія Unity у жанрі головоломок, було проведено аналіз трьох успішних ігор:

- Monument Valley на базі рушія Unity;
- The Room на базі рушія Unity;
- Limbo на базі рушія Unity.

Ці ігри представляють собою високоякісні продукти, створені з використанням рушія Unity, і демонструють різноманітні підходи до створення головоломок, графіки та геймплею. Кожна з них заслуговує на детальну увагу, оскільки пропонує унікальні рішення та інноваційні ідеї, які можуть стати джерелом натхнення для розробників.

Monument Valley є однією з найбільш відомих головоломок, що використовує елементи оптичних ілюзій та перспективних змін. Гра відзначається своєю естетичною красою, мінімалістичною графікою та інтуїтивно зрозумілим геймплеєм. Вона використовує потенціал свого жанру для створення унікальних і захоплюючих рівнів, де гравці мають взаємодіяти з оточенням, щоб змінювати його структуру та проходити далі (див. рис. 1.1).

Розробка цієї гри була здійснена за допомогою C# та Unity, що дозволило створити унікальні механіки та стилістику. [3]

Рисунок 1.1 – Представлення гри Monument Valley

The Room – серія головоломок, відома своєю атмосферою та складними механіками. Гра використовує інтерактивні елементи та деталізовані 3D-моделі для створення захоплюючого досвіду (див. рис. 1.2). Використання Unity дозволило розробникам легко створювати та анімувати складні об’єкти та механізми. [4]

Рисунок 1.2 – Представлення гри The Room

Limbo представляє собою платформер з елементами головоломок, що вирізняється своєю унікальною монохромною стилістикою та атмосферою тривожності. Гра використовує двовимірну графіку та фізичні головоломки, що базуються на взаємодії персонажа з оточенням. Завдяки можливостям Unity, розробникам вдалося створити незабутній світ, де кожен елемент служить не лише частиною фону, але й функціональною частиною головоломки. [5]

Рисунок 1.3 – Представлення гри Limbo

Ці три гри є прикладами того, як рушій Unity може бути використаний для створення різноманітних та захоплюючих головоломок, що залучають гравців своєю графікою, геймплеєм та інноваційним підходом до вирішення завдань. Вони демонструють, як використання сучасних технологій та креативних рішень може перетворити звичайну гру в унікальний досвід. Тобто необв’язково захоплювати гравця лише складністю, існують ігри-головоломки у різних піджанрах, які мають надати ігроку різну палітру емоцій, сюжетом, красою та деталізованістю світу або звуковим чи музикальним супроводом.

1.3 Методи для розробки нейро-мережі з метою класифікації об’єктів

Однією з важливих задач у розробці цього ігрового додатку є створення та тренування нейронної мережі для виконанння задачі класифікації об’єктів. Це буде включати розпізнавання малюнків гравця. Існує кілька підходів до реалізації такого, включаючи машинне навчання та алгоритми комп’ютерного зору.

Для класифікації об’єктів в ігрових додатках можна використовувати бібліотеки з функціонал навчання нейронних мереж, для реалізації натренованої моделі у Unity можна використовувати бібліотеку нейромережевого виводу, таку як Barracuda. Але перед тим треба конвертувати модель отриману у результаті тренування на датасеті та аугментації даних, також перетворення створенної моделі класифікації у підтримуємий бібліотекою Barracuda формат .onnx .

Також важливо враховувати вимоги до продуктивності та оптимізації, оскільки системи класифікації об’єктів можуть бути досить ресурсомісткими. Правильне налаштування та оптимізація алгоритмів є ключовими для забезпечення плавної роботи гри.

Одним з ефективних підходів до реалізації нейронних мереж у ігрових додатках є використання попередньо натренованих моделей. Існує багато готових нейронних мереж, які можна використовувати для різних задач, включаючи класифікацію об'єктів, розпізнавання зображень та інше. Використання таких моделей дозволяє скоротити час на розробку та тренування нейронної мережі, що є значною перевагою для розробників ігор.

Переваги використання готових нейронних мереж:

- Швидкість розробки: Готові моделі дозволяють зекономити час, оскільки не потрібно витрачати час на тренування нової нейронної мережі.
- Точність: Попередньо натреновані моделі зазвичай мають високу точність, оскільки вони тренуються на великих та різноманітних наборах даних.
- Простота інтеграції: Більшість готових моделей легко інтегруються в Unity за допомогою бібліотек, таких як Barracuda.

Недоліки використання готових нейронних мереж:

− Обмежена гнучкість: Використання готових моделей може бути обмеженим у специфічних задачах, які потребують налаштування або додаткового тренування.

− Ресурсомісткість: Деякі попередньо натреновані моделі можуть вимагати значних обчислювальних ресурсів для ефективної роботи.

Приклади використання готових нейронних мереж у ігрових додатках

Одним з прикладів успішного використання готових нейронних мереж є гра "Pokemon Go", де використовуються алгоритми машинного навчання для розпізнавання об'єктів у реальному світі через камеру смартфона. Іншим прикладом є гра "Angry Birds", де використовуються нейронні мережі для аналізу поведінки гравця та адаптації рівнів відповідно до його навичок.

1.4 Інструменти та ресурси для навчання та впровадження нейронних мереж

Для успішного впровадження нейронних мереж у ігрові додатки необхідно використовувати відповідні інструменти та ресурси. Unity пропонує кілька потужних інструментів та бібліотек, які спрощують процес розробки та інтеграції нейронних мереж.

Інструменти для навчання нейронних мереж:

TensorFlow: Це одна з найпопулярніших бібліотек для машинного навчання, яка надає широкий набір інструментів для створення, тренування та тестування нейронних мереж.

PyTorch: Ще одна популярна бібліотека для машинного навчання, яка відома своєю гнучкістю та простотою використання.

Keras: Високорівнева бібліотека для нейронних мереж, яка працює поверх TensorFlow і надає зручний інтерфейс для швидкої розробки моделей.

Інструменти для впровадження нейронних мереж у Unity:

Unity Barracuda: Бібліотека для виводу нейронних мереж у Unity, яка підтримує формат .onnx. Вона дозволяє легко інтегрувати та використовувати нейронні мережі у ігрових додатках.

Unity ML-Agents: Інструмент, який дозволяє створювати та тренувати інтелектуальні агенти, які можуть навчатися різним завданням у ігрових середовищах.

1.5 Оптимізація та продуктивність нейронних мереж у іграх

Оптимізація нейронних мереж є важливим аспектом для забезпечення високої продуктивності ігрових додатків. Використання ефективних алгоритмів та технік оптимізації дозволяє зменшити навантаження на систему та забезпечити плавну роботу гри.

Методи оптимізації нейронних мереж:

- Квантування: Зменшення точності ваг нейронної мережі для зменшення обчислювальних витрат та зменшення використання пам'яті.
- Прунинг: Видалення маловажливих нейронів або шарів з моделі для зменшення її складності та прискорення виконання.
- Паралелізація: Використання багатоядерних процесорів та графічних процесорів для прискорення обчислень.

Важливість оптимізації для ігрових додатків:

- Зменшення затримок: Оптимізація допомагає зменшити затримки в обробці даних, що є критичним для реального часу.
- Покращення користувацького досвіду: Плавна робота гри без затримок та збоїв значно покращує загальний користувацький досвід.
- Ефективне використання ресурсів: Оптимізовані нейронні мережі використовують менше ресурсів, що дозволяє запускати гру на пристроях з обмеженими можливостями.

1.6 Виклики та перспективи використання нейронних мереж у ігрових додатках

Використання нейронних мереж у ігрових додатках надає багато нових можливостей, але також супроводжується певними викликами.

Основні виклики:

- Комплексність розробки: Розробка та інтеграція нейронних мереж вимагає знань та досвіду.
- Ресурсомісткість: Нейронні мережі можуть вимагати значних обчислювальних ресурсів, що може бути проблемою для мобільних та вбудованих пристроїв.
- Забезпечення продуктивності: Необхідно постійно оптимізувати моделі для забезпечення плавної роботи гри.

Перспективи використання нейронних мереж:

- Розширення можливостей геймплею: Нейронні мережі можуть створювати нові можливості для інтерактивності та варіативності геймплею.

1.7 Створення 3D-моделей та анімація в Unity

Важливим аспектом створення ігрових додатків є розробка 3D-моделей та їх анімація. Unity надає потужні інструменти для створення та інтеграції 3D-моделей у ігрове середовище. Крім того, Unity підтримує технології фізики та освітлення, що дозволяє створювати реалістичні сцени та взаємодію об'єктів у грі.

Створення 3D-моделей: Для створення 3D-моделей використовуються різні програми, такі як Blender, Autodesk Maya або 3ds Max. Після створення моделі вона імпортується у Unity, де її можна налаштовувати, додавати текстури та матеріали. Unity підтримує формат FBX, який забезпечує високу якість імпортованих моделей.

Анімація: Unity також підтримує анімацію 3D-моделей, яка може бути створена безпосередньо в редакторі або імпортована з інших програм, таких як Blender або Maya. Для цього використовуються анімаційні кліпи та система Animator, яка дозволяє налаштовувати складні анімаційні переходи та стан машини.

1.8 Висновки до першого розділу. Постановка задач дослідження.

Unity є одним з найпопулярніших ігрових рушіїв завдяки своїй доступності, потужності та активній спільноті користувачів. Багатоплатформені можливості Unity дозволяють створювати ігри, що можуть працювати на різних пристроях, включаючи мобільні платформи та пристрої віртуальної реальності.

Для створення ігрових додатків на Unity використовуються мова програмування C# та візуальне програмування. Unity надає зручні інструменти для розробки та тестування ігрових механік, що дозволяє розробникам швидко вносити зміни та вдосконалювати гру.

Unity підтримує створення та анімацію 3D-моделей, що дозволяє розробникам інтегрувати високоякісні моделі та анімації у свої проекти.

Однією з найважливіших задач у розробці цього ігрового додатку є створення та тренування нейронної мережі для класифікації об’єктів з використанням алгоритмів машинного навчання.

На основі попереднього аналізу предметної області для досягнення мети проекту у ході виконання кваліфікаційної роботи буде розроблено ігровий додаток у жанрі головоломка на базі рушія Unity, а також нейронну мережу для класифікації об’єктів.

У відповідності до поставленої мети в кваліфікаційній роботі ставились та наступні завдання дослідження:

1. Проведення огляду літератури та онлайн-ресурсів стосовно створення ігор на базі Unity;

2. Написання вимог до гри та специфікацій;
3. Створення повноцінних геймплейних механік;
4. Розробка 3D-моделей персонажа та елементів оточення;
5. Реалізація ігрових рівнів (сцен);
6. Створення та тренування нейронної мережі з метою класифікації об’єктів з використанням алгоритмів машинного навчання;
7. Оптимізація продуктивності гри;
8. Тестування та відлагодження гри;
9. Випуск тестової версії гри на платформі Windows.

Загалом Unity є потужним інструментом для створення ігор у жанрі головоломка, що дозволяє реалізувати складні механіки, інтерактивні елементи та забезпечує підтримку багатьох платформ. Ці можливості роблять рушій привабливим вибором для розробників, які прагнуть створювати інноваційні та захоплюючі ігрові досвіди.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ІГРОВОГО ДОДАТКУ З ВИКОРИСТАННЯМ НЕЙРОННОЇ МЕРЕЖІ

2.1 Системи гравця

Системи, що належать об'єкту гравця, включають стани та класи, які є специфічними для цього об'єкта. Гравець у першій особі може перебувати в таких станах, як ходьба, стрибок, перетягування та обертання предметів.

Основні класи, які входять до системи гравця:

1. PersonController: Клас відповідає за основні дії гравця, такі як ходьба, стрибки та активація режиму малювання.
2. PlayerInteraction: Клас, що дозволяє гравцеві взаємодіяти з об'єктами, піднімати, переміщати та обертати їх.
3. CameraController: Клас, який керує камерою від першої особи, забезпечуючи правильне відображення світу та взаємодію з ним.
4. Animation Controller: Клас, який керує анімаціями гравця, забезпечуючи більшу імерсивність с ігровим світом.

Рисунок 2.1 – Схема зв’язків між класами гравця

Діаграма станів гравця (див. рис. 2.2) відображає можливості та обмеження, що були впроваджені, і стане в пригоді для створення дерева станів в ігровому рушії для анімацій.

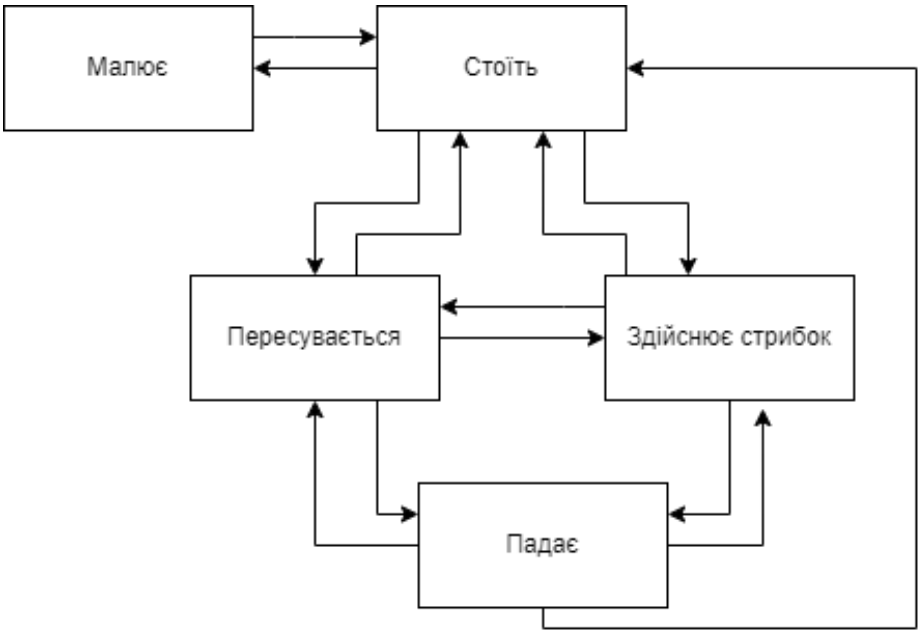


Рисунок 2.2 – Діаграма станів гравця

2.2 Проектування ігрової системи

Цикл ігрового процесу можна зобразити на діаграмі стані, тобто діаграмі, що визначає зміну станів об’єкта у часі.

Діаграма станів – діаграма, що визначає стан об’єкта у часі, одна з діаграм поведінки в UML. Вона дозволяє відслідковувати зміни станів ігрових об'єктів під час їхньої взаємодії з користувачем, що особливо корисно для проектування та відлагодження поведінки в ігрових додатках [6].

Елементами діаграми є:

- Коло, що позначає початковий стан
- Коло з обводкою, що позначає кінцевий стан.
- Стрілка що позначає перехід. Назва події, що викликає перехід.

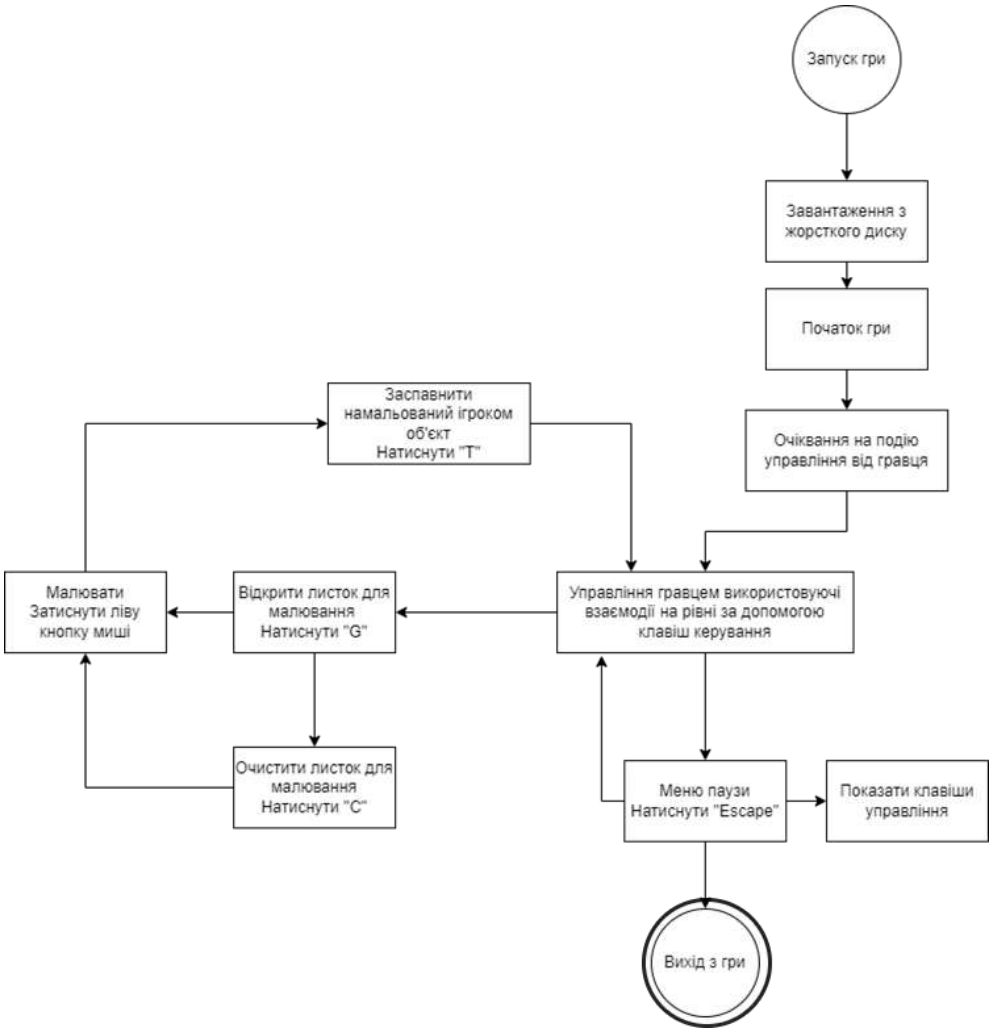


Рисунок 2.3 – Діаграма станів ігрової системи

Опис діаграми станів:

- Запуск гри: Початковий стан, коли гра запускається.
- Завантаження з жорсткого диску: Стан завантаження необхідних ресурсів для гри.
- Початок гри: Після завершення завантаження гравець переходить до початкового стану гри.
- Очікування на подію управління від гравця: Гра знаходиться в стані очікування дій від гравця.
- Управління гравцем: Гравець керує персонажем, взаємодіє з об'єктами та використовує клавіші управління.

- Меню паузи: При натисканні клавіші "Escape" гравець потрапляє в меню паузи.
- Показати клавіші управління: В меню паузи гравець може переглянути клавіші управління.
- Відкрити листок для малювання: Натиснувши "G", гравець відкриває режим малювання.
- Малювати: Затиснувши ліву кнопку миші, гравець може малювати.
- Очистити листок для малювання: Натиснувши "C", гравець очищує листок для малювання.
- Заспавнити намальований об'єкт: Натиснувши "T", гравець створює об'єкт на основі малюнка.
- Вихід з гри: Кінцевий стан, коли гравець вирішує завершити гру.

Додаткові деталі ігрової системи:

- Взаємодія з об'єктами: Гравець може взаємодіяти з різними об'єктами у грі, такими як двері або інші інтерактивні елементи. Це включає підняття, перетягування та обертання предметів.
- Система малювання: Гравець може малювати предмети, які розпізнає нейронна мережа та класифікує. Цей процес включає перетворення малюнків та класифікацію малюнків використовуючи натреновану згорткову мережу. Після завершення малюнка предмет з'являється у світі гри і може бути використаний для вирішення головоломок або досягнення цілей.
- Ігрові події: У грі можуть відбуватися різні події, такі як зустрічі з пазлами, фізичні головоломки.
- Анімація персонажа: Кожен стан анімації відповідає конкретним діям гравця, таким як ходьба, біг, стрибки. Анімації допомагають зробити гру більш реалістичною та захоплюючою.

2.3 Планування першого рівня

Працюючи із жанром головомки з унікальною механікою треба провести гравцю невеликий туторіал з легкими пазлами, але не розказувати всі способи проходження цих пазлів. По-перше інтуїтивне ознайомлення гравця з основною механікою гри без надавання йому належної інформації про використання цієї ігрової механіки.

Ігри та архітектура відрізняються тим, що архітектура реального світу повинна відповідати правилам реального світу. Наприклад, реальні будівлі повинні мати як інтер'єр, так і екстер'єр, причому форма одного впливає на форму іншого. Реальна архітектура також повинна враховувати погодні умови, геологію, правила зонування та структурні реалії. Це не ті речі, з якими повинен мати справу ігровий простір.

Оскільки перший рівень це знаймовство гравця із грою – це найважливіша частина проекту, він має зацікавити його на продовження ігрової сесії. Було вирішено показати головні механіку у першому рівні та плавно розширяти можливісті їх використання та виклики гравцю [7].

2.4 Вибір фреймоворку для тренування нейронної мережі. Вибір формату датасета

Нейронні мережі стали важливим інструментом у сучасних іграх, особливо для задач, пов'язаних з розпізнаванням образів та класифікацією об'єктів. Було розглянуто як використовувати нейронну мережу для класифікації об'єктів у грі, а також визначено її структуру та необхідні компоненти.

Згорткові нейронні мережі або CNN є спеціалізованим типом нейронних мереж для обробки даних, які мають відому топологію, подібну до сітки. Приклади включають дані часових рядів, які можна розглядати як одномірну сітку, що робить вибірки через регулярні проміжки часу, і дані зображень, які

можна розглядати як двомірну сітку пікселів. Згорткові мережі виявилися надзвичайно успішними в практичному застосуванні. Назва «згорткова нейронна мережа» вказує на те, що мережа використовує математичну операцію, яка називається згортка. Згортання - це спеціалізований вид лінійної операції. Згорткові мережі - це просто нейронні мережі, які використовують згортку замість загального множення матриць принаймні в одному своєму шарі.

Саме тому як архітектуру було вибрано саме згорткову [8].

Основні компоненти системи з нейронною мережею:

1. Модель нейронної мережі: Основний компонент, який відповідає за розпізнавання та класифікацію об'єктів на основі малюнків, створених гравцем.

2. Обробка малюнків: Підготовка зображень для передачі в нейронну мережу.

3. Класифікація та створення об'єктів: Інтерпретація результатів нейронної мережі та створення відповідних об'єктів у грі.

Для завдання класифікації об'єктів використовується згорткова нейронна мережа, яка має декілька шарів:

1. Вхідний шар: Приймає зображення (малюнок гравця) як вхідні дані.

2. Приховані шари: Обробляють вхідні дані, витягуючи з них особливості та патерни.

3. Вихідний шар: Видає результат у вигляді ймовірностей приналежності зображення до певного класу об'єктів.

Процес обробки та класифікації:

1. Підготовка зображень: Малюнок, створений гравцем, обробляється та перетворюється у формат, придатний для передачі до нейронної мережі.

2. Класифікація: Нейронна мережа обробляє зображення та видає ймовірності для кожного класу об'єктів.

3. Інтерпретація результатів: На основі отриманих ймовірностей вибирається клас з найвищою ймовірністю, і створюється відповідний об'єкт у грі.

Для тренування нейронних мереж існує кілька популярних фреймворків, кожен з яких має свої переваги та недоліки. Розглянемо деякі з них, щоб зробити обґрунтований вибір для проекту:

Таблиця 2.1

Порівняння популярних фреймворків для тренування нейро-мереж

Фреймворк	Плюси	Мінуси
TensorFlow	Широкий набір інструментів для створення, тренування та деплойменту моделей. Підтримка як високорівневого, так і низькорівневого API. Відмінна документація та велика спільнота. Інтеграція з TensorBoard для візуалізації навчального процесу. Підтримка багатьох мов, включаючи Python, C++, JavaScript.	Відносно великий розмір бінарних файлів при розгортанні. Може бути громіздким для невеликих проектів.
PyTorch	Інтуїтивно зрозумілий та "pythonic" синтаксис. Динамічні обчислювальні графи, що спрощують відлагодження. Сильна підтримка дослідницьких проектів та швидка адаптація нових технологій.	Менша кількість інструментів для деплойменту та відстежування прогресу порівняно з TensorFlow. Відносно молода спільнота та екосистема.

	Гарна інтеграція з іншими бібліотеками Python.	
Keras	Високорівневий API, простий у використанні. Працює як фронтенд для TensorFlow, Theano або CNTK. Швидке прототипування завдяки простому та зрозумілому синтаксису.	Обмежена гнучкість порівняно з низькорівневими API. Менша кількість функцій для глибокої настройки та оптимізації.
MXNet	Потужний та гнучкий для розробки нових моделей. Підтримка обчислень на GPU.	Менша популярність та спільнота порівняно з TensorFlow та PyTorch. Більш складний синтаксис порівняно з PyTorch.

Згідно з порівняння буде використано Tensorflow [9], але також буде використано Keras, як його частину для зручного та інтуїтивного створення моделі потрібної нейронної мережі та роботи з даними, через його зручний синтаксис для визначення та тренування моделей мереж.

Для успішного тренування згорткової нейронної мережі важливо правильно підготувати та організувати датасет. Вибір формату датасету визначається специфікою завдання, типом вхідних даних та вимогами до обробки. Формат датасету для класифікації малюнку гравця є зображення, тобто формат .png. Підготовка до датасету включає в себе такі кроки:

1. Розбиття на тренувальні, валідаційні та тестові набори за співвідношення 80:10:10
2. Визначення кількості класів: у датасеті буде 5 класів.

3. Визначення розміру зображення: фіксована величина, у випадку класифікації простих малюнків невеликий розмір у 128*128 пікселей буде достатнім.

4. Визначення кольорової моделі зображення: чорно-білий колір для облегчення механіки малювання гравцем та задачі класифікації.

5. Створення зображень шляхом малювання у довільній програмі.

6. Аналіз та очищення даних: Видалення або коригування неправильних та відсутніх даних.

7. Аугментація даних: Створення нових зразків на основі наявних для збільшення розміру датасету (наприклад, обертання зображень, зміна яскравості тощо).

2.5 Налаштування архітектури нейронної мережі

Архітектура нейронної мережі визначає кількість шарів, кількість нейронів у кожному шарі та типи активаційних функцій. Використання фреймворка TensorFlow полегшує процес налаштування.

Вибір типу нейронної мережі: згорткова мережа для зображень [10].

Згорткова нейронна мережа (CNN) - це категорія моделей машинного навчання, а саме тип алгоритму глибокого навчання, який добре підходить для аналізу візуальних даних. ШНМ, які іноді називають згортковими мережами, використовують принципи лінійної алгебри, зокрема операції згортки, для вилучення ознак та виявлення закономірностей у зображеннях. Хоча ШНМ переважно використовують для обробки зображень, їх також можна адаптувати для роботи з аудіо та іншими сигнальними даними.

Архітектура Convolutional neural network натхненна моделями зв'язків людського мозку - зокрема, зорової кори, яка відіграє важливу роль у сприйнятті та обробці візуальних стимулів. Оскільки CNN настільки ефективно ідентифікують об'єкти, їх часто використовують для завдань комп'ютерного зору,

таких як розпізнавання зображень і виявлення об'єктів, зокрема, у самокерованих автомобілях, розпізнаванні облич і аналізі медичних зображень.

На відміну від CNN, старіші форми нейронних мереж часто обробляли візуальні дані фрагментарно, використовуючи сегментовані або з низькою роздільною здатністю вхідні зображення. Комплексний підхід CNN до розпізнавання зображень дозволяє їм перевершувати традиційні нейронні мережі в низці завдань, пов'язаних із зображеннями, і, меншою мірою, з обробкою мови та аудіо.

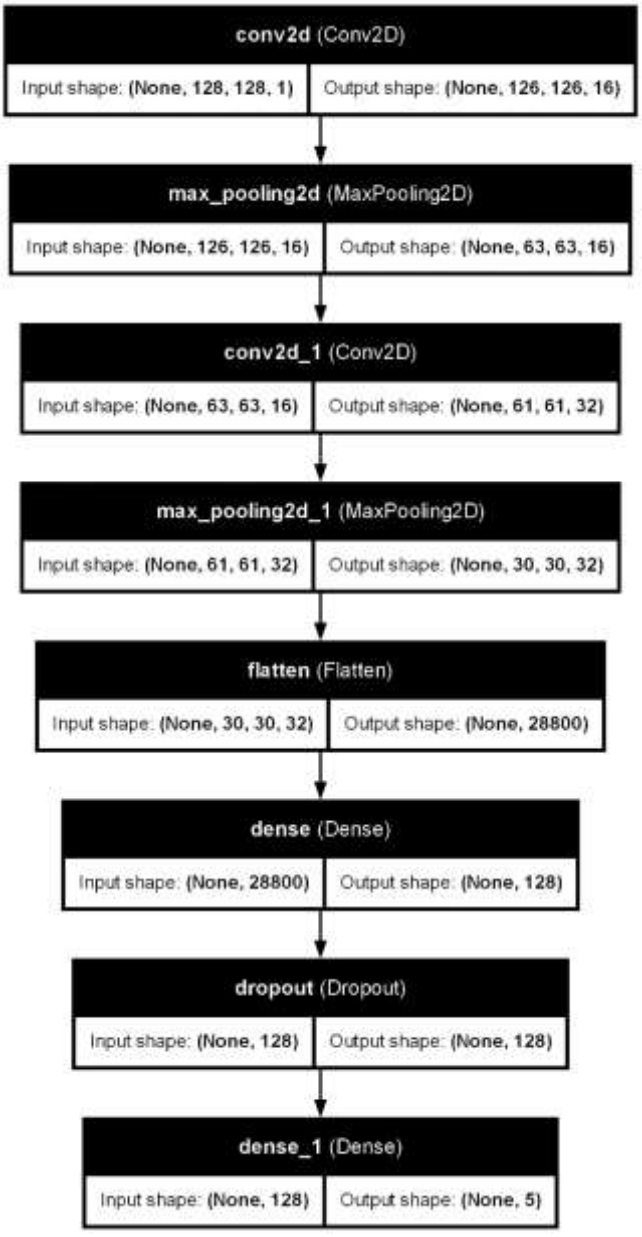


Рисунок 2.4 - Архітектура нейронної мережі

При проектуванні архітектури нейронної мережі згідно з завданням було випробувано декілька архітектур, але найточнішою опиналася архітектура , що складається з декількох ключових шарів, кожен з яких відіграє важливу роль у загальній функціональності та продуктивності моделі. Архітектура включає шари згортки (Conv2D), максимального підсумовування (MaxPooling2D), шару згортки (Conv2D), максимального підсумовування (MaxPooling2D), шару розгортки (Flatten), повнозв'язного шару (Dense), шару виключення (Dropout) та фінального повнозв'язного шару (Dense). Ця структура дозволяє нейронній мережі ефективно обробляти та класифікувати вхідні зображення, використовуючи шари згортки для витягування ознак та повнозв'язні шари для класифікації (див. рис. 2.4).

Типові згорткові нейронні мережі складаються зі згорткових шарів, що перемежовуються з шарами які зменшують вибірку вдвічі. По мірі того, як мережа прогресує, просторові розміри зазвичай зменшуються вдвічі, а кількість каналів збільшується вдвічі. В кінці мережі, як правило, є один або кілька повністю пов'язаних шарів які інтегрують інформацію з усіх входів і створюють бажаний вихід. Якщо виходом є зображення, дзеркальний «декодер» збільшує вибірку до початкового розміру.

У згорткових шарах кожна прихована одиниця обчислюється шляхом взяття зваженої суми сусідніх входів, додавання зсуву та застосування функції активації. Ваги та зсув однакові в кожній просторовій позиції, тому параметрів набагато менше, ніж у повністю зв'язній мережі, і кількість параметрів не збільшується зі збільшенням розміром вхідного зображення. Різні ваги та зміщення, щоб створити кілька каналів у кожній просторовій позиції [11].

2.6 Висновки до другого розділу

У другому розділі описано ключові системи та класи, що належать до об'єкта гравця, включаючи PersonController, PlayerInteraction, CameraController та Animation Controller. Представлено схему зв’язків між цими класами та діаграму станів гравця, що ілюструє можливі дії та переходи між станами.

Проектування ігрової системи охоплює ключові етапи ігрового процесу: запуск гри початок гри, управління персонажем, використання меню паузи, режим малювання та створення об'єктів на основі малюнків гравця.

Планування першого рівня включає створення туторіалу з легкими пазлами для інтуїтивного ознайомлення гравця з основною механікою гри, що має зацікавити гравця та стимулювати продовження ігрової сесії.

Обрано бібліотеки TensorFlow з Keras для тренування нейронної мережі завдяки їх зручності та функціональності. Визначено структуру згорткової нейронної мережі, основні компоненти системи та етапи підготовки датасету, включаючи розбиття на тренувальні, валідаційні та тестові набори, а також методи аугментації даних.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІГРОВОГО ДОДАТКУ

3.1 Розробка та тренування нейронної мережі

Нейронна мережа використовується для розпізнавання малюнків та створення відповідних об'єктів у грі. Це включає підготовку даних, налаштування архітектури мережі та тренування моделі.

3.1.1 Підготовка даних

Для тренування нейронної мережі необхідно зібрати великий набір даних малюнків, що представляють різні об'єкти. Ці дані мають бути попередньо оброблені, щоб відповідати вимогам моделі.

Датасет буде ділитися на 5 класів: двері, шестерня, ключ, лампа та сходи.

У тренувальному наборі буде представлено 200 намальованих вручну зображень формату .png розміром 128*128 пікселей. У валідаційному та тренувальному наборі представлено по 20 зображень.

Розпочнемо збір даних:

Створений вручну тренувальний набір ключей ручною аугментацією проворотом:

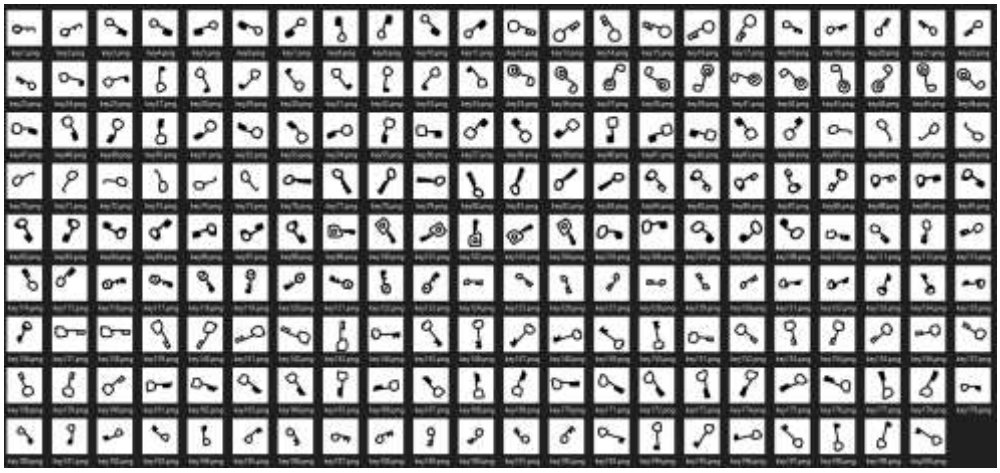


Рисунок 3.1 – Тренувальний набір ключей

Створений вручну тренувальний набір дверей з ручною аугментацією проворотом:

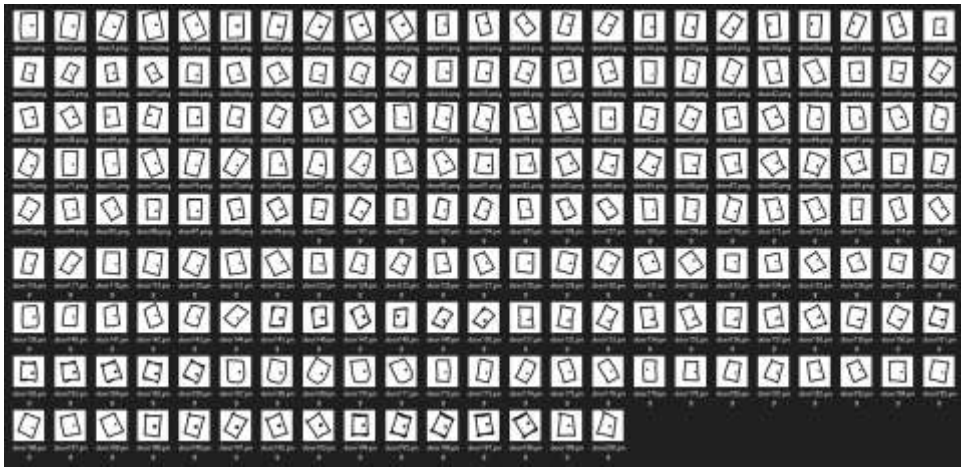


Рисунок 3.2 – Тренувальний набір дверей

Створений вручну тренувальний набір шестерень з ручною аугментацією проворотом:

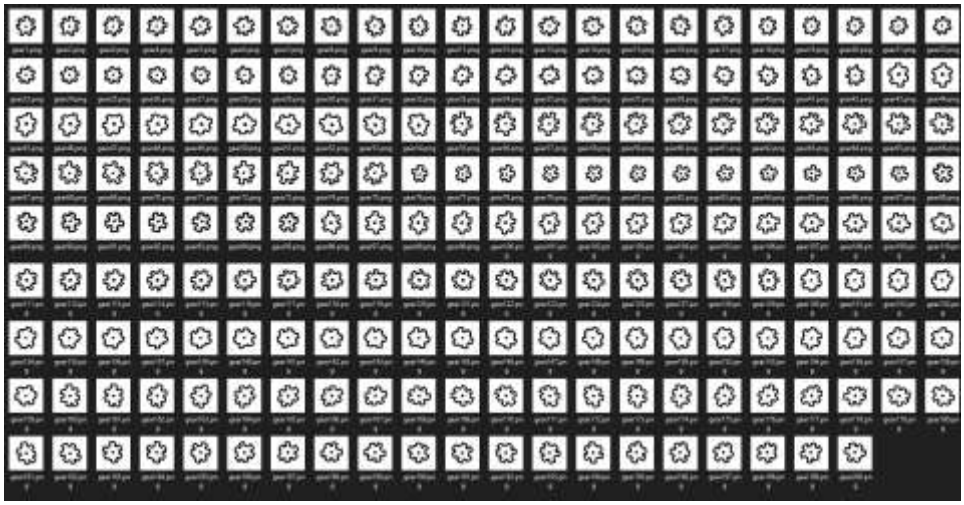


Рисунок 3.3 – Тренувальний набір шестерень

Створений вручну тренувальний набір ламп з ручною аугментацією проворотом:

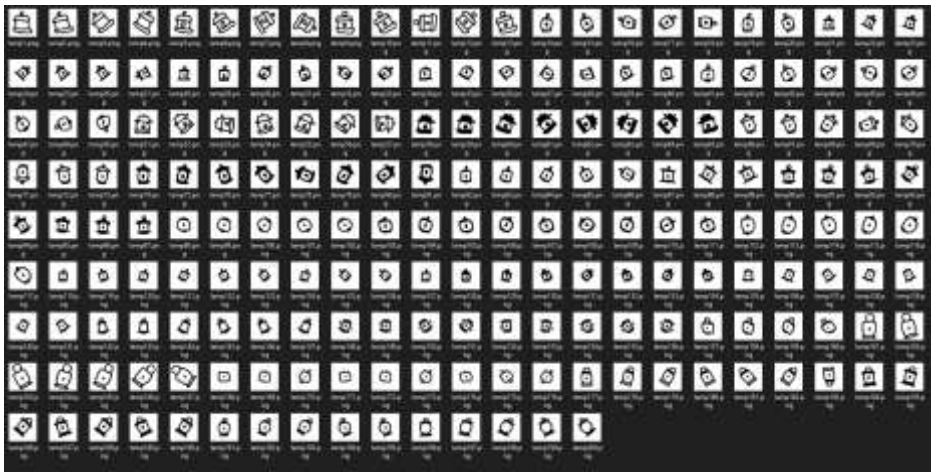


Рисунок 3.4 – Тренувальний набір ламп

Створений вручну тренувальний набір сходів з ручною аугментацією проворотом:

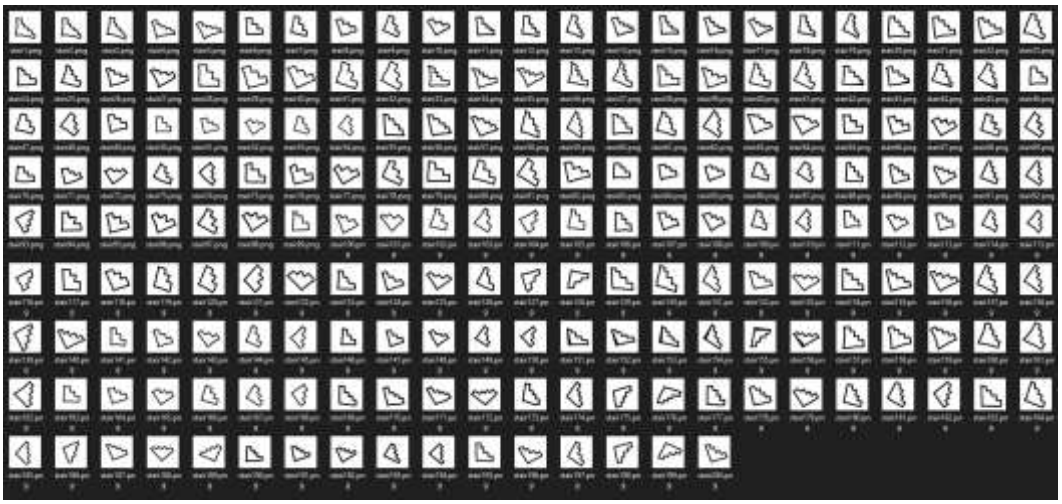


Рисунок 3.5 – Тренувальний набір сходів

Також були створені валідаційні та тестові набори по 20 малюнків кожен для всіх 5 класів.

Застосовуючи закони абстрактно-логічного мислення, основ методології наукового пізнання, аналізу, обробки та синтезу було забезпечено якісну підготовку даних для тренування нейронної мережі.

3.1.2 Тренування моделі

Тренування моделі включає навчання нейронної мережі на заздалегіть підготовлених даних, використовуючи методи зворотнього поширення помилки та оптимізації.

Ключові моменти:

Використання оптимізаційних алгоритмів: Adam, бо Adam - це адаптивний алгоритм швидкості навчання, розроблений для підвищення швидкості навчання в глибоких нейронних мережах і швидкого досягнення збіжності. Він автоматично налаштовує швидкість навчання для кожного параметра, використовуючи середньозважені перші та другі моменти градієнтів. Це робить Adam особливо ефективним для роботи з великими обсягами даних та високорозмірними параметрами [12].

Налаштування параметрів тренування: Кількість епох (epochs): Епоха означає один повний прохід через всі навчальні дані. Під час кожної епохи модель оновлює свої параметри, намагаючись мінімізувати функцію втрат. Кількість епох визначає, скільки разів модель буде навчатися на всіх наявних даних. Занадто мала кількість епох може призвести до недонавчання, тоді як занадто велика кількість може спричинити перенавчання. Дана нейронна мережа не потребує занадто великої кількості епох через можливість перенавчання через малу кількість даних у датасеті, але використовує максимальну кількість та завдяки методу запобігання перенавчання не перетинає цю межу.

Метод tf.keras.callbacks.EarlyStopping використовується для запобігання перенавчанню моделі під час тренування. Він слідкує за заданим показником (в даному випадку за втратою на валідаційному наборі даних val_loss) і припиняє тренування, якщо цей показник перестає покращуватись протягом визначеної кількості епох.

Розмір батчу (batch size): Батч - це кількість зразків, оброблюваних моделлю перед оновленням її параметрів. Малий розмір батчу може зробити

процес навчання більш стабільним, але потребує більше часу. Великий розмір батчу може прискорити навчання, але ризикує пропустити деталі в даних. Розмір батчу даної нейронної мережі буде дорівнювати 8.

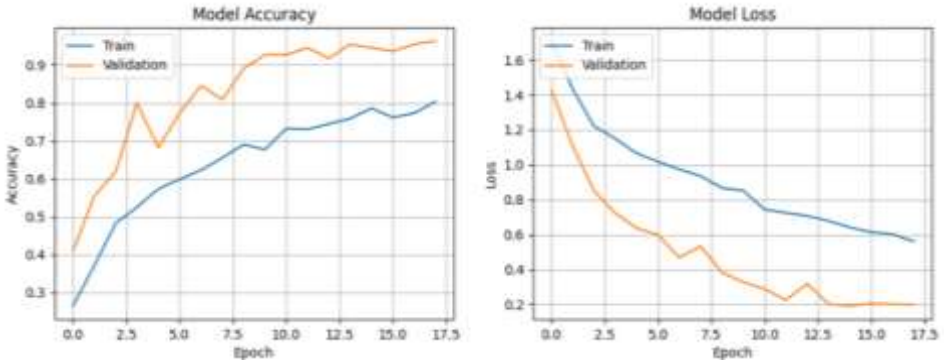


Рисунок 3.6 – Метрика навчання моделі.

Модель добре навчалася і показує хорошу узагальнену здатність(див. Рис. 3.6), оскільки валідаційна точність досить висока, а валідаційні втрати низькі. Це вказує на те, що модель не перенавчена (overfitting) і не недонавчена (underfitting).

Тестовий малюнок двері (див. рис. 3.7) вона класифікує за 0 індексом, тобто вона класифікує її як дверь. Зпрогнозований клас: [0], : Точність [0.9905862]

Рис 3.7 – Тестовий малюнок двері

Тестовий малюнок лампу(див. рис. 3.8) вона класифікує за 3 індексом, тобто вона класифікує її як лампу. Зпрогнозований клас: [3], : Точність [0.8823178].

Рисунок 3.8 – Тестовий малюнок лампи

З двох тестових прикладів видно, що нейромережа добре класифікує малюнки класів, на які була натренована.

3.2 Розробка графіки для гри

Графіка є ключовим елементом, що визначає візуальне враження від гри. Вона включає створення моделей персонажів, об'єктів та середовища, а також налаштування освітлення та тіней.

3.2.1 Малювання скетчу для персонажу гравця у 2D

Завдяки безкоштовній програмі для маніпуляції зображеннями та малювання GIMP було намальовано скетч для персонажу гравця (див. рис. 3.9).

Було також текстово прописано яким саме має бути персонаж:

1) Персонаж повинен мати темні волоси.
2) Футболку будь-якого кольору.
3) Джинси з довільним кольором.
4) Чорні кросівки.

Цей опис допомагає чітко уявити візуальні та стилістичні особливості персонажа, що сприяє кращій інтеграції його в ігровий процес та надає розробникам чіткі вказівки для подальшої роботи над дизайном.

Рис 3.9 – Скетч персонажу гравця

3.2.2 Моделювання персонажів та об'єктів

Моделювання вимагає використання 3D-редакторів, таких як Blender. Моделі з Blender можуть на пряму експортуватися у розширення моделей Unity .fbx. Що надає зручну зв’язку при якій можна одночасно робити моделі в блендері та відразу тестувати їх у самій грі [14].

При моделюванні моделі гравця, як і всіх об’єктів у грі було використано стиль low-poly. Цей стиль відзначається використанням невеликої кількості полігонів, що дозволяє створювати більш прості та абстрактні форми, зберігаючи при цьому високу продуктивність гри.

Рисунок 3.10 – Створена у 3D-редакторі Blender за скетчем модель гравця

3.2.3 Текстурування та матеріали

Текстурування додає деталізацію моделям, роблячи їх більш реалістичними. Blender дозволяє створювати UV розгортку персонажей та точно розмальовувати їх завдяки цьому (див. рис. 3.11) [15].

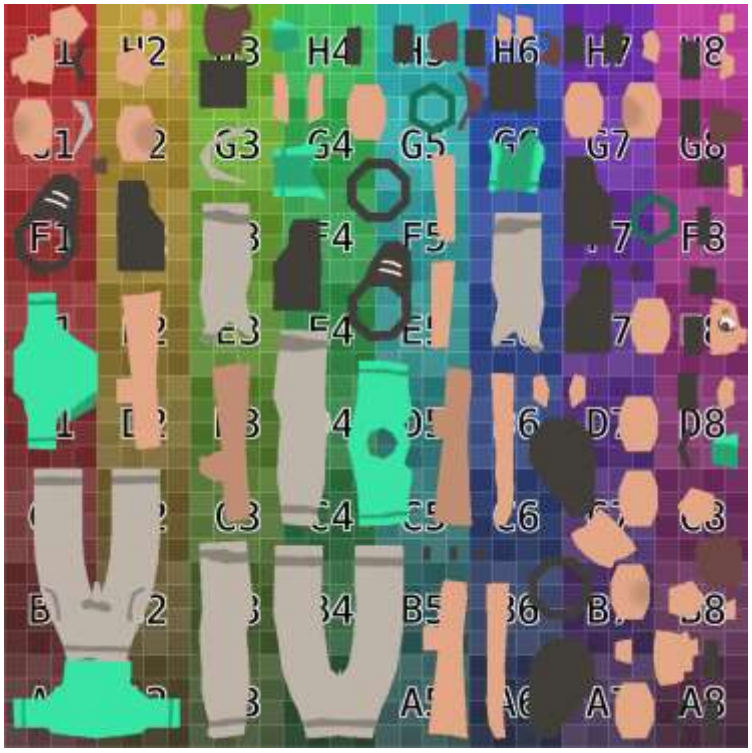


Рисунок 3.11 – UV Map створеного персонажу гравця

Створення безшовного матеріалу вимагає доброго розуміння ключових аспектів, які забезпечують плавний та природний вигляд матеріалу при його використанні на великих поверхнях, таких як розуміння текстури та патерну, або використання спеціалізованого програмного забезпечення, такого як Material Maker [16].

У Material Maker можна створювати процедурні матеріали за допомогою заготовок у самій програмі та візуального корегування матеріалів під свої потреби (див. рис. 3.12).

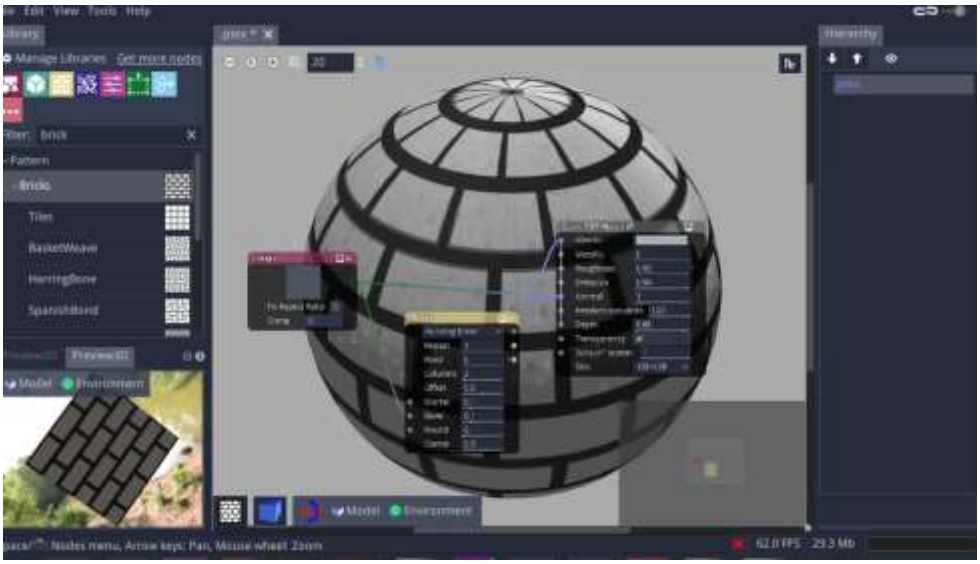


Рис. 3.12 – Створення матеріалу стінки

Завдяки візуальному редактору програми було створено процедурну текстуру для стін та полу.

3.2.4 Освітлення та тіні

Освітлення грає важливу роль у створенні атмосфери гри. Використання різних типів світильників, таких як напрямлене світло, точкове світло та світло прожектора, дозволяє створювати реалістичні сцени з глибокими тінями та м'яким освітленням (див. рис. 3.13).

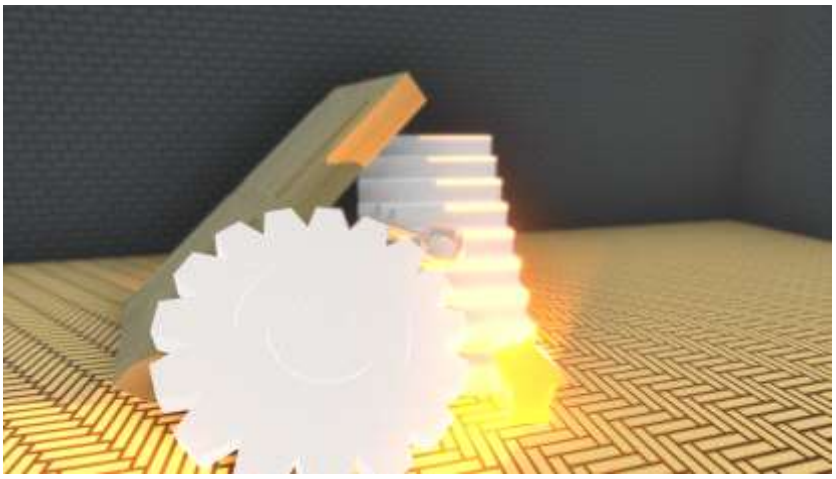


Рисунок 3.13 – Створені предмети у грі

Також в блендері можна використовувати різні налаштування освітлення, зробити пітьму де щось видно тільки з фонарем, або яскравий день завдяки сонцю та налаштовуваному небу.

3.3 Розробка механік гравця

Механіки гравця є основою будь-якої гри. Вони визначають, як гравець взаємодіє з ігровим світом, і створюють основу для загального ігрового процесу.

Розробка ігрового застосунку розпочалась із розробки механік гравця не пов’яз. Потрібно було забезпечити наступну функціональність:

- звичайне переміщення(біг);
- стрибки із описаними вище особливостями

Для більш зручної роботи було вирішено розбити функціонал гравця на чотири скрипта.

1) PlayerController.
2) CameraController.
3) PlayerInteraction
4) InteractionItem

3.3.1 Управління гравцем

Розробка механік управління гравцем є важливим етапом створення ігрового додатку. Вона включає налаштування контролерів для руху гравця, стрибків, взаємодії з об'єктами та керування камерою, яка слідкує за гравцем.

Контролер гравця реалізовано за допомогою CharacterController [17] компоненту Unity, який дозволяє ефективно керувати переміщенням гравця. Код контролера гравця міститься у файлі PlayerController.cs і включає наступні основні функціональні можливості:

– Рух гравця: Використання стандартних методів для обробки вводу з клавіатури дозволяє гравцеві переміщуватися в різних напрямках. При натисканні на клавіші "W", "A", "S", "D" або аналогічні на контролері гравець рухається відповідно вперед, вліво, назад та вправо.

– Гравітація та стрибки: Реалізовано механіку гравітації, яка дозволяє гравцеві природно падати під впливом гравітаційної сили. Також є можливість стрибка при натисканні клавіші "Space", якщо гравець знаходиться на землі (див. рис. 3.14).

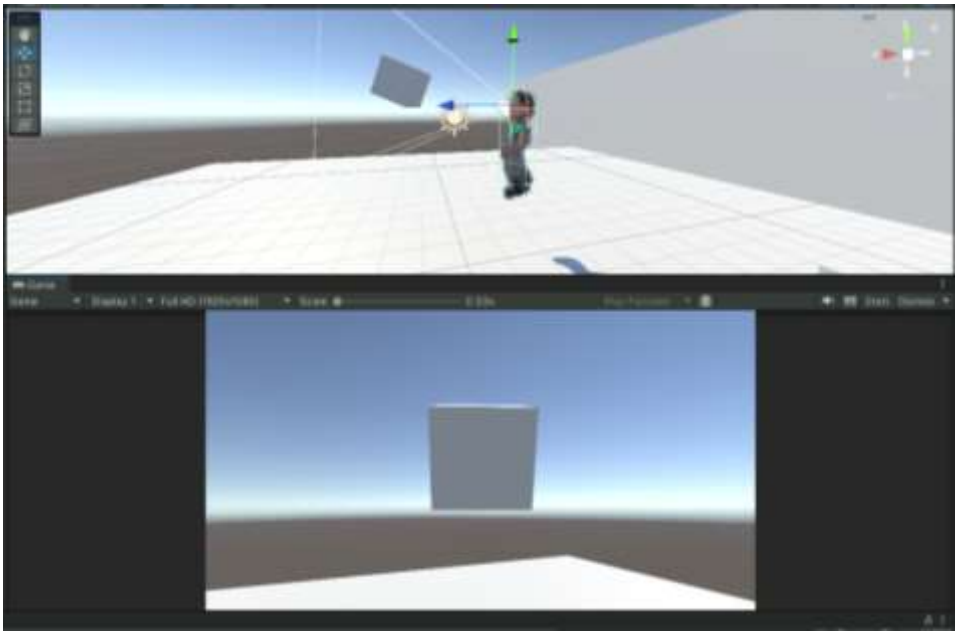


Рисунок 3.14 – Реалізація стрибка гравця

Налаштування компонента PersonController можуть бути змінені у інспекторі Unity задля більш гнучкої та швидкої розробки унікальних рівнів зі збільшеною або зменшеною гравітацією, можливим прискоренням гравця або збільшення сили стрибка (див. рис. 3.15).

Рисунок 3.15 – Налаштування компонента Person Controller

Контролер камери забезпечує плавне слідування за гравцем, забезпечуючи оптимальний огляд ігрового середовища. Код контролера камери міститься у файлі CameraController.cs і включає наступні основні функціональні можливості: Налаштування компонента, такі як чутливість миші, тіло гравця, контроль камери можуть бути змінені у інспекторі проекту (див. рис. 3.16).

Рисунок 3.16 – Налаштування компонента Camera Controller

Чутливість миші: Встановлення параметру чутливості миші дозволяє налаштувати швидкість повороту камери відповідно до рухів миші.

Ротація камери: Реалізовано можливість обертання камери навколо осі Y, що дозволяє гравцеві оглядати навколишнє середовище.

Блокування курсору: Блокування курсору при початку гри для забезпечення кращого контролю камери та уникнення випадкових рухів курсору за межами ігрового вікна.

3.3.2 Анімації гравця

Анімації є важливим елементом, що додає реалістичності рухам гравця. Використання Animator Controller в Unity дозволяє створювати складні анімаційні переходи між різними станами гравця, такими як ходьба, біг, стрибки та взаємодія з об'єктами.

Анімації та оснащування було виконано у Blender через якісний та зрозумілий інтерфейс. Зроблено такі анімації: стан спокою, стрибок, біг (див. рис. 3.17).

Рисунок 3.17 – Створення оснащування та анімація у Blender

Для експортованої в .fbx модель з анімаціями потрібно створити Animation Controller об’єкт та перенести анімації об’єкту гравця у вікно Animator, зв’язати їх та поставити умови при яких вони виконуються. Прикладом є якщо зафіксовано переміщення гравця булевий операнд isRunning повертає True, що завдяки зв’язкам у Animator включає анімацію бігу (див. рис. 3.18)..

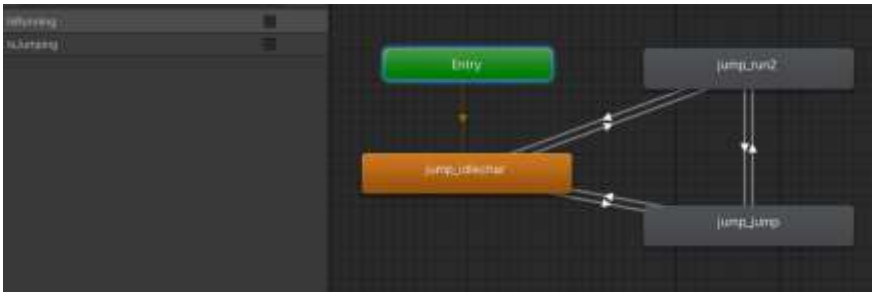


Рисунок 3.18 – Створення анімаційної схеми зв’язків для об’єкту гравця

3.3.3 Взаємодія з об'єктами

Взаємодія з об'єктами включає в себе розпізнавання та використання предметів в ігровому світі. Реалізована за допомогою скриптів, що визначають, коли гравець підходить до об'єкта та починає взаємодію (див. рис. 3.19).

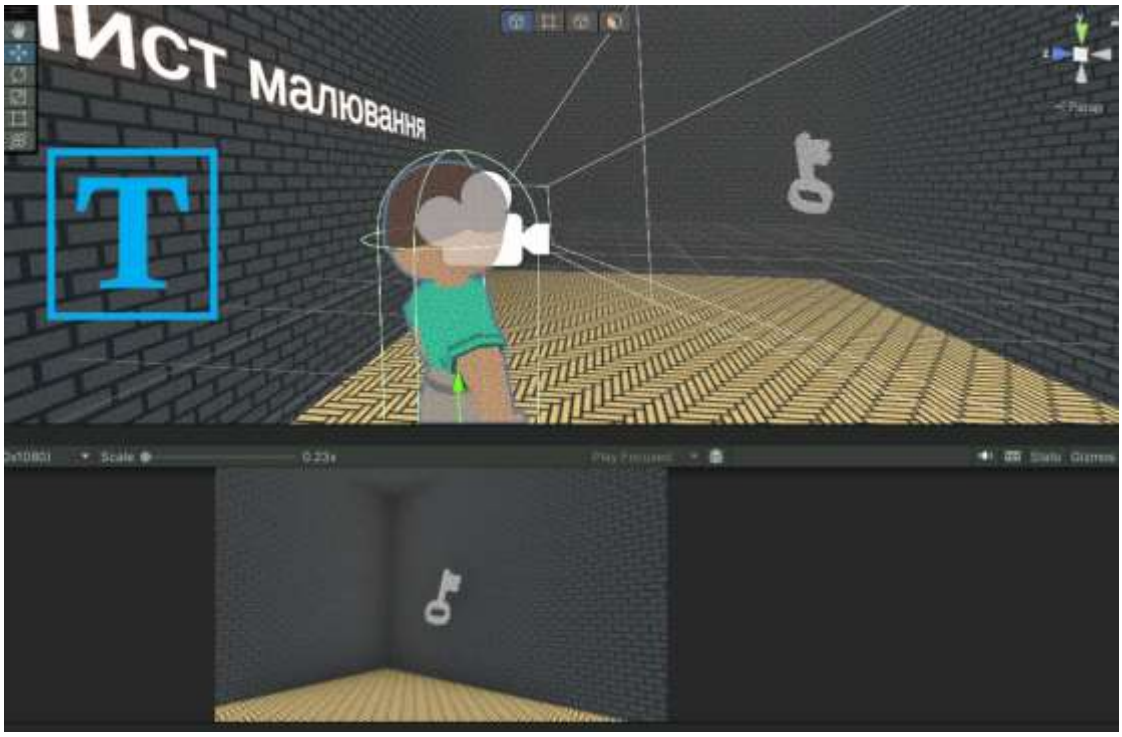


Рисунок 3.19 – Взаємодія гравця з об'єктами

Скрипт PlayerInteraction визначає чи об’єкт на який клікають має шар Interactable та після цього запускає логіку підняття об’єкту. Для більших налаштувань було також додано клас InteracitonItem (див. рис. 3.20).

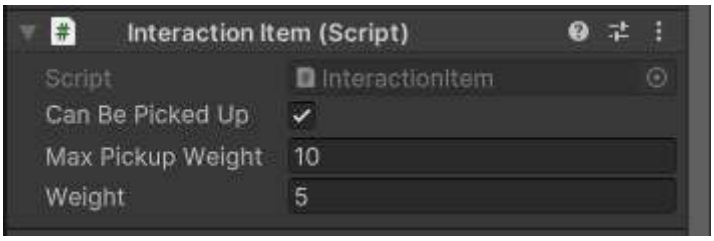


Рисунок 3.20 – Клас InteractionItem

В цьому класі можна налаштувати такі параметри:

Checkbox Can Be Picked Up – Параметр створений для того щоб в грі були не тільки предмети, які може підіймати гравець, а ще й тяжкі, що гравець рухає тільки собою.

Max Pickup Weight – Параметр для розрахунку ваги, щоб гравець не міг одночасно підіймати багато важких предметів.

Weight – Параметр особистої ваги об’єкту.

3.4 Створення механіки малювання та класифікації малюнків

Для створення цієї механіки малювання та спавну після класифікації потрібно мати конвертовану в .onnx модель та встановлену в проекті Unity бібліотеку Barracuda.

Конвертація моделі відбувається завдяки бібліотекам ONNX та tf2onnx (див. рис. 3.21) [15].

```
import tensorflow as tf
import tf2onnx
import os

model = tf.keras.models.load_model('testeddecent/my_model.h5')
saved_model_dir = "my_saved_model2"

tf.saved_model.save(model, saved_model_dir)

output_path = "model_5class.onnx"
os.system(f"python -m tf2onnx.convert --saved-model {saved_model_dir} --output {output_path} --opset 13")
print(f"ONNX model saved to {output_path}")
```

Рисунок 3.21 – Скрипт який дозволяє перетворити натреновану модель у .onnx

Далі модель формату .onnx було імпортовано у проект Unity та підключено.

3.4.1 Створення механіки малювання

Створення механіки що дозволяє малювати на текстурі було здійснено завдяки таким крокам:

1) Ініціалізація текстури та параметрів малювання
   - Створюємо нову текстуру Texture2D розміром 128x128 пікселів з форматом RGBA32.
   - Очищаємо текстуру, заповнюючи її білим кольором.
   - Призначаємо текстуру матеріалу об'єкта, до якого прикріплений цей скрипт.
2) Малювання за допомогою миші:
   - Визначаємо координати кліку миші.

- Перевіряємо, чи клік миші був на об'єкті з текстурою.
- Обчислюємо координати на текстурі та малюємо в цих координатах.

3) Функції для малювання та очищення текстури:

- Draw(int x, int y): малює чорним кольором на текстурі в заданих координатах з врахуванням розміру пензля.
- ClearTexture(): очищує текстуру, заповнюючи її білим кольором.
- GetDrawingTexture(): повертає текстуру малювання.

Створена механіки скрипту Drawing.Handler забезпечує основну функціональність малювання на текстурі в Unity (див. рис. 3.22).

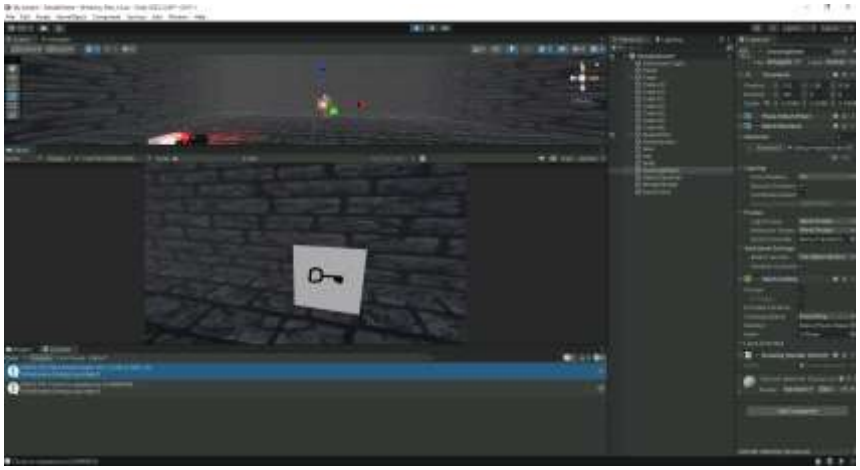
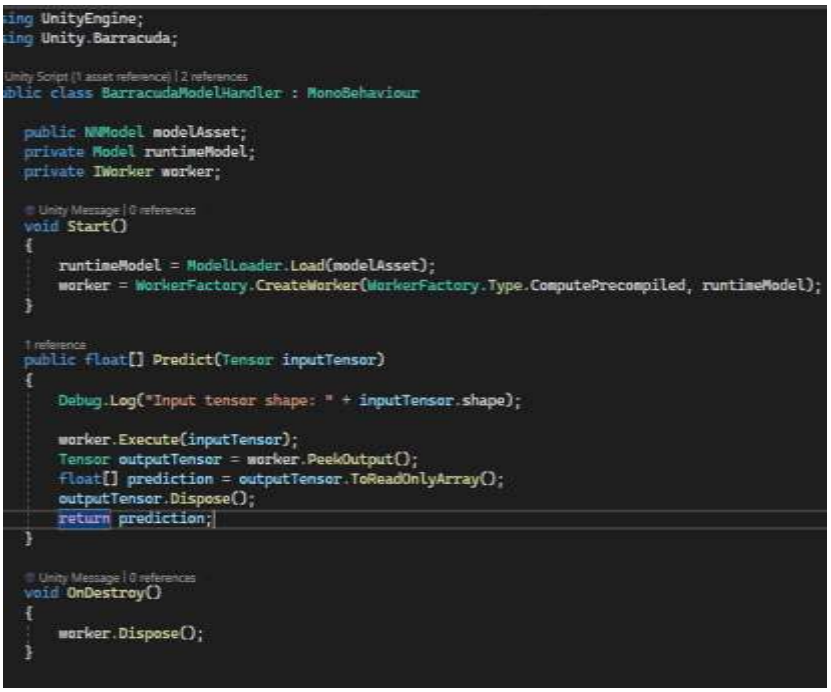


Рисунок 3.22 – Вигляд механіки малювання на ігровому папері

3.4.2 Створення механіки для класифікації зображення

Завдяки скрипту DrawingToTensor Намальоване іграком зображення конвертується з (1, 128, 128, 4(RGB)) у (1, 128, 128, 1(grayscale)), та тензор який приймає попередньо натренована модель.

Скрипт BarracudaModelHandler, який оброблює стан моделі та має публічний метод прогнозування класу намальованого об’єкту завдяки натренованій нейромережі (див. рис. 3.23).



```
ing UnityEngine;
ing Unity.Barracuda;

Unity Script (1 asset reference) | 2 references
blic class BarracudaModelHandler : MonoBehaviour

  public NNModel modelAsset;
  private Model runtimeModel;
  private IWorker worker;

  Unity Message | 0 references
  void Start()
  {
      runtimeModel = ModelLoader.Load(modelAsset);
      worker = WorkerFactory.CreateWorker(WorkerFactory.Type.ComputePrecompiled, runtimeModel);
  }

  1 reference
  public float[] Predict(Tensor inputTensor)
  {
      Debug.Log("Input tensor shape: " + inputTensor.shape);

      worker.Execute(inputTensor);
      Tensor outputTensor = worker.PeekOutput();
      float[] prediction = outputTensor.ToReadOnlyArray();
      outputTensor.Dispose();
      return prediction;
  }

  Unity Message | 0 references
  void OnDestroy()
  {
      worker.Dispose();
  }
```

Рисунок 3.23 – Скрипт BarracudaModelHandler

Створений тензор передається у скрипт ObjectSpawner, який завдяки публічному методу Predict функції BarracudaModelHandler створює перед гравцем об’єкт відповідного до прогнозування класу.

3.5 Створення меню

Меню є важливим елементом, що дозволяє гравцеві керувати грою, змінювати налаштування та отримувати доступ до різних функцій керування ігровим процесом. Воно є ключовим елементом взаємодії гравця з грою, забезпечуючи зручний та ефективний спосіб взаємодії з ігровим середовищем. Меню також є важливим інструментом для підтримки іммерсивності геймплею, оскільки воно дозволяє гравцеві швидко та зручно здійснювати різні дії, не виходячи з гри, і зосереджуватися на ігровому процесі.

3.5.1 Головне меню

Головне меню включає кнопку запуску гри, назву та задній фон у вигляді текстури, воно є відправною точкою у туторіал гри (див. рис. 3.24).

Рисунок 3.24 – Головне меню гри

Натискаючи на кнопку “Розпочати гру” гравець переміщується до головної сцени гри.

3.5.2 Меню паузи

Скрипт меню паузи має назву PauseMenuController, він забезпечує можливість ставити гру на паузу, перезапускати гру, переглядати налаштування управління та виходити з гри.

Меню паузи має три кнопки:

- Розпочати з початку – кнопка, яка відповідає за рестарт гри у разі, якщо гравець хоче перепройти гру або застряг у текстурах.
- Переглянути управління – кнопка, яка при натисканні відкриває незмінні налаштування керування та дає гравцю переглянути їх.
- Закрити гру – кнопка, яка закриває процес гри.

Якщо гравець натискає кнопку “Escape” він має можливість переглянуть ці кнопки (див. рис. 3.25).

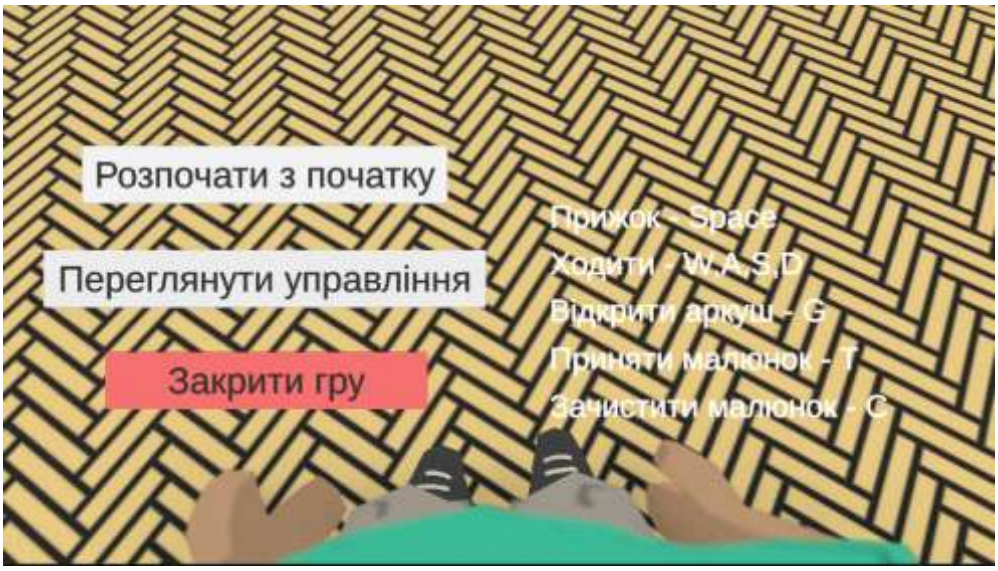


Рисунок 3.25 – Меню паузи гри

3.6 Постпроцесинг

Постпроцесинг є невід'ємною частиною сучасної розробки ігор, забезпечуючи покращення візуальної якості та загальної естетики гри.

Постпроцесинг — це етап обробки зображень, який виконується після основного рендерингу сцени. Цей процес включає в себе різні методи обробки, що дозволяють покращити візуальну якість гри, такі як згладжування країв, корекція кольорів, додавання ефектів глибини різкості, розмиття руху, та інші візуальні ефекти:

- Антиаліасинг: Технологія, що згладжує краї об'єктів, зменшуючи ефект "пилкоподібних" ліній. Основні методи включають FXAA, MSAA та TAA.
- Корекція кольорів: Включає в себе налаштування яскравості, контрасту, насиченості та інших параметрів для досягнення бажаної естетики гри.
- HDR (High Dynamic Range): Технологія, що дозволяє відтворювати ширший діапазон яскравості та контрасту, що робить зображення більш реалістичним.

– Ефекти глибини різкості (Depth of Field): Ці ефекти імітують фокусування камери, розмиваючи об'єкти, що знаходяться поза фокусом.

– Розмиття руху (Motion Blur): Додає ефект розмиття до об'єктів, що рухаються, створюючи відчуття швидкості та динаміки.

– Ефекти Bloom: Ефекти, що додають світіння навколо яскравих об'єктів, імітуючи рефлекси світла.

В ігровому додатку поспроцесинг був реалізований завдяки влаштованій функції камери в редакторі Unity під назвою Post Processing (див. рис. 3.26) [16].

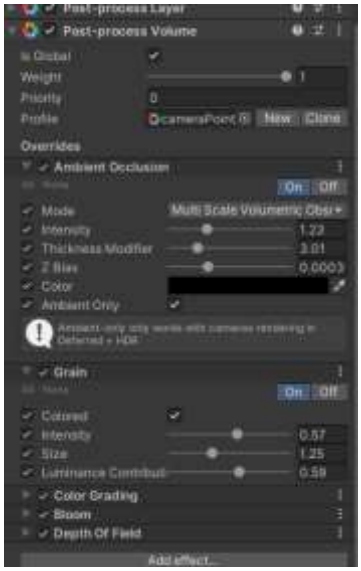


Рисунок 3.26 – Ефекти пост-процесингу

На скріншотах можна побачити працю таких фільтрів як Depth of Field, Grain, Color Grading, Ambient Occlusion та Bloom (див. рис. 3.27-3.28).

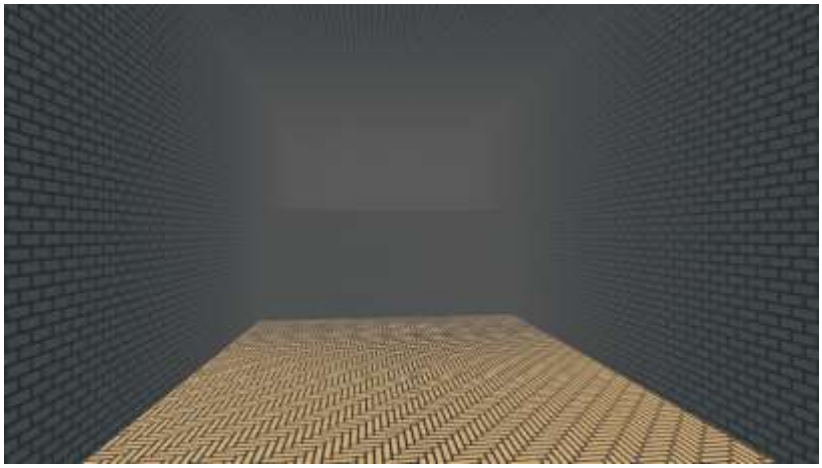


Рисунок 3.27 – Гра з вимкнутим постпроцесингом

Рисунок 3.28 – Гра з ввімкнутим постпроцесингом

3.7 Тестування та налагодження

Тестування розробленого ігрового додатку проводилося з метою перевірки його працездатності та відповідності вимогам. Для забезпечення високої якості та надійності додатку було застосовано ручне та модульне тестування.

Ручне тестування включало перевірку всіх можливих сценаріїв використання додатку з метою виявлення помилок та несправностей. Кожен елемент інтерфейсу та функціонал гри ретельно перевіряли на коректність роботи та відповідність очікуванням користувача.

Модульне тестування проводилося для перевірки окремих компонентів додатку на предмет їх правильної взаємодії між собою. Це дозволило виявити та виправити помилки на ранніх стадіях розробки, що значно підвищило загальну стабільність системи.

Крім того, проводилося інтеграційне тестування, яке забезпечувало належну взаємодію між різними модулями додатку. Це дало можливість переконатися, що всі компоненти працюють злагоджено і додаток функціонує як єдине ціле.

Після успішного завершення тестування, ігровий додаток був підготовлений до випуску. Завдяки комплексному підходу до розробки та

тестування, вдалося досягти високої якості продукту, який задовольняє потреби гравця та відповідає всім встановленим вимогам.

3.8 Публікація гри

Itch.io є популярною платформою для публікації незалежних ігор, що надає розробникам зручні інструменти для розповсюдження своїх проектів. Часто інді-розробники публікують свої ігри туди з метою заробити або отримати відгуки про їх програму або гру. В випадку кваліфікаційної роботи було створено особисту сторінку розробленої гри для отримання відгуків та покращення ігрового процесу.

Процес публікації гри на безоплатній основі:

Реєстрація та налаштування аккаунту: Було зареєстровано новий акаунт на itch.io.

Створення нової сторінки гри:

Назвою гри буде “Take Control”, відсилаючі гравця на те що він бере контроль над ігровим процесом у більшому.

Додані скріншоти, які відсилають на ігровий процес (див. рис. 3.29).

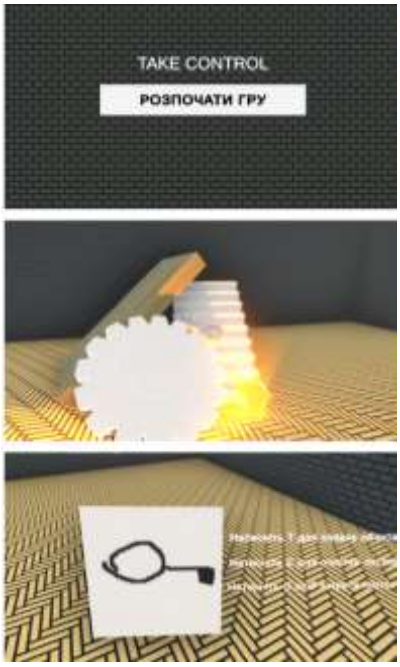


Рисунок 3.29 – Скриншоти з гри для ілюстрації ігрового процесу

Було підготовано поточну версію гри заархівовано його у відповідному форматі. Завантажено архів на платформу та опубліковано гру (див. рис. 3.30) [20].

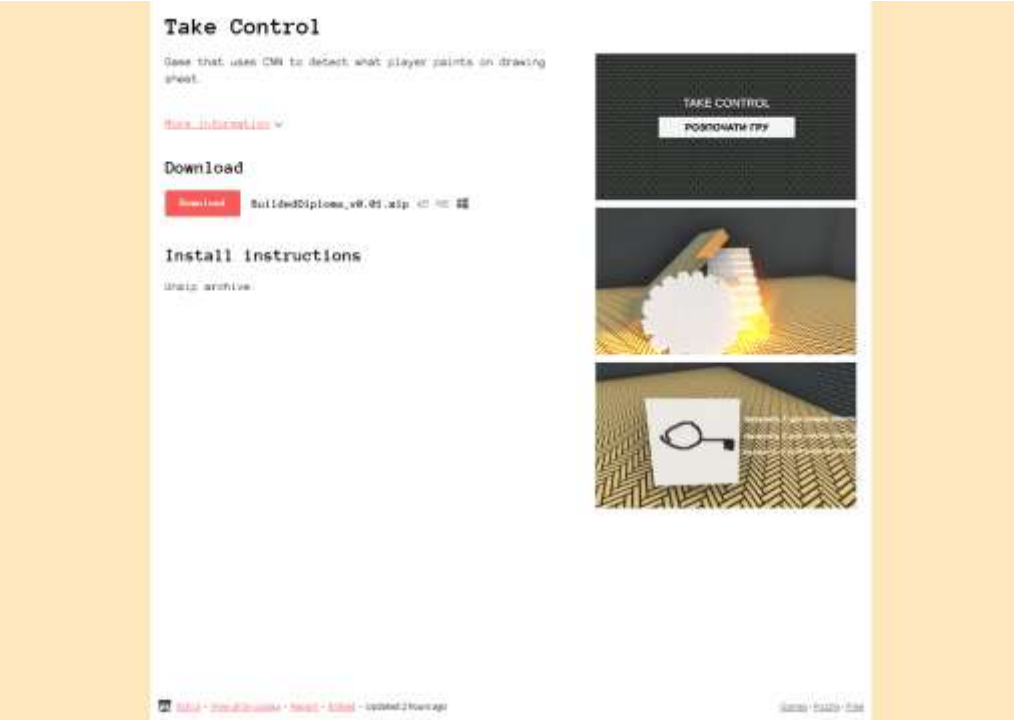


Рисунок 3.30 – Вигляд сторінки гри з боку користувача

Натискаючи на кнопку “Download” користувач завантажує архів з грою на свій персональний компьютер.

3.9 Висновки до третього розділу

У третьому розділі було практично реалізовано спроектований додаток з нейронною мережею для розпізнавання малюнків і створення об'єктів у грі. Для реалізації використовувалися сучасні технології та інструменти, такі як Unity і бібліотека Tensorflow для навчання нейронної мережі. Основні етапи роботи включали розробку та тренування нейронної мережі, створення графіки для гри, розробку механік гравця, реалізацію механіки малювання класифікації малюнків та публікації гри.

Використовувалися методи машинного навчання та нейромережевої обробки даних для розв’язання задач розпізнавання та класифікації об’єктів. Було зібрано та оброблено великий набір даних малюнків різних об'єктів. Нейронна мережа була налаштована і натренована для класифікації малюнків за допомогою методів обчислювального інтелекту, зокрема машинного навчання.

Методологія системного аналізу відігравала важливу роль у процесі розробки системи, охоплюючи аналіз вимог, проектування архітектури та реалізацію. Завдяки цьому було забезпечено системний підхід до вирішення завдання, що дозволило створити цілісну та функціональну систему. Інструментальні засоби, такі як Tensorflow, для створення та тренування згорткової нейронної мережі для задачі класифікації, рушій Unity для створення гнучкої, масштабованої системи.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено та впроваджено ігровий додаток з використанням нейронної мережі на рушії Unity. Основною метою проекту було створення унікального ігрового додатку з механікою, яка дозволяє розпізнавати малюнки гравців та створювати відповідні об'єкти у грі, забезпечуючи інтерактивність та гнучкість геймплею.

Розробка ігрового додатку включала кілька етапів. Було проведено аналіз існуючих ігрових додатків та вибрано найбільш підходящі технології для розробки. Аналіз включав вивчення різних підходів до створення ігрових механік, графіки та взаємодії з гравцем. Було розглянуто кілька варіантів рушіїв та бібліотек, проте вибір зупинився на Unity завдяки його широким можливостям та гнучкості. Основою для навчання нейронної мережі було обрано TensorFlow з високорівневим API Keras, що забезпечує зручність та гнучкість у створенні та налаштуванні моделі. Цей вибір був обґрунтований тим, що TensorFlow має потужні інструменти для навчання та оптимізації нейронних мереж, а Keras надає простий та зрозумілий інтерфейс для швидкого прототипування та експериментування з різними архітектурами мереж.

У процесі розробки ігрового додатку з використанням нейронної мережі на рушії Unity були визначені основні компоненти та модулі, які забезпечують його функціональність. Центральним елементом є нейронна мережа, що класифікує малюнки гравців і створює відповідні об'єкти у грі. Бібліотека Barracuda, відповідає за обробку малюнків та взаємодію з нейронною мережею у Unity. Графічна частина забезпечує зручний інтерфейс для взаємодії з грою, малювання, анімації та управління персонажем.

Тестування та налагодження додатку проводилося з використанням різних методів. Результати тестування підтвердили ефективність та коректність роботи додатку. Виявлені помилки були виправлені, а система вдосконалена на основі

отриманих результатів. Це забезпечило високу надійність, продуктивність та зручність використання додатку у реальних умовах експлуатації.

Таким чином, розроблений ігровий додаток з використанням нейронної мережі повністю відповідає поставленим вимогам та задачам. Він забезпечує ефективне розпізнавання малюнків, створення відповідних об'єктів у грі та інтерактивність геймплею, що значно підвищує залученість гравців та оптимізує процес розробки ігор. Інтерактивність додатку дозволяє гравцям активно впливати на ігровий процес вирішуючи головоломки з набором предметів, що робить гру більш захопливою та динамічною. Використання нейронної мережі додає новий рівень складності та інноваційності, дозволяючи автоматично розпізнавати та обробляти малюнки гравців у режимі реального часу. Це підвищує реіграбельність та стимулює гравців до експериментів та творчого підходу.

Подальші дослідження можуть зосереджуватися на розширенні функціональності та розміру гри для більшого часу гри, додані нові та перероблені старі механіки інтеракції з предметами. Додавання більшої кількості класів для нейронної мережі, покращення точності та проведення тестів на різних комп’ютерах дозволить підвищити гнучкість та масштабність гри.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 122 – «Комп’ютерні науки» і демонструє володіння такими компетентностями як:

- Здатність застосовувати знання основних форм і законів абстрактно-логічного мислення, основ методології наукового пізнання, форм і методів вилучення, аналізу, обробки та синтезу.
- Здатність проектувати та розробляти здатність проектувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагльненого, об’єктно-орієнтованого, функціонального, логічного, з відповідними моделя, методами й алгоритмами обчислень, структурами даних і механізмами управління.

- Здатність використовувати методи обчислювального інтелекту, машинного навчання, нейромережевої та нечіткої обробки даних, генетичного та еволюційного програмування для розв’язання задач розпізнавання, прогнозування, класифікації, ідентифікації об’єктів керування.

Серед результатів навчання, визначених стандартом кваліфікаційна робота реалізовує наступні:

- Розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв’язання задач в галузі комп’ютерних наук.
- Використовувати методи та засоби для вирішення завдань в галузі комп’ютерних наук.
- Використовувати інструментальні засоби розроби клієнт-серверних застосувань.
- Використовувати методи обчислювального інтелекту, машинного навчання, нейромережевої та нечіткої обробки даних для розв’язання задач класифікації та прогнозування об’єктів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Unity for Games. Unity. URL: https://unity.com/ (дата звернення 10.05.2024)
2. Scripting in Unity. Unity Learn. URL: https://learn.unity.com/course/create-with-code (дата звернення 10.05.2024)
3. Why Monument Valley is the Most Beautiful Game Ever. Shaunak Deo. URL: https://medium.com/@shaunakdeo/why-monument-valley-is-the-most-beautiful-game-ever-548794b7a4be (дата звернення 10.05.2024)
4. The Room. Fireproof games. URL: https://www.fireproofgames.com/games/the-room (дата звернення 12.05.2024)
5. Michael Thomsen. How Limbo came To Life. URL: https://www.ign.com/articles/2010/09/14/how-limbo-came-to-life (дата звернення 12.05.2024)
6. What is Unified Modeling Language. Lucidchart. URL: https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language (дата звернення 17.05.2024)
7. Christopher W. T. An architectural approach to level design : George Mason University, 2014, С. 103-195.
8. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning (Adaptive Computation and Machine Learning series), 2016, С. 326 - 356
9. Keras. Tensorflow. URL: https://www.tensorflow.org/guide/keras (дата звернення 17.05.2024)
10. Convolutional neural network (CNN). Lev Craig. URL: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network (дата звернення 17.05.2024)
11. Simon J.D. Prince. Understanding Deep Learning, 2024, С. 161-180.

12. Complete Guide to the Adam Optimization Algorithm. Builtin. URL: https://builtin.com/machine-learning/adam-optimization (дата звернення 21.05.2024)
13. Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers URL: https://arxiv.org/pdf/1909.08490 (дата звернення 21.05.2024)
14. Modeling manual. Blender. URL: https://docs.blender.org/manual/en/latest/modeling/index.html (дата звернення 21.05.2024)
15. Blender: UV Mapping – Simply Explained. All3DP. URL: https://all3dp.com/2/blender-uv-mapping-simply-explained/ (дата звернення 21.05.2024)
16. Material Maker. URL: https://rodzilla.itch.io/material-maker (дата звернення 21.05.2024)
17. CharacterController. Unity. URL: https://docs.unity3d.com/ScriptReference/CharacterController.html (дата звернення 21.05.2024)
18. Converting TensorFlow to ONNX. ONNX. URL: https://onnxruntime.ai/docs/tutorials/tf-get-started.html (дата звернення 24.05.2024)
19. Post-processing and full-screen effects. Unity. URL: https://docs.unity3d.com/Manual/PostProcessingOverview.html (дата звернення 24.05.2024)
20. Гра під назвою Take Control. URL: https://solaqq.itch.io/take-control (дата звернення 24.05.2024)