МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

Моделювання системи «гравець-компютерна гра» за допомогою методів машинного навчання

Кваліфікаційна робота(проєкт)
на здобуття ступеня вищої освіти магістр

Виконав: здобувач 2 курсу, 231М групи
Спеціальності 122 Комп'ютерні науки
Освітньо-професійної програми
Інформаційні системи та технології другого (магістерського) рівня вищої освіти
Шувалов Андрій Кирилович
Керівники: Доктор педагогічних наук, професор Шерман Михайло Ісаакович,
Кандидат фізико-математичних наук,
доцент Єрмолаєв Вадим Анатолійович
Рецензент: доцент кафедри менеджменту та інформаційних технологій Херсонський державний аграрно-економічний університет
Лобода Олена Миколаївна

Херсон – 2021

ЗМІСТ

ВСТУП……4

РОЗДІЛ 1. Теоретична частина ……7

1.1 Основні поняття штучного інтелекту……...7

1.2 Історія розвитку систем штучного інтелекту…….11

1.3 Різні способи побудови систем штучного інтелекту……17

РОЗДІЛ 2. Аналіз предметної області і огляд аналогів ……...23

2.1 Предметна область проєкту……..23

2.2 Аналіз аналогічних проєктів та існуючих рішень……..25

РОЗДІЛ 3. Аналіз вимог до програмної системи ……..31

3.1 Функціональні і нефункціональні вимоги……..31

3.2 Діаграма варіантів використання……32

РОЗДІЛ 4. Архітектура системи……..33

4.1 Діаграма компонентів……..33

4.2 Макети для користувача інтерфейсів…….36

РОЗДІЛ 5. Реалізація системи……..40

5.1 Реалізація компоненту збереження даних……..41

5.2 Реалізація компоненту звукового супроводу…….42

5.3 Реалізація компоненту ігрових меню……..43

5.4 Реалізація компоненту ігрового процесу……45

5.5 Реалізація компоненту ігрового поля……..46

5.6 Реалізація компоненту конструктора фігур……46

5.7 Реалізація компоненту даних про фігури……...48

5.8 Реалізація компоненту анімацій……..48

5.9 Шейдери……50

РОЗДІЛ 6.Тестування……54

6.1 Функціональне тестування……...54

6.2 Юзабіліті тестування……57

ВИСНОВОК……59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……60

ДОДАТКИ……63

Додаток А……...63

Додаток Б……67

Додаток В……...70

ВСТУП

Актуальність роботи. Аналіз поведінки людини в іграх є актуальною і складною у вирішенні проблемою. Під грою розуміється процес, в якому беруть участь дві або більше сторін, які ведуть боротьбу за реалізацію своїх інтересів. Вивчення таких ігор використовується для пошуку оптимальної стратегії при конфлікті інтересів. Економісти за допомогою теорії ігор зазвичай вивчають поведінку людини на вільному ринку. Вони проводять експерименти, які породжують великі обсяги інформації і потім аналізують ці дані за допомогою економічних моделей. Деякий час назад до аналізу цих даних підключилися фахівці з області машинного навчання.

Машинне навчання (Machine Learning) - це розділ прикладної математики, який об'єднує методи математичної статистики, оптимізації та штучного інтелекту. Машинне навчання досліджує методи побудови алгоритмів, здатних навчатися на підставі емпіричних даних[9].

Частиною машинного навчання є інтелектуальний аналіз даних - сукупність підходів, методів і алгоритмів, які використовуються для автоматичного отримання будь-яких нетривіальних висновків на основі емпіричних даних, в тому числі сигналів, зображень, текстів тощо.

Машинне навчання необхідно для автоматизації вирішення складних професійних завдань. В силу універсальності використовуваних алгоритмів машинне навчання може бути застосовано в самих різних областях діяльності. Наприклад, воно використовується для розпізнавання мови, жестів, рукописного тексту, в сфері медичної діагностики і біржового аналізу, для боротьби зі спамом, здійснення фінансового контролю і в багатьох інших сферах.

Сфера застосувань машинного навчання постійно розширюється, так як стрімка інформатизація суспільства призводить до накопичення величезних обсягів емпіричних даних в науці, виробництві, соціальних

областях. А виникачи при цьому завдання прогнозування, управління та прийняття рішень часто вимагають швидкого аналізу цих обсягів.[13]

В цьому випадку методи машинного навчання виявляються не тільки застосовними, але і часто більш ефективними в умовах, коли людський мозок не в силах проаналізувати величезний обсяг даних досить швидко. Величезні обчислювальні потужності сучасних комп'ютерів можуть надати людині в цьому істотну допомогу.

У даній роботі я вивчав можливість застосування машинного навчання для вивчення поведінки людей в іграх.

За своєю суттю підходи економістів і фахівців машинного навчання протилежні. Економісти на першому етапі дослідження будують модель економічної поведінки. Потім проводиться ігровий експеримент, який повинен підтвердити або спростувати побудовану ними модель. На останньому етапі проводиться аналіз отриманих в експерименті даних. Якщо дані повністю або майже повністю відповідають економічній моделі, то вона вважається дійсною і може бути використана як в аналогічних іграх, так і для прогнозування економічної поведінки в реальному житті. Це підхід «від моделі до даних».

Машинне навчання використовує підхід «від даних до моделі». Існує кілька досить широко відомих програм, за допомогою яких можна знаходити різні закономірності в даних, виводити правила, будувати дерева рішень і так далі. На підставі виявлених закономірностей будується модель поведінки людини в грі.

Мета роботи. Створення гри «Тетріс 3D» на мові програмування C#.

Для досягнення поставленої мети необхідно було виконати ряд **завдань**:

- Розглянути базові поняття ШІ
- Розглянути різноманітні підходи до побудови систем ШІ

- Провести аналіз предметної області і огляд аналогів
- Провести аналіз вимог до прогромної системи
- Реалізувати систему та провести тестування

Об’єкт дослідження - гра «Тетріс 3D» на мові програмування C#.

Предмет дослідження - моделювання системи «гравець-компютерна гра»

за допомогою методів машинного навчання.

Практичне значення одержаних результатів. Одержані результати дослідження дають змогу проаналізувати та побачити на практиці взаємодію в системі «гравець-комп’ютерна гра» на прикладі гри «Тетріс 3D» на мові програмування C#.

Робота складається зі вступу, 6 розділів, висновку, переліку використаних джерел та додатків.

РОЗДІЛ 1
Теоретична частина

1.1 Основні поняття штучного інтелекту

На сьогодні, максимальна ефективність роботи програмістів можлива лише у тому випадку, коли частину їх роботи будуть виконувати комп'ютерні програми, що функціонують на принципах штучного інтелекту. Саме він, за останній період часу, досяг найбільшого прогресу та став активно використовуватися найбільшими компаніями, пов'язаними з інформаційними технологіями. Головний принцип штучного інтелекту полягає не лише у виконанні ідентичних дій програмою, але й наявність у неї можливості отримувати знання, навчатися та розвиватися. Потрібно зауважити, що розвиток «штучного інтелекту» відкриває нові можливості не лише для компанії, що працюють в комп'ютерній галузі, а й для усього людства, загалом.[1]

Потрібно зауважити, що сама дефініція інтелект походить з латинської мови та у прямому перекладі має значення «розум». Тобто, під інтелектом потрібно розуміти здібності людини приймати рішення, в тому числі й нестандартні. Слідчо, штучний інтелект в комп'ютерних технологіях означає здатність системи виконувати дії, які притаманні людини. Наприклад, з безлічі варіантів підбирати той, який буде найбільш актуальним для вирішення проблеми.

В даній курсовій роботі під інтелектом мається на увазі можливість штучного інтелекту виконувати дії, що безпосередньо пов'язані з перетворенням знань та досвіду, що були отримані протягом розвитку програми. Цей розвиток відбувається шляхом запам'ятовування різного роду дій.

Відповідно до інформації, що наведена вище, сам термін «знання» пов’язаний не лише з інформацією, яка отримується мозком людини. Подібні знання є важливими для розвитку індивіда, але при цьому вони не є достатніми для рішення складних завдань .

Дуже важливо, щоб система знань мала змогу отримувати світову модель. Таким чином, усі об’єкти інтелектуального розвитку можуть не лише отримуватися системою та згодом запам’ятовуватися, але й перетворюватися на ті дані, які безпосередньо необхідні для вирішення певних завдань.

У цьому сенсі найбільш релевантним значенням є дефініція «інтелектуальне завдання». Завдяки йому можна чітко пояснити різницю між простими завданнями та тими, що потребують використання інтелекту. Також, потрібно зауважити, що в рамках інтелектуального завдання використовується таке поняття, як алгоритм. Тобто, це перелік етапів, завдяки яким людина, або система виконує певну дію. На сьогодні, алгоритм є однією з найважливіших складових сучасної кібернетики[6].

За допомогою алгоритмів можна чітко розуміти, яка дія у якому випадку повинна виконуватися для того, щоб людина або система отримала найбільш відповідні результати. Найперші алгоритми використовувалися ще у IX столітті. У площині кібернетики та інших математичних наук потрібно зауважувати, що при використанні алгоритмів, певна проблема уже вирішена.

Якщо аналізувати увесь процес розвитку людства, то потрібно зауважувати, що пошук алгоритмів є найбільш важливою задачею, яка є актуальною, протягом тисячоріч.

Щодо проблем усталених алгоритмів, на що вказує відомий експерт у галузі штучного інтелекту М. Мінський, надмірно приписувати їм такі загадкові атрибути, як «інтелект». Його можуть точно виконувати люди,

комп’ютери (правильно запрограмовані) або роботи, які не розуміють природи самого завдання.

Тільки особа, відповідальна за виконання завдання, здатна виконувати основні операції, що складають процес, а крім того, їй потрібно ретельно керуватися запропонованим алгоритмом. Такі люди, як говорять в цьому випадку, діють суто механічно і можуть ефективно досягати поставлених цілей щодо подібних проблем. Відповідно до цього, можна сказати, що вони не відносяться до категорії, пов’язаної з інтелектуальними завданнями. Подібні завдання, як правило, мають досить прості та традиційні, з точки зору математики, рішення. Яскравим прикладом подібних задач є ті, що безпосередньо пов’язані з обчисленням. Для їх вирішення можна використовувати традиційні математичні алгоритми, які використовуються певною математичною послідовністю[2].

Окремо потрібно розглянути процес створення алгоритмів щодо вирішення різного роду інтелектуальних задач, до яких потрібно віднести, наприклад, доведення математичних законів, або створення програми, пов’язаною з грою в шашки. Сама побудова алгоритму, у цьому випадку, є складним завданням. Навіть, у тому випадку, коли підсумкове рішення буде простим та примитивним.

Згідно з цим, поняття інтелекту можна пов’язувати з супер алгоритмом, який може створювати алгоритми для вирішення конкретних проблем.

Інший цікавий коментар, згідно з нашим визначенням, кар’єра програмістів одна з найрозумніших, так як кінцевий результат діяльності розробника є програма - алгоритм у чистому вигляді. Тобто, навіть створення частин штучного інтелекту має значно покращити ефективність його роботи.

Саме функціонування мозку безпосередньо пов’язане з вирішенням різного роду інтелектуальних проблем,. Інтелект людини безпосередньо пов’язаний з логічним аналізом, розпізнаванням ситуації, плануванням поведінки, іграми та управлінням у невизначених умовах. Інтелектуальні характеристики, виявлені в процесі вирішення проблем - це здатність вчитися, узагальнювати, здобувати досвід (знання та вміння) та адаптуватися до мінливих умов у процесі вирішення проблем.

Завдяки цим якостям інтелекту мозок може вирішувати різні проблеми і легко може реконструювати одне завдання в інше. Тому, мозкова діяльність безпосередньо пов’язана з вирішенням проблем (включаючи неформальні) стандартного, заздалегідь відомого рішення цих проблем немає.

В контексті цього питання можна розглянути теорію, що виклав відомий науковець А.Тьюрінг. Він розглядав процес взаємодії комп’ютерів та людей у наступному вигляді:

Вони знаходяться у різних кімнатах та у будь-якому випадку не можуть зустрітися. При цьому, між представниками цих кімнат відбувається обмін інформацією, наприклад, за допомоги повідомлень у чатах. У випадку, якщо у процесі спілкування людина не може визначитися з тим, що його співрозмовник є комп’ютером, то такий штучний інтелект можна вважати розумним.

Якщо аналізувати цей процес в контексті майбутнього розвитку інформаційних технологій, то подібний процес побудови алгоритмів будуть використовувати майже усі системи, побудовані на базі штучного інтелекту. Пов’язано це з тим, що на сьогодні, додати усі людські знання в одну глобальну систему майже неможливо. Також, лише подібними чином будуть проявлятися такі ознаки діяльності людини, як отримання знань та досвіду, а також її адаптація під навколишню середу[5].

1.2 Історія розвитку систем штучного інтелекту

Традиційно, визначають три найбільш актуальні шляхи створення та розвитку штучного інтелекту.

Склалося так, що історично моделювання штучного інтелекту має три основні напрямки.

Об’єктом дослідження методу номер один є будова та механізм людського мозку, кінцевою й ж метою є розкриття таємниці мислення. Найбільш важливою складовою аналізу в даному напрямку є проектування та розробка моделей, якій будуть безпосередньо пов’язані з даними психофізіологічного типу. Над зазначеними моделями в рамках досліджень проводяться різного роду досліди, головним завданням яких є пошук найбільш продуктивних шляхів розвитку цього напрямку.

У другому методі об’єктом дослідження є безпосередньо штучний інтелект, та його головні складові. В рамках цього методу розглядається проектування та створення моделей інтелектуальної діяльності людини, за допомоги сучасних комп’ютерів. Головною метою подібних досліджень є розробка алгоритмів, за допомоги яких комп’ютерний штучний інтелект зможе виконувати різного роду інтелектуальні дії для вирішення наявних проблем. Тобто, на дистанції розглядається можливість вирішення побутових та глобальних проблем, без безпосередньої участі в них людей.

Третім методом є створення штучного інтелекту, який буде пов’язувати в собі складові машини та людини. Тобто, якщо казати глобально, то це симбіоз людського та комп’ютерного інтелекту, який за своїм потенціалом буде випереджувати обидва. Головним питанням подібних досліджень є розробка алгоритмів діалогу у форматі комп’ютер – людина.

Першою з проблем, яку почали вирішувати за допомогою різного роду штучних інтелектів є примітивні комп'ютерні ігри, в рамках яких комп'ютер виконує різні алгоритми реакції на дії людини. Звісно, на той час, коли була розроблена ця програма, алгоритми штучного інтелекту були далекі від сучасних, але вже тоді подібний інформаційний застосунок мав змогу не лише розуміти правила, але й оброблювати їх під ситуацію, що мала місце під час гри. Система чітко розуміла, які дії їй вигідно виконувати, а які можуть призвести до поразки. Крім цього, швидкість прийняття рішень у неї нижче ніж, у звичайної людини, а тому через час вона вже має змогу на рівних грати з нею[11].

Сучасні машини, що функціонують на базі систем штучного інтелекту, використовують в рамках аналізу велику кількість критеріїв. Після ходу супротивника, вона досліджує усі можливі індекси ефективності та обирає для себе найбільш ефективний та дієвий.

Потрібно зауважити, що такий вид автоматизації не завжди дає ефективний результат, але при цьому він підтверджує можливість програми самостійно обирати свій майбутній шаг. Як результат, машина, що грає протягом певного відрізку часу, буде регулярно покращувати свої навички, за допомоги розуміння тих помилок, що мали місце у минулі рази. Подібне навчання, якщо казати глобально, є коригуванням помилок, що призвели до поразки минулий раз.

Звісно, з часом алгоритми пошуку в машинах, що функціонують на базі штучних інтелектів покращуються. Наприклад, якщо кількість дії, що мали змогу обчислити традиційні для 1974 року машини, не перевищувала декілька за секунду, то вже у 21 столітті, в матчі між штучним інтелектом та Каспаровим, їх кількість збільшилася до 100 мільйонів. Це було можливо за рахунок використання 256 процесорів, кожен з яких мав 4 ГБ дискової пам'яті.

На сьогодні, з кожним днем кількість комп’ютерів, що мають змогу повноцінно приймати участь у військових іграх, збільшується. Найбільш важливу роль у цьому процесі відіграє можливість адаптації знань та навичок, що були отримані за минулі партії. Найбільш важливим є можливість розпізнавання різних образів, та їх подальша адаптація під ситуацію, що має місце в грі.

Також, потрібно зауважити, що у розробках сучасних машин, функціонуючих на базі штучного інтелекту, приймають участь представники різних наук. Це дає змогу створити найбільш адаптовану систему, яка зможе приймати продуктивні рішення у будь-яких завданнях.

Як вже зазначалося вище, Ф.Розенблатт у середині 20 століття розробив і запропонував спільноті так званий перцептрон. Поява подібної машини змінила галузь досліджень, пов’язаних з розпізнаванням різних об’єктів.

Треба зауважити, що усі програми, що займаються розпізнаванням об’єктів, працює на базі двох режимів, а саме навчання та адаптації. У першому випадку, людина, яка виконує роль вчителя, представляє об’єкт машині та повідомляє кожному об’єкту, до якого поняття (класу) він належить. На основі цих даних встановлюється вирішальне правило, яке насправді є офіційним описом поняття. У режимі розпізнавання машина представляє нові об’єкти (зазвичай відрізняються від тих, що були представлені раніше), і, якщо є така можливість, то вона має класифікувати їх абсолютно достовірно[14].

Проблематика розрізнення щільно пов'язана з рештою інтелектуальних завдань - проблематикою перекладу з однієї мови на іншу та проблемою навчання мовних машин. Завдяки більш офіційній обробці та класифікації базових граматичних порядків та словникових

навичок можуть бути створені цілком задовільні алгоритми перекладу, такі як наукові чи комерційні тексти.

Для деяких мов подібні системи були створені наприкінці 1960-х років. Але для того, щоб послідовно перекласти досить об'ємний розмовний текст, потрібно зрозуміти його значення. Робота над проектом такого типу тривала вже тривалий час, але це ще далеко від повного успіху. Існують також програми, які забезпечують розмову між людьми та машинами усіченою природною мовою.

Щодо моделювання логічного мислення, то тут хорошим модельним завданням може стати автоматизація доведення теорем. З 1960 року було розроблено багато програм, які можуть знайти докази теорем у багатьох предикатах першого порядку. За словами американського експерта зі штучного інтелекту Дж. Маккетті, ці програми мають «здоровий глузд», тобто здатність робити цілком логічні висновки.

У програмі К. Гріна та інших для реалізації системи запитів знання записуються мовою логіки предикатів у вигляді набору аксіом, а питання, поставлені машиною, виражаються як теми теорем. Великий інтерес викликав проект «інтелекту» американського математика Хао Ванга. Протягом 3 хвилин після роботи IBM-704 програма виводить 220 порівняно простих лем і теорем з основних математичних монографій, а потім генерує 130 більш складних доказів теорем за 8,5 хвилин, деякі з яких ще не отримані математиками.

Однак до сьогоднішнього дня не існувало жодної програми для виведення або доведення єдиної теореми, тобто так звана «обмежена» потрібна математикам і є абсолютно новою.

Дуже великим напрямком систем штучного інтелекту є робототехніка. Яка головна відмінність між роботом та загальною комп'ютерною інформацією?

Щоб відповісти на це питання, можемо згадати великого російського фізіолога І. М. Сеченова: «... Різні зовнішні прояви діяльності мозку в кінцевому підсумку приписуються явищу - руху м’язів» при активному спілкуванні із зовнішнім світом за допомогою рухів.

Таким же чином складові одиниці інтелекту робота в основному використовуються для формування його планомірних рухів. Водночас основною метою Pure Computer System ШІ є вирішення абстрактних або допоміжних інтелектуальних проблем, зазвичай це не має нічого спільного з сприйняттям навколишнього середовища за допомогою штучних органів чуття або рухів.

Важко сказати, що перший робот був розумним. Лише у 1960-х роках з’явилися розумні роботи, керовані комп’ютерами загального призначення. Наприклад, у 1969 році в Лабораторії електротехніки (Японія) було розпочато розробку проекту «Промисловий інтелектуальний робот». Метою цієї розробки є створення розумного робота-маніпулятора з елементами штучного інтелекту для виконання монтажних робіт за допомогою візуального контролю.

Робот-маніпулятор має шість ступенів свободи. Він керується мікрокомп'ютером NEAC-3100 (оперативна пам'ять 32 000 слів, зовнішнє сховище диска 273 000 слів) для формування необхідного програмного руху, яке контролюється електрогідравлічною системою, маніпулятор оснащений тактильним датчиком.

Дві телевізійні камери, обладнані червоно-зелено-синіми фільтрами для ідентифікації кольору об'єкта, використовуються як система зорового сприйняття. Поле зору телевізійної камери поділено на 64*64 осередки. В результаті обробки отриманої інформації приблизно визначається площа, яку займає об'єкт, що представляє інтерес. Крім того, щоб детально вивчити предмет, область розпізнавання знову поділяється на 4096

клітинок. Якщо об’єкт не помістити у вибране «вікно», він автоматично переміститься, ніби людина ковзає поглядом на об’єкт.

Роботи в електротехнічній лабораторії можуть розпізнавати прості предмети, оточені плоскими та циліндричними поверхнями під спеціальним освітленням. Ціна цього експериментального зразка складала майже 400 000 доларів США.

З плином часу особливості роботів суттєво покращувалися, але вони все ж далекі від якостей людини, проте слід відмітити, що окремі операції досягли рівня найкращих жонглерів. Як приклад, вони здатні втимувати кульку для пінг-понгу не лезі.

Можливо, тут можна наголосити на роботі Київського інституту кібернетики: керініками якого були М. М. Амосов та В. М. Глушков (покійні на сьогоднішній день) проводиться група досліджень, котрі були спрямовані на розробку та аналіз інтелектуальних складових одиниць роботів. Ці дослідження особливо стосуються розпізнавання зображень та мовлення, логічних міркувань (автоматичного доведення теорем) та використання нейронних мереж для управління.

Наприклад, ми можемо розглядати модель Транспортного автономного інтегрованого робота (TAIR), створену в 1970-х роках. Конструктивно TAIR-це триколісне шасі, на якому встановлена система датчиків та блок управління. Система датчиків включає містить наступні методи зондування: оптичний далекомір, навігаційну систему з двома маяками та компасом, контактний датчик, датчик кута візка, таймер тощо. TAIR відрізняється від більшості інших систем в країні та за кордоном тим, що не включає комп’ютери у такій формі, до якої ми звикли.

Ядром системи управління є повітряна нейронна мережа, яка виконує різні алгоритми для обробки сенсорної інформації, планування поведінки та управління рухом робота.

Наприкінці цього дуже короткого огляду розглянемо приклад великої експертної системи.

MICIN – представляє собою експертну систему діагностичної медицини. Розроблено групою Стенфордського університету. Система надає можливість встановлювати відповідний діагноз на основі наявної симптоматики, які їй представлені, і призначає курс лікування будь-якої діагностованої інфекції. База даних складається з 450 правил.

PUFF – представляє собою аналіз респіраторних захворювань. Система являє собою MICIN, з якого видаляються дані про інфекції та вставляються дані про захворювання легень.

DENDRAL - визначає хімічні структури. Ця система є найстарішою і має звання експерта. Перша версія системи з'явилася в тому ж Стенфордському університеті в 1965 році. Користувач надає деяку інформацію про речовину та спектральні дані (інфрачервоний, ядерно-магнітний резонанс та мас-спектрометрію) до системи DENDRAL, а потім дає діагноз у вигляді відповідної хімічної структури.

PROSPECTOR - це експертна система, призначена для полегшення пошуку комерційно вигідних родовищ.

1.3 Різні способи побудови систем штучного інтелекту

На сьогодні, є різноманітні варіанти щодо створення систем штучного інтелекту. Потрібно зазначити, що такий поділ не є історичним. Це пов’язано з тим, що не дивлячись на те, що одна думка щодо певного питання змінює іншу, різні методи залишаються актуальними й надалі. Крім того, враховуючи, що немає ідеальної системи штучного інтелекту, а тому неможливо стверджувати, який метод є найбільш ефективним та продуктивним.

Для початку аналізу, потрібно розглянути логічний підхід. Чому він актуальний у системах штучного інтелекту та взагалі, як з'являється у подібних системах. Подібне питання з'являється у зв'язку з тим, що людський розум пов'язаний не лише з логічними винаходами. При цьому, якщо порівнювати людини та тварин, то головна різниця пов'язана саме зі здатністю логічного мислення.

Ключовою особливістю зазначеного методу є булева алгебра. Вона використовуються у всіх рішеннях програмістів, що займаються розробкою систем штучного інтелекту. Для початку, це різні логічні оператори, які використовуються щодо рішень завдань, пов'язаних з побудовою подібних систем. Надалі, розвиток булевої алгебри безпосередньо пов'язаний з численнями предикатів. Якщо аналізувати це питання більш глобально, то кожна система штучного інтелекту пов'язана з різними логічними принципами.

Дані, що використовуються для побудови системи штучного інтелекту зберігається у БД. В її рамках вони виглядають, як аксіоми. При цьому, усі результати та висновки з їх аналізу пов'язують між ними. Будь-який штучний інтелект має блок, який використовується для генерації цілей. Система висновку, у свою чергу, аналізує мету та намагається довести її, як теорему. У випадку, якщо мета доводиться такою системою, то система активує ряд дій, які необхідні для досягнення поставленої мети. Звідси, потрібно зауважувати, що можливості подібної системи безпосередньо пов'язані з генератором, а також блоками, що відповідають за доведення теореми.

Необхідно зауважити, що булевої алгебри та її можливостей не завжди достатньо для того, щоб якісно реалізувати штучний інтелект. При цьому, у розрахунках необхідно не забувати про те, що головною апаратною частиною всіх персональних комп'ютерів є одиниця пам'яті, яка приймає значення 1 або 0. Тобто, згідно з цією інформацією, можна

стверджувати, що результати комп'ютерної діяльності безпосередньо пов'язані з логікою предикатів.

Також, відповідно до формування систем штучного інтелекту, потрібно зазначити відносно новий напрямок, а саме нечітку логіку. Вона може надавати більш виразний результат, якщо використовувати її в логічних методах. Ключова різниця нечіткої логіки та булевої алгебри пов'язана з тим, що в ній можуть використовуватися не лише значення 1 або 0, а також й проміжні значення. Наприклад, якщо 0 використовують у якості «так», а 1 у якості «ні», то в рамках нечіткої алгебри до них може додаватися додаткове значення, наприклад, 0.5 у якості «я не знаю».

Левова частина сучасних логічних методів доволі об'ємні. Це пов'язано з тим, що протягом пошуку, як правило, використовується повний варіант. Згідно з цим, зазначений метод потребує більш продуктивної реалізації процесів обчислення, якщо казати про базу даних маленького розміру.

Якщо казати про структуровані методи, необхідно аналізувати способи створення систем штучного інтелекту, пов'язані з відтворенням, аналізом та моделюванням структуру мозку людини. Першою людиною, яка отримала реальні результати у цьому напрямку, був Френк Розенблатт. Протягом своєї наукової кар'єри він працював над створенням відповідного пристрою, який отримав назву перцептрон. Головною основою цього пристрою є нейрон. Звісно, що з часом цей пристрій розвивався та доповнювався, а на сучасному етапі розвитку інформаційних технологій, подібні системи отримали назву «нейронні мережи»[9].

Відмінністю цих моделей є структура окремих нейронів, топологія зв'язку між ними та алгоритм навчання. Найвідоміші варіанти надзвичайних ситуацій - це зворотне поширення помилок, мережа Хопфілда та випадкова нейронна мережа.

НМ найбільш успішно використовується для розпізнавання зображень, включаючи дуже галасливі завдання, але є також приклади його успішного застосування у створенні систем ШІ, яким є вищезгаданий TAIR.

Модель, що пов’язана з мотивацією мозку людини, має головну характеристику, а саме досить невелику виразну силу, алгоритм якої досить легко дослідити та вивчити. Ключовою характеристикою подібних мереж є особливість, що максимально відтворює складові людського мозку. Необхідно зауважити, що сучасні нейронні системи можуть ефективно функціонувати навіть у випадку, якщо інформація про середовище, в якому вони функціонують буде неповною. Тобто, у цьому випадку, вони максимально схожі на людей, бо можуть надавати результати власних дослідженнях не лише у виразах «так» чи «ні».

На сьогодні, досить популярними стають еволюційні методи. Використовуючи їх в рамках проектування систем ШІ, головна проблема пов’язана з особливостями створення стартової моделі та правил, які необхідно використовувати в її рамках. Тобто, така модель може не тільки еволюціонувати, але містити в собі декілька методів.

Зазначені особливості є безпосередньо адаптацією процесів, що виконуються розробником в контексті створенням моделі. Вона, у свою чергу, не отримує оновлену інформацію щодо середовища, в якому знаходиться. При цьому, така модель штучного інтелекту сама по собі стає частиною такого середовища.

Також, до найбільш популярних методів побудови систем, працюючих на базі ШІ необхідно віднести моделювання. Це найбільш класичний метод в кібернетиці. Його складовою, відомою усім розробникам є так звана «чорна скринька». Під цим пристроєм потрібно розуміти модуль з набором даних, які потрібні для функціонування системи. Уся інформація щодо внутрішньої структури такого пристрою

повністю відсутня. При цьому, розробник має розуміння про те, які вхідні та вихідні дані така система використовує. Що це за модель та як вона функціонує, для розробника ця інформація не є ключовою. Це пов’язано з тим, що подібна модель буде виконувати ідентичні дії в ідентичних ситуаціях[9].

Відштовхуючись від інформації, що наведена вище, імітація в подібних системах відіграє можливості відтворювати дії, що в будь-який момент виконали інші. Тобто, у перші дні функціонування штучного інтелекту подібний спосіб створення економить системі багато часу.

Ключовим та найбільш важливим недоліком є досить мала інформаційна ємність в моделях, які створюються з його допомогою.

У якості прикладу можна навести ситуацію, в рамках якої за діями людини, протягом великого проміжку часу, буде стежити подібний пристрій. Вона аналізує кожен крок, а людина у цьому випадку відіграє роль чорної скриньки. Тобто, протягом часу навчання, такий пристрій може відображати всю інформацію, що надходить до нього з боку людини, за якою він спостерігає. Зокрема, це смак, слух та зір.

Через певний час такий пристрій буде намагатися відображати ідентичні до людини дані. Вона активує цей процес відповідно до тих сигналів що отримує від людини. У випадку, якщо подібні дії будуть ефективно реалізовані, то для спостерігачів за такою моделлю, вона буде такою ж людиною, яка й та за якою вона спостерігала.

Також, подібні моделі можуть використовуватися для того, щоб людина мала можливість створити для себе брата близнюка, який буде мати ідентичні «думки».

Відповідно до філософії цього курсу, свідомість є відносно невеликою надбудовою над нашою підсвідомістю. Вона відстежує діяльність певних центрів мозку, таких як центр мови та остаточна обробка візуальних образів, а потім «повертає» ці образи до початкового

етапу обробки інформації. Водночас ці зображення переробляються, ми, здається, здатні бачити і чути думки мозку.

Водночас, коли ми «активно» беремо участь у цьому процесі, можна уявити та змоделювати навколишню дійсність. Це процес, за допомогою якого ми спостерігаємо за цими кількома центральними діями, які ми називаємо свідомістю. Якщо ми «бачимо» і «чуємо» свої думки, ми свідомі; якщо ні, то ми несвідомі.

РОЗДІЛ 2
Аналіз предметної області і огляд аналогів

2.1 Предметна область проєкту

«Тетріс 3D» буде базуватися на всім відомій грі «Тетріс», винайденої Олексієм Пажитновим і випущеної 6 червня 1984 року [28].

Правила гри «Тетріс»

Розмір ігрового поля 10 клітин в ширину і 20 у висоту. Над ігровим полем з'являється фігура тетраміно (геометрична фігура, що складається з чотирьох квадратів, з'єднаних сторонами [27]). Через певні проміжки часу фігура зсувається вниз на одну клітку. Поки що фігура падає, гравець може переміщати її вліво або вправо, обертати по або проти годинникової стрілки, а також прискорювати її падіння. Фігура буде падати до тих пір, поки не наткнеться на іншу фігуру, або на нижню границю ігрового поля. При заповненні горизонтального ряду цей ряд знищується, а все, що знаходиться вище, зсувається на одну клітку вниз. Гра закінчується, якщо фігура, що впала виходить за межі верхньої границі. Гравець отримує очки за кожен заповнений ряд, тому його задача - заповнювати ряди, не заповнюючи саме поле (по вертикалі) якомога довше.

Даний проект є 3D аналогом гри «Тетріс». Нижче наведено список відмінностей від класичного тетрису.

1. Ігрове поле є паралелепіпедом розміром 9x9x15 кубів.

2. Гравець може переміщати камеру навколо ігрового поля, щоб розглядати його під різними кутами.

3. Гравець може наближати і віддаляти камеру.

4. Фігури складаються з кубів, причому кількість кубів може бути різною, а не тільки рівною 4, як в оригінальному тетрісі.

5. Фігуру можна переміщати по горизонтальній площині у всіх

напрямках.

6. Фігуру можна повертати на 90 ° щодо будь-якої з 3-х осей.

7. Видалення кубів з ігрового поля буде відбуватися не при заповненні горизонтального ряду, а при заповненні всієї горизонтальної площини.

«Тетріс 3D», як і його класична версія, буде поєднувати в собі жанри «Головоломка» і «Аркада».

Жанр «Головоломка»

Головоломка - це жанр комп'ютерних ігор, суть яких полягає в рішенні логічних задач, що вимагають від гравця задіяння логіки, стратегії або інтуїції [3]. Ігри цього жанру, як правило, відрізняються своєю низькою вимогливістю до комп'ютерних ресурсів, у зв'язку з чим, їх можна легко

адаптувати під різні платформи, в тому числі мобільні пристрої або браузерні ігри.

Жанр «Аркада»

Аркада - жанр комп'ютерних ігор, що характеризується коротким за часом, але інтенсивним ігровим процесом [7]. Нижче наведено список основних властивостей жанру.

1. Гра відбувається на одному екрані. Тобто, гравець завжди може бачити все ігрове поле.

2. Нескінченна гра. Гра триватиме до тих пір, поки гравець не програє. Для цього складність гри поступово підвищується.

3. Ігровий рахунок. Гравець отримує очки за виконання певних задач. Мета гри, набрати якомога більше очок.

4. Швидке навчання і простий ігровий процес.

5. Відсутність сюжету / історії.

2.2 Аналіз аналогічних проєктів та існуючих рішень

В ігровій індустрії не рідкісні випадки, коли при розробці нової комп'ютерної гри використовуються ігрові механіки з уже існуючих ігор. Одні розробники додають старі ігри в свої проєкти під видом міні-ігор. Інші - повністю відтворюють ігрові механіки прототипу, але доповнюють його новими механіками, які роблять гру цікавіше для споживача. «Тетріс 3D» якраз є другим випадком. Ідею перенести класичний тетріс в 3D я взяв для того, щоб звести творчий процес до мінімуму і більше сконцентруватися на написанні коду та інших технічних аспектах гри.

Нижче представлений аналіз 3-х ігор, які також базуються на ігрових механіках тетрису: «Block Out», «Tricky Towers» і «Tetris 99». Block Out «Block Out» - гра, що вийшла в 1989 році [10]. Являє собою 3D тетріс з видом зверху. Правила гри такі ж, як і в звичайному тетрісі. Єдина відмінність - гравець може обертати фігури щодо всіх 3-х осей. Оскільки камера розташована не збоку від ігрового поля, а над ним, щоб було простіше орієнтуватися, кожен шар позначається своїм кольором, а падаюча фігура є повністю прозорою (крім меж), що дозволяє їй не закривати собою огляд. На рисунку 2.1 представлений скріншот з гри «Block Out».

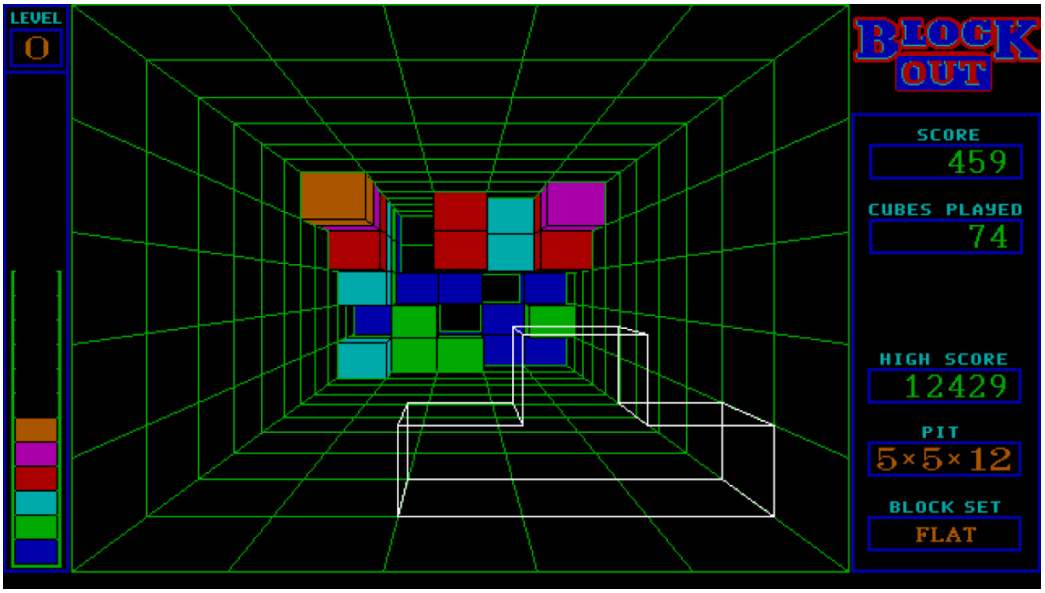

Рисунок 2.1 - Гра «Block Out»

Плюси гри

1. Наявність 3D, що робить ігровий процес більш цікавим і складним.

Мінуси гри

1. Починаючи приблизно з 4-го шару стає складно орієнтуватися на ігровому полі.

2. Складно зрозуміти, куди саме впаде фігура.

3. Поки фігура квадратної форми розташована близько до камери, майже неможливо зрозуміти якого саме вона розміру: 1x1, 1x2 або 2x2.

4. Звуковий супровід вкрай незбалансовано. Підібрана музика швидко набридає.

Tricky Towers

«Tricky Towers» - розрахована на багато користувачів гра, розрахована на 1-4 гравців [12]. В основі гри лежить механіка тетрису з фізикою. Тобто, якщо неакуратно класти одну на одну фігури, вони можуть обвалитися. Гра містить 3 режиму.

1. Гонка - перемагає той гравець, який швидше побудує вежу зазначеної висоти.

2. Виживання - кожному гравцеві видається 3 життя. За кожну втрачену фігуру 1 життя віднімається. Відповідно, гравець, що упустив 3 фігури, програє. Перемагає той, хто зможе протриматися довше інших.

3. Головоломка - виставляється обмеження по висоті вежі. Якщо гравець перевищить це обмеження - він вибуває, а його вежа фіксується (На фігури перестає діяти фізика). Коли всі гравці перевищать обмеження, перемагає той, хто зміг вмістити на ігровому полі більше всіх фігурок.

Також, по ходу гри, гравці можуть задіяти різні бонуси, що впливають на ігровий процес всіх гравців. На рисунку 2.2 представлені скріншоти екранів гри «Tricky Towers».

Рисунок 2.2 - Гра «Tricky Towers»

Плюси гри

1. Можна грати як одному, так і з іншими гравцями.

2. Наявність різних режимів гри.

3. Наявність різноманітних ігрових механік, що роблять ігровий процес більш цікавим.

4. Приємна візуальна складова гри.

Мінуси гри

Гра явно розрахована на кількох гравців. Хоч в грі і є можливість грати одному, це швидко набридає.

Tetris 99

«Tetris 99» - гра, в якій 99 гравців, перебуваючи в одному лоббі, одночасно грають в тетріс [15]. Перемагає той гравець, хто за підсумком набере найбільшу кількість очок. Правила гри такі ж, як і в оригінальному тетрісі, за одним винятком: у грі присутня можливість заважати іншим гравцям. Робиться це за рахунок додавання на ігрове поле іншого гравця додаткового ряду знизу (позначається сірими квадратами). Додатковий ряд, по суті, являє собою звичайний ряд, де не вистачає одного квадрата. При цьому, чим краще грає гравець (знищує по кілька рядів за раз і т.д.), тим більше додаткових рядів він може відправити іншим гравцям.

«Tetris 99» вийшла за часів, коли в ігровій індустрії був дуже популярний жанр «Battle Royale» (королівська битва). Даний жанр спочатку з'явився в іграх шутерах і представляв собою режим гри, де N кількість гравців (як правило, близько 100) одночасно потрапляє на одну велику карту і воює один з одним до тих пір, поки в живих не залишиться тільки один гравець (або команда). При цьому розмір карти поступово зменшується, щоб гравцям рано чи пізно довелося зіткнутися один з одним.

В 2017 року жанр «Battle Royale» придбав настільки високу популярність, що став з'являтися мало не у всіх іграх шутерах того часу, а пізніше, і в іграх, що не мають до шутерів ніякого відношення, оскільки розробники ігор хотіли отримати якомога більше вигоди з популярного тоді жанру. «Tetris 99» - одна з таких ігор.

На рисунку 2.3. представлений скріншот гри «Tetris 99».

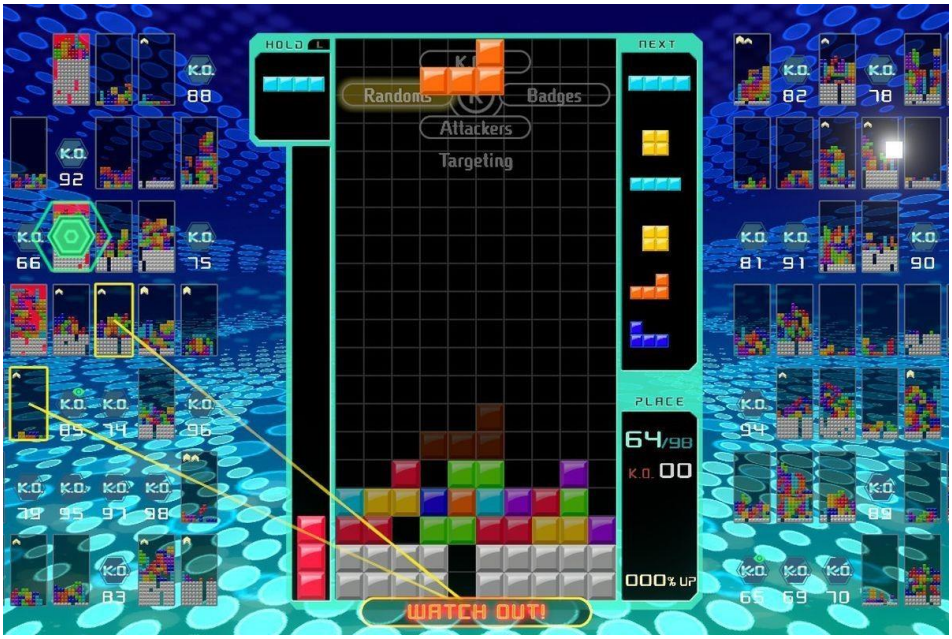

Рисунок 2.3 - Гра «Tetris 99»

Плюси гри

1. Простий і зрозумілий ігровий процес, в якому зможе швидко освоїтися будь-який гравець.

2. Наявність змагального аспекту робить гру більш захоплюючою.

Мінуси гри

Гра втратила свою популярність так само швидко, як і отримала її, а без гравців ця гра перетворюється в звичайний тетріс.

Висновок до розділу

В ході аналізу предметної області та існуючих робіт по тематиці випускної роботи було виявлено ряд аспектів, які потрібно врахувати при розробці гри «Тетріс 3D».

1. У грі повинна бути своя відмінна риса, яка виділятиме гру на тлі інших ігор-тетрисів. Одного лише перенесення в 3D буде недостатньо. Для цього, в грі буде реалізований конструктор фігур, тобто можливість, при натисканні певної клавіші поставити гру на паузу і перейти в режим

редагування наступної фігури, де гравець зможе додати до неї додаткові куби. На дану ігрову можливість будуть накладені обмеження, на кшталт: обмеження за розміром фігури, обмеження на кількість додаткових кубів і т.д.

2. Необхідно забезпечити в грі можливість легко орієнтуватися на ігровому 3D полі. Можливість вільного обертання камери повинна бути обов'язково, але варто також додати ще якусь ігрову механіку, спрямовану на вирішення цієї проблеми. Даною ігровою механікою буде можливість робити зріз. При натисканні певної клавіші, найближчий до камери вертикальний шар буде ставати напівпрозорим. Таким чином, якщо гравець натисне на клавішу п'ять разів, то напівпрозорими стануть найближчі п'ять шарів.

РОЗДІЛ 3
Аналіз вимог до програмної системи

Даний пункт містить перелік функціональних і нефункціональних вимог до програмної системи, а також діаграму варіантів використання.

3.1 Функціональні і нефункціональні вимоги

Функціональні вимоги до проєктованої системи:

1) система повинна видаляти горизонтальну площину, повністю заповнену кубами;

2) система повинна завершувати гру, якщо встановлена гравцем фігура виходить за межі верхньої межі ігрового поля;

3) система повинна нараховувати гравцеві очки за кожну заповнену площину;

4) система повинна поступово підвищувати темп гри (підвищувати швидкість падіння фігур);

5) система повинна зберігати максимальний рекорд гравця і ігрові настройки в файл.

Нефункціональні вимоги до проєктованої системи:

1) система повинна працювати на операційній системі Windows 7 і вище;

2) система повинна бути написана з використанням платформи Unity;

3) система повинна бути написана на мові програмування C # [19].

3.2 Діаграма варіантів використання

На рисунку 3.1 представлена діаграма варіантів використання комп'ютерної гри «Тетріс 3D». З системою взаємодіє тільки один основний актор - гравець.

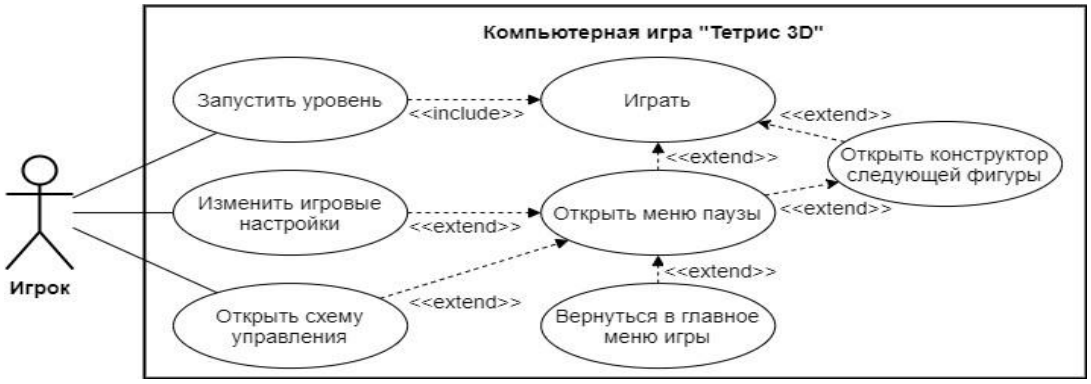

Рисунок 3.1 - Діаграма варіантів використання

Гравець - це людина, що грає в комп'ютерну гру «Тетріс 3D». Запустивши рівень, завдання гравця - набрати якомога більше очок, заповнюючи вертикальні площини фігурами.

В ході гри гравець може відкрити конструктор наступної фігури, щоб змінити її початкову форму. Крім цього, гравець може змінити ігрові налаштування (якість графіки, дозвіл вікна гри, гучність звуку і т.д.), і відкрити схему управління. У схемі управління наведено список всіх дій, доступних гравцеві, і те, які клавіші для цього потрібно натиснути / затиснути.

Специфікації основних варіантів використання наведені в додатку А.

Висновок до розділу

В даному розділі були виділені функціональні і нефункціональні вимоги до системи, а також визначені і описані варіанти використання.

РОЗДІЛ 4
Архітектура системи

4.1 Діаграма компонентів

Система складається з двох сцен:

1) головне меню;

2) гра.

При запуску програми завжди завантажується сцена «Головне меню». У головному меню є призначений для користувача інтерфейс, який дозволяє гравцеві запустити сцену «Гра», а також подивитися схему управління, змінити налаштування і вийти з гри.

Налаштування гри і рекордна кількість очок, отриманих гравцем під час гри, система зберігає в окремому файлі збереження на комп'ютері гравця.

При запуску сцени «Головне меню» система перевіряє наявність файлу збереження, і, якщо він присутній, бере дані звідти, інакше - виставляє настройки за замовчуванням. При запуску сцени «Гра» система формує пусте ігрове поле і запускає ігровий процес.

У сцені «Гра» гравцеві доступно кілька вікон і призначених для користувача інтерфейсів, між якими гравець може перемикатися вручну.

1. Ігрове поле - вікно, що відкривається за замовчуванням. Тут гравець бачить ігрове поле, може змінювати положення камери, переміщати і обертати падаючу фігуру, а також робити зріз.

2. Конструктор фігур - вікно, в якому гравець може розглянути наступну фігуру з різних сторін, а також додати або видалити додаткові куби.

3. Меню паузи - призначений для користувача інтерфейс, що дозволяє гравцеві подивитися схему управління, змінити налаштування

гри, повернутися в сцену «Головне меню» або вийти з гри. За замовчуванням цей інтерфейс закритий. Відкривається при натисканні певної клавіші на клавіатурі.

4. Результати гри - призначений для користувача інтерфейс, що відображає результат гри і рекордний рахунок, а також дозволяє гравцеві почати гру заново або повернутися в сцену «Головне меню». За замовчуванням цей інтерфейс закритий. Відкривається системою, тільки при завершенні гри (Коли встановлена фігура виходить за верхню межу ігрового поля).

На рисунку 4.1 представлена діаграма компонентів гри «Тетріс 3D».

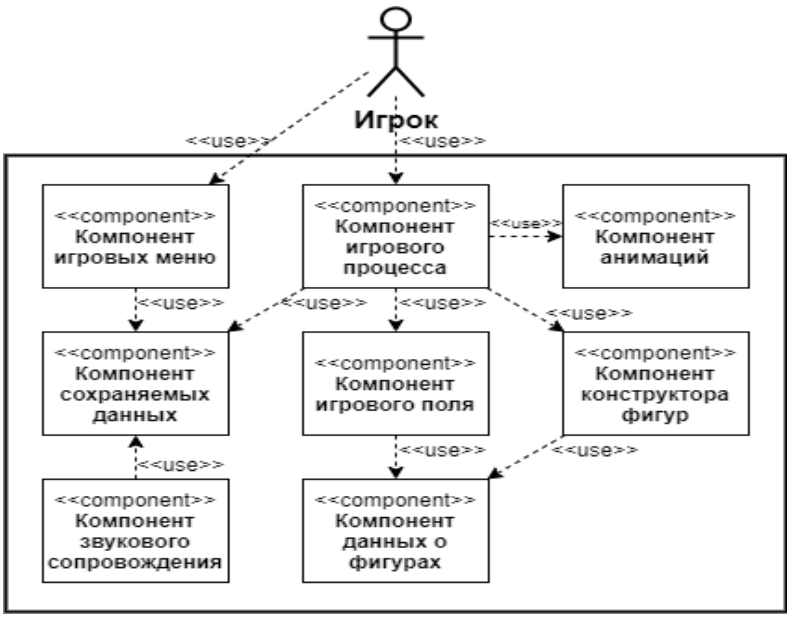

Рисунок 4.1 - Діаграма компонентів

Компонент ігрових меню - містить набір класів, що реалізують головне меню, меню паузи, а також меню, що з'являється при завершенні гри. Дозволяє гравцеві перемикатися між вікнами меню, якщо такі є і користуватися функціоналом, яке дане меню надає. Компонент даних, що зберігається - містить набір класів, в яких реалізована можливість зберігати і завантажувати певні дані з файлу. До його записала даними

відноситься рекордну кількість очок і налаштування гри. При збереженні або завантаженні налаштувань гри система автоматично буде застосовувати ці налаштування до самої гри.

Компонент звукового супроводу - містить набір класів, дозволяють системі взаємодіяти з музикою і звуковими ефектами. Так само як включення/ вимикання звуків, а також можливість плавної зміни деяких параметрів протягом заданого часу (наприклад, зменшення гучності музики з 1.0 до 0.5 на протязі 3 секунд і т.д.).

Компонент ігрового процесу - містить набір класів, що реалізують ігровий процес системи. Даний компонент відстежує весь контекст гри: яке вікно відкрите в даний момент (основне або конструктор), швидкість гри, кількість набраних очок, чи є на ігровому полі заповнена площина, яку необхідно видалити і т.д. Також, даний компонент стежить за діями гравця (натискання клавіші, рух мишею і т.д.) і викликає всі необхідні функції, якщо такі є.

Компонент ігрового поля - містить набір класів, що реалізують ігрове поле. Зберігає інформацію про стан всіх осередків ігрового поля, а також реалізує функції, що дозволяють перевірити, чи є на поле заповнені вертикальні площини, чи може фігура опуститися ще на одну клітинку вниз, не зустрівшись з перешкодою, чи можна пересунути/повернути фігуру в зазначеному гравцем напрямку і т.д.

Компонент конструктору фігур - містить набір класів, що реалізують конструктор фігур. Даний компонент реалізує можливість додавати і видаляти додаткові куби до фігури. При додаванні куба відбувається перевірка, чи не порушує це встановлених обмежень (Наприклад, гравець і так вже використовував максимальну кількість додаткових кубів або ж гравець намагається розширити фігуру, яка і так вже досягла своєї максимальної ширини і т.д.). При видаленні куба

відбувається перевірка, чи не розіб'є це фігуру на дві незв'язні постаті. В такому разі видалення цього куба забороняється.

Компонент даних про фігури - містить набір класів, що зберігають інформацію про фігуру у вигляді тривимірної матриці. Даний компонент дозволяє додавати в матрицю нові куби і видаляти вже наявні. Також, в компоненті реалізована перевірка, чи не призведе видалення конкретного куба до розриву фігури на два незв'язні.

Компонент анімацій - містить набір класів, що реалізують анімації, які були написані не за допомогою вбудованого в Unity редактора анімацій, а вручну, у вигляді програмного коду, де відбувається плавна зміна певних параметрів протягом заданого проміжку часу (наприклад, плавне переміщення об'єкта з координат {0, 0, 0} на координати {1; 1; 1}).

4.2 Макети для користувача інтерфейсів

На рисунках 4.2 – 4.5 представлені макети деяких користувальницьких інтерфейсів, які будуть реалізовані в грі. Дані макети є лише зразковим уявленням про те, як будуть виглядати інтерфейси в підсумковій версії гри. В ході етапів реалізації і юзабіліті тестування деякі елементи інтерфейсів можуть бути додані, прибрані, змінені або перенесені в іншу частину екрана.

У головному меню гравцеві буде доступно чотири кнопки:

1) грати - запускає сцену «Гра»;

2) управління - відкриває розділ меню «Управління»;

3) налаштування - відкриває розділ меню «Налаштування»;

4) вийти з гри - завершує роботу системи.

Рис. 4.2. Макет інтерфейсу головного меню

У меню паузи гравцеві доступно п'ять кнопок:

1) продовжити - закриває меню паузи;

2) управління - відкриває розділ меню «Управління»;

3) налаштування - відкриває розділ меню «Налаштування»;

4) в головне меню - запускає сцену «Головне меню»;

5) вийти з гри - завершує роботу системи

Рисунок 4.3 - Макет інтерфейсу меню паузи

Під час ігрового процесу, коли на екрані відкрито основне вікно (Тобто вікно з ігровим полем), курсор ніяк не бере участі в ігровому

процесі, тому він не буде відображатися. З тієї ж причини інтерфейс не містить будь-яких кнопок, тільки графічну інформацію.

1. Швидкість гри - вказує поточний темп гри. Спочатку даний параметр дорівнює 1.0. Раз в якусь кількість часу швидкість гри буде підвищуватися на 0.1, і так до тих пір, поки гра не завершиться.

2. Режим камери - може бути сфера (камера рухається навколо ігрового поля, як по поверхні сфери) або площина (камера рухається відносно площини, перпендикулярної напрямку камери).

3. Режим фігури - може бути переміщення або обертання.

4. Наступна фігура - виводить на екран те, як буде виглядати наступна фігура.

5. Підказки в управлінні - нагадує гравцеві, які клавіші необхідно натискати для вчинення дій, якими гравець буде користуватися нечасто (наприклад, відкриття конструктора).

6. Рахунок - кількість очок, набраних гравцем в даний момент.

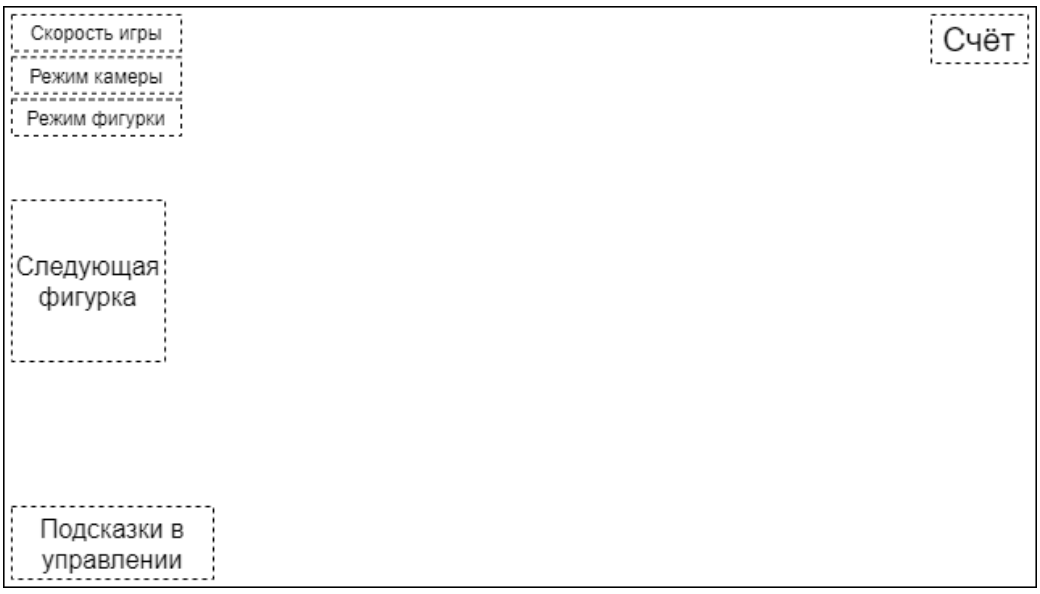

Рисунок 4.4 - Макет інтерфейсу основного вікна ігрового процесу

Результат гри з'являється тільки при завершенні гри. Відображає кількість очок, набраних за гру і рекордний рахунок. Також, даний інтерфейс містить дві кнопки:

1) заново - перезапускає сцену «Гра»;

2) в головне меню - запускає сцену «Головне меню».

Рисунок 4.5 - Макет інтерфейсу результату гри

Висновок до розділу

В даному розділі була визначена архітектура системи, описані всі компоненти системи і різні режими гри. Також, були складені макети ігрових меню та користувальницьких інтерфейсів з описом кожного елемента інтерфейсу.

РОЗДІЛ 5
Реалізація системи

Платформа Unity була обрана, оскільки вона є безкоштовною і досить популярною серед розробників. За Unity існує докладна офіційна документація [12] і безліч текстових і відео уроків від інших користувачів.

Платформа Unity надає широкий набір інструментів і бібліотек, що дозволяють з легкістю створювати гри зі складними, з технічної точки зору, ігровими механіками, якісної графікою і високою оптимізацією. Нижче представлені основні можливості Unity.

1. Створення різних джерел освітлення (глобальне, точкове, прожектори і так далі).
2. Динамічний розрахунок тіней від об'єктів. У тому числі - можливість створювати м'які тіні.
3. Динамічний розрахунок відображень.
4. Підтримка фізики твердих тіл і тканин.
5. Підтримка системи Level Of Detail [20], суть якої полягає в тому, що на далекій відстані від гравця високо деталізовані моделі замінюються на менш деталізовані, і навпаки.
6. Наявність інструменту для редагування текстур.
7. Наявність інструменту, для створення шейдерів (шейдер - спеціальна програма комп'ютерної графіки, яка визначає остаточні параметри об'єкту) дозволяє додати поверхні об'єкту властивості поглинання/розсіювання/ відбиття світла, додати поверхні ефект обсягу і багато іншого [21].

8. Наявність інструменту для створення партиклів. Партикли - ефекти, що складаються з великої кількості частинок. Кожна частка має певний набір атрибутів, що впливають на її поведінку [22].

5.1 Реалізація компоненту збереження даних

Всі дані ігри зберігаються в файлі data.t3ddata. Платформа Unity сама визначає розташування цього файлу, в залежності від системи. В таблиці 1 наведено опис всіх змінних, що зберігаються в файл. При запуску гри насамперед відбувається перевірка наявності файлу збереження. Якщо файл існує, всі дані налаштувань гри беруться звідти, інакше - виставляються значення за замовчуванням. При закінченні ігрового процесу відбувається порівняння поточного рекорду з максимальним, узятим з файлу, і, якщо поточний рекорд більше, файл перезаписується з новим значенням

Таблиця 5.1

Збереження даних

Назва змінної	Тип даних	Опис	Значення за замовчуванням
score	long	Максимальна кількість очок, набраних гравцем	0
cameraSpeed	float	Швидкість переміщення камери, відносно швидкості руху миші	4,0
languageIndex	int	Індекс мови тексту: 0 – англійский, 1 – російський	0
musicVolume	float	Гучність музики	10,0
soundsVolume	float	Гучність звукових ефектів	15,0
fullscreen	bool	Повноекранний режим (якщо false – гра буде прцювати у віконному режимі)	true
windowHeight	int	Висота вікна в пікселях	Висота екрану
windowWidth	int	Ширина вікна в пікселях	Ширина екрану
msaaIndex	int	Індекс згладжування:	0

		– 0 – без згладжування; – 1 – згладжування x2; – 2 – згладжування x4; – 3 – згладжування x8.	

5.2 Реалізація компоненту звукового супроводу

На платформі Unity є вбудований компонент Audio Source, який використовується для відтворення різних звуків. На даний компонент передається звуковий файл, який потрібно відтворити, а також вказується гучність, швидкість відтворення (може бути і негативною: звук вибуде відтворюватися в зворотному напрямку) і інші параметри. Важливим параметром є Spatial Blend, який приймає значення від 0 до 1. При значенні 0 звук буде повністю фоновим - тобто, він буде звучати однаково, незалежно від того, де знаходиться гравець на ігровому просторі. При значенні 1 звук буде повністю 3D - тобто, буде звучати голосніше з тієї сторони, з якої знаходиться джерело звуку щодо гравця, і буде ставати тихіше, при віддаленні від джерела звуку.

У «Tetris 3D» все звуки поділені на музику і звукові ефекти. Оскільки вся музика є фоновою - всі компоненти Audio Source, що відтворюють музику, створюються на тому ж ігровому об'єкті, що і клас SoundsManager, який відповідає за роботу зі звуком. При відтворення будь-якого звукового ефекту створюється новий порожній ігровий об'єкт, на якому створюється компонент Audio Source з необхідним звуковим ефектом. Якщо звуковий ефект є частково або повністю 3D - при створенні ігрового об'єкта вказуються також координати джерела звуку. Як тільки звук закінчує своє відтворення, ігровий об'єкт віддаляється.

5.3 Реалізація компоненту ігрових меню

У грі реалізовано 3 ігрових меню:

1) головне меню;

2) меню паузи;

3) меню закінчення гри.

Головне меню і меню паузи мають схожий функціонал: гравець може змінити ігрові налаштування, подивитися схему управління і вийти з гри. Відмінністю головного меню від меню паузи є те, що в головному меню є анімований задній фон, а відповідно і додаткові класи, які його реалізують.

На рисунку 5.1. представлена діаграма класів, що відображає реалізацію головного меню.

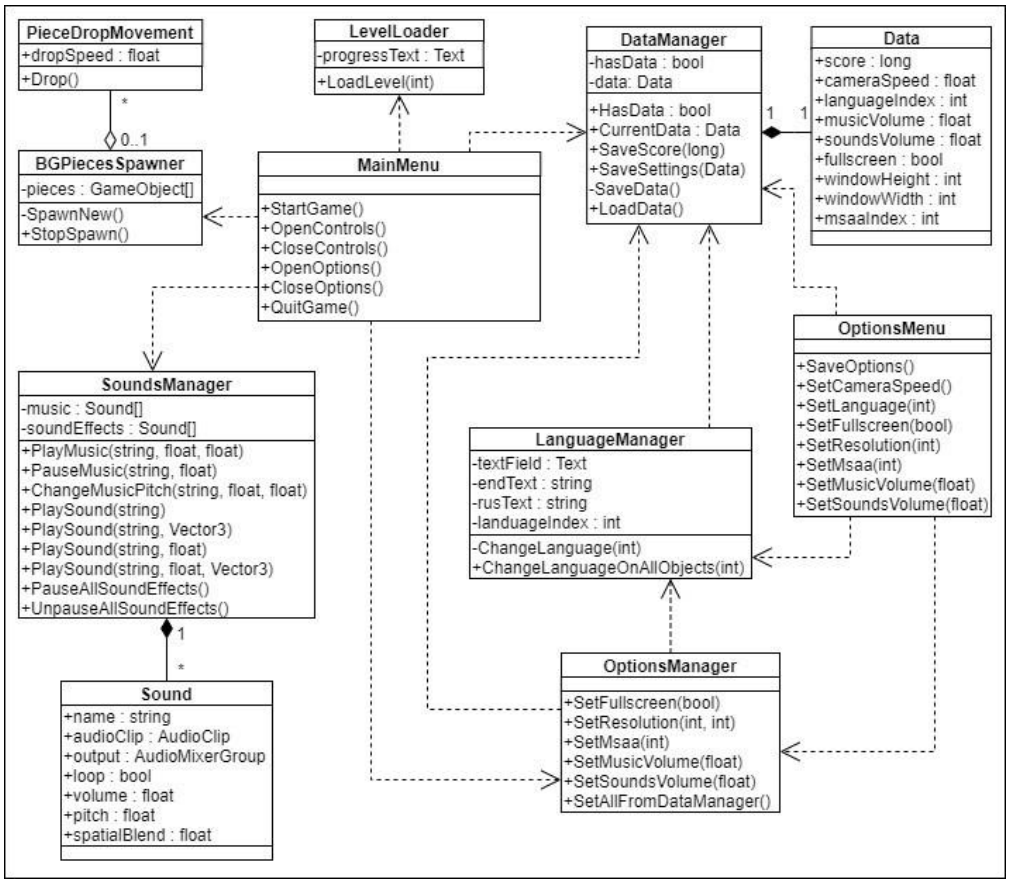

Рис.5.1. Діаграма класів «Реалізація головного меню»

Основним класом тут є клас MainMenu, в якому реалізована функція початку гри і функції перемикання між різними розділами меню.

Клас DataManager використовується для збереження і завантаження ігрових налаштувань з файлу, OptionsManager - для застосування зазначених ігрових налаштувань, таких налаштувань, які вибираються зі файлу збереження, а OptionsMenu – для миттєвого відображення результатів зміни ігрових налаштувань (Наприклад, гравець відключив в налаштуваннях повноекранний режим - вікно гри тут же переходить у віконний режим).

Клас LanguageManager використовується для перемикання мови тексту. Даний клас прив'язується до кожного текстового поля гри, де необхідна можливість зміни мови, а в атрибутах endText і rusText вказується текст поля на англійській і російській мовах відповідно.

На рисунку 5.2 представлені скріншоти меню налаштувань на різних мовах.

Рисунок 5.2 - Меню налаштувань російською та англійською мовами

Клас BGPiecesSpawner використовується для створення падаючих фігур тетрису в головному меню. При натисканні на кнопку «Грати» даний клас перестає створювати нові фігури і значно прискорює падіння вже створених фігур. Після, викликається функція LoadLevel класу LevelLoader для переходу на екран завантаження.

Клас SoundsManager використовується для відтворення фонової музики, а також звукових ефектів при наведенні / натисканні на кнопки меню.

5.4 Реалізація компоненту ігрового процесу

В даному компоненті реалізована можливість перемикатися між ігровим полем і конструктором, облік призначеного для користувача введення, підрахунок кількості очок, зміна швидкості гри і відстеження поточного стану гри. Кожні 1/V секунд, де V - поточна швидкість гри, відбувається запит ігровому полю на переміщення поточної фігури вниз на одну клітинку. Даний запит повертає true, якщо фігура була переміщена вниз і false, якщо фігура врізалася в нижню межу ігрового поля, або в іншу фігуру. Якщо запит повернув false, відбувається перевірка, чи не привела установка нової фігури до завершення гри (фігура вийшла за верхню межу поля). Якщо привела - на екрані з'являється меню закінчення гри, де написано, скільки очок зміг набрати гравець в перебігу гри і максимальний рекорд. Якщо установка нової фігури не привела до закінчення гри, гравцеві нараховуються 5 ігрових очок. Після відбувається перевірка, чи не привела установка нової фігури до появи горизонтальних рядів, повністю заповнених кубами. Якщо привела - гравцеві нараховуються ігрові очки в розмірі N2*100, де N - кількість заповнених площин. Крім нарахування очок відбувається зменшення швидкості гри на (N * 2-1)/2, де N - кількість заповнених площин (швидкість гри не може впасти нижче 1,0). Кожні 25 секунд швидкість гри підвищується на 0,1.

5.5 Реалізація компоненту ігрового поля

На рисунку 5.3 представлена діаграма класів, що відображають реалізацію ігрового поля. В даній діаграмі, у всіх класах, крім Playground вказані тільки ті функції, в яких відбувається звернення до класу Playground, або які викликаються з класу Playground.

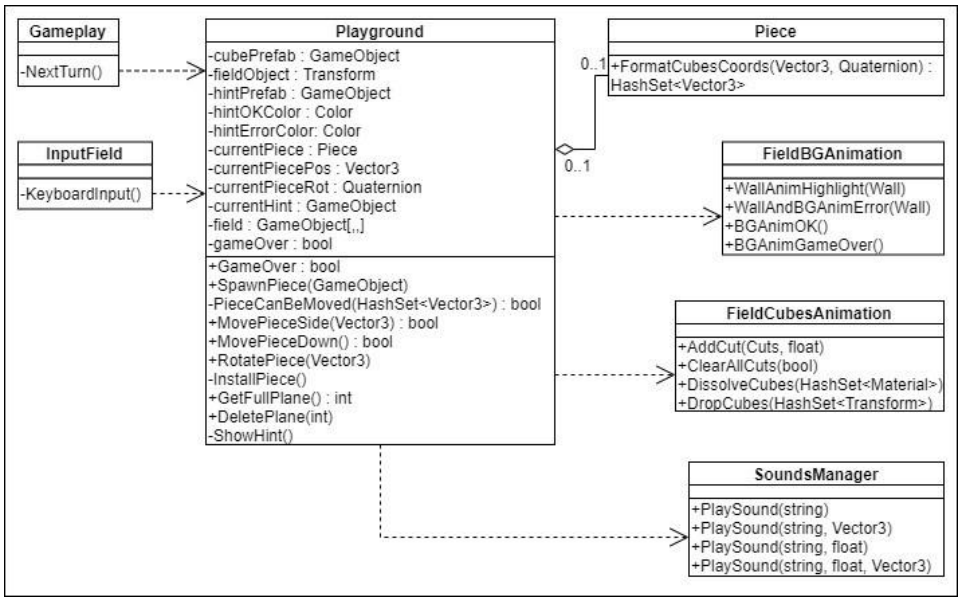

Рисунок 5.3 - Діаграма класів «Реалізація ігрового поля»

Клас Playground зберігає інформацію про ігрове поле у вигляді тривимірної матриці розміром 9x9x15. В даному класі реалізована можливість переміщення і обертання фігури по ігровому полю, відображення підказки про те, де виявиться фігура, якщо дозволити їй впасти, а також можливість знаходити і видаляти заповнені кубами горизонтальні площини.

Переміщення фігури, видалення ряду, спроба перемістити фігуру за межі поля і інші дії гравця супроводжуються різними графічними та звуковими ефектами.

5.6 Реалізація компоненту конструктора фігур

В даному компоненті реалізована можливість додавати або видаляти додаткові куби до наступної фігури. При заході в режим конструктора, поточна швидкість гри зменшується в 5 разів. Гравець може додати не більше 4-х додаткових кубів на фігуру. При цьому по кожній з трьох осей розмір фігури не може перевищувати 4 куба.

Гравець може видалити будь-який з додаткових кубів, але тільки в тому випадку, якщо це не призведе до розриву на декілька незв'язних фігур. Інакше, гра підсвітить червоним все ті куби, які заважають видаленню обраного. До тих пір, поки наступна фігура не з'явилася на ігровому полі, гравець може скільки завгодно разів користуватися конструктором і редагувати цю фігуру. Як тільки фігура з'явиться на полі, гра вважатиме, скільки на ній додаткових кубів і заблокує конструктор для наступних N фігур, де N - кількість додаткових кубів. Таким чином, якщо гравець не став додавати на фігуру до кубів, то і для наступної фігури конструктор буде доступний. Якщо ж гравець додав, наприклад, 3 куба, то для наступних 3-х фігур конструктор буде заблокований. Для відображення стану конструктора на інтерфейсі ігрового поля існує показник «Заряд конструктора», виражений кольоровий рамкою навколо вікна наступної фігури. Якщо рамка повністю зафарбована (Заряд конструктора: 100%), значить в даний момент конструктор доступний для використання. Якщо ж рамка зафарбована в повному обсязі, значить конструктор заблокований, і для його розблокування потрібно продовжувати

встановлювати на ігровому полі фігури. На рисунку 5.4. представлено, як відображається заряд конструктора при різних рівнях заряду.

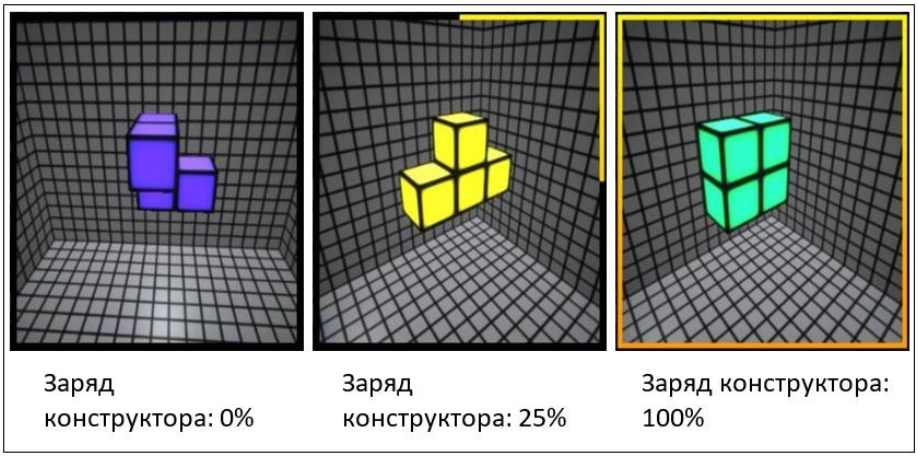

Рисунок 5.4 - Відображення заряду конструктора на інтерфейсі ігрового поля

5.7 Реалізація компоненту даних про фігури

Інформація про куби, з яких складається фігура, зберігається у вигляді 3-х мірної матриці, де індексами елементів є їх локальні координати щодо центрального куба фігури з локальними координатами {0, 0, 0}. В даному компоненті реалізовані функції для додавання/видалення кубів, а також функції, які здійснюють перевірку, чи можна додати/видалити куб за заданими координатами. Всі ці функції активно використовуються в роботі конструктора.

Також, в даному компоненті реалізована функція, форматує локальні координати кубів фігури в глобальні, в залежності від координат і кута повороту самої фігури на ігровому полі.

5.8 Реалізація компоненту анімацій

В даному компоненті реалізовані різні анімації, виробництво не за допомогою вбудовананих інструментів Unity, а написані вручну, у вигляді програмного коду.

Для того, щоб реалізувати анімацію у вигляді програмного коду, необхідно через рівні проміжки часу потроху змінювати значення

заданого параметра, наближаючи його до необхідного. Чим частіше змінюється значення параметра, тим більше плавно буде виглядати анімація. Для створення таких анімацій в Unity використовуються coroutine-функції - це такі функції, виконання яких може бути призупинено на зазначений термін.

Важливо відзначити, що coroutine-функції не є паралельними: вони все ще виконуються послідовно основним пітом до тих пір, поки не відбудеться вихід з функції, або поки не буде зроблений запит на призупинення роботи функції. Так що занадто велика кількість одночасно запущених coroutine-функцій може уповільнити основний потік, що негативно позначиться на продуктивності гри. Нижче представлений приклад реалізації анімації підсвічування кубів, з використанням coroutine-функції. Дана анімація використовується конструктором для підсвічування всіх кубів, що заважають при видалення зазначеного гравцем куба.

```
private Coroutine errorAnim = null;

public void ShowError(HashSet<GameObject> weakSpots)
{
    if (errorAnim != null) return;

    HashSet<Material>      weakSpotMaterials      =      new
    HashSet<Material>(); foreach   (GameObject   weakSpot   in
    weakSpots)
        weakSpotMaterials.Add(
            weakSpot.GetComponent<MeshRenderer>().material);

    errorAnim = StartCoroutine(ShowErrorAnim(weakSpotMaterials));
}

private IEnumerator ShowErrorAnim(HashSet<Material> materials)
{
    float t = 0;
    while (t < 1) // підсвічуємо куби
    {
        t          +=          Time.deltaTime          *
        addCubeErrorColorChangeSpeed;if (t > 1) t = 1;
        foreach       (Material       material       in       materials)
            material.SetColor("Emission_Color", Color.Lerp(
                addCubeOriginalColor, addCubeErrorColor, t));

        yield return 0; // чекаємо наступного кадру
    }
```

```
t = 0;
while (t < 1) // повертаємо кубам початкове освітлення
{
    t          +=          Time.deltaTime          *
    addCubeErrorColorChangeSpeed;if (t > 1) t = 1;
    foreach       (Material       material       in       materials)
        material.SetColor("Emission_Color", Color.Lerp(
            addCubeErrorColor, addCubeOriginalColor, t));

    yield return 0; // чекаємо наступного кадру
}
errorAnim = null;
```

Як видно, дана функція в кожному кадрі потроху змінює колір зазначених кубів з початкового на колір підсвічування. Як тільки необхідний колір буде досягнутий, функція з тією ж швидкістю буде повертати кубам їх початковий колір.

5.9 Шейдери

У «Tetris 3D» для додання ігровим об'єктів особливого візуального стилю, реалізації деяких анімацій заднього фону, а також реалізації ігрової механіки «Зріз» використовуються різні шейдери. На платформі Unity шейдери можна створити двома способами: через код, або за допомогою вбудованого інструменту шейдер-граф. Шейдер-граф - спеціальний інструмент створення шейдерів, який представляє шейдер у вигляді графа, де кожен вузол реалізує будь-яку операцію по перетворенню вхідного потоку даних. У шейдер-графі існує величезна кількість вузлів, що реалізують найрізноманітніші операції, починаючи від простих математичних перетворень і закінчуючи урахуванням позиції об'єкта в ігровому просторі або щодо екрану, урахуванням матриці глибини і т.д. Шейдери в Unity дозволяють навіть спотворювати форму ігрового об'єкта, що часто використовується при створенні анімації вітру на листі дерев і траві.

На рисунку 5.5. представлений приклад найпростішого використання шейдер-графа.

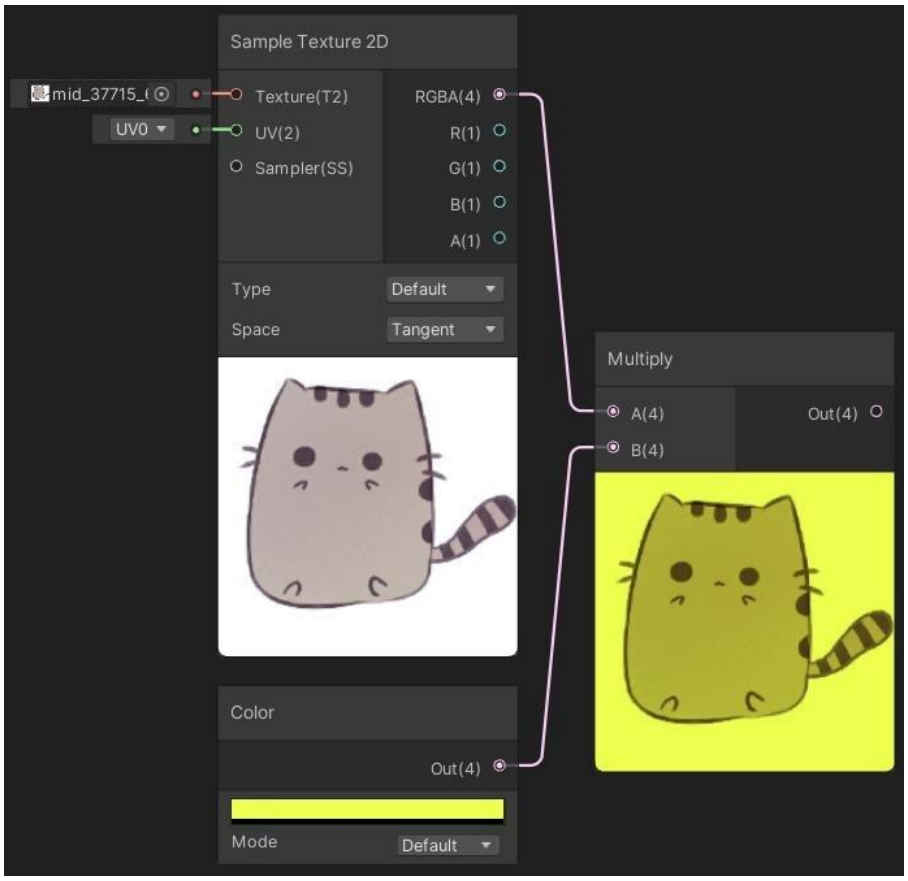

Рисунок 5.5 - Приклад роботи вузла «Помножити» в шейдер-графі

Як видно з рисунку, якщо вузлу «Помножити» передати на один вхід текстуру, а на інший колір, на виході отримаємо текстуру, забарвлену в вказаний колір.

Ще однією особливістю шейдер-графа є те, що будь-який простий параметр (число, колір, текстура і т.д.) можна зробити зовнішнім. В такому випадку, значення цього параметра можна буде змінювати за межами шейдер-графа, в тому числі і через код.

Найскладнішим шейдером, написаним для «Tetris 3D», є шейдер «Cube», застосовуваний в грі до всіх кубів, з яких складаються фігури тетрису. У цьому шейдері, крім стандартних налаштувань текстури, кольору, властивостей відбиття поверхні і іншого, реалізовані також 3 унікальних ефекти

1. Ефект появи. Виявляється, коли фігура, перебуваючи вище ігрового поля, наближається до його верхньої межі.

2. Ефект зникнення. Виявляється, коли гравець заповнює кубами горизонтальну площину і ця площину отримує червону картку.

3. Ефект зрізу. Використовується в ігровій механіці «Зріз».

Висновок до розділу

Відповідно до певних вимог і спроектованої архітектурою системи була реалізована комп'ютерна гра «Tetris 3D». Було складено детальний опис реалізації всіх компонентів гри, з приведенням вихідного коду, різних діаграм класів і скріншотів окремих аспектів гри.

Для гри було написано 27 класів. У таблиці 5.2. представлені кількісні метрики кожного класу, виражені кількістю рядків вихідного коду і кількістю рядків коду, що виконується.

Таблиця 5.2.

Кількісні метрики класів

№	Назва класу	Кількість рядків вихідного коду
1	BGPiecesSpawner	46
2	CameraMovements	217
3	Constructor	121
4	ConstructorAnimation	55
5	Data	14
6	DataManager	79
7	FieldBGAnimation	213
8	FieldCubesAnimation	224
9	FieldWallAnimation	74
10	GameoverMenu	28
11	Gameplay	273
12	InputConstructor	80
13	InputField	195
14	LanguageManager	40
15	LevelLoader	38
16	MainMenu	48
17	OptionManager	63
18	OptionMenu	92
19	PauseMenu	128

20	Piece	187
21	PieceDropMovement	13
22	Playground	303
23	Sound	14
24	SoundManager	116
25	Transition	106
26	UIConstructor	8
27	UIField	29
ИТОГО	2804	1134

РОЗДІЛ 6
Тестування

6.1 Функціональне тестування

В ході даного тестування перевірялося відповідність програмного продукту функціональним вимогам. У таблиці 6.1. представлені результати тестування.

Таблиця 6.1

Функціональне тестування гри «Tetris 3D»

№	Назва тесту	Дії	Результат	Тест пройдений?
1.	Працездатність головного меню	1. У головному меню обрати пункт «управління». 2. Натиснути кнопку «Назад»	Перемикання між різними розділами головного меню.	Так
2.	Працездатність меню налаштувань	1. У головному меню обрати пункт «Налаштування». 2. Відключити повноекранний режим.	Вікно гри перейшло в віконний режим.	Так
3.	Збереження ігрових налаштувань	1. У головному меню обрати пункт «Налаштування». 2. Відключити повноекранний режим. 3. Натиснути кнопку «Зберегти». 4. Запустити знову гру	Гра запускається в віконному режимі.	Так
4.	Перемикання між ігровими рівнями	1. У головному меню обрати пункт «Грати». 2. Дочекатися	При запуску ігрового процесу спочатку з'являється екран	Так

		завантаження ігрового поля 3. Відкрити меню паузи натисканням кнопки ESC. 4. У меню паузи вибрати пункт «Головне меню».	завантаження, і коли процент завантаження досягає 100%, на екрані відображається ігрове поле. При поверненні в головне меню також, спочатку з'являється екран загрузки і тільки потім саме меню.	
5.	Переміщення камери	1. Запустити ігровий процес. 2. Поводити компьютерною мишею з і без зажиму правої кнопки. 3. Покрутити коліщатко миші вгору і вниз. 4. Натиснути на коліщатко миші.	Камера обертається навколо ігрового поля, переміщається уздовж перпендикулярної площини, наближається, віддаляється і повертається на початковий радіус.	Так
6.	Переміщення фігури	1. Запустити ігровий процес. 2. Натиснути на клавіші переміщення (W, A, S, D).	Фігура переміщується по ігровому по- лю.	Так
7.	Працездатність конструктора фігур	1. Запустити ігровий процес. 2. Відкрити конструктор натисканням клавіші V. 3. За допомогою курсору і лівої кнопки миші додати на фігуру додатковий куб. 4. Закрити конструктор натисканням клавіші V. 5. Дочекатися появи наступної	На ігровому полі з'явилася фігура з встановленим на неї додатковим кубом.	Так

		фігури на ігровому полі.		
8.	Видалення заповненої горизонтальної площини	1. Запустити ігровий процес. 2. Грати в гру до тих пір, поки не заповниться горизонтальна площина	Заповнена горизонтальна площина пішла з ігрового поля, а всі куби, що знаходились вище неї впали на одну клітинку вниз.	Так
9.	Нарахування ігрових очок	1. Запустити ігровий процес. 2. Встановити на поле декілька фігур. 3. Заповнити кубами одну горизонтальну площину.	За кожну установлену фігуру гра нарахувала 5 ігрових очок. За одну заповнену площину: 100 очок.	Так
10.	Завершення гри	1. Запустити ігровий процес. 2. Встановлювати фігури одну на одну доти, поки чергова установ-лена фігура не з’явиться за межами верхньої межі поля.	Запустилася анімація закінчення гри, після якої на екрані з'явилося меню закінчення гри. В даному меню відображається набраний в ході гри рахунок.	Так
11.	Підвищення швидкості гри	1. Запустити ігровий процес. 2. Грати на протязі приблизно однієї хвилини.	Швидкість гри підвищилась з 1,0 до 1,1, а після до 1,2.	Так
12.	Зниження швидкості гри	1. Запустити ігровий процес. 2. Грати до тих пір, поки швидкість гри не стане рівною 1.5 або вище. Заповнити одну горизонтальну площину.	Швидкість гри знизилась на 0,5.	Так
13.	Збереження максимального рекорду	1. Запустити ігровий процес. 2. Грати до тих	На екрані закінчення гри відображається	Так

		пір, поки поточний рахунок не поб'є максимальний рекорд. 3. Завершити ігровий процес. 4. Запустити знову гру. 5. Запустити ігровий процес. 6. Завершити ігровий процес.	новий максимальної ний рекорд.	

6.2 Юзабіліті тестування

Юзабіліті тестування проводилося за участю одногрупників. В ході даного тестування було визначено деякі параметри гри, а також додані нові, які було визначено на етапі проектування. Йдеться про параметри, що впливають на складність гри, а також на зручність ігрового процесу, які неможливо точно визначити до етапу реалізації. Нижче наведено список змін, внесених в гру після реалізації.

1. Підвищення швидкості гри. Було вирішено, що швидкість гри повинна підвищуватися на 0,1 кожні 25 секунд. Якщо швидкість гри буде підвищуватися частіше, в гру стає занадто складно грати, якщо рідше - занадто просто. Так що 25 секунд - це найоптимальніший варіант, відповідний більшій кількості гравців.

2. Зниження швидкості гри. Спочатку швидкість гри повинна була тільки підвищуватися. В ході тестування було виявлено, що при такому ігровому процесі кращої тактикою буде встигнути заповнити якомога більше одиночних горизонтальних площин, поки швидкість гри не стане занадто високою. Для того, щоб дати гравцеві мотивацію ретельніше продумувати свої дії і заповнювати відразу кілька горизонтальних площин, було введено додаткове заохочення у вигляді зниження

швидкості гри за заповнення площин. Чим більше площин одночасно заповнить гравець, тим сильніше знизиться швидкість гри.

3. Зміна інтерфейсу ігрового процесу. Було вирішено прибрати з даного інтерфейсу пункти «Режим камери», і «Режим фігури». Дані пункти не надавали будь-якої корисної інформації і лише відволікали увагу гравця, адже гравець і без цих підказок знає, що, якщо він затиснув клавішу L.Shift, то фігура знаходиться в режимі обертання, а інакше, в режимі переміщення. Крім цього, було вирішено перемістити підказку про наступну фігуру в правий нижній кут інтерфейсу, так як в лівій частини і так вистачало різної інформації, а ось права частина була майже порожня. У додатку Б представлені скріншоти гри після реалізації та проведення тестування.

Висновок до розділу

В ході тестування були підготовлені і проведені тести для функціонального тестування, а також проведено юзабіліті тестування. Всі тести були успішно пройдені - реалізована комп'ютерна гра повністю відповідає всім своїм вимогам.

ВИСНОВОК

В ході виконання випускної кваліфікаційної роботи була розроблена комп'ютерна гра «Tetris 3D» на платформі Unity.

Основні результати роботи.

1. Проведено аналіз предметної області. На основі виявлених плюсів і мінусів у аналогічних проектів для розроблюваної гри були сформовані додаткові ігрові механіки.

2. Проведено аналіз функціональних та нефункціональних вимог до системи і складена діаграма варіантів використання.

3. Спроектована архітектура комп'ютерної гри і описані всі компоненти системи. Також, складені макети ігрових меню та користувальницьких інтерфейсів.

4. Реалізована комп'ютерна гра «Tetris 3D». Складено докладний опис реалізації всіх компонентів гри. Також, представлені кількісні метрики написаних для реалізації гри класів.

5. Проведено функціональне і юзабіліті тестування. Всі тести успішно пройдені.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. А.П. Еремеев, А.А. Кожухов «Реализация методов обучения с подкреплением на основе темпоральных различий и мультиагентного подхода для интеллектуальных систем реального времени» // Программные продукты и системы 2017. № 1 Стр. 28-33.
2. Википедия. Генетический алгоритм URL: https://ru.wikipedia.org/wiki/Генетический_алгоритм
3. Голомб С. В. Полимино – Москва: Мир, 1975 г. – 207 с.
4. Гурьянов А.К. Стратегии исследования окружений в обучении с подкреплением с непрерывными пространствами состояний: дис. магистр — Москва, 2017. — С. 38.
5. Д. Б. Рохлин, “Q-обучение в стохастической игре Штакельберга между неинформированным лидером и наивным ведомым”// Теория вероятн. и ее примен., 2019. №64:1 Стр.53–74
6. Д.А .Коробов, С.А. Беляев «Современные подходы к обучению интеллектуальных агентов в среде ATARI» // Программные продукты и системы 2018. № 2 Стр. 284-290.
7. Иван Подсекальников. Red Dead Redemption 2 заняла в истории индустрии развлечений 2-е место по кассовым сборам за первые три дня продаж. [Электронный ресурс] URL: http://gadgets-news.ru/red-dead- redemption-2/
8. Игровая индустрия в России и мире. Исследования Mail.Ru Group URL: https://gamestats.mail.ru/
9. Машинное обучение. URL: www.machinelearning.ru/wiki/index.php?title=Маши нное_обучение
10. Официальный сайт игры «Red Dead Redemption 2».

[Электронный ресурс] URL: https://www.rockstargames.com/reddeadredemption2/

11. Рудь М.Н. Разработка системы управления роботом-гексаподом: дис. магистр — Томск, 2016. — С. 103. 11.С. А. Князятов, Г. Г. Малинецкий, “Решение задачи распознавания блефа в игре «верю – не верю» с помощью алгоритмов обучения с подкреплением” // Препринты ИПМ им. М. В. Келдыша. 2018. №170. Стр. 21.
12. Среда разработки Unity. [Электронный ресурс] URL:https://unity.com/ru
13. Теория случайных процессов / А. В. Булинский, А. Н. Ширяев. — Физматлит, 2005. — 408 с. — ISBN 5- 9221-0335-0.
14. Ю.И.Еременко, Е.Г.Доронина «Модель адаптивного поведения агентов мультиагентной системы управления экологической безопасность» // Прикладная информатика 2010. № 2(26) Стр. 71-82.
15. Kent L., Steven. The Ultimate History of Video Games — New York: Three Rivers Press, 2001 г. – 608 с.
16. Страница браузерной игры «Block Out». [Электронный ресурс] URL: https://playminigames.ru/game/block-out-3d-tetris-3d-tetris
17. Официальный сайт игры «Tricky Towers». [Электронный ресурс] URL: https://www.trickytowers.com/ (дата обращения 10.02.2021 г.).
18. Официальный сайт игры «Tetris 99». [Электронный ресурс]URL: https://tetris99.nintendo.com/ (дата обращения 10.02.2021 г.).
19. Документация по C#. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/
20. Habbs. Level Of Detail (LOD). [Электронный ресурс] URL:

http://www.gamedev.ru/articles/?id=60001

21. Сергей Кормишин. Что такое шейдеры? Просто о сложном для начинающих. [Электронный ресурс] URL: https://coremission.net/gamedev/chto-takoe-sheidery/
22. Сергей Кормишин. Партиклы – система частиц. [Электронный ресурс] URL: https://coremission.net/gamedev/partikly-sistema-chastits/
23. Anthony D'Alessandro. «Avengers: Infinity War» Marches Toward
24. McGuire, Morgan; Jenkins, Odest Chadwicke. Creating Games: Mechanics, Content, and Technology – Wellesley, Massachusetts: A K Peters, 2009г. – С. 22–23.
25. Reuters – Investing in the Soaring Popularity of Gaming. [Электрон- ный ресурс] URL: https://www.reuters.com/article/sponsored/popularity-of- gaming
26. Skyler Miller. The History of Puzzle Games: Tetris. GameSpot UK. [Электронный ресурс] URL: https://web.archive.org/web/20071015102031/http://www.gamespot.com:80/feat ures/vgs/universal/puzzle_hs/
27. Unity User Manual. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/index.html
28. Vadim Gerasimov. Tetris Story. [Электронный ресурс] URL: https://vadim.oversigma.com/Tetris.htm

ДОДАТКИ

Додаток А.

Специфікація варіантів використання

У таблицях 1-7 наведені специфікації основних варіантів використування.

Таблиця 1 - Специфікація варіанту використання «Запустити рівень»

Варіант використання: запустити рівень
ID: 1
Анотація: запуск рівня з ігровим процесом
Головні актори: гравець
Другорядні актори: немає
Передумови: немає
Основний потік: 1. Варіант використання починається, коли гравець натискає на кнопку «Грати». 2. Система завантажує рівень з порожнім ігровим полем і починає ігровий процес.
Післяумови: немає
Альтернативні потоки: немає

Таблиця 2 - Технічна специфікація варіанту використання «Грати»

Варіант використання: грати
ID: 2
Анотація: основний ігровий процес
Головні актори: гравець
Другорядні актори: немає
Передумови: гравець запустив рівень
1. Варіант використання починається, коли завершується варіант використання 1. 2. Система створює фігуру, яка поступово рухається вниз. 3. Гравець може переміщати і обертати фігуру, рухати камеру і робити зріз. 4. Коли фігура, при русі вниз, натикається на яку-небудь перешкоду, система зупиняє цю фігуру: 4.1. Якщо зупинена фігура не виходить за верхню межу ігрового поля – повертається на етап 2. 4.2. Інакше - перехід до етапу 5. 5. Система виводить на екран повідомлення про завершення гри і кількість набраних очок. 6. Гравець може почати гру заново або повернутися в головне меню гри.
Післяумови: якщо гравець побив свій попередній рекорд, новий рекорд записується в файл
Альтернативні потоки: якщо на етапі 4.1 система виявляє одну або кілька повністю заповнених вертикальних площин - ці площини видаляються, а всі куби, що знаходяться вище, спускаються вниз

Продовження додатку А

Таблиця 3 - Специфікація варіанту використання «Відкрити конструктор наступної фігури»

Варіант використання: відкрити конструктор наступної фігури
ID: 3
Анотація: відкриття спеціального вікна для редагування наступної фігури
Головні актори: гравець
Другорядні актори: немає
Передумови: гравець почав гру, але ще не завершив її
Основний потік: 1. Варіант використання починається, коли гравець натискає на кнопку відкриття конструктора. 2. Система уповільнює темп гри (швидкість падіння фігури) і відкриває вікно конструктора. 3. Гравець може додавати до фігури нові куби і видаляти їх, а також обертати саму фігуру.
Післяумови: система зберігає нову форму наступної фігури
Альтернативні потоки: немає

Таблиця 4 - Специфікація варіанту використання «Відкрити меню паузи»

Варіант використання: відкрити меню паузи
ID: 4
Анотація: відкриття меню паузи
Другорядні актори: немає
Передумови: гравець почав гру, але ще не завершив її або коли гравець відкрив конструктор наступної фігури
Основний потік: 1. Варіант використання починається, коли гравець натискає на кнопку відкриття меню паузи. 2. Система ставить ігровий процес на паузу і виводить на екран меню паузи. 3. Гравець може повернутися в головне меню, зайти в налаштування гри, відкрити схему управління або вийти з гри.
Післяумови: немає
Альтернативні потоки: немає

Продовження додатку А

Таблиця 5 - Специфікація варіанту використання «Повернутися в головне меню гри»

Варіант використання: повернутися в головне меню гри
ID: 5
Анотація: повернення гравця в головне меню гри
Головні актори: гравець
Другорядні актори: немає
Передумови: гравець завершив основний ігровий процес або відкрив меню паузи
Основний потік: 1. Варіант використання починається, коли гравець натискає на кнопку повернення в головне меню. 2. Система виводить на екран головне меню гри. 3. Гравець може запустити рівень, зайти в налаштування гри, відкрити схему управління або вийти з гри.
Післяумови: немає
Альтернативні потоки: немає

Таблиця 6 - Специфікація варіанту використання «Змінити ігрові настройки»

Варіант використання: змінити ігрові налаштування
ID: 6
Анотація: зміна налаштувань гри
Головні актори: гравець
Другорядні актори: немає
Передумови: гравець знаходиться в головному меню або в меню паузи
Основний потік: 1. Варіант використання починається, коли гравець натискає на кнопку відкриття настройки. 2. Система виводить на екран меню налаштувань. 3. Гравець може змінити налаштування гри: якість графіки, дозвіл вікна, гучність звуку і т.д.
Післяумови: система зберігає нові налаштування в файл
Альтернативні потоки: немає

Закінчення додатку А

Таблиця 7 - Специфікація варіанту використання «Відкрити схему управління»

Варіант використання: відкрити схему управління
ID: 7
Анотація: перегляд схеми управління
Головні актори: гравець
Другорядні актори: немає
Передумови: гравець знаходиться в головному меню або в меню паузи
Основний потік: 1. Варіант використання починається, коли гравець натискає на кнопку відкриття схеми управління. 2. Система виводить на екран схему управління. 3. Гравець може дізнатися, які дії йому доступні і на які при цьому клавіші необхідно натискати.
Післяумови: немає
Альтернативні потоки: немає

Додаток Б

Скріншоти підсумкової версії гри
На рисунках 1-5 представлені скріншоти гри «Tetris 3D».

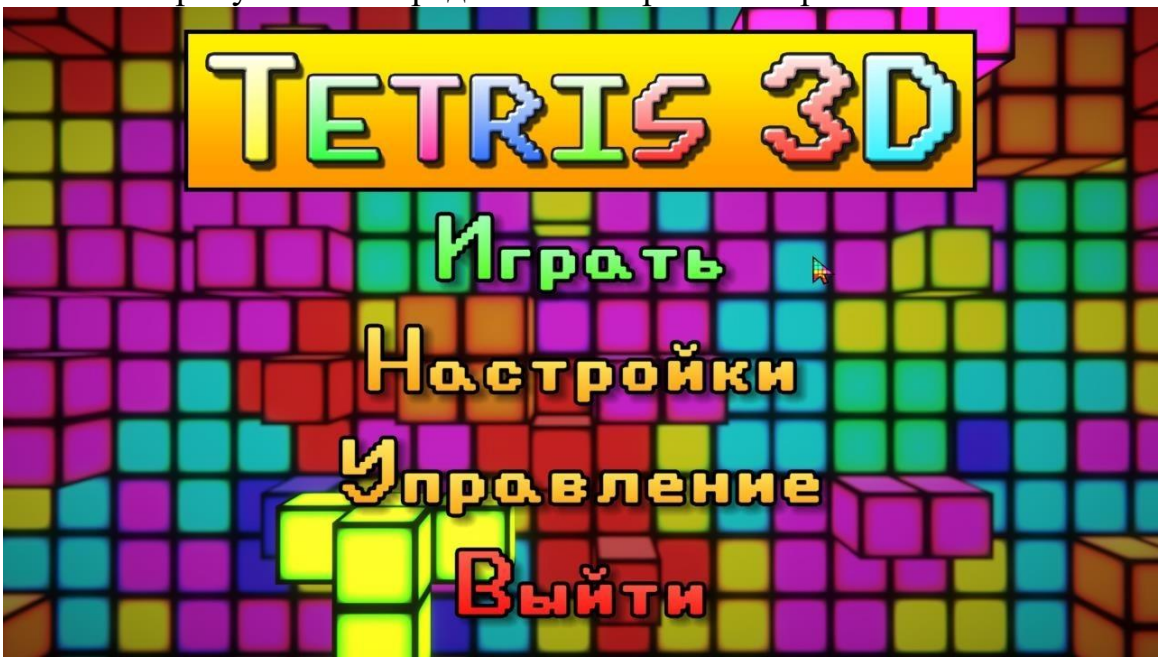

Рисунок 1 - Головне меню гри

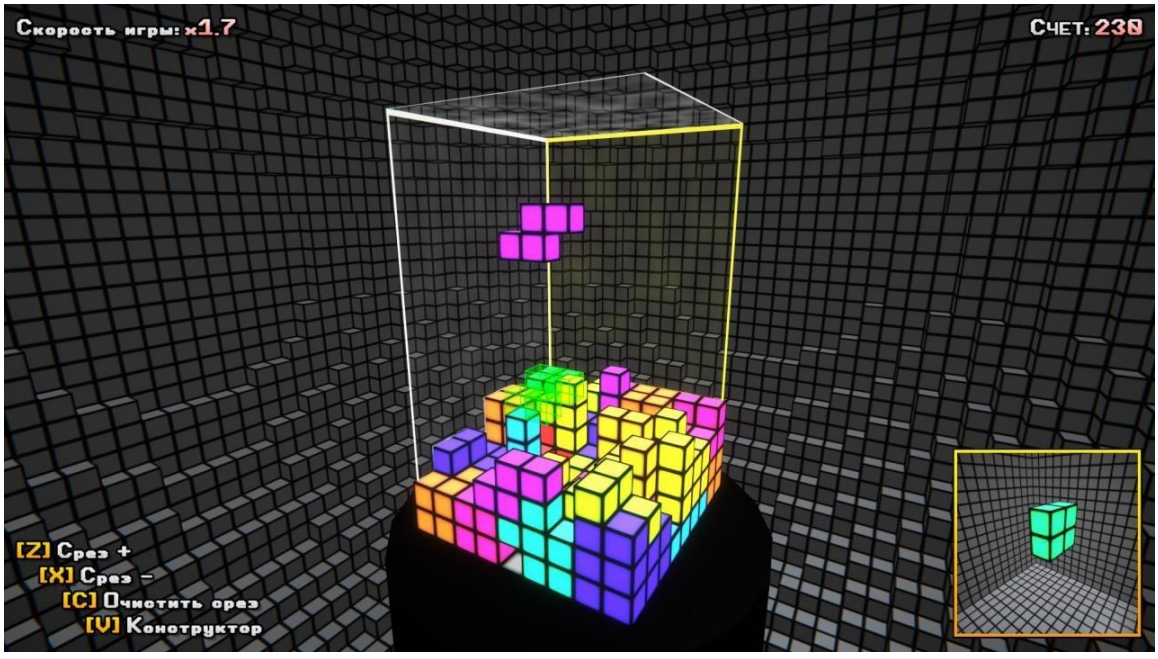

Рисунок 2 - Режим ігрового поля

Продовження Додатку Б

Рисунок 3 – Меню паузи

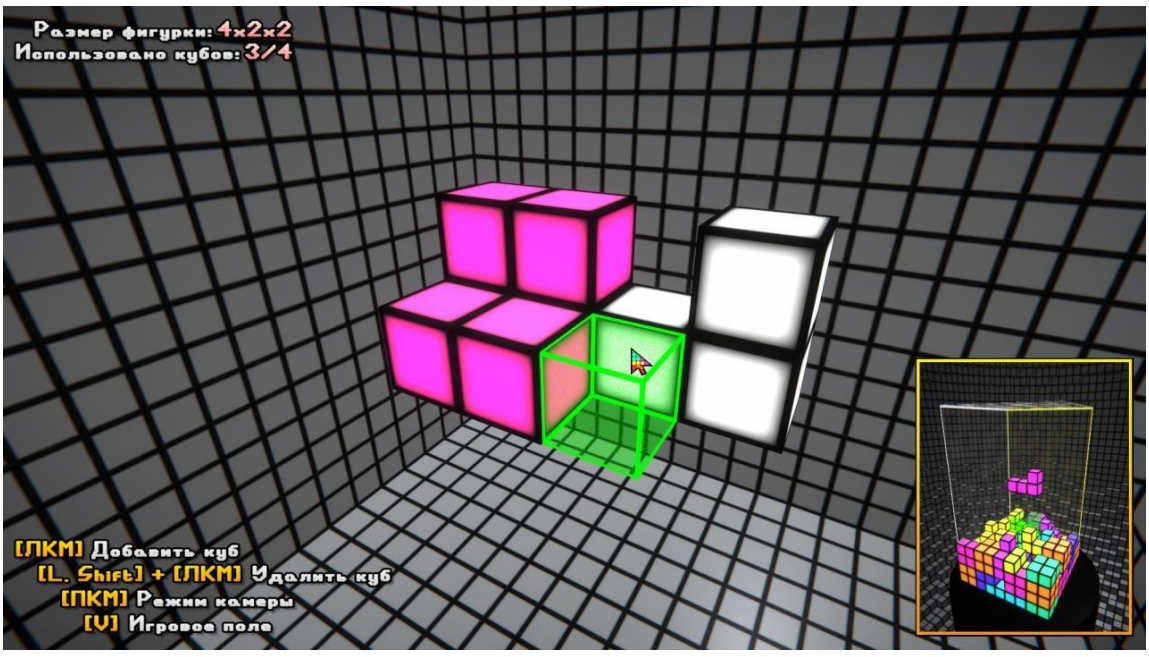

Рисунок 4 – Режим конструктора

Закінчення додатку Б

Рисунок 5 – Меню закінчення гри

Додаток В

Додаток 1

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Шувалов Андрій Кирилович, учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

18.10.2021 (дата) _____ (підпис) Андрій Шувалов (ім'я, прізвище)