Міністерство освіти і науки України
Прикарпатський національний університет імені Василя Стефаника
кафедра комп’ютерних наук та інформаційних систем

БАКАЛАВРСЬКА РОБОТА

на тему: Гейміфікована система підтримки вивчення математики в середній школі

Виконав: студент 4 курсу гр.ІСТ-41
Спеціальності 126 Інформаційні системи та технології
Сем’яник Мирослав-Іван Мирославович
Керівник: к.т.н., старший викладач, Ізмайлов Артем Вікторович
Рецензент: старший викладач Никорак Я.Я

м. Івано-Франківськ – 2024

Анотація

В цьому дослідженні пропонується гейміфікована система підтримки вивчення математики для учнів середньої школи. Система допомагає учням краще засвоїти матеріал, розвинути творче мислення та отримувати більше задоволення від навчання. Досягається це за допомогою елементів дизайну та геймплею відеоігор, які інтегровані в навчальне середовище.

Розглядаються декілька ігор, де навчальний досвід вбудований в ігровий контекст, і впроваджують гейміфікацію в навчальний процес. Аналіз існуючих рішень виявив у них системні недоліки.

Додатково проведено аналіз існуючих інструментів, таких як ігрові рушії, мови програмування та інші технології, що використовуються для розробки гейміфікованих додатків.

Розроблений додаток, який забезпечує гейміфікацію процесу вивчення математики в середній школі.

Дипломна робота містить 31 рисунок, 22 посилання, та 2 додатки

Автор є призером Міжнародної науково-технічної конференції здобувачів вищої освіти та молодих вчених “Комп’ютерні науки, інформаційні технології та системи управління”.

Ключові слова: Гейміфікація, Математика, інтерактивне навчання, Середня школа, Система.

Abstract

This study proposes a gamified support system for learning mathematics aimed at middle school students. The system helps students better understand the material, develop creative thinking, and derive more enjoyment from their studies. This is achieved through the integration of video game design elements and gameplay into the educational environment.

Several games are examined where the learning experience is embedded in a gaming context, implementing gamification in the educational process. An analysis of existing solutions revealed systemic shortcomings.

Additionally, an analysis of existing tools, such as game engines, programming languages, and other technologies used for developing gamified applications, is conducted.

A developed application provides gamification of the mathematics learning process in middle school.

This thesis contains ... figures, ... references, ... and 2 appendices.

The author is a prize-winner of the International Scientific and Technical Conference for Higher Education Students and Young Scientists "Computer Science, Information Technologies and Control Systems".

Keywords: Gamification, Mathematics, Interactive Learning, Middle School, System.

Зміст

Вступ

Актуальність

Сучасний освітній процес активно інтегрує інформаційні технології, що суттєво змінюють підходи до навчання, а гейміфікація застосовується для підвищення мотивації та залучення учнів через використання ігрових елементів у навчальних програмах. Математика, будучи фундаментальною наукою, часто викликає труднощі у засвоєнні через абстрактність та складність матеріалу, що призводить до недостатньої мотивації і зацікавленості учнів, а відтак до низької успішності та негативного ставлення до предмету. Використання гейміфікації у навчанні математики може значно покращити ситуацію, стимулюючи активну участь учнів у навчальному процесі та підвищуючи їхню зацікавленість через ігрові механізми, такі як бали, рівні, нагороди і конкуренція. Гейміфікація навчання математики має потенціал для створення більш привабливого та ефективного освітнього середовища, дозволяючи підвищити залученість учнів, зробити навчання більш інтерактивним та наочним, що сприяє кращому розумінню та запам’ятовуванню матеріалу. Водночас важливо підходити до цього процесу науково обґрунтовано, аналізуючи існуючі методи та інструменти, визначаючи їхні переваги та недоліки, адаптуючи їх до специфіки навчання математики у середній школі.

Мета

Метою даної дипломної роботи є розробка гейміфікованої системи підтримки вивчення курсу математики у середній школі, що сприятиме

підвищенню мотивації учнів та ефективності засвоєння ними знань. Для досягнення цієї мети необхідно вирішити такі **завдання**:

1. Провести аналіз сучасних методів і засобів гейміфікації у навчанні, зокрема у вивченні математики.
2. Визначити переваги та недоліки наявних підходів до гейміфікації в освітньому процесі.
3. Розробити модель гейміфікованої системи, адаптовану до потреб учнів середньої школи, що включатиме основні елементи та механізми гейміфікації.
4. Реалізувати запропоновану модель у вигляді програмного продукту та провести його тестування.

Об’єктом дослідження є гейміфікація процесу вивчення математики у середній школі. Предметом дослідження виступає гейміфікована система підтримки цього процесу, яка включає елементи ігрового дизайну, інтерактивні завдання та мотиваційні механізми.

Практична цінність роботи полягає у створенні ефективного інструменту для вчителів математики, який допоможе підвищити зацікавленість учнів у навчанні та покращити їхні результати.

Застосовано кілька методів для досягнення поставленої мети та виконання завдань: аналіз інформаційних джерел, що включає детальний огляд наукової літератури, статей, досліджень та інших матеріалів, що стосуються гейміфікації у навчанні математики; тестування програмного забезпечення, проведене для оцінки ефективності, зручності використання та впливу на мотивацію і успішність учнів; а також порівняльний аналіз результатів тестування з існуючими аналогами, що дозволило визначити сильні та слабкі сторони розробленого додатку.

Основна частина роботи включає кілька розділів:

У першому розглядаються теоретичні основи гейміфікації у навчанні, її роль та значення, а також приклади використання у навчанні математики, огляд та порівняння існуючих гейміфікованих додатків, визначено їхні переваги та недоліки.

Другий розділ описує методологію розробки додатку, використані інструменти та технології.

Третій розділ присвячено реалізації та тестуванню додатку, його ефективності та зручності використання.

РОЗДІЛ 1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Методи навчання математики у середній школі

Математика є фундаментальною наукою, яка базується на логіці, абстракції, доказах і строгих математичних структурах. Математика є мовою, за допомогою якої висловлюються закономірності, відносини та моделюються реальні процеси [1,2].

Математика поділяється на безліч галузей, від арифметики та геометрії до алгебри, аналізу та топології. Кожна з цих галузей має свої особливості та застосування. Хоча математика здебільшого асоціюється зі складними формулами та абстрактними концепціями, її застосування в повсякденному житті є надзвичайно широким та різноманітним [3]. Вивчення математики є важливим для розвитку різних навичок [4] і практичного застосування в багатьох сферах життя. Ось кілька аргументів, які підкреслюють значущість вивчення математики:

1. Розвиток Математичних Навичок:

   Вивчення математики вдосконалює здатність до обчислень, аналізу даних та вирішення складних задач, що є основою для багатьох академічних і професійних дисциплін.

2. Просторове Мислення:

   Математика допомагає розвивати просторове мислення, необхідне для розуміння та роботи з формами, розмірами і відстанями, що є критичним у багатьох інженерних та наукових галузях.

3. Критичне та Логічне Мислення:

Математичні задачі вимагають аналітичного підходу, логічної послідовності та здатності до критичного аналізу, що сприяє розвитку цих важливих когнітивних навичок.

4. Підготовка до Професійних Галузей:

   Математика є фундаментом для багатьох професій, таких як інженерія, економіка, комп'ютерні науки та медицина, забезпечуючи базові знання та навички для успішної кар'єри.

5. Креативність та Дизайн:

   Вивчення математики стимулює креативність, оскільки часто вимагає пошуку нестандартних підходів до розв'язання задач, що важливо в дизайні, архітектурі та мистецтві.

Вивчення математики у середній школі може здійснюватися за допомогою різноманітних методів [5], спрямованих на стимулювання зацікавленості учнів і полегшення їх розуміння математичних концепцій. Ось деякі з них:

Традиційні Викладацькі Методи:

Учитель пояснює новий матеріал, демонструючи його на дошці через розв'язання задач і вправ, що дозволяє учням спостерігати за процесом та методами розв'язання. Учні виконують математичні завдання в групах або індивідуально, що сприяє активному освоєнню та застосуванню отриманих знань.

Переваги:

- Дозволяє перевірити рівень засвоєння матеріалу.
- Допомагає виявити слабкі місця у знаннях учнів.

Недоліки:

- Може викликати стрес у деяких учнів.

- Не завжди відображає справжній рівень знань через можливість помилок або невпевненості учнів під час тестування.

Інтерактивні Методи:

Використання онлайн-тестів та інтерактивних засобів оцінювання для перевірки рівня засвоєння матеріалу та надання зворотного зв'язку учням, а також проведення інтерактивних дослідів та проектів, що дозволяють учням застосовувати математичні знання до реальних ситуацій та вирішувати реальні проблеми.

Переваги:

- Зворотний зв'язок: Використання онлайн-тестів та інтерактивних засобів оцінювання дозволяє швидко отримувати зворотний зв'язок щодо рівня засвоєння матеріалу, що допомагає вчителю коригувати навчальний процес.
- Практичне застосування: Інтерактивні досліди та проекти дають можливість учням застосовувати математичні знання в реальних ситуаціях, що підвищує їхню мотивацію та інтерес до предмету.

Недоліки:

- Технічні вимоги: Використання онлайн-тестів та інтерактивних засобів оцінювання вимагає наявності комп'ютерів та доступу до Інтернету, що може бути проблемою для деяких шкіл.
- Складність організації: Проведення інтерактивних дослідів та проектів потребує значних зусиль та часу на підготовку з боку вчителя.
- Стрес для учнів: Онлайн-тести можуть викликати стрес у деяких учнів, що може негативно впливати на їхні результати.

Гейміфіковані Підходи:

Учні мають можливість розв'язувати завдання та виклики, які структуровані у вигляді ігрових рівнів або місій, що поступово зростають у складності. Це відбувається в навчальному середовищі, яке нагадує гру або пригоду, з використанням тематичних елементів, персонажів та історій, що робить навчання цікавим та захоплюючим.

Переваги:

1. **Мотивація та залучення**: Ігрова атмосфера та поступове ускладнення завдань підвищують мотивацію учнів, роблячи навчання цікавим та захоплюючим.
2. **Поетапне навчання**: Структуровані рівні та місії дозволяють учням поступово освоювати новий матеріал, що сприяє кращому засвоєнню знань.
3. **Розвиток навичок вирішення проблем**: Завдання та виклики стимулюють учнів до аналізу, стратегічного мислення та вирішення проблем.
4. **Творчий підхід**: Використання тематичних елементів, персонажів та історій сприяє розвитку уяви та творчого підходу до навчання.

Недоліки:

1. **Ресурсоємність**: Створення ігрової атмосфери та розробка відповідних завдань вимагають значних зусиль, часу та ресурсів з боку вчителя.
2. **Відволікання**: Ігрові елементи можуть відволікати учнів від основної мети навчання, якщо не будуть правильно інтегровані у навчальний процес.

3. **Обмеження тематики**: Деякі математичні теми можуть бути складними для адаптації до ігрової форми, що обмежує застосування такого підходу.

Проектно-Орієнтовані Методи:

Учні працюють у групах, співпрацюючи та взаємодіючи для досягнення спільних цілей, що розвиває навички комунікації та співпраці, а також беруть участь у проектах, орієнтованих на реальні життєві ситуації або проблеми, що дозволяє їм бачити зв'язок між навчанням та практикою.

Переваги:

- Розвиток комунікативних навичок: Спільна робота в групах сприяє розвитку навичок комунікації та співпраці серед учнів.
- Практичне застосування знань: Проекти, орієнтовані на реальні життєві ситуації або проблеми, допомагають учням бачити зв'язок між навчанням та практикою.
- Розвиток критичного мислення: Реальні проблеми вимагають від учнів аналізу, оцінки різних варіантів та прийняття обґрунтованих рішень.
- Мотивація та залучення: Застосування знань у практичних проектах підвищує мотивацію учнів до навчання.

Недоліки:

- Складність координації: Організація групової роботи та координація діяльності учнів може бути складною і вимагати додаткового часу.

- Нерівномірний внесок: У групах можуть виникати ситуації, коли одні учні виконують більше роботи, ніж інші, що може призвести до конфліктів та незадоволення.
- Ресурсоємність: Проекти, орієнтовані на реальні життєві ситуації, часто потребують додаткових ресурсів, матеріалів та часу.
- Різний рівень підготовки: Учні з різним рівнем підготовки можуть відчувати труднощі у виконанні складних проектів, що може знизити їхню ефективність.

В результаті аналізу встановлено що гейміфікація є ефективним підходом до навчання завдяки своїй здатності створювати мотивацію та захоплення серед учнів. Ігрові елементи, такі як нагороди, рівні, бейджі та змагання, стимулюють активну участь, підвищують зацікавленість та забезпечують ефективне засвоєння матеріалу. Цей підхід сприяє покращенню результатів навчання та створює стимулюючу атмосферу для навчання.

1.2 Аналіз існуючих рішень

В сучасному освітньому середовищі існує ряд засобів розробки програмного забезпечення, спрямованих на навчання математики. Деякі з них використовують інтерактивні підходи та гейміфікацію, щоб зробити навчання більш захоплюючим та ефективним.

1.2.1 **Prodigy Math Game**(рис. 1.1) [6]:

Опис: Prodigy Math Game - це онлайн-гра, яка поєднує в собі елементи RPG та математичні завдання [6].

Методологія: Гравці просуваються через ігровий світ, розв'язуючи математичні завдання для боротьби з монстрами та виконання завдань.

Переваги:

- Привабливий для учнів, мотивує до навчання.

- Відстежує прогрес учнів.
- Надає індивідуалізовані завдання.

Недоліки:

- Може бути використаний виключно як додатковий ресурс.
- Не завжди відповідає вимогам навчальної програми.

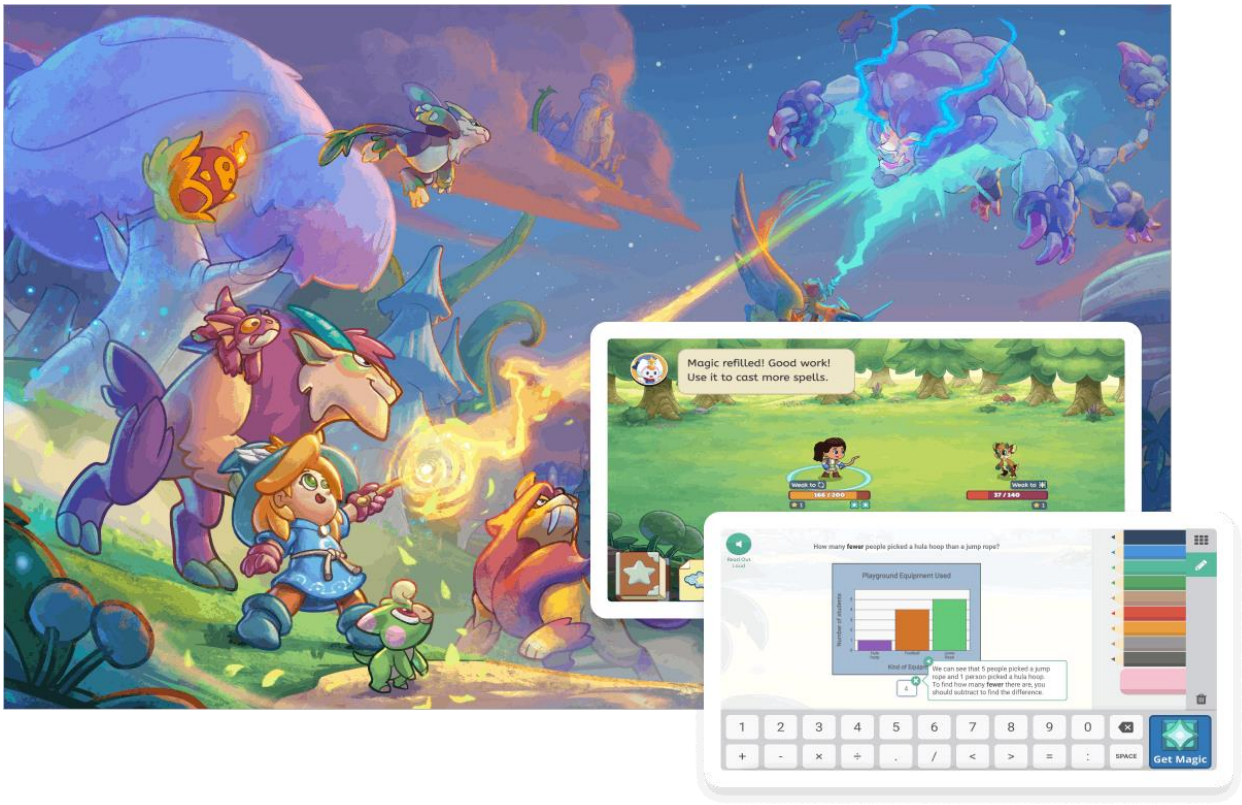

Рисунок 1.1 - Prodigy Math Game

1.2.2 **Khan Academy** (рис. 1.2) [7]:

Опис: Khan Academy - це безкоштовна онлайн-платформа для навчання, яка пропонує відеоуроки, вправи та тести з різних предметів, включаючи математику [7].

Методологія: Учні можуть самостійно вивчати матеріал, виконуючи вправи та переглядаючи відеоуроки на різні теми.

Переваги:

- Широкий вибір матеріалів.
- Можливість навчання власним темпом.
- Індивідуалізація навчання.

Недоліки:

- Відсутність живого вчителя.
- Може виявитися недостатнім для деяких учнів.

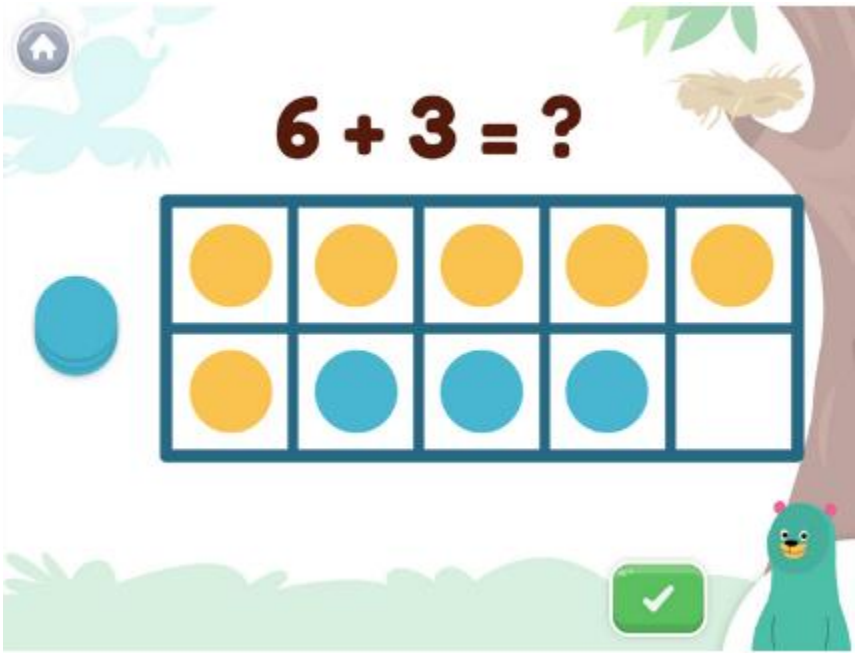

Рисунок 1.2 - Khan Academy

1.2.3 **DragonBox Algebra** (рис. 1.3) [8]:

Опис: DragonBox Algebra - це навчальна гра, яка використовує гейміфікацію для навчання алгебри [8].

Методологія: Гравці виконують різні завдання та головоломки, які поступово навчають основам алгебричних операцій.

Переваги:

- Залучає увагу учнів.
- Використовує інноваційні методи навчання.

- Допомагає зрозуміти складні математичні концепції.

Недоліки:

- Може бути не досить різноманітним для деяких учнів.
- Може вимагати додаткових пояснень.

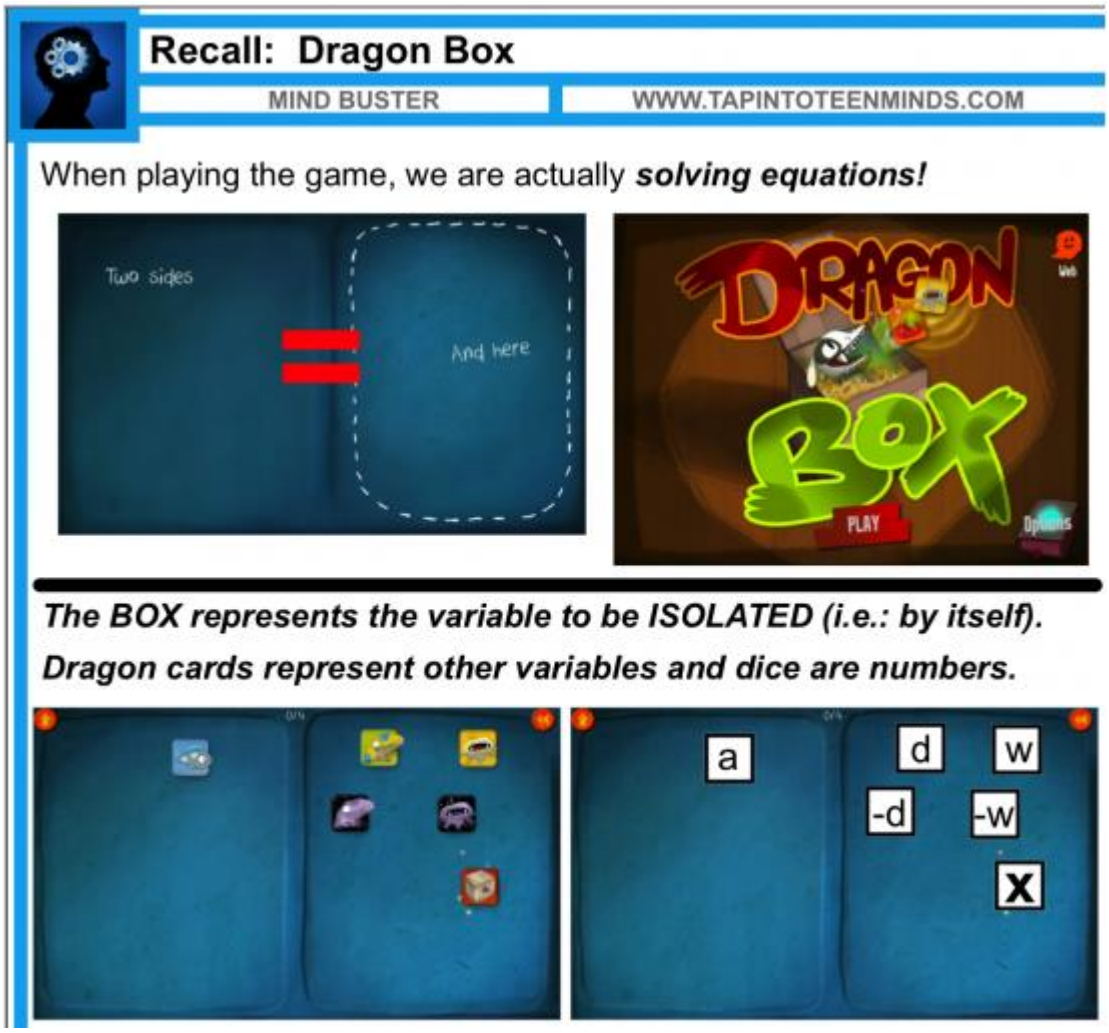

Рисунок 1.3 - DragonBox Algebra

1.2.4 **Mathletics**(рис 1.4) [9]:

Опис: Mathletics - це онлайн-ресурс для навчання математики, який містить в собі вправи, тести та інтерактивні завдання [9].

Методологія: Учні працюють над завданнями та вправами, заробляючи бали та змагаючись з іншими користувачами.

Переваги:

- Широкий вибір вправ та завдань.
- Можливість змагатися з іншими учнями.
- Мотивуюча система нагород.

Недоліки:

- Може бути важким для деяких учнів.
- Відсутність індивідуалізації навчання.

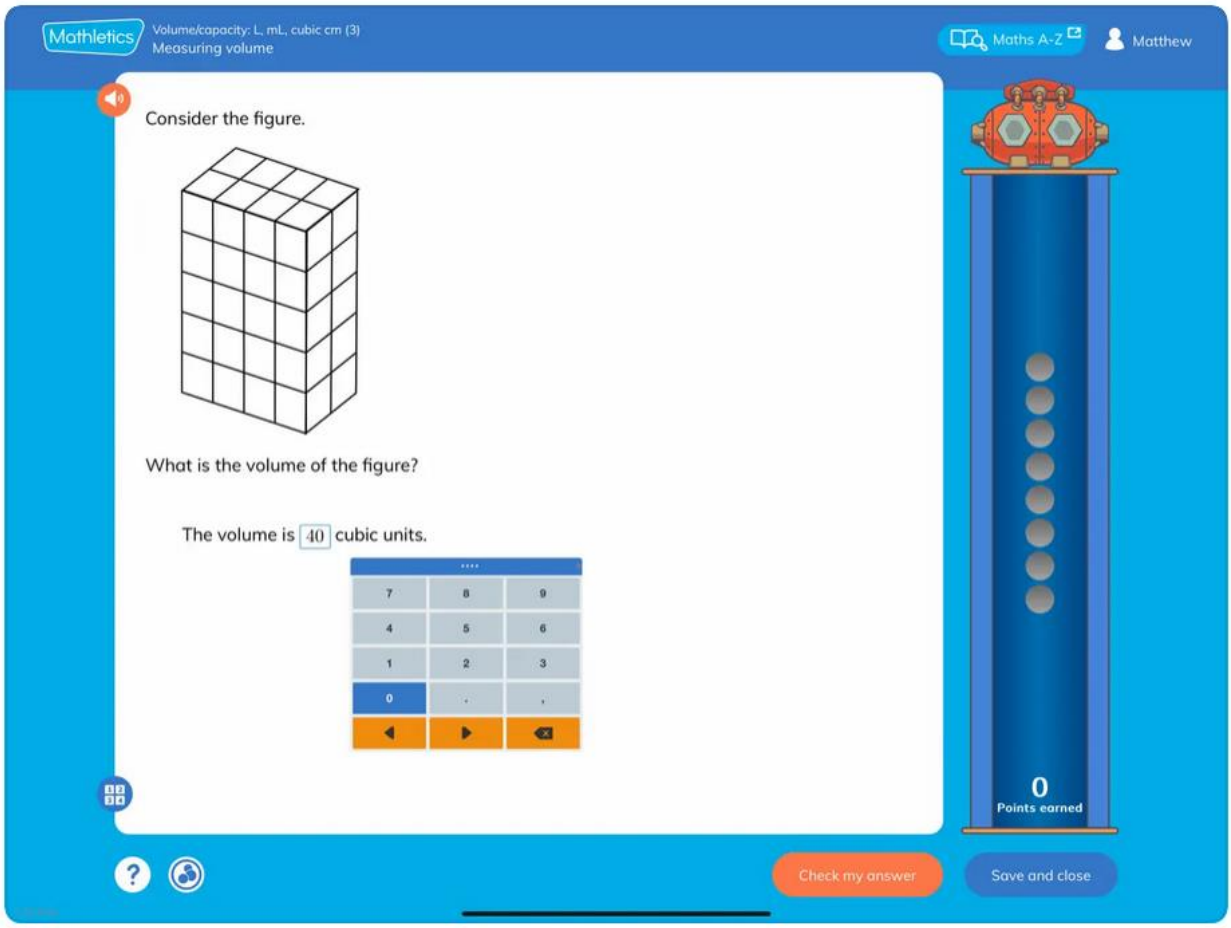

Рисунок 1.4 - Mathletics

1.2.5 ST Math (рис. 1.5) [10]:

Опис: ST Math - це програма для навчання математики, яка використовує візуальні головоломки та завдання для розвитку математичного мислення [10].

Методологія: Учні вирішують різні головоломки та завдання, які представлені у вигляді візуальних пазлів та задач.

Переваги:

- Заснований на доказаних методах навчання.
- Відповідає вимогам навчальних програм.
- Розвиває логічне мислення.

Недоліки:

- Може бути складним для деяких учнів.
- Відсутність можливості індивідуалізації завдань.

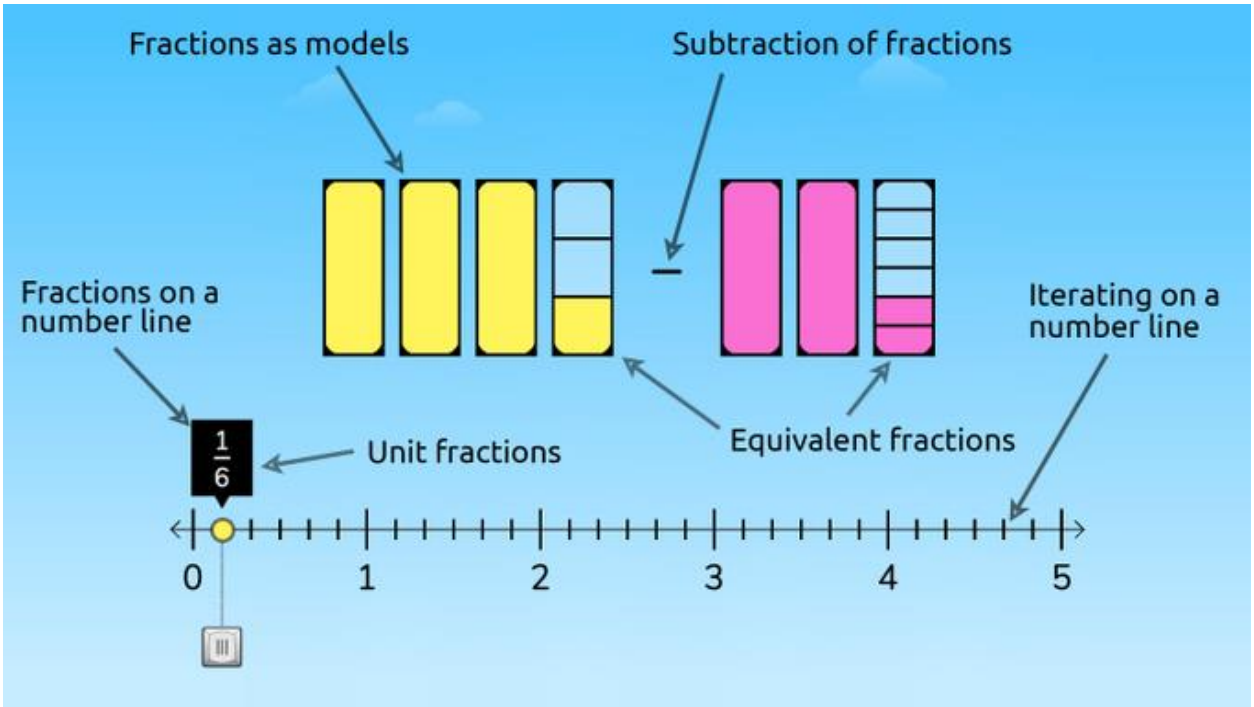

Рисунок 1.5 - ST Math.

1.2.5 Результати аналізу

Проаналізувавши програмні забезпечення, такі як Prodigy Math Game, Khan Academy, DragonBox Algebra, Mathletics та ST Math, можна відзначити їх переваги та недоліки. Prodigy Math Game привертає увагу учнів та заохочує до навчання, але може виявитися складним для початківців. Khan Academy пропонує широкий спектр матеріалів і можливість навчання власним темпом, але відсутність вчителя може бути проблемою для деяких учнів. DragonBox Algebra залучає увагу та допомагає зрозуміти складні математичні концепції, але може бути недостатньо різноманітним для деяких користувачів. Mathletics пропонує індивідуалізовані завдання та можливість змагатися з іншими, але може бути важким для деяких учнів. ST Math успішно демонструє фізичні концепції, але може бути недостатньою для вивчення математики.

Кожен має свої позитивні та негативні аспекти, і вибір конкретного залежить від вікової групи учнів, конкретних навчальних цілей та вподобань вчителя. Успішна реалізація програмного забезпечення для навчання математики вимагає уважного підходу та врахування особливостей конкретного навчального середовища.

1.3 Постановка завдачі дослідження

Задачою дослідження є розробка системи гейміфікованого навчання математики в середній школі, яка має на меті досягнення кількох ключових цілей, спрямованих на поліпшення якості освіти та створення захопливого навчального середовища для учнів.

Підзадачі

1. Проведення дослідження існуючих додатків для інтерактивного навчання математики.
2. Розробка інтерфейсу програмного забезпечення, який буде зручним та дружнім до користувача.

3. Застосування елементів гейміфікації: анімації, гейміфіковані завдання, час та бали.
4. Забезпечення оптимальної продуктивності для різних пристроїв на платформі Unity.
5. Створення механізму для генерування різноманітних математичних завдань та тестів з урахуванням навчальних цілей.
6. Реалізація системи оцінювання результатів учнів з метою відстеження їхнього прогресу та успішності.
7. Підготовка документації, яка пояснює функціонал та правила використання програмного забезпечення для вчителів та учнів.

Очікувані результати

- Функціона гейміфікаовний додаток зі всіма вище зазначеними цілями.
- Високоякісний інтерфейс з функціональність на на високому рівні.
- Забезпечити максимальний рівень продуктивності.

Висновки до першого розділу

Вивчення математики розвиває навички аналізу ситуацій і важливе в багатьох сферах життя. Існує багато програм для навчання математики з різними перевагами та недоліками. Вибір залежить від віку учнів, цілей навчання та вподобань вчителя. Успішне впровадження програм потребує уважного підходу та врахування особливостей навчального середовища. Гейміфікація ефективна, бо створює захоплюючу атмосферу та стимулює активність учнів. Для середньої школи важливо створити інтерактивне середовище, розвивати просторове мислення, індивідуалізувати підхід до кожного учня і постійно оновлювати програмне забезпечення. Очікувані

результати включають функціональний гейміфікований додаток з високоякісним інтерфейсом та високим рівнем продуктивності.

РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ГЕЙМІФІКОВАНОЇ СИСТЕМИ

2.1 Аналіз платформ

Аналіз платформ для створення гейміфікованих систем полягає в оцінці різних програмних засобів, які надають інструменти для розробки інтерактивних навчальних додатків з елементами гри. Цей аналіз включає огляд функціональних можливостей, легкість використання, наявність шаблонів і компонентів гейміфікації, підтримку різних платформ та мов програмування, а також вартість і ліцензійні умови. Після проведення аналізу можна визначити найбільш підходящу платформу для конкретного проекту гейміфікованої системи навчання. Нижче наведено декілька мов програмування для використання:

1. **C# [11,12]**

   Переваги:

   1. **Об'єктно-орієнтоване програмування (ООП)**: Підтримка ООП дозволяє створювати класи, об'єкти, успадкування та поліморфізм, що полегшує структурування та повторне використання коду.
   2. **Платформа .NET**: Включення до складу технологічного стеку .NET забезпечує доступ до великої кількості бібліотек і фреймворків.
   3. **Unity**: Ця мова є основною для роботи з одним із найпопулярніших ігрових рушіїв – Unity. Unity забезпечує зручне середовище для створення графічних ігор та візуалізаційних додатків з елементами гейміфікації.

4. **Глибока інтеграція**: Глибока інтеграція C# з Unity сприяє ефективній розробці та полегшує створення високоякісного та легко розширюваного коду.
5. **Велика спільнота**: Широка спільнота розробників, велика кількість готових бібліотек та ресурсів спрощують процес розробки ігор.

Недоліки:

1. **Продуктивність**: C# може поступатися за продуктивністю мовам нижчого рівня, таким як C++.
2. **Залежність від платформи**: Незважаючи на широку підтримку, C# все ще залежить від середовища .NET, що може обмежувати його використання поза цим екосистемою.

2. **JavaScript [13]**

Переваги:

1. **Веб-орієнтованість**: JavaScript є основною мовою для розробки ігор для браузера. Це дозволяє створювати ігри, що працюють безпосередньо в веб-браузерах, без потреби у додаткових плагінах.
2. **Велика кількість бібліотек**: Існує багато бібліотек і фреймворків, таких як Phaser, Babylon.js та Three.js, що спрощують розробку 2D та 3D ігор.
3. **Швидка розробка**: Легкий синтаксис та інтерактивна природа JavaScript дозволяють швидко створювати та тестувати прототипи.
4. **Кросплатформеність**: Ігри, створені на JavaScript, можуть працювати на будь-якій платформі, що підтримує веб-браузер, включаючи мобільні пристрої.

Недоліки:

1. **Продуктивність**: Хоча JavaScript став значно продуктивнішим завдяки сучасним рушіям, він все ще поступається продуктивністю мовам нижчого рівня, таким як C++.
2. **Обмеженість ресурсів**: Обмежені можливості доступу до системних ресурсів та графічних API порівняно з мовами, такими як C++ або C#.
3. **Безпека**: Веб-ігри можуть бути вразливими до різних типів атак, таких як XSS (Cross-Site Scripting).

3. **Python [14]**

Переваги:

1. **Легкість використання**: Простий і зрозумілий синтаксис робить Python легким для навчання і використання, особливо для прототипування і швидкої розробки.
2. **Різні парадигми**: Підтримка різних парадигм програмування (об'єктно-орієнтованого, процедурного, функціонального) надає гнучкість у розробці.
3. **Широка підтримка бібліотек**: Велика кількість бібліотек та фреймворків для роботи з графікою, звуком та іншими аспектами ігрової розробки.

Недоліки:

1. **Продуктивність**: Низька продуктивність у порівнянні з C++ або C#, що може бути критичним для високоефективних ігор.
2. **Відсутність потужних ігрових рушіїв**: Хоча існують ігрові рушії, які підтримують Python, вони не настільки потужні, як ті, що підтримують C++ або C#.

Після детального аналізу розглянутих мов програмування стає очевидним, що C# є оптимальним вибором для розробки гейміфікованої

системи через його високу сумісність із платформою Unity, глибоку інтеграцію, сприяючу ефективній розробці, просту синтаксичну структуру та підтримку об'єктно-орієнтованого підходу, що полегшує створення високоякісного та легко розширюваного коду. Використання C# у проектах на платформі Unity також забезпечує доступ до великої кількості готових бібліотек та ресурсів, спрощуючи взаємодію з графікою, анімацією та іншими компонентами, необхідними для гейміфікованого навчання, і легко інтегрує елементи гейміфікації, сприяючи залученню та зацікавленню користувачів.

2.2 Аналіз ігрових рушіїв

Ігровий рушій є важливим інструментом для розробки відеоігор, що надає розробникам потужність і гнучкість для творення різноманітних ігрових виробів. Він включає в себе графічний двигун для створення та рендерингу графіки, фізичний двигун для моделювання реалістичної фізики, а також набір інструментів для роботи зі звуком, анімацією, штучним інтелектом, управлінням ресурсами та іншими аспектами гри. Ігрові рушії дозволяють розробникам швидко створювати якісні ігри для різних платформ, забезпечуючи при цьому зручний та ефективний робочий процес. Найпопулярніші ігрові рушії включають Unity, Unreal Engine, CryEngine та Godot Engine.

Основні характеристики ігрових рушіїв включають:

1. Графічний двигун: Ігровий рушій має потужний графічний двигун, який відповідає за відображення графіки, освітлення, тіні, та ефекти.
2. Фізичний двигун: Він забезпечує реалістичну фізику в грі, таку як рух об'єктів, колізії та взаємодії.

3. Інтегровані розробницькі інструменти: Ігрові рушії надають інтегровані середовища для розробки, включаючи редактори сцен, анімацій та програмування.
4. Підтримка платформ: Вони зазвичай підтримують різні платформи, такі як PC, консолі, мобільні пристрої та віртуальна реальність.
5. Комунітета підтримка: Багато ігрових рушіїв мають активні спільноти розробників, які надають поради, рішення проблем та ресурси для підтримки розробки.
6. Мультимедійні можливості: Ігрові рушії часто підтримують роботу з аудіо, відео та іншими мультимедійними ресурсами для створення іммерсивного геймплею.

Проаналізовано декілька популярних ігрових рушіїв:

Unity [17]:

Переваги:

- **Кросплатформенність:** Unity дозволяє розробляти ігри для різних платформ, таких як Windows, macOS, Android, iOS, Xbox, PlayStation та інші. Це робить його відмінним вибором для розробки ігор на різних пристроях.
- **Легкість вивчення:** Unity має дружній інтерфейс та широкий набір документації та навчальних матеріалів, що робить його доступним для новачків у галузі розробки ігор.
- **Велика спільнота розробників:** Unity має велику та активну спільноту розробників, що означає, що завжди є можливість знайти відповіді на питання, отримати допомогу або навіть знайти готові рішення для певних завдань.

- **Багатофункціональність:** Unity має великий екосистему, що включає в себе готові ресурси, активні ринки активів, інтеграцію сторонніх сервісів та інше, що полегшує процес розробки.
- **Можливості гейміфікації:** Unity пропонує різноманітні інструменти для реалізації елементів гейміфікації у ваших проектах, що дозволяє створювати захоплюючі та мотиваційні ігрові досвіди.

Недоліки:

- **Менш потужний у графіці порівняно з Unreal Engine:** У порівнянні з Unreal Engine, Unity може мати меншу потужність у сфері графіки, хоча в останні роки ця різниця зменшилася.
- **Ростуть витрати на користувацьку підтримку:** З ростом проектів збільшуються витрати на користувацьку підтримку та додаткові функції, що може стати проблемою для невеликих студій або незалежних розробників.
- **Невисока продуктивність на деяких платформах:** Хоча Unity дозволяє розробку для різних платформ, на деяких з них може бути менша продуктивність порівняно з іншими ігровими рушіями.

Unreal Engine [18]:

Переваги:

- **Потужний двигун для графіки:** Unreal Engine відомий своєю вражаючою графікою та реалістичними візуальними ефектами, що робить його відмінним вибором для розробки громадських ігор та інтерактивних досвідів.

- **Велика стійкість та оптимізація:** Unreal Engine володіє високою стійкістю та оптимізацією, що дозволяє створювати великі та складні ігри без значних проблем з продуктивністю.
- **Широкі можливості функціональності:** Unreal Engine має вбудовані інструменти для різних аспектів розробки, включаючи графіку, анімацію, фізику, штучний інтелект та багато іншого, що полегшує роботу розробників.
- **Підтримка віртуальної реальності та доповненої реальності:** Unreal Engine має вбудовану підтримку для розробки віртуальної реальності та доповненої реальності, що робить його ідеальним вибором для проектів у цих областях.
- **Доступність джерел відкритого коду:** Unreal Engine став доступним для користувачів за умовами, що дозволяють доступ до вихідного коду. Це сприяє співпраці розробників та спільному вдосконаленню рушія.

Недоліки:

- **Висока вимогливість до ресурсів:** Через свою високу якість графіки та широкий функціонал, Unreal Engine може бути вимогливим до ресурсів комп'ютера, що може вплинути на продуктивність під час розробки.
- **Складність вивчення:** У порівнянні з іншими ігровими рушіями, Unreal Engine може мати складнішу криву вивчення через свою потужну функціональність та розширені можливості.
- **Ліцензійні витрати:** При деяких умовах використання, використання Unreal Engine може призвести до великих витрат на ліцензії та відсотки від доходу від продажу гри.

CryEngine [19]:

Переваги:

- **Графічна якість:** CryEngine славиться своєю вражаючою графікою та реалістичними візуальними ефектами, що робить його відмінним вибором для розробки високоякісних ігор та інтерактивних досвідів.
- **Інструменти розробки:** CryEngine має потужні та розширені інструменти розробки, що дозволяють створювати складні та деталізовані ігрові світи з різноманітними можливостями.
- **Фізична модель:** Цей рушій володіє потужною фізичною моделлю, що дозволяє реалістично моделювати рух об'єктів, динаміку середовища та інші аспекти фізики.
- **Підтримка великої кількості платформ:** CryEngine може розробляти ігри для різних платформ, включаючи ПК, консолі та мобільні пристрої, що розширює потенційну аудиторію гри.

Недоліки:

- **Висока вимогливість до ресурсів:** Як і у випадку з іншими рушіями високої якості, CryEngine може бути вимогливим до ресурсів комп'ютера, що може вплинути на продуктивність розробки.
- **Складність вивчення:** У порівнянні з іншими ігровими рушіями, CryEngine може мати складнішу криву вивчення через свою потужну функціональність та розширені можливості.
- **Обмежена спільнота та документація:** У порівнянні з більш популярними рушіями, такими як Unity або Unreal Engine, у

CryEngine менша спільнота користувачів та обмеженіша кількість документації та ресурсів для вивчення.

Після аналізу чотирьох ігрових рушіїв - Unity, Unreal Engine, CryEngine – Unity виділяється як найкращий вибір для розробки гейміфікованих додатків та ігор. Завдяки своїй мультиплатформеності, великій спільноті користувачів, гнучкості, широкому функціоналу та широкому використанню в індустрії, Unity надає розробникам необхідні інструменти для створення високоякісних та захоплюючих ігор та навчальних додатків **[21]**.

Unity також відомий своєю відмінною підтримкою для розробки малоємних додатків та ігор порівняно з Unreal Engine. Вона забезпечує швидкий процес розробки, легку інтеграцію, та оптимізацію для різних платформ, що робить її відмінним вибором для невеликих команд розробників та початківців.

2.3 Проектування гейміфікованої системи

Процес проектування гейміфікованої системи складається з кількох послідовних етапів, кожен з яких включає конкретні кроки та взаємодії між компонентами системи, що зображено на UML-схемі (рис. 2.3.1).

1. **Взаємодія з головним меню**

   Користувачу дається можливіть ознайомитися з головним інтерфейсом.
2. **Вибір класу**

   Дальше користувач має змогу вибрати рівень шкільного курсу з 5 по 7 клас.
3. **Вибір предмету(Алгебра чи Геометрія)**

   Пропонується вибрати предмет для проходження.
4. **Вибір складності рівня**

   3 рівня складності з алгебри і геометрії

5. **Міні ігри**

Hexagons і ClassWork пропонують завдання з варіантами відповіді

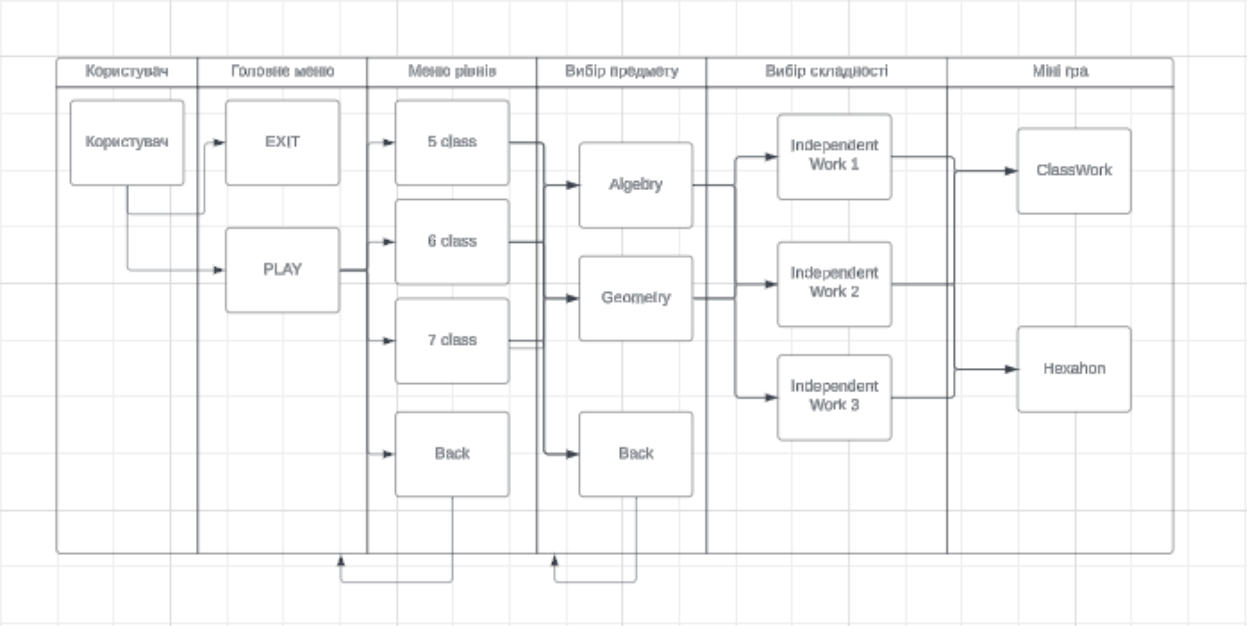

Рисунок 2.3.1 – UML-схема.

Висновки до другого розділу

Вибір ігрового рушія та мови програмування для розробки гейміфікованих систем є ключовим етапом у процесі створення інтерактивних навчальних додатків та ігор. Unity, як ігровий рушій, і мова програмування C# в цьому контексті виявляються особливо ефективними засобами завдяки своїм перевагам.

Unity відкриває широкі можливості для розробки гейміфікованих додатків та ігор завдяки своїй мультиплатформеності, гнучкості та великій спільноті користувачів. Використання мови програмування C# в Unity дозволяє створювати високоякісний та легко розширюваний код, а також забезпечує зручну інтеграцію з різними компонентами гри та гейміфікації.

Правильний вибір ігрового рушія та мови програмування є важливим, оскільки від цього залежить якість та ефективність розробки, а також можливості для подальшого розширення та підтримки проекту. Необхідно враховувати потреби та специфіку конкретного проекту, а також здатність обраного інструменту відповідати цим вимогам.

Таким чином, вибір Unity та мови програмування C# для розробки гейміфікованих систем навчання може забезпечити ефективний та успішний процес розробки, а також створити продукт високої якості, який відповідає потребам користувачів.

РОЗДІЛ 3. РОЗРОБКА ГЕЙМІФІКОВАНОЇ СИСТЕМИ

3.1 Реалізація компонентів застосунку

3.1.1 Реалізація основного меню

Початкова сцена або головне меню (рис. 3.1.1) складається з кількох компонентів:

- Карта
- UI Канвас
- Камера
- Загрузочний екран (для переходу на сцени з міні-іграми)

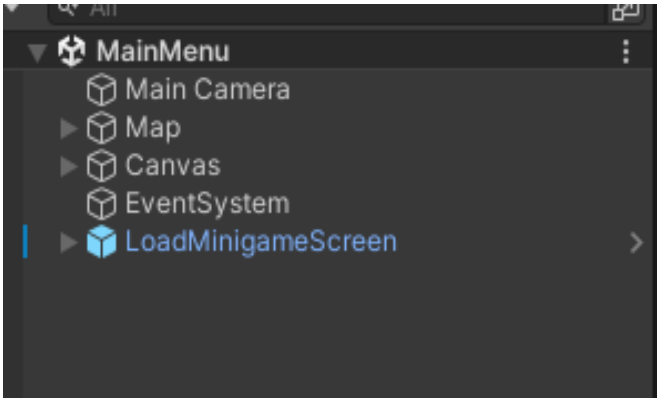

Рисунок 3.1.1 – Компонентів в інспекторі

На сцені (рис. 3.1.2) гравець взаємодіє лише з UI для вибору завдань та переходу до міні-ігор.

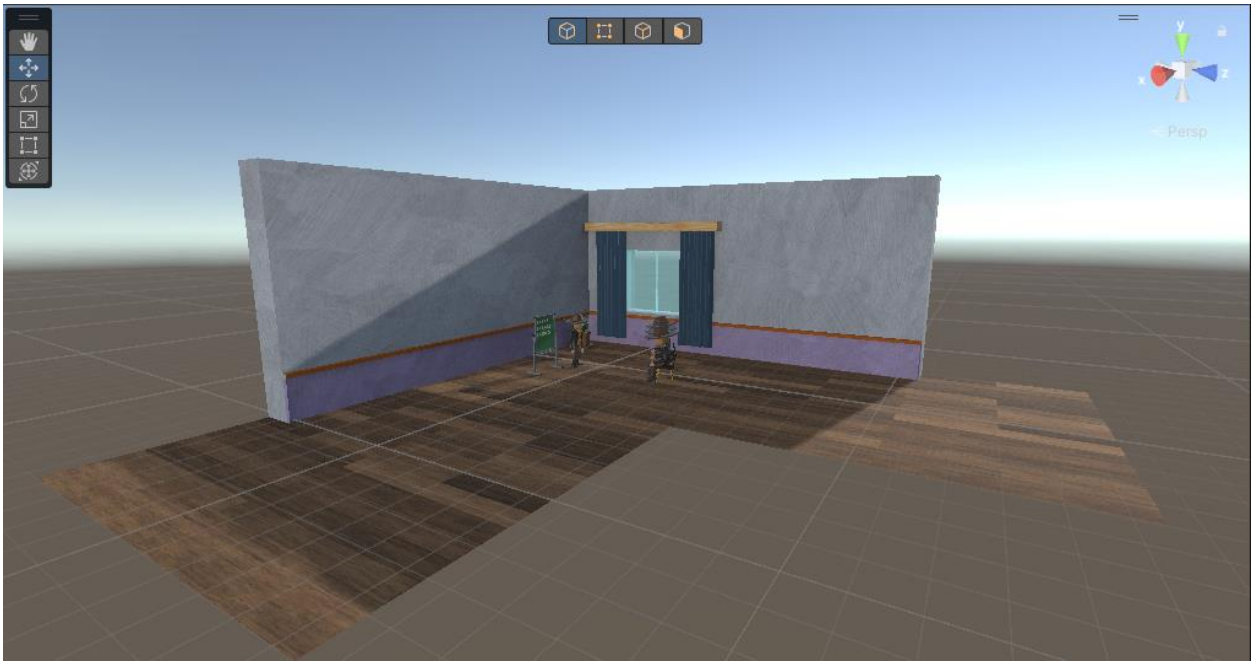

Рисунок 3.1.2 – Сцена

UI(рис. 3.1.3) має досить просту і зрозумілу структуру:

- Головне меню
- Меню вибору класу
- Меню вибору завдання

Рисунок 3.1.3 – Структура UI

Всі переходи між меню відбуваються за допомогою скрипта ChangeScreen.cs який лежить на об’єктах меню, таких як: MainMenu, LevelsMenu, Class5, Class6, Class7. Цей скрипт допомагає робити анімований

перехід між меню, а саме – відтворює анімацію закриття меню (після цієї анімації відбувається вимкнення об’єкту активного меню і увімкнення цільового об’єкту).

MainMenu та LevelsMenu(рис. 3.1.4) є лише провідниками до меню з завданнями (Class5, Class6, Class7). Перейшовши до меню з завданнями гравець може обрати якого типу йому будуть попадатись завдання, а саме між Алгеброю та Геометрією.

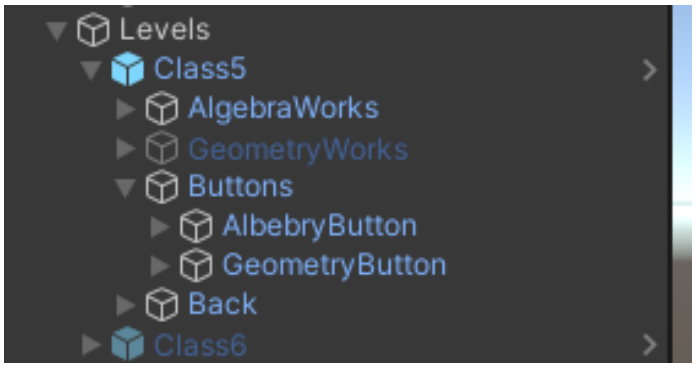

Рисунок 3.1.4 - Меню класу

Об’єкти AlgebraWorks та GeometryWorks є під-меню класу, саме в них знаходяться завдання які може обрати гравець. Ці під-меню також перемикаються за допомогою скрипту ChangeScreen.cs.

Для перемикання між цими меню використовуються кнопки в верхньому лівому куті екрану.

Рисунок 3.1.5 – Меню класу

В меню класу при виборі однієї з тем (Алгебра, Геометрія) відбуваються наступні дії:

- Вимикається Interactable у цієї кнопки який дозволяє натиснути кнопку повторно.
- Змінюється меню з GeometryWorks на AlgebraWorks (рис. 3.1.5)
- Вмикається Interactable у кнопки GeometryButton для того, щоб переключитись назад на меню з Геометрією)

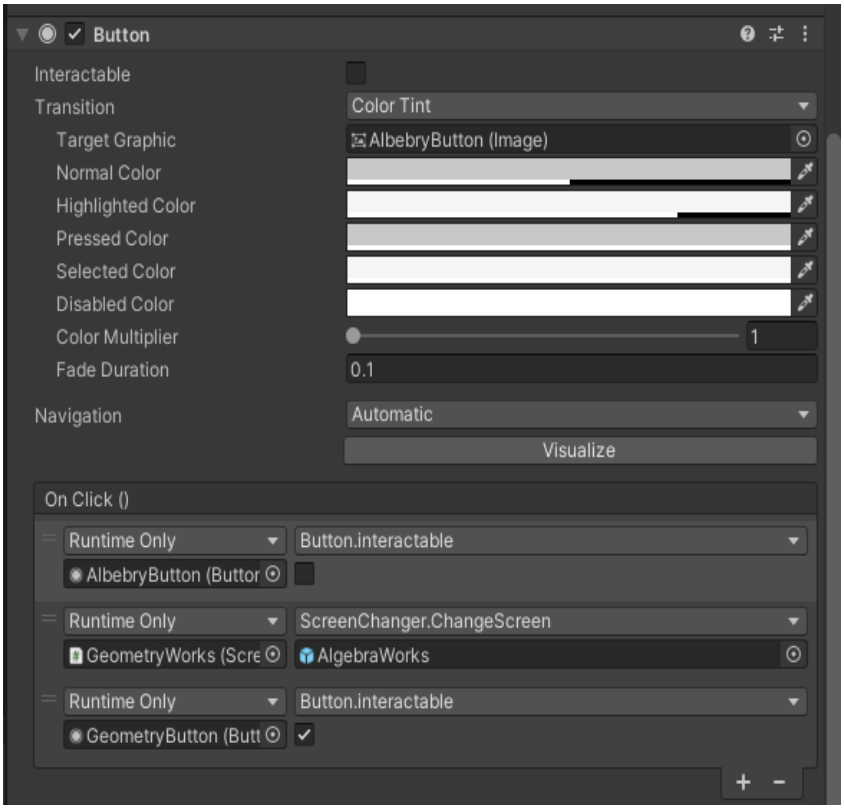

Рисунок 3.1.5 – Компонент Button для кнопки AlgebryWorks

AlgebraWorks та GeometryWorks (рис. 3.1.6) містять в собі об’єкти завдань які встановлюють потрібні запитання та включають одну з можливих міні-ігор.

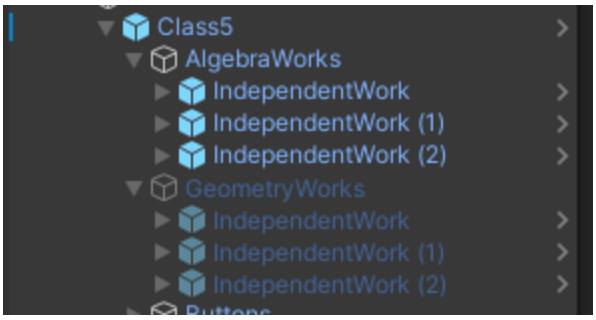

Рисунок 3.1.6 – Структура об’єктів AlgebraWorks та GeometryWorks

Встановлення завдань та запуск міні гри відбувається за допомогою скрипта WorkButton.cs (рис. 3.1.7).

Скрипт WorkButton.cs має кілька посилань:

- Загрузочний екран (Використовується для завантаження сцени)
- Можливі міні ігри (ScriptableObjects цих міні ігор)
- Пак з запитаннями які використовуються в міні грі

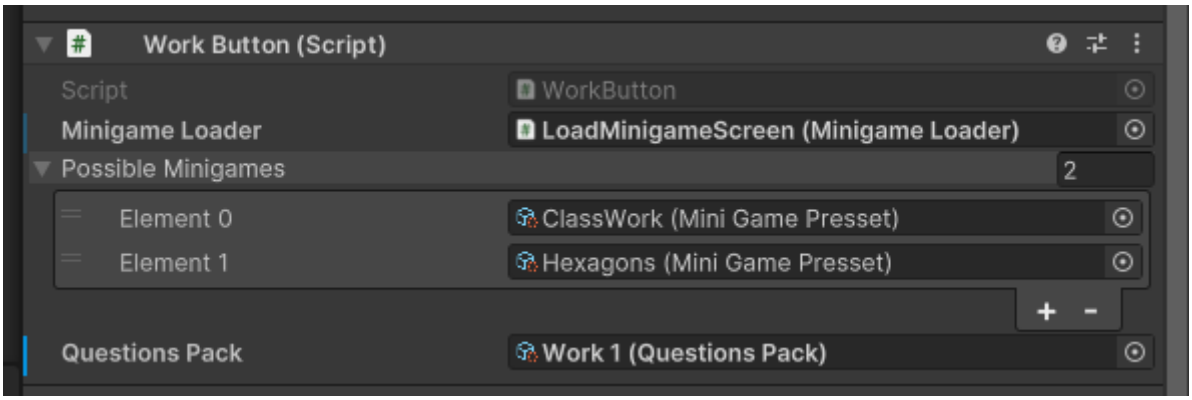

Рисунок 3.1.7 – Скрипт WorkButton.cs

Після натискання однієї з кнопок відбувається завантаження сцени з міні грою, тому розглянемо деякі елементи які тут використовуються, а саме ScriptableObject міні ігор та паку питань.

Об’єкти міні ігор в інспекторі(рис. 3.1.8) містять таку структуру:

- Картинка міні-гри (Sprite)
- Туторіал (string)
- Туторіал по керуванню (string)

- Індекс сцени (int)

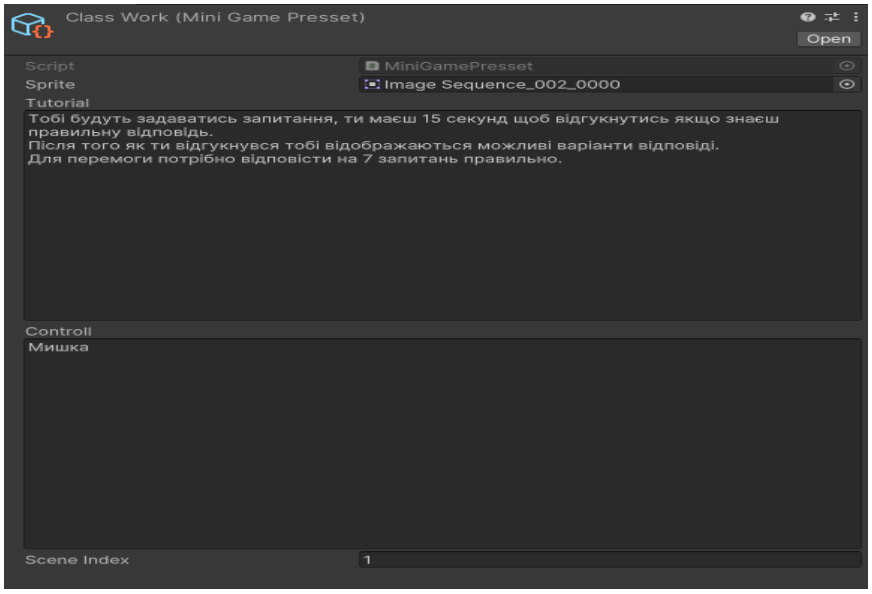

Рисунок 3.1.8 – Вигляд об’єкту міні-гри в інспекторі

Індекс сцени визначається в настройках білда(рис. 3.1.9) і є унікальним для кожної сцени.

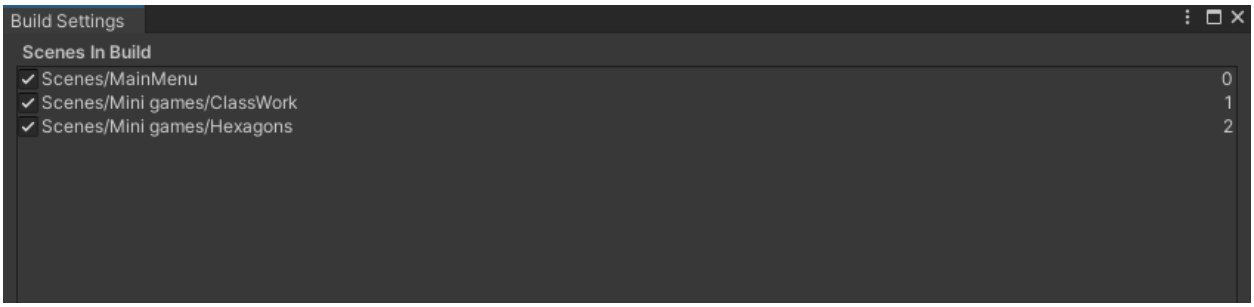

Рисунок 3.1.9 – Сцени, що використовуються в білді

Ці об’єкти використовуються для запуску міні ігор і відображенні інформації про них в загрузочному екрані.

Щодо об’єктів з питаннями, їх є декілька, а саме:

- Самі питання (Question.cs). Містять в собі питання, можливі відповіді на них (Перша відповідь завжди правильна. Це зроблено щоб уникати помилок коли правильна відповідь не збігається) і за потреби картинку до питання.

- Набір з питаннями (QuestionsPack.cs)(рис.3.1.10). Він містить в собі список питань (Questions.cs)(рис 3.1.11) і час відповіді на кожне (По дефолту – 15 секунд)

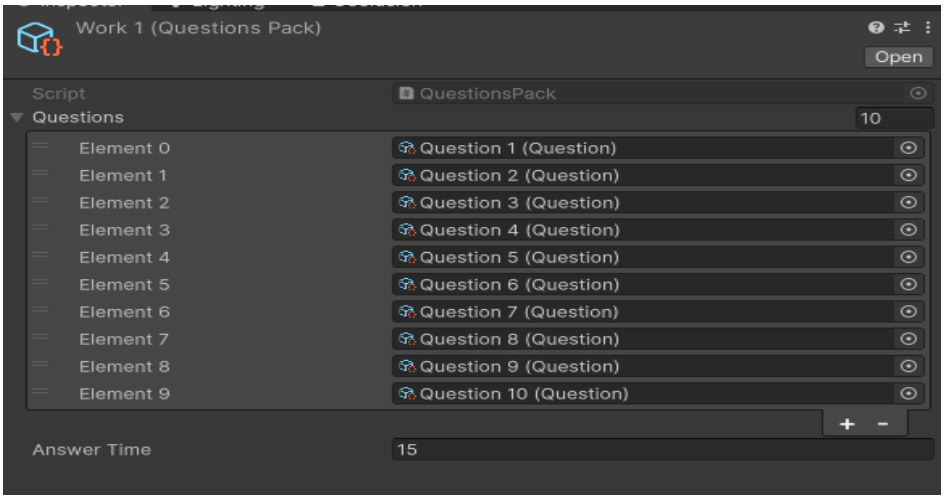

Рисунок 3.1.10 – Набір з питаннями

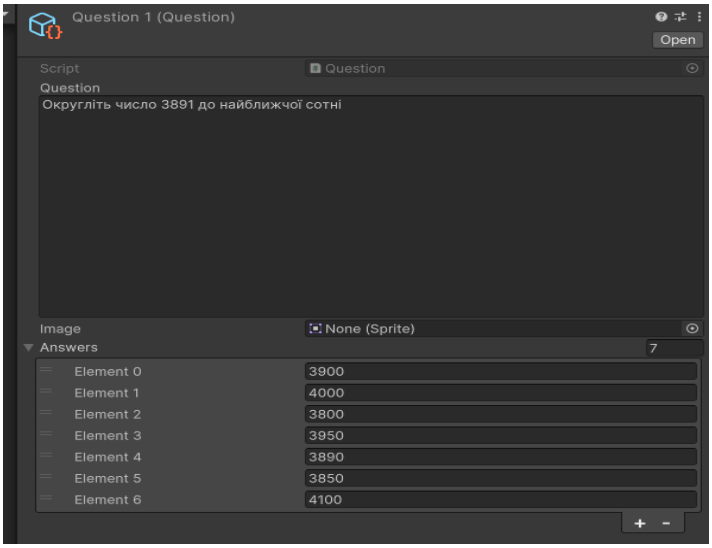

Рисунок 3.1.11 – Об’єкт з питанням

3.1.2 Екран завантаження

Екран завантаження (рис. 3.1.12) змінює активну сцену на сцену з обраною міні грою. Під час завантаження гравець може побачити картинку з рівня, назву рівня і туторіал в якому описано як грати.

Сцена не завантажується відразу, спочатку йде завантаження самої сцени (прогрес цього завантаження відображається справа знизу) і коли сцена готова - з’являється кнопка Start, натиснувши на яку гравець запускає сцену з грою.

Рисунок 3.1.12 - Вигляд екрану завантаження

Завантаження відбувається за допомогою двох скриптів, а саме LoadingMiniGameScreen.cs та MiniGameLoader.cs. Перший відповідає за сам процес завантаження та анімації перемикання сцен, а другий за інформацію з цільової міні гри.

3.1.3 Міні гра Hexagons

У грі Hexagons(рис. 3.2.1) потрібно обрати правильну відповідь стаючи на шестикутник з відповіддю. Після таймеру всі шестикутники з неправильними відповідями опустяться вниз. Якщо стояти на одному з таких шестикутників втрається 1 здоров'я, після 3 невдач - поразка.

Керування: W, A, S, D – рух.

Space - показати правильну відповідь.

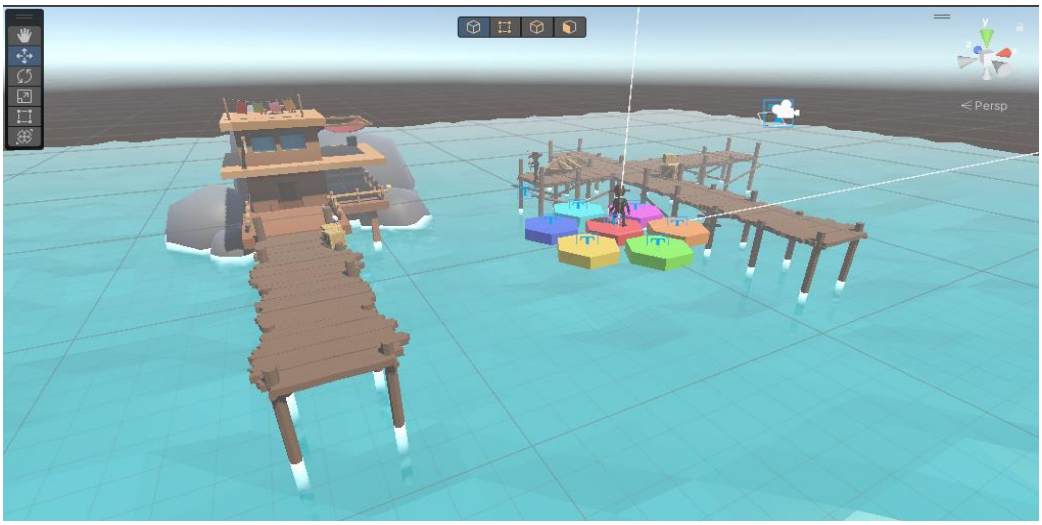

Рисунок 3.2.1 - Hexagons

В цій міні грі гравець керує об’єктом Player за допомогою скрипта PlayerControll.cs (рис. 3.2.2) (Код наведено в додатку А). Цей скрипт відповідає за переміщення гравця по карті.

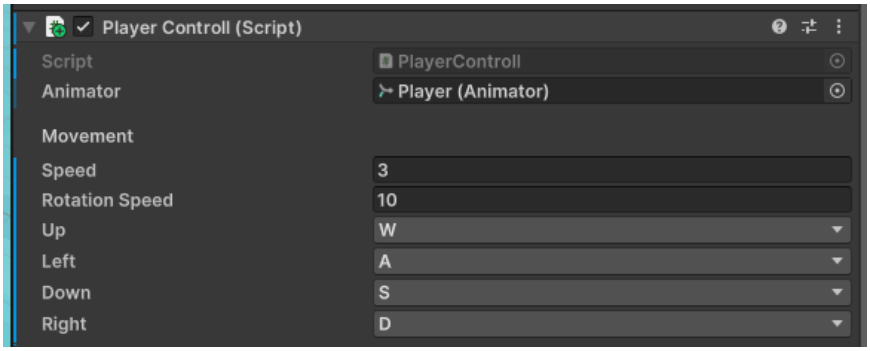

Рисунок 3.2.2 – Скрипт PlayerControll на об’єкті Player

Об’єкт, що відповідає за саму міні гру знаходиться в дочірніх об’єктах об’єкту Map (рис. 3.2.3), він містить в собі два скрипта, а саме:

- PlatformsGame.cs. Відповідає за керуванням платформами залежно від питання.
- PlayerRespawner.cs. Відроджує гравця, якщо попередній раунд завершено і об’єкт гравця неактивний.

Ці об’єкти виконують основну частину роботи гри.

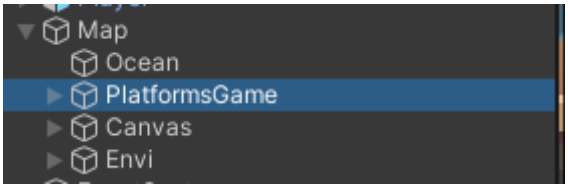

Рисунок 3.2.3 – Об’єкт Map

PlatformsGame.cs та PlayerRespawner.cs (рис. 3.2.4) на об’єкті PlatformsGame

Об’єкт, що знаходиться біля PlatformsGame, а саме Canvas – відповідає за UI гри і відображає актуальну інформацію щодо гри, а саме: кількість життів, таймер для відповіді, поточний раунд.

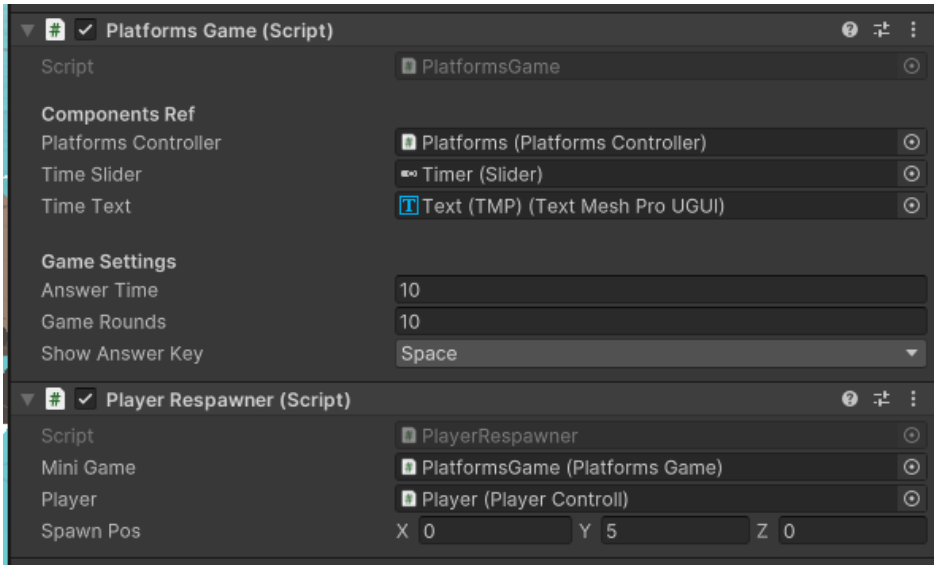

Рисунок 3.2.4 – Скрипти PlatformsGame.cs та PlayerRespawner.cs

Об’єкт WinLoseScene(рис. 3.2.5) відповідає за сцену після завершення гри і активує дочірні об’єкти залежно від результату гри при цьому переключаючи камеру за допомогою посилання на скрипт CameraController.cs.

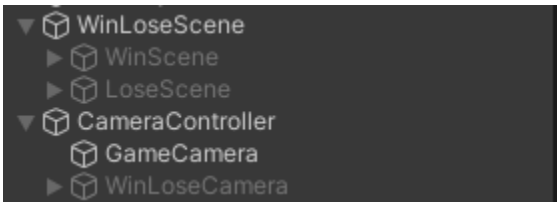

Рисунок 3.2.5 – Об’єкти WinLoseScene та CameraController.cs

На сцені знаходиться два завантажувальні екрани (рис. 3.2.6), один з них потрібен для переходу на міні гру (якщо гравець натисне кнопку Retry), а другий – для переходу в меню. Це створено через те, що вони мають різну анімацію та функціонал.

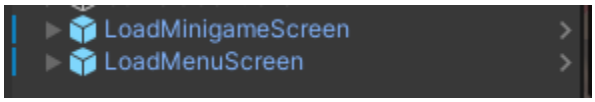

Рисунок 3.2.6 – Об’єкти загрузочних екранів

3.1.4 Міні гра Class Work

У грі Class Work потрібно дати відповідь на запитання, відповісти на нього потрібно протягом 15 секунд, щоб відгукнутись якщо знаєш правильну відповідь.

Після готовності відображаються можливі варіанти відповіді.

Для перемоги потрібно відповісти на 7 запитань правильно.

Керування мишкою.

В цій міні грі гравець взаємодіє з UI елементами, але на його дії реагують персонажі на сцені.

Питаннями керує скрипт ClassWork.cs(код наведено в додатку В), що знаходиться на об’єкті ClassWorkGame (рис 3.3.1) інші скрипти та об’єкти доповнюють його та дають змогу взаємодіяти з ним.

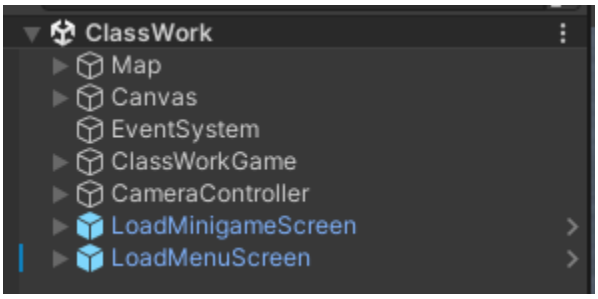

Рисунок 3.3.1 – Об’єкти сцени

Такими скриптами є:

- **AnswerButton.cs**. Знаходиться на об’єкті кнопки відповіді та використовується для взаємодії з класом Answerer.cs(опис далі).
- **ClassWorkEndGame.cs**. Перемикає камеру за допомогою CameraController.cs після завершення гри.
- **QuestionSetter.cs**. Використовується для встановлення відповідей в об’єкти з класами AnswerButton.cs, відображення поточного питання і рандомізації питань.

- **ButtonsController.cs**. Керує кнопками з відповідями, кнопкою відповіді, а також таймером на відповідь.
- **SchoolBoard.cs**. Відображає спрайти запитань на дошці та керує її анімаціями. Також використовується для відображення кінцевого результату.

Також на сцені присутні об’єкти з класами Answerer.cs, цей клас використовується для керування анімаціями персонажа, що відповідає на запитання.

3.2 Демонстрація роботи застосунку

Користувачу пропонує початкове меню(рис. 3.2.1), де пропонується вибрати два варіанти

- PLAY
- EXIT

Рисунок 3.2.1 – Початкове меню

В головному меню(рис 3.2.2), якщо обрати Exit, то це значить що користувач вийде із застосунку. Якщо вибрати PLAY, то користувачеві дасть на вибір рівні з 5-7 класів, або BACK, щоб повернутися назад.

Рисунок 3.2.2 – головне меню

Після того як користуватися вибрав клас, дається можливість вибрати предмет навчання(рис 3.2.3), це може бути або алгебра або геометрія (рис 3.2.3). Або користувач може повернутися назад.

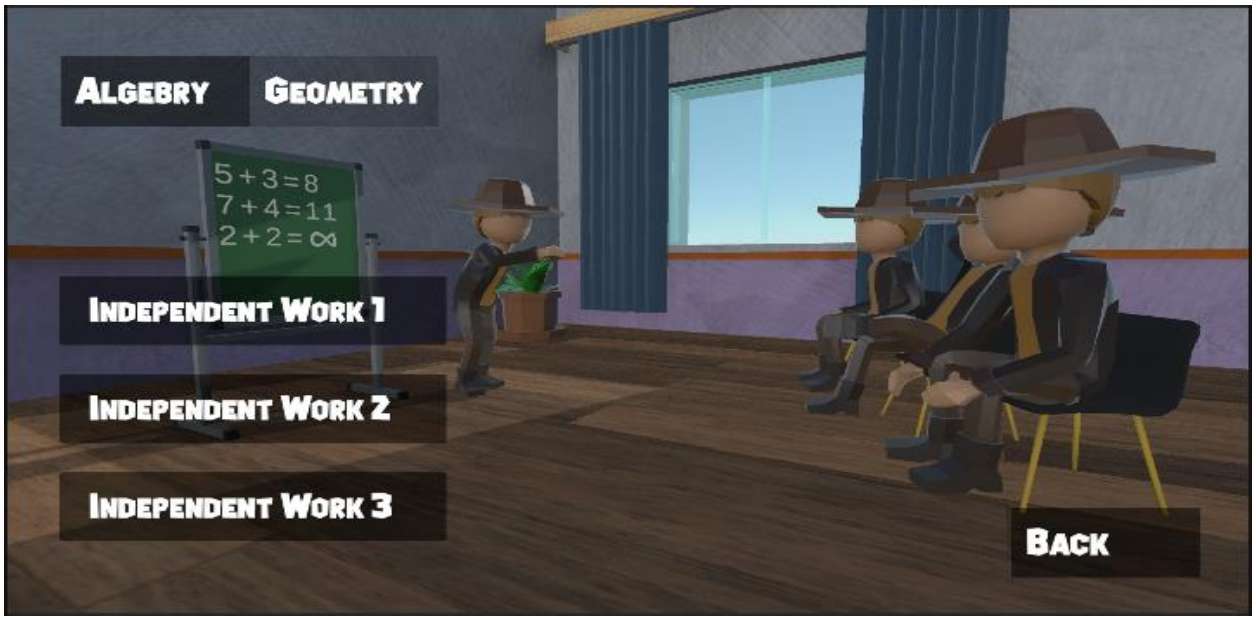

Рисунок 3.2.3 – Предмет навчання

Є три рівня роботи на кожен клас, окремо із алгебри і геометрії. Після того як ми оберемо рівень, запускається одна з міні ігор Hexagons(рис. 3.2.4) або ClassWork(рис 3.2.5).

Рисунок3.2.4 – Hexagons

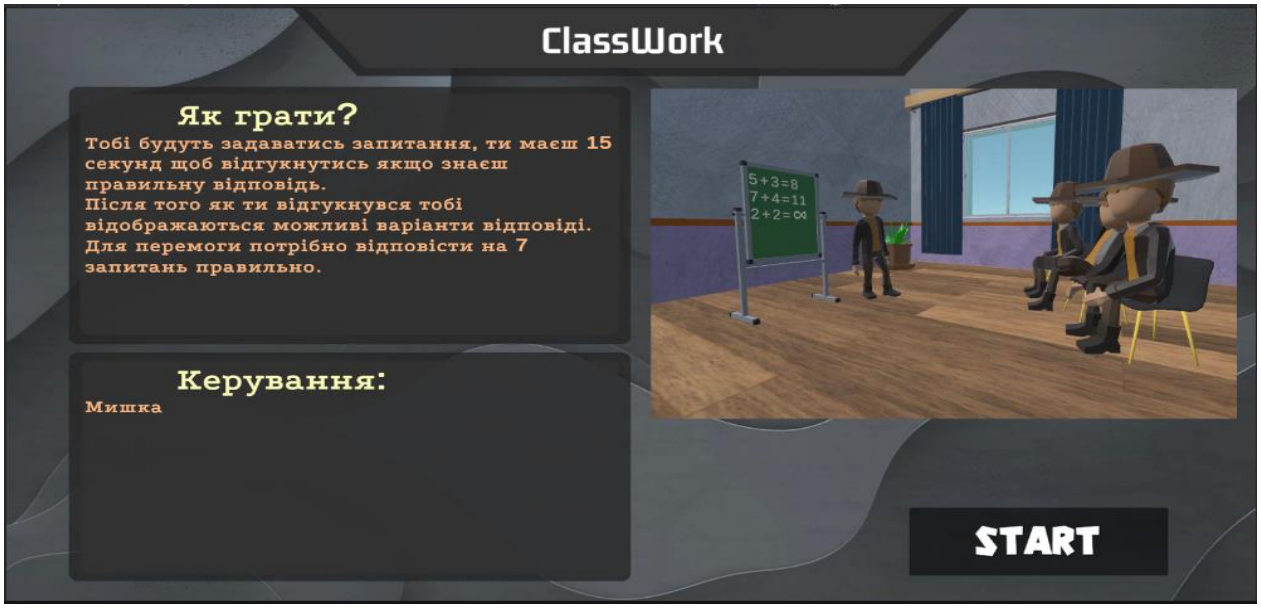

Рисунок 3.2.5 – ClassWork

Висновки до третього розділу

У ході розробки програмного забезпечення було створено інтуїтивно зрозуміле головне меню, яке забезпечує зручну навігацію та швидкий доступ до різних частин навчального додатку. Впроваджено анімовані переходи між меню, що покращує користувацький досвід. Реалізовані механізми вибору завдань і міні-ігор, що дозволяють легко додавати нові елементи та розширювати функціональність додатку. Завдяки скриптам для завантаження сцен, мінімізовано час очікування користувачів, забезпечуючи безперервність геймплею. На прикладі міні-гри Hexagons продемонстровано інтерактивний геймплей з використанням скриптів для керування гравцем і платформами, а також інтеграцію UI для відображення інформації про гру.

Висновки

На основі проведеного дослідження висновується, що гейміфікація є перспективним напрямом у покращенні навчального процесу з математики в середній школі. Її застосування сприяє створенню захоплюючого навчального середовища, що стимулює активну участь учнів, підвищує їхню мотивацію та зацікавленість у предметі. Аналіз існуючих програмних засобів для навчання математики показує, що вибір конкретного програмного забезпечення повинен здійснюватися з урахуванням потреб та характеристик учнів та вчителів. Використання Unity та мови програмування C# виявляється успішним і ефективним для розробки гейміфікованих систем навчання математики, забезпечуючи якісний інтерфейс та зручність у використанні. Такий підхід дозволяє покращити якість навчання математики, розвиваючи

навички самостійного навчання та критичного мислення учнів, та створити продукт, який відповідає сучасним вимогам освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1]“FutureNow” [Наукова стаття] - https://futurenow.com.ua/matematyka-v-zhytti-lyudyny/ [Дата звернення : 02.03.2024]

[2]“Всеосвіта” [електронний ресурс] - https://vseosvita.ua/library/metodika-navcanna-matematiki-ak-nauka-i-navcalna-disciplina-345370.html . [Дата звернення: 04.03.2024]

[3]“SuperFrof” [електронний ресурс блог] - https://www.superprof.com.ua/blog/rol-matematyky-v-zhytti-lyudyny [Дата звернення: 06.03.2024]

[4]“Освіта.ua” [електорнний ресурс] - https://osvita.ua/vnz/reports/pedagog/14001. [Дата звернення: 07.03.2024]

[5] “НУШ” [електронний ресурс] - https://nus.org.ua/view/yak-zrobyty-navchannya-matematyky-tsikavym-i-produktyvnym. [Дата звернення: 08.03.2024]

[6] “Prodigy Math Game” [електронний ресурс] – https://www.prodigygame.com/main-en/ [Дата звернення: 12.03.2024]

[7] “Khan Academy” [електронний ресурс] – https://uk.khanacademy.org/ [Дата звернення: 15.03.2024]

[8] “DragonBox Algebra” [електронний ресурс] – https://dragonbox.com/products/algebra-12 [Дата звернення: 17.03.2023]

[9] “Mathletics” [електронний кабінет] – https://www.mathletics.com/uk/ [Дата звернення: 20.03.2024]

[10] “ST Math” [електронний кабінет] – https://www.stmath.com/ [Дата звернення: 22.03.2024]

[11] “C# Documentation” [електронний ресурс] -

https://learn.microsoft.com/en-us/dotnet/csharp . [Дата звернення: 24.03.2024]
[12] “C# Game Development” [електронний ресурс] - https://www.javatpoint.com/c-sharp-game-development [Дата звернення: 27.03.2024]
[13] “Learn Game Development with JavaScript” [електронний ресурс] - https://www.freecodecamp.org/news/learn-javascript-game-development-full-course/. [Дата звернення: 29.03.2024]
[14]“Real Python” [електронний ресурс] - https://realpython.com/tutorials/gamedev/. [Дата звернення: 04.04.2024]
[15] “Java” [електронний ресурс] - https://dev.to/dbillion/java-for-game-development-an-introduction-10c6 [Дата звернення: 09.04.2024]
[16] “HTML and CSS” [електронний ресурс] https://developer.mozilla.org/en-US/docs/Games/Introduction_to_HTML5_Game_Development
[Дата звернення: 14.04.2024]
[17] Goldstone W. Unity Game Development Essentials. / W Goldstone. Birmingham: Packt Publishing Ltd. – 2015.
[18] “Unreal Engine documentation” [електронний ресурс] - https://docs.unrealengine.com/5.3/en-US/understanding-the-basics-of unreal-engine . [Дата звернення: 22.04.2024]
[19] “CryEngine” [електронний ресурс] – https://www.cryengine.com/tutorials
[Дата звернення: 24.04.2024]
[20] “Godot Engine” [електронний ресурс] - https://docs.godotengine.org/en/stable . [Дата звернення: 26.03.2024]
[21] Крейтон, P.X. Основи розробки ігор у Unity / P.X. Крейтон. – Packt Publishing, 2017.
[22] 15. Lehinovych A., Soroka R., Nevmerzhytskyi V., Semianyk M.-I., Izmailov A. Gamified Web Assistant for Support of Learning Process in Secondary School –

Zgamifikowany Asystent Webowy do Wspierania Procesu Uczenia w Szkole Średniej. Zeszyty Studenckiego Towarzystwa Naukowego AGH. Wydawnictwo Studenckiego Towarzystwa Naukowego AGH – ISSN 1732-0925; 2024. In print.

ДОДАТОК А

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

public class PlayerControll : MonoBehaviour
{
    [SerializeField] private Animator animator;

    [Header("Movement")]
    [Space(5f)]
    [SerializeField] private float speed = 1f;
    [SerializeField] private float rotationSpeed = 1f;
    [SerializeField] private KeyCode up = KeyCode.W;
    [SerializeField] private KeyCode left = KeyCode.A;
    [SerializeField] private KeyCode down = KeyCode.S;
    [SerializeField] private KeyCode right = KeyCode.D;

    private bool isDead = false;
    private Vector3 moveVector = Vector3.zero;
    private Quaternion rotateVector;
    private Rigidbody rigidbody;
```

```
public int lives { get; private set; } = 3;
public Action<int> OnLivesUpdate;

private void Awake()
{
   rigidbody = GetComponent<Rigidbody>();
}

private void OnTriggerEnter(Collider other)
{
   if(other.tag == "Water")
   {
      lives--;
      OnLivesUpdate?.Invoke(lives);
      gameObject.SetActive(false);
   }
}

public void RespawnPlayer(Vector3 spawnPos)
{
   if(lives > 0)
   {
      transform.position = spawnPos;
      gameObject.SetActive(true);
   }
   else
   {
```

```
            GameSettings.OnGameLose?.Invoke();
        }
    }

    private void FixedUpdate()
    {
        if (InactiveCheck())
            return;

        Move();
    }

    private void Update()
    {
        if (InactiveCheck())
            return;
    }

    private void Move()
    {
        moveVector = Vector3.zero;
        if (Input.GetKey(up))
        {
            moveVector += Vector3.forward;
            rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, 0, 0), rotationSpeed * Time.deltaTime);
        }
```

```
        if (Input.GetKey(down))
        {
            moveVector += -Vector3.forward;
            rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, 180, 0), rotationSpeed * Time.deltaTime);
        }

        if (Input.GetKey(right))
        {
            moveVector += Vector3.right;
            rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, 90, 0), rotationSpeed * Time.deltaTime);
            if (Input.GetKey(up))
                rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, 45, 0), rotationSpeed * Time.deltaTime);
            else if (Input.GetKey(down))
                rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, 135, 0), rotationSpeed * Time.deltaTime);
        }

        if (Input.GetKey(left))
        {
            moveVector += -Vector3.right;
            rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, -90, 0), rotationSpeed * Time.deltaTime);
            if (Input.GetKey(up))
```

```
            rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, -45, 0), rotationSpeed * Time.deltaTime);
        else if (Input.GetKey(down))
            rotateVector = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, -135, 0), rotationSpeed * Time.deltaTime);
    }

    moveVector = Vector3.ClampMagnitude(moveVector, 1f);

    rigidbody.MovePosition(transform.position + moveVector * speed *
Time.fixedDeltaTime);
    transform.rotation = rotateVector;
    animator.SetFloat("Speed", moveVector.magnitude * speed);
  }

  private bool InactiveCheck()
  {
    if (isDead)
    {
      return true;
    }

    return false;
  }
}
```

ДОДАТОК В

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using Random = UnityEngine.Random;

public class ClassWork : MiniGame
{
    public const float passingScore = 0.7f;
    public QuestionSetter questionSetter;
    public QuestionsPack questionsPack { get; private set; }
    private List<Question> randomizedQuestions = new List<Question>();
    public Action OnAnswerGeted;
    public Action<Sprite> OnQuestionImage;

    private void Awake()
    {
        questionsPack = GameSettings.questionsPack;
        RandomizeQuestions();
    }
```

```
    private void Start()
    {
        StartCoroutine(StartGame());
    }

    public void NextQuestion()
    {
        OnRoundChanged?.Invoke(currentRound+1);
        if (currentRound >= randomizedQuestions.Count)
        {
            StartCoroutine(EndGame());
            return;
        }
        questionSetter.SetQuestion(randomizedQuestions[currentRound]);

OnQuestionImage?.Invoke(randomizedQuestions[currentRound].image);
    }

    private IEnumerator StartGame()
    {
        yield return new WaitForSeconds(2f);
        NextQuestion();
    }

    private IEnumerator EndGame()
    {
        questionSetter.ClearQuestionText();
```

```
        yield return new WaitForSeconds(2);
        OnGameEnded?.Invoke();
    }

    public bool CheckAnswer(string answer, Answerer answerer)
    {
        OnAnswerGeted?.Invoke();
        if (answer == randomizedQuestions[currentRound].answers[0])
        {
            answerer.rightAnswers++;

            currentRound++;
            NextQuestion();
            return true;
        }
        else
        {
            currentRound++;
            NextQuestion();
            return false;
        }
    }

    private void RandomizeQuestions()
    {
        List<Question> questionsList =
questionsPack.questions.ToList<Question>();
```

```
        for(int i = 0; i < questionsPack.questions.Length; i++)
        {
            int index = Random.Range(0, questionsList.Count);
            randomizedQuestions.Add(questionsList[index]);
            questionsList.RemoveAt(index);
        }
    }
}
```