ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Пояснювальна записка
до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: **“РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ЕЛЕМЕНТАМИ ВИВЧЕННЯ МАТЕМАТИКИ НА ОСНОВІ ПЛАТФОРМИ UNITY ТА МОВИ ПРОГРАМУВАННЯ C#”**

Виконав: студент 4 курсу, групи ПД-42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Кононенко І.В.
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)
Нормоконтроль
(прізвище та ініціали)

Київ - 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ ТА ІНФОРМАТИЗАЦІЇ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти «Бакалавр»

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Негоденко О.В.
“ ” _____ 2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Кононенкові Іллі Віталійовичу

1. Тема роботи: “Розробка програмного забезпечення з елементами вивчення математики з використанням Unity2D для платформи Android”,
керівник роботи: Гаманюк Ігор Михайлович, старший викладач кафедри Інженерії програмного забезпечення,
затверджені наказом вищого навчального закладу від «12» березня 2021 року №65.

2. Строк подання студентом роботи «1» червня 2021 року.
3. Вихідні дані до роботи:
   1. Міжплатформовий рушій для розробки ігрових застосунків – Unity 5.6.7f1.
   2. Середовище розробки MonoDevelop.
   3. Графічний редактор Aseprite.
   4. Мова програмування C#.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   1. Актуальність дослідження
   2. Основна ідея
   3. Розгляд аналогів

4. Розробка застосунку
5. Перелік графічного матеріалу

6. Дата видачі завдання «19» квітня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів	Примітка
1.	Ознайомлення з необхідними матеріалами та літературою	19.04 – 25.04	Виконано
2.	Встановлення вимог до системи	25.04 – 03.05	Виконано
3.	Розробка та тестування	05.05 – 17.05	Виконано
4.	Письмова частина	19.05 – 26.05	Виконано
5.	Здача роботи		

Студент Кононенко І.В.
(підпис) (прізвище та ініціали)

Керівник роботи Гаманюк І.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина роботи містить 45 с., 18 табл., 22 рис., 2 дод., 15 джерел.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ЕЛЕМЕНТАМИ ВИВЧЕННЯ МАТЕМАТИКИ НА ОСНОВІ ПЛАТФОРМИ UNITY ТА МОВИ ПРОГРАМУВАННЯ C#.

Об'єкт дослідження – хід розробки ігрового застосунку для покращення вивчення математики.

Предмет дослідження – програмний застосунок, що дає можливість покращити знання з математики.

Мета дослідження – покращення процесу вивчення математики шляхом створення та застосування програмного забезпечення.

У процесі виконання дипломної роботи був досліджений ринок мобільних застосунків. Було проаналізовано наявні аналоги програмного продукту, порівняно с продуктом у розробці та виявлено їх переваги та недоліки. У роботі було описано інструменти, що використовуються у розробці програмного забезпечення.

Після дослідження наявних програм для розробки було обрано ігровий рушій «Unity» для розробки на платформі «Android». Як основну мову програмування було використано «C#», а у якості середовища розробки обрано швидкий у праці «MonoDevelop». Для дизайну оформлення гри та анімації використовувався графічний редактор «Aseprite».

Цей застосунок призначений до використання учнями шкіл та студентами.

Ключові слова: Unity, C#, Aseprite, MonoDevelop, Android, UML.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HCD – Human-centered Design. Дизайнерський підхід, ставить користувачів на перше місце, у результаті чого виходять корисні продукти та послуги.

UML – Unified Modeling Language. Стандартизована мова моделювання, складається з набору діаграм для визначення, візуалізації, побудови та документування артефактів програмних систем.

ПЗ – Програмне забезпечення.

OS – Operational System. Операційна система.

UI – User Interface. Інтерфейс користувача.

ВСТУП

Об'єкт дослідження – хід розробки ігрового застосунку для покращення вивчення математики.

Предмет дослідження – програмний застосунок, що дає можливість покращити знання з математики.

Мета дослідження – покращення процесу вивчення математики шляхом створення та застосування програмного забезпечення.

Технологія смартфонів принесла багато переваг суспільству, наприклад, дозволила мільйонам людей, які не мають доступу до банків, здійснювати фінансові операції, наприклад, або дозволяла рятувальникам у зоні стихійного лиха точно визначити, де їхня допомога потрібна найбільш терміново. Користувачам смартфонів доступні програми для відстеження того, скільки вони ходять вдень і як добре сплять вночі. Саме тому використання смарт-пристроїв набуло великої відомості в усіх куточках світу.

Однак ці вигоди дорого затратили наше психічне та соціальне життя. Постійний зв'язок та доступ до інформації, яку пропонують смартфони, зробили пристрої чимось на кшталт наркотиків для мільярдів користувачів. В середньому людина витрачає 3 год. 15 хв. кожен день на використання смарт-пристрою.[1] Вчені тільки починають досліджувати це явище, але їх дослідження свідчать про те, що ми дедалі більше відвертаємо увагу, проводимо менше часу в реальному світі і все глибше втягуємось у віртуальний світ. Спираючись на це, можна зробити висновок, що виникає проблема недостачі часу для навчання, яке є дуже актуальним для студентів вищих навчальних закладів та школярів. З урахуванням цього було вирішено розробити програму, яка поліпшить вивчення математики в ігровій формі.

У процесі дослідження вирішувалися наступні завдання:

- Актуальність дослідження. Чи актуальна розробка ігрового застосунку з елементом навчання математиці?

- Вибір інструментів для розробки. Які інструменти підійдуть для розробки найкраще.
- Розгляд аналогів застосунку, який розробляється. Порівняння аналогів у різних категоріях.
- Формулювання системних вимог.
- Опис процесу розробки. Детальний опис усіх дій, які відносяться стосовно нього.

Наукова новизна справжнього дослідження полягає у проектуванні та розробці програмного забезпечення для покращення процесу вивчення математики, використанні статистики, роботі з рушієм Unity, а також з графічним редактором Aseprite для створення оформлення.

Практична значущість полягає у можливості використання мого застосунку для покращення процесу розвитку математичної кмітливості студентами вищих навчальних закладів та школярами.

Дані про використання мобільних пристроїв за останній рік наведена у «Global Digital Overview» від «DataReportal»[2]. Unity 2017.4.40f1 був обраний в якості ігрового рушія.[8] Всю необхідну інформацію можна знайти на офіціальному сайті.[9] Мовою програмування для написання застосунку буде C#. Середовище розробки MonoDevelop буде використане для написання коду.[11] Для оформлення застосунку використовуватиметься графічний редактор та інструмент анімації піксельних зображень Aseprite.[10] Інформація стосовно цієї програми знаходиться на офіціальному сайті розробника.[12]

Загальний процес проектування відноситься до загальної дисципліни «Інженерія програмного забезпечення». Дана дисципліна надає можливість дізнатися про всі тонкощі розробки програм та їх основні принципи та характеристики.

1 РОЗРОБКА МОБІЛЬНИХ ЗАСТОСУНКІВ. ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. ОГЛЯД АНАЛОГІВ

1.1 Важливість мобільних застосунків та принципи проектування

1.1.1 Поширеність цифрових технологій сьогодні

Не можливо не помітити швидке зростання ринку мобільних застосунків. Згідно з даними «Global Digital Overview 2020» від «DataReportal»[2] (рис. 1.1) можна побачити процент кількості користувачів різних мобільних застосунків:

- Користувачі мобільних пристроїв – 5.19 млрд – 67% від населення планети;
- Інтернет-користувачі – 4.54 млрд – 59% від населення планети;
- Користувачі соціальних мереж – 3.80 млрд – 49% від населення планети.

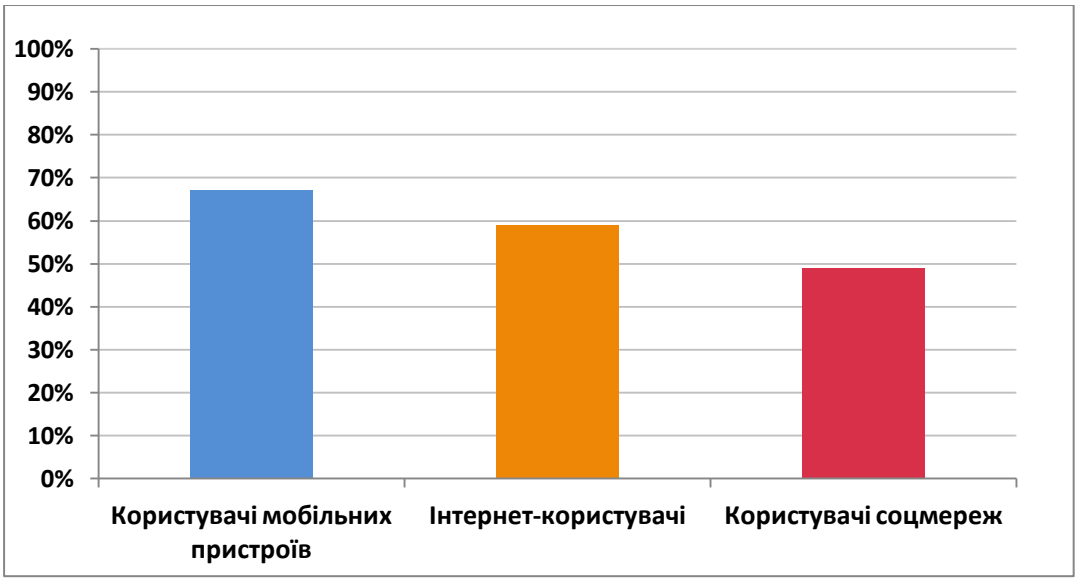


Рисунок 1.1 – Поширеність цифрових технологій на 2020-й рік

На ігрові програми припадає найбільша кількість завантажень та споживчих витрат користувачів Android та iOS. Статистика «The State of Mobile in 2020»[7]

показує, що мобільні застосунки займають 56% ринку, а прибуток від ринку застосунків за 2020-й рік склав приблизно $112 млрд.

Через це розробку під мобільні платформи можна вважати дуже прибутковою і актуальною. Розробка мобільних програм – це процес написання програмного забезпечення для того, щоб скористатися унікальними можливостями конкретних пропозицій мобільних пристроїв. Зазвичай подібні програми пишуться для даних операційних систем: Android OS, Apple iOS, Windows Phone OS, Tizen OS, Symbian OS, KaiOS.

1.1.2 Особливості та принципи розробки

У процесі розробки програмного забезпечення для мобільних пристроїв потрібно звернути увагу на деякі особливості:

- Різний технічний стандарт пристроїв;
- Постійна взаємодія з Інтернетом;
- Адаптованість інтерфейсу користувача;
- Різна потужність мобільних пристроїв.

При розробці мобільного застосунку потрібно зосередитися на принципах Human-centered Design. Це спосіб розробки програмного забезпечення, який бере до уваги потреби користувача. Принципи проектування HCD описані в міжнародному стандарті «ISO 9241-210:2010».[6]

Рисунок 1.2 – Етапи HCD

Процес HCD складається з п’яти етапів, причому етапи 2-5 складають цикл:

- Планування HCD: вибір команди проекту і діяльностей, які будуть відбуватися на кожному етапі.
- Специфікація контексту користування: збір інформації про цілі, завдання, обмеження та контекст роботи користувачів.
- Специфікація потреб: узагальнення отриманої на попередньому етапі інформації та формулювання призначених для користувача вимог до продукту.
- Проектування прототипу: створення макетів, прототипів і різних версій дизайну відповідно до призначених для користувача вимог.
- Оцінка відповідності потребами: оцінка створених рішень силами експертів або із залученням реальних користувачів. Якщо оцінка виявляє будь-які проблеми, то призначені для користувача вимоги необхідно уточнити і доопрацювати макет, прототип або дизайн продукту в відповідно до нових даних.

Подібним чином, розробляючи програмне забезпечення, важливо також правильно вибрати інструменти для використання в процесі. Для розробки ігрового застосунку потрібно вибрати операційні системи, на які дана програма розроблятиметься, а також усі необхідні для неї інструменти.

1.2 Мобільні операційні системи

1.2.1 Ринок мобільних операційних систем

В останні роки провідними операційними системами на ринку є Android та iOS. Разом вони контролюють понад 99% всього світового ринку. Згідно з даними «StatCounter» [3], станом на 2020-й рік (рис. 1.3), відомий наступний процент ринкової долі мобільних пристроїв з даними операційними системами:

- Android – 72.48%.
- iOS – 26.91%.
- Samsung – 0.23%.

- KaiOS – 0.13%.
- Windows – 0.02%.
- Інше – 0.14%.

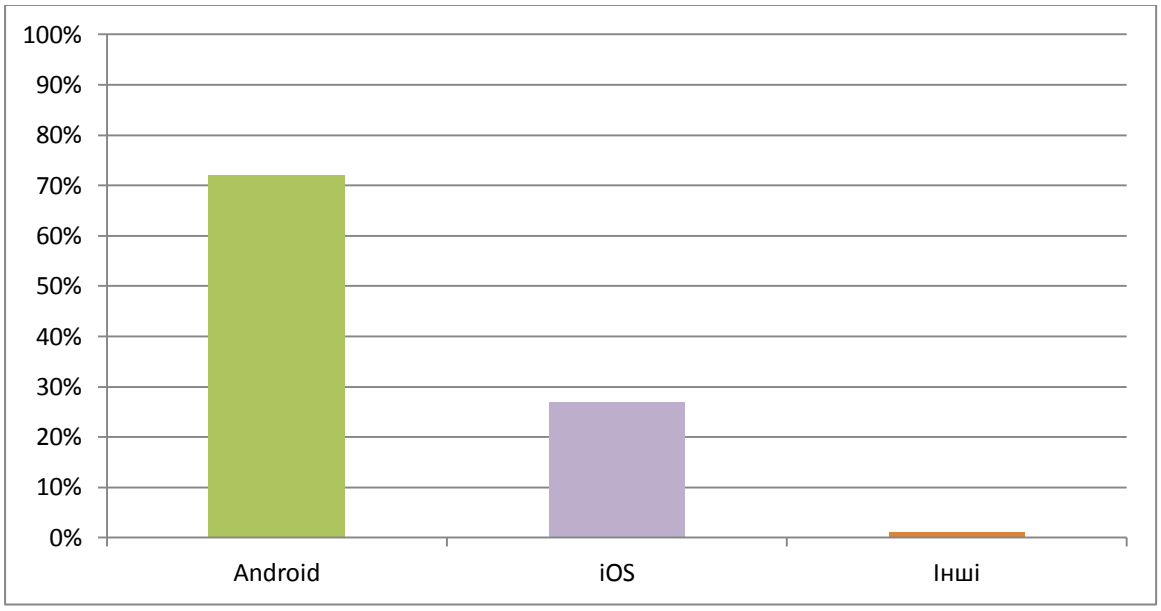


Рисунок 1.3 – Ринкова доля операційних систем

З цього можна зробити висновок, що розробка програм на операційні системи Android та Apple є найбільш прибутковою.

1.2.2 Різниці Android та Apple та публікації в їх магазинах

Можна виділити наступні різниці між Google Android та Apple iOS:

- iOS – це закрита система, тоді як Android – більш відкрита. Користувачі майже не мають системних дозволів в iOS, але в Android користувачі можуть легко налаштувати свої телефони;
- Програмне забезпечення Android доступне багатьом виробникам, таким як Samsung, LG тощо. Така доступність може призвести до деяких проблем із якістю дешевих телефонів. Однак iOS жорстко контролюється Apple, і проблем із якістю немає, оскільки моделей мало;

- Застосунки Android отримуються з Google Play, тоді як Apple – з App Store;

Чим же відрізняються магазини Google Play та AppStore та публікація в них застосунків? Спочатку розберемо опублікування:

- Google Play: для того, щоби висунути свою програму у магазин потрібна всього лише реєстрація та разова плата в розмірі 25$, після чого можна подати застосунок на публікацію. Невдовзі він вже буде доступним для завантаження;
- AppStore: даний магазин має жорсткі критерії відбору. Для публікації ви повинні сплачувати щорічну плату у розмірі 99$, а після завантаження програми дочекатися оцінки проекту, яка зазвичай триває два тижні.

Незважаючи на жорсткість відбору, платоспроможність користувачів Apple вища за Android. Проте, для починаючих розробників мобільних програм Android буде пріоритетною платформою, адже публікація більш доступна, а аудиторія користувачів ширша за iOS. Відповідно буде більше відгуків стосовно програми для подальшої розробки. Попри все це є і мінус – Google Play має велику конкуренцію через більшу кількість завантажених застосунків. Статистика «Number of apps available in leading app stores» від «Statista»[4] (рис. 1.4) показує, що на сьогодні користувачі Android мають вибір між 2.8 млн програмами. Це робить Google Play магазином застосунків з найбільшою кількістю доступних варіантів.

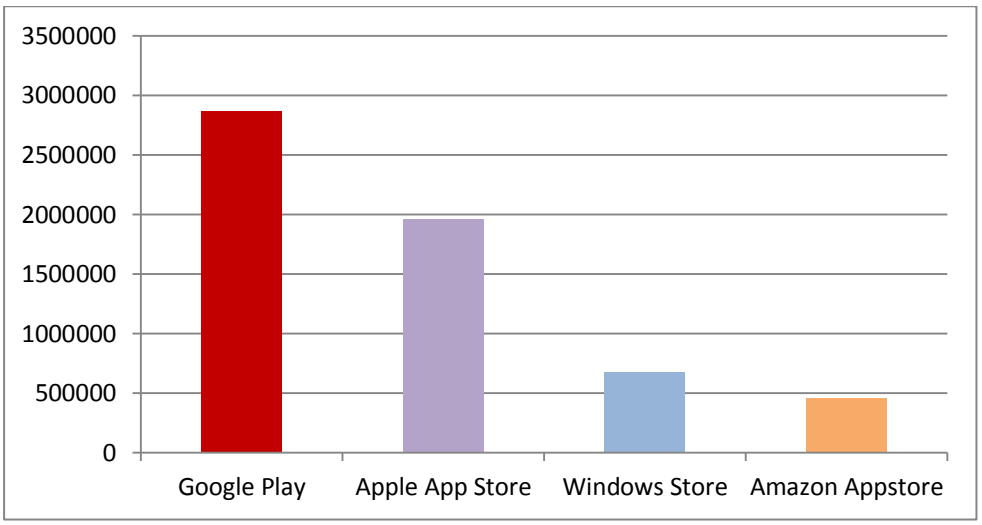


Рисунок 1.4 – Кількість доступних програм у провідних магазинах застосунків

1.3 Unity як ігровий конструктор

Unity[8] – це умовно безкоштовне, інтегроване середовище розробки, а також міжплатформовий рушій для розробки ігор із закритим вихідним кодом. У ньому можна створити свою гру, маніпулюючи об’єктами і прикріплюючи до них різні компоненти.

Для більшості розробників Unity – це улюблений ігровий двигун у світі для створення мобільних ігор. Чому це так популярно? Є декілька причин:

- Розгортання в один клік на Android, iOS, Windows Phone та Tizen;
- Висока оптимізація завдяки таким функціям, як вилучення оклюзій (перепон), групування компонентів з активів, та зменшення розміру складання;
- Послуги монетизації та утримання для мобільних ігор;
- Спеціальні двовимірні та трьохвимірні інструменти, прості в опануванні.

Обираючи для розробки застосунку Unity, вам відкривається доступ до магазину Unity Asset Store. Він містить у собі різноманітні ресурси для розробки проектів. Вести пошук потрібних матеріалів можна за допомогою фільтрів, у тому

числі й цінових. Додатковим плюсом є можливість користування магазином без припинення роботи над програмою.

Також для мобільних розробників стає доступною комплексна платформа Unity Ads. Її можна використовувати для легкої монетизації наявної бази гравців або підсилення стратегії придбання гравців, рекламуючи свою гру в найбільшій ігровій спільноті світу. Unity також підтримує графічні редактори (Adobe Photoshop, Aseprite тощо) та редактор 3D графіки Maya, якими користуються художники та дизайнери для оформлення програмного забезпечення.

1.4 MonoDevelop – інтегроване середовище розробки

MonoDevelop[11] – це інтегроване середовище розробки з відкритим кодом для Windows, Linux, та macOS. Його основним напрямком є розробка проектів, що використовують .NET Framework та Mono. MonoDevelop інтегрує функції, подібні до функцій Microsoft Visual Studio, такі як автоматичне заповнення коду, керування джерелом, графічний інтерфейс користувача та веб-дизайнер. Налаштована версія MonoDevelop була інтегрована в Unity для Windows і Mac до 2018 року. Вона містить просунутий сценарій C#, що використовується для компіляції Unity. Також на відміну від Visual Studio MonoDevelop відчувається легшою та швидшою.

1.5 Aseprite як засіб оформлення застосунку

Aseprite[10] – умовно безкоштовний редактор для створення анімацій та піксельних зображень. Спрайти – це маленькі зображення, які можна використовувати у вашому ігровому застосунку. Ви можете малювати персонажів з рухом, вступи, текстури, візерунки, фони, логотипи, створювати палітри кольорів, ізометричні рівні тощо.

В інструменти Aseprite входить базовий комплекс, який має майже кожний графічний редактор, але з особливостями, які властиві лише піксельним

зображенням (градація з матрицею Байєра), а також лінією часу спеціально для анімацій, з додатковими інструментами для праці з нею (Onion Skin, яка дозволяє бачити попередні та наступні кадри для поліпшення праці з анімацією). Також редактор дає всі необхідні можливості для створення текстурних плиток (Tileset), які зазвичай використовуються у двовимірних іграх.

1.6 Тестування як інструмент перевірки математичних знань

На сьогодні математика є дуже актуальною. Вона дозволяє розвинути важливі розумові якості, а саме: аналіз, дедукція, критика, передбачення.

Також вивчення цієї дисципліни покращує здатність абстрактного мислення[15], концентрації уваги та тренує пам’ять і підсилює швидкість мислення.

Знання математики допомагає і в повсякденному житті: керування часом, планування витрат, ремонт будинку.

Програмісти повинні розуміти математику. При розвитку абстрактного мислення, програміст може краще вивчити проблему, розбити її на логічні частини та знайти нові шляхи розв’язання проблеми. Ви повинні вміти працювати в будь-якій ситуації – незалежно від того, яку програму ви пишете, ви повинні вміти оцінювати її. Навіть більше, перед написанням коду, лише для розуміння складності та ефективності, потрібно скористатися певним алгоритмом і, можливо, скасувати інші алгоритми, щоб зробити його більш ефективними.

Тестування є дуже ефективним інструментом контролю знань. Воно дозволяє швидко отримати відомості про рівень знань перевіреної особи. Тести мають дуже широкий спектр використання. Наприклад, його можна використовувати для оцінки знань, а також для вивчення та підготовки студентів. У систематизованому дистанційному навчанні це основна форма контролю, тому тестування можна розглядати як важливу частину освітнього процесу, а від його змісту – загальний рівень освіти, особливості управління якістю технологій навчання.

1.7 Порівняння з наявними аналогами

В процесі проектування для порівняння були розглянуті наявні аналоги програмного забезпечення в розробці. Дані аналоги всі представлені в магазині Google Play.

Після проведеного розгляду було складено дану порівняльну характеристику:

Таблиця 1.1 – Порівняння аналогів застосунку

	1	2	3	4	5	6	7
Назва	Super Meat Boy	That Level Again	Tricky Castle	Math – Quiz Game	Witcheye	Sonic Runners	CatchUP!
Масшатбованість	Так	Так	Так	Так	Так	Так	Так
Онлайн-магазин	Ні	Ні	Ні	Ні	Ні	Так	Ні
Орієнтація екрану	Landscape	Landscape	Landscape	Landscape	Landscape	Landscape	Landscape
Ел. швидкості	Так	Ні	Ні	Так	Ні	Ні	Так
Лише одна мапа	Ні	Так	Ні	Ні	Ні	Ні	Так
Ел. головоломки	Ні	Так	Так	Так	Ні	Ні	Так
Ел. навчання	Ні	Ні	Ні	Так	Ні	Ні	Так

Виходячи з таблиці до переваг застосунку «CatchUP!» можна віднести:

- Наявність елементу навчання;
- Наявність елементу головоломки;
- Наявність елементу швидкості.

1.8 Висновки дослідження

На початку роботи над застосунком було проведено дослідження ринку мобільних операційних систем. Виявилося, що найбільш використовуваною та простою у опануванні та праці системою є «Android». У процесі підготовки було оглянуто та описано інструменти для розробки, їх переваги.

Як результат дослідження для розробки було обрано рушій «Unity». Даний рушій є дуже популярним, простим у опануванні та зручним для розробки мобільних додатків.

В якості мови програмування виступатиме C#, через її різноманітність. Як середовище розробки я обрав MonoDeveloper, адже він є дуже простим та зручним, а також легким і швидким для праці.

За елемент навчання відповідальні тестові запитання з математики. На них потрібно дати відповідь, якщо гравець встигне дістатися призначеного пункту за виділений для нього час. Математика обрана через її актуальність, адже саме вона відповідає за розвиток таких навичок, як аналіз, дедукція, критика, передбачення. Вона також допомагає мислити абстрактно та дуже сильно допомагає у повсякденному житті, будь то керування часом або ремонт будинків.

2 ВИМОГИ ДО ЗАСТОСУНКУ

2.1 Вимоги до програмного забезпечення та їх види

Для того, аби почати складати вимоги до ігрового застосунку, потрібно розібрати їх значення та класифікацію.

Вимоги до програмного забезпечення ─ це сукупність тверджень стосовно атрибутів, властивостей або якостей програмної системи, яку потрібно реалізувати. Дані твердження створюються в процесі розробки вимог до програмного забезпечення, в результаті аналізу вимог.

Вимоги можуть виражатися у вигляді текстових тверджень або графічних моделей.

У класичному технічному підході сукупність вимог використовується на стадії проектування ПЗ. Вимоги також використовуються в процесі перевірки ПЗ, бо тести ґрунтуються на певних вимогах.

Етапу розробки вимог, можливо, передувало техніко-економічне обґрунтування, або концептуальна фаза аналізу проекту. Фаза розробки вимог може бути розбита на виявлення вимог (збір, розуміння, розгляд та з’ясування потреб зацікавлених осіб), аналіз (перевірка цілісності і закінченості), специфікація (документування вимог) і перевірка правильності.

Розрізняють два види вимог за характером: функціональний (вимоги стосовно поведінки системи) та нефункціональний (вимоги стосовно характеру поведінки системи).

Вимогами функціонального характеру є:

- Бізнес-вимоги;
- Користувацькі вимоги;
- Функціональні вимоги.

Бізнес-вимоги висловлюють мету, заради якої створюється продукт (навіщо він взагалі потрібен, яким чином він буде приносити прибуток). Вони часто представлені простим текстом, без будь-яких технічних подробиць.

Користувацькі вимоги описують завдання, які може виконувати користувач за допомогою системи. Оформляються у вигляді користувацьких історій (user stories, cases). Ці вимоги можуть бути використані для оцінки часу, складності, вартості розробки.

Функціональні вимоги описують що повинна і не повинна робити система.

Вимогами нефункціонального характеру є:

- Бізнес-правила;
- Системні вимоги та обмеження.

Бізнес-правила визначають обмеження, що виникають з предметної області і властивостей об’єкта, що автоматизується (підприємства).

Системні вимоги та обмеження визначають елементарні операції, які повинна мати система, а також різні умови, які вона може задовольняти. До системних обмежень відносяться обмеження на програмні інтерфейси, вимоги до атрибутів якості, вимоги до вживаного обладнання та програмного забезпечення.

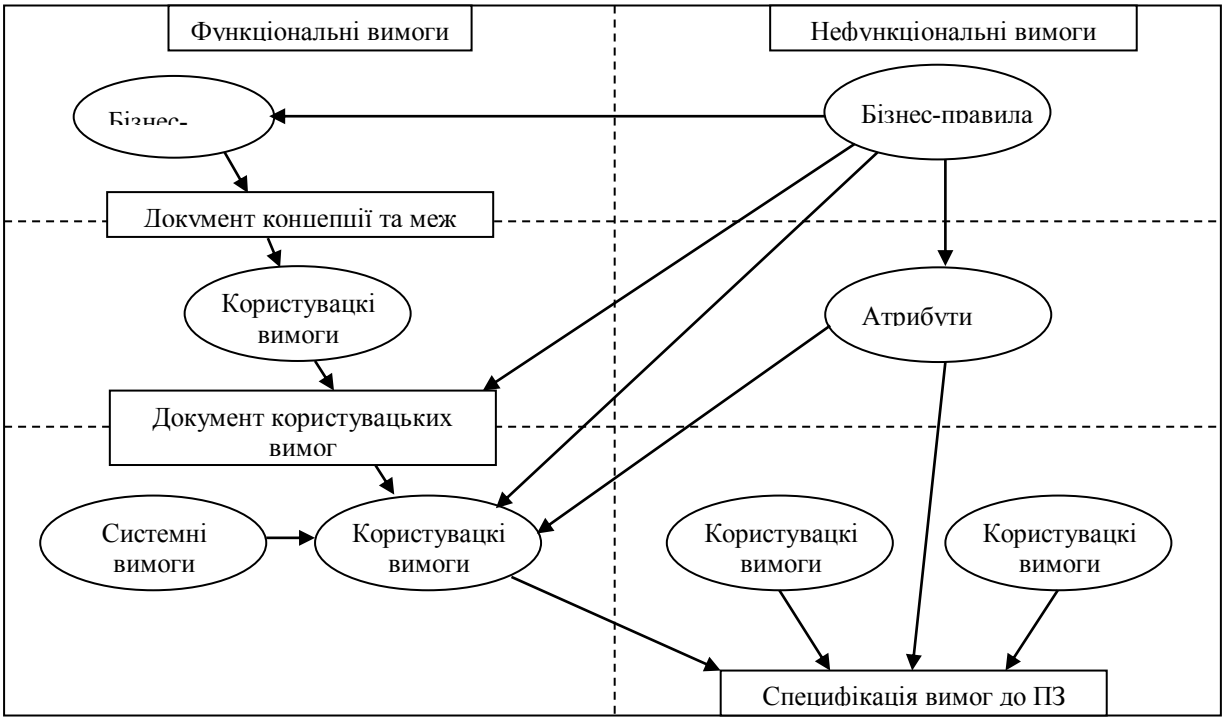


Рисунок 2.1 – Класифікація вимог до програмного забезпечення за Карлом Вігерсом

2.2 User stories як формулювання намірів

User story (історія користувача) ─ це коротке формулювання наміру, що описує щось, що система повинна робити для користувача.

Всупереч тому, що user story грають у величезній мірі роль, яка раніше належала специфікаціям вимог, вони все ж відчутно відрізняються рядом тонких, але критичних нюансів:

- Вони не є детальним описом вимог, а являють собою скоріше обговорюване подання наміру;
- Вони є короткими та легкими для розуміння;
- Їх відносно легко оцінювати, тобто зусилля для реалізації можуть бути швидко визначенні;
- Вони не займають документи, а скоріше складають список, який значно легше упорядкувати при отриманні нової інформації;
- Вони вимагають мінімум або зовсім не вимагають супроводу і можуть бути безпечно скасовані після їх виконання.

Текст самої user story повинен пояснювати роль або дії користувача в системі, його потребу і та цінність після виконання історії.

Формулювання складається приблизно таким чином: як користувач, я щось хочу отримати, з якоюсь метою.

З використанням даного формулювання мною були складені наступні користувацькі історії для застосунку «CatchUP!»:

- Як користувач, я хочу мати можливість бачити таймер на кожному рівні, для планування моїх подальших дій;
- Як користувач, я хочу мати можливість вивчити мапу рівня, без обмеження таймером, для простішого планування дій;
- Як користувач, я хочу мати можливість регулювати гучність застосунку, для комфортного його використання в людних місцях;
- Як користувач, я хочу мати можливість переглядати рекорди рівнів, для відстеження ігрового прогресу;

- Як користувач, я хочу мати можливість вільно вибирати вже виконані рівні, для повторної гри;
- Як користувач, я хочу, аби в грі було збереження даних, для збереження ігрового прогресу;
- Як користувач, я хочу, аби можна було поставити гру на паузу, щоб тимчасово зупинити рівень та зробити перерву;
- Як користувач, я хочу мати можливість стерти весь мій ігровий прогрес, аби почати гру заново;
- Як користувач, я хочу мати кнопку виходу зі гри, аби завершити працю застосунку.

Саме за допомогою користувацьких історій ми можемо побачити, чого саме очікує користувач під час використання програмного застосунку, важливі функції та причини виникнення подібних вимог.

2.3 Use case як опис взаємодії зі системою

Use case (варіант використання, прецедент) ─ це сценарна техніка опису взаємодії. За допомогою use case може бути описана і призначена для користувача вимога, і вимога до взаємодії систем, і опис взаємодії людей і компаній в реальному житті.

У розробці ПЗ цю техніку часто застосовують для проектування й опису взаємодії користувача і системи, тому назва use case часто сприймають як синонім вимоги людини-користувача до вирішення певної задачі в системі.

Під час проектування застосунку я створив дані вимоги до мого програмного продукту «CatchUP!»:

Після запуску додатку, користувач повинен мати можливість регулювати гучність додатку в налаштуванні. У вікні меню є кнопка «Settings», яка дозволяє відкрити вікно налаштувань. В ньому є повзунок, який дозволяє змінювати рівень гучності. За замовчуванням встановлена висока гучність (табл. 2.1).

Таблиця 2.1 – Регулювання гучності

Use case 1	
Дієві особи	Користувач, система
Ціль	Зміна рівню гучності
Сценарій	- Користувач запускає застосунок; - В головному меню користувач натискає на кнопку «Settings»; - В вікні налаштування користувач регулює повзунок гучності.
Результат	Гучність змінена згідно рівню повзунка

Для перегляду рекордів пройдених тестів в відповідних рівнях, буде створене спеціальне вікно для кожного рівня. Перейти до даного вікна можна, натиснувши на кнопку «Levels» та вибравши відповідний рівень (табл. 2.2).

Таблиця 2.2 – Перегляд рекордів

Use case 2	
Дієві особи	Користувач, система
Ціль	Перегляд рекордів тестів у відповідних рівнях
Сценарій	- Користувач запускає застосунок; - В головному меню користувач натискає на кнопку «Levels»; - В меню вибору рівнів користувач натискає на вже пройдений ним рівень.
Результат	Відчиняється вікно рівня, де можна побачити рекорд тесту

У грі буде реалізований музикальний супровід, який не переривається та не зупиняється у процесі гри. У процесі перемикання між меню та/або рівнями музика не зупиняється (табл. 2.3).

Таблиця 2.3 – Безперервна музика

Use case 3	
Дієві особи	Користувач, система
Ціль	Музика, що не переривається
Сценарій	- Користувач запускає застосунок; - Починає гру за допомогою вибору рівнів.
Результат	Музика грає без перерви при переході з меню в саму гру

У грі буде створена кнопка, що стирає весь ігровий прогрес. Для того, аби його стерти (рекорди, відкриті рівні тощо) потрібно натиснути на кнопку «New Game» (табл. 2.4).

Таблиця 2.4 – Нова гра

Use case 4	
Дієві особи	Користувач, система
Ціль	Стерти всі дані гри для нової гри
Сценарій	- Користувач запускає застосунок; - В головному меню користувач натискає на кнопку «New Game»;
Результат	Всього буде відкритим лише перший рівень, а всі рекорди зникнуть

Буде реалізований перезапуск рівня при проходженні або програшу. Перезапустити рівень можна, програвши або виграви у відповідних вікнах (табл. 2.5).

Таблиця 2.5 – Перезапуск рівня

Use case 5	
Дієві особи	Користувач, система
Ціль	Перезапуск рівня в разі проходження або програшу
Сценарій	- Користувач запускає застосунок; - В головному меню користувач натискає на кнопку «Levels» або «New Game»; - Вибирає будь-який рівень; - Програє або виграє рівень;
Результат	Відчиняється вікно програшу або виграшу, де є кнопка «Reset»

Після запуску додатку, якщо користувач пройшов декілька рівнів, він повинен мати можливість вибирати між ними та наступним рівнем, який йому потрібно пройти (табл. 2.6).

Таблиця 2.6 – Вибір рівнів

Use case 6	
Дієві особи	Користувач, система
Ціль	Вибрати рівень
Примітка	Вибирати можна лише доступні рівні
Сценарій	- Користувач запускає застосунок; - В головному меню користувач натискає на кнопку «Levels»; - В меню вибору рівнів користувач натискає на вже пройдений ним рівень, або на рівень, який для нього відкритий, але він ще його не пройшов; - Йому відкриється вікно рівня, де він натисне на кнопку «Start».
Результат	Рівень обрано і розпочато.

В середині рівня, якщо користувач встигне дібратися до призначеної двері, йому відкриється тест. Користувач повинен розпочати тест та відповісти на запитання. У разі правильної відповіді, його оцінка збільшиться на 1. У разі неправильної відповіді, оцінка за одне питання не буде збільшена (табл. 2.7).

Таблиця 2.7 – Відповідь на запитання тесту

Use case 7	
Дієві особи	Користувач, система
Ціль	Відповідь на запитання тесту
Сценарій	- При досягненні та взаємодії з призначеною точкою, користувачу відкриється вікно тесту; - Користувач розпочинає тест, натиснувши на кнопку «Start Quiz»; - Користувач обирає вірний варіант.
Результат	До оцінки додається 1 бал.
Розширення:	
1)	Користувач обирає невірний варіант. - За невірну відповідь на запитання 1 бал до оцінки не додається.

Під час тесту, користувач повинен відповісти на 5 запитань. Якщо він відповість хоча б на 3 запитання, відкриється вікно перемоги. В інакшому випадку, відкриється вікно програшу (табл. 2.8).

Таблиця 2.8 – Проходження тесту

Use case 8	
Дієві особи	Користувач, система
Ціль	Проходження тесту
Сценарій	- Користувач розпочинає тест, натиснувши на кнопку

Продовження таблиці 2.8

Сценарій	- «Start Quiz»; - Користувач обирає хоча б 3 вірних варіанти.
Результат	Після відповіді на всі 5 запитань, відкриється вікно перемоги
Розширення:	
1)	Користувач відповідає на 3 запитання невірно - Після відповіді на всі 5 запитань, відкриється вікно програшу

2.4 Оформлення use case за допомогою діаграм UML

Use case можна оформити не лише текстом, а й за допомогою UML-діаграм. Основними складовими такої діаграми є:

- Actor (дієва особа) – користувач системи;
- Use case (варіант використання) – прецедент взаємодії користувача з системою;
- Коментарі;
- Відношення між actor і use case.

Взаємодіями між дієвою особою та варіантом використання є:

- Include – використовується для вилучення фрагментів випадків використання, які дублюються у багатьох випадках використання. Включений варіант використання не може стояти окремо, а оригінальний варіант використання не є повним без виключеного. Це слід використовувати економно лише у випадках, коли дублювання є значним і існує за задумом (а не за збігом обставин);
- Extend – використовується, коли варіант використання умовно додає кроки до іншого випадку використання першого класу.

Тобто якщо казати простіше, то Include показує, що саме використовує оригінальний варіант для виконання задачі і хоча б один варіант є обов’язковим

до виконання, коли як Extend всього лише розширює додатковими функціями варіант використання.

Однак, ці функції не є обов'язковими до виконання, аби продовжити працю з ПЗ.

Коли як діаграма use case показує процес взаємовідносин між дієвою особою та варіантами використання, діаграма діяльності (activity) показує потік управління. Вершинами є дії, і переходи потоку між ними здійснюються, коли дія завершена.

Основними складовими діаграми діяльності є:

- Чорне коло – початкове положення;
- Стрілки – перехід;
- Прямокутник з округленими кутами – положення дії;
- Ромб – положення рішення;
- Риски – синхронізатор процесів;
- Обведене чорне коло – кінцеве положення.

Діаграма розгортання показує графічне представлення інфраструктури, на якій буде розроблено додаток: топологія системи та розподіл компонентів за її вузлам, а також з'єднання – маршрути передачі даних між вузлами. Діаграма допомагає більш раціонально організовувати компоненти, поміж інших і продуктивність системи, а також вирішити додаткові завдання, наприклад, пов'язані з безпекою.

У процесі планування ПЗ за допомогою UML-діаграм варіантів використання були представлені основні функції системи, їх взаємодія та користувацько-системна взаємодія. Функціонал системи був описаний з використанням UML-діаграми діяльності.

Для проектування діаграм я використаю UMLetino, дуже надійний редактор UML-діаграм з усім потрібним функціоналом.[5]

На першій діаграмі (рис.2.2) було зображене головне меню ігрового застосунку. У ньому користувач має можливість почати нову гру, перейти в меню

вибору рівнів, керувати звуком та припинити дію застосунку. Під час виходу система зберігає дані щодо рекордів та пройдених рівнів.

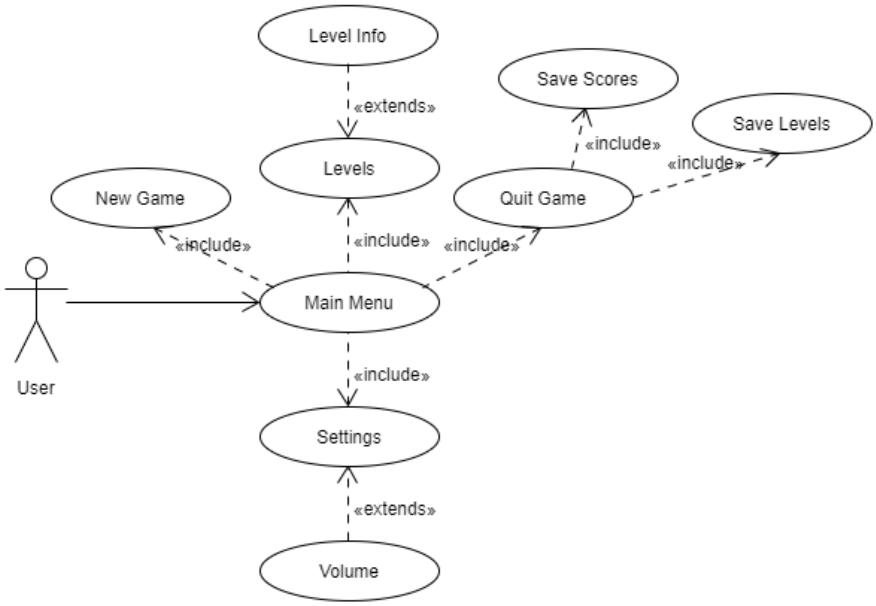


Рисунок 2.2 – Діаграма головного меню

Аби переглянути рекорди та доступні рівні, користувач має зайти в меню рівнів (рис 2.3). Натиснення кнопки «Levels» в головному меню відправить його прямо туди. Якщо в меню рівнів натиснути на один з них, тоді можна побачити рекорд пройденого користувачем тесту на відповідному рівні. Вибір можна здійснити лише між відкритими рівнями (що враховує у себе як пройдені рівні, так і доступні, але ще не пройдені).

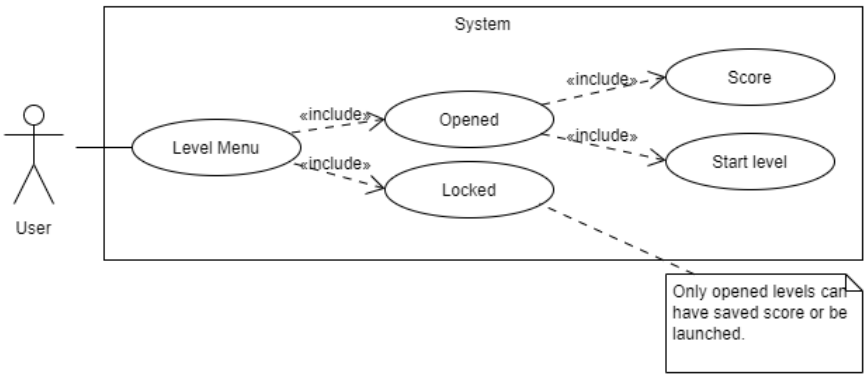


Рисунок 2.3 – Діаграма меню вибору рівнів

При вході в рівень (рис. 2.4) починається його проходження. Стартує таймер, який по закінченню автоматично зараховує програш та видає вікно програшу. Якщо користувач добереться до зазначеного пункту та взаємодіє з ним вчасно, розпочнеться тест. Після завершення тесту, рекорд зберігається та підраховується системою. Якщо користувач не пройшов тест (не відповів на достатню кількість запитань), то зараховується програш. Інакше з'являється вікно перемоги.

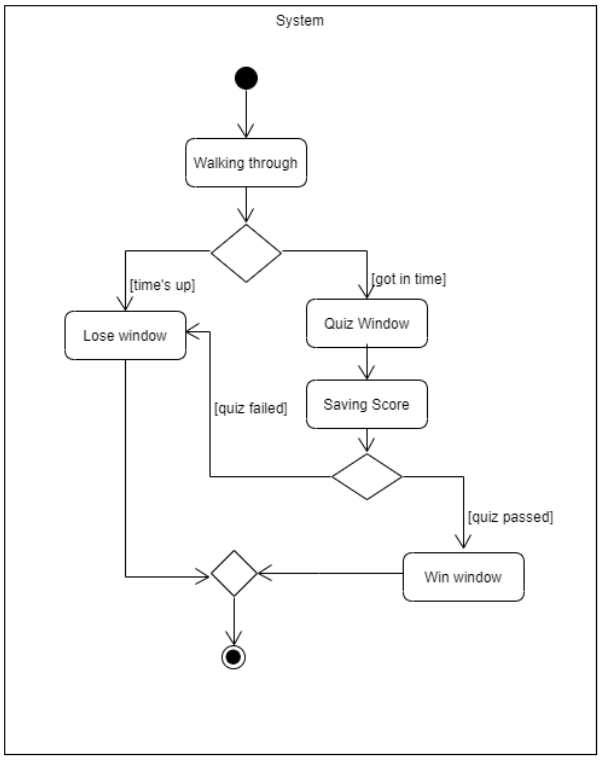


Рисунок 2.4 – Діаграма ігрового процесу

Після початку тестування користувачеві задаватимуться п'ять питань (рис. 2.5). Незалежно від того, правильна відповідь, чи ні, кількість питань, на які було дано відповідь, збільшуватиметься на один за кожне питання. Якщо кількість відповідей досягла п'яти, система підраховує результат. У разі, якщо користувач відповів на три запитання з п'яти невірно, то зараховується програш. Якщо користувач відповів хоча б на три запитання вірно, тоді з'являється вікно перемоги.

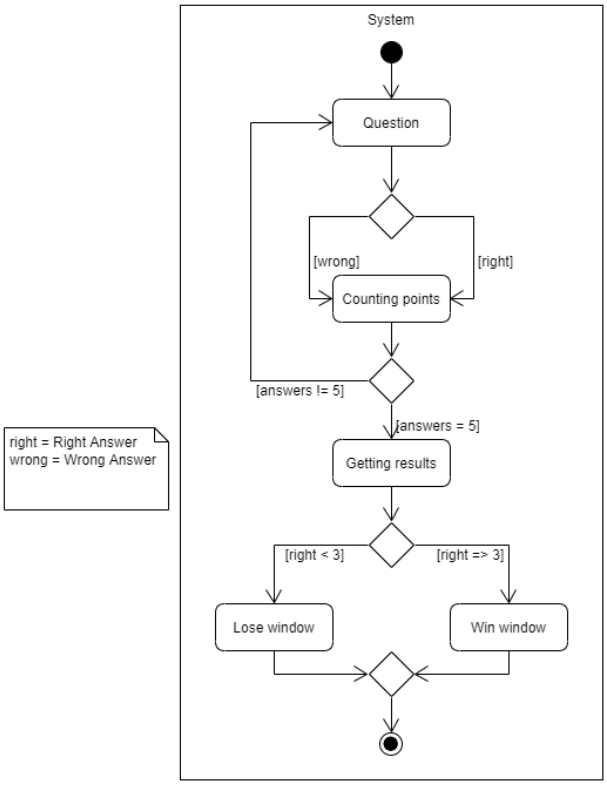


Рисунок 2.5 – Діаграма тесту

У процесі гри користувач має можливість поставити застосунок на паузу. Після даної дії час у грі (а тобто й таймер) зупиняються та з’являється вікно. У вікні користувач може перейти до головного меню, вийти зі гри або продовжити рівень. Після повернення до гри таймер відновить свою дію, а користувач зможе знову рухатися (рис. 2.6).

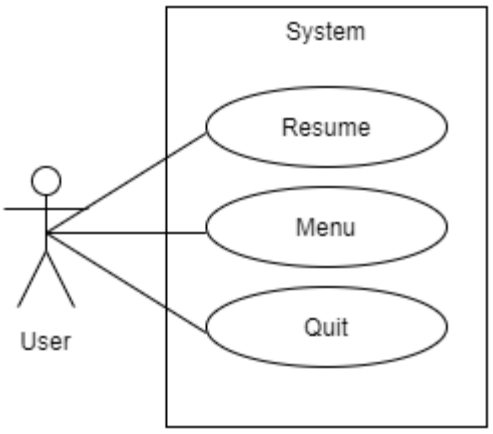


Рисунок 2.6 – Діаграма меню паузи

До меню рівня зберігаються рекорд та відкриті рівні. Ці дані потрапляють у першу чергу до вікна інформації рівня, потім до меню, а потім користувач бачить ці дані на екрані його застосунку (рис. 2.7).

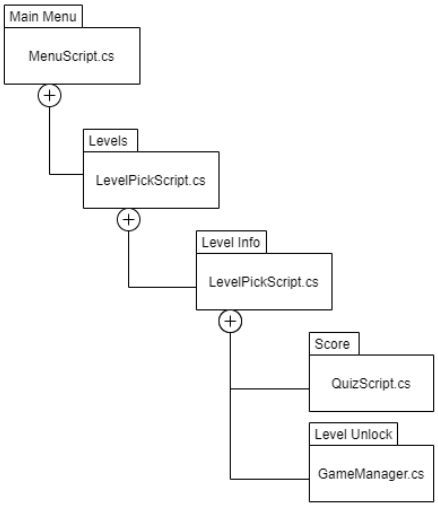


Рисунок 2.7 – Збереження та показ даних рівнів

Коли під час проходження рівня гравець досягає тесту та розпочинає його, генерується одне з п’яти запитань. Дане запитання має чотири відповіді, так лише один з цих варіантів є істино вірним. При новій генерації запитання те питання, що було раніше, не генерується заново.

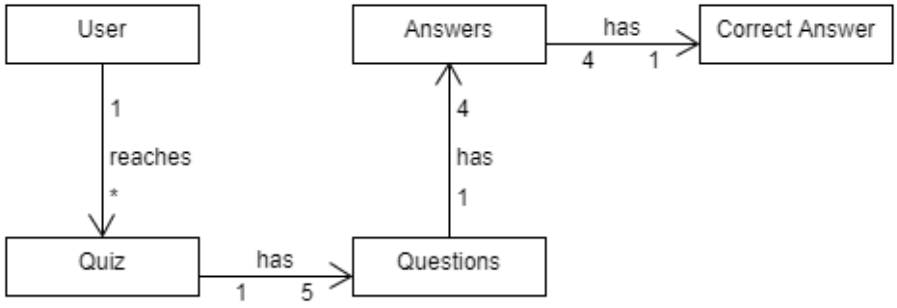


Рисунок 2.8 – Діаграма предметної галузі

2.5 Висновки проектування

На етапі проектування програмного продукту я визначив головні вимоги до нього, а саме склав користувацькі історії та прецеденти використання. Історії були створені для виявлення наміру користувача з приводу того, які саме дії повинна для нього виконувати система. Також були описані дійові сценарії для формулювання того, як користувач має взаємодіяти з системою.

Після цього я відтворив дані вимоги та сценарії за допомогою UML-діаграм, які значно спростили сприйняття їх сутності.

Були створенні наступні діаграми:

- Діаграми діяльностей;
- Діаграми архітектури;
- Діаграма класів;
- Діаграми прецедентів;
- Діаграма предметної галузі;

Не можливо не відмітити те, що під час виконання даного етапу я покращив теоретичні, а також і практичні знання з аналізування вимог щодо програмного забезпечення. Я дізнався про різновиди вимог, їх підтипи, які вимоги є функціональними та навпаки. Також я дізнався, як саме потрібно формулювати вимоги щодо програмного продукту, та як потрібно описувати варіанти використання.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Початок розробки. Підготовка матеріалу, завантаження необхідних матеріалів та середовища розробки

3.1.1 Встановлення Unity – середовища розробки ігор

Аби почати роботу над розробкою ігрового застосунку, нам потрібен власне рушій, на якому ця гра буде будуватися. Встановити Unity можна, якщо завантажити його з офіційного сайту розробника [8]. Є можливість встановити інтегроване середовище розробки Microsoft Visual Studio для написання коду на C#, але через більш повільну роботу комп'ютера я зосередив свій вибір на MonoDevelop, який є значно легшим. Після встановлення середовищ, потрібно створити новий проект в Unity (рис. 3.1).

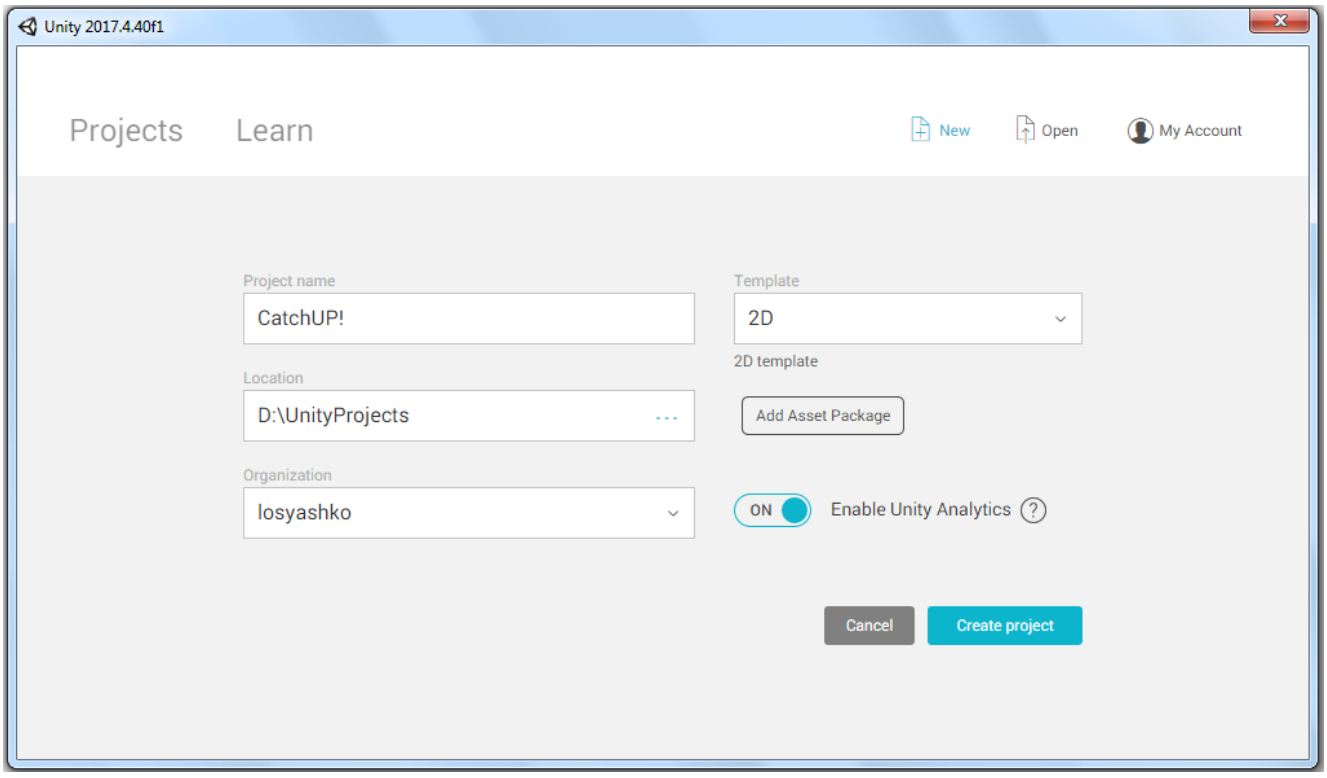


Рисунок 3.1 – Створення нового проекту в Unity

Всього в Unity є 2 шаблони ігор. Дані шаблони відповідають за конфігурацію ігор, відповідно шаблон «2D» є шаблоном, що спрощує розробку

для двовимірних ігор, а «3D» – для тривимірних. Я вирішив, що моя гра буде двовимірною, тому вибрав перший шаблон. Після цього треба вказати місце папки проекту та його назву. Для своєї гри я обрав назву «CatchUP!».

3.1.2 Встановлення Aseprite – графічного редактору для спрайтів

Для розробки ігрового застосунку потрібне візуальне оформлення гри. Встановити Aseprite можна з офіціального сайту розробника [10]. Після завантаження редактору, потрібно його відкрити та створити спрайт (рис. 3.2). Можна змінити початкове налаштування спрайту, такі як ширина та висота, режим кольору та налаштування заднього фону. Всього є три режими кольору:

- RGBA – кожен піксель має RGB та Alpha-компоненти (32 біти/піксель);
- Grayscale – кожен піксель має сіре значення та Alpha-компоненти (16 бітів/піксель);
- Indexed – кожен піксель є виноскою з палітри (8 бітів/піксель);

З налаштуванням заднього фону все дещо простіше: можна змінити його на прозорий, білий, або чорний фон.

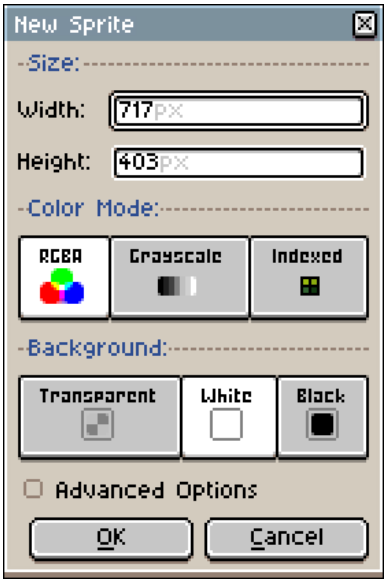


Рисунок 3.2 – Створення нового проекту в Aseprite

Оскільки я працюю зі спрайтами персонажа та набором плиток для мапи, я обрав режим кольору RGBA та прозорий задній фон.

У програмі є непоганий набір для роботи з піксель-артом. Лише у самому інтерфейсі (рис. 3.3) можна побачити різні типи виділення, пензлик, піпетку для копіювання кольорів, заливку (яку можливо змінити на градацію), використання фігур, ліній, розмиття, та багато чого іншого.

Рисунок 3.3 – інтерфейс програми Aseprite

Окрім малювання, програма є дуже хорошим інструментом для піксельної анімації. Тут (рис. 3.4) наявна можливість розробки декількох різних анімацій у одній роботі для подальшого їх окремого збереження. Для цього всього лише потрібно додати окремим сегментам окремі позначки. Як, наприклад, я надав позначку PlayerRun анімації бігу, аби зберегти лише ці 15 кадрів.

Ще однією хорошою функцією можна назвати режим «Onion Skin», що переводиться як «цибулева шкірка», або «тонкий прозорий папір». Дана функція

допомагає орієнтуватися між поточним та попередніми або наступними кадрами (кількість кадрів можна налаштувати).

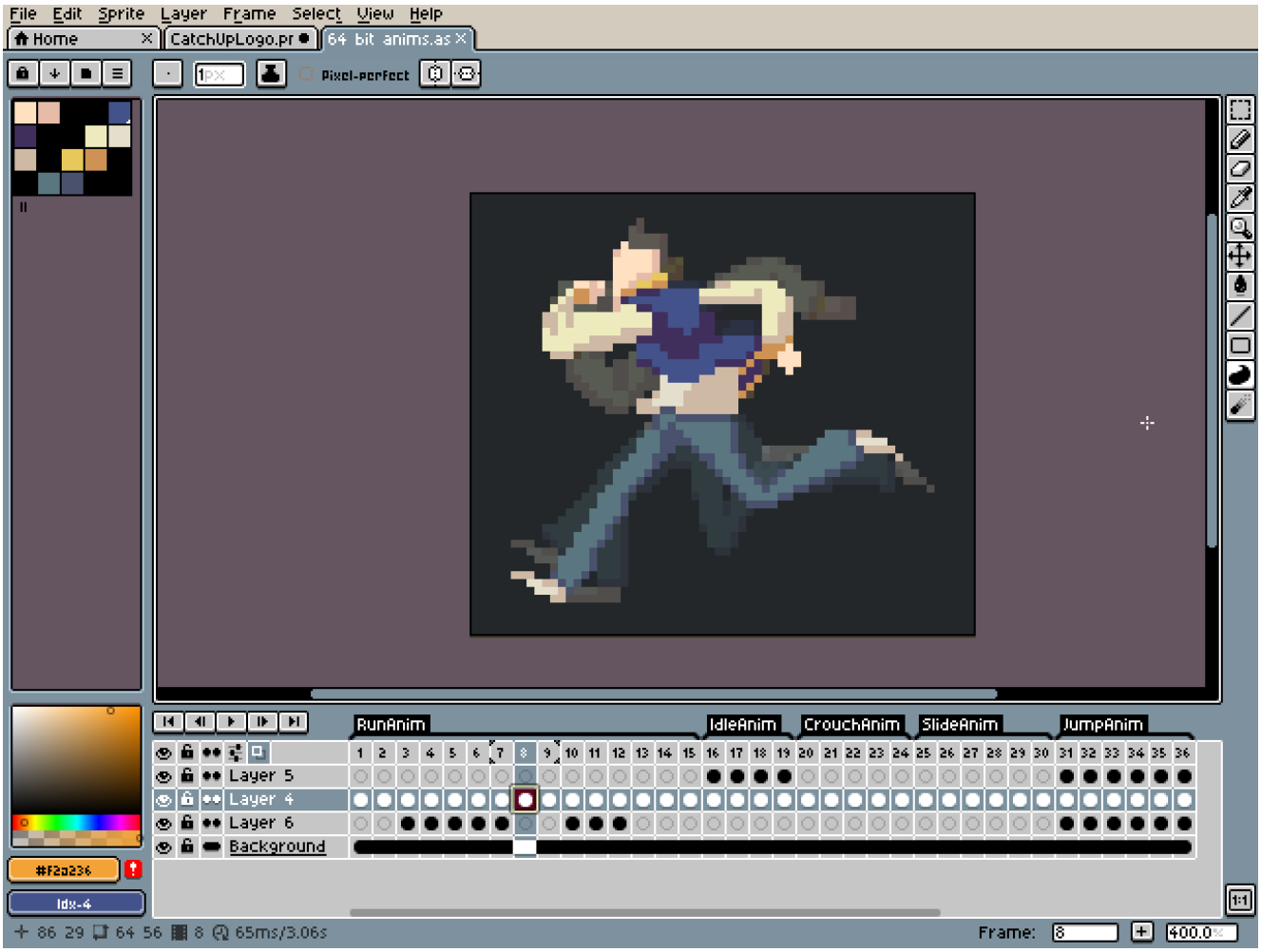


Рисунок 3.4 – анімація в Aseprite

3.1.3 Знаходження матеріалу для розробки

Незважаючи на те, що основною частиною ресурсів для гри займаюсь я, деякі сторонні ресурси ніколи не завадять. Для завантаження нових ресурсів можна скористатися інтегрованим в середовище Unity магазином – Unity Asset Store (рис. 3.3).

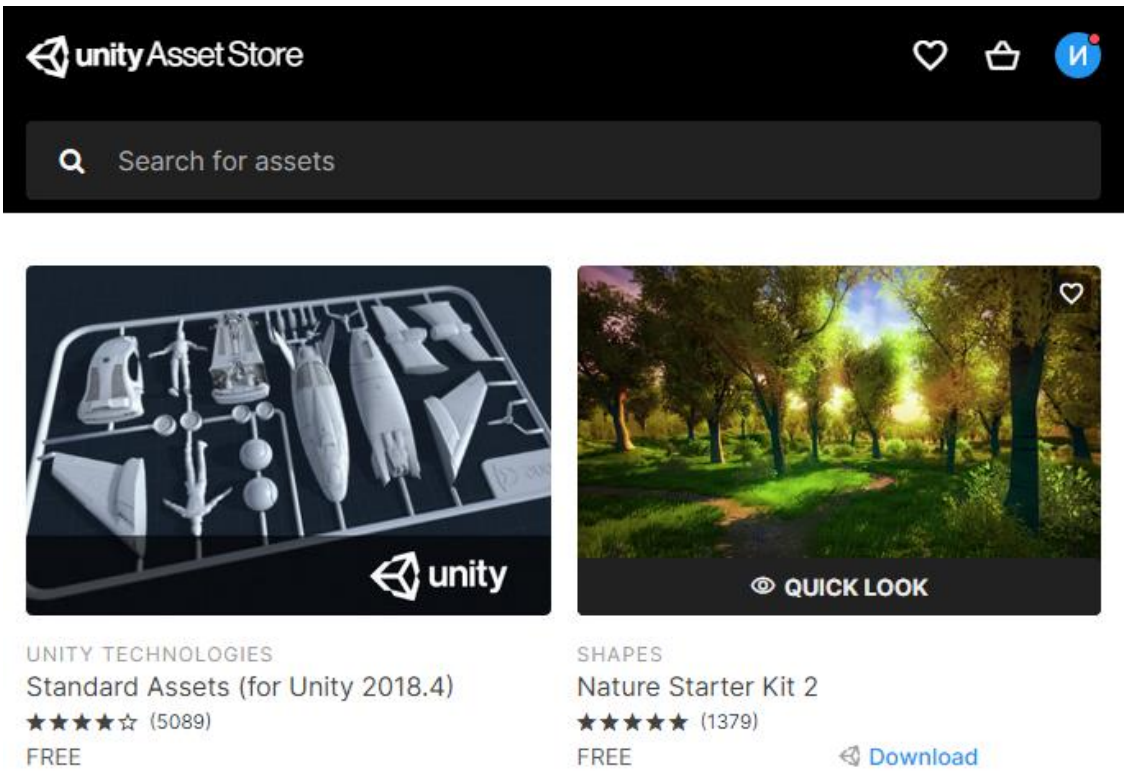


Рисунок 3.5 – Магазин Unity Asset Store

Для розробки будуть корисними безкоштовні пакети ресурсів, які мають в собі:

- UI елементи;
- Музику.

Після завантаження пакетів потрібно імпортувати їх в Unity. Якщо це пакети з Asset Store, то потрібно натиснути на кнопку «Import». Якщо це сторонні пакети, то потрібно в меню ресурсів натиснути на праву кнопку миші та обрати «Import Asset».

3.2 Дизайн рівнів для ігрового застосунку

Кожен рівень використовуватиме одну мапу, на якій будуть розміщені ігрові об’єкти, такі як задній фон, платформи, двері-телепорти та двері-активатори. Для завершення рівня персонажу потрібно встигнути активувати відповідні двері-активатори та пройти тест. Мапа проходиться у випадковому

порядку, визначеному лише самим рівнем, при цьому кінцева точка (активатор) відмічена яскравим жовтим вказівником. По закінченню таймера або неправильних відповідях на тест є можливість перезапустити рівень (рис. 3.4).

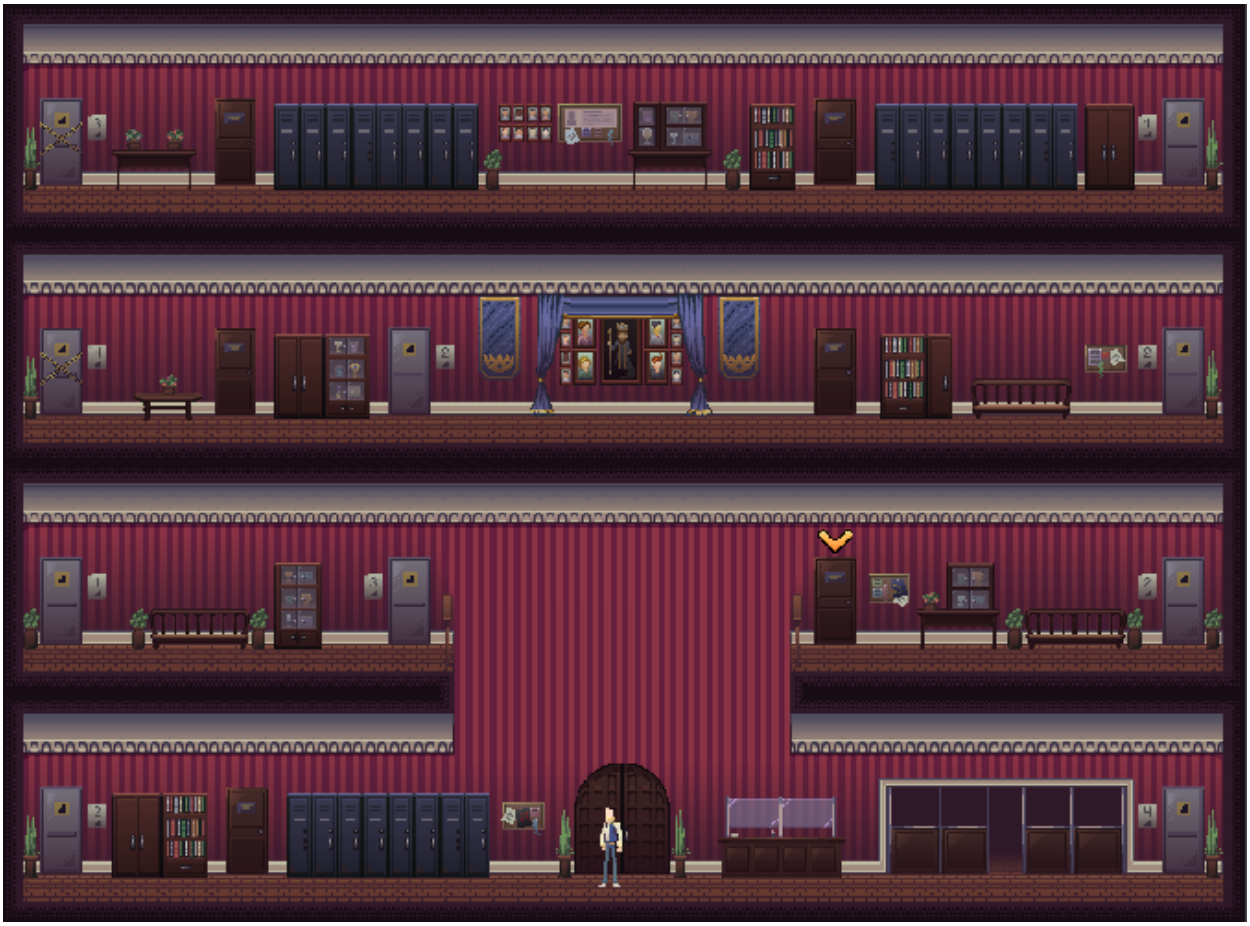


Рисунок 3.6 – Сцена першого рівня

В якості дверей-телепортів виступатимуть двері зі знаком сходів на них. Плакат біля двері вказує, на який поверх персонаж потрапить, якщо скористується нею. Як двері-активатори виступатимуть двері класу (рис. 3.7).

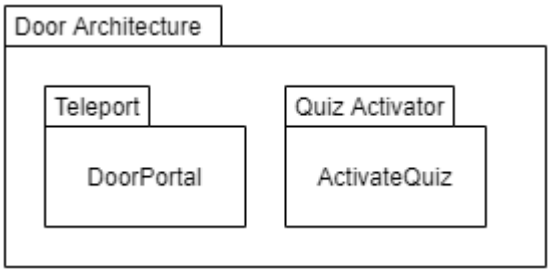


Рисунок 3.7 – Архітектура дверей

На відміну від дверей-телепортів, активувати можна тільки зазначені двері для появлення тесту. На даний момент всього будуть доступні три рівні для проходження і один рівень для дослідження мапи та вивчення шляхів, адже деякі шляхи не поверхи не зовсім очевидні. Наприклад, аби потрапити на четвертий поверх, потрібно скористатися дверми на першому поверсі, адже двері на третьому поверсі «зачинені на обслуговування», а другий поверх розділений надвоє і потрібно знайти обхідний шлях.

У користувача є можливість керувати персонажем, а саме рухатися по горизонталі, стрибати та присідати.

3.3 Написання коду для ігрового застосунку

Створити файл коду у середовищі Unity можна двома чинами:

- При натисненні правої кнопки миші у вікні «Project» обрати пункт «Create C# Script»;
- Додати до GameObject нову складову, дати їй довільну назву (зазвичай це назва того, що робить сценарій), натиснути на кнопку «Create and Add».

Ознайомитись з кодом та всіма сценаріями можна використовуючи рисунки, які наведені у додатку А.

3.3.1 Керування рухом персонажу

Для реалізації керування персонажем потрібно створити два файли-сценарії. Перший файл відповідатиме за контроль персонажу, другий же буде відповідати за власне рух.

Спочатку задамо змінні:

- JumpForce – вертикальна сила, яка додається при виконанні стрибку;
- CrouchSpeed – яка частина максимальної швидкості використовується під час присідання (1 = 100%);
- MovementSmoothing – наскільки сильно буде згладжуватися рух;

- AirControl – можливість контролювати рух у повітрі;
- WhatIsGround – маска, що визначає землю для персонажа, використовує категорію шарів, яка задається у середовищі;
- GroundCheck – об’єкт, який перевіряє, чи стоїть гравець на землі (позиція маркеру налаштовується через GameObject);
- CeilingCheck – об’єкт, який перевіряє, чи є стеля над головою (також налаштовується через GameObject);
- CrouchDisableCollider – колайдер, який буде відключено при присіданні.

SerializeField дозволяє редагувати змінні, при цьому не надаючи їм рівень доступу доступ Public.

Далі задамо змінні, які не можна буде редагувати у середовищі розробки:

- GroundedRadius – радіус кола перевірки того, чи стоїть гравець;
- Grounded – перевірка того, чи стоїть гравець на землі;
- CeilingRadius – перевірка на стелю над головою і на те, чи зможе встати персонаж;
- Rigidbody2D – визначення фізичного тіла персонажу;
- FacingRight – перевірка на те, куди дивиться персонаж;
- Velocity – швидкість.

Тепер потрібно створити події для приземлення. Створимо подію Unity для приземлення OnLandEvent. Після цього зробимо клас, який є булевою подією. У цей клас запишемо подію OnCrouchEvent, а потім додамо змінну wasCrouching, яка відповідає за те, чи підвівся персонаж з присідання.

Тепер, перейдемо до власне коду руху. Створимо клас Move, де задамо змінні move, crouch, jump. Якщо персонаж стоїть на землі, або у нього активний контроль руху у повітрі, тоді активується рух. Персонаж рухається за допомогою знаходження його швидкості, а після цього до неї додається згладжування. Якщо input (вхідна інформація) змушує персонажа рухатися вправо та він дивиться вліво, тоді спрайт персонажа перевертається по горизонтальній осі і починає дивитися в іншу сторону, так і навпаки, якщо рухається вправо і дивиться вліво – перевертається і дивиться вправо.

Сама команда перевертання виглядає таким чином. Якщо FacingRight не є правдою, тоді ми множимо вісь x на -1.

Тепер налаштуємо стрибок персонажу. В разі, якщо персонаж повинен стрибнути, тоді до нього додається вертикальна сила.

Якщо персонаж присідає, тоді його швидкість зменшується згідно швидкості в присіданні. Одночасно з цим відключається верхній колайдер персонажа, що змінює розмір і доступність персонажа до місцевостей. Також може працювати як механізм ухилу. Якщо персонаж більше не присідає, тоді колайдер повертається на місце. Також тут є події. Якщо гравець присідає і не виходить з присяду, тоді активується бінарна змінна wasCrouching і визивається подія OnCrouchEvent. Якщо гравець перестав натискати присідання, тоді wasCrouching деактивується разом з подією.

Тепер потрібно визначитися, коли працюють події приземлення та яким чином перевіряється те, як гравець стоїть на землі. Створимо метод Awake, який активується до початку будь-яких функцій, а також відразу після ініціалізації префабу. У ньому ми отримуємо тіло персонажа, потім якщо подія приземлення більше не є актуальною, то створюється нова така подія. У разі, якщо подія присідання не є актуальною, тоді створюється нова булева подія.

Перевірка того, що гравець стоїть на землі, відбувається наступним чином. Створимо метод FixedUpdate, який викликається кожен раз при розрахунку фізичних показників. Всі фізичні розрахунки проводяться саме у ньому.

Створимо булеву змінну wasGrounded, якій присвоюємо змінну Grounded. За умовчанням задамо йому значення false. Після цього йде перевірка на те, чи стоїть гравець на землі. Якщо так, то змінна Grounded буде активована. Сама перевірка враховує в себе перевірку позиції на землі, наскільки близько знаходиться радіус Grounded та перевіряє те, що являється землею.

Тепер, перейдемо до коду руху. Задамо такі змінні:

- CharacterController2D – посилання на контроль фізики руху;
- Animator – посилання на аніматор персонажа;
- RunSpeed – налаштування швидкості бігу;

- HorizontalMove – рух по горизонталі;
- Jump – стрибок;
- Crouch – присідання.

Зробимо метод Update. Даний метод викликається всього лише раз у кадр, але є залежним від частоти кадрів пристрою. Тобто він викликається тим менше, чим вона нижче. У цьому методі horizontalMove присвоюється ввід з клавіатури або з кнопки (завдяки ресурсу SimpleInput з Unity Asset Store). Далі викликається аніматор, який активує анімацію бігу, якщо horizontalMove більше або менше 0 (менше – рух вліво, більше – вправо). Також, якщо натиснути на кнопку, яка прив’язана до команди стрибку або присідання, то змінні jump та crouch зміняться відповідно до вводу з однією відмінністю:

- Якщо кнопка присідання буде зажатою, тоді присідання буде працювати;
- Якщо кнопку присідання відпускають, присідання більше не активне.

Просто використати SimpleInput для стрибку та присідання не вдалося, тому довелося додати простий сценарій, який активує їх з методів. Його потім можна додати до кнопки у середовищі розробки. IsJumping – одна з умов для аніматора, за допомогою якого програється анімація стрибка.

Створимо методи, які будуть прив’язані до контролеру персонажу для подій приземлення та присідання.

Оскільки персонаж пересуватиметься по сцені, потрібно зробити так, аби камера слідувала за ним. Для цього я створив невеликий сценарій, де головна камера отримує позицію гравця та підлаштовується під неї.

3.3.2 Взаємодія персонажа з об’єктами та таймер рівнів

Аби прогресувати у рівні, персонаж має пересуватися через поверхи на активувати тести з допомогою відповідних дверей. Розберемо код порталу:

- Portal – двері-портал;
- Player – персонаж;
- ActionButton – кнопка активації.

Створимо метод OnTriggerStay2D, який перевіряє те, чи зіткнувся колайдер персонажу з колайдером дверей. Якщо це правда, тоді кнопка активації з'являється, та при натисненні активує локацію порталу-виходу. Також дана кнопка розпочинає корутину, яка активується після очікування в пів секунди. При виході персонажа з тригеру, кнопка активації зникає. Даний код прикріплюється до двері-телепорту.

Рисунок 3.7 – Контроль персонажа

Для пам'яті кнопки, а саме який телепорт був попереднім, можна написати наступний короткий код, який отримується ActionButton.GetComponent. Після створення його потрібно прикріпити до самої кнопки активації.

Далі, потрібно написати сценарій, який змусить кнопку активації виконувати ще одну дію при взаємодії з іншим об'єктом, а саме с дверима класу, які активують початок тестування. Створимо метод OnTriggerStay2D, який також перевіряє зіткнення колайдерів. При натисненні на кнопку буде виконуватися метод ActivateQuiz, який відключить таймер та контроль персонажу, але відкриє вікно тесту. При виході персонажа з тригеру дверей-активаторів, кнопка активації зникає.

При активації рівня повинен стартувати таймер, по закінченню якого рівень повинен припинитися, та повинне з'явитися вікно програшу. Максимальний час можна регулювати у середовищі розробки. На початку рівня таймеру присвоюється картина лінії часу, а залишок часу прирівнюється до максимального часу. Кожен наступний кадр, якщо залишок часу більше 0, то він прирівнюється до різниці залишку часу та Time.deltaTime (який є інтервалом від попереднього кадру до наступного), а наповненість лінії часу прирівнюється до поділу залишку часу на максимальний час. В інакшому випадку, якщо залишок дорівнює нулю, то

з’являється надпис «Time’s Up!», яка інформує нас про те, що час вийшов, а меню керування зникає. Затим відбувається корутина, яка після 1,5 секунди реального часу активує вікно програшу (рис. 3.7).

Рисунок 3.8 – Програш за часом

3.3.4 Розробка вікна тестування

Після появлення відкриття тестування, гравець має натиснути на кнопку «Start Quiz». При натисканні пропаде дана кнопка, відповіді та текст рекорду з’являються, qList буде дорівнювати списку заданого масиву, та виконуватиметься метод генерації запитань. Метод генерації запитань складається наступним чином:

Якщо запитань у списку більше 0, тоді обирається випадкове запитання від 0 до заданого у середовищі розробки цілого числа. Текст запитання відображує вибране випадкове питання. Список відповідей стає новим списком заздалегідь зазначених відповідей до нього. Якщо елементів масиву менше, ніж довжина масиву відповідей, додається по елементу кожну повірку, при цьому генерація відповідей на кнопках теж буде випадковою. У середовищі розробки в списку відповідей елемент з індексом 0 завжди буде вірним, але без такої випадкової генерації можна дуже легко вгадати варіант. Інакше, якщо кількість не згенерованих питань дорівнює 0, зникають елементи запитання, відповідей та

рекорд и з’являється вікно. Якщо кількість правильних відповідей більша за 2, тоді це вікно перемоги. Якщо їх кількість менша за 3, тоді це вікно програшу.

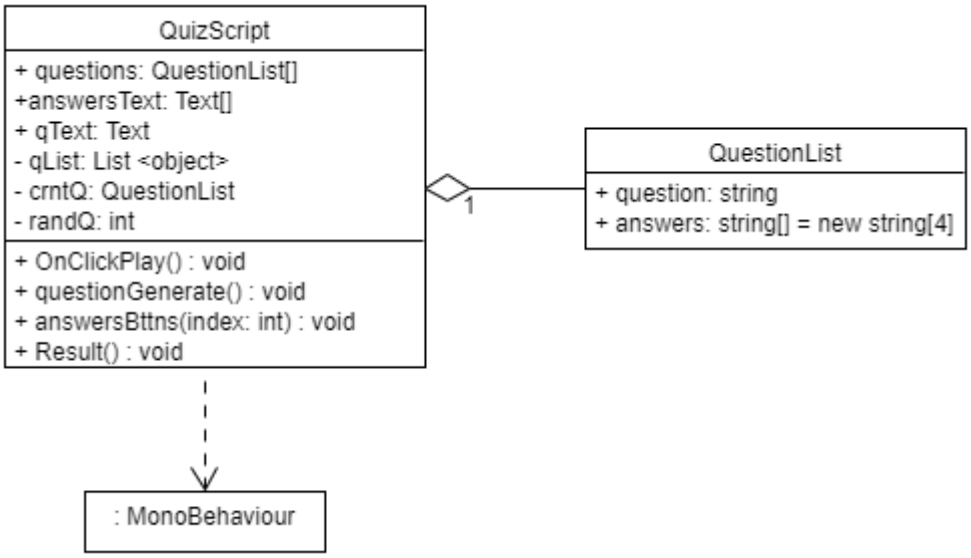


Рисунок 3.8 – Діаграма класу QuizScript

Потрібно за допомогою сценарію виявити, яка відповідь є правильною. Створимо метод answersBttns. Якщо текст відповіді належить до індексу 0, тоді до рекорду додається +1. В інакшому випадку не відбувається нічого, окрім надпису «Wrong answer…» у консолі. Після цього дане запитання видаляється, і нове запитання генерується.

Видача результату виконується дуже просто. Для цього буде використана зручна команда Player.Prefs, за допомогою якої можна зберігати та виставляти дані. Створимо простий метод Result, у якому отримаємо індекс та ім’я сцени. Після цього перевіряємо ім’я сцени, і тоді код зберігає значення рекорду до відповідного рівня.

Написання питань виконується так: створюється новий клас, де можна редагувати надписи. Є можливість змінити запитання та відповіді, а також їх кількість, в Unity.

Для відображення рекорду потрібно створити окремий сценарій. У ньому задаватиметься статичний рекорд, який змінюється лише при втручанні інших файлів коду. Елемент тексту, до якого прикріплений даний код, прирівнюється до

рекорду. На старті ми отримуємо компонент тексту для нього, а при оновленні текст змінюватиметься за допомогою форматування, де є елементи string.

До сценарію тесту також прив’язаний невеликий код ігрового менеджеру. У менеджері ми задаємо ім ’я та індекс рівня, який потрібно відкрити, налаштувати їх можна у Unity. Метод WinLevel зберігає число досягнутого рівня та рівень, який при цьому відкривається.

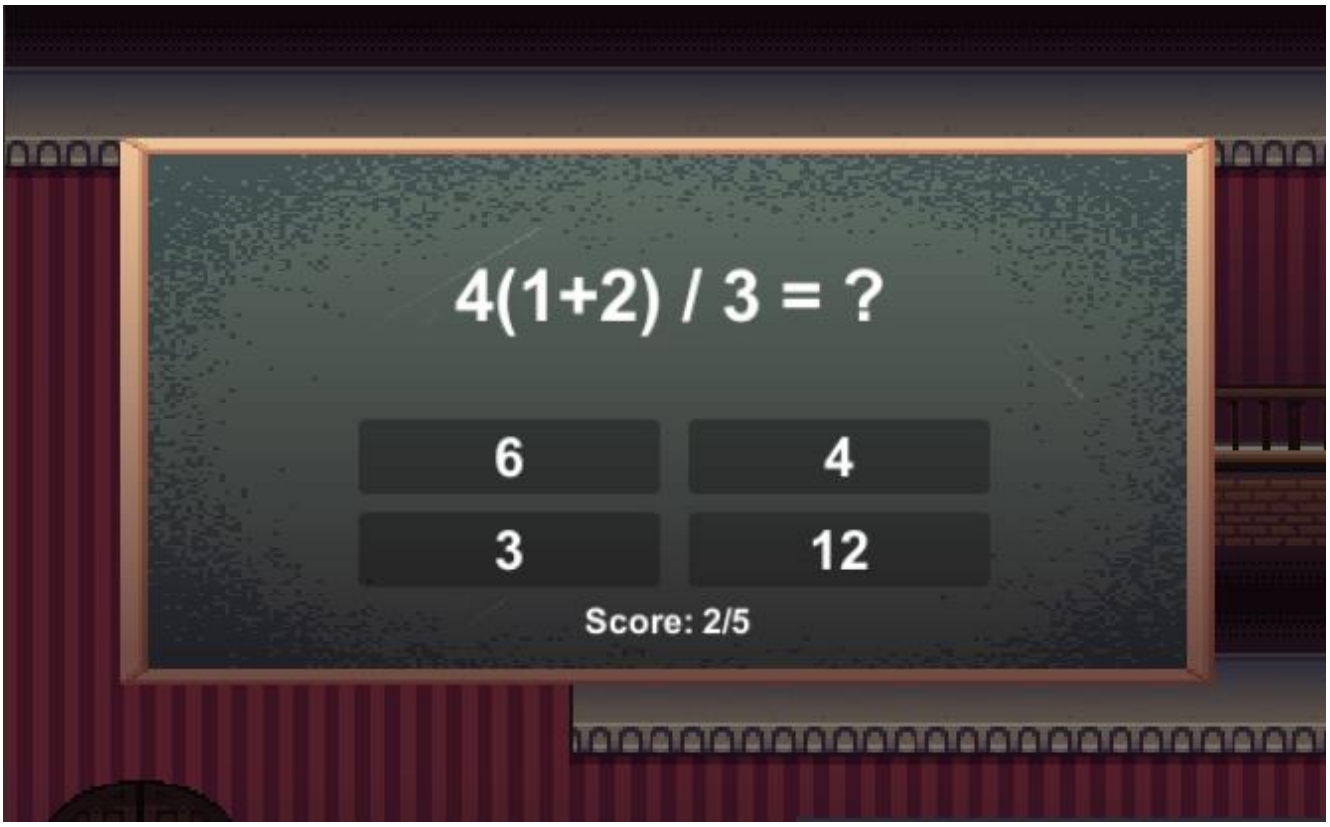


Рисунок 3.9 – Вікно тесту

3.3.5 Розробка меню та контроль звуку

І таким чином ми плавно перепливаємо до теми рівнів. У коді селектору рівнів задаємо довільний масив кнопок, який можна налаштувати. На старті отримується число досягнутого рівня, яке дорівнює одиниці. Потім додаються кнопки згідно довжині масиву. Якщо елемент масиву + 1 менший за число досягнутого рівня, тоді всі подальші кнопки стають неактивними.

Також рівні можна призупинити з використанням кнопки паузи. При натисканні на кнопку з’являється меню паузи, де можна продовжити гру, вийти у головне меню або вийти зі гри.

Тепер стосовно головного меню (рис. 3.8). Наявна можливість анулювати всі рекорди та відкриті рівні. Для цього я створив кнопку «New Game» у головному меню, яка стирає всі збережені дані та повертає значення досягнутого рівня назад до 1. Після виконання анулювання запускається вікно рівнів. Ще у головному меню є кнопка переходу до власне вікна рівнів без анулювання даних, кнопка, що відповідає за вікно налаштувань та кнопка виходу зі застосунку.

Рисунок 3.10 – Головне меню

У налаштуванні можна регулювати гучність музики (рис. 3.9). До повзунку я приєднав сценарій, де його рівень є рівнем звуку. Сама музика була додана за допомогою AudioSource, який відповідає за її програш.

Рисунок 3.11 – Вікно налаштування

Якщо спробувати перейти в іншу сцену, то можна помітити, що звук працює тільки в меню. У цьому випадку я хочу, аби музика плавно переходила між сценами без перезапуску або зупинки. Якщо просто додати AudioSource до іншої сцени, то вона буде перезапускатися. Для плавного переходу фонової музики я написав сценарій, що допоможе у цьому. На старті ним буде отримуватися рівень гучності, який був заданий у головному меню. Після цього я створив перевірку на те, чи є інші аудіо менеджери (на випадок, якщо я випадково залишу десь лишній менеджер). Якщо таких немає, тоді програється наша музика з меню, без видалення при завантаженні сцени.

3.3.6 Тестування функцій застосунку

Тестування ПЗ – це засіб перевірки дотримання фактичного програмного продукту очікуваним вимогам для забезпечення відсутності недосконалостей. Воно завбачає виконання системних та програмних компонентів з використанням автоматизованих або ручних інструментів, аби оцінити одне або декілька властивостей. Мета тестування – це виявлення відсутніх вимог або помилок різних видів.

Для того, аби перевірити вимоги, потрібно описати варіанти тестування (test-case). У них я опишу перевірку на дотримання вимог, заявлених у дипломі.

Таблиця 3.1 – Варіант тестування №1

Номер тестування	1
Назва	Таймер
Кроки	1. Запустити програму; 2. Перейти у меню рівнів; 3. Вибрати довільний рівень окрім «Explore!»; 4. Натиснути «Start».
Результат	При завантаженні рівня з’являється таймер, який дозволяє слідкувати за часом рівня.

Таблиця 3.2 – Варіант тестування №2

Номер тестування	2
Назва	Рівень без обмежень
Кроки	1. Запустити програму; 2. Перейти у меню рівнів; 3. Вибрати рівень «Explore!»; 4. Натиснути «Start».
Результат	Відбувається перехід на новий рівень, у якому немає таймера або обов’язкових завдань.

Таблиця 3.3 – Варіант тестування №3

Номер тестування	3
Назва	Регулювання гучності
Кроки	1. Запустити програму; 2. У меню натиснути на кнопку «Options»; 3. Перетягнути повзунок по вподобанню.
Результат	Змінюється гучність музики згідно рівню повзунка.

Таблиця 3.4 – Варіант тестування №4

Номер тестування	4
Назва	Перегляд статистики рівня
Кроки	1. Запустити програму; 2. Перейти у меню рівнів; 3. Натиснути на пройдений рівень, статистику якого потрібно переглянути.
Результат	З'являється вікно інформації, де можна переглянути рекорд даного рівня.

Таблиця 3.5 – Варіант тестування №5

Номер тестування	5
Назва	Вільний вибір пройдених рівнів
Кроки	1. Запустити програму; 2. Почати нову гру, натиснувши на «New Game»; 3. Пройти рівень; 4. Перейти у головне меню; 5. Перейти у меню рівнів; 6. Вибрати довільний розблокований рівень; 7. Натиснути «Start».
Результат	Перехід до вже виконаного рівня є успішним.
Розширення:	
1)	Якщо наявні пройдені рівні: 1. Перейти у меню рівнів; 2. Вибрати довільний розблокований рівень; 3. Натиснути «Start».
Результат	Перехід до вже виконаного рівня є успішним.

Таблиця 3.6 – Варіант тестування №6

Номер тестування	6
Назва	Збереження даних
Кроки	1. Запустити програму; 2. Почати нову гру, натиснувши на «New Game»; 3. Пройти рівень; 4. Перейти у головне меню; 5. Перейти у меню рівнів; 6. Натиснути на щойно пройдений рівень та перевірити «Score».
Результат	Дані успішно збережено.
Розширення:	
1)	Якщо наявні пройдені рівні: 1. Перейти у меню рівнів; 2. Натиснути на довільний пройдений рівень та перевірити «Score».
Результат	Дані успішно збережено.

Таблиця 3.7 – Варіант тестування №7

Номер тестування	7
Назва	Тимчасова зупинка рівня
Кроки	1. Запустити програму; 2. Перейти у меню рівнів; 3. Натиснути на довільний рівень; 4. Натиснути «Start»; 5. Натиснути на кнопку паузи у верхньому лівому кутку.
Результат	Зупиняється таймер, а разом з ним і усі рухомі об’єкти.

Таблиця 3.8 – Варіант тестування №8

Номер тестування	8
Назва	Нова гра
Кроки	1. Запустити програму; 2. Натиснути на кнопку «New Game».
Результат	Весь ігровий прогрес успішно стерто, а користувач переноситься у меню рівнів.

Таблиця 3.9 – Варіант тестування №9

Номер тестування	9
Назва	Вихід зі гри
Кроки	1. Запустити програму; 2. Натиснути на кнопку «Quit».
Результат	Застосунок успішно зачинено.

3.3.7 Висновки розробки

Для розробки програми використовувались інструменти, які я обрав на початку роботи на розробкою проекту. Для різних функцій та оформлення застосунку були використані різноманітні активи, деякі мого виробництва, а деякі завантажені з мережі Інтернет. Для виробництва своїх активів я використовував графічний редактор Aseprite. Періодично проводилося тестування продукту. По створенню програми було проведено працю з її головними налаштуваннями. Була налагоджена орієнтація елементів гри згідно з розширенням екрану та була використана вкладка «Player Settings», яка відповідає за налаштування проекту. Тут я обрав орієнтацію екрану «Landscape Right».

За допомогою варіантів тестування я перевірив продукт на дотримання вимог, що були складені на етапі проектування. За допомогою них можна зробити висновок, що розроблений продукт «CatchUP!» відповідає усім представленим у дипломному проекті вимогам. Не було виявлено жодних критичних помилок, які могли б перешкоджати роботі програми або навіть порушувати її.

4 ВИСНОВКИ

Не можна не помітити великий вплив смартфонів на наше життя. Дані пристрої тепер можуть допомогти вам слідкувати за чим завгодно, від новин до своєї гігієни. Нині однією з найпопулярніших категорій використання смартфонів є сфера розваг. Популярність розважальних додатків на портативних смарт-пристроях пояснює їх доступність де завгодно і коли завгодно. Звідси можна зробити висновок, що розробка мобільних застосунків дуже популярна і не менш прибуткова.

Однак подібні вигоди дорого затратили наше психічне та соціальне життя. Постійний зв’язок з навколишнім світом, який пропонують смартфони, зробили їх чимось дуже схожим на наркотики для користувачів. Ми дедалі більше відвертаємо увагу, проводимо менше часу в реальному світі та все глибше занурюємося у віртуальний. Це стосується й студентів та школярів, оскільки виникає недостача часу для навчання. Для того, аби вирішити проблему нестачі часу, потрібно розробити застосунок, який привертав би увагу та покращив процес навчання.

Результатом даної дипломної роботи став розважальний застосунок для платформи «Android» у жанрі 2D-платформер. За процес навчання відповідають тестові запитання з математики. На них користувач має відповісти, якщо встигне добігти до класу у відзначений для нього час.

Перед тим, як почати працю, я розглянув та дослідив існуючі популярні аналоги мого програмного продукту. Використовуючи результати дослідження була складена таблиця для порівняння даних аналогів з моїм продуктом. Виявилося, що у продукту «CatchUP!» є перевага елементів головоломки, навчання та швидкості.

Для правильної роботи з продуктом потрібно було провести його моделювання та планування. Було складено вимоги, які значно спрощують бачення того, яким повинен бути продукт. У процесі роботи були складені:

- Користувацькі історії, які дозволяють чітко зрозуміти те, що саме хоче бачити користувач;
- Прецеденти, або варіант використання, які відповідають за те, як саме користувач має працювати з застосунком;
- UML-діаграми, для простої візуалізації того, як повинна відбуватися взаємодія між користувачем та системою. Саме тут було розібрано, як приблизно повинен виглядати інтерфейс користувача, як проходить ігровий процес, як саме підраховуються відповіді та результат тесту і куди зберігаються дані пройдених рівнів.

У дипломній роботі я розробив програмне забезпечення та з цим створив робочий програмний продукт, який легко можна використовувати в інших технологіях та переробити під інші платформи

У процесі розробки даного застосунку я засвоїв та закріпив багато теоретичних та практичних знань, що стосуються життєвого циклу ПЗ. Зробив огляд ринку мобільних ігор та на основі цього сформулював висновки з приводу того, наскільки актуальна розробка подібного програмного забезпечення, а відповідно і тема та практична значущість дипломної роботи. У результаті було створено програмний продукт, який відповідає всім наявним вимогам і не має жодних критичних помилок, які могли б перешкоджати або порушувати роботу програми. Він буде покращуватись та оновлюватись у процесі підтримки ПЗ.

ПЕРЕЛІК ПОСИЛАНЬ

1. Screen Time Statistics 2021 [Електронний ресурс] // Elite Content Marketer ─ Режим доступу до ресурсу: https://elitecontentmarketer.com/screen-time-statistics/#:~:text=A%20study%20of%2011k%20RescueTime,minutes%20on%20their%20mobile%20devices.
2. Digital 2020: Global Digital Overview [Електронний ресурс] // DataReportal – 2020 – Режим доступу до ресурсу: https://datareportal.com/reports/digital-2020-global-digital-overview.
3. Mobile Operating System Market Share Worldwide [Електронний ресурс] // StatCounter – 2020 – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/mobile/worldwide.
4. Number of apps available in leading app [Електронний ресурс] // Statista – 2020 – Режим доступу до ресурсу: https://www.statista.com/chart/12455/number-of-apps-available-in-leading-app-stores/.
5. UMLetino – Free Online UML tool [Електронний ресурс] // UMLetino – Режим доступу до ресурсу: https://www.umletino.com/umletino.html.
6. ISO 9241-210:2010 [Електронний ресурс] // ISO – Режим доступу до ресурсу: https://www.iso.org/standard/52075.html.
7. The State of Mobile in 2020 [Електронний ресурс] // App Annie – 2021 – Режим доступу до ресурсу: https://www.appannie.com/en/insights/market-data/state-of-mobile-2020/?sfdcid=7016F000002MS1c&utm_campaign=ww-logo-201910-1910-digital-2020-partnership&utm_content=report-&utm_medium=partnership&utm_source=digital-2020.
8. Download – Unity [Електронний ресурс] // Unity Technologies – Режим доступу до ресурсу: https://unity3d.com/get-unity/download.
9. Unity User Manual [Електронний ресурс] // Unity Technologies – Режим доступу до ресурсу: https://docs.unity3d.com/2017.4/Documentation/Manual/.
10. Aseprite Download [Електронний ресурс] // Igara Studio – Режим доступу до ресурсу: https://www.aseprite.org/.

11. MonoDevelop Download [Електронний ресурс] // MonoDevelop Project – Режим доступу до ресурсу: https://www.monodevelop.com/download/.
12. Aseprite – Docs [Електронний ресурс] // Igara Studio – Режим доступу до ресурсу: https://www.aseprite.org/docs/.
13. ТЕСТУВАННЯ ЯК ЕФЕКТИВНИЙ ІНСТРУМЕНТ ВИМІРЮВАННЯ РІВНЯ ЗНАНЬ СТУДЕНТІВ [Електронний ресурс] / Б. С. Гриник, О. Г. Пилипів. – 2013 – Режим доступу до ресурсу: http://irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Nzspp_2013_3_22.pdf.
14. К. Вігерс, Розробка вимог до програмного забезпечення / Карл Вігерс., 2004 – 576 с.
15. Тісний зв’язок математики та інформатики [Електронний ресурс] – Присяжнюк Т.А., 2009 – Режим доступу до ресурсу: http://eprints.zu.edu.ua/5160/1/4.pdf.

ДОДАТОК А

CharacterController2D.cs

```
using UnityEngine;
using UnityEngine.Events;

public class CharacterController2D : MonoBehaviour
{
    [SerializeField] private float m_JumpForce = 400f;
    [Range(0, 1)] [SerializeField] private float m_CrouchSpeed = .36f;
    [Range(0, .3f)] [SerializeField] private float m_MovementSmoothing = .05f;
    [SerializeField] private bool m_AirControl = false;
    [SerializeField] private LayerMask m_WhatIsGround;
    [SerializeField] private Transform m_GroundCheck;
    [SerializeField] private Transform m_CeilingCheck;
    [SerializeField] private Collider2D m_CrouchDisableCollider;

    const float k_GroundedRadius = .05f;
    private bool m_Grounded;
    const float k_CeilingRadius = .2f;
    private Rigidbody2D m_Rigidbody2D;
    private bool m_FacingRight = true;
    private Vector3 m_Velocity = Vector3.zero;

    [Header("Events")]
    [Space]

    public UnityEvent OnLandEvent;

    [System.Serializable]
    public class BoolEvent : UnityEvent<bool> { }

    public BoolEvent OnCrouchEvent;
    private bool m_wasCrouching = false;

    private void Awake()
    {
        m_Rigidbody2D = GetComponent<Rigidbody2D>();

        if (OnLandEvent == null)
            OnLandEvent = new UnityEvent();

        if (OnCrouchEvent == null)
            OnCrouchEvent = new BoolEvent();
    }

    private void FixedUpdate()
    {
        bool wasGrounded = m_Grounded;
        m_Grounded = false;

        Collider2D[] colliders = Physics2D.OverlapCircleAll(m_GroundCheck.position, k_GroundedRadius, m_WhatIsGround);
        for (int i = 0; i < colliders.Length; i++)
        {
            if (colliders[i].gameObject != gameObject)
            {
                m_Grounded = true;
                if (wasGrounded)
                    OnLandEvent.Invoke();
            }
        }
    }


    public void Move(float move, bool crouch, bool jump)
    {
        if (!crouch)
        {
            if (Physics2D.OverlapCircle(m_CeilingCheck.position, k_CeilingRadius, m_WhatIsGround))
            {
                crouch = true;
            }
        }

        if (m_Grounded || m_AirControl)
        {
            if (crouch)
            {
                if (!m_wasCrouching)
                {
                    m_wasCrouching = true;
                    OnCrouchEvent.Invoke(true);
                }

                move *= m_CrouchSpeed;

                if (m_CrouchDisableCollider != null)
                    m_CrouchDisableCollider.enabled = false;
            } else
            {
                if (m_CrouchDisableCollider != null)
                    m_CrouchDisableCollider.enabled = true;

                if (m_wasCrouching)
                {
                    m_wasCrouching = false;
                    OnCrouchEvent.Invoke(false);
                }
            }

            Vector3 targetVelocity = new Vector2(move * 10f, m_Rigidbody2D.velocity.y);
            m_Rigidbody2D.velocity = Vector3.SmoothDamp(m_Rigidbody2D.velocity, targetVelocity, ref m_Velocity, m_MovementSmoothing);

            if (move > 0 && m_FacingRight)
            {
                Flip();
            }
            else if (move < 0 && !m_FacingRight)
            {
                Flip();
            }
        }
        if (m_Grounded && jump)
        {
            m_Grounded = false;
            m_Rigidbody2D.AddForce(new Vector2(0f, m_JumpForce));
        }
    }


    private void Flip()
    {
        m_FacingRight = !m_FacingRight;

        Vector3 theScale = transform.localScale;
        theScale.x *= -1;
        transform.localScale = theScale;
    }
}
```

PlayerMovement.cs

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

public class PlayerMovement : MonoBehaviour {

    public CharacterController2D controller;
    public Animator animator;

    public float runSpeed = 40f;

    float horizontalMove = 0f;
    bool jump = false;
    bool crouch = false;

    void Update () {

        horizontalMove = SimpleInput.GetAxisRaw("Horizontal") * runSpeed;

        animator.SetFloat("Speed", Mathf.Abs(horizontalMove));

        if (Input.GetButtonDown("Jump"))
        {
            jump = true;
            animator.SetBool("IsJumping", true);
        }

        if (Input.GetButtonDown("Crouch"))
        {
            crouch = true;
        } else if (Input.GetButtonUp("Crouch"))
        {
            crouch = false;
        }

    }

    public void JumpDown ()
    {
        jump = true;
        animator.SetBool("IsJumping", true);
    }

    public void JumpUp ()
    {
        crouch = false;
        animator.SetBool("IsJumping", false);
    }

    public void CrouchDown ()
    {
        crouch = true;
    }

    public void CrouchUp ()
    {
        crouch = false;
    }

    public void OnLanding ()
    {
            animator.SetBool("IsJumping", false);
    }

    public void OnCrouching (bool isCrouching)
    {
        animator.SetBool("IsCrouching", isCrouching);
    }

    void FixedUpdate ()
    {
        controller.Move(horizontalMove * Time.fixedDeltaTime, crouch, jump);
        jump = false;
    }
}
```

Teleport.cs

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

public class Teleport : MonoBehaviour {

    [Header("Player and Portal")]
    public GameObject Portal;
    public GameObject Player;

    [Header("Indicator")]
    public GameObject ActionButton;

/// UI Button
            private void OnTriggerStay2D(Collider2D other)
    {

/// UI Button----------------------------------------------------------------
        ActionButton.SetActive(true);

        var btn = ActionButton.GetComponent<UnityEngine.UI.Button>();

        btn.onClick.RemoveAllListeners();
        btn.onClick.AddListener(() => {
            ActionButton.GetComponent<PortalDoor>().Portal = Portal;
            Debug.Log("~Shooooown~");
            StartCoroutine(DoorPortal());
        });

/// Keyboard Button----------------------------------------------------------

        if(Input.GetKeyDown("e"))
        {
            ActionButton.GetComponent<PortalDoor>().Portal = Portal;
            StartCoroutine(DoorPortal());
        }
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        ActionButton.SetActive(false);
    }

    private IEnumerator DoorPortal()
    {
        yield return new WaitForSeconds(0.5f);

        var p = ActionButton.GetComponent<PortalDoor>().Portal;
        Player.transform.position = new Vector2(p.transform.position.x, p.transform.position.y);

    }
}
```

PortalDoor.cs

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

public class PortalDoor : MonoBehaviour {

    public GameObject Portal;
}
```

QuizActivator.cs

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

public class QuizActivator : MonoBehaviour {

    [Header("Player")]
    public GameObject Player;

    [Header("Indicator")]
    public GameObject ActionButton;
    [Header("Quiz")]
    public GameObject Quiz;
    [Header("Objects to disable")]
    public GameObject Timer;
    public GameObject Control;

            private void OnTriggerStay2D(Collider2D other)
    {
/// UI Button-------------------------------------------------------------------
        ActionButton.SetActive(true);

        var btn = ActionButton.GetComponent<UnityEngine.UI.Button>();

        btn.onClick.AddListener(ActivateQuiz);

/// Keyboard Button-------------------------------------------------------------

        if(Input.GetKeyDown("e"))
        {
            Quiz.SetActive(true);
            Timer.SetActive(false);
            Control.SetActive(false);

        }
    }

    public void ActivateQuiz()
    {
        Quiz.SetActive(true);
        Timer.SetActive(false);
        Control.SetActive(false);
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        ActionButton.SetActive(false);
    }
}
```

TimerScript.cs

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

public class TimerScript : MonoBehaviour {

    Image timerBar;
    public float maxTime = 5f;
    float timeLeft;
    public GameObject timesUpText;
    public GameObject loseWindow;
    public GameObject UI;

    public void Start ()
    {
        timesUpText.SetActive(false);
        timerBar = GetComponent<Image> ();
        timeLeft = maxTime;
    }

    public void Update ()
    {
        if (timeLeft > 0)
        {
            timeLeft -= Time.deltaTime;
            timerBar.fillAmount = timeLeft / maxTime;
        }
        else
        {
            timesUpText.SetActive (true);
            UI.SetActive (false);
            StartCoroutine(LosePopup());
        }



    }

        public IEnumerator LosePopup()
    {
        yield return new WaitForSecondsRealtime(1.5f);
        loseWindow.SetActive(true);

    }
}
```

QuizScript.cs

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

public class QuizScript : MonoBehaviour {

    public QuestionList[] questions;
    public Text[] answersText;
    public Text qText;

    List<object> qList;
    QuestionList crntQ;
    int randQ;

    [Header("Button Activation")]
    public GameObject button;

    [Header("Quiz Jazz")]
    public GameObject answrs;
    public GameObject scoreText;
    public GameObject qstn;

    [Header("Win and Lose")]
    public GameObject win;
    public GameObject lose;

    [Header("GameManager")]
    public GameManager gameManager;

    public void OnClickPlay()
    {
        qList = new List<object>(questions);
        questionGenerate();

        button.SetActive(false);
        answrs.SetActive(true);
        scoreText.SetActive(true);
    }

    public void questionGenerate()
    {
        if (qList.Count > 0)
        {
            randQ = Random.Range(0, qList.Count);
            crntQ = qList[randQ] as QuestionList;
            qText.text = crntQ.question;
            List<string> answers = new List<string>(crntQ.answers);
            for (int i = 0; i < crntQ.answers.Length; ++i)
            {
                int rand = Random.Range(0, answers.Count);
                answersText[i].text = answers[rand];
                answers.RemoveAt(rand);
            }
        }

        else
        {
            print("You've completed the level!");

            answrs.SetActive(false);
            scoreText.SetActive(false);
            qstn.SetActive(false);

            if (ScoreScript.scoreValue > 2)
            {
                win.SetActive(true);
                Result();

                gameManager.WinLevel();
            }

            if (ScoreScript.scoreValue < 3)
            {
                lose.SetActive(true);
                Result();
            }

        }

    }

    public void answersBttns(int index)
    {
        if (answersText[index].text.ToString() == crntQ.answers[0])
        {
            print ("Right answer!");
            ScoreScript.scoreValue += 1;
        }
        else print ("Wrong answer...");
        qList.RemoveAt(randQ);
        questionGenerate();
    }

    public void Result ()
    {

        Scene currentScene = SceneManager.GetActiveScene();
        string sceneName = currentScene.name;

        if (sceneName == "Level1")
        {
        PlayerPrefs.SetInt ("Level1Highscore", ScoreScript.scoreValue);
        }

        if (sceneName == "Level2")
        {
        PlayerPrefs.SetInt ("Level2Highscore", ScoreScript.scoreValue);
        }

        if (sceneName == "Level3")
        {
        PlayerPrefs.SetInt ("Level3Highscore", ScoreScript.scoreValue);
        }
    }

[System.Serializable]

    public class QuestionList
    {
        public string question;
        public string[] answers = new string[4];
    }
}
```

ScoreScript.cs

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

public class ScoreScript : MonoBehaviour {

    public static int scoreValue = 0;
    Text score;

    void Start ()
    {
        score = GetComponent<Text> ();
    }

    void Update ()
    {
        score.text = "Score: " + scoreValue + "/5";
    }
}
```

GameManager.cs

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

public class GameManager : MonoBehaviour {

    public string nextLevel = "Level2";
    public int levelToUnlock = 3;

    public void WinLevel ()
    {
        PlayerPrefs.SetInt("levelReached", levelToUnlock);
    }
}
```

LevelSelector.cs

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

public class LevelSelector : MonoBehaviour {

    public Button[] levelButtons;


    void Start ()
    {
        int levelReached = PlayerPrefs.GetInt("levelReached", 1);

        for (int i = 0; i < levelButtons.Length; i++)
        {
            if (i + 1 > levelReached)
                levelButtons[i].interactable = false;
        }
    }
}
```

CameraFollow.cs

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

public class CameraFollow : MonoBehaviour {

    private Transform playerTransform;

    void Start () {
        playerTransform = GameObject.FindGameObjectWithTag("Player").transform;
    }

    void LateUpdate () {
        Vector3 temp = transform.position;

        temp.x = playerTransform.position.x;
        temp.y = playerTransform.position.y;

        transform.position = temp;
    }
}
```

```
public class MenuScript : MonoBehaviour {

    public Text highscoreLvl1;
    public Text highscoreLvl2;
    public Text highscoreLvl3;
    public GameObject options;
    public GameObject menuWindow;

    void Start ()
    {
        Scene currentScene = SceneManager.GetActiveScene();
        string sceneName = currentScene.name;

        if (sceneName == "LevelPickMenu")
        {
            highscoreLvl1.text = "Score: " + PlayerPrefs.GetInt("Level1Highscore") + "/5";
            highscoreLvl2.text = "Score: " + PlayerPrefs.GetInt("Level2Highscore") + "/5";
            highscoreLvl3.text = "Score: " + PlayerPrefs.GetInt("Level3Highscore") + "/5";
        }
    }

    public void PlayGame()
    {
        SceneManager.LoadScene("LevelPickMenu");
        ScoreScript.scoreValue = 0;
    }

    public void LoadOptions()
    {
        options.SetActive(true);
        menuWindow.SetActive(false);
    }

    public void CloseOptions()
    {
        options.SetActive(false);
        menuWindow.SetActive(true);
    }

    public void QuitGame()
    {
        Application.Quit();

        Debug.Log("Quit");
    }

    public void Restart()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        ScoreScript.scoreValue = 0;
    }

    public void NextScene()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
        ScoreScript.scoreValue = 0;
    }

    public void LoadMenu()
    {
        SceneManager.LoadScene("MainMenu");
        ScoreScript.scoreValue = 0;
    }
}
```

LevelPickScript.cs

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

public class LevelPickScript : MonoBehaviour {

    public GameObject level1;
    public GameObject level2;
    public GameObject level3;
    public GameObject freerun;
    public GameObject buttonsToDisable;

    public void Lvl1Window ()
    {
        buttonsToDisable.SetActive(false);
        level1.SetActive(true);
    }

    public void Lvl2Window ()
    {
        buttonsToDisable.SetActive(false);
        level2.SetActive(true);
    }

    public void Lvl3Window ()
    {
        buttonsToDisable.SetActive(false);
        level3.SetActive(true);
    }

    public void FreerunWindow ()
    {
        buttonsToDisable.SetActive(false);
        freerun.SetActive(true);
    }

    public void CloseAllWindows ()
    {
        level1.SetActive(false);
        level2.SetActive(false);
        level3.SetActive(false);
        freerun.SetActive(false);
        buttonsToDisable.SetActive(true);
    }

    public void LoadLevel(int level)
    {
        SceneManager.LoadScene(level);
    }

    public void NewGame()
    {
        PlayerPrefs.DeleteAll();
        PlayerPrefs.SetInt("levelReached", 1);
        PlayerPrefs.Save();
        SceneManager.LoadScene("LevelPickMenu");

    }
}
```

Pause.cs

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

public class Pause : MonoBehaviour {

    public static bool isGamePaused = false;

    [SerializeField] GameObject pauseMenu;

    void Update ()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (isGamePaused)
            {
                ResumeGame();
            }
            else
            {
                PauseGame();
            }
        }
    }

    public void PauseButton()
    {
        if (isGamePaused)
        {
            ResumeGame();
        }
        else
        {
            PauseGame();
        }
    }

    public void ResumeGame()
    {
        pauseMenu.SetActive(false);
        Time.timeScale = 1f;
        isGamePaused = false;
    }

    public void PauseGame()
    {
        pauseMenu.SetActive(true);
        Time.timeScale = 0f;
        isGamePaused = true;
    }

    public void LoadMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void QuitGame()
    {
        Application.Quit();

        Debug.Log("Quit");
    }

}
```

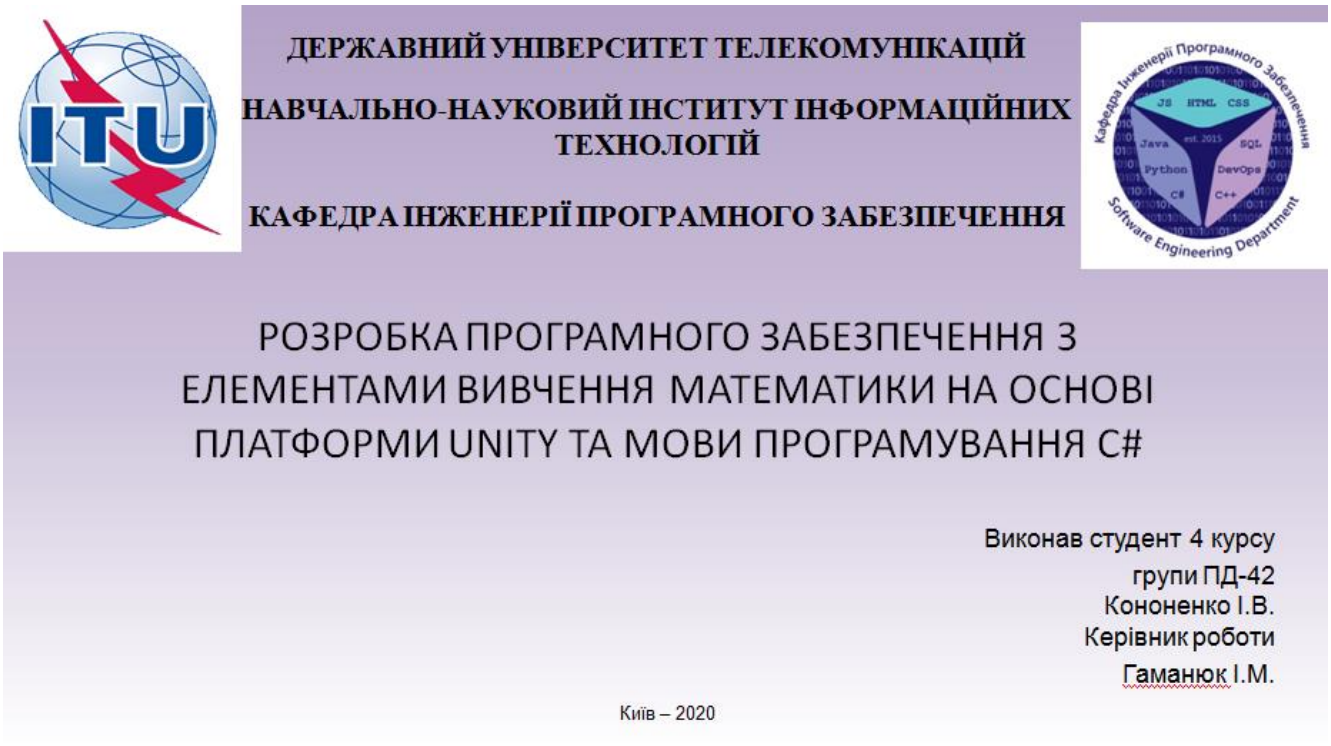
VolumeValueChange.cs

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

public class VolumeValueChange : MonoBehaviour
{
    public AudioSource music;
    public Slider volume;

    void Start ()
    {
        PlayerPrefs.GetFloat("MusicVolume");

    }

    void Update ()
    {
        music.volume = volume.value;
    }

    public void VolumePrefs()
    {
        PlayerPrefs.SetFloat("MusicVolume", music.volume);
    }
}
```

SeamlessAudio.cs

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

public class SeamlessAudio : MonoBehaviour {

    public AudioSource music;
    private static SeamlessAudio instance;

    void Start ()
    {
        music.volume = PlayerPrefs.GetFloat("MusicVolume");
    }

    void Awake ()
    {
        if (instance != null)
        {
            Destroy(gameObject);
        }
        else
        {
            instance = this;
            DontDestroyOnLoad (transform.gameObject);
        }
    }
}
```

ДОДАТОК Б



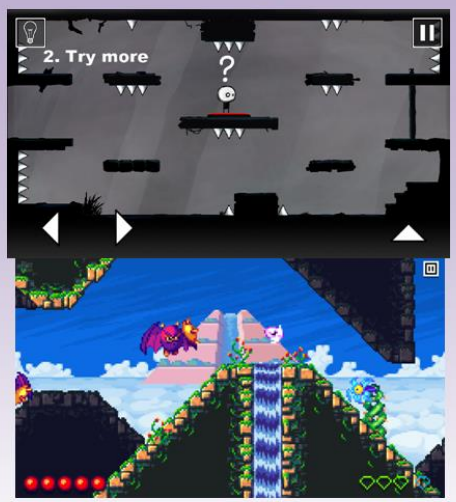
МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** — покращення процесу вивчення математики шляхом створення та застосування програмного забезпечення.
- **Об'єкт дослідження** — хід розробки ігрового застосунку для покращення вивчення математики.
- **Предмет дослідження** — програмний застосунок, що дає можливість покращити знання з математики.

2

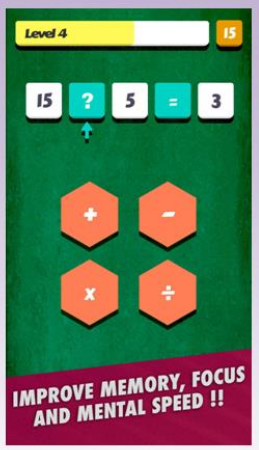
АНАЛОГИ

That Level Again



WitchEye

Math – Quiz Game



Sonic Runners

Tricky Castle

3

АНАЛОГИ

Назва	That Level Again	Tricky Castle	Math – Quiz Game	Witcheye	Sonic Runners	CatchUP!
Масштабованість	Так	Так	Так	Так	Так	Так
Онлайн-магазин	Ні	Ні	Ні	Ні	Так	Ні
Орієнтація екрану	Landscape	Landscape	Portrait	Landscape	Landscape	Landscape
Ел. швидкості	Ні	Ні	Так	Ні	Ні	Так
Одна мапа	Так	Ні	Ні	Ні	Ні	Так
Ел. головоломки	Так	Так	Так	Ні	Ні	Так
Ел. навчання	Ні	Ні	Так	Ні	Ні	Так

4

ТЕХНІЧНІ ЗАВДАННЯ

- *1. Назва програми: “CatchUP!”;*
- *2. Призначення: поліпшення процесу вивчення математики;*
- *3. Область використання: учні шкіл та студенти вищих навчальних закладів;*
- *4. Платформа: “Android”;*
- *5. Розповсюдження: магазин ігрових застосунків “Google Play Market”, на безкоштовній основі.*

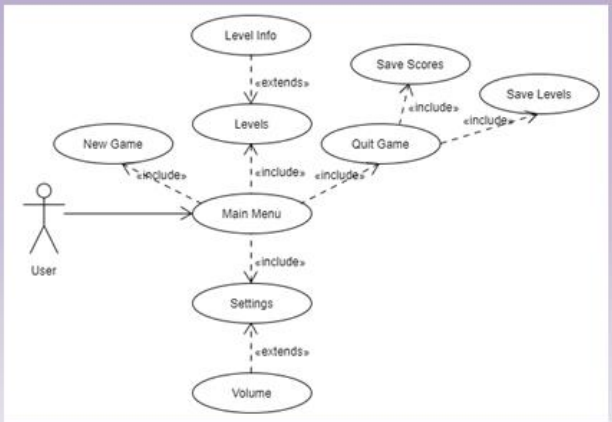
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

1. Ігровий рушій “Unity”;
2. Середовище розробки “MonoDevelop”;
3. Мова програмування C#;
4. Редактор піксельних зображень та анімації Aseprite.

6

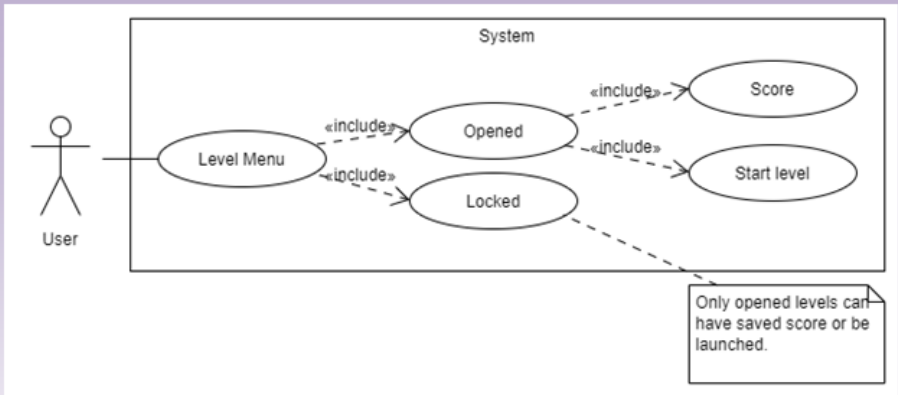
ГОЛОВНЕ МЕНЮ ГРИ



UML-діаграма прецеденту

7

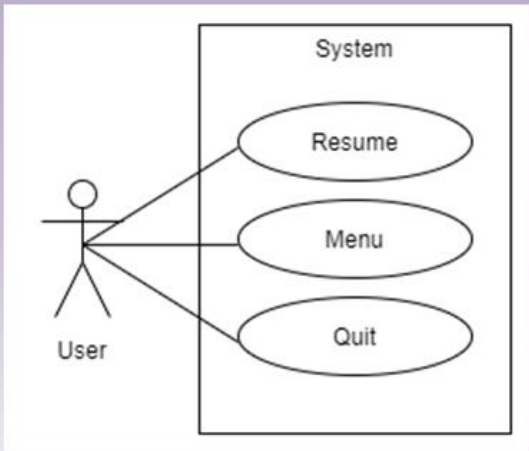
ВИБІР РІВНЯ



UML-діаграма прецеденту

8

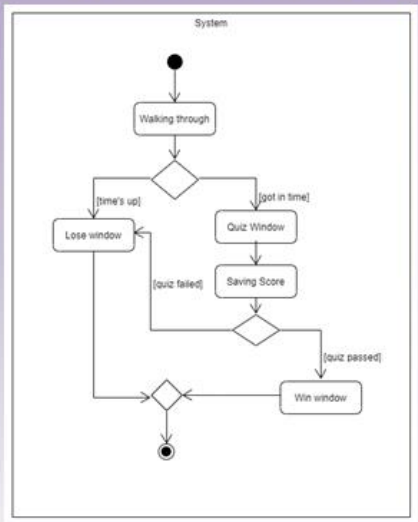
МЕНЮ ПАУЗИ



UML-діаграма прецеденту

9

ІГРОВИЙ ПРОЦЕС

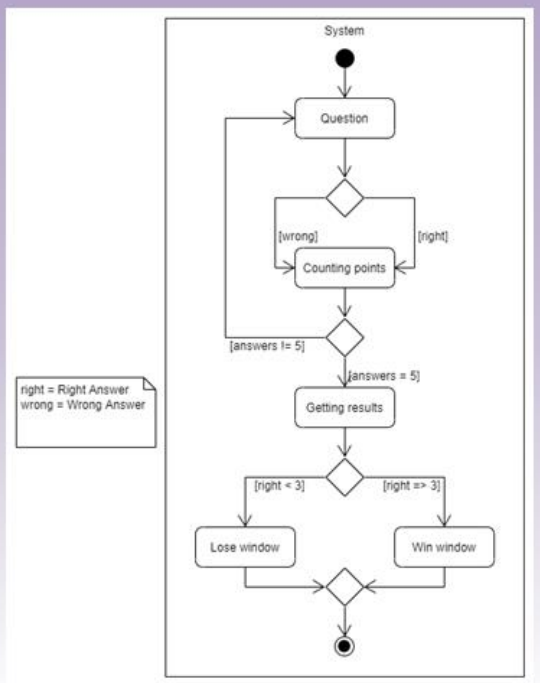


UML-діаграма діяльності

10

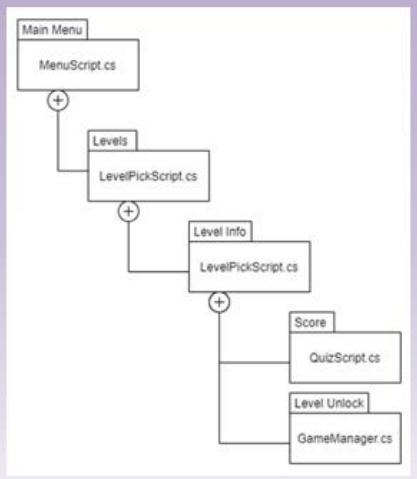
ПРОЦЕС ТЕСТУ

UML-діаграма діяльності



11

ЗБЕРЕЖЕННЯ ДАНИХ



UML-діаграма архітектури

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Кононенко І.В. РОЗРОБКА ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ З ЕЛЕМЕНТАМИ ВИВЧЕННЯ МАТЕМАТИКИ З ВИКОРИСТАННЯМ UNITY2D ДЛЯ ПЛАТФОРМИ ANDROID / Кононенко І.В. — Київ, 2021: Матеріали всеукраїнської науково-технічної конференції "Застосування програмного забезпечення в інфокомунікаціних технологіях". Збірник тез. 05.02.2021, ДУТ, м. Київ — К.: ДУТ, 2020. — С. 111.

13

ВИСНОВКИ

Я розглянув та дослідив ринок мобільних ігор та порівняв наявні аналоги зі своєю грою. Потім були розглянуті інструменти, які у подальшому використовувались у розробці, та були описані їх переваги.

Було складено вимоги, які значно спрощують бачення того, яким повинен бути продукт. У процесі роботи були складені:

- Користувацькі історії;
- Прецеденти;
- UML-діаграми, які відповідають за візуалізацію вимог.

Використовуючи описані раніше інструменти я створив програмний продукт та майже все його оформлення.

Результатом даної дипломної роботи став розважальний застосунок, який поліпшить вивчення математики.

У подальшому я планую додати більший функціонал та більше особливостей, а згодом опублікувати застосунок у "Google Play Market".

14

ДЯКУЮ ЗА УВАГУ!