**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Львівський національний університет імені Івана Франка**

**Факультет електроніки та комп'ютерних технологій**

**Кафедра радіоелектронних та комп'ютерних систем**

Допустити до захисту
Завідувач кафедри РКС
проф. Оленич І.Б.
«___» _____ 2023 р.

**КВАЛІФІКАЦІЙНА РОБОТА**

**Бакалавр**

**на тему:**

# «СТВОРЕННЯ ІНТЕРАКТИВНОГО НАВЧАЛЬНО-ІГРОВОГО ЗАСТОСУНКУ МОВОЮ C#»

Виконав:
студент IV курсу групи ФЕП-41
спеціальності 121 – Інженерія програмного забезпечення
_____ІЖАК В. П.

Науковий керівник:
Проф. ОЛЕНИЧ І.Б.

Оцінка «_____»

«_____» _____ 2023 р.
Рецензент: доц. Шувар Р.Я.

Львів – 2023

# АНОТАЦІЯ

У даній бакалаврській роботі було реалізовано навчальний тренажер у ігровому стилі, написаний мовою C# на рушії рендеру в реальному часі Unity Engine. Було пройдено повний шлях розробки продукту, реалізовано випуск гри в Play Market, робота з GitHub, шейдерами, користувацьким інтерфейсом та Admob. Велика кількість технологій забезпечила чудовий результат. Продемонстровано та пояснено усі етапи створення контенту.

# ABSTRACT SUMMARY

In this bachelor's thesis, a game-style training simulator was implemented, written in C# on the Unity Engine real-time rendering engine. We went through the entire product development process, realized the release of the game in the Play Market, worked with GitHub, shaders, user interface, and Admob. A large number of technologies provided an excellent result. All stages of content creation were demonstrated and explained.

Посилання на GitHub репозиторій проекту: https://github.com/TopYozhik/TheLastWord

Link to GitHub repository of the project: https://github.com/TopYozhik/TheLastWord

# ЗМІСТ

**АНОТАЦІЯ** ..... 1
ЗМІСТ ..... 2
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ, ТЕРМІНІВ ..... 5
ВСТУП ..... 6
РОЗДІЛ 1. ДОСЛІДЖЕННЯ СУТНОСТІ ПРОБЛЕМИ ..... 7
1.1 Огляд проблемних конкурентів ..... 7
1.2 Інші недоліки існуючих додатків ..... 9
1.3 Огляд мобільної вбудованої сумісності ..... 11
1.5 QA це ключ до успіху ..... 14
1.6 Принцип Waterfall ..... 15
1.7 Проблема десятипальцевого друку та доступності ..... 16
РОЗДІЛ 2. ТЕХНОЛОГІЇ ТА ЗАСОБИ РЕАЛІЗАЦІЇ ..... 19
2.1 Microsoft Visual Studio ..... 19
2.2 Unity Engine ..... 19
2.3 Blender 3D ..... 19
2.4 Github ..... 21
2.5 Git ..... 21
2.6 Google Admob ..... 22
2.7 Play Market ..... 22
2.8 Android Studio ..... 22
2.9 Adobe Photoshop ..... 24
2.10 RizomUV Virtual Spaces ..... 24
2.11 Adobe Substance 3D Painter ..... 25
2.12 Мови програмування ..... 25
2.13 Результати теоретичних досліджень ..... 26
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ НАВЧАЛЬНО-ІГРОВОГО ЗАСТОСУНКУ ..... 28
3.1 Створення проекту та налаштування ..... 28
3.2 Написання кодової бази ..... 29
3.3 Створення ассетів. ..... 34
3.4 Полірування коду та проєкту загалом. ..... 43
3.5 Тестування продукту ..... 46

3.6 Реліз у Play Market ..... 46
ВИСНОВКИ ..... 48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 49

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ, ТЕРМІНІВ

1. Гіт – Git, система керування версіями файлів.
2. Білд – готова до запуску збірка проекту.
3. Арт – 3д або 2д матеріали для проекту.
4. Ассети – будь-які матеріали для створення проекту.
5. Рендер – процес візуалізації контенту за доп. комп'ютерної програми.
6. RP – Render Pipeline, технологія візуалізації контенту в реальному часі.
7. C# - C-подібна мова програмування, що використовується здебільшого Microsoft, проте окрім цього багатьма студіями для розробки інтерактивного цифрового контенту.
8. Unity – популярне інтегроване середовище для розробки.
9. Сцена – область, в якій відбуваються всі операції під час запуску проекту.
10. Префаб або Prefab – шаблон, заготовка об'єкту з певним набором модулів для повторного створення однотипних об'єктів.
11. Інспектор – панель, що відображає властивості та компоненти об'єктів.
12. Дебаг або Дебагінг – процес налагодження проекту, перевірки працездатності та тестування механік.
13. IDE – інтегроване середовище розробки.
14. Референс – взірець, натхнення для створення чогось.
15. QA – Quality Assurance, перевірка якості продукту.

# ВСТУП

Популяризація української мови набуває все більшого значення, оскільки світ стає все більш взаємопов'язаним і глобалізованим. З розвитком цифрових технологій та збільшенням використання Інтернету важливо, щоб молоде покоління українців добре знало свою національну мову. Саме тому розробка навчального застосунку зі складання слів для українських дітей є актуальним і своєчасним проектом. А оскільки дітям подобається грати в ігри, то можна поєднати корисне з цікавим.

Метою цього проекту є створення навчальної гри, яка допоможе українським дітям покращити свої навички правопису та набору слів, а також загальне розуміння української мови.

Для досягнення цієї мети необхідно буде використати ряд технологій, включаючи Unity Engine, Blender, C# та інші технології розробки ігор.

# РОЗДІЛ 1. ДОСЛІДЖЕННЯ СУТНОСТІ ПРОБЛЕМИ

Розробка освітніх додатків для популяризації української мови триває вже багато років. Я особисто користувався багатьма додатками для вивчення різних мов, не тільки української. Вже існує низка додатків та ігор, спрямованих на покращення словникового запасу, граматики та навичок друкування українською мовою. Однак, незважаючи на найкращі наміри цих додатків, вони часто не забезпечують ефективного та приємного досвіду для користувача.

Однією з найбільших проблем, з якою стикаються ці додатки, є необхідність збалансувати простоту і доступність з глибиною і складністю, необхідними для забезпечення повноцінного навчального процесу. Деякі з існуючих додатків є занадто простими і їм бракує контенту і привабливості, необхідних для залучення користувача, в той час як інші є занадто складними і вимагають високого рівня технічних навичок для ефективного використання. Деякі ж, пропопонують цілком хороший контент, проте просять за нього великі гроші, часто навіть за підпискою. Враховуючи це, мало батьків готові купляти такі продукти своїм дітям, тим паче платити щомісячний платіж за підписку на подібні сервіси. А самі діти, очевидно, дозволити собі цього не можуть, вони ще надто малі й не заробляють самостійно. Крім того, багато з цих додатків не враховують потреби та інтереси дітей, що може зробити їх менш привабливими та менш ефективними. А як нам усім відомо, діти полюбляють грати у ігри.

## 1.1 Огляд проблемних конкурентів

Одним із таких додатків є Duolingo. Незважаючи на численні переваги, у Duolingo як платформи для вивчення мови є й певні обмеження. Одним з найбільших недоліків є те, що курси проходять в індивідуальному темпі, що може бути як перевагою, так і недоліком. Хоча користувачі можуть просуватися у власному темпі, вони можуть не отримати такого ж рівня персоналізованого навчання та зворотного зв'язку, як у традиційному курсі.

Ще одним обмеженням Duolingo є те, що він може не підійти тим, хто шукає більш комплексного підходу до вивчення мови. Платформа в першу чергу фокусується на словниковому запасі та граматиці, але не приділяє стільки уваги розмовній мові та вимові. Це може ускладнити користувачам розвиток їхніх навичок говоріння та аудіювання. [1]

Окрім усього вище сказаного, також Duolingo постійно старається забрати у вас ваші гроші, постійно пропонуючи придбати платну підписку задля отримання всіляких, насправді зовсім не потрібних переваг, таких як якісь незрозумілі кристали та сердечки.

Отже, Duolingo - це популярна і доступна платформа для вивчення мов, яка пропонує веселий та інтерактивний спосіб вивчення нових мов. Хоча вона може підійти не всім, це чудовий варіант для тих, хто шукає доступний і цікавий спосіб вивчити іноземну мову.

Окрім Duolingo, існують ще багато схожих навчальних застосунків-тренажерів, один з них це Memrise. Memrise - це додаток для вивчення мови, який має на меті допомогти користувачам запам'ятати нову лексику та фрази за допомогою повторення та гейміфікації. Однак, хоча це може здатися привабливим і цікавим способом вивчити нову мову, насправді він не дотягує у кількох ключових сферах.

Однією з головних проблем Memrise є відсутність комплексного викладання мови. Хоча він використовує повторення, щоб допомогти користувачам запам'ятати нові слова, він не забезпечує глибшого розуміння мови, її граматики та структури речень. Це може призвести до поверхневого розуміння мови та нездатності використовувати її в реальних ситуаціях.

Крім того, аспект гейміфікації Memrise також може бути палицею з двома кінцями. Хоча це може зробити процес навчання більш приємним, це також означає, що користувачі можуть бути не вмотивовані докладати зусиль, необхідних для справжнього вивчення нової мови. Аспект гейміфікації повинен бути правильно реалізованим, а інакше він може призвести до зосередження не на запам'ятовуванні,

а на ігровому досвіді. У моєму ж проекті, гейміфікація тісно по’вязана з ігровим процесом, що буде змушувати гравців не відволікатись від основної мети – дізнатись щось нове.

Ще одним недоліком Memrise є відсутність персоналізації. Хоча він пропонує курси різними мовами, він не пристосовує контент до окремих користувачів відповідно до їхнього стилю чи темпу навчання. Це може призвести до універсального підходу, який може не підходити для всіх. У моєму проекті в планах персоналізувати досвід гравців основуючись на їхніх потребах та виборах.

Нарешті, цінова модель Memrise також є потенційним недоліком. Хоча додаток є безкоштовним у використанні, доступ до більш розширених функцій і контенту вимагає платної підписки. Це може швидко скластися і зробити додаток дорожчим, ніж інші варіанти вивчення мови.

На закінчення, хоча Memrise може здатися веселим і легким способом вивчити нову мову, в реальності він обмежений у своїй сфері застосування та ефективності. Він може підійти деяким користувачам, які шукають швидкий спосіб запам'ятати нову лексику, але тим, хто серйозно ставиться до вивчення нової мови, краще пошукати деінде більш комплексний і персоналізований підхід, плюс його дизайн заставляє бажати кращого.

### 1.2 Інші недоліки існуючих додатків

Ще однією проблемою, з якою стикаються ці додатки, є їхня несумісність із сучасними мобільними пристроями. З появою смартфонів і планшетів стає все більш важливим, щоб освітні додатки були розроблені з урахуванням потреб мобільних користувачів. Однак багато існуючих додатків для популяризації української мови не оптимізовані для мобільних пристроїв і не враховують унікальні вимоги та обмеження цих платформ, і можуть нормально працювати лише на ПК. Хоча додатки для вивчення слів і стають дедалі популярнішими в останні роки, багато з них доступні лише для персональних комп'ютерів, а не для мобільних пристроїв. Це може

бути великим недоліком для людей, які вважають за краще вчитися на ходу або не мають доступу до персонального комп'ютера.

На додаток до проблем доступності, відсутність мобільної підтримки також може обмежувати ефективність додатків для вивчення слів. Мобільні пристрої стають дедалі популярнішими і часто використовуються як основний пристрій для багатьох людей. Не пропонуючи мобільну підтримку, додатки для вивчення слів втрачають можливість вийти на великий і зростаючий ринок користувачів, які віддають перевагу навчанню на своїх мобільних пристроях.

Незважаючи на ці обмеження, деякі додатки для вивчення слів починають пропонувати мобільну підтримку, і очікується, що ця тенденція продовжиться в майбутньому. Завдяки мобільній підтримці ці програми можуть охопити ширшу аудиторію та забезпечити більш гнучкий і доступніший навчальний процес для користувачів. Проте мобільний інтерфейс сильно перегружений(як на Рис. 1.1)

Рис. 1.1 - Кілька Референсів мобільних застосунків з поганим інтерфейсом[2]

Щоб вирішити ці проблеми та запропонувати більш ефективне та цікаве рішення для популяризації української мови серед дітей, цей проект має на меті розробити мобільний додаток, спеціально розроблений для смартфонів та планшетів.

## 1.3 Огляд мобільної вбудованої сумісності

Гра буде побудована з використанням Unity Engine та Blender - двох найпопулярніших та найпотужніших технологій розробки ігор, доступних на сьогоднішній день. Це дозволить створити високоякісну графіку, анімацію та інтерактивність, що зробить гру візуально привабливою та цікавою для дітей. Візуальне середовище Unity дуже зрозуміле та функціональне, що дозволяє швидко створювати продукти високої якості. Нижче зображено саме інтегроване робоче середовище, включаючи такі панелі та вкладки як «Інспектор», «Ієрархія», «Сцена», «Гра», «Проект» та «Консоль». Всі вони будуть використовуватися під час розробки проекту, а побачити їх вигляд можна на рисунку 1.2

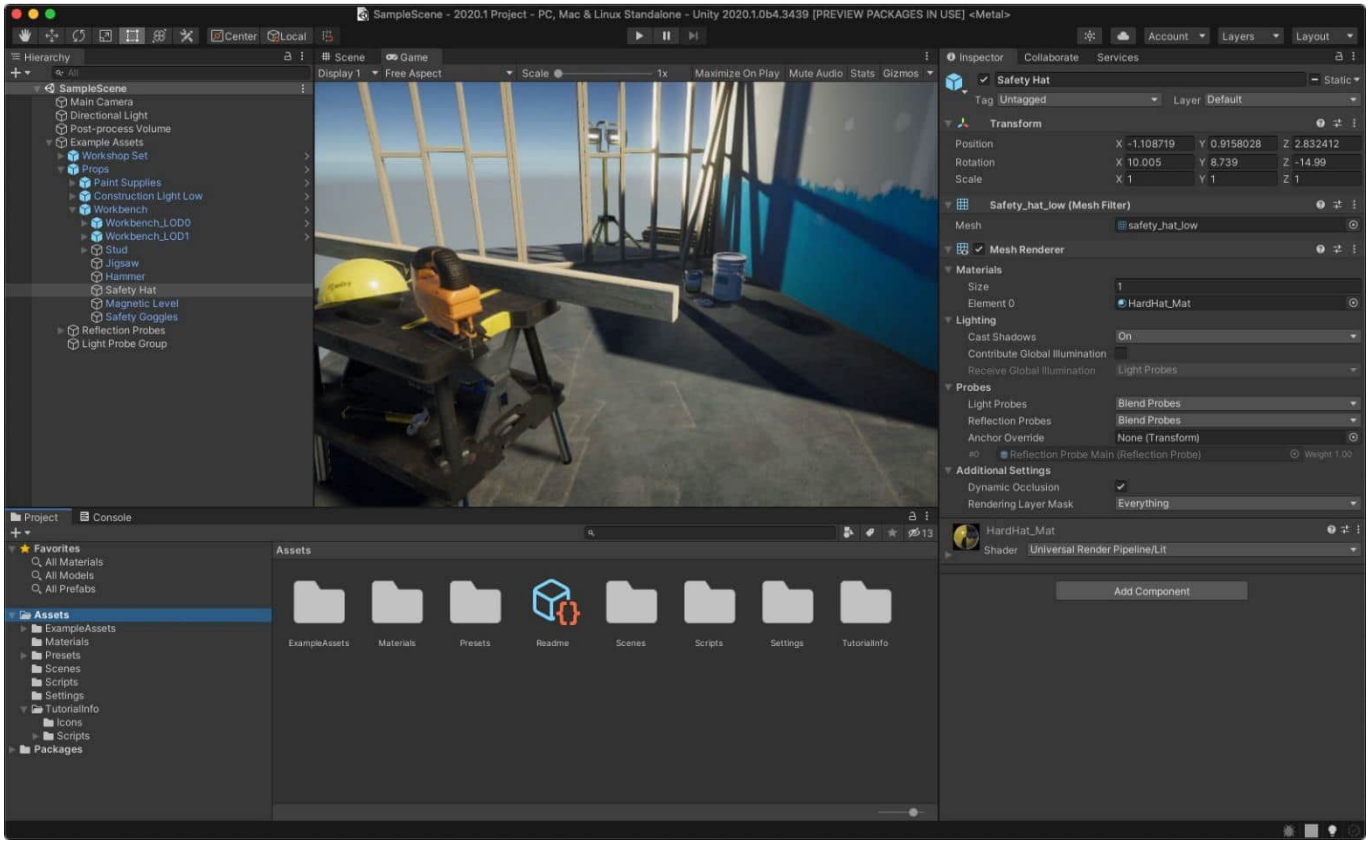


Рис. 1.2 - Робоче середовище Unity на MacOS [3]

Крім орієнтації на мобільну сумісність, цей проект також буде спрямований на подолання обмежень існуючих додатків, надаючи гру, пристосовану до потреб та інтересів дітей. Гра буде розроблена таким чином, щоб бути високоінтерактивною та захоплюючою, з низкою завдань і нагород, щоб підтримувати мотивацію та зацікавленість дітей. Гра також має бути доступною та простою у використанні, що робить її ідеальним інструментом для покращення навичок української мови серед дітей. Маленькі діти люблять грати в слова, адже це дозволяє їм простіше пізнавати світ та весело проводити час, це загальновідомий факт.

Враховуючи потреби, проект буде написаний на C#, потужній та широко використовуваній мові програмування, яку легко вивчити та використовувати. Це дозволить створити надійний та масштабований додаток, який можна буде легко підтримувати та оновлювати з часом.

Однією з найбільших переваг використання C# є її універсальність. C# - дуже гнучка мова, яку можна використовувати для широкого спектру проектів, що робить її популярним вибором серед розробників. Незалежно від того, чи створюєте ви простий десктопний додаток, складний ігровий рушій або надійну веб-платформу, C# має інструменти та функціональність, які вам потрібні для виконання роботи.

Ще однією перевагою C# є її висока продуктивність. C# - це мова зі статичною типізацією, що означає, що тип даних, які ви використовуєте у своєму коді, перевіряється під час компіляції. Це призводить до швидшого та ефективнішого виконання коду порівняно з динамічно типізованими мовами, такими як Python або JavaScript. Крім того, C# має вбудовану підтримку паралельного програмування, що робить її чудовим вибором для розробки додатків, які повинні виконувати складні операції в режимі реального часу.

Що стосується недоліків, то одним з потенційних обмежень C# є те, що її може бути не так легко вивчити, як деякі інші мови програмування. Хоча C# є відносно простою для досвідчених розробників, вона може бути складною для початківців або тих, хто не знайомий з об'єктно-орієнтованими концепціями програмування. Крім того, C# є мовою, що належить Microsoft, а це означає, що вона тісно пов'язана з

фреймворком .NET і може бути менш портативною, ніж інші мови, такі як Java або Python.

Ще одним потенційним недоліком C# є те, що вона не так широко використовується, як деякі інші мови програмування, такі як Java або Python. Це може ускладнити пошук допомоги та підтримки для мого проекту. Крім того, C# може бути не найкращим вибором для проектів, які вимагають високого ступеня сумісності з іншими платформами та технологіями, наприклад, побудованими на програмному забезпеченні з відкритим вихідним кодом.

Тому, розробка гри зі складання слів для українських дітей є унікальним та інноваційним підходом до популяризації української мови. Цей проект є надзвичайно важливим та актуальним, оскільки він спрямований на сприяння зростанню нового покоління україномовних людей та забезпечення того, щоб мова залишалася сильною та живою. Зі зростанням ролі технологій у нашому повсякденному житті важливо забезпечити дітей цікавими та інтерактивними способами вивчення та практики мовних навичок.

Гра буде розроблена з думкою про дітей, враховуючи їхні потреби, інтереси та стилі навчання. Це зробить процес навчання приємнішим і збільшить шанси на успіх, оскільки діти будуть більш зацікавлені у вивченні матеріалу та запам'ятовуватимуть вивчене. Використання принципів дизайну, орієнтованого на дитину, гарантує, що гра буде легкою у використанні та розумінні, а також забезпечить позитивне та сприятливе середовище для розвитку мовних навичок дітей.

Окрім інноваційного та цікавого дизайну гри, цей проект також буде спрямований на усунення деяких обмежень існуючих додатків для вивчення мови. Наприклад, багато таких додатків доступні лише для ПК, що обмежує їхню доступність та зручність для дітей, які мають доступ лише до мобільних пристроїв. Цей проект буде спрямований на подолання цього обмеження шляхом надання мобільного рішення, яке буде доступним для дітей, де б вони не знаходилися.

Насамкінець зазначу, що глибоке розуміння української мови має вирішальне значення для збереження культурної ідентичності та спадщини, і цей проект відіграватиме важливу роль у збереженні та популяризації української мови для наступних поколінь, які будуть користуватись мобільними пристроями. З часом їх стає все більше, графік можна побачити на рисунку 1.3

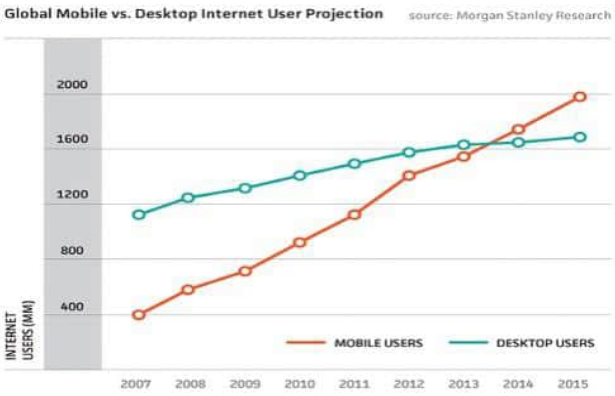


Рис. 1.3 - Співвідношення користувачів Mobile та PC за 2007-2015 роки [4]

Проаналізувавши проблемну тему, можна зробити висновок, що для хорошого та якісного продукту, необхідно забезпечити якісний QA.

## 1.5 QA це ключ до успіху

Під час проведення QA надзвичайно важливим є те, щоб зрозуміти, як працює мозок гравця, куди він буде тиснути, як він буде тиснути, чому він буде робити саме такі дії. Дуже рідко вдається зробити так, щоб все вийшло вдало з першого разу, тому велика вірогідність того, що доведеться переробляти певні елементи, такі як, UI, чи певні ігрові механіки. Задля кращого розуміння буде хорошою ідеєю покликати на тестування інших людей – знайомих, друзів, тощо. Це все дозволить якнайкраще зрозуміти потреби гравця й виправити все до релізної версії застосунку. Головне – це притримуватись певних правил Quality Assurance і Product Testing’у і все вдасться на відмінно. На рисунку 1.4 зображено приклад блок-схеми граничного контролю якості. Граничний контроль означає, що тестування задовільняє мінімальні необхідні вхідні дані, себто знаходиться на межі еквівалентної області.

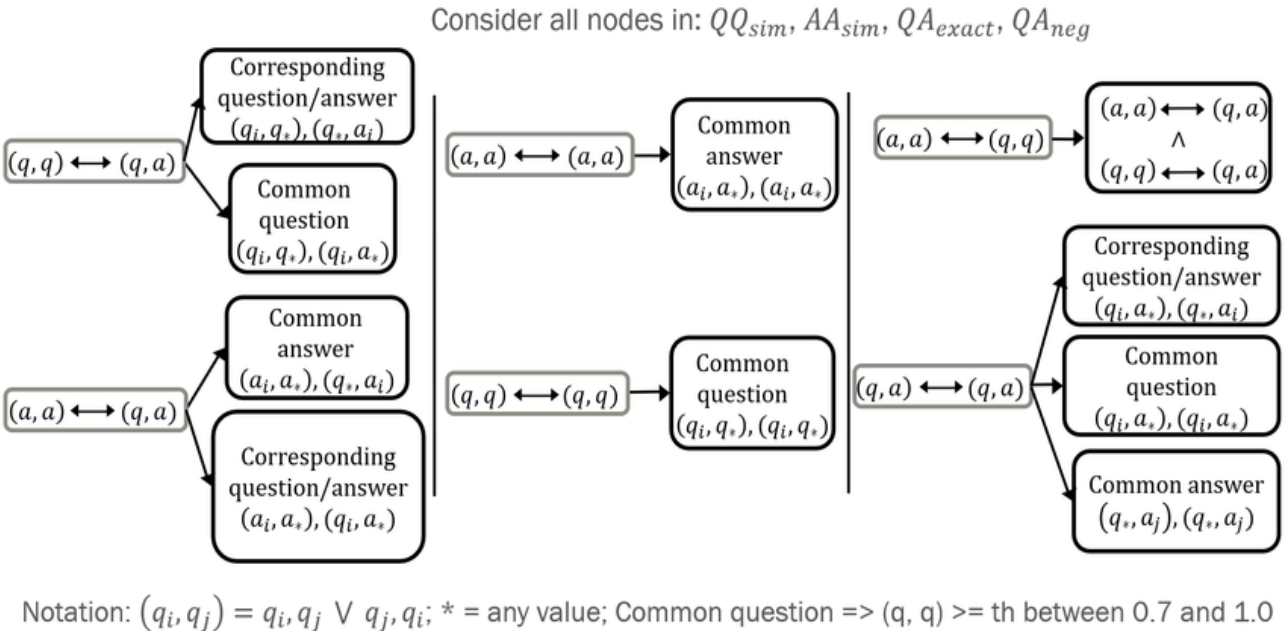


Рис. 1.4 - Блок-Схема граничного QA [5]

Процес оцінки та тестування включатиме поєднання ручного та автоматизованого тестування, включаючи юзабіліті-тестування, функціональне тестування та тестування продуктивності. Юзабіліті-тестування проводитиметься для того, щоб переконатися, що гра проста у використанні та розумінні, а функціональне тестування - для того, щоб переконатися, що гра функціонує за призначенням. Тестування продуктивності проводитиметься для того, щоб переконатися, що гра працює безперебійно та ефективно на різних пристроях.

## 1.6 Принцип Waterfall

Скоріш за все, проект буде слідувати принципу водопаду. Водоспадна модель (англ. waterfall model) – послідовний метод розробки програмного забезпечення, названий так через діаграму схожу на водоспад (як на рисунку 1.5).

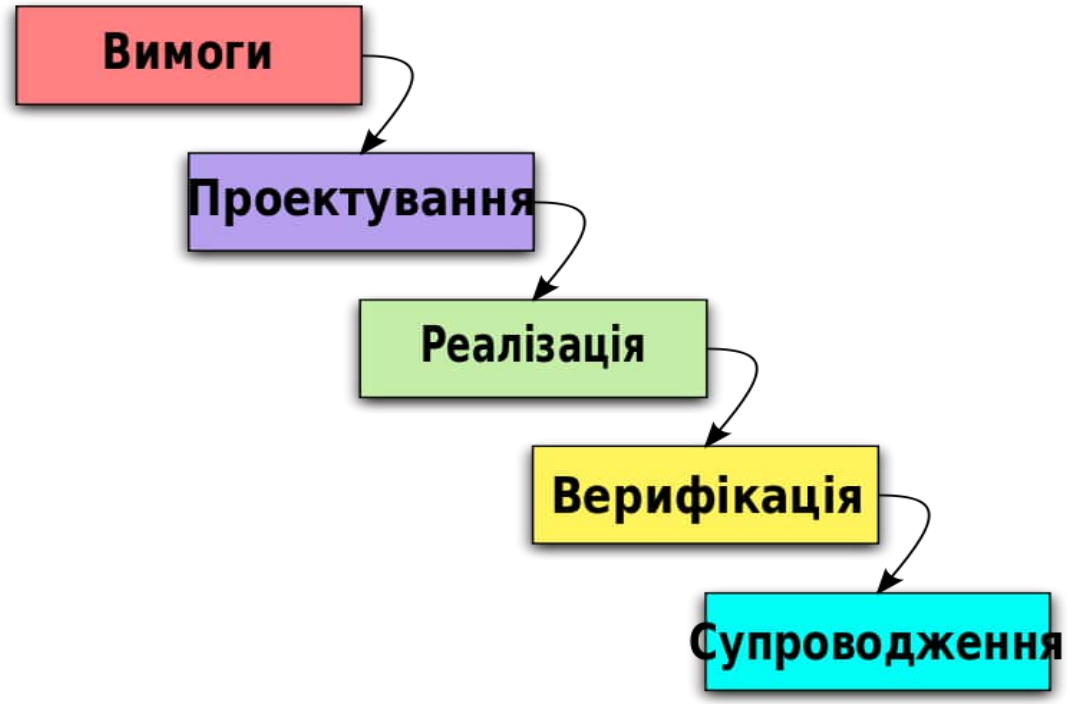


Рис. 1.5 - Блок-схема Waterfall [6]

### 1.7 Проблема десятипальцевого друку та доступності

Також, хотілося б відзначити, що оскільки здебільшого схожі програми-тренажери зустрічаються на персональні комп’ютери, то найбільш популярним способом який вони можуть запропонувати користувачу є сліпий десятипальцевий метод набору тексту.

Десятипальцевий сліпий метод друку є одним з найефективніших та швидких способів набору тексту на клавіатурі. Його суть полягає в тому, що користувач вміє друкувати, не дивлячись на клавіатуру.

У цьому методі друку руки розташовані на клавіатурі таким чином, щоб кожен палець відповідав певній групі клавіш. Зазвичай використовуються основні пальці рук: великий, вказівний, середній, безіменний та мізинець. Кожен палець має свою відповідальність за певний ряд клавіш. Ілюстрація цього наведена на рис. 1.6.

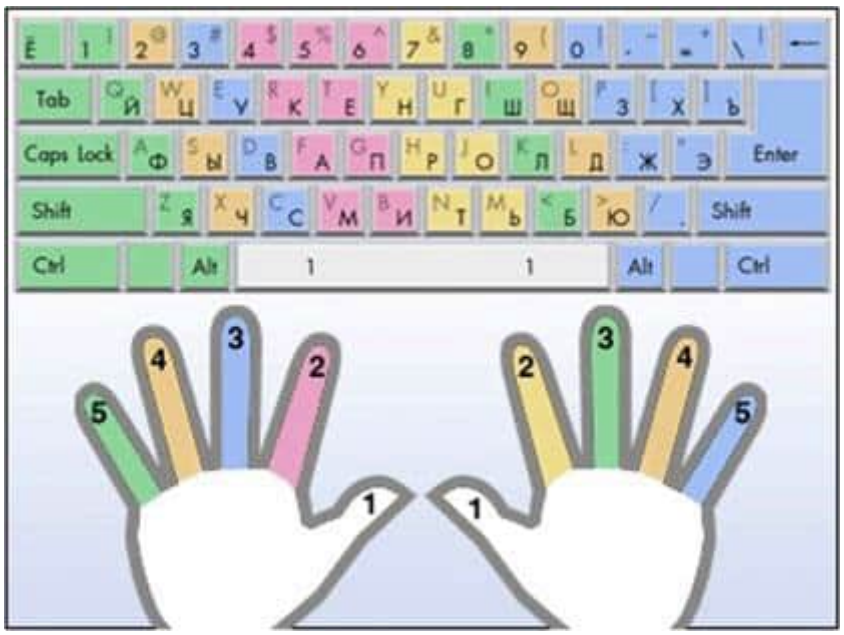


Рис. 1.6 - Схема десятипальцевого методу набору тексту [7]

Операційна система "QWERTY" використовується для організації клавіш у більшості клавіатур. Знання розташування цих клавіш дозволяє користувачу швидко й точно набирати текст.

Навички десятипальцевого сліпого друку можуть бути набуті через тренування і практику. Для цього існують спеціальні навчальні програми та ігри, які допомагають покращити швидкість і точність набору тексту.

Основна перевага десятипальцевого сліпого методу друку полягає в тому, що користувач може фокусуватись на змісті тексту, не втрачаючи часу на перегляд клавіатури. Це дозволяє працювати швидше та більш продуктивно.

Завдяки навичкам десятипальцевого сліпого друку люди можуть швидше і зручніше виконувати завдання, пов'язані з набором тексту, такі як робота з документами, написання повідомлень або набір коду. Проте далеко не у всіх випадках життя цей метод набору тексту може допомогти. Дуже часто ми знаходимося далеко від наших персональних комп'ютерів та ноутбуків, і під рукою в нас знаходиться лише телефон або планшет. Саме тут починаються проблеми.

Програмного забезпечення, яке створено для тренування набору тексту на телефонах дуже мало. Окрім дітей, яким це дуже корисно опанувати, існують професії, де одноручний набір тексту є важливою навичкою, наприклад, журналісти, письменники, редактори та інші, які вимагають продуктивної роботи на мобільних пристроях. Відсутність відповідного програмного забезпечення обмежує їх можливості тренування та поліпшення навичок набору тексту однією рукою.

Та навіть якщо таке програмне забезпечення чи ігри існують, часто в них використовується об’єктивно погана розкладка клавіатури, яка не завжди дозволяє зручно друкувати текст однією рукою. Приклад такої розкладки бачимо на рис. 1.7.

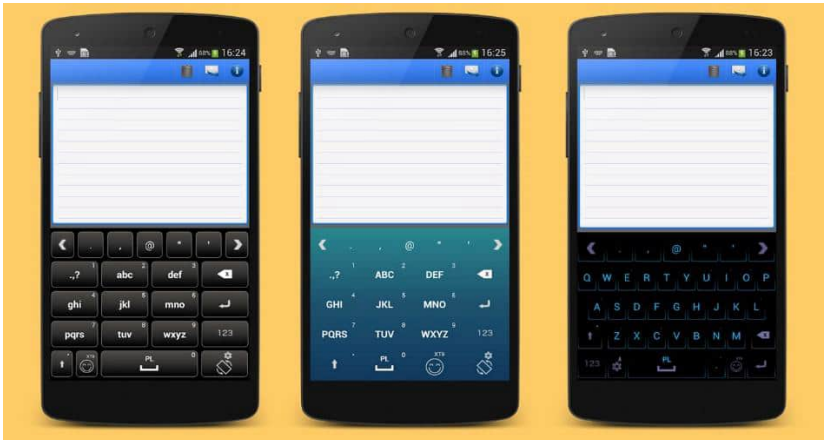

Рис. 1.7 - Приклад поганих дизайнів мобільної клавіатури [8]

Як висновок можна сказати, що проаналізувавши обмеження та недоліки існуючих додатків для популяризації української мови, та завдяки використанню сучасних технологій, таких як Unity Engine, Blender та C#, проєкт сприятиме тому, що наступне покоління українців матиме глибоке розуміння своєї національної мови, а також навички та знання, необхідні для її збереження та популяризації на довгі роки.

# РОЗДІЛ 2. ТЕХНОЛОГІЇ ТА ЗАСОБИ РЕАЛІЗАЦІЇ

## 2.1 Microsoft Visual Studio

Найбільш повне інтегроване середовище розробки для розробників на Windows. Повноцінний набір інструментів і функцій для поліпшення і вдосконалення кожного етапу розробки програмного забезпечення.

Відверто кажучи я обрав саме це IDE тільки через те, що у ньому є найзручніші плагіни, або розширення, які тісно інтегруються з робою в Unity, та дозволяють швидко автодоповнювати розпоширені методи щоб по кілька разів не писати одне й те ж. Окрім цього, у ньому є дуже широкий вибір тем інтерфейсу, що дозволяють відпочити очам довгими вечорами. У всьому ж решта, це просто повноцінне середовище розробки для будь яких цілей.

## 2.2 Unity Engine

Багатоплатформовий інструмент для розроблення цифрового контенту в реальному часі, тобто ігор і застосунків, і рушій, на якому вони працюють. Створені за допомогою Unity програми працюють на настільних комп'ютерних системах, мобільних пристроях та гральних консолях у дво- та тривимірній графіці, та на пристроях віртуальної чи доповненої реальності, тобто в будь яких **вбудованих системах**. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL.

## 2.3 Blender 3D

Моє улюблене програмне забезпечення, яким я заробляю на хліб кожен день. Блендер – це ультимативне програмне забезпечення, яке дозволяє створювати будь-які 3Д-моделі усіх рівнів складності, стилістики, якості та анімувати їх, рендерити відео, робити UV-розгортку, запікати текстури, та навіть текстурувати вище згадані мною моделі. Тут я зробив Арт-дизайн для проекту, та витратив більш ніж 500+ годин на те, щоб опанувати весь функціонал.

Робота з тривимірними моделями відбувається у сцені, розкресленій координатною сіткою. Об'єкти сцени об'єднуються в так звані колекції. За

промовчуванням кожна сцена має одну колекцію, але користувачі вільні створювати нові колекції та переміщувати між ними об'єкти для згрупування своєї роботи. Схема сцени відображається за промовчуванням праворуч вгорі. До сцени може додаватися фон або площина із зображенням-зразком для моделювання. Об'єкти можуть бути неактивними (з ними не відбувається взаємодії), активними (відбувається непряма взаємодія) та вибраними (користувач взаємодіє конкретно з цим об'єктом). Всі вони мають координати походження, що враховуються при переміщенні та деформації, та виходять з поворотної точки, що може перебувати за межами самого об'єкта. Blender містить інструменти для моделювання методом скульптингу, що симулює ліплення з глини [9]. Приклад мого інтерфейсу наведено на рис. 2.1

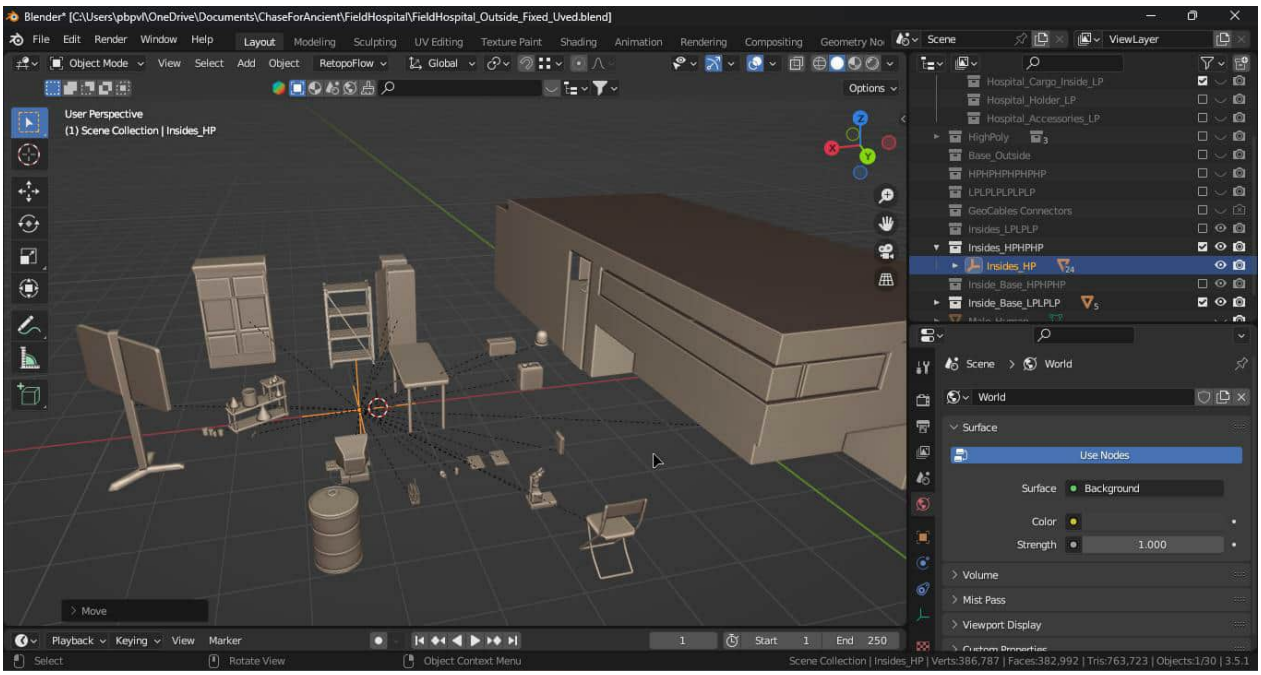


Рис. 2.1 - Мій інтерфейс Blender

Також варто зазначити, що у Blender є потужна система шейдерів, якою я скористався для створення раннього прототипу моделей. Ілюстрацію інтерфейсу шейдерів наведено на рис. 2.2.

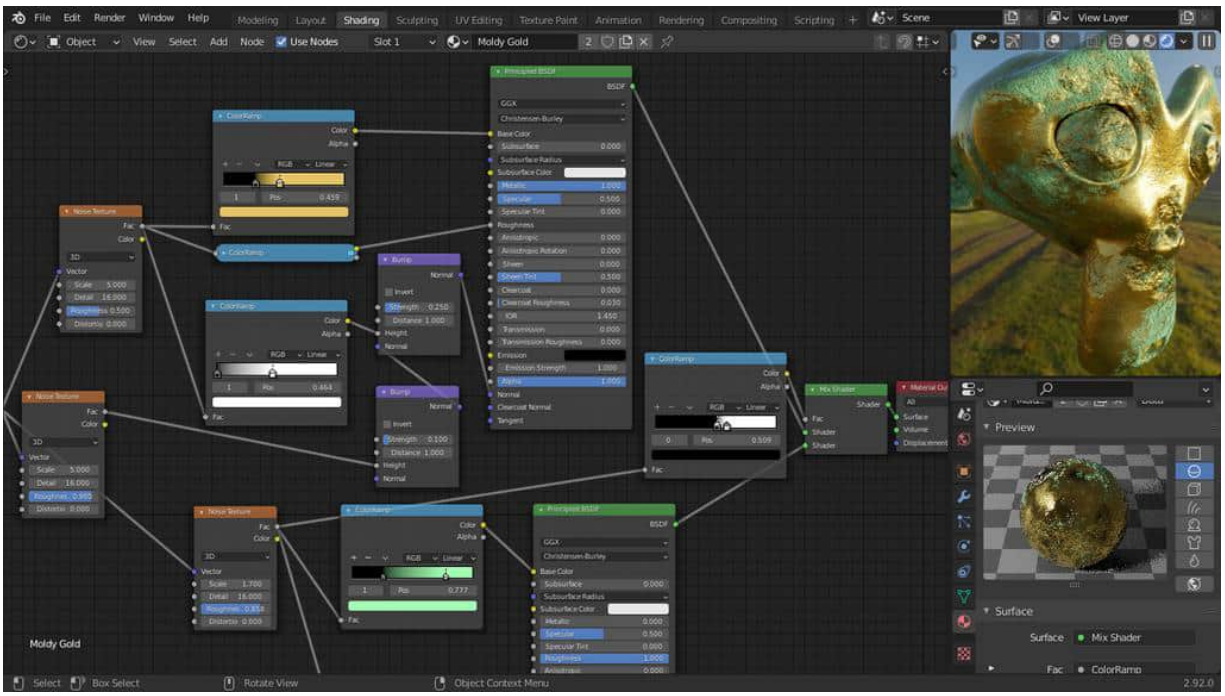


Рис. 2.2 - Шейдерний інтерфейс у Blender [10]

## 2.4 Github

Один з найбільших веб-сервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome).

Надзвичайно зручне ПЗ, яке дозволяє в будь який момент «відкатити» проект до минулої версії, якщо щось пішло не так, або завантажити його на любий пристрій, на якому користувач хоче працювати. Звісно, для розробки наодинці важко знайти ще якесь призначення йому, проте при роботі у команді цей інструмент знає просто незамінним, особливо якщо розробники знаходяться не в одній кімнаті.

## 2.5 Git

Те, заради чого був створений гітхаб. Це розподілена система керування версіями файлів та спільної роботи. Проєкт створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано (англ. Junio C. Hamano). Git є

однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

## 2.6 Google Admob

Це продукт компанії Google, що допомагає розробникам заробляти гроші на своїх додатках. AdMob добирає оголошення для додатка на основі критеріїв, які вибирає розробник. Оголошення створюють і оплачують рекламодавці, які мають на меті просування своєї продукції. Оскільки вартість різних оголошень для рекламодавців різниться, ваш прибуток варіюватиметься.

## 2.7 Play Market

Крамниця застосунків від Google, що дозволяє власникам пристроїв з мобільною операційною системою Android та іншими завантажувати і купувати різні застосунки, книги, фільми і музику. Рахунок розробника, який дає можливість публікувати програми, коштує 25 $. Оскільки я вже не вперше розробряю проект та роблю його реліз у Google Play(синонім до Play Market), то акаунт розробника в мене вже є.

## 2.8 Android Studio

Інтегроване середовище розробки (IDE) для платформи Android. Використовувалось мною лише для створення keystore файлу.

Android Studio прийшло на зміну плагіну ADT для платформи Eclipse. Середовище побудоване на базі вихідного коду продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains. Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією Apache 2.0. Бінарні складання підготовлені для Linux (для тестування використаний Ubuntu), macOS і Windows. Середовище надає засоби для розробки застосунків не тільки для смартфонів і планшетів, але і для носимих пристроїв на базі Wear OS, телевізорів (Android TV),

окулярів Google Glass і автомобільних інформаційно-розважальних систем (Android Auto). Для застосунків, спочатку розроблених з використанням Eclipse і ADT Plugin, підготовлений інструмент для автоматичного імпорту існуючого проекту в Android Studio. Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо).

Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції. Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу застосунку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub. До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови.[11]

Це ПЗ є необхідним для того, щоб вдало зкомпілювати білд під мобільні пристрої, правильно його протестувати, після чого створити ключ доступу до білду

та зарелізити проект у крамницю Гугл. Без ключа доступу розробника крамниця просто-напросто буде відмовлятись приймати будь-які білди на схвалення.

## 2.9 Adobe Photoshop

Графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в галузі комерційних засобів редагування растрових зображень і найвідомішим продуктом фірми Adobe.

## 2.10 RizomUV Virtual Spaces

Після створення низькополігональної сітки моделі потрібно обов’язково зробити її розгортку. Розгортка – це процес, схожий чимось на оригамі. Вам необхідно рівномірно розкласти об’ємну модель на плоский квадрат, роблячи надрізи у правильному місці. Чимось нагадує конструктори з картону з нашого дитинства. Саме тут з’являється програма, значно зручніша ніж вище згаданий Блендер, і спеціалізована для цього, та ім’я їй – Rizom. Це повнофункціональний додаток UV-розгортання, з функціями, що перевершують більшість пропозицій, які зараз є на ринку, призначені для індустрії комп’ютерної графіки, від інді-кадрів до ігор, в які ви грали, або фільмів, які ви бачили. Сила RizomUV полягає в тому, що він розрахований на швидкість робочого процесу. Графічний об’єкт може бути розгорнутий за лічені секунди, упакований у секунди, оптимізований за лічені секунди, тим самим скорочуючи години, які ви витрачаєте на UV-розгортання в інших додатках, до хвилин [12] .Для кращого розуміння простоти використання, приклад інтерфейсу наведено на рис 2.3

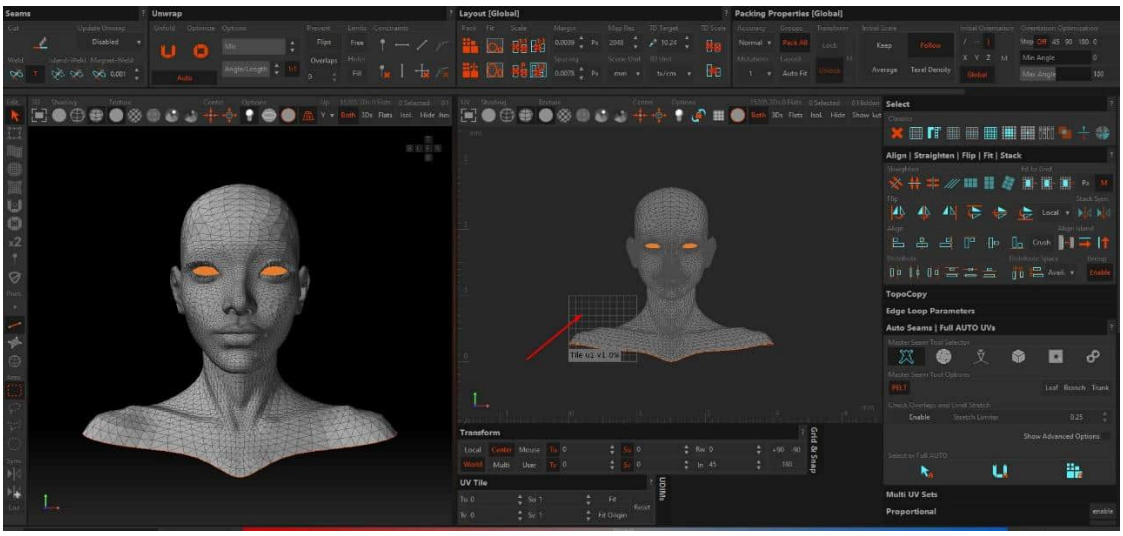


Рис. 2.3 - Приклад простоти інтерфейсу RizomUV [13]

## 2.11 Adobe Substance 3D Painter

Це програмне забезпечення – це, без перебільшення, свого часу революція та магія у світі створення 3Д контенту. Програма поєднює у собі найпотужніший функціонал для текстурування 3Д моделей, та чудовий софт для рендеру, якщо це необхідно. Вона дозволяє у реальному часі, просто і легко малювати по поверхні ассетів, так, як би ви це робили у реальному житті, комбінуючи різні шари та матеріали. Також, можна використовувати генератори матеріалів, які гнучно підлаштовуються під потреби розробника. Саме це я і робив під час розробки проекту.

## 2.12 Мови програмування

C# (вимовляється як Сі-Шарп) - Об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft). Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників — мов C++, Object Pascal, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі

моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів. Станом на 2021 рік поточна стабільна версія мови C# 10.0, яка була випущена в 2021 році як частина платформи .NET 6.0. [14]

Вона була обрана через те, що Unity Engine найкраще підтримує саме цю мову програмування, і працює з нею на природньому рівні.

GLSL (OpenGL Shading Language)

Мова високого рівня для програмування шейдерів. Синтаксис мови базується на мові програмування ANSI C, однак, через його специфічну спрямованість, з нього були вилучені багато можливостей, для спрощення мови та підвищення продуктивності. У мову долучені додаткові функції і типи даних, наприклад для роботи з векторами і матрицями. [15]

### 2.13 Результати теоретичних досліджень

Проаналізувавши мету роботи, я прийшов до висновку, що потрібно буде зробити проект, який при бажанні можна буде інтегрувати хоч у мікрохвильовку, якщо на тій буде клавіатура та процесор. Саме тому іншого та кращого вибору, аніж Unity не знайшлось. Після цього мови програмування обрались автоматично під це ПЗ. Далі зрозумівши, що гра спрямована на користувачів малого віку, приємний візуальний стиль був необхідністю. Так як я спеціалізуюсь у цій сфері, я обрав те програмне забезпечення, яке знаю найкраще: Blender, Substance Painter, RizomUV та Marvelous Designer. В процесі розробки звісно ж, я зрозумів, що через високу якість ассетів застосунок вже не можна буде запустити на умовній «мікрохвильовці», тому від високої планки якості візуального оформлення довелось відмовитись. Проте все ще був обраний приємний, «мультяшний» та дитячий стиль, на який буде приємно дивитись. Замість багатьох 3Д-об’єктів я вирішив перенести деякі у 2Д-вимір задля оптимізації застосунку, використовуючи Photoshop, а моделі та інші ассети, що залишились у звичному 3Д прийшлось добряче так «урізати» та оптимізувати під вбудовані системи, включаючи мобільні пристрої.

На цьому етапі ціль вже була зрозумілою, всі технології та методи обрані, а темпи розробки продукту почали набирати темп. Незрозумілим залишилось тільки те, чи встигну я самостійно імплементувати всі бажані мною функції та механіки, щоб вирізнятися серед проектів, описаних у розділі аналізу проблеми.

# РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ НАВЧАЛЬНО-ІГРОВОГО ЗАСТОСУНКУ

## 3.1 Створення проекту та налаштування

Підготуємо усі інструменти та програми, завантаживши їх з офіційних джерел.

Після цього будь якому поважаючому себе фулл-стак розробнику необхідно набрати референсів. Недовго думаючи, я поліз у крамницю Google та почав листати конкурентів. Зайшов у Pinterest та пошукав приклади дизайну користувацького інтерфейсу. Коли все було готове, я приступив до наступного кроку.

Зайдемо у Юніті, та через Unity Hub створимо новий проект. Як темплейт(заготовку) я обрав чистий проект, де не було нічого окрім пустого 3д простору. Ілюстрація створення проекту зображена на рис. 3.1

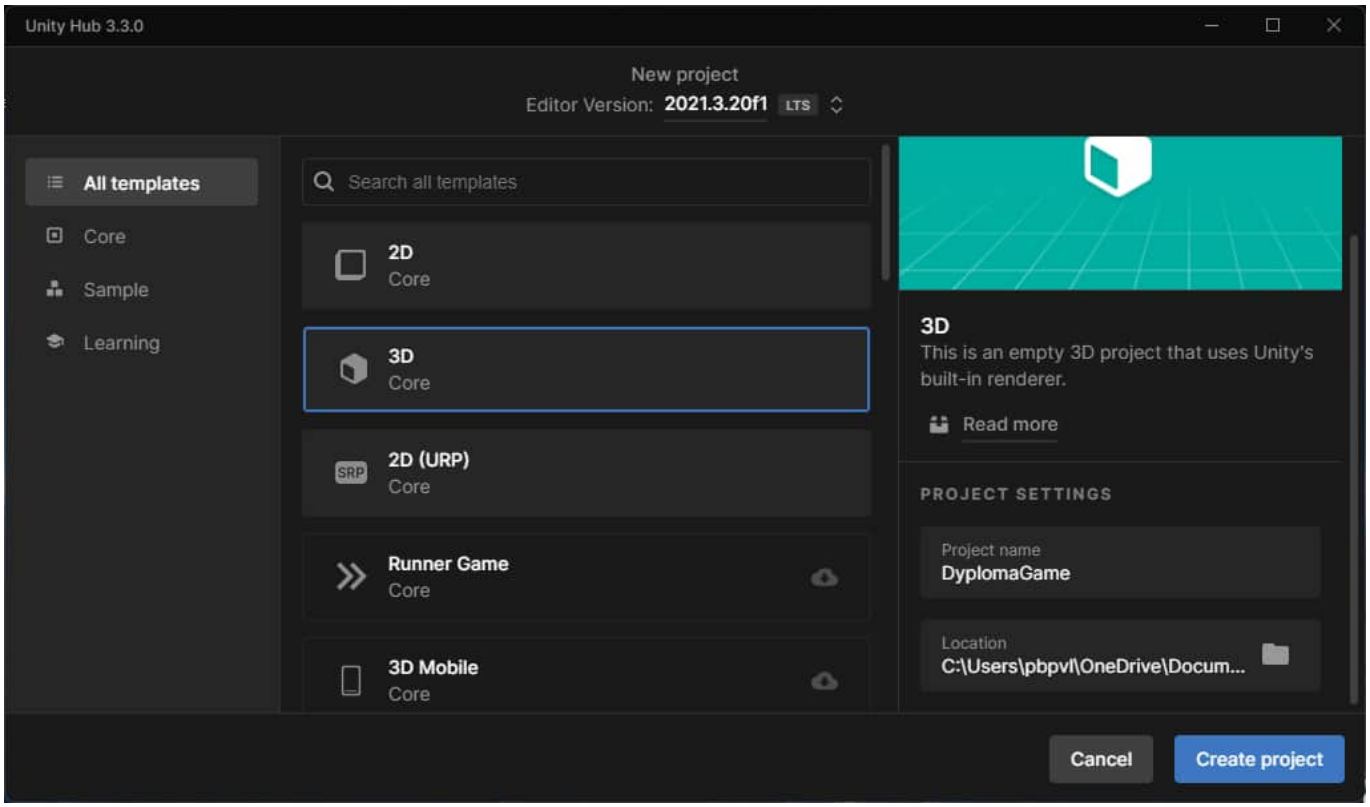


Рис. 3.1 - Створення нового проекту в Unity Hub

Нас зустрічає таке вікно пустого проекту, як на рис. 3.2(я змінив назву проекту щоб в подальшому не виникло конфліктів):

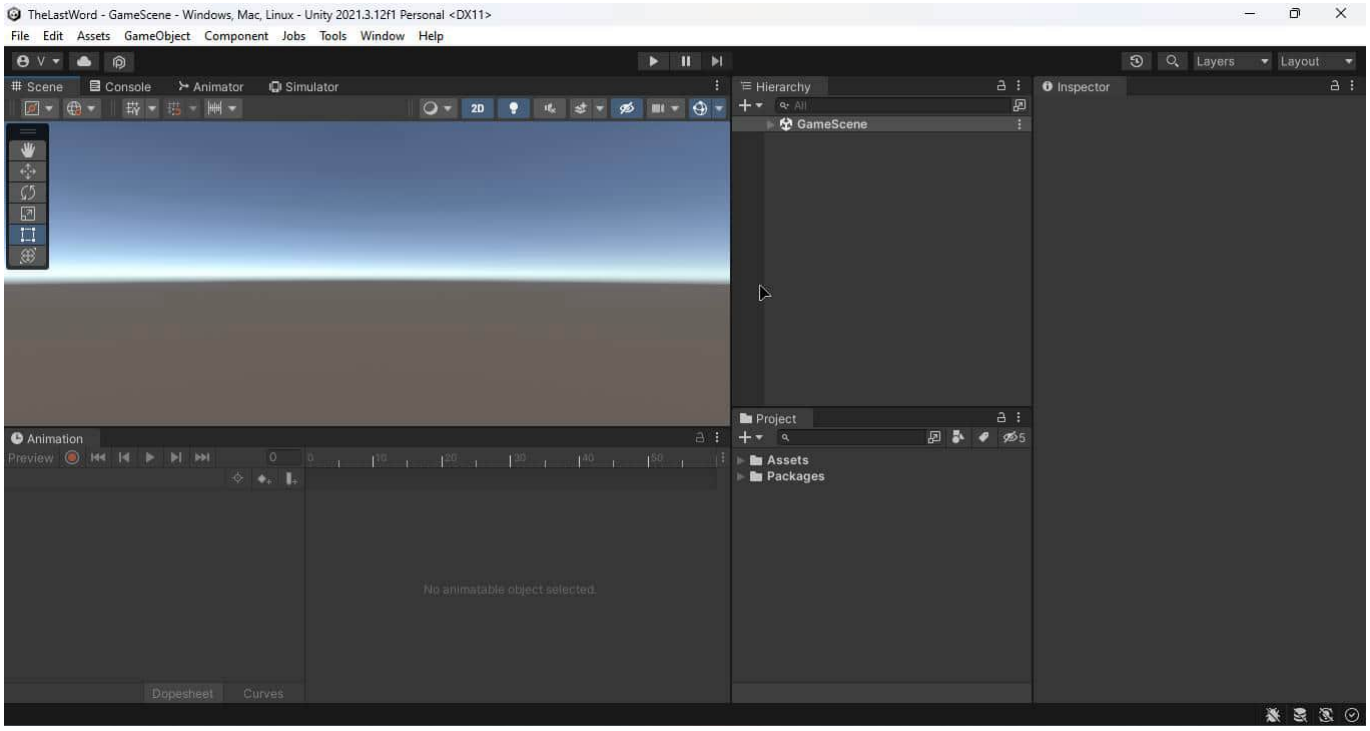


Рис. 3.2 - Інтерфейс "голого" проекту

## 3.2 Написання кодової бази

Перш за все, потрібно написати скрипт, що буде брати інформацію з Json файлу, що слугуватиме базою даних, та парсити її на потрібні мені ігрові об’єкти.

Головна логіка застосунку в тому, що гравцю на екран будуть виводитись випадкові слова, та буде індикатор часу, що залишився, перш ніж той програє. Гравець повинен якомога швидше набрати на віртуальній клавіатурі слово, що зображене на екрані, щоб індикатор часу збільшився на кілька секунд. В будь-який момент гравець може натиснути кнопку, щоб дізнатися більше про слово, яке він вводить, себто його опис. Наприклад «морозиво» - *клац* - «Солодкий та прохолодний десерт, що робиться з молока». Щоб додати елемент цікавості та ігрового духу для дітей, буде створено цікаву сцену з персонажем, за життєвим станом якого гравець повинен слідкувати. У цього персонажа буде 4 елементи стану:

здоров’я, настрій, голод та витривалість. Усі ці елементи і будуть чотирма таймерами, що будуть зменшуватись до нуля. Якщо хоча б один з елементів стану дійде до нуля, гра завершиться та гравець програє. Щоразу як гравець введе правильно слово, застосунок перевірятиме, до якої категорії належало слово, і буде прибавляти очки часу елементу стану цієї категорії. Наприклад, слово «лікар» додасть +5 секунд до стану «здоров’я». Чи слово «морозиво» дасть +5 секунд стану «голод», і так далі.

За вище описаною логікою гри, у базі даних у мене буде всього три таких елемента:

- Категорія
- Слово
- Опис слова

З цього виходить, що потрібно створити метод у публічному новому класі, з трьома string елементами. У статичному методі Awake() на початку гри ми будемо загружати заздалегідь підготовлений файл. **(ДОДАТОК – WordsPool.cs).**

Далі створимо перевірку, щоб ненароком слова не містили лишніх символів, таких як пробіли або великі літери. Після конвертації в методі викличемо інший статичний метод щоб за допомогою вбудованої функції рандома він брав випадковий елемент з бази даних. **(ДОДАТОК А – Keyboard.cs).**

Після цього створимо простеньку(поки що) сцену, на яку додамо необхідні GameObject’и – об’єкти, що дозволяють містити в собі будь-які компоненти, функції, тощо. Створимо користувацький інтерфейс, в якому помістимо 1 текстове поле, 1 поле для вводу тексту, та таймер, що відображатиме кількість часу, яка залишилась у гравця, надалі ми змінимо його на індикатори стану.

Коли я переконався, що все працює, прийшов час додати екранну віртуальну клавіатуру, що буде працювати на мобільних пристроях, адже до цього моменту я працював тільки на клавіатурі. Її я створював шляхом канвасу та сортувальників, я створив 3 горизонтальні ряди, та в кожному розмістив клавіші української абетки.

Також я додав 2 спец. символи – тире, та апостроф, якщо слова будуть містити такі символи. Вирівнявши все, я зайшов у фотошоп і створив зображення для самих кнопок клавіш, та заднього фону клавіатури. Фінальний вигляд запрограмованої та готової до використання віртуальної клавіатури можна побачити на рис. 3.3.

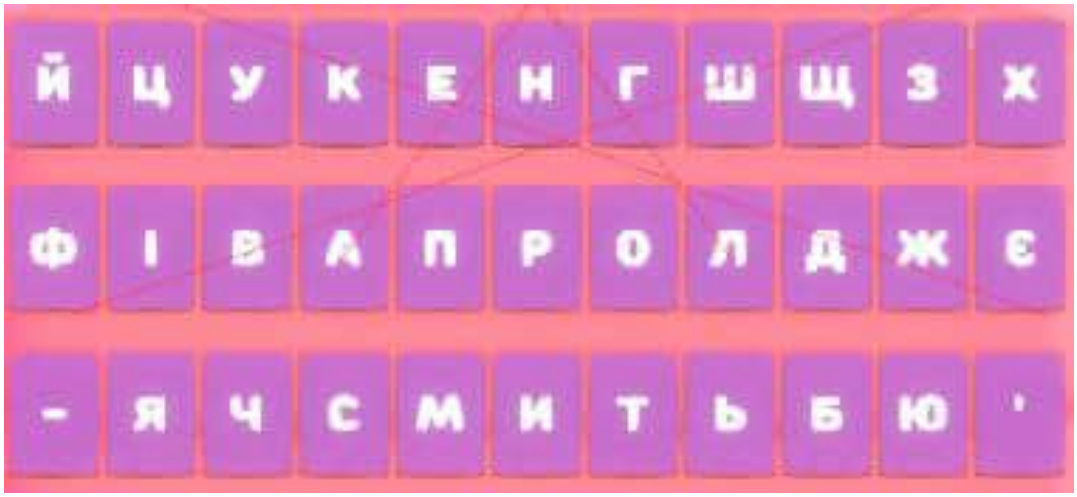


Рис. 3.3 - Фінальний вид віртуальної клавіатури

Після цього, я вирішив реалізувати до кінця механіку стану. Для цього довелось написати багато коду, та відсортувати слова по категоріях. На рисунку 3.4 нижче можна побачити приклад бази даних.

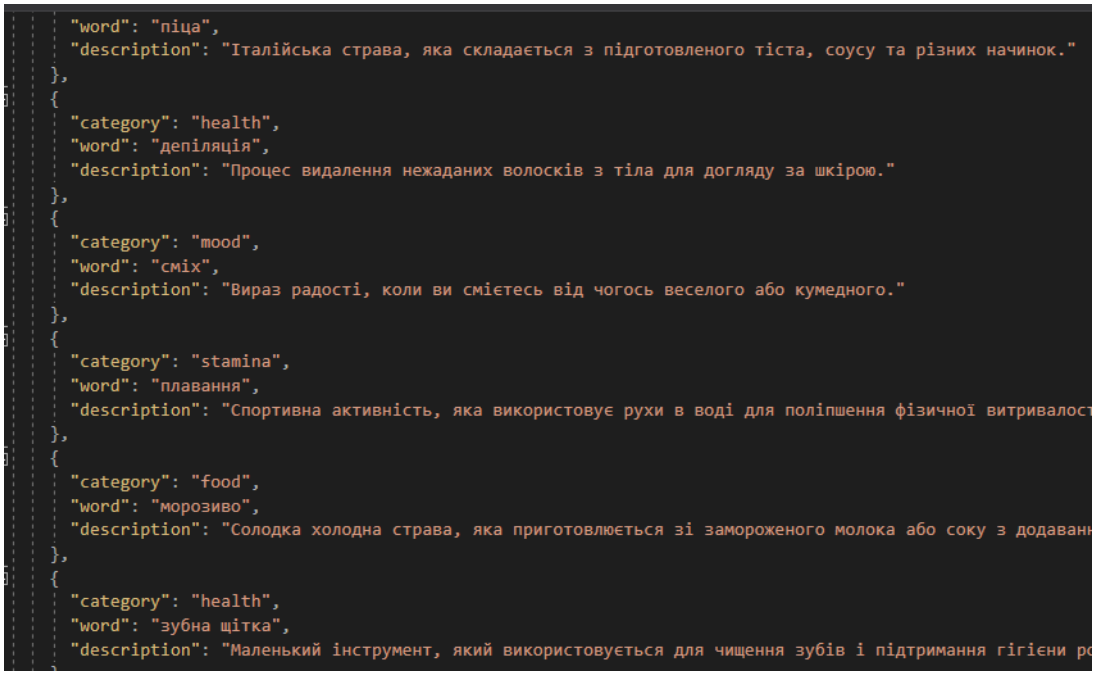


Рис. 3.4 - Вигляд бази даних

Далі я створив код, що дозволяв зменшувати чотири стани синхронно чи навіть асинхронно, з плином часу. Також він буде мати перевірку на те, чи натиснута пауза, щоб поки гравець читає опис слова таймер не йшов. Цей файл змінює колір барів(елементів стану) згідно з їхньою категорією прямо у інспекторі, після чого зменшує його значення. **(ДОДАТОК А - ProgressBar.cs)**

Також для зручності напишемо функціонал, що буде змінювати колір слова згідно того, скільки букв гравець вже правильно натиснув. Тобто якщо гравець ввів пів слова, то ця половина слова буде іншого кольору. Для цього потрібно ділити стрінг слова на дві частини, і фарбувати їх в різний колір. Оскільки до цього букви що гравець вже ввів просто зникали, то така реалізація є набагато кращою. **(ДОДАТОК А – Keyboard.cs).**

Для всього цього функціоналу довелось реалізувати кілька допоміжних класів, таких як **GameManager.cs** та **ButtonClickDetector.cs** для того, щоб слідувати всім законам та правилам ООП та SOLID. А на рис. 3.5 вид альфа-версії сцени на ілюстрації.**(ДОДАТОК А – GameManager.cs та ButtonClickDetector.cs).**

Рис. 3.51 - Альфа-версія сцени

Далі я анімував елементи стану, так звані прогрес бари, щоб вони змінювались в реальному часі на сцені під час запуску застосунку. У мене була така заготовка під прогрес бар, як на рис. 3.6:

Рис. 3.6 - Зображення елементу стану персонажа

У коді оголошуються змінні для керування прогрес-барами здоров'я, їжі, витривалості та настрою. Крім того, є змінна, що вказує, чи гра наразі призупинена.

Клас має метод Awake, який викликається при створенні об'єкта. У цьому методі встановлюється значення складності гри на основі збережених налаштувань.

Також отримується посилання на аніматор головного персонажа гри, якого ми створимо на наступному етапі.

Метод ShowEnd відповідає за показ екрану завершення гри, з якого буде перехід в метод GoToMainMenu, що переводить гравця до головного меню гри, яке буде створено незабаром.

У методі Start використовується так звана «Корутина». Це IEnumerator, який створений для того, щоб оновлювати значення прогрес-барів у циклі з певною частотою. Якщо гра не призупинена (isPaused = false), викликається метод UpdateBars, який зменшує значення прогрес-барі. Якщо значення прогрес-барів стають меншими або рівними 0, викликається метод ShowEnd.

Методи InitializeBars та UpdateBars відповідають за ініціалізацію та оновлення значень прогрес-барів відповідно. Значення зменшуються залежно від складності гри, а якщо хоча б одне значення прогрес-бару стає меншим або рівним 0, викликається метод ShowEnd.

Метод CheckBarValue виконує різні перевірки значень прогрес-барів і викликає метод ApplyAnims, який змінює анімації головного персонажа відповідно до значень прогрес-барів.

Методи PauseGame та ResumeGame відповідають за призупинення та продовження гри відповідно шляхом зміни значення змінної isPaused і налаштування швидкості часу в грі.

Увесь код знаходиться у додатку**. (ДОДАТОК А – BarsManager.cs).**

### 3.3 Створення ассетів.

Оскільки застосунок орієнтований на дитячу авдиторію, за мету я поставив створити красиві ассети. Я обрав за персонажа «Слайма», як такого ну дуже нейтрального персонажа, який не виглядає жорстоко, навіть якщо помирає. За об’єкти сцени я довго думав, і вирішив зупинитися на тропічному лісі у воксельному стилі. Оскільки я не вмію створювати 2д-ассети ні в одному стилі окрім як піксель-арт, мені

довелось адаптувати 3д-стилістику під нього, щоб все виглядало більш-менш органічно. Також я створив та знайшов необхідні UI(User Interface) елементи.

Приступимо до створення сцени у Blender, RizomUV та Substance Painter 3D.

Для початку я зробив так званий GreyBox, який використовується в розробці для відмітки загального вигляду об’єкту. Його вигляд можна побачити нижче на рис. 3.7.

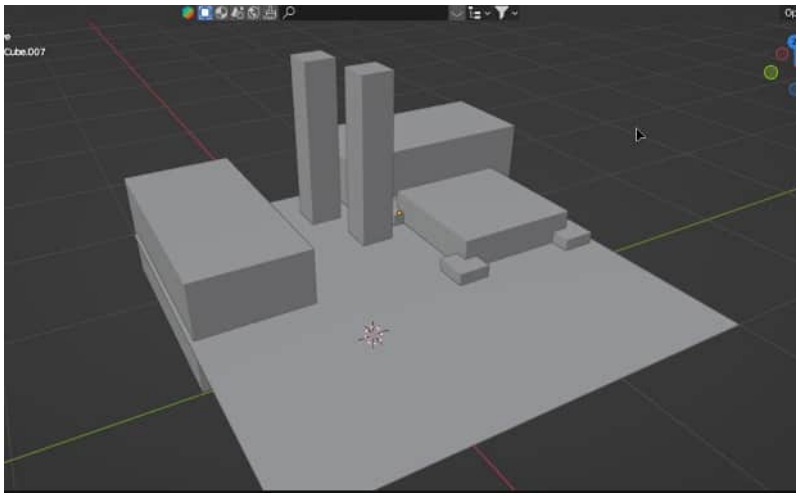

Рис. 3.7 - GreyBox сцени

Після цього я приступив до створення деталей та міні-обє’ктів, дочірніх елементів. Для зручності замість сірого кольору поставив тимчасово насичений зелений колір, щоб краще відчувати атмосферу лісу. Так, це необхідно для правильного розуміння композиції. Я створив основну форму застосовуючи всі правила геометрії та математики загалом, також правила дизайну, стилю, композиції та оптимізував модель під вбудовані системи та мобільні платформи. Вигляд самого «мешу», без текстур наведено на рис. 3.8.

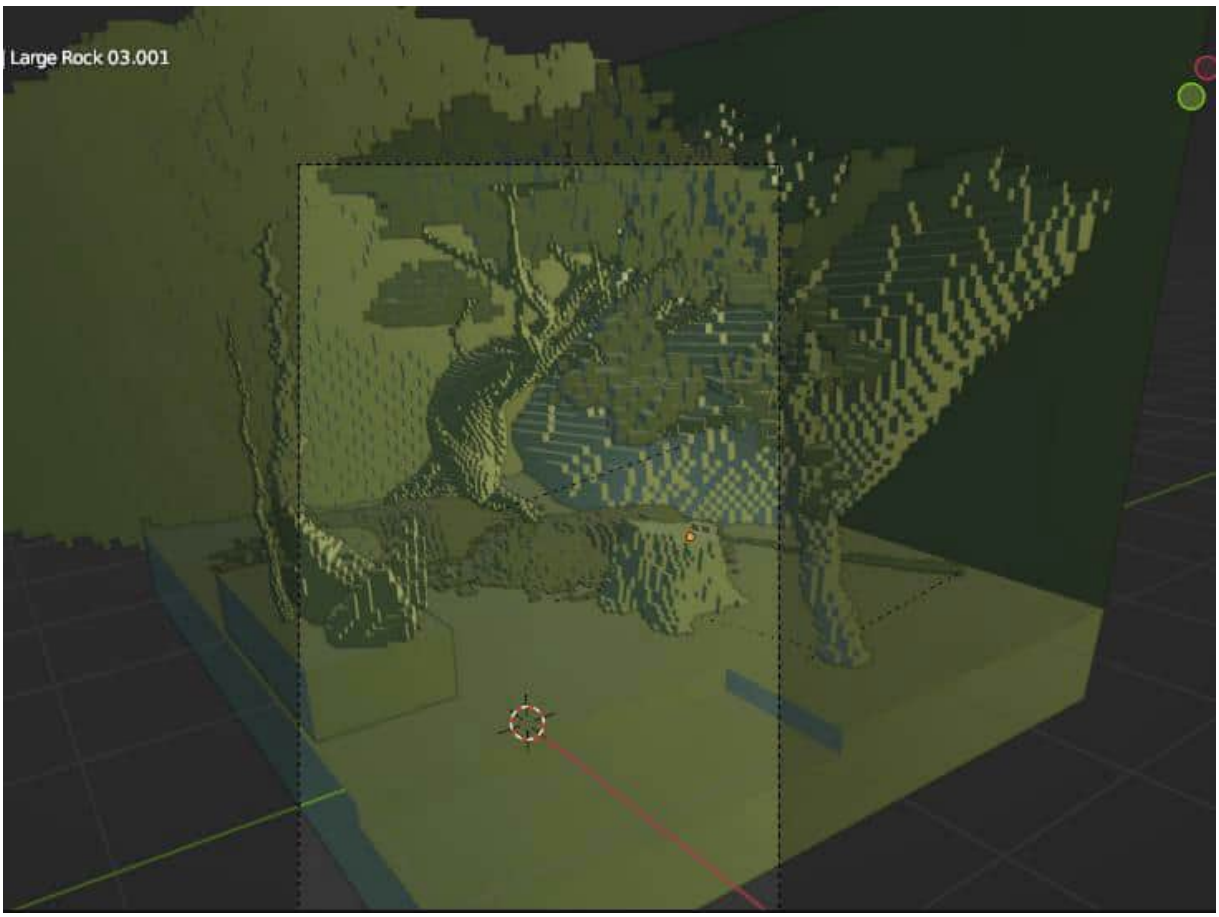


Рис. 3.8 - Вигляд мешу сцени

Для того, щоб правильно пофарбувати будь-яку модель, і щоб на неї в майбутньому правильно падало світло, для неї потрібно зробити правильну UV-розгортку, так зване «оригамі». Для цього переходимо у RizomUV, та акуратно нарізаємо наші кубики. Процес розгортки включає в себе нарізання моделі на маленькі острівки та рівномірне упакування її на плоскому квадраті, який в майбутньому міститиме текстуру. Сам процес розгортки можна бачити на рис. 3.9, де зліва показано 3д-вигляд моделі та тестову текстуру чорно-білої дошки для розуміння розмірів, а зліва – 2д-розгортку.

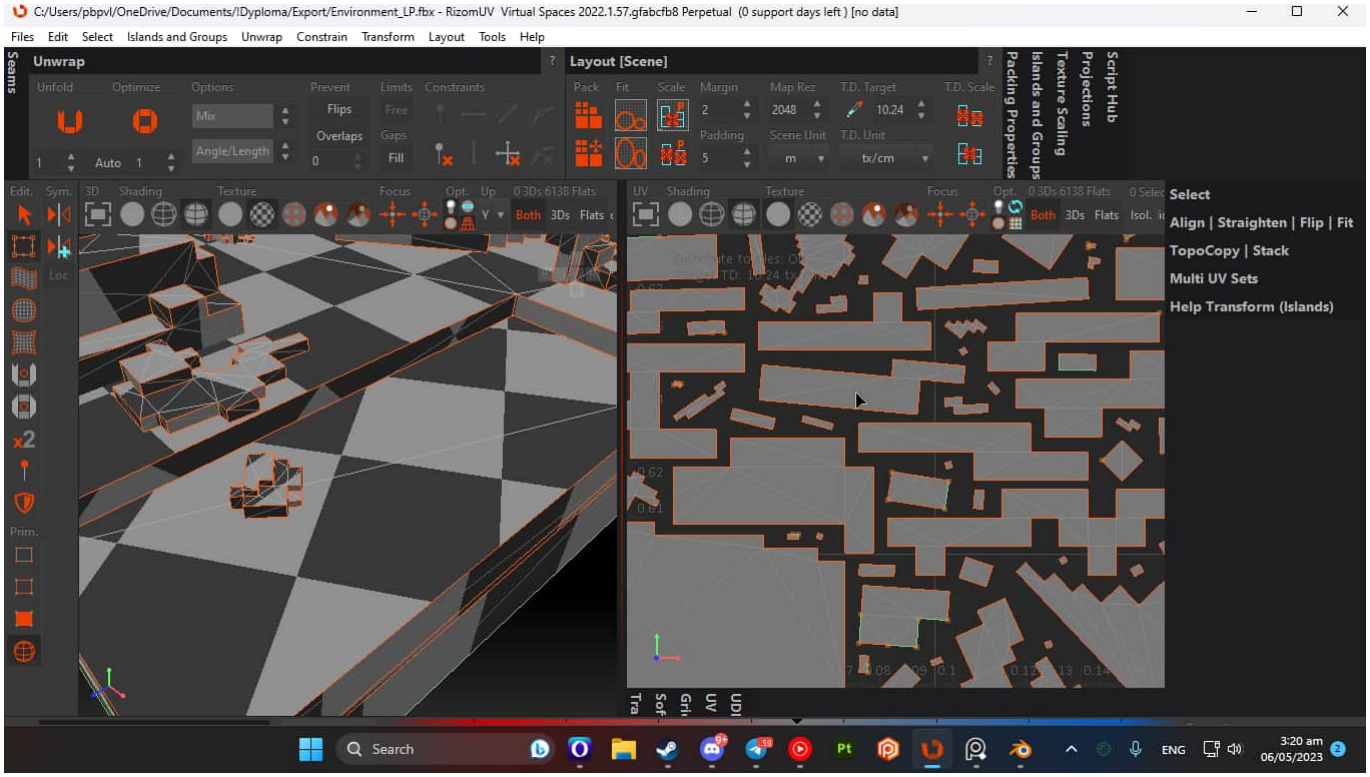


Рис. 3.9 - Процес розгортки моделі

Після успішного нарізання моделі, ми переносимось у Substance Painter. В ньому ми, заздалегідь поділивши модель на матеріали, будемо фарбувати її у різні кольори та запікати на ній деталі та карти глибини, нормалей, кривизни, гладкості, та інші. Процес текстурингу моделі можна бачити на рисунку 3.10.

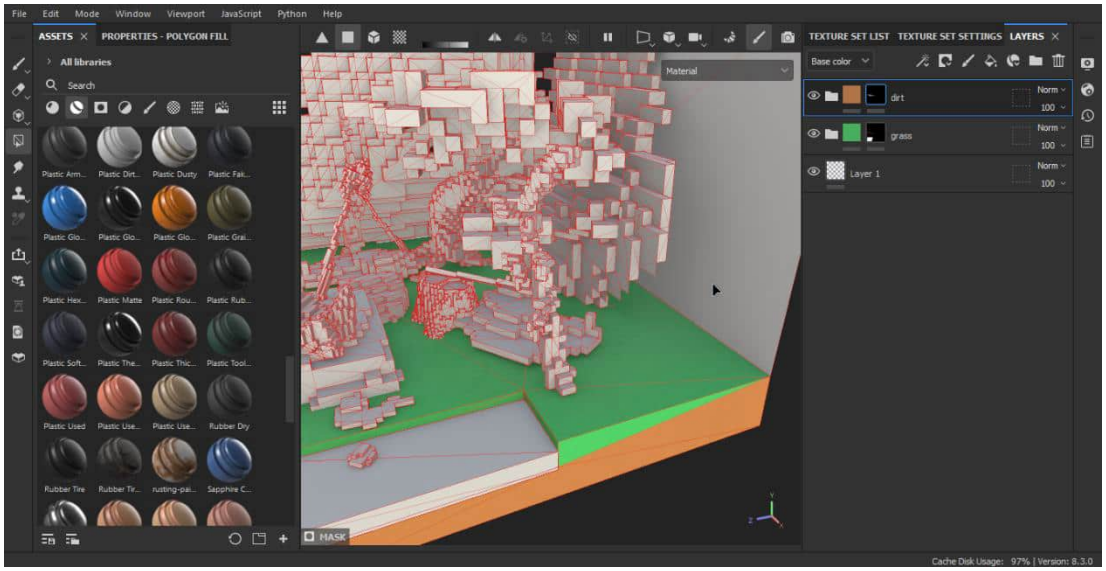


Рис. 3.10 - Процес текстурингу

Після завершення повного циклу створення моделі, нам потрібно перенести її на сцену у рушії, імпортувавши її. Це можна зробити просто перетягнувши її з провіднику до вікна проєкту. Дехто каже, що для успішного імпорту потрібно використовувати спеціальні плагіни, так от, це міт. Справжнісінький, нічим не підкріплений міт, адже офіційно Unity не рекомендує ніяких плагінів для імпорту.

Після успішного імпорту моделі звісно ж з'явився баг, який я не продумав, але я швидко його виправив, просто змінивши позиції елементів у ієрархії та шари. Він полягав у тому, що сама модель була на передньому плані, в той час як користувацький інтерфейс був на задньому. Звісно ж, повинно бути навпаки.

Ось як сцена виглядала до(рис. 3.11):

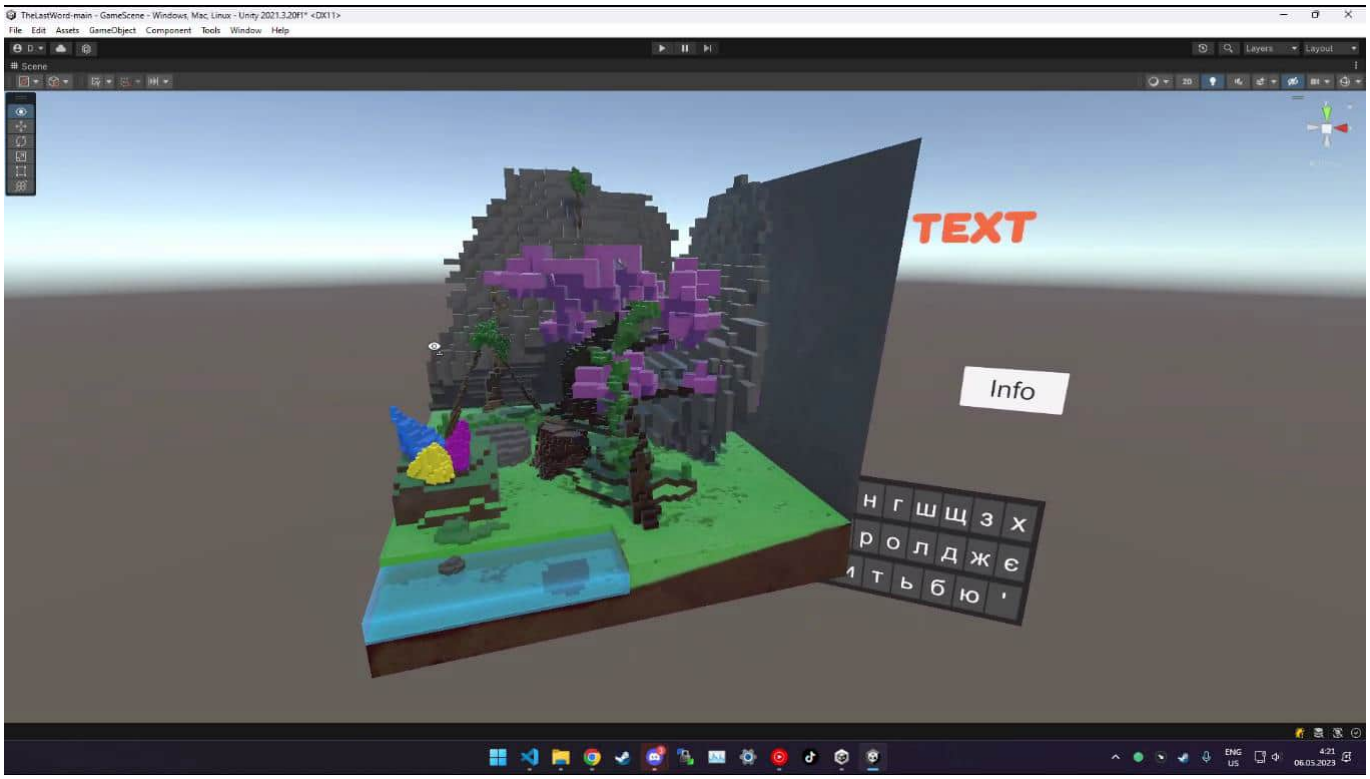


Рис. 3.11 - Вигляд сцени у рушії до виправлення багу

Таке часто трапляється, якщо відразу правильно не налаштувати користувацький інтерфейс, що я, звісно ж, за браком досвіду вчасно не зробив. Дякувати Богу, це швидко виправляється всього в декілька кліків. В процесі розробки

в мене виникали складніші баги та поломки, проте нажаль я не встиг їх закоментувати та зробити скріншот. Серед таких багів були:

- Неправильне розташування клавіатури
- Неправильний вивід слова у текстовий об’єкт
- Невірно створена база даних
- Проблеми з діленням строки зі словом навпіл, де вона ділила всі слова
- Та інші, менш значні проблеми

Після цього потрібно було правильно виставити камеру та інтерфейс відносно об’єкту. На скриншоті нижче можна побачити вже виправлену сцену(рис. 3.12):

Рис. 3.12 - Виправлена бета-сцена

Згодом, я приступив до створення головного персонажа. Як я вже сказав, був обраний «Слайм», що українською буде звучати як «Слимак» чи «Слизняк». Проте ці слова рідковживані, та частіше ви зустрінете англоцизм слайм. Його потрібно було

анімувати, щоб він змінював свій візуальний стан згідно з поточним станом елементів-таймерів, так званих прогрес-барів.

Оскільки стилістика – піксель-арт, то тип анімації сам прийшов до голови, і це спрайтова покадрова анімація. Зайшовши в фотошоп і створивши головного персонажа, зображеного на рис. 3.13(окуляри це тестовий варіант), я приступив до анімацій.

Рис. 3.132 - Альфа-версія персонажа

Анімацій повинно бути всього три:

- Анімація радості
- Анімація нейтрального стану
- Анімація плачу, паніки

Кожен з цих станів відображатиме поточний стан прогрес-барів. Він прив’язаний таким чином, що як тільки хоча б один прогрес-бар падає нижче певного рівня,

анімація змінюється на наступну, якщо ж він повертається назад – то і анімація повертається.

Нижче можна бачити три рисунки(3.14, 3.15 та 3.16) анімацій, які у мене вийшли. Так сталось, що різні анімації містять у собі різну кількість кадрів, адже я вирішив, що так буде більш оптимізований ігровий процес. Та й не потрібно анімації радості великої кількості кадрів.

Рис. 3.14 - Анімація радості

Рис. 3.15 - Анімація нейтрального стану

Рис. 3.16 - Анімація плачу та паніки

У цілях тестування я ще імпортував персонажа до Blender’у, щоб подивитись чи буде він дивитись органічно з іншими кольорами при правильному освітленні. Справа у тому, що кожна програма використовує свої алгоритми для рендеру, і один і той самий об’єкт у різних програмах може виглядати по-різному. Усі моделі створені у Blender чи 3DMax часто йдуть у переробку, через те, що вони не співпадають з очікуваннями. Саме тому, для синхронізації результатів я спочатку подивився, як об’єкт виглядає на іншому рушії рендур. Результат тестування можна побачити нижче на рисунку 3.17.

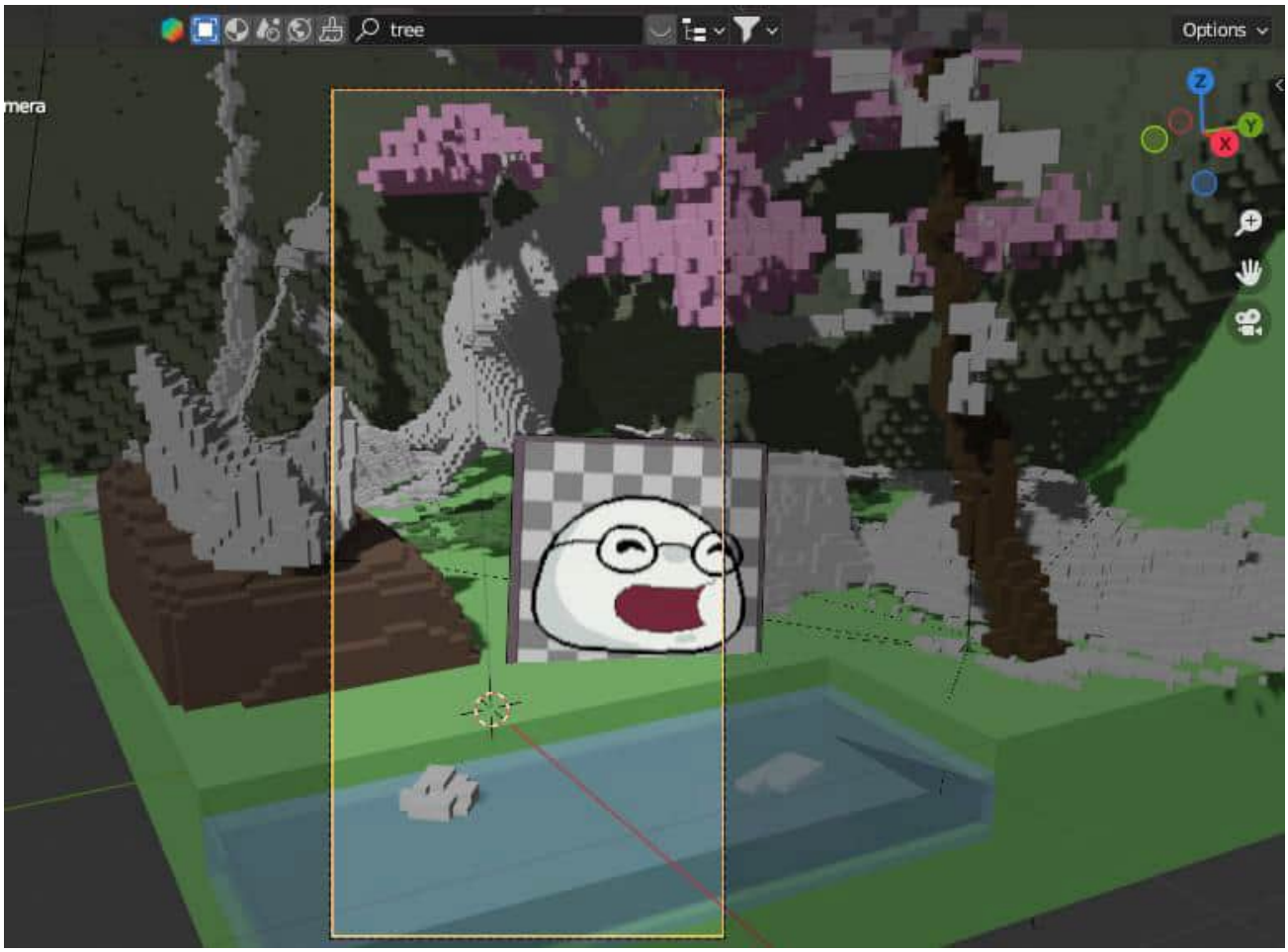


Рис. 3.17 - Тестування Рендеру

Зрештою, коли усі анімації були готові, я імпортував персонажа у ігровий рушій та приступив до створення ассетів інтерфейсу.

Я на швидку руку створив кілька картинок різних геометричних фігур, квадратів, трапецій, шестикутників, кіл, прямокутників, тощо. З них я розробив красивий графічний інтерфейс.

### 3.4 Полірування коду та проєкту загалом.

Я прийняв рішення зробити більш-менш красиве головне меню, адже до цього воно виглядало м’яко кажучи не дуже. Я використовував плейсхолдери – це так звані тимчасові елементи, що використовуються для того, щоб замінити ними щось, чого ще поки що не існує у проекті. Це може бути будь що, від моделей до елементів інтерфейсу. В моєму випадку інтерфейс головного меню виглядав приблизно так, як показано на рисунку 3.18.

Рис. 3.18 - Плейсхолдери у меню

Оскільки я вже створив усі необхідні ассети для інтерфейсу, я потратив час на те, щоб привести його то ладу та зробити красивим. Нове меню зображене на рисунку 3.19.

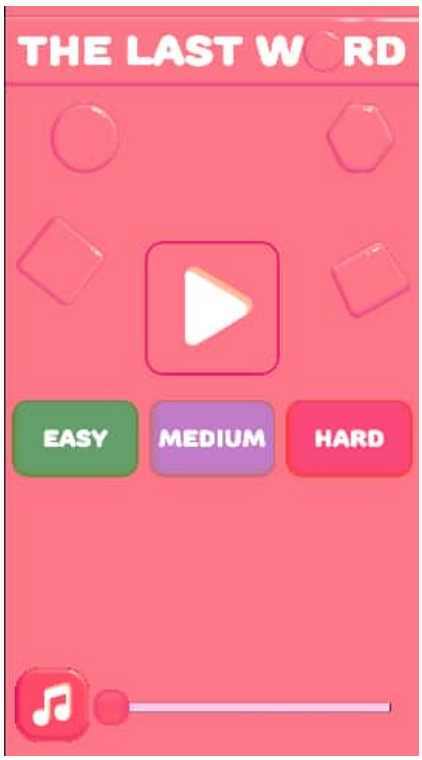


Рис. 3.19 - Вигляд нового меню

Як можна бачити з зображення вище, я додав у гру музичний супровід, знайшовши безкоштовну музику без авторських прав на певних приватних ресурсах. І хоча я звик робити все самостійно, я не «людина-оркестр» і далеко не композитор, тому не здатен самостійно створювати музичні композиції. Знайшовши кілька 8-бітних треків, я створив для них плеєр в кодовому файлі, який на старті проекту сортував їх у **абстрактний** плейлист, після чого запускав композиції по колу. Щоб було зручніше коригувати гучність, я вивів у головне меню повзунок гучності, який зберігає своє положення навіть при перезапуску гри, створюючи файл налаштувань.

Потім я ще зробив декілька мінорних покращень, серед них:

- Зробив так, щоб персонаж змінював свій колір на колір елементу стану, що є найменшим, тобто має найменший рівень стану.
- Зробив так, щоб колір слова відповідав його категорії
- Зробив так, щоб при введенні кількох слів підряд без помилок, гравцю виводився текст про кількість ідеальних слів

- Зробив так, щоб при натисканні на невірну букву з’являлась тряска екрану у гравця
- Зробив елементи стану круглими, адже так вони краще відображають свою суть на мою думку
- Зробив адаптивну верстку під різні формати екранів, оскільки системи у всіх різні, з різними розширеннями та форматами співвідношення сторні.
- Зробив так, щоб у головному меню на сцені можна було обирати складність, адже я вирішив, що гравці бувають різних рівнів досвідченості, і так застосунок підійде усім категоріям користувачів.
- Придумав грі назву та вивів в головному меню: “The Last Word”

Після всіх цих операцій я виправив кілька незначних багів, та доповнив базу даних слів новими словами, додавши до них відповідні категорії та описи.

Кінцевий вигляд ігрової сцени виглядає так(рис. 3.20):

Рис. 3.20 - Кінцевий вигляд ігрової сцени

### 3.5 Тестування продукту

Тестування було захоплюючим та важливим етапом розробки. Я, як розробник, розумів, наскільки важливо мати якісне тестування, щоб гарантувати коректну роботу гри перед її випуском.

Однак, у мене не було достатньо коштів, щоб найняти професійних тестувальників, тому я звернувся до своїх друзів, які погодилися взяти участь у QA тестуванні. Вони були чудовими помічниками та віддано допомагали мені з тестуванням гри.

Під час тестування ми фокусувалися на введенні слів на швидкість та спостереженні за станом персонажа слайма. Ми перевіряли, чи правильно реагує гра на введені слова, чи відбувається зміна швидкості гри відповідно до рівня складності, а також чи коректно відображаються стани персонажа.

Ми активно тестували різні сценарії та введення слів, переконуючись, що гра працює стабільно та без помилок. Також ми перевіряли, як геймплей стає важчим з часом, та реагували на будь-які непередбачувані ситуації або помилки.

Цей процес QA тестування з допомогою друзів дав мені цінні відгуки та знання про можливі проблеми, які потребували виправлення перед випуском гри. Без їхньої допомоги і залучення, цей етап розробки був би складнішим. Я дуже вдячний своїм друзям за їхню підтримку та вклад у тестування The Last Word.

### 3.6 Реліз у Play Market

Для релізу гри у крамницю Google, необхідно було:

- створити робочий білд гри через Android Studio
- згенерувати ключ розробника
- зробити скріншоти гри для сторінки
- підключити хоча б тестову рекламу в проект використовуючи заготовки AdMob

- Пройти вікову перевірку адміністрацією Google
- Пройти перевірку проекту на недостовірну інформацію, та перевірку правил крамниці
- Залити фінальний білд на сайт розробника
- Натиснувши кнопку “Release Build” чекати, поки гра з’явиться у крамниці

# ВИСНОВКИ

Я опанував велику кількість нових технологій, зокрема роботу з C#, Unity, анімаціями, Google Developer Console, покращив свої навики володіння Visual Studio Code, Adobe Photoshop, Blender, створенням користувацького інтерфейсу та базами даних. Навчився правильно дотримуватись візуального стилю застосунку, розприділяти задачі по рівнях складності, правильно оцінювати об’єм роботи, та час на її виконання, поглиблено працювати з системами контролю версій, тестувати продукти та доводити їх до фінальної стадії розробки.

У кінці розробки було зроблено повністю готовий застосунок, що можна завантажити на смартфон та користуватись. Була зроблена база даних, геймплей, візуальне оформлення, та багато інших речей. Застосунок вирішує проблему лагідної українізації, і тепер дозволяє дітям українцям спокійніше вивчати нові слова, без примушення, та у приємній ігровій формі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Duolingo, веб-ресурс. URL: https://uk.duolingo.com (дата звернення: 02.03.2023)
2. Quora, веб-ресурс. URL: https://www.quora.com/How-do-game-developers-earn-money-from-people-downloading-their-free-app (дата звернення: 03.03.2023)
3. Академія професій майбутнього, веб-ресурс. URL: https://academy.ck.ua/ru/kursu-po-razrabotke-igr-na-unity-3d-cherkassy/ (дата звернення: 03.03.2023)
4. Winiarska, веб-ресурс. URL: https://winiarska.com/blog/digital-state-of-mind/graph-showing-global-mobile-vs-desktop-user-numbers/ (дата звернення: 03.03.2023)
5. Researchgate, веб-ресурс. URL: https://www.researchgate.net/figure/QA-graph-edge-creation_fig3_359079736 (дата звернення: 03.03.2023)
6. Вільна енциклопедія Wikipedia. ULR: https://uk.wikipedia.org/wiki/Файл:Waterfall_model.svg (дата звернення – 03.03.2023)
7. Naurok, веб-ресурс. URL: https://naurok.com.ua/urok-klaviatura-uvedennya-okremih-simvoliv-zmina-movnih-rezhimiv-204805.html (дата звернення: 04.03.2023)
8. Geckoandfly, веб-ресурс. URL: https://www.geckoandfly.com/8529/smart-keyboard-pro-t9-keyboard-for-google-android/ (дата звернення: 04.03.2023)
9. Вільна енциклопедія Wikipedia. URL: https://uk.wikipedia.org/wiki/Blender (дата звернення: 16.03.2023)
10. Вільна енциклопедія Wikipedia. URL: https://uk.m.wikipedia.org/wiki/Файл:Blender_shader_node_editor_showcasing_a_Moldy_Gold_Materiel.png (дата звернення: 16.03.2023)

11. Вільна енциклопедія Wikipedia. URL: https://uk.wikipedia.org/wiki/Android_Studio (дата звернення: 15.03.2023)

12. IdealSoft, веб-ресурс. URL: https://idealsoft.com.ua/rizom-lab-in-war/ (дата звернення: 20.03.2023)

13. 3D-Export, веб-ресурс. URL: https://ua.3dexport.com/3dmodel-demon-character-2-444613.htm (дата звернення: 20.03.2023)

14. Вільна енциклопедія Wikipedia. URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 20.03.2023)

15. DBPedia, веб-ресурс. URL: https://dbpedia.org/page/OpenGL_Shading_Language (дата звернення: 25.03.2023)