ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Гаврило Артем Михайлович

УДК 004.4

Розробка мобільної гри в жанрі “Idle”

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Гаврило А.М.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гаврило Артем Михайлович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:

Розробка мобільної гри в жанрі "Idle"

керівник роботи:

Ващишак Сергій Петрович, к.т.н., доцент

затверджена наказом вищого навчального закладу від «12» березня 2023 року № 19/01

2. Термін подання студентом роботи 05.06.24

3. Вихідні дані роботи: методи та алгоритми, рушій

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

   1. Опис об'єкту та порівняння аналогів
   2. Розробка функціоналу та модель гри
   3. Програмна реалізація проекту

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Дослідження існуючих аналогів	21.03.2024	Виконано
2.	Проектування архітектури та створення дизайну додатку	29.03.2024	Виконано
3.	Програмна реалізація додатку	14.04.2024	Виконано
4.	Розгортання та тестування додатку	26.04.2024	Виконано
5.	Формування висновків	02.05.2024	Виконано
6.	Оформлення пояснювальної записки	03.05.2024	Виконано
7.	Оформлення графічного матеріалу та підготовка до захисту роботи	16.05.2024	Виконано

Студент _____ Гаврило А.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ващишак С.П.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

АНОТАЦІЯ

Кваліфікаційна робота присвячується вивченню, аналізу та розробці відеогри для мобільних пристроїв в жанрі Idle, створеної в розважальних цілях і для подальшої публікації і монетизації.

Перший розділ присвячений опису відео ігрової індустрії, аналізу конкурентів та існуючих інструментів для покращення майбутньої гри, виявлено переваги та недоліки конкурентів.

В другому розділі описується концепція проекту, візуальний дизайн, архітектура, рушій, технології та інструменти, використані під час розробки.

В третьому розділі розповідається про програмну складову проекту, та покроково описуються кроки, пройдені під час розробки проекту, їх цінність і причини використання.

В результаті було створено повноцінну відеогру, що в подальшому буде опублікована на платформі «Play Market». Гра буде спроможна задовольнити потреби значної аудиторії, і в подальшому буде розширюватись, тестуватись і зростати, до подальшої публікації.

КЛЮЧОВІ СЛОВА: UNITY, IDLE, СКРИПТ, ACET, PLAY MARKET.

SUMMARY

The qualification work is devoted to the study, analysis and development of a video game for mobile devices in the Idle genre, created for entertainment purposes and for further publication and monetization.

The first chapter is devoted to the description of the video game industry, competitor analysis and existing tools to improve the future game, the advantages and disadvantages of competitors are revealed.

The second chapter describes the project concept, visual design, architecture, engine, technologies and tools used during development.

The third section describes the software component of the project, and describes step by step the steps taken during the development of the project, their value and reasons for use.

As a result, a full-fledged video game was created, which will later be published on the "Play Market" platform. The game will be able to meet the needs of a large audience and will continue to expand, test and grow until further publication.

KEYWORDS: UNITY, IDLE, SCRIPT, ASSET, PLAY MARKET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Idle – жанр відеоігор, що базується на накопиченні ігрових ресурсів без залучення гравця.

Відеоігри – програмне забезпечення, створене в розважальних цілях.

Геймплей – це повторюваний процес або патерн взаємодії між гравцем і грою. Є ключовою складовою відеоігор .

Свайп – спосіб взаємодії гравця з грою, дія при якій гравець проводить пальцем по екрану.

Скрипт – послідовність команд, які виконує ігровий рушій.

Використовується для автоматизації виконання певних завдань або операцій.

Мокап – це спеціальний макет, зображення реального предмета, на яке накладається елемент дизайну: логотип, обкладинка, скріншот та інше. Мокапи дозволяють відразу побачити, як об'єкт буде виглядати в реальності.

Канвас – це графічний контейнер, який використовується для відображення інтерфейсу користувача (UI) в грі. Він є основним елементом для створення і розміщення UI елементів, таких як кнопки, тексти, зображення.

ВСТУП

Актуальність теми: людям завжди, навіть в найдавніші часи було потрібно якось себе розважати, це частина нашої природи, так як людський мозок просто не може витримати багато часу в статичному середовищі. Для цього були створені ігри, які в основному були мімікою реальних дій дорослих, і відповідні іграшки [1]. З тих пір минуло багато часу, але наші потреби залишились тими самими. Стимуляція, прогресс, розвиток, це те що нам необхідно. І відеоігри розроблені спеціально для цього.

І серед всіх платформ для відеоігор, безсумнівно найбільшою популярністю користуються мобільні пристрої, на що є безліч причин, основними з яких є загальнодоступність і простота в створенні. В відеоігри можна грати завжди і будь де, адже в наш час всі користуються телефонами.

Є безліч жанрів мобільних відеоігор, які користуються популярністю в абсолютно різних вікових і етнічних групах.

Одним із цих жанрів є «Idle». Він є актуальним в наш час завдяки зростанню інтересу до мобільних ігор, технологічному прогресу, популярності гейміфікації та можливостям монетизації, які він пропонує. Цей жанр дозволяє короткі ігрові сесії, завдяки чому в нього можна грати на роботі, в перервах між уроками, і в повсякденному життю.

Мета і завдання дослідження: метою роботи є розробка відеогри в жанрі Idle у виді мобільного додатку з використанням ігрового рушія Unity.

Для досягнення поставленої мети передбачається виконання наступних завдань:

– перегляд аналогів у вибраному жанрі та прикладів реалізації ключових ідей;
– розробка дизайну гри;
– втілення ідей у вибраному рушії;
– реалізація системи здобування прогресу;

– проведення тестування розробленого продукту.

Методи дослідження: для досягнення поставленої цілі будуть використовуватися наступні методи: метод аналізу – для дослідження сучасних методів та інструментів гейм дизайну, а також аналізу результатів роботи ігрового додатку. Метод порівняння – для вивчення аналогів в жанрі, ефективності певних механік та вибору найкращого з можливих варіантів. Метод тестування – для запевнення працездатності додатку, та раннього виключення можливих дефектів Метод програмування на мові C# – для реалізації Idle механік та її інтеграції в додаток

Ці дослідницькі підходи дозволять глибоко проаналізувати обраний об'єкт, розробити відеогру з використанням рушія Unity, а також здійснити тестування для забезпечення її ефективної функціональності.

Об’єкт дослідження: процес реалізації ігрових механік та проекту на ігровому рушії Unity.

Предмет дослідження: методи, та інструменти рушія Unity, система офлайн процесів, інтеграція новітніх механік в архітектуру додатку, а також оптимізація розробленого додатку.

Практичне значення одержаних результатів: результатом даної кваліфікаційної роботи є готова мобільна гра у жанрі Idle, яку можна запустити на Android пристроях.

Апробація результатів дослідження: Матеріали кваліфікаційної роботи були представлені на XI Міжнародній науковій конференції «Студентські наукові дискусії поза форматом», яка відбулася 11 квітня 2024 року в Університеті Короля Данила.

Структура роботи: розділи – 3. Обсяг основної частини – 45. Кількість використаних джерел – 20.

РОЗДІЛ 1. ОПИС ОБ'ЄКТУ ТА ПОРІВНЯННЯ АНАЛОГІВ

1.1 Опис відеоігор як один з способів провести вільний час

Відеоігри є очевидним кроком в розвитку розважальних технологій, так перша відеогра «Tennis for Two» з'явилась в 1958, і це при тому що популяризація персональних комп'ютерів почала зростати тільки з 1970, і інтернет як явище з'явився тільки в 1969. Тобто відеоігри були створені як тільки в людства з'явилась на це можливість.

Вони виконують безліч функцій, але перш за все це розважальний фактор. Як вже було описано вище, в людей є природна необхідність в розвагах, і відеігри для мобільних пристроїв ідеально для цього підходять.

Звісно, не всі ігри підходять для всіх, існують різні категорії населення, і вони всі потребуються різних, конкретних видів розваг. Для цього існують «Жанри» – підвиди ігор, спрямовані на певну діяльність в іграх, через що самі ігри стають більш орієнтованими на певні групи населення. Симулятори, шутери, головоломки і так далі.

Але ігри існують не тільки для того щоб розважати людину, хоча це і є основним фактором. Часто відеоіграм приписують здібність розвивати гравця в певному напрямку – точність, уважність, швидкість реакції, вміння адаптуватися, стратегічність, це лишень невелика частина від усіх навичок, які гравці здобувають і розвивають, користуючись ігровими додатками.

Значна частина сучасних ігор мають навчальних характер. Гравцям може надаватись базова інформація з різних областей науки, технологій і мистецтва. Часто ігри базуються на справжніх технологіях, і надають інформацію, яка в не ігровому вигляді важко засвоюється.

Також слід виділити соціальний фактор, який також є дуже значним фактором в ігровій індустрії. Велика кількість відеоігор використовує

соціальний фактор для стимуляції гравців. Таблиці лідерів, можливість порівняння своїх результатів з друзями, онлайн змагання і ігрова кооперація, всі ці явища сильно сприяють популярності відеоігор, тому що

Популярність відео ігор та ігрових платформ, таких як Play Market, відкрила можливості для багатьох невеликих компаній та незалежних розробників зайняти своє місце на ринку геймінгу.

В цілому, відеоігри представляють собою складне явище, яке впливає на різні аспекти життя людини. Вони не лише забавляють, але й надають можливість для особистісного розвитку, відпочинку та соціальної взаємодії. Проте, важливо зберігати баланс та уникати перевантаження відеоігровою діяльністю, оскільки зловживання нею може мати негативні наслідки, як і в випадку з будь яким іншим видом розваг.

1.2 Суть жанру «Idle»

Жанр Idle (також відомий як «Incremental») – це особливий тип відеоігор, який відрізняється від інших жанрів особливими механіками, і ставленням до гравця, особливо до його часу [2]. Основна ідея цього жанру полягає в тому, щоб гравці могли отримувати задоволення, не потребуючи постійної активності чи уважності до гри [3].

Основні переваги жанру:

— мінімальна активність гравця, в порівнянні з іншими жанрами цей вирізняє себе дуже спокійним ставленням до активності гравця, його ніяк не буде покарано, якщо він раптом вимкне додаток, тоді як інші жанри часто при цьому скасовують останні здобутки;

— поступовий прогрес, гравці можуть спостерігати за зростанням свого "прогресу" в грі, навіть якщо вони не активно взаємодіють з нею. Це дозволяє створювати відчуття досягнень та задоволення від гри, навіть у відсутність активної участі;

— розвиток без обмежень часу: багато ігор у жанрі Idle працюють в

реальному часі, навіть після того, як гравець вийшов з гри. Це призводить до того що гравець може отримувати ресурси або досягати прогресу, навіть коли він не грає;

— заборона на безперервну діяльність – геймплейний цикл жанру Idle

не дозволяє грати в нього безперервно, час від часу гравцю буде необхідно відволікатись на посторонню діяльність, тому що потрібен час, щоб ігрові ресурси встигли накопичитись. Не всі вважають це плюсом, але саме тому цей жанр є значно менш шкідливим для буденного життя людини, до нього спокійно можна допускати дітей, не переймаючись про виникнення ігрової залежності.

1.3 Огляд та аналіз існуючих конкурентів

Існує чимало ігор в жанрі Idle, хороших і не дуже. Загалом якість ігор в цьому жанрі оприділяються якістю ігрових механік, оптимізацією додатку, кількістю ігрового контенту і реклами.

Перед тим як почати розробляти ігровий додаток, було проведено детальне вивчення конкурентів на ринку, користуючись пошуком в магазині «Google Play Store» за запитом «Idle». Серед результатів було відібрані наступні:

— Deep Town Idle Mining Shelter;

— IdleOn – The Idle RPG;

— Earth Inc. Tycoon Idle Miner;

— Idle Cave Miner.

Хоча ці ігри і можуть здаватись достатньо хорошими на перший погляд, при більш детальному аналізі в них виявляються достатньо суттєві недоліки, достатньо сильні, щоб у користувача зникло бажання в них грати, що веде до подальшого видалення додатку.

Deep Town Idle Mining Shelter – популярна та довго існуюча гра, що була завантажена в Google Play Store ще в 2017 році. В грі є незвична механіка розвідування глибокої печери, щоб діставатись до глибших і цінніших копалин (рис. 1.1), створення виробництв, велике різноманіття ресурсів, битви з

супротивниками та розробка інструментів для цього. Певний час гравець часто здобуває новий досвід.

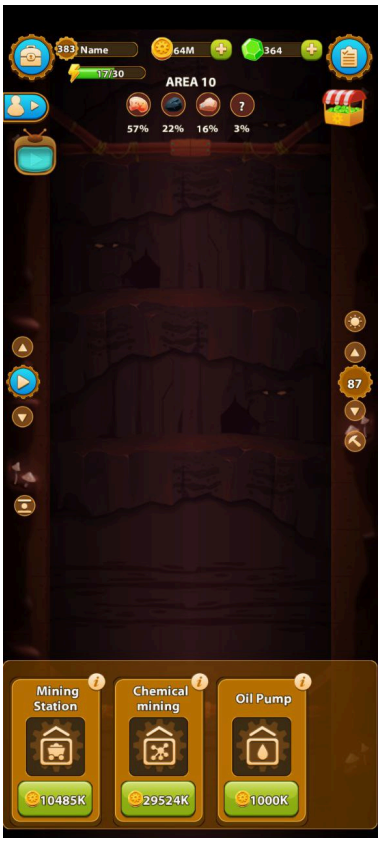


Рисунок 1.1 – Печера, в якій можна створити виробництво

Але нажаль з часом, випадково чи через жадібність розробників, в грі з’явились досить неприємні мінуси. Першим з них, як і в більшості Free to Play ігор, є надлишкова кількість рекламних вікон, і тому подібне(рис. 1.2). Ці банери можуть бути корисним для збільшення прибутку від гри, але тільки в лімітованій кількості, і при не надто значній нав’язливості. В цьому ж випадку ці банери дуже швидко починають дратувати гравця, і відволікають його, що залишить в нього погане враження.

Також тут присутній звичний мінус жанру Idle, а саме така річ як «Pay Wall», з якою швидко стикається гравець. Суть цього мінусу заключається в тому, що з часом прогресія гравця штучно сповільнюється, для створення наступного рівня виробництва постає необхідність в захмарних кількістях ресурсів, які безоплатним способом можна добути тільки за дуже великий

проміжок часу (дні, тижні, місяці). Це зроблено заради того щоб гравець більш охоче скористався можливістю купити ігрові ресурси за справжню валюту.

Хоча цей метод і є дієвим, серед розробників це вважається поганим тоном, адже при такій архітектурі нормальне, безоплатне проходження гри є майже неможливим, і точне не настільки приємним.

Рисунок 1.2 – Дратуючі рекламні вікна

IdleOn – The Idle RPG – гра з досить незвичною концепцією для цього жанру, тому що конкретно в цій грі у гравця є персонажі, якими він може керувати, а не лишень ігрові екрани, з яких він взаємодіє з грою, а також набір здібностей. Гравець повинен віддавати персонажам різноманітні команди, після чого переключатись на інших персонажів гри (рис. 1.3). Також в грі присутня велика кількість здобуваємих ресурсів, онлайн система, ігровий ринок, битви між гравцями, битви з босами, і тому подібне.

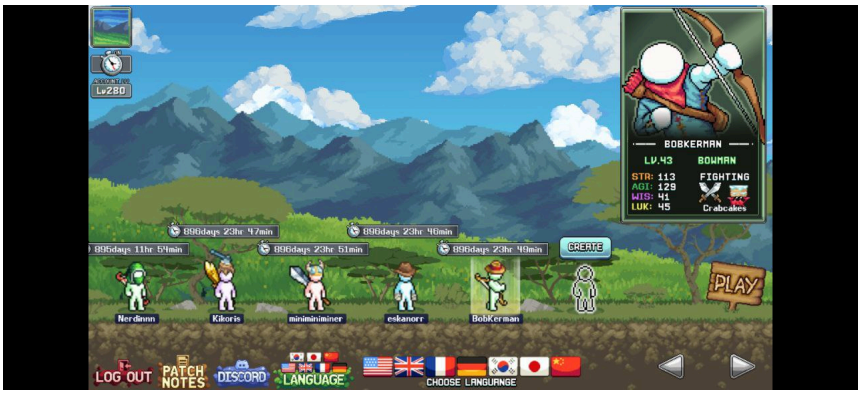


Рисунок 1.3 – Екран вибору персонажів

З плюсів також можна виділити онлайн механіки. Кожна ігрова локація є спільною для всіх гравців на сервері (об’єм серверів обмежений) і тому можна споглядати за іншими гравцями, якщо вони знаходяться неподалік (рис. 1.4).

Хоча з ними і не можна взаємодіяти напряму, можна писати їм повідомлення в чаті, а також обмінюватись предметами на ігровому ринку.

Рисунок 1.4 – Інший гравець займається своїми справами

Серед плюсів також можна виділити значну кількість ігрового контенту, локацій і предметів, завдяки чому в цій грі хочеться залишатись на значний час.

Але і тут також вистачає мінусів, хоч і не звичних для жанру, але достатньо суттєвих щоб відштовхнути гравця. Першим серед них слід виділити погану оптимізацію самого додатку. Всі внутрішні дії виконуються з помітною

затримкою, а при переході між локаціями гравець значний час змушений просто дивитись на порожній екран, поки додаток завантажить все необхідне.

Було помічено, що якість мобільного пристрою не надто впливає на ці ефекти, тобто проблема швидше знаходиться на стороні ігрових серверів, ніж на недостатній потужності ігрових пристроїв. В наш час оптимізація є надзвичайно важливим фактором, і майже всі гравці звертають на це увагу. Дуже просто втратити потенційного гравця, якщо тримати його на екрані ігрового завантаження надмірну кількість часу.

Також тут присутня проблема, яка часто зустрічається в онлайн іграх – необхідність в постійному інтернет підключенні. І не в простому підключенні, а в хорошій пропускній здатності. Хоча онлайн фактор і не надто впливає на основний геймплей, гра зобов'язує гравця бути постійно підключеним до інтернету, можливості грати без інтернет підключення гравцю не надається.

І ще одна проблема, присутність онлайн ринку. Хоча це часто називають плюсом, адже через це ігровий світ стає більш живим, він також несе за собою значні проблеми, які не так просто помітити з першого погляду.

На ігровому ринку гравці можуть обмінюватись ігровими предметами, і виставляти на них таку ціну, яку вважатимуть справедливою. Розробникам приходиться це враховувати, і робити добування найпростіших ресурсів більш складним, бо інакше ці ресурси дуже швидко перенаповнять ринок. Через це в гравців по суті немає необхідності робити значну кількість речей самостійно, адже все можна купити на ринку. І це якщо на ринку є люди, адже не завжди ігровий онлайн тримається високого рівня. У випадку якщо кількість гравців падає, ринок згасає, але ресурси все ще важко добувати, через регуляцію зі сторони розробників, що також ускладнює здобування задоволення від гри

Earth Inc. Tycoon Idle Miner – гра досить звична для цього жанру в плані геймплею, з можливістю будівництва заводів, добування руд, вдосконалювання виробництв, і тому подібного. Але і в ній є декілька деталей що вирізняють її на фоні конкурентів. Основним з них можна назвати сеттинг, і те як ігровий світ реагує на дії гравця. Зазвичай в подібних іграх світ ніяк не змінюється, тільки

з’являються нові виробництва, заводи і так далі, але фон, на якому це відбувається, залишається статичним. Тут це не так, адже в грі є параметр «Потепління», який поступово зростає з створенням нових виробництв гравцем, що напряму відображається на фонових зображеннях, які гравець постійно споглядає – якщо спочатку на фоні можна побачити зелені поля і квіти, то після певного ігрового прогресу фон змінюється на пекельні ландшафти, а потім і на лавові озера, з чорними хмарами (рис. 1.5).

Рисунок 1.5 – Кінець гри

Звісно ж, в цій грі вистачає мінусів, особливо важливо врахувати їх під час власної розробки, тому як вони можуть залишити сильно негативне враження в гравця.

Перш за все це викривлена мораль. В грі є безліч морально невірних дій, які тим не менш потрібно робити заради ігрового прогресу. Серед них: сприяння, або навіть спричинення глобального потепління, повне руйнуванням екосистем, вдосконалення з назвами в стилі «Ухилення від податків» або «Експлуатація робочих», закупка ресурсів на «чорному ринку» і так далі.

Хоча це і подано в сатиричному вигляді, але молодий користувач ймовірніше за все цього не зрозуміє, що може спричинити те що молоде покоління буде зростати, думаючи що глобальне потепління це добре, і що експлуатація робочих це хороший спосіб збільшення доходу.

Серед інших мінусів значним є майже повна відсутність сюжету як такого, єдиним стимулом до подальшого розвитку є короткі повідомлення завдань, які зводяться до «добудь 10 цього» або «прокачай це 3 рази» (рис. 1.6).

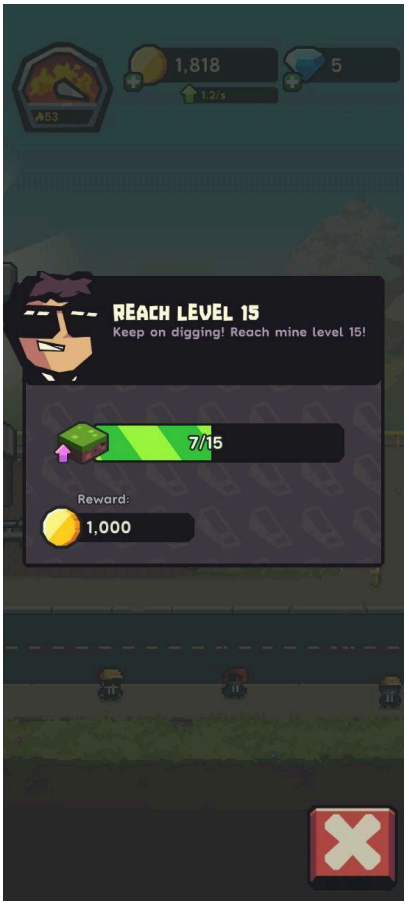


Рисунок 1.6 – Бідні на сюжет завдання

Хоча ці повідомлення і є настільки зрозумілі, наскільки це можливо, що сприяє швидкому освоювання в молодшої аудиторії, вони не надають гравцю абсолютно ніякої важливої чи цікавої інформації, і є надзвичайно однотипними. Якщо гравець хоче чогось більшого, ніж найпростіші завдання, то він нічого знайти не зможе.

По суті в грі прогрес нічим не виправдовується, це просто «прогрес заради прогресу», або як сказав один публіцист «Рішення, яке створене щоб шукати проблему».

При розробці продукту важливо враховувати ці фактори. Ігровий прогрес повинен бути морально виправданим, і етичним, а сюжетні пояснення повинні цікаво і розгорнуто пояснювати, що і для чого гравець робить, а не просто заповнювати простір.

Idle Cave Miner – в цій грі нам необхідно прокопувати шари в печері, добуваючи (самостійно або за допомогою найманців) блоки породи, в яких також знаходяться руди. Чим глибше гравець прокопується, тим більш цінні та рідкісні руди можна знайти в породі (рис. 1.7).

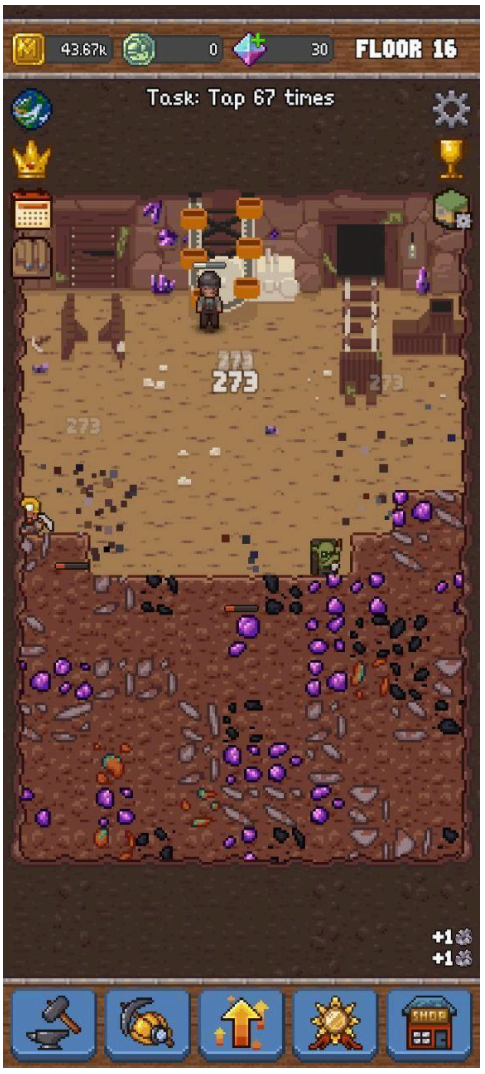


Рисунок 1.7 – Видобування породи

Серед плюсів в цій грі можна виділити достатньо приємну музику, на яку попередні ігри були скупі, а також загалом приємний дизайн

Але через те що ігрові механіки тут доволі стандартні для жанру, то і звичні проблеми жанру тут також присутні – слабка сюжетна складова, мала кількість механік взаємодії з грою, присутність донатної валюти, необхідність часто переглядати рекламу заради стабільного прогресу.

Також можна виділити мінус ігрової архітектури, який не настільки часто зустрічається в жанрі, а саме репетативність. В грі є два види ресурсів, тимчасові і постійні. На початку гравець здобуває тільки тимчасові ресурси, але як тільки прогрес зайде достатньо далеко, гравець може зробити «ресет» гри, відновивши весь прогрес до початкового стану, зі втратою всіх тимчасових ресурсів, і получивши за це енну кількість постійного ресурсу, який можна обміняти на постійне покращення роботи виробництв. Цей метод іноді використовується в Idle жанрі, як ефективний спосіб затримати гравця в грі, і приховати невелику кількість загального контенту, показуючи йому в основному тільки той контент, який він вже бачив. Очевидно, що для розробників це вважається поганим тоном, тому як це один з найдешевших способів приховати невелику кількість контенту в грі, і цю механіку дуже важко реалізувати в цікавому вигляді.

Висновки до розділу 1

В даному розділі було проведено аналіз предметної області проекту мобільної розробки, а також були оглянуті конкуренти, що займають близьку нішу на ігровому ринку.

Вивчення сильних сторін конкурентів допомагає зрозуміти, що приваблює гравців до цих ігор та які функції або елементи мають найбільший успіх. Це надає можливість вдосконалити власний продукт та зробити його більш привабливим для аудиторії.

З іншого боку, аналіз слабких сторін конкурентів дозволяє виявити можливі прогалини чи недоліки, які можна уникнути або покращити у власній грі. Це може включати в себе розробку нових функцій, які компенсують недоліки конкурентів або покращення якості геймплею.

На основі проведеного дослідження було обрано основну задачу кваліфікаційної роботи – розробка idle гри з допомогою рушія Unity, в якій буде рівномірне наповнення контентом, будуть включені елементи сюжету, збалансований геймплей, помірна кількість реклами та загальна оптимізація.

Цей розділ становить фундамент для подальшого дослідження та розробки, забезпечуючи чітке розуміння потреб користувачів та вимог до функціоналу майбутнього додатку.

РОЗДІЛ 2. РОЗРОБКА ФУНКЦІОНАЛУ ТА МОДЕЛЬ ГРИ

2.1 Рушій Unity

Для створення власної мобільної відеогри, перший крок полягає в виборі відповідного рушія, призначеного спеціально для відео-ігрової розробки. Це важливий етап, який визначає можливості проекту, візуалізує концепції та допомагає оцінити складність реалізації ідей.

Ігровий рушій, який також кличуть ігровим двигуном, є фундаментальним компонентом у процесі розробки відеоігор. Він представляє собою комплексну платформу, яка забезпечує розробника необхідними інструментами та ресурсами для втілення креативних ідей. Від графічного дизайну та фізичної моделювання до штучного інтелекту та аудіо ефектів – ігровий рушій об'єднує всі ці компоненти в єдину систему.

На ринку існує безліч різних ігрових рушіїв, кожен з яких має свої особливості, переваги та недоліки. В даній роботі було обрано зосередитись на Unity, як на одному з найпопулярніших ігрових рушіїв на сучасному ринку. Unity відзначається своєю гнучкістю, широким набором інструментів та дружньою для користувача інтерфейсною платформою. Ці переваги роблять його ідеальним вибором для розробки ігор на всі платформи, особливо для початківців та тих, хто шукає ефективний інструмент для реалізації своїх ідей.

Unity вирізняється своєю оптимізованою архітектурою, що робить його ідеальним вибором для розробки невеликих та середніх ігор. Порівняно з конкурентами, такими як Unreal Engine, Unity вимагає менших ресурсів комп'ютера для запуску та роботи проектів. Це особливо важливо для розробників, які працюють на менш потужних пристроях або мають обмежені ресурси для розробки.

Крім того, Unity славиться своєю простотою та інтуїтивно зрозумілим інтерфейсом. Незважаючи на його простоту, платформа пропонує глибокий

функціонал, що дає розробникам можливість втілювати найскладніші ідеї з відносною простотою (рис. 2.1).

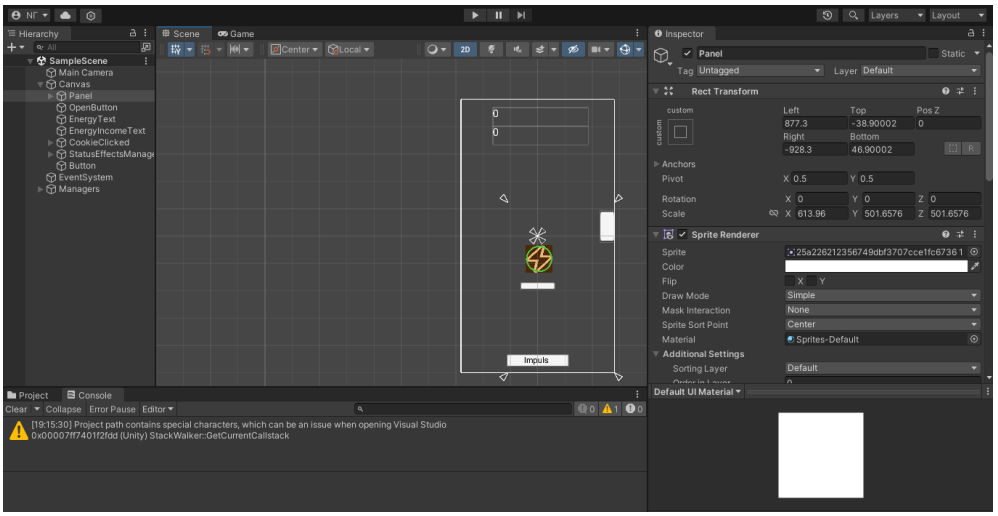


Рисунок 2.1 – Робочий простір Unity

Іншою ключовою перевагою Unity є активна та велика спільнота розробників, яка надає величезний обсяг ресурсів, підтримки та інструментів. Існує безліч сторонніх розширень та додаткових інструментів, які полегшують розробку ігор, а також допомагають вирішувати різноманітні проблеми. Завдяки активній спільноті, розробники можуть швидко знаходити рішення на форумах, статтях та онлайн-курсах, що спрощує процес розробки та надає безліч натхнення для створення нових ідей та функціоналу для їхніх проектів.

Також слід виділити такий корисний інструмент як Asset Store, вбудований в Unity. Це незамінний ресурс для проекту завдяки доступу до широкого вибору готових ресурсів, таких як моделі персонажів, текстури, анімації, звуки та ефекти. Використання цих готових активів дозволяє ефективно заощадити час і ресурси на розробку проекту. Також, Asset Store пропонує різноманітні розширення та плагіни, що можуть розширити функціональність проекту, надаючи нові можливості та інструменти для його розвитку.

Цих причин цілком достатньо, щоб зрозуміти що Unity чудово підходить як для розробки самостійних проектів, так і для командної роботи, відзначається високою продуктивністю та гнучкістю.

Обрана мова програмування C# ідеально синхронізується з Unity, що робить процес розробки інтуїтивно зрозумілим і продуктивним. Строга типізація C# дозволяє виявляти та коригувати помилки на ранніх стадіях розробки, забезпечуючи високу стабільність та надійність коду. Також, простий і чистий синтаксис мови робить її доступною навіть для початківців, одночасно надаючи гнучкість та потужність для впевнених в собі досвідчених розробників. Ця комбінація особливостей робить C# відмінним вибором для розробки ігор в Unity, дозволяючи швидко та ефективно реалізовувати ідеї і концепції.

2.2 Загальний опис і технології

Створюючи гру, потрібно звернути увагу на велику кількість елементів – механіки, рівень складності, сценарій. Вибір домінуючої колірної палітри гри не здається вирішальним для всіх розробників. Але кольори суттєво впливають на загальне сприйняття гравців.

Цей проект планується як сучасна, добре структурована та ефективна розважальна програма для мобільних пристроїв, орієнтована на доступні технології в рушії Unity, та мову програмування C#.

Проект планується до випуску на мобільні платформи, тому рушій був оптимізований спеціально для цього, а саме були встановлені необхідні розширення. Також слід виділити що кінцевий продукт буде розроблятись тільки з використанням ліцензійного програмного забезпечення, задля гарантії стабільність і надійність роботи додатку.

Для забезпечення зручності користувачів та безпеки їх даних до моменту онлайн релізу, всі налаштування та досягнення користувача будуть зберігатися локально на пристрої. Це забезпечує миттєвий доступ до здобутого прогресу без

зайвих завантажень або затримок, завдяки використанню вбудованих в Unity інструментів, таких як PlayerPrefs для зберігання ключових значень, і механізмів серіалізації файлів для ефективного керування всіма даними.

В подальшому, задля розширення функціональності та забезпечення синхронізації даних між різними пристроями користувачів, планується підключення до хмарних сервісів, таких як Google Play Market. Ці хмарні рішення будуть функціонувати як серверні бази даних, що гарантують безперебійну синхронізацію геймерських досягнень та налаштувань між різними пристроями користувачів, та їх збереження. Таким чином, гравці зможуть насолоджуватися грою на будь якому пристрої, не втрачаючи свій прогрес та досягнення.

При основній розробці будуть задіяні такі інструменти Unity:

– **Sprite Editor** (Редактор спрайтів): цей інструмент надає можливість не тільки створювати, але й редагувати та анімувати 2D спрайти. Він дозволяє легко створювати живу графіку для персонажів, об'єктів та ефектів, оптимізуючи процес розробки;

– **Tilemap Editor** (Редактор тайлів): дозволяє створювати та редагувати тайлмапи (карти, побудовані з тайлів). З його допомогою можна швидко створити складні та деталізовані рівні гри;

– **2D Physics Engine** (Фізичний двигун для 2D): відповідає за симуляцію реалістичних фізичних ефектів у 2D просторі, включаючи гравітацію, зіткнення та інші важливі аспекти;

– **Animator** (Аніматор): створення та керування анімаціями для персонажів, об'єктів та інших елементів. Він підтримує створення складних анімацій, що додає грі більше живості та динаміки;

– **Particle System** (Система частинок): цей інструмент дозволяє створювати різноманітні візуальні ефекти, такі як вогні, вибухи, дим, дощ та інші, використовуючи частинки для створення реалістичних та динамічних сцен;

– **Audio Mixer** (Звуковий мікшер): інструмент для керування

звуковими ефектами та музикою у грі;

– **UI** (Користувацький інтерфейс): інструмент для створення об’єктів користувацького інтерфейсу, включаючи кнопки, текстові поля, списки та інші елементи, що підвищують зручність взаємодії користувача з грою;

– **IL2CPP** (проміжна мова для C++) : це транслятор, який потрібен для конвертації C# коду у вихідний код мови C++. Дозволяє Unity отримати переваги швидкості та оптимізації, які зазвичай пов'язані з використанням нативних мов програмування.

Вищевказані інструменти надають широкі можливості і творчий простір при розробці ігрових проектів.

2.3 Візуальний стиль

Основний графічний стиль проекту базується на двовимірній графіці з чистим і простим інтерфейсом. Ця концепція була обрана з урахуванням цільової аудиторії – дітей віком від 7 до 16 років.

Колір є найбільш чутливим елементом у зоровому сприйнятті предметів людьми. Дослідження показують, що коли користувач стикається з цільовим об’єктом, колір привертає основну візуальну увагу протягом перших 20 секунд [3]. Колір може не лише справляти на людей візуальний вплив завдяки своєму сильному, інтуїтивно зрозумілому та яскравому вигляду, але й пробуджувати внутрішній контакт людей своїм справжнім, делікатним та багатим досвідом.

Також колір дозволяє людям швидше розпізнавати об’єкти, тобто на стадії чуттєвого сприйняття стимулу, і краще їх запам’ятовувати та розрізняти. Це може виявитися особливо корисним у грі, де візуальний інтерфейс та елементи відображення дуже важливі для гравця. Колірна палітра, яку використовується в грі, може вплинути на настрій гравця, підсилюючи емоційну взаємодію з світом. Таким чином, правильний вибір кольорів може не лише покращити загальний дизайн гри, але й підвищити її привабливість та ефективність щодо залучення гравців.

Для молодшої аудиторії важливо використовувати чіткі і яскраві кольори, уникаючи складних градієнтів та гострих кутів, що може бути менш привабливим та заплутаним для їхнього сприйняття [4]. Такий підхід не тільки спростить навігацію та взаємодію з грою, але і зробить процес навчання та адаптації до гри більш приємним і зрозумілим для юних гравців.

Деякі конкретні кольори можуть виражати певне культурне походження та особливості. Коли люди надають конкретне значення групі кольорів і продовжують використовувати їх на значному проміжку часу, зрештою це стане символічним кольором предмета. Це найчастіше зустрічається в релігії, стародавніх класових системах і різних звичаях. В даному випадку використовуються градієнти коричневого і сірого в змішуванні з яскравим червоним та синім (рис 2.2).

Коричневий в даному випадку дає відчуття індустріалізації, важкої техніки та робочих процесів, що добре комбінується з сеттингом гри. В той час як червоний і синій необхідні для того щоб показати гравцю прогресивність технологій в грі, надати певне відчуття футуризму і масштабу з порівнянні з його буденністю.

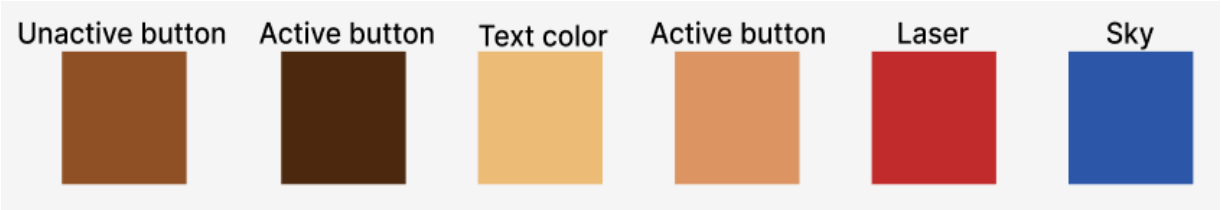


Рисунок 2.2 – Основна палітра кольорів для проекту.

Прості елементи були створені одразу в Figma. Що ж до складніших графічних елементів, то оскільки проект не є великим, графіка для нього буде створюватись за допомогою безкоштовних асетів з Asset Store, який вбудовано в Unity. Там присутній досить широкий асортимент доступних матеріалів, на які не поширюється авторське право.

У подальшому планується залучення 2D художників для розробки необхідних спрайтів та анімацій. Такий підхід дасть змогу ефективно використовувати доступні ресурси і забезпечить відповідну якість графічних елементів у проекті.

2.4 Мокап і архітектура

Мокап був створений інструментом для дизайну та прототипування Figma. Це відмінний інструмент, що надає широкий спектр можливостей для створення та редагування дизайну. Він включає різноманітні компоненти, стилі та шаблони, які дозволяють ефективно створювати та налаштовувати елементи інтерфейсу гри [5].

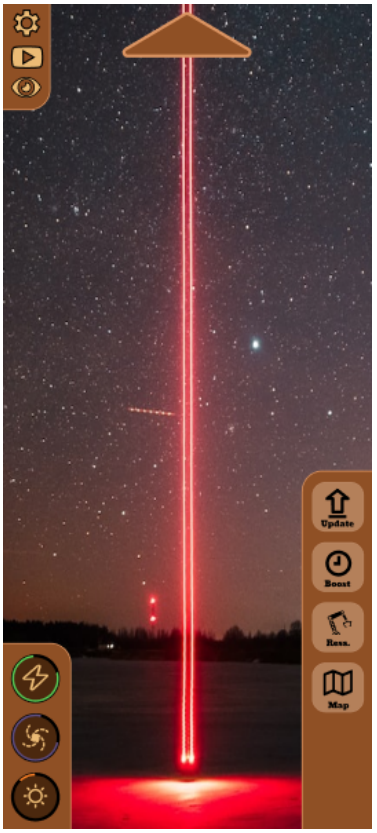


Рисунок 2.3 – Головний екран (макет в Figma)

В грі існує ряд внутрішніх екранів, між якими гравець може перемикатися за потребою, а також вікна, які відкриваються для конкретних дій або

інформації. Основний екран слугує центральною точкою взаємодії з гравцем, де він може спостерігати за своїм прогресом та здобутими ресурсами. У початкових етапах гри цей екран може здаватися досить простим, але по мірі прогресу гравця він буде активно наповнюватися новими об’єктами, анімаціями та ефектами, які відображатимуть активність гравця та його досягнення (рис 2.3). Така стратегія підтримує динаміку гри та підвищує зацікавленість гравця у подальшому розвитку.

З основного екрану гравець може переходити на сусідні вікна, а також відкривати підменю.

З цього екрану гравець може відкривати такі вікна:

– налаштування, де гравець має можливість персоналізувати ігровий досвід під свої індивідуальні уподобання. Тут можна налаштувати звукові ефекти, обрати мову інтерфейсу та звертатися до допоміжних зовнішніх ресурсів для отримання додаткової інформації (рис 2.4);

– ресурси, екран де гравець може детально ознайомитися зі всіма ресурсами, які він зібрав протягом гри, а також отримати інформацію про темпи їх накопичення за певний період часу;

– прогрес, яке є ключовим елементом для розвитку гравця, де він може використовувати накопичені ресурси для підвищення продуктивності виробництва та відкриття нового контенту. Цей розділ сприяє активному взаємодії гравця з грою та мотивує до досягнення нових висот;

– магазин, виступає як основний пункт монетизації, де гравець може придбати цінні ресурси або прискорити процес виробництва за допомогою реальних грошей, що дозволяє гравцям отримати значний прогрес у грі за короткий проміжок часу.

У майбутньому планується розширення ігрового інтерфейсу за допомогою додаткових екранів, які призначені для покращення користувацького досвіду та розширення геймплею. Перш за все, буде введено екран "Мапа", який надасть можливість гравцям з легкістю переміщуватися між різними виробництвами та

ресурсами, дозволяючи оптимізувати свою стратегію та збільшити продуктивність.

"Історія" стане місцем для зберігання сюжетних повідомлень, які допоможуть гравцям краще розуміти контекст ігрового світу, поглибити взаємодію з персонажами та зробити ігровий процес більш насиченим та захоплюючим для гравця.

Крім цього, планується введення додаткових вікон та елементів інтерфейсу для урізноманітнення геймплею, що надасть гравцям більше можливостей для самовираження. Це включає в себе різноманітні бонуси, виклики, досягнення та інші елементи, які стимулюють активну взаємодію та зацікавленість гравців.

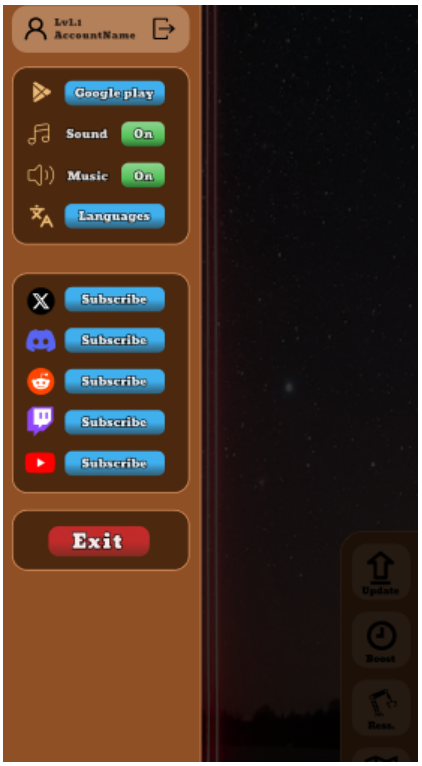


Рисунок 2.4 – Налаштування (макет в Figma)

2.5 Система ресурсів

В проекті великий акцент робиться на системі накопичення внутрішніх ресурсів, що є основою геймплею та визначає стратегічний аспект гри. Це

ключова механіка, яка впливає на всі аспекти гри, від її розвитку до взаємодії гравця з ігровим світом.

Основна концепція жанру Idle полягає в пасивному накопиченні ресурсів, яке відбувається автоматично навіть у тих випадках, коли гравець не активно взаємодіє з грою. Гравець продовжує збирати ресурси, навіть коли він вийде з гри, що створює відчуття постійного прогресу та розвитку.

Проте, щоб надихнути гравця активніше взаємодіяти з грою та не дозволити йому просто накопичувати ресурси безмежно, було встановлено максимальний об’єм збережених ресурсів. Коли додаток дійде до цього ліміту, внутрішні виробництва припиняють свою роботу до тих пір, поки гравець не повернеться до гри. Це стимулює гравця регулярно повертатися до гри, щоб оптимально використовувати свої ресурси та досягти максимального прогресу в ігровому світі.

В проекті основний акцент робиться на оптимізації та розвитку виробництва за допомогою зібраних ресурсів, які є ключовим елементом прогресу гравця. Гравець буде активно вдосконалювати свої виробничі механізми, зокрема збільшувати продуктивність виробництва, оптимізувати процеси та розвивати нові технології для максимізації накопичення ресурсів.

Крім того, гравцям буде надана можливість використовувати спеціальні помножувачі та конвертувати ресурси з одного підвиду в інший, що буде обов’язковим для подальшої прогресії на більш пізніх етапах гри.

Ця механіка взаємодії з ресурсами не лише стимулює гравця до постійного розвитку та оптимізації своїх виробничих потужностей, але й забезпечує зацікавленість гравця завдяки комплексності механік та необхідності приймати осмислені рішення.

Такий підхід до геймплею мотивує гравця витрачати більше часу в додатку, досліджуючи нові можливості та досягаючи високих результатів у виробничому секторі гри.

2.6 Фонові розрахунки

У сучасних мобільних пристроях велика увага приділяється оптимізації енергоспоживання та ресурсозбереженню, що вимагає ефективного підходу до роботи програм в фоновому режимі, через буде не доцільно і складно реалізовувати справжню роботу додатку в фоновому режимі. Замість цього в проекті реалізовано систему псевдо фонової роботи, що дозволить створити ілюзію постійних розрахунків в користувача, не використовуючи при цьому ресурси пристрою без необхідності та в не належний час.

Замість того, щоб тримати всі процеси в активному режимі, додаток ретельно відслідковує момент, коли користувач закрив його. А саме запам’ятовується точна дата та час, коли додаток було вимкнено, ці дані беруться з внутрішніх даних телефону. При наступному запуску додатка, вбудований скрипт автоматично розраховує, скільки ресурсів мало б бути накопичено за час відсутності користувача. Після цього ці ресурси будуть автоматично додані до рахунку гравця, забезпечуючи йому відчуття прогресу та досягнення цілей, навіть у випадках, коли гравець не активно взаємодіє з грою.

При запуску обов’язково виводиться вікно з інформацією про зараховані ресурси. Такий підхід допомагає стимулювати гравця до повернення в додаток, надаючи йому чітке розуміння того, які результати він отримав під час своєї відсутності. Виведення повідомлення з кількістю накопичених ресурсів під час відсутності користувача є важливою частиною стимулюючого механізму в жанрі Idle, який сприяє збереженню зацікавленості гравця та підтримці його активного користування додатком.

Висновки до розділу 2

На основі проведеного аналізу, детальної розробки мокапу, кольорової гами та вибору інструментів, було розроблено план для створення мобільного додатку. Мокап був важливими кроком в розробці, оскільки це дозволило не

лише візуалізувати кінцевий продукт, але й створити зрозумілий та інтуїтивний дизайн користувацького інтерфейсу, що сприяє покращенню взаємодії з програмою. Вибір Unity для розробки додатку має вирішальне значення, оскільки цей інструмент дозволяє значно скоротити час, потрібний для впровадження програми в експлуатацію.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Розгортання середовища Unity

Для початку розробки мобільної гри, необхідно перш за все встановити все необхідне програмне забезпечення, в якому будуть реалізовуватись всі ідеї і буде проводитись тестування.

Visual studio code було завантажено з офіційного веб сайту [6]. Те саме було зроблено з Unity[7].

Після встановлення всього необхідного програмного забезпечення, можна приступити до створення нового проекту (рис. 3.1), а також завантажити додаткові розширення та інструменти.

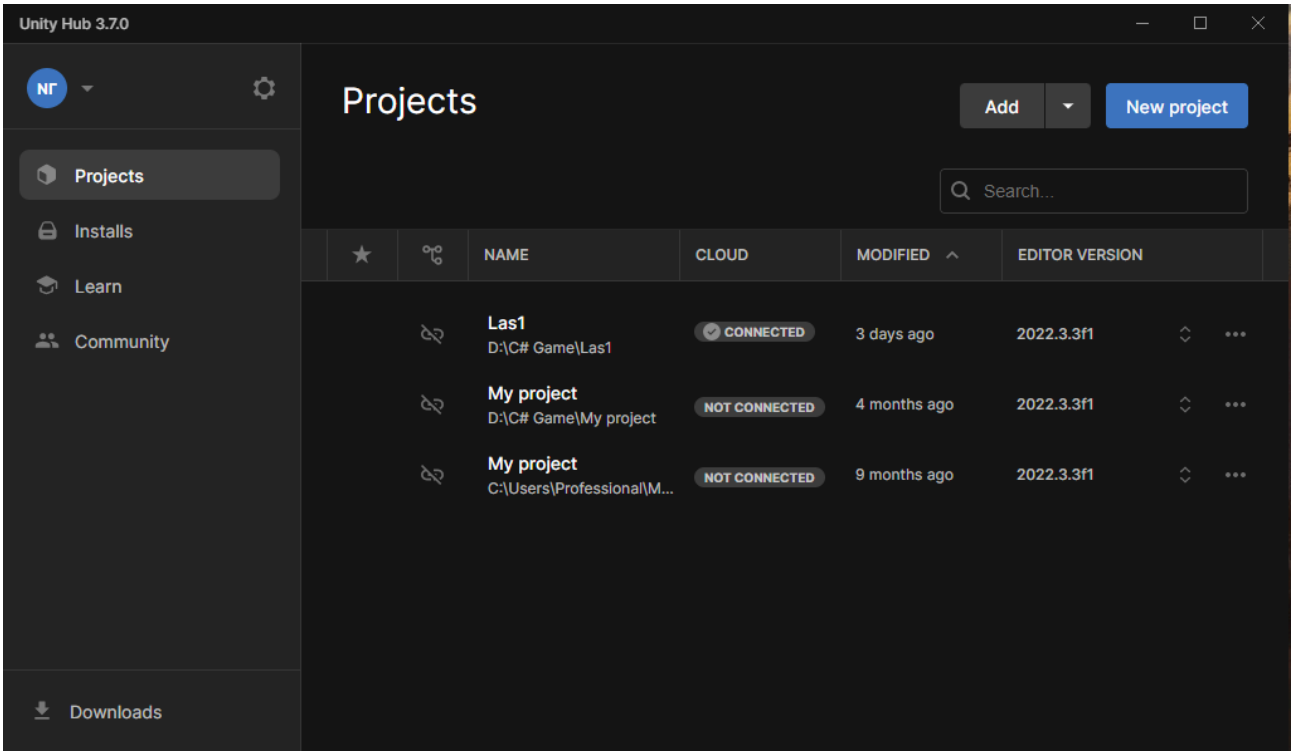


Рисунок 3.1 – Вікно створення проектів

Завершивши створення нового проекту, необхідно провести важливі модифікації над рушієм, задля того щоб проект був спроможним до запуску на

мобільних пристроях. Для цього потрібно налаштувати сам проект як Android додаток(рис. 3.2), а також надати необхідний мінімум даних про розробку, як то робоча назва продукту, назва компанії, і поточна версія. Також гостро необхідно вказати розміри і характеристики для камери, так як буде неможливо запустити додаток на мобільному пристрої, якщо весь інтерфейс буде підлаштовано для персональних комп'ютерів.

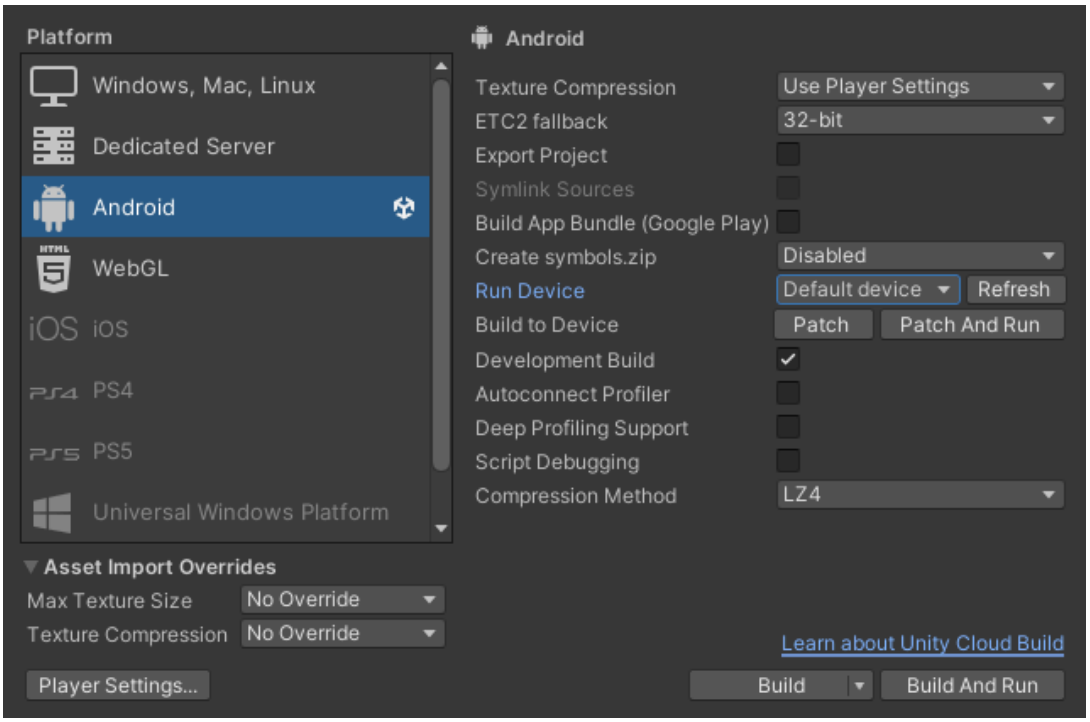


Рисунок 3.2 – Модифікація проекту під мобільні пристрої.

Далі можна приступити до встановлення додаткових інструментів, призначених для спрощення розробки, і зменшення часових витрат. Доцільним є використання ProBuilder, в комбінації з ProGrids. ProBuilder являється простим інструментом для моделювання об'єктів [8]. Він дозволяє створювати кастомні фігури і площини, які не можна створити з допомогою стандартних інструментів Unity, керувати їх відображенням та оптимізацією. По своїй суті цей інструмент є спрощеною версією Blender, і хоча його можливості сильно уступають оригінальному продукту, в цьому заключається його основний плюс. Такі фактори як відсутність необхідності перемикатись між різними

середовищами, заморочуватись з перенесенням моделей, а також значно простіший відносно оригіналу інтерфейс, роблять цей інструмент не просто наглою копією оригіналу, але валідним способом спростити розробнику життя, а також зменшити витрати бюджету на 2д і 3д художників.

ProGrids не є настільки ж важливим в розробці інструментом, але він також вносить свій вклад в розробку, додаючи до проекту магнітну сітку, до якої автоматично кріпляться найближчі активні об’єкти. Завдяки цьому відпадає необхідність самостійно вирівнювати їх по просторовим площинам, що сильно економить час розробки.

По завершенню встановлення всіх необхідний інструментів та розширень, можна приступити до розробки проекту.

3.2 Експорт дизайн елементів з Figma в проект

Після того як Unity був підготовлений до роботи, можна приступити до впровадження базового інтерфейсу, і створення робочого макету прямо в середовищі розробки. Завдяки цьому відпаде необхідність постійного перемикання на Мокап, і можна буде приступити до розробки програмного коду. Оскільки макет для цієї роботи вже був попередньо створений в Figma (рис. 3.3), було проведено його поступове перенесення в Unity. Це сильно прискорить розробку базової версії проекту, адже завдяки можливості використовувати вже створені асети, відпаде необхідність в використанні сторонніх ресурсів, а також можна зекономити на дорогих художниках.

Для того щоб перенести елемент з Figma в Unity, необхідно відокремити необхідний елемент від інших, з допомогою тимчасового вирізання з оригінального макету. Після чого використовується функція “Export”, яка завантажує виділений об’єкт на ПК користувача в вигляді PNG файлу. Масштаб при цьому задається 1.5, так як це буде найближчим до того, що є необхідним в проекті. Завантажений PNG файл необхідно перенести в завчасно відкритий проект, в вкладку асети, або також можна перенести файл напряму в папку

проекту. Хоча і можна зберігати всі файли прямо в папці “Асети”, в подальшому це призведе до переповнення цієї папки, через що потім буде значно важче знаходити необхідні деталі і макети.

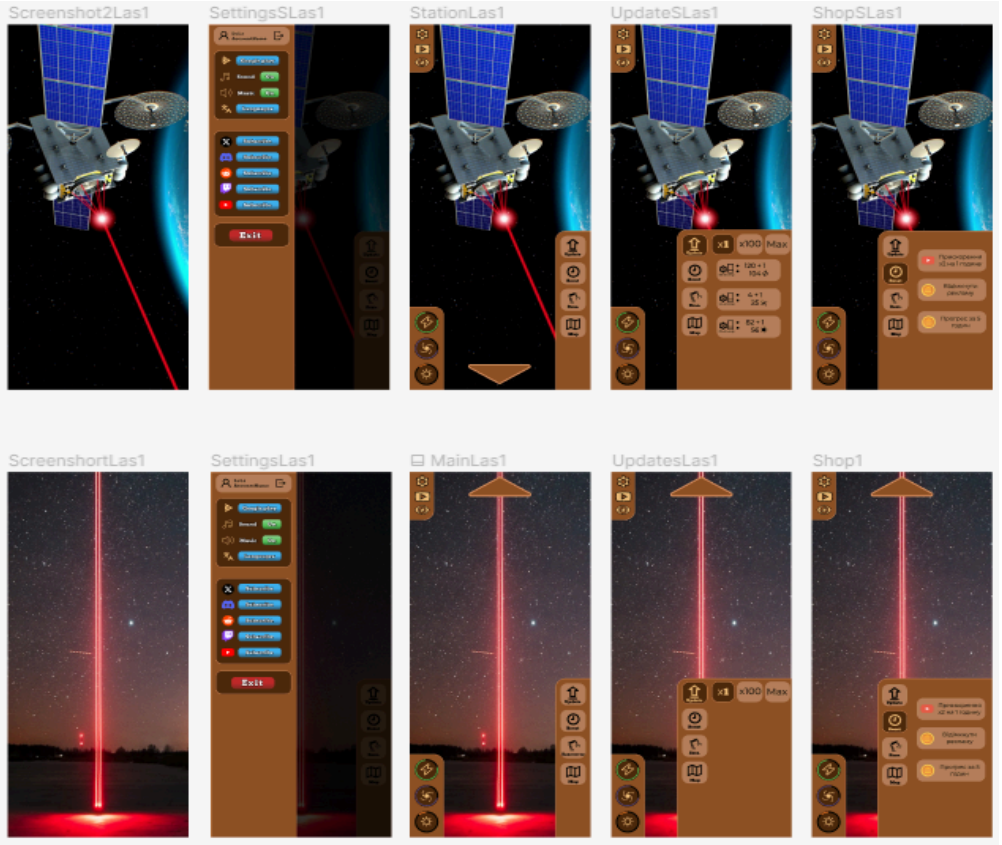


Рисунок 3.3 – Макет проекту в Figma.

В проектах розробки важливо притримуватись організованості, і зберігати файли не разом в одній папці, а розділено, згідно типу і призначення, як Scenes, в якій зберігаються збережені сцени, Scripts, для збережених скриптів, і Materials, в якій зберігаються робочі матеріали, зокрема перенесені з Figma, точніше ці матеріали зберігаються в папці всередині цієї папки, на випадок якщо виникне потреба використовувати матеріали з інших джерел. Дуже важливо при роботі над проектами тримати своє середовище в чистому і організованому стані, оскільки це сильно прискорює роботу на пізніх стадіях розробки, а також надзвичайно сильно спрощує роботу в випадку якщо до проекту залучено групу з чисельних розробників, як частіше всього і буває в

випадку з розробкою відео ігрових продуктів, як мобільних так і для персональних комп’ютерів.

3.3 Рухомі елементи

Після перенесення всього необхідного з зовнішніх джерел в Unity, можна приступити до роботи над програмною частиною додатку. Було вирішено розпочати з розробки системи рухомих панелей, які гравець буде здатен відкривати і закривати з допомогою натискань на екран телефону. Важливо пам’ятати, що цей проект цілком розробляється як мобільний додаток, тому всі дії користувача повинні проводитись сугубо натисканнями на екран, без використання клавіатури чи інших зовнішніх пристроїв.

Для того щоб реалізувати функцію рухомих вікон, потрібно завчасно створити в середовищі декілька об’єктів, які виступатимуть посередниками між діями гравця та кодом. Спершу створюється порожній об’єкт – менеджер, до якого прив’язується скрипт в вигляді компоненти, і вже до цього скрипта будуть прив’язуватись об'єкти в середовищі. Також необхідно добавити на сцену канвас, на якому будуть показані виведенні вікна. Створеному вікну обов’язково необхідно присвоїти властивість “RectTransform”, цей компонент відповідає за просторове розміщення і відображення UI елементів на екрані гравця [9] і є динамічним при масштабуванні проекту до різних розширень екрану. Саме вікно виноситься за межі канвасу, але якорі компоненти Rect Transform залишаються на екрані, в тій області, де в подальшому планується бачити вікно в відкритому стані. Створюються дві кнопки, одна з них буде знаходитись на канвасі, для відкривання вікна, інша ж буде знаходитись на площині самого вікна, і буде його зачиняти.

Можна приступити до написання коду, який буде виконувати переміщення вікна при натисканні на кнопки. Потрібно зробити об’явлення всіх необхідних об'єктів системи:

```
    public Canvas canvas;
    public GameObject OpenSettingsBtn;
    public GameObject EnergyText;
    public GameObject HideInterfaceButton;
    public RectTransform Panel;
    public float CanvasWidth;
```

Після чого в методі Start при початку роботи додатку ми дізнаємось ширину канвасу, щоб пізніше можна було по цій ширині динамічно виставляти панелі:

```
public void Start(){
CanvasWidth =
canvas.gameObject.GetComponent<RectTransform>().rect.width;
    }
```

Далі створюється метод відкривання панелі. Цей метод буде викликатись натисканням на кнопку в рушії, тому немає необхідності в додаткових викликах методу в коді:

```
    public void OpenPanel(){
        Panel.anchoredPosition = new Vector2(0, 0);
        OpenSettingsBtn.SetActive(false);
        EnergyText.SetActive(false);
        HideInterfaceButton.SetActive(false);
```

Закривання панелі робиться подібним чином, але сама кнопка закривання вікна буде знаходитись на самому вікні:

```
    public void ClosePanel(){
        Panel.anchoredPosition = new Vector2(CanvasWidth, 0);
        OpenSettingsBtn.SetActive(true);
        EnergyText.SetActive(true);
        HideInterfaceButton.SetActive(true);
    }
```

Також, оскільки тут робиться логіка роботи з відкриванням і закриванням вікна, тут також добавлено функцію закривання всього додатку, простим однорядковим методоми:

```
public void ExitTheGame(){
    Application.Quit();
}
```

Стан об'єктів часто змінюється методом SetActive(), оскільки це є найпростішим способом як швидко добавити або прибрати об’єкт зі сцени. Панель буде переміщуватись по горизонталі з допомогою модифікації параметру Vector2, і сила цього переміщення буде розраховуватись динамічно, адже при цьому буде враховуватись ширина канвасу з змінної CanvasWidth.

Після створення коду, необхідно в менеджері прив’язати до нього всі необхідні об’єкти, такі як кнопки і канвас (рис. 3.4).

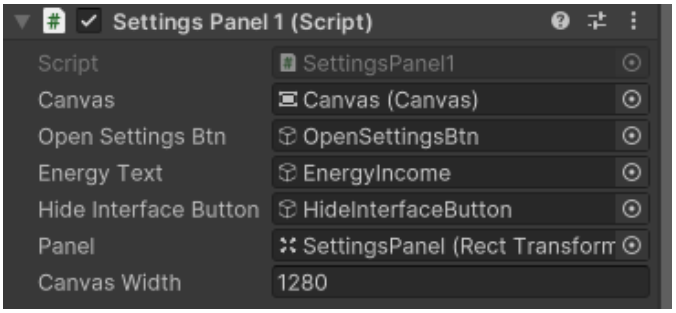


Рисунок 3.4 – Заповнені дані в скрипті

Цього буде достатньо щоб рухомі панелі справно працювали, а сам код є достатньо універсальним для того щоб одночасно задовольнити потреби декількох панелей.

3.4 Система розрахунків

Після впровадження базового інтерфейсу, можна приступити до розробки найбільш основної частини всього додатку, а саме системи нарахування очок.

Для роботи цієї системи було створено основний скрипт для гри під назвою “Game Manager”. Створивши порожній скрипт, можна почати наповнювати його логікою по нарахуванню очок.

Спершу створюються змінні, які будуть модифіковані подальшим кодом. Загалом використовуються два типи доступу до змінних. а саме public та [SerializeField] private. Різні методи потрібні для дотримання правил інкапсуляції в ООП [10]. Так якщо метод є приватним, то на нього не можна буде впливати іншим скриптом, що робить код більш надійним. Від звичних об’явлень ці змінні відрізняються використанням [SerializeField], це особливий модифікатор для Unity, який дозволяє взаємодіяти з приватною змінною в середовищі розробки, прив’язувати об’єкти та змінювати значення полів [11].

Назвою для основного ресурсу буде “energy”. а “cursors” буде значенням для нарахування енергії в секунду. Також добавлено поле “additiveHits”, для нарахування додаткової енергії при натисканні на клавіші, та поля для відображення цін покращень:

```
public class GameManager : MonoBehaviour
{   public Text EnergyText;
    public Text EnergyIncomeText;
    public Text BuyCursorText;
    public Text BuyHitText;
    [SerializeField] private float energy;
    [SerializeField] private int cursors;
    [SerializeField] private int cursorPrice;
    [SerializeField] private int additiveHits;
    [SerializeField] private int hitsPrice;}
```

Слід виділити, що змінна energy записується як float, хоча всі інші інші змінні використовують тип int. Справа в тому, що гейм дизайн Idle ігор передбачає роботу зі справді великими числами, такими як мільйони і мільярди, в той час як тип int не може перейти значення в 2,147,483,647 [12]. Тому замість використання int використовується float. Також можливим є використання типу даних double, який володіє ширшим за float діапазоном даних, але в процесі

розробки від double прийшлось відмовитись, по причині того що його збереження не підтримується в функції PlayerPrefs. що є критично важливим для роботи даного додатку.

При впровадженні логіки програми, постійно будуть використовуватись методи. Крім звичних переваг використання методів, як то можливість перевикористання коду і підвищення читабельності, використання методів є необхідним через те що методи можна просто викликати в середовищі юніті, з допомогою натискання на кнопки, або через базові функції рушія.

Першим способом здобути очки енергії буде натискання на кнопку, яка даватиме певну кількість очок за натискання:

```
public void EnergyClicked()
    {
        energy++;
        energy += additiveHits;
        SaveData();
        UpdateText();
    }
```

Але згідно заданого гейм дизайну, дане значення повинно змінюватись регулярно від дій гравця. Тому у внутрішньому магазині покращень, який знаходиться на окремому відкриваємому вікні, буде можливість збільшити здобування енергії за одне натискання:

```
public void BuyHits()
    {
        if (energy < hitsPrice) return;
        energy -= hitsPrice;
       additiveHits += 5;
        hitsPrice *= 2;
        SaveData();
        UpdateText();
    }
```

Проводиться перевірка, чи доступної гравцю енергії достатньо для придбання покращення. Якщо енергії недостатньо, метод припиняє виконання командою return. Якщо ж збереженої енергії достатньо, від неї віднімається ціна за покращення.

Далі залишається тільки змінити ціну за покращення, адже якщо ціна не буде змінюватись, то вже при четвертій покупці, гравець буде здатен безкінечно закуповувати дане покращення. Також потрібно при кожній взаємодії з внутрішніми ресурсами зберігати дані, і змінювати текст на екрані гравця, що робиться окремими методами.

Після цього можна приступити до функції автоматичного нарахування очок, поки що в присутності гравця. Реалізується метод закупівлі курсору, подібно до попереднього методу:

```
public void BuyCursor() {
        if (cursorPrice < 10 ) {cursorPrice = 10;}
        if (energy < cursorPrice) return;
        energy -= cursorPrice;
        cursors += 1;
        cursorPrice *= 2;
        SaveData();
        UpdateText();
    }
```

Далі реалізується сама система нарахування енергії в секунду, в окремому методі. Цей метод починає свою роботу не при нажиманні на кнопку, як інші методи, а працює постійно через створену на початку скрипту функцію InvokeRepeating(nameof(PayIncome), 1f, 1f). це метод у Unity, який дозволяє викликати функцію через певні інтервали, починаючи після затримки [13]. Цей метод зручний для реалізації повторюваних завдань, таких як оновлення стану, генерація об'єктів або перевірка умов через регулярні проміжки часу:

```
    private void PayIncome()
    {
```

```
        energy += cursors;
        SaveData();
        UpdateText();
    }
```

В цьому методі також виконується робота з анімаціями, а саме для кращого візуального відображення прогресу гравця вмикаються і вимикаються елементи об'єктів анімацій:

```
 if (cursors > 0 & cursors < 5)
        {
            WeakConstLazer.SetActive(true);
        }
        if (cursors >= 5)
        {
            WeakConstLazer.SetActive(false);
            StrongConstLazer.SetActive(true);
        }
```

На цьому закінчуються розрахунки набутих балів, залишається тільки вивести їх на екран. Краще всього буде це зробити виведенням тексту в попередньо підготовлений текстовий об’єкт:

```
private void UpdateText()
    {
        EnergyText.text = $"{energy} energy";
        EnergyIncomeText.text =$"+{cursors} energy/second";
        BuyCursorText.text = $"Generator (+1 e/s) {cursorPrice}
energy";
        BuyHitText.text = $"Impulses = (+5 energy/click) {hitsPrice}
energy";
    }
```

Текст задається напряму в скрипті, а не в середовищі Unity. Це необхідно для того щоб текст був динамічним, і міг відображати такі актуальні дані як ціну за покупку і набуті ресурси правильно. Тут також реалізована система

скорочення та округлення чисел з допомогою методу Math.Round() [14], яка необхідна для простішого читання на пізніх етапах гри. Так, якщо показати гравцю мільйон енергії, це буде важко прочитати, а також такі числа буде важко помістити на екран. Тому над ними проводиться ділення, після чого вони округлюються і показуються в більш зрозумілому вигляді:

```
if (energy < 1000){
    EnergyText.text = $"{energy} energy";}
else if (energy < 1000000){
    convertuedEnergy = energy / 1000;
    showEnergy = (int)Math.Round(convertuedEnergy);
    EnergyText.text = $"{showEnergy} K energy";}
else if (energy < 1000000000){
    convertuedEnergy = energy / 1000000;
    showEnergy = (int)Math.Round(convertuedEnergy);
    EnergyText.text = $"{showEnergy} M energy";}
```

Округленню піддається тільки окрема змінна showEnergy, а не сам елемент energy, це зроблено тому що якщо піддавати округлення сам параметр energy, то буде відбуватись втрата даних, що негативно скажеться на всьому проекті.

3.5 Офлайн нарахування

Всі дії, які необхідно виконати при запуску програми слід писати в методі private void Start(). Він виконується в обов’язковому порядку перед запуском першого кадру методом Update() [15]. Для цього також можна використати метод Awake(), але Start() є більш динамічним, тому використання цього методу вважається більш доцільним.

Для роботи системи збереження ресурсів перш за все потрібно при запуску додатку завантажити з пам’яті пристрою всі ті дані, які було збережено попереднього разу. для цього використовується метод PlayerPrefs.GetInt(). У випадку якщо це перший запуск, дані залишаються порожніми:

```
private void Start()
    {
        energy = PlayerPrefs.GetInt("Energy", 0);
        cursors = PlayerPrefs.GetInt("Cursors", 0);
        additiveHits = PlayerPrefs.GetInt("AdditiveHits", 0);
        cursorPrice = PlayerPrefs.GetInt("CursorPrice", 0);
    }
```

Після того як з пам’яті пристрою було вилучено всі необхідні дані, розпочинається розрахунок пройденого часу, і нарахування здобутих ресурсів.

Для того щоб знати скільки пройшло часу з попереднього запуску, потрібно дізнатись який зараз точний час. Можна виділити два способи як додаток може дізнатись точний час. Першим способом буде використання API спеціалізованих сервісів[16], що є надійним, але потребує наявність інтернет підключення, що не є бажаним в даному випадку, адже гра загалом базується як офлайн застосунок. Другим способом, який і використовується в даному випадку, є використання вбудованого в C# методу System.DateTime.UtcNow, який здобуває дані про поточний час з пристрою, на якому був запущений додаток[17]. Цей спосіб не потребує інтернет підключення, і являється найшвидшим та ефективним способом дізнатись точний час.

Хоча метод System.DateTime.UtcNow і є вельми корисним і ефективним, в цього способу є один головний мінус, який можна виправити тільки використанням онлайн сервісів, а саме можливіть гравця обійти необхідність очікування не надто чесним способом. Гравець може особисто змінити поточний час на додатку, відключивши функцію автоматичного оприділення часу на пристрої[18]. Хоча це і не є значною проблемою в наш час, адже більшість користувачів ніколи не користувались функцією ручного перемикання часу, в майбутньому, при додаванні в додаток функцій, потребуючих онлайн підключення, буде доцільно змінити цю систему на використання API, замість внутрішнього часу пристрою.

Після здобуття всіх необхідних даних, а зокрема дат, потрібно дізнатись кількість пройденого часу. Це можна вирахувати, якщо відняти від теперішньої точної дати та часу момент точної дати попереднього збереження. Після чого ці дані слід запарсити, адже вони зберігаються не в int форматі. Далі необхідно перетворити складне числове значення дати в секундне число, з якими буде значно простіше працювати. Це робиться простим множенням, спочатку на 7, тому що в тижні 7 днів, потім на 24, тому що в одному дні 24 години, потім на 60 хвилин в годині, і 60 секунд в хвилині. Так ми здобуваємо єдине значення для пройденого часу в секундах, після чого залишається тільки використати формулу, по якій нараховуються очки в онлайн режимі. в даному випадку вираховуємо получену енергію множенням секунд які пройшли на кількість здобуваємих за секунду очок:

```
private void Start(){
        DateTime lastSaveTime = Utils.GetDateTime("LastSaveTime",
DateTime.UtcNow);
        TimeSpan timePassed = DateTime.UtcNow - lastSaveTime;
        int secondsPassed = (int)timePassed.TotalSeconds;
        secondsPassed = Mathf.Clamp(secondsPassed, 0, 7 * 24 * 60 *
60);
        energy += secondsPassed * cursors;
        UpdateText();
        InvokeRepeating(nameof(PayIncome), 1f, 1f);
    }
```

Власне збереження всіх даних проходить в методі SaveData(), який запускається після кожної взаємодії додатку з ігровими ресурсами:

```
private void SaveData(){
        PlayerPrefs.SetInt("Energy", energy);
        PlayerPrefs.SetInt("Cursors", cursors);
        PlayerPrefs.SetInt("AdditiveHits", additiveHits);
        PlayerPrefs.SetInt("CursorPrice", cursorPrice);
        Utils.SetDateTime("LastSaveTime", DateTime.UtcNow);
    }
```

PlayerPrefs – це механізм збереження даних у вигляді пар "ключ-значення" у Unity. Він дозволяє зберігати прості дані, такі як числа, рядки або булеві значення, між сеансами гри або навіть після завершення програми. Ці дані зберігаються локально на пристрої гравця і можуть бути використані для збереження будь яких простих значень. Основна перевага використання PlayerPrefs полягає у простоті використання. Для збереження даних потрібно лише вказати ключ і значення, яке потрібно зберегти, і Unity буде самостійно утримувати ці дані на пристрої гравця[19]. Також PlayerPrefs досить ефективно працює з невеликими обсягами даних, що робить його ідеальним для збереження простих налаштувань гри або іншої невеликої інформації. Але потрібно пам’ятати, що PlayerPrefs не підходить для зберігання великих даних або конфіденційної інформації.

Також не можна забувати про можливість зберігати тільки певні типи даних, як то float або int, але не double [20].

Наостанок слід пояснити такі методи як SetDateTime та GetDateTime. Дані методи відповідають за збереження даних про час, щоб пізніше ці дані міг використати метод UtcNow(), необхідний для нарахування ресурсів при запуску додатку, враховуючи пройдений час:

```
public static void SetDateTime(string key, DateTime value){
        string convertedToString = value.ToString("u",
CultureInfo.InvariantCulture);
        PlayerPrefs.SetString(key, convertedToString);}
public static DateTime GetDateTime(string key, DateTime defaultValue){
        if (PlayerPrefs.HasKey(key)){
            string stored = PlayerPrefs.GetString(key);
            DateTime result = DateTime.ParseExact(stored, "u",
CultureInfo.InvariantCulture);
            return result;}
        else {return defaultValue;}
```

Метод SetDateTime дозволяє зберігати об'єкти DateTime у PlayerPrefs, перетворюючи їх у рядок за допомогою формату "u". В той час як метод GetDateTime дозволяє отримувати об'єкти DateTime з PlayerPrefs, перевіряючи наявність ключа і перетворюючи збережений рядок назад у об'єкт DateTime. Якщо ключ не існує, повертається значення за замовчуванням. Загалом можна сказати що ці методи є необхідними для правильної роботи UtcNow. Ці дані необхідно неодноразово перетворювати в інші типи даних, тому як метод UtcNow може зберігати тільки певні типи даних, як то float або int, але ні в якому разі не double.

Висновки до розділу 3

В даному розділі було детально описано кроки розробки відеогри для мобільних телефонів в жанрі Idle, а саме робота з рушієм юніті, та кодом на мову програмування C#. Початковий етап включав створення проекту в Unity, де було додано базовий інтерфейс, що забезпечує взаємодію користувача з грою. Було реалізовано основний інтерфейс користувача, включаючи головне меню, екран ресурсів та прогресу. де гравець може споглядати здобуті ресурси та результати прогресії. Було додано функції відкривання вікон, які дозволяють гравцю переміщатися між різними розділами гри, включаючи основний екран, вікно прогресу та вікно налаштувань. Розроблено систему нарахування очок енергії, яка працює як за натисканням на кнопку, так і в автоматичному режимі. Було впроваджено можливість здобувати ресурси на основі пройденого часу, навіть якщо додаток був вимкнений. Ця функція реалізована за допомогою PlayerPrefs, який зберігає дату і час останнього виходу з гри. Розроблено функцію роботи з великими числами, яка включає їх використання та конвертацію в інші типи даних.

ВИСНОВКИ

В рамках дослідження було ретельно вивчено предметну область проекту, яка включає розробку мобільного додатку відеогри в жанрі Idle, з допомогою технологій Unity, на мові програмування C#. Дослідження чотирьох існуючих аналогів показало їх недоліки, серед яких надлишок рекламних інтеграцій, та занадто агресивна монетизація.

При виборі інструментів для розробки додатку було враховано їхню доступність, надійність та популярність, що дозволило створити стабільно працюючий ігровий додаток. Рушій Unity зробив розробку швидкою та ефективною, в той час як використання Unity Assets в комбінації з Figma задовільнило потреби в графічних елементах.

Було створено систему прогресії з використанням функцій Player Prefs, з можливістю накопичення ігрових ресурсів без необхідності тримати додаток відкритим. Реалізовано систему роботи з великими числами, та їх динамічне відображення. З інтерфейсом можна взаємодіяти, він є простим і зрозумілим.

В результаті, було створено повноцінний ігровий додаток для мобільних пристроїв. що здатен виконувати поставлені задачі по розважанню та заохочуванню користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Спорт, театр і розваги в стародавньому світі. atiner. Веб-книга. URL: https://atiner.gr/2020/Pappas.pdf (дата звернення 18.03.2024)

2. Жанр Idle. Wikipedia. Веб-сайт. URL: https://en.wikipedia.org/wiki/Incremental_game (дата звернення 20.03.2024)

3. Як колір привертає увагу genius.space. Веб-сайт. URL: https://genius.space/lab/psihologiya-koloriv-u-veb-dizajni-yak-vibrati-kolirnu-palitru-shhob-pidvishhiti-konversiyu-sajtiv/ (дата звернення22.03.2024)

4. Дизайн інтерфейсу для молоді researchgate. веб-сайт. URL:https://www.researchgate.net/publication/354131974_Color_design_in_application_interfaces_for_children (дата звернення 23.03.2024)

5. Особливості Figma help.figma. веб-сайт URL:https://help.figma.com/hc/en-us/categories/360002051613-Get-started (дата звернення 24.03.2024)

6. Завантаження Visual Studio Code code.visualstudio. веб-сайт. URL: https://code.visualstudio.com/ (дата звернення 24.03.2024)

7. Завантаження Unity store.unity. веб-сайт. URL: https://store.unity.com/download (дата звернення 29.03.2024)

8. ProBuilder unity. веб-сайт. URL: https://unity.com/features/probuilder (дата звернення 12.04.2024)

9. Документація RectTransform docs. unity3d. веб-сайт. URL: https://docs.unity3d.com/ScriptReference/RectTransform.html (дата звернення 26.02.2024)

10. Основні принципи ООП training.epam. веб-сайт. URL: https://training.epam.ua/ua/blog/275 (дата звернення 13.04.2024)

11. Документація SerializeField docs.unity3d. веб-сайт. URL: https://docs.unity3d.com/ScriptReference/SerializeField.html (дата звернення 15.04.2024)

12. Розмір типів даних learn.microsoft.веб-сайт. URL: https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2008/cs7y5x0x(v=vs.90)?redirectedfrom=MSDN (дата звернення 18.04.2024)

13. Документація InvokeRepeating docs.unity3d. веб-сайт. URL: https://docs.unity3d.com/ScriptReference/MonoBehaviour.InvokeRepeating.html (дата звернення 06.05.2024)

14. Документація Math.Round() docs.unity3d. веб-сайт. URL: https://docs.unity3d.com/ScriptReference/Mathf.Round.html (дата звернення 07.05.2024)

15. Документація MonoBehaviour.Start() docs.unity3d. веб-сайт. URL: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html#:~:text=Start%20is%20called%20on%20the,not%20the%20script%20is%20enabled (дата звернення 10.05.2024)

16. Завантаження часу і дати з інтернету forum.unity. веб-сайт. URL: https://forum.unity.com/threads/retrieve-date-and-time-from-the-internet.441144/ (дата звернення 10.05.2024)

17. Завантаження часу і дати з пристрою learn.microsoft. веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/api/system.datetime.utcnow?view=net-8.0 (дата звернення 10.05.2024)

18. Що станеться якщо змінити час на телефоні reddit. веб-сайт. URL: https://www.reddit.com/r/foshelter/comments/3mvcuh/what_happens_when_change_your_phones_clock_to/ (дата звернення 12.05.2024)

19. Принцип роботи PlayerPrefs bootcamp. веб-сайт. URL: https://bootcamp.uxdesign.cc/unity-feature-101-basic-saving-using-playerprefs-2fc737d1ac7b (дата звернення 12.05.2024)

20. Документація PlayerPrefs unity3D. веб-сайт. URL: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html#:~:text=PlayerPrefs%20is%20a%20class%20that,into%20the%20user's%20platform%20registry (дата звернення 13.05.2024)

метадані

Заголовок

Розробка мобільної гри в жанрі “Idle”

Автор
Гаврило А.

Науковий керівник / Експерт
кандидат технічних наук Сергій Ващишак

підрозділ
King Danylo University

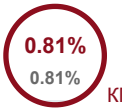
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	4

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КП 2

25
Довжина фрази для коефіцієнта подібності 2

8488
Кількість слів

64168
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://krs.chmnu.edu.ua/jspui/bitstream/123456789/1916/1/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B5%D1%84%D0%B5%D1%80%D0%B0%D1%82%20%D0%92%D1%80%D1%83%D1%87%D0%B5%D0%B2%D0%B8%D1%87%20%D0%9C.%20%D0%90..pdf	24	0.28 %
2	Розробка веб-сайту агрегатора новин з елементами веб-скрапінгу [redacted] 6/12/2023 King Danylo University (King Danylo University)	12	0.14 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/396/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D0%A1%D1%82%D1%80%D1%96%D0%BB%D0%B5%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1	11	0.13 %