МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка ігрового додатку для пристроїв під ОС Android з використанням міжплатформного середовища Unity

Виконав студент 2 курсу групи МІС-21
спеціальності 122 Комп'ютерні науки
Мірієв Мірсахіб Мірішевич

Керівник к.т.н., доцент
Фразе-Фразенко Олексій Олексійович

Рецензент Начальник ЦІТ ОНЕУ
к.т.н., Домаскін О.М.

Одеса 2022

АНОТАЦІЯ

на магістерську кваліфікаційну роботу
«Розробка ігрового додатку для пристроїв під ОС Android з використанням міжплатформного середовища Unity»,
студента
Мірієва Мірсахіба Мірішевича

Актуальність теми магістерської кваліфікаційної роботи обумовлюється необхідністю розробки зрозумілого інтерфейсу, зручного та простого у використанні ігрового застосунку з індивідуальним сюжетом.

Мета роботи – аналіз методів, засобів та технологій розробки ігрового застосунку для пристроїв під ОС Android.

Об'єкт дослідження – ігрові міжплатформні середовища та засоби розробки графічних ігрових застосунків.

Предмет дослідження – технології створення ігрового застосунку для пристроїв під ОС Android.

Методи дослідження – емпіричний метод із використанням експертних оцінок, сучасних програмних продуктів та технологій.

Перший розділ присвячений аналізу сучасних движків, що використовуються для розробки ігрових застосунків. В даному розділі, представлено опис game engines та проведено аналіз для вибору ігрового движка для подальшої розробки. У другому розділі проведено аналіз та здійснено опис основних етапів розробки/створення ігрового застосунку. Третій розділ описує аналіз використання під час розробки готового ігрового движка або ж самописного. В цьому розділі представлено переваги та недоліки кожного з варіантів движків. В четвертому розділі, описується хід проведення розробки ігрового застосунку під ОС Android. На початку описуються правила для даної гри, після чого описані етапи розробки.

Результатом роботи є розроблений за допомогою ігрового міжплатформного движка Unity мобільний ігровий застосунок під ОС Android. Даний

застосунок, простий у використанні, має зручний та зрозумілий інтерфейс для користувача. Ігровий застосунок, допоможе скоротати час під знаходження у черзі, пробці або ж просто під час обідньої перерви. Застосунок, не має обмежень по віку. Слід зауважити, що орієнтований він на людей з підвищеною концентрацію та увагою. Але, у випадку розсіяності – даний застосунок, добре тренує та розвиває концентрацію.

Магістерська кваліфікаційна робота містить 70 сторінок, 28 рисунки та 13 джерел посилань.

Ключові слова: ІГРОВИЙ ЗАСТОСУНОК, КОРИСТУВАЧ, ОПЕРАЦІЙНА СИСТЕМА, ANDROID, GAME DEVELOPMENT , GAME ENGINES, UNITY .

SUMMARY

for a master's degree

"Development of a game application for Android devices using the Unity cross-platform environment",

students of

Miriev Mirsahib Mirishevich

The relevance of the topic of the master's qualification work is determined by the need to develop a clear interface, a convenient and easy-to-use game application with an individual plot.

The purpose of the work is the analysis of methods, means and technologies of game application development for Android OS devices.

The object of research is game cross-platform environments and tools for the development of graphic game applications.

The subject of the study is the technology of creating a game application for Android devices.

Research methods – an empirical method using expert assessments, modern software products and technologies.

The first chapter is devoted to the analysis of modern engines used for the development of game applications. In this chapter, a description of game engines is presented and an analysis is carried out to select a game engine for further development. In the second chapter, an analysis was carried out and a description of the main stages of development/creation of a game application was carried out. The third chapter describes the analysis of use during the development of a ready-made game engine or a self-written game engine. This chapter presents the advantages and disadvantages of each engine option. The fourth chapter describes the process of developing a game application for the Android OS. At the beginning, the rules for this game are described, after which the stages of development are described.

The result of the work is a mobile game application for the Android OS, developed with the help of the Unity cross-platform game engine. This application, easy to use, has a convenient and understandable interface for the user. The game application will help to reduce the time spent in a queue, in a traffic jam or simply during a lunch break. The application has no age restrictions. It should be noted that it is aimed at people with increased concentration and attention. But, in the case of absent-mindedness, this application trains well and develops concentration.

The master's thesis contains 70 pages, 28 figures and 13 sources.

Keywords: ANDROID, GAME APPLICATION, GAME DEVELOPMENT, GAME ENGINES, OPERATING SYSTEM, UNITY, USER.

ЗМІСТ

ТЕРМІНИ, СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Бібліотеки – збірка об'єктів чи підпрограм для вирішення близьких за тематикою задач. У залежності від мови програмування бібліотеки містять об'єктні модулі чи сирцевий код та дані, допоміжні для задіяння та інтеграції нових можливостей в програмні рішення.

Движок – набору інструментів, який дозволяє працювати з графікою, фізикою, скриптами та іншим.

Движок гри (game engine) – це її основне ядро, базове програмне забезпечення, на основі якого будуються всі інші складові гри. Програмний код, який може використовуватися для створення варіацій гри, аддонов до неї або навіть зовсім нового ігрового світу.

Ком'юніті – спільнота; це об’єднання людей, згуртованих спільними умовами життя, метою та інтересами.

Мобільний застосунок або додаток – програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях.

Операційна система – це базовий комплекс програм, що виконує керування апаратною складовою комп'ютера або віртуальної машини; забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем. Операційна система звичайно складається з ядра операційної системи та базового набору прикладних програм.

Препродакшн – це процес підготовки до створення гри.

Програмний продукт – програмне забезпечення, розроблене для вирішення задачі масового попиту та призначене для постачання користувачам.

Продакшн – кінцева версія програми або сайту, доступна пересічним користувачам (production). Простіше кажучи, те, що ми можемо знайти в Google, скачати з Google Play або Apple Store.

Сеттінг – сукупність різнопланових елементів, які однозначно ідентифікують світ, де відбуваються події певного художнього твору, відеогри або настільної гри.

Софт-лончу – це підхід до запуску додатків, який передбачає ретельне тестування самого додатка та рекламної кампанії щодо нього буквально у бойових умовах, а не лише на рівні гіпотези. У перекладі з англійської Soft Launch означає м'який запуск.

Стейджінг – проміжна, отладочная версія сайту, викладка для тестувальників (staging).

Сценарій (скрипт) – це програма, яка автоматизує деяке завдання, яке без сценарію користувач робив би вручну, використовуючи інтерфейс програми. Мови таких скриптів спочатку орієнтувалися на використання як внутрішні керуючі мови у складних системах.

Фідбек – зворотний зв'язок; відгук, критичний коментар до чогось.

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проєкту та написання коду.

APK-файл – формат файлів, що складається з повних архівованих кодів Android-додатків. Ці компоненти упаковані в один інсталяційний файл – це і є APK.

App Store – платформа цифрової дистрибуції, що розроблена і підтримувана Apple Inc. для комп'ютерних програм та мобільних застосунків на своїх операційних системах.

GUI – програмна оболонка, яка надає користувачеві зручний інтерфейс для роботи з операційною системою. GUI візуалізує багато компонентів у вигляді графічних об'єктів, наприклад, кнопки, меню, стрілки тощо.

Google Play – крамниця застосунків від Google, що дозволяє власникам пристроїв з мобільною операційною системою Android та іншими завантажувати і купувати різні застосунки, книги, фільми і музику.

Ini – розширення файлів конфігурації, що містять дані налаштувань для сімейства операційних систем Windows, а також для їх деяких додатків.

БД – база даних.

ЗБТ – закрите бета-тестування.

ОС – операційна система.

ПЗ – програмне забезпечення.

ПП – програмний продукт.

API – Application Programming Interface.

APK – Android Package Kit.

FPS – First-Person Shooter.

GUI – Graphical User Interface.

RTS – Real-Time Systems.

SDK – Software Development Kit.

TPS – Third-Party Shooters.

ВСТУП

Розробка простого та зручного у використання ігрового застосунку завжди буде користуватися попитом у користувачів мобільним пристроєм. Станом на 2021 рік, кількість мобільних застосунків та їх завантаження стрімко збільшилося порівняно з минулими роками. На сьогоднішній день у світі більше 5 млрд. мобільних користувачі, а це складає 66,6% усього населення світу.

Використання мобільних програм, постійно зростає, а мобільні застосунки стають одним із кращих спосіб комунікацій розробника та клієнта (користувача). Відповідно до статистики, від початку пандемії COVID-19 це питання стало особливо актуальним.

Технології розвиваються швидко, а сучасні користувачі мобільними пристроями, досить сильно залежать від своїх пристроїв. Якщо, враховувати результати проведеного дослідження We Are Social і Hootsuite [1], 3 години 39 хвилин – це середній показник щоденного проведення часу, який витрачає людина на проведення в мобільному телефоні, що на 20% більше, чим в 2020 році. Враховуючі це можна зробити висновок, що з кожним роком користувач все більше часу проводить в своєму мобільному пристрою, використовуючи різноманітні програми та застосунки.

Актуальність магістерської кваліфікаційної роботи обумовлюється необхідністю розробки зрозумілого інтерфейсу, зручного та простого у використанні ігрового застосунку з індивідуальним сюжетом.

Метою кваліфікаційної роботи є аналіз методів, засобів та технологій розробки ігрового застосунку для пристроїв під ОС Android.

Мобільний застосунок – це невід’ємна частина повсякденного життя більшості людей. Якщо вірити статистики, то за 2020 рік було завантажено приблизно 218 млрд. мобільних застосунків та програм. А це на 7% більше, ніж в 2020 році. Згідно даних статистики 92% свого денного часу люди про-

водять в мобільних пристроях, використовуючи різні застосунки, з них 44% від всього часу займають соціальні мережі та комунікаційні програми.

Сучасна ігрова індустрія з кожним роком набирає все більших оборотів. Це в першу чергу стосується мобільної розробки ігрових застосунків. Раніше всі ігрові застосунки створювалися вручну під кожну платформу. Кожного разу необхідно було ігровий застосунок створювати з нуля, а частіше всього ще і залізо комп'ютерів доопрацьовувати, щоб можна було реалізувати нові та цікаві ідеї, можливості на ті часи. Після чого, розробники помітили, що у процесі розробки ігрових застосунків є багато рутинних задач, які краще та легше автоматизувати, чим програмувати, написавши їх один раз. В подальшому звертатися до них з коду. Таким чином, було створені ігрові бібліотеки, фреймворки, а в подальшому і повноцінні движки.

Дана кваліфікаційна робота магістра, складається з 72 сторінок, 41 рисунку, 1 таблиці та 12 джерел посилання.

1 АНАЛІЗ СУЧАСНИХ ДВИЖКІВ ДЛЯ РОЗРОБКИ ІГРОВИХ ЗАСТОСУНКІВ

Сучасні движки для створення ігор зазвичай поєднують у собі безліч функцій: візуальний редактор сцен, утиліту для анімації, зону для скриптингу та тестування, налаштування окремих об'єктів та багато інших повсякденних інструментів.

При цьому вони зберігають гнучкість. Також можна створювати об'єкти не через візуальний редактор, а за допомогою мов програмування. Що дозволяє реалізувати такі популярні механіки, як процедурна генерація, коли всі етапи розробки пройдені, саме двигуни допомагають упакувати гру для кожної сумісної платформи.

Якщо проаналізувати існуючі движки, можна відокремити найпопулярніші із них. На останніх місцях слід поставити всі типові 2D редактори. Щоб бути точнішим мова йде про Solar2D та Defold. У цих двигунах закладена кросплатформність. Ось тільки немає 3D, яке для першої гри і не потрібне, але все ж таки приємно мати його про запас. А тому в топ-3 цим інструментам не потрапити, адже навіть вищезазначене 2D реалізовано в них не краще, ніж движках які розглядаються далі. Хоча варто їх хоча б поставити та випробувати особисто.

Ігрові движки Ren’Py та Scratch варто розібрати та поставити після Solar2D та Defold.

Далі буде доцільним розташувати Unreal Engine та Unity3D, відповідно. Вони дуже схожі за своєю суттю. Неймовірно функціональні движки, здатні відмалювати чудову графіку, детально обробити звук або зіштовхнути сотню гравців у мережі. У обох движків величезна база користувачів, уроків, прикладів та ассетів, які можна купити або завантажити безкоштовно. Unity трохи більше заточений саме під 2D графіку та фізику. А Unreal незамінний для розробки великих проєктів, що особливо потребують мережевих взаємодій. Але обидва вони гіпертрофовані розробки простих ігор. Недолік в тому, що обид-

ва двигуни імітують 2D через 3D, що позначається на продуктивності, на певних аспектах розробки. Тому для розробки 2D ігор в Unity і Unreal використовується певною мірою костилі.

На перше місце, можна поставити Godot Engine. У ньому реалізовані 2D та 3D сцени. Причому двовимірні ігри не на костилях у тривимірному просторі, а реалізовано на піксельних 2D. Для двовимірної та тривимірної середовищ є свої фізичні об'єкти, свої ефекти, камери та освітлення. І це дає відмінну продуктивність та зручність розробки. Але що куди важливіше, у процесі зовсім не потрібно виходити з двигуна.

Більше того, дані про навантаження заліза також можна переглядати і дізнаватися, де слабкі місця гри, що розробляється. Типові рішення можна переглянути у демонстраційних проєктах, доступних на офіційному сайті, а також у сотнях відкритих проєктів та плагінів. Цей підхід значно прискорює процес розробки простих ігор та спрощує роботу на слабких комп'ютерах.

1.1 Розгляд Solar2D Game Engine

Solar2D – це ігровий движок на основі Lua, який зосереджений на легкості ітерацій і використання. Це проєкт із повністю відкритим вихідним кодом, який розгалужується від добре запровадженого та широко використовуваного ігрового движка Corona SDK, який більше не підтримується комерційно.

Ігровий движок використовується при розробці для мобільних пристроїв, настільних комп’ютерів і підключених телевізійних пристроїв лише за допомогою однієї кодової бази: iOS, tvOS, Android, Android TV, macOS, Windows, Linux або HTML5.

Solar2D [2] є офіційним форком Corona SDK, який активно розробляється більше 10 років і використовується сотнями тисяч програм і розробників. Lua — це мова сценаріїв з відкритим вихідним кодом, розроблена як легка, швидка та водночас потужна. Lua наразі є провідною мовою сценаріїв в

іграх і використовується в Warcraft™, Angry Birds™, Civilization™ та багатьох інших популярних франшизах.

При розробці ігрового застосунку можна обрати один із численних плагінів, які розширюють ядро Solar2D для таких функцій, як реклама в програмі, аналітика, медіа та багато іншого. Велика різноманітність плагінів доступна через безкоштовний каталог Solar2D або сторонні магазини, як от Solar2D Marketplace і Solar2D Plugins.

Якщо бібліотеки немає в ядрі або вона не підтримується через плагін, можна викликати будь-яку нативну (C/C++/Obj-C/Java) бібліотеку або API за допомогою Solar2D Native.

Solar2D не відстежуватиме користувачів розробленого ігрового застосунку. Ні збору анонімних даних, ні викликів сервера. нічого ігрові застосунки, які створюються, надсилаються лише запити до мережі, про які розробник запитує.

Ніяких прихованих комісій, зборів або роялті. Незалежно від того, чи незалежний розробник, чи великий видавець, все одне не потрібно буде платити за використання двигуна. Весь вихідний код і ресурси доступні за ліцензією MIT (рис.1) .

Рисунок 1 – Логотип Solar2D Game Engine

1.2 Розгляд Defold Game Engine

Defold – доступний вихідний ігровий движок із зручною для розробників ліцензією, яка походить від популярної ліцензії Apache 2.0. Ліцензія надає свободу розробляти ігри, не турбуючись про ліцензійні збори чи роялті [3]. Даний ігровий застосунок вважається найкращим ігровий движок для Інтернету та мобільних пристроїв

Defold (рис.2) розроблений таким чином, щоб його можна було легко розширити додатковими функціями та функціями за допомогою розширень. Великий вибір офіційних розширень із відкритим кодом, розроблених спільнотою, доступний на порталі ресурсів.

Ігровий движок Defold є і завжди залишатиметься абсолютно безкоштовним у використанні. Defold ніколи не змінюватиметься на модель передплати, не вимагатиме виплати роялті, не вводитиме ліцензійних зборів чи іншим чином стягуватиме плату за доступ до основного продукту. В цілях фонду чітко зазначено, що Defold має бути доступним безкоштовно та що його не можна комерціалізувати. Цілі захищені шведським законодавством про заснування і не можуть бути змінені.

Рисунок 2 – Логотип Defold Game Engine

Сервери збірки власних розширень є настільки невід’ємною частиною пропозиції продуктів Defold, що вони повинні залишатися доступними як безкоштовні послуги для наших користувачів. І тепер, коли вихідний код сервера збірки доступний, будь-хто може вільно налаштувати та запустити свій власний сервер, локально або як загальнодоступний чи приватний сервер.

Defold є набагато більш стійким до майбутнього. Той факт, що вихідний код Defold доступний, дозволить будь-кому покращити Defold, виправляючи помилки, оновлюючи його новими функціями або виправляючи його для роботи з новими оновленнями операційної системи, не залежачи від іншої компанії для виконання роботи.

1.3 Розгляд Ren’Py Game Engine

Ren'Py – це механізм візуальних романів, яким користуються тисячі творців з усього світу, який допомагає використовувати слова, зображення та звуки, щоб розповідати інтерактивні історії, які запускаються на комп’ютерах і мобільних пристроях. Це можуть бути як візуальні романи, так і ігри-симулятори життя. Проста для вивчення мова сценаріїв дозволяє будь-кому ефективно писати великі візуальні романи, тоді як його сценаріїв на Python достатньо для складних ігор-симуляторів (див.рис.3).

Рисунок 3 – Логотип Ren'Py Game Engine

Перш ніж ми зможемо описати мову Ren'Py, ми повинні спочатку описати структуру сценарію Ren'Py. Це стосується того, як файли розбиваються на блоки, що складаються з рядків, і як ці рядки розбиваються на елементи, які складають оператори.

Сценарій гри Ren'Py складається з усіх файлів, які знаходяться в каталозі гри з розширенням .rpy. Ren'Py розглядатиме кожен із цих файлів (у порядку Юнікод їхніх шляхів) і використовуватиме вміст файлів як сценарій.

Як правило, немає різниці між сценарієм, розбитим на кілька файлів, і сценарієм, який складається з одного великого файлу. Управління можна передавати між файлами, переходячи до мітки в іншому файлі або викликаючи її. Це робить поділ сценарію на файли питанням особистого стилю – деякі розробники ігор віддають перевагу малим файлам (наприклад, по одному на подію або по одному на день), тоді як інші вважають за краще мати один великий сценарій.

Щоб пришвидшити час завантаження, Ren'Py скомпілює файли .rpy у файли .rpyc під час запуску. Якщо файл .rpy змінено, файл .rpyc буде оновлено під час запуску Ren'Py. Однак якщо файл .rpyc існує без відповідного файлу .rpy, буде використано файл .rpyc. Це може призвести до проблем, якщо файл .rpy буде видалено без видалення файлу .rpyc. Імена файлів мають складатися з літери або цифри та не можуть починатися з «00», оскільки Ren'Py використовує такі файли для власних цілей [4].

Базовий каталог – це каталог, який містить усі файли, які поширюються разом із грою. (Він також може містити деякі файли, які не розповсюджуються разом із грою.) Такі речі, як файли README, слід розміщувати в базовому каталозі, звідки вони розповсюджуватимуться.

Базовий каталог створюється під каталогом Ren'Py і містить назву гри. Наприклад, якщо каталог Ren'Py має назву renpy-6.11.2, а гра – «HelloWorld», основним каталогом буде renpy-6.11.2/HelloWorld.

Каталог гри майже завжди є каталогом під назвою «гра» під базовим каталогом. Наприклад, якщо основним каталогом є renpy-6.11.2/HelloWorld, каталогом гри буде renpy-6.11.2/HelloWorld/game.

Однак Ren'Py шукає каталоги в такому порядку:

- ім'я виконуваного файлу без суфікса, наприклад, якщо виконуваний файл має назву moonlight.exe, він шукатиме каталог із назвою moonlight у базовому каталозі;
- назва виконуваного файлу без суфікса та з видаленим префіксом, що закінчується на _, наприклад, якщо виконуваним файлом є moonlight_en.exe, Ren'Py шукатиме каталог із назвою en;
- каталоги «game», «data» і «launcher» у такому порядку;
- однак програма запуску правильно розпізнає лише каталоги «game» і «data».

Каталог гри містить усі файли, які використовує гра. Він, включно з усіма підкаталогами, сканується на наявність файлів .rpy і .rpyc, які об’єднуються для створення сценарію гри. Він сканується на наявність архівних файлів .rpa, і вони автоматично використовуються грою. Нарешті, коли гра вказує шлях до файлу для завантаження, він завантажується відносно каталогу гри. Варто зауважити, що config.searchpath може це змінити.

Більшість операторів Ren'Py мають спільний синтаксис. За винятком оператора say, вони починаються з ключового слова, яке вводить оператор. За цим ключовим словом йде параметр, якщо його приймає оператор.

Потім за параметром слідує одна або кілька властивостей. Властивості можна надавати в будь-якому порядку, за умови, що кожна властивість надається лише один раз. Властивість починається з ключового слова. Для більшості властивостей за назвою властивості слідує один із наведених вище синтаксичних елементів.

Якщо оператор займає блок, рядок закінчується двокрапкою (:). В іншому випадку рядок просто закінчується.

Ren'Py це ігровий движок з відкритим кодом, також він є безкоштовним для комерційного використання.

1.4 Розгляд Scratch Game Engine

Scratch – найбільша у світі спільнота програмування для дітей та мова програмування з простим візуальним інтерфейсом [5], що дозволяє молодим людям створювати цифрові історії, ігри та анімацію. Scratch спроєктований, розроблений та модерується некомерційною організацією Фонд Scratch (рис.4).

Рисунок 4 – Логотип Scratch Game Engine

Scratch розвиває обчислювальне мислення та навички вирішення проблем; творче викладання та навчання; самовираження та співробітництво; та рівність у обчисленнях.

Scratch – це високорівнева блочна мова візуального програмування та веб-сайт, орієнтований насамперед на дітей як навчальний інструмент для програмування. Користувачі сайту, які називаються Scratchers, можуть створювати проєкти на веб-сайті за допомогою блокового інтерфейсу. Проєкти можна експортувати у файли HTML5, JavaScript, програми Android, Bundle

(macOS) і EXE за допомогою зовнішніх інструментів. Сервіс розроблено медіа-лабораторією Массачусетського технологічного інституту використовується в більшості країн світу. Scratch викладають і використовують у позашкільних центрах, школах і коледжах, а також в інших державних установах. Станом на 2022 рік статистика спільноти на офіційному веб-сайті мови свідчить про понад 104 мільйони проєктів, якими поділилися понад 90 мільйонів користувачів, понад 686 мільйонів проєктів, коли-небудь створених (включно з проєктами, які не використовуються), і понад 100 мільйонів відвідувань веб-сайту щомісяця [6].

Scratch отримав свою назву від техніки, яку використовують диск-жокеї під назвою «скретчінг», коли вінілові платівки скріплюються разом і маніпулюють ними на вертушці для створення різних звукових ефектів і музики. Подібно до Scratch, веб-сайт дозволяє користувачам змішувати різні засоби масової інформації (включно з графікою, звуком та іншими програмами) у творчий спосіб, створюючи та «реміксуючи» проєкти, такі як відеоігри, анімація, музика та симуляції.

Інтерфейс Scratch поділено на три основні секції: область сцени, палітру блоків і область кодування для розміщення та впорядкування блоків у сценарії, які можна запускати, натиснувши зелений прапорець або клацнувши сам код. Користувачі також можуть створювати власні блоки коду, і вони відображатимуться в «Моїх блоках».

Область сцени містить результати (наприклад, анімацію, графіку черепахи, у малому або нормальному розмірі, також доступна повноекранна опція) і всі мініатюри спрайтів, перераховані в нижній області. Сцена використовує координати x і y, де 0,0 є центром сцени.

Коли спрайт вибрано в нижній частині робочої області, до нього можна застосувати блоки команд, перетягнувши їх із палітри блоків у область кодування. Вкладка «Костюми» дозволяє користувачам змінювати вигляд спрайту за допомогою векторного та растрового редактора, щоб створювати різнома-

нітні ефекти, зокрема анімацію. Вкладка Звуки дозволяє додавати звуки та музику до спрайту.

Створюючи спрайти, а також фони, користувачі можуть намалювати власний спрайт вручну, вибрати спрайт із бібліотеки або завантажити зображення. Scratch завжди безкоштовний і доступний (перекладений) більш ніж 70 мовами.

1.5 Розгляд Unreal Engine Game Engine

Один з найпопулярніших ігрових рушіїв – Unreal Engine [7] від Epic Games. Оригінальна версія була випущена в 1998 році, а через 21 рік вона все ще продовжує використовуватися для великої кількості ігор щороку. Unreal Engine вважається найвідкритішим і найдосконалішим у світі інструмент, який викристовується для створення 3D у реальному часі (див.рис.5)

Рисунок 5 – Логотип Unreal Engine Game Engine

Найвідомішими іграми, створеним разом з Unreal Engine, є серія Gears of War, серія Mass Effect, серія Bioshock та серія Batman: Arkham.

Сила Unreal Engine полягає в тому, що його достатньо легко можна модифікувати, щоб кожну гру можна було зробити дуже унікальною. Передостання версія, Unreal Engine 4, вважається найпростішою у використанні в руках професіонала. Однак є й інші двигуни, які легші для нових дизайнерів.

За допомогою вбудованого в Unreal Engine 5 набору інструментів для розробки анімації, зручного для художників, можна виконувати ітерацію швидше й точніше, зменшуючи потребу в трудомісткому обході за допомогою інструментів DCC. Швидко дозволяє створювати риги в Control Rig, а потім позуйте та анімуйте їх у Sequencer або використовувати новий набір інструментів для перенацілювання, щоб легко повторно використовувати існуючі анімації.

Розробник навіть може динамічно налаштовувати анімацію під час виконання, щоб компенсувати різні сценарії ігрового процесу, наприклад зміну швидкості або місцевості.

Unreal Engine дозволяє розробникам і творцям ігор у різних галузях створювати 3D-контент наступного покоління в реальному часі з більшою свободою, точністю та гнучкістю, ніж будь-коли раніше.

У версії Unreal Engine 5 розроблені нові та зручні для художника інструменти для створення анімації, перенацілювання та часу виконання – разом із значними розширеним набором інструментів для моделювання – зменшують ітерації та усувають зворотні переходи, прискорюючи творчий процес.

Unreal Engine 5 отримав оновлений інтерфейс. Завдяки повній інтеграції Quixel Bridge у розробника є доступ до всієї бібліотеки Megascans за допомогою перетягування. Це частина нового меню «Створити» для отримання вмісту та створення та розміщення акторів.

Щоб звільнити місце у вікні перегляду, тепер розробник може легко викликати та зберігати вміст-браузер і прикріплювати будь-яку вкладку редактора до бічної панелі. Крім того, на панелі «Деталі» можна швидше знайти потрібні властивості.

Unreal Engine дозволяє створювате ігри та світи з величезною кількістю геометричних деталей за допомогою Nanite, віртуалізованої системи геометрії мікрополігонів і нової системи Virtual Shadow Map. Напряму можна імпортувати та копіювати багатомільйонні полігональні сітки, зберігаючи частоту кадрів у реальному часі без будь-якої помітної втрати точності.

Ці системи інтелектуально передають і обробляють лише ті деталі, які розробник може сприйняти, значною мірою усуваючи обмеження підрахунку полігонів і викликів малювання.

1.6 Розгляд Unity3D Game Engine

Один з них – Unity, багатоплатформенний ігровий рушій, який дозволяє легко створювати інтерактивний 3D-контент. Дуже багато Інді-розробників віддають перевагу Unity за його відмінні функціональні можливості, якісний контент та можливість його використання майже для будь-яких ігор (див.рис.6).

Рисунок 6 – Логотип Unity3D Game Engine

Останні відомі ігри, що створені використовуючи Unity, це Lara Croft Go, Her Story, Pillars of Eternity, та Kerbal Space Program. Найкраще, що можна сказати про Unity 5 – це Персональне видання, яке безкоштовне для заван-

таження. Це видання включає в себе рушій з усіма функціями і може (здебільшого) використовуватися для створення ігор на будь-якій платформі. Проблема полягає в тому, що Професійне видання, яке має безліч чудових інструментів, вимагає сплати щомісячної плати. Ці функції включають бета-доступ, звітування про ефективність ігор, можливість налаштувати екран заставки, ліцензію команди тощо.

Відповідно до інформації джерела [8] Unity – це набагато більше, ніж найкраща у світі платформа розробки в реальному часі, це також надійна екосистема. Програмне забезпечення для розробки ігор Unity дозволяє розробникам створювати високоякісні 3D- і 2D-ігри та легко розгортати їх на настільних комп'ютерах, VR/AR, консолях, веб- і мобільних платформах.

1.7 Розгляд Godot Game Engine

Рушій Godot [9] чудово підходить для створення 2D та 3D ігор. Рушій пропонує величезний набір поширених інструментів, тому розробник може просто зосередитись на створенні своєї гри (рис.7).

Рисунок 7 – Логотип Godot Game Engine

Godot Game Engine підтримує багатоплатформне розгортання, а саме:

– експорт на настільні платформи: Windows, macOS, Linux, UWP і *BSD;
– експорт на мобільні платформи: iOS і Android;
– консолі: Nintendo Switch, PlayStation 4, Xbox One (через сторонніх постачальників);
– експорт в Інтернет за допомогою HTML5 і WebAssembly;
– розгортання та експорт на більшість платформ одним клацанням миші, також легко створювати власні збірки.

У Godot є спільнота, яка постійно виправляє помилки у роботі рушія та розробляє нові функції, що завжди є хорошим знаком. Активна спільнота також означає, що ви отримаєте відповіді навіть на ваші найбільш конкретні питання, пов’язані з Godot. Даний движок також посилається на інші свої Інтернет-хаби, включаючи форуми Reddit, групи Facebook, спільноту у Steam, форуми Godot тощо.

Постачається Godot Game Engine із сотнями вбудованих вузлів, які роблять дизайн гри легким. Розробник також може створити власний для власної поведінки, редакторів і багато іншого. При розробці передбачається створення композиції вузлів із підтримкою екземплярів і успадкування. Візуальний редактор із усіма необхідними інструментами, упакованими в красивий і лаконічний контекстно-залежний інтерфейс користувача. Постійне редагування в реальному часі, де зміни не втрачаються після зупинки гри. Слід звернути увагу, що він працює також і на мобільних пристроях.

Движок має інноваційну архітектуру, яка поєднує в собі найкраще від прямого рендерингу з ефективністю відкладеного рендерингу. Фізична візуалізація з повною підтримкою MSAA та FXAA. Повний принциповий BSDF з підповерхневим розсіюванням, відображенням, заломленням, анізотропією, прозорим покриттям, коефіцієнтом пропускання тощо. Проста у використанні мова шейдерів (shader) на основі GLSL із вбудованим редактором і доповненням коду. Godot постачається з повністю спеціалізованим двовимірним

механізмом, наповненим багатьма функціями. Дружнє використання файлової системи , яка чудово працює з такими системами контролю версій, як Git, Subversion, Mercurial. Наявний також текстовий опис і оптимальний формат сцени, при цьому синтаксис сценарію підтримується GitHub.

Godot спрощує розробку міжплатформної доповненої та віртуальної реальності. Працює з багатьма гарнітурами , включаючи Meta Quest, Valve Index, HTC Vive, Oculus Rift, усі гарнітури Microsoft MR та багато іншого. Підтримує OpenXR , нового відкритого стандарту, на який переходить більшість постачальників апаратного забезпечення. Плагіни в свою чергу надають доступ до різноманітних власних SDK, таких як OpenVR (SteamVR) і застарілих Oculus SDK. WebXR може надавати досвід AR/VR через веббраузер, а ARKit дозволяє створювати програми AR для iOS

Godot є повністю безкоштовним і з відкритим вихідним кодом під дуже дозвільною ліцензією MIT. Ніяких умов, жодних роялті, нічого. Розроблена гра належить розробнику, аж до останнього рядка коду двигуна.

Сьогодні існує багато різних ігрових движків. Хоча всі вони мають свої сильні та слабкі сторони, зрештою окремий розробник вирішує, який з них він хоче використовувати. Інші вважають механізми самонавчання дуже корисними завдяки величезній бібліотеці ресурсів, доступних в Інтернеті. Зрештою, це зводиться до особистих уподобань або того, що найкраще відповідає потребам окремого розробника.

Багато новачків недооцінюють складність створення ігор. Розробка ігор є одним із найскладніших видів розробки програмного забезпечення. Є кілька речей, про які кожен розробник ігор повинен пам’ятати, починаючи.

Дуже важливо мати чітке бачення своєї гри. Чітке бачення допоможе вам у процесі розробки та допоможе вибрати правильні інструменти та ресурси для досягнення ваших цілей.

Важливо переконатися, що завжди є потрібні інструменти та ресурси. Необхідно мати доступ до найновішого програмного забезпечення, інструментів і матеріалів для швидкої та ефективної розробки ігор.

Завжди потрібно бути готовим до непередбачених проблем під час розробки ігор. Це дозволить подолати проблеми на ранній стадії та продовжити свій проєкт з максимальною ефективністю та успіхом.

Перед початком розробки ігрового застосунку, слід завершити всі проєкти. Є багато переваг завершення ігрових проєктів, особливо якщо розробник – новачок у розробці ігор. Виконуючи завдання, можна краще зрозуміти кодову базу та ігрову механіку. Крім того, це може допомогти покращити навички розвитку та краще зрозуміти, як працюють ігри.

Не турбуйтеся про те, щоб гра мрії стала саме першою грою. Хоча надихаючих історій небагато, розробка ігор є складною. Перша гра, ймовірно, буде поганою, і це нормально. Найважливіше, це закінчити проєкти і вчитися враховуючи всі етапи розробки з якими розробник зіткнеться безпосередньо під час розробки ігрового застосунку.

2 АНАЛІЗ І ОПИС ОСНОВНИХ ЕТАПІВ СТВОРЕННЯ ІГРОВОГО ЗАСТОСУНКУ

Створення комп'ютерних ігор не лише цікава, а й прибуткова справа. За прогнозами аналітичної компанії Newzoo, ігрова індустрія за обсягами цього року перевищить $152 млрд, а доходи окремих ігрових компаній уже давно обчислюються мільярдами доларів.

Процес створення мобільної гри можна грубо розділити на 2 етапи – препродакшн та продакшн: підготовка до випуску ігрового контенту та саме виробництво. Однак, щоб розібратися у важливих тонкощах, варто копнути глибше. Найчастіше при створенні гри з нуля продукт проходить 7 стадій розробки [10] (див.рис.8)

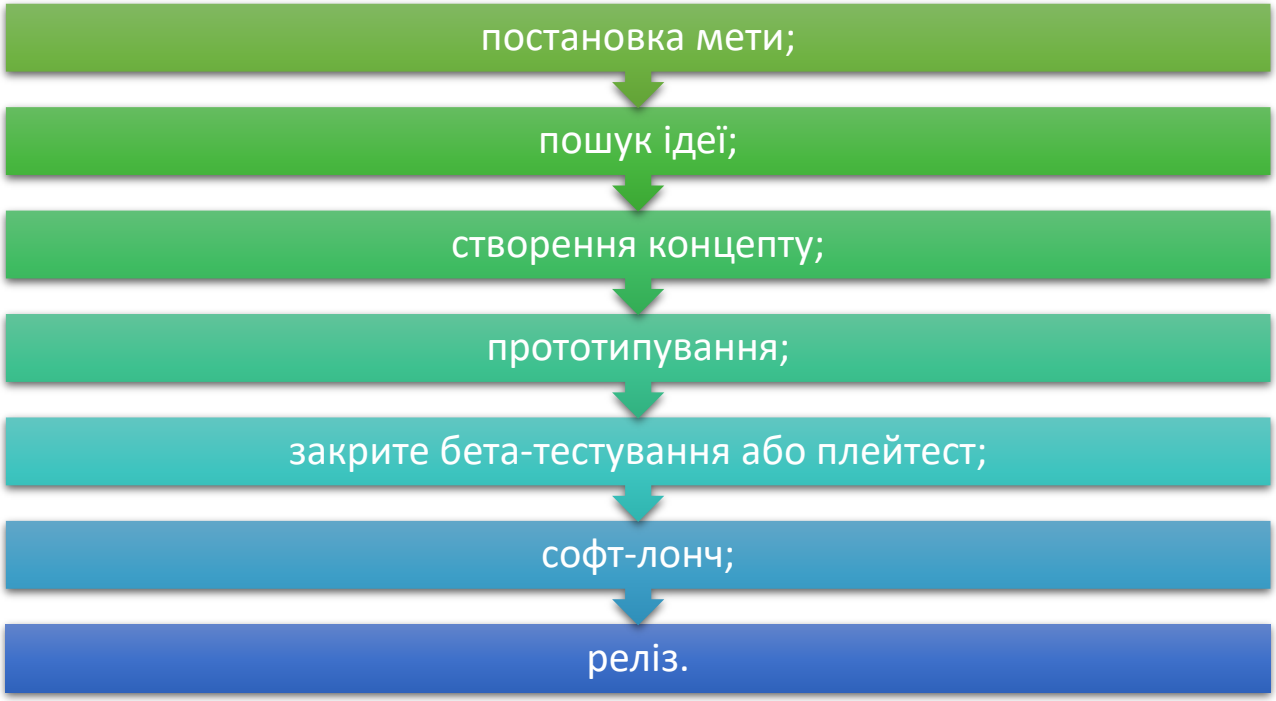

Рисунок 8 – Схема етапів розробки ігрового застосунку

Приступаючи до нового проєкту почати із встановлення мети. Мета це перше, з чим необхідно визначитися перед початком розробки. Мета має бу-

ти конкретною, вимірною, досяжною та реалістичною. І не варто забувати про те, що завжди будуть обмеження часу. Забути про мету це все одно, що піти в похід без карти або почати будівництво будинку без плану. Постановка мети гарантує, що розробник не тільки знає, що збираєтеся робити, а як саме він збираєтеся це робити. Коли ціль сформульована, можна переходити до наступного, не менш важливого етапу.

Препродакшн – це процес підготовки до створення гри. Етап препродакшна включає 3 підетапи: пошук ідеї, створення концепту та прототипування. Мета препродакшна – визначити ідею гри, продумати, що зробить гру унікальною і чим вона відрізнятиметься від усіх інших, як і чому вона стане актуальною та затребуваною. Для цього необхідно підготувати основну документацію (геймдизайн-документ) та опрацювати всі слабкі місця перед запуском гри в продакшн.

На цьому етапі розробник повинен створити документацію, яка детально описує гру. Грамотно спланований препродакшн скорочує тривалість наступного етапу та економить бюджет. Якщо враховані ризики, виявлені чинники успішності проєкту, а економіка гри, унікальні фічі та механіка доведені до досконалості, глобальних правок під час продакшну буде менше.

Можна створити десятки проєктів зі зрозумілими механіками, які впізнаються сюжетними поворотами та чудовою графікою, і при цьому не отримати визнання гравців. Щоб проєкт став справді успішним, у ньому має бути «родзинка». Потрібно викликати у гравців емоції, які хоч трохи відрізнятимуться від того, що можна випробувати у реальному світі.

Але при цьому необхідно дотримуватися балансу між знайомим для гравця світом і чимось незвичайним, адже приходять у гру саме за новими враженнями. В ігровій промисловості є багато різних жанрів та сеттингів. Наприклад, такі жанри, як FPS/TPS, RTS, Horror та всілякі симулятори. А сеттінг може бути історичний, фентезі, стімпанк чи кіберпанк. На відміну від сеттингу далеко не всі жанри підходять для мобільних ігор. Тому вже на етапі пошуку ідеї розробник повинен уявляти, як гра виглядатиме в результаті.

Наприклад, ситуація: гравець після тяжкого робочого дня їде додому. Він їде стоячи, а у вагоні сильна товкучка. Щоб трохи відсторонитися від сірої реальності і полоскотати нерви, він тягнеться за своїм телефоном і хоче насолодитися грою. Мобільний пристрій не дозволяє досягти того ж рівня занурення, що PC, приставка або VR. Розробник повинен продумати класні і, головне, доречні для мобільного геймінгу ідеї до початку розробки гри.

Створення концепту – це підготовчий етап, мета якого – відшліфувати ідею майбутньої гри та зібрати базову геймдизайнерську документацію (Pitch Document або Concept Document). Вони викладають короткий, але при цьому ємний опис гри з ключовими аспектами. У процесі створення концепт-документу в першу чергу необхідно відповісти на запитання: що за гра розробляється і для яких платформ, про що ця гра, для кого призначено фінальний продукт та хто потенційні конкуренти. У Concept Document варто описати сюжет, сеттінг, основну механіку гри. Такий документ використовують усередині компанії та представляють видавцеві. Важливий момент, усі геймдизайнерські документи мають регулярно актуалізуватися та бути доступними всім співробітникам. Це необхідно для того, щоб кожен член команди, чи то художник, програміст чи сценарист, могли будь-якої миті звернутися до документа та зрозуміти, чи сходиться проміжний результат з тим, що було заплановано.

На етапі прототипування команда розробників гри прототипує свій концепт: перевіряє ідеї та гіпотези, втілюючи їх у невеликих простих прототипах. Наприклад, створює 2-3 варіанти візуального оформлення рівнів, щоб у результаті вибрати найвдаліше.

Те саме стосується механіки та поведінки персонажів. На прототип зазвичай витрачають мінімум часу – тестують лише те, що потрібно, і не звертають увагу на його «красу». Складний у реалізації та «причесаний» прототип ніяк не вплине на фінальний вид гри, тож витрачати ресурси на його підготовку не варто.

Продакшн найбільш ресурсозатратний етап створення мобільної гри. Завдання – випустити повноцінну версію гри для внутрішнього перегляду продюсерами, закритого бета-тестування, техлончу або софт-лончу. На цьому етапі створюється гра, а саме команда готує необхідний арт, анімує, налаштовує ігровий баланс, програмує та багато тестує. Результат продакшну – перша версія гри.

На етапі закритого бета-тестування (ЗБТ) готову гру демонструють новим та обмеженим за кількістю гравців аудиторію. Це допомагає знайти геймдизайнерські помилки, виявити проблеми та побачити загальну картину того, як гравці взаємодіють із фінальним продуктом. Розробники відеоігор аналізують дані проведених плейтестів і роблять висновки: що в грі добре, а що погано, які потрібні зміни та доопрацювання. Але навіть під час таких тестів продакшн на великих проєктах продовжується: допрацьовуються фічі, гру призводять до фіналу та зустрічі з більш численною аудиторією.

У момент софт-лонч починається справжнє "життя" гри. На етапі софт-лончу продукт представляють більш широкому загалу, наприклад, відкривають доступ до гри в декількох країнах. Здійснюють перевірку технічних метрик і параметрів, ваги гри, швидкості завантаження та якості роботи на різних платформах, оптимізують гру під прийом трафіку з постійним його збільшенням. Йде збір статистики, обробка геймерського фідбеку. Виходячи з даних програмісти шліфують знайдені недоробки, продюсер і геймдизайнери аналізують статистику.

У індустрії мобільних ігор існує два основних ринки: Android із Google Play Store та iOS (iPhone/iPad) із App Store (див.рис.9). В обох є як позитивні, так і негативні сторони. Потрібно буде вирішити, на якій платформі публікувати свою гру. Розробник також може опублікувати свою гру для обох, але це потребуватиме додаткових ресурсів і витрат.

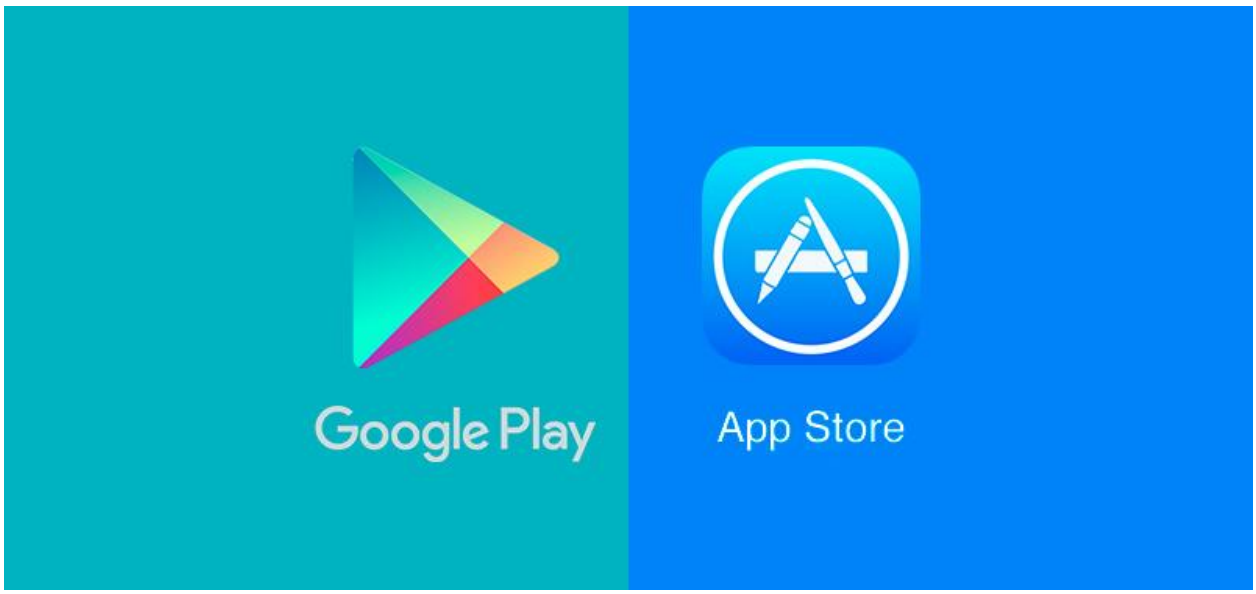

Рисунок 9 – Логотипи двох основних ринків в індустрії мобільних ігор Android – Google Play Store та iOS – App Store

Плата розробнику iOS App Store становить 99 доларів на рік. У Google Play Store є одноразова комісія розробника в розмірі 25 доларів США. Обидві платформи отримують 30% скорочення доходу від покупок програм.

iOS App Store робить більше, щоб просувати нові ігри та програми, але обидві платформи мають високу конкуренцію та вимагають від розробника креативності, щоб виділити свої програми.

Google Play Store на Android має набагато менш суворий процес затвердження для надісланих програм. Набагато важче схвалити програми для iOS App Store, але App Store набагато краще надає розробникам відгуки, коли програми не схвалено.

Користувачі Android, як правило, віддають перевагу безкоштовним програмам, тоді як користувачі iOS набагато більше звикли платити за програми. iOS App Store використовує модель пошуку за ключовими словами. Це вимагає від розробників надати список ключових слів, які користувачі мають ввести, щоб знайти вашу програму. Пошук у магазині Google Play не покладається на ключові слова, а не за назвою програми, описом тощо.

На етапі релізу – гру просувають у App Store та Google Play, розвивають ком'юніті і, якщо необхідно, розширюють сервіс техпідтримки, запускають маркетингові заходи, ведуть роботу над новим контентом.

Продюсер і геймдизайнери опрацьовують статистичні дані, шукають шляхи поліпшення показників (наприклад, показники утримання гравців) та впроваджують нові фічі – настає період Live-Ops. У цей час у грі з'являються заздалегідь заготовлені події, змагання, пропозиції про знижки – робота над продуктом продовжується. Після релізу розробники отримують більше фідбеку від гравців, а отже, можуть вивчити аудиторію, її інтереси та потреби. Все подальше виробництво зводиться до покращення продукту, регулярних оновлень, наповнення гри новим контентом і можливостями, які стимулюватимуть геймерів грати частіше і довше.

На розробку гри необхідно виділити багато чого. Є дослідження, програмування, графічний дизайн, звуковий дизайн, створення музики, маркетинг і багато іншого. Розуміння того, якими навичками володіє розробник, це допоможе в подальшому придумати ідеї на основі сильних сторін.

Можливо, розробник чудовий програміст, але не такий великий художник. Тоді слід зосередитися на ігровій механіці, але покладатися на мінімалістичний художній стиль. Можливо – чудовий графічний дизайнер, але не дуже добрі навики програмувати. В такому випадку, можна знайти ігровий движок, який подбає про більшу частину кодування, та більше зосередитеся на художньому дизайні.

Хоча розробка ігор хаотична за своєю природою, все ще існують структури та рамки, які забезпечують ефективну роботу і виконання проєктів. Існує інша варіація 7 етапів розробки ігрового застосунку (див.рис.10).

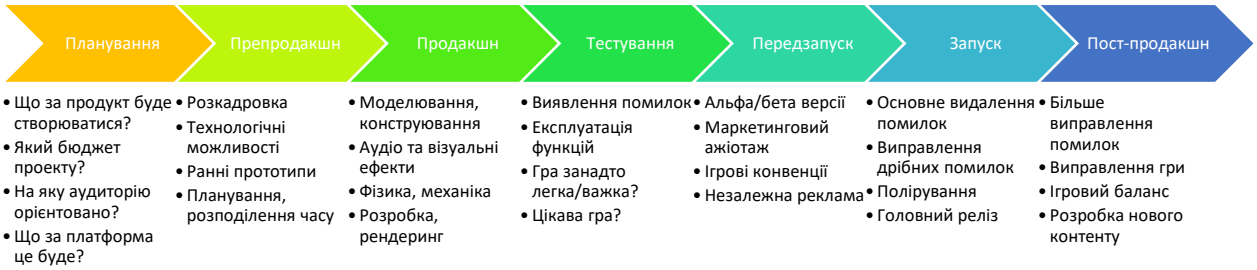

Рисунок 10 – Схема етапів розробки ігрового застосунку

Етап планування розпочинається перш за все з постановки ідеї. На етапі планування потрібно буде дати відповіді на основні питання, наприклад:

- Який тип ігор створюватиметься?
- Це буде 2D чи 3D?
- Які основні функції вона повинна мати?
- Хто персонажі гри?
- Коли і де це відбувається?
- Хто цільова аудиторія?
- На якій платформі це створюється?

Це може здатися не таким, але створення ідеї гри є однією з найважчих частин її розробки. Ідея, запропонована ігровою студією, стане основою всієї гри. Це те, що встановлює стандарти для кожного працівника, який бере участь у створенні гри, а також дає видавцям загальне уявлення про те, чого очікувати. Це підводить до наступної частини розробки – перевірки концепції.

Підтвердження концепції включає всі ідеї, які були згенеровані, і визначає, наскільки вони життєздатні для створення ігровою студією. Після цього потрібно буде відповісти на додаткові запитання, наприклад:

- Яка орієнтовна вартість розробки цієї гри?
- Чи є технологічні можливості для його створення?
- Чи потрібен буде новий ігровий движок?
- Наскільки великою має бути команда?

- Чи наймаються зовнішні актори озвучення та сценаристи?
- Який приблизний термін запуску?
- Як це монетизується?

Для студій, які створюють гру під егідою видавця, перевірка концепції потрібна перед тим, як приступити до попереднього виробництва, і може навіть знадобитися вертикальний зріз. Це пов'язано з тим, що видавець має затвердити презентацію щодо часу, бюджету та маркетингу.

Для незалежних студій без нагляду видавців на цьому етапі є трохи більше гнучкості. Недоліком незалежного видання є створення бюджету на розробку та маркетинг, хоча краудфандингові веб-сайти, такі як Kickstarter і Fig, стають у нагоді. Власне кажучи, такі успішні ігри, як Pillars of Eternity та Shovel Knight , були повністю фінансовані краудфінансуванням.

Який би шлях не був обраний, доказ концепції є життєво важливим для успіху гри, оскільки він ставить ідеї в перспективі того, що можливо. Після чого настає час розпочати попереднє виробництво.

На наступному етапі розробки гри, який називається підготовчим виробництвом (препродакшн), обмірковується, як втілити в життя багато ідей, викладених на етапі планування. Це місце, де сценаристи, художники, дизайнери, розробники, інженери, керівники проєктів та інші важливі відділи співпрацюють над масштабами гри та де підходить кожна частина головоломки. Ось кілька прикладів такої співпраці. Зустріч письменників із проєктом призводить до конкретизації оповіді історії. Хто є головними героями цієї казки? Які їхні передісторії? Як кожен персонаж співвідноситься один з одним? Чи є вільні кінці, які нам потрібно буде зв'язати пізніше? Інженери зустрічаються з авторами, повідомляючи їм, що за поточних технологічних обмежень не можнв заповнити це середовище 100 символами, інакше гра вийде з ладу. Художники зустрічаються з дизайнерами, щоб переконатися, що візуальні ефекти, кольорові палітри та художні стилі узгоджені з тим, що було викладено на етапі планування. Розробники зустрічаються з інженерами, щоб кон-

кретизувати всю ігрову механіку, фізику та те, як об’єкти відображатимуться на екрані гравця.

На цьому етапі зазвичай створюються прототипи персонажів, середовища, інтерфейсів, схем керування та інших елементів гри, щоб побачити, як вони виглядають, почуваються та взаємодіють один з одним. По суті, це момент «давайте подивимося, з чим працюємо», перш ніж перейти до етапу виробництва.

Більшість часу, зусиль і ресурсів витрачається на розробку ігор на стадії виробництва. Це також один із найскладніших етапів розробки ігор. Під час цього процесу:

- моделі персонажів розробляються, відтворюються та повторюються, щоб виглядати саме так, як вони повинні виглядати в історії;
- аудіодизайн працює невпинно, щоб кожен раз, коли персонаж ступає на пісок, гравій або цемент, він звучить автентично;
- дизайнери рівнів створюють середовища, які є динамічними, захоплюючими та підходять для багатьох типів стилів гри;
- актори озвучування читають великі стоси сценаріїв, дублюючи дубль за дублем, щоб отримати потрібну емоцію, час і тон;
- розробники пишуть тисячі рядків вихідного коду, щоб оживити кожну частину вмісту в грі;
- керівники проєкту встановлюють етапи та графіки спринтів, забезпечуючи відповідальність кожного відділу та членів його команди (це особливо важливо, якщо видавець регулярно перевіряє оновлення статусу).

Ці та багато інших подій можуть зайняти роки ітерацій, щоб стати правильними, і це за умови, що попутно буде внесено лише кілька змін, що навряд чи є реальністю. У розробці ігор нерідкі випадки, коли цілі сегменти гри – місяці роботи – викидаються на скасування після її завершення. Розробники можуть собі уявити, як це засмучує залучених працівників. Ці типи змін зазвичай розглядаються на етапі тестування.

Кожну функцію та механіку в грі необхідно перевірити на контроль якості. Гра, яка не пройшла ретельного тестування, – це гра, яка навіть не готова до випуску альфа-версії. Ось деякі речі, які тестувальник може вказати на цьому етапі:

– Чи є області або рівні з помилками?
– Чи все відображається на екрані?
– Чи можна пройти через цю стіну чи замкнене середовище?
– Чи є функції, які можна використовувати для використання гри?
– Чи даний персонаж назавжди застрягне в цьому місці?
– Діалог персонажа застарілий і нудний?

Існують навіть різні типи плейтестерів. Деякі плейтестери проводять стрес-тести, натикаючись на стіни сотні, якщо не тисячі разів, намагаючись «зламати» гру. Інші плейтестери проводять тести на «фактор веселощів», щоб визначити, чи гра занадто складна чи надто легка, або проходять повну гру, щоб перевірити, чи вона була достатньо задовільною.

Після незліченних годин тестування та повторення гра має бути готова до випуску пізньої альфа-версії або навіть бета-версії, залежно від того, наскільки вдосконалені ігрові функції. Це перший раз, коли публіка отримає в свої руки гру.

Етап підготовки до запуску (передзапуск) – це напружений час для ігрових студій. Питання сумнівів можуть просочуватися, коли розробник думає, як публіка відреагує на перший функціональний продукт.

Перш ніж буде випущена офіційна бета-версія, гра потребує певного маркетингу. Адже як інакше люди про це дізнаються.

Видавці майже завжди очікують, що рекламне відео з поєднанням кінематографії та зразкового геймплея приверне увагу. Вони також можуть запланувати місце на одній із головних ігрових конференцій, як-от E3 або PAX, для ексклюзивного попереднього перегляду гри.

Незалежні студії не завжди можуть дозволити собі солідні маркетингові бюджети, щоб привернути увагу до своїх ігор. На щастя, краудфандинг і

реклама можуть бути такими ж плідними. Надсилання копій бета-версії з раннім доступом провідним гравцям онлайн-ігор, щоб вони могли транслювати свою аудиторію в прямому ефірі, є поширеним методом для незалежних студій [11].

Час до запуску гри, здебільшого витрачаються на придушення великих резервів помилок – деяких старих, деяких нових, знайдених на етапі тестування. Для ігор із багатьма помилками студія створить ієрархію помилок, яку потрібно придушити. Ця ієрархія включатиме помилки, що призводять до збою гри, у верхній частині та незначні помилки внизу.

Окрім усунення помилок, розробники зазвичай максимально вдосконалюють гру перед її запуском. Можливо, той гірський масив може мати більшу глибину. Можливо, шкіряні ремінці персонажа можуть бути більш фактурними. Давайте нарешті змусимо ці дерева колихатися на вітрі. Ці типи змін, хоч і незначні, можуть бути важливими для того, щоб зробити гру більш захоплюючою.

Після запуску – один із найцікавіших періодів для будь-якої ігрової студії. Роки наполегливої праці нарешті окупилися, і продажі відеоігор (сподіваємося) ллються. Але навіть зараз ще є над чим працювати.

Це не рідкість, коли відеоігри запускаються з пакетами дрібних помилок. Перші кілька місяців на етапі після запуску зазвичай витрачаються на виявлення та придушення цих помилок. Ігрові студії також покладаються на те, що гравці надсилають звіти про помилки або говорять про помилки на онлайн-форумах. Усе це є частиною підтримки після запуску.

Інша частина післязапуску – регулярне оновлення програмного забезпечення для гри. Ці оновлення варіюються від патчів для балансування гри до нового завантажуваного вмісту або DLC.

Випуск свіжого вмісту є звичайним явищем у сучасній ігровій індустрії, оскільки це підвищує цінність відтворення та привабливість гри. Нові рівні, сюжетні лінії та багатокористувацькі режими – це лише деякі з багатьох варіантів DLC, які може вивчити ігрова студія.

Розробка гри – це бурхливий процес навіть для найдосвідченіших ігрових студій із сотнями співробітників. Але розуміння припливів і відпливів кожного етапу має вирішальне значення для побудови ретельної та відшліфованої гри.

Також важливо розуміти, що немає двох однакових ігор, навіть від однієї студії. Під час розробки ігор блокпости неминучі, дедлайни будуть пропущені, а інструменти матимуть свої обмеження. Така природа індустрії, і саме тому хороші студії відрізняються від найкращих, незважаючи на розмір компанії, завдяки лідерам і режисерам, які можуть виправити корабель.

Якщо розробник зацікавлений у створенні гри або бажає відшліфувати свої навички, щоб приєднатися до студії, слід ознайомитися з найкращими на сьогодні мовами програмування для ігор.

Дуже важливо, щоб перший ігровий движок був доступним і допомагав навчитися правильно створювати ігри. Хоча можуть існувати кращі ігрові движки, як-от ігрові движки, які не вимагають розуміння того, як кодувати, деякі з них не мають середовища, зручного для початківців, не мають достатньо ресурсів і, що ще гірше, можуть уповільнити ваше навчання.

Але, зрештою, вибір першого ігрового движка не має значення. Немає значення, чи розробник віддає перевагу іншим ігровим движкам; головне спробувати, навчитися і почати перший проект.

За допомогою правильних інструментів розробник може зробити будь-яку гру простою. А при наявності ідей та розробці чогось незвичайного, цікавого можна зробити гру веселішою та легшою це завжди є хорошою ідеєю під час розробки.

3 РОЗБІР ТА АНАЛІЗ ВИКОРИСТАННЯ У РОЗРОБЦІ ГОТОВОГО ТА САМОПИСНОГО ДВИЖКІВ

В останні роки розробка ігор набула більшої популярності як популярна домашня індустрія. Особи, які розробляють відеоігри для особистого задоволення або малого бізнесу, використовують цю форму мистецтва, щоб створювати унікальні та захоплюючі назви, які можуть приносити значні прибутки.

З розвитком цифрових платформ розповсюдження та легкістю, з якою ігри можна перенести на нові носії, створити успішну гру з нуля стає дедалі складніше. Є кілька доступних ресурсів, які допоможуть початківцям розробникам на цьому шляху. Проте деякі основні моменти залишаються ключовими (див.рис.11)

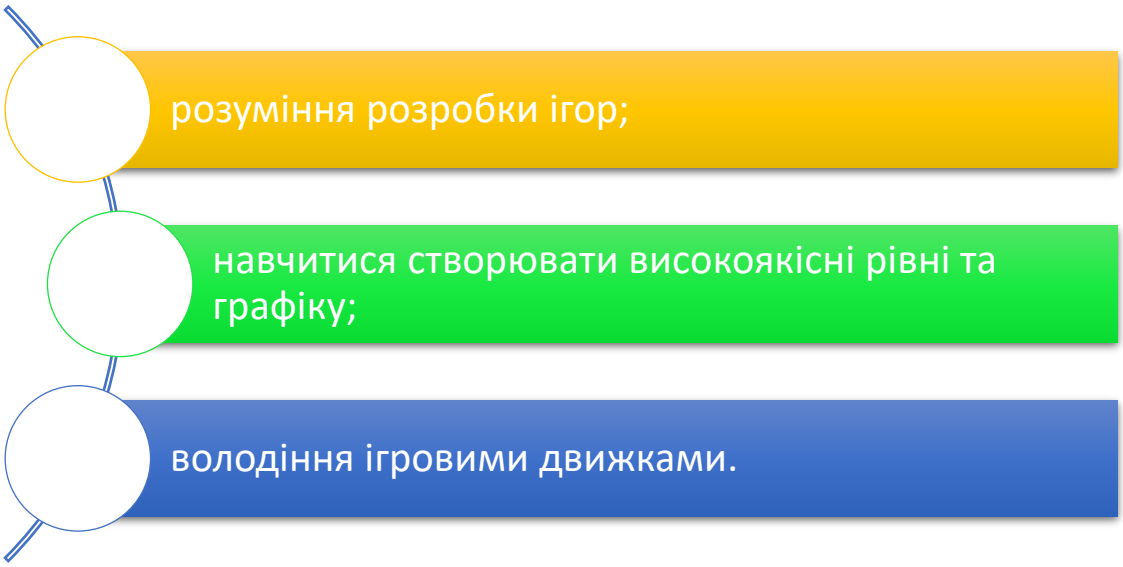

Рисунок 11 – Список фреймворків що можуть бути використані для відповідних мов програмування

Ігровий движок – це програма, яка допомагає створювати ігри. Це дозволяє розробникам створювати ігри з нуля або переробляти старі ігри для нових платформ. Ігрові механізми бувають різних форм і розмірів, але всі во-

ни мають одну спільну мету: зробити створення ігор і гру доступнішими, ніж будь-коли раніше. Доступні різні ігрові движки, кожен зі своїми сильними та слабкими сторонами.

Використання під час розробки ігрового додатка готового движка має свої переваги [12]:

– все вже зроблено за розробника – потрібно використовувати готові інструменти, щоб реалізувати свої ідеї;
– є спільнота – інші розробники, які користуються цим же движком, можуть допомогти;
– більшість движків дозволяють зв декілька кліків імпортувати гру на іншу платформу; іноді потрібно повозитися, наприклад, адаптувати керування під тачскрин або геймпад;
– є безкоштовні варіанти – скачав, і одразу приступити до розробки;
– у багатьох двигунів є магазини з готовими скриптами, моделями, ефектами та іншими корисностями.

До недоліків використання готового движка можна віднести:

– іноді можуть потрапити баги, з якими нічого не можна зробити – тільки чекати, поки автори движка щось виправлять;
– менше свободи;
– при розробці слід погодитися з ліцензією;
– автори можуть кинути або переробити улюблений движок;
– багато того, що розробнику ніколи не знадобиться, а це роздмухує розмір гри.

Використання під час розробки ігрового додатка самописного движка в свою чергу має свої переваги:

– максимальна свобода при розробці;
– заточеність під конкретні потреби: розробники самі вирішують, що буде у движку розробника;

– хороша оптимізація: відсутність зайвих модулів робить гру легшою;
– інтуїтивність: розробник орієнтуватиметься в движку із закритими очима.

До недоліків використання самописного движка можна віднести:

– займає багато часу;
– досить затратно фінансово;
– потребує великих знань;
– помилки у проєктуванні можуть поховати не тільки гру, а й сам движок;
– додаткові витрати на імпортування.

Одним з найпопулярніших движків який використовується при розробці ігрових застосунків є Unity. За допомогою мов програмування розробник писатимете для комп'ютера умови та команди: якщо А, зроби Б, а якщо В, зроби Д . Незважаючи на те, що движки беруть на себе велику частину роботи, програмувати доведеться багато.

Переміщення по меню, перехід між локаціями, керування персонажем, рух камери, зміна музики, діалоги, система квестів – все це та багато іншого потрібно буде запрограмувати. Не говорячи вже про ігровий штучний інтелект. Якщо розробник вже вибрав якийсь конкретний двигун, то й мову потрібно вибирати потрібну. Наприклад, Unity підтримуються C# і JavaScript (його модифікація, яка називається UnityScript) , а UE4 – C++. Також даний движок підтримує наступні мови програмування:

– C#;
– C++;
– Java;
– Python;
– Swift;
– Objective-C;
– JavaScript;

– PHP;

– Lua та інші.

При розробці можна використовувати мову, щоб написати гру без двигуна. Наприклад, на JavaScript створюються браузерні ігри, на C++ або C# – ігри для комп'ютерів, на Java – для пристроїв на Android, і так далі.

Для цих мов є бібліотеки для роботи з графікою, або цілі фреймворки для створення ігор. Фреймворк – це каркас, майже готовий додаток. Розробник просто дописує для цього каркаса якісь додаткові функції, підганяючи його цим під свої потреби.

Використання бібліотек чи фреймворків, поруч із написанням власного движка, дає максимальну свободу. Але у розробника зникає можливість користуватися графічним інтерфейсом, а всі налаштування та параметри доводиться писати за допомогою коду. Список фреймворків на рис.12

Рисунок 12 – Список фреймворків що можуть бути використані для відповідних мов програмування

Бібліотеки, на відміну від фреймворків, не дають майже готову програму, але надають певні інструменти. Найпростіший приклад – бібліотека Math (математика) , яка є практично у кожній мові програмування.

Використання Math дозволяє без проблем зводити числа в міру, знаходити коріння, шукати модулі, вираховувати синуси, косинуси і таке інше. Розробник не реалізує все це самостійно, а просто викликаєте потрібну функцію та передаєте їй параметри.

У геймдеві використовують складніші бібліотеки, які дозволяють працювати з графікою чи фізикою. Наприклад, графічні бібліотеки дозволяють раструвати ігрові об'єкти.

Тобто розробник не пише для відеокарти інструкцію, які пікселі виводити їй. Натомість він додає в гру спрайти (зображення) або 3D-моделі, а графічна бібліотека сама вираховує, як це має виглядати на моніторі.

До списку графічних бібліотек можна віднести:

– OpenGL;
– WebGL;
– DirectX.

В свою чергу до списку фізичних бібліотек відносяться:

– Havok;
– PhysX.

На Unity зроблено майже всі сучасні мобільні ігри, а також деякі комп'ютерні. Виділити можна такі:

– Mario Kart Tour;
– Life is Strange: Before the Storm;
– Superliminal;
– Mobile Legends: Bang Bang;
– Outlast та інші.

Велика перевага движка – підтримка C#. Це потужна, але проста мова, тому досить швидко можна заскриптувати перші проєкти. Сам движок теж нескладний, тому його використовують багато інді-розробників.

Ігровий движок – це набір програмних засобів, які оптимізують розробку відеоігор. Ці механізми можуть бути невеликими та мінімалістичними, забезпечуючи лише ігровий цикл і кілька функцій рендерингу, або бути великими та комплексними, подібними до програм, подібних до IDE, де розробники можуть створювати сценарії, налагоджувати, налаштовувати логіку рівнів, AI, проектувати, публікувати, співпрацювати , і зрештою створити гру від початку до кінця, не виходячи з движка.

Ігрові движки та ігрові фреймворки зазвичай надають користувачеві API. Цей API дозволяє програмісту викликати функції двигуна та виконувати важкі завдання, наче це чорні ящики.

Щоб по-справжньому зрозуміти, як працює цей API, розглянемо це в контексті. Наприклад, API ігрового механізму нерідко надає функцію під назвою «IsColliding()», яку розробники можуть викликати, щоб перевірити, чи стикаються два об’єкти гри чи ні. Програмісту не потрібно знати, як реалізована ця функція або який алгоритм необхідний для правильного визначення того, чи дві форми накладаються. Функція IsColliding – це чорний ящик, який робить деяку магію та правильно повертає true або false, якщо ці об’єкти стикаються один з одним чи ні. Це приклад функції, яку більшість ігрових движків надають своїм користувачам.

```
if (IsColliding(player, bullet)) {
  lives--;
  if (lives == 0) {
    GameOver();
  }
}
```

Окрім API програмування, ще одним великим обов’язком ігрового движка є апаратна абстракція. Наприклад, механізми 3D зазвичай створюються на основі спеціального графічного API, наприклад OpenGL, Vulkan або Direct3D. Ці API забезпечують абстракцію програмного забезпечення для графічного процесора (GPU).

Говорячи про апаратну абстракцію, існують також бібліотеки низького рівня (наприклад, DirectX, OpenAL і SDL), які забезпечують абстракцію та багатоплатформний доступ до багатьох інших апаратних елементів. Ці бібліотеки допомагають нам отримувати доступ і обробляти події клавіатури, рухи миші, підключення до мережі та навіть аудіо.

Існує багато безкоштовних, потужних і професійних комерційних механізмів, які розробники можуть використовувати для створення та розгортання власних ігор. . Головні причини, чому програмісти можуть вирішити створити ігровий движок з нуля:

- можливість навчання: невелике розуміння того, як працюють ігрові движки під капотом, може допомогти розвиватися як розробнику;
- контроль робочого процесу: буде більше контролю над спеціальними аспектами гри та налаштуванням рішення відповідно до потреб робочого процесу;
- налаштування: можна налаштувати рішення для унікальних вимог гри;
- мінімалізм: менша кодова база може зменшити накладні витрати, пов’язані з більшими ігровими двигунами;
- інновація: може знадобитися реалізувати щось абсолютно нове або орієнтуватися на нестандартне обладнання, яке не підтримує жоден інший механізм.

4 РОЗРОБКА ІГРОВОГО ЗАСТОСУНКУ ПІД ОС ANDROID

Ігри створюються за допомогою двигунів – набору інструментів, який дозволяє працювати з графікою, фізикою, скриптами та іншим (див.рис. 13).

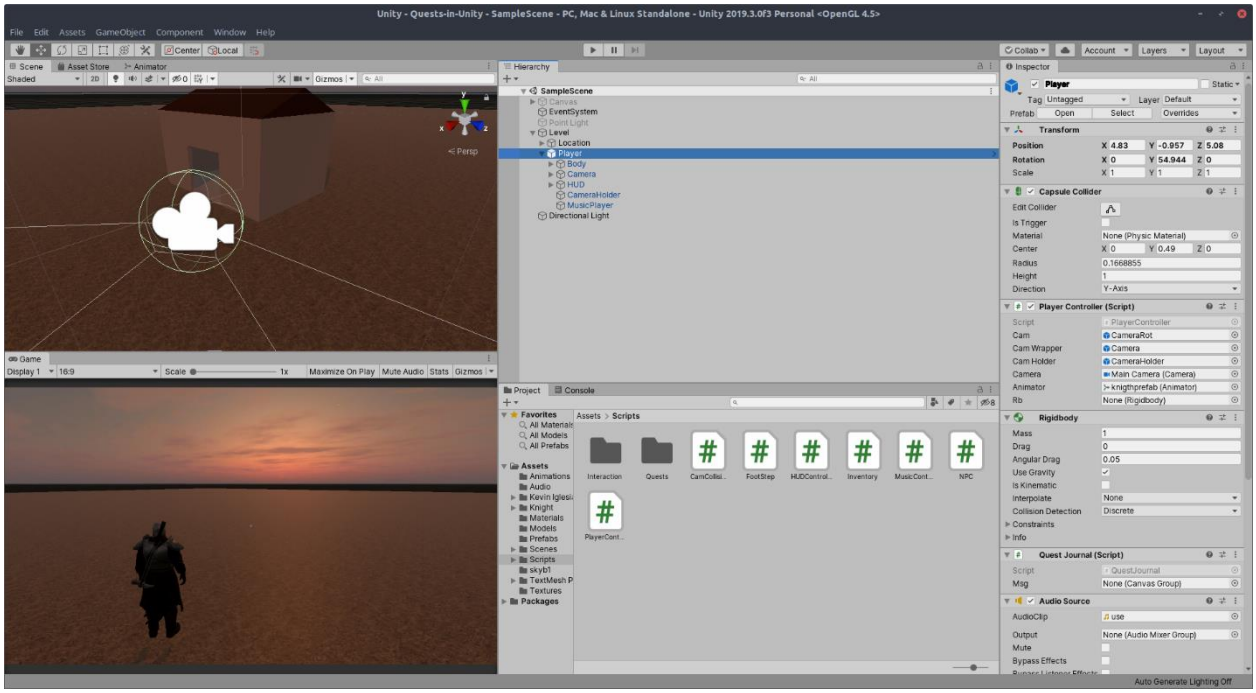

Рисунок 13 – Скріншот інтерфейсу движка Unity

У лівому верхньому кутку – ігрова сцена, на яку можна додавати об'єкти, рухати їх, прибирати і таке інше. Нижче розташоване ігрове вікно – в ньому можна побачити, як виглядатиме готова гра. Можна навіть натиснути кнопку Play і пограти.

Далі можна побачити ієрархію об'єктів на сцені, файловий менеджер та вкладку Inspector – в ній є різні налаштування для вибраного об'єкта. Крім того, можна зайти в налаштування проєкту та вказати бажані показники для гравітації, освітлення, тіней, якості графіки та іншого.

Також у движка є підтримка скриптів та API. Скрипти допомагають розробнику писати команди, які виконуватимуться грою весь час або після якихось дій гравця. API допомагає спростити написання скриптів. Тобто роз-

робник ігрового додатку не проводите складних математичних розрахунків, щоб змінити положення або обертання об'єкта – розробник просто пише команду на кшталт «Unity, rotate object A 5 degrees on the X axis».

4.1 Опис правил гри ігрового застосунку

За допомогою даного ігрового застосунку можна скоротати час, якщо їхати в громадському транспорті, чекати стоячи у черзі чи стоячи у автомобільній пробці. Правила даного ігрового застосунку досить елементарні, а сама гра є захоплюючою, з неповторним сюжетом (оскільки аналогів з даною концепціє не було знайдено).

Що стосується правил, воно одне – увійди в кураж та набери більше балів, щоб побити свій рекорд або ж навіть рекорд знайомих чи друзів. Також, даний ігровий додаток допоможе навчитися та тренуватися концентруватися під час гри, оскільки без неї –рекорди не побити.

Використовувати ігровий додаток можна як самостійно, тобто лише один користувач може скористатися пристроєм на якому встановлений даний застосунок, або ж компанією – що дозволить розігріти і підвищити інтерес та конкуренцію.

4.2 Опис етапів розробки ігрового застосунку під ОС Android

Під час розробки ігрового додатку, буде використовуватися міжплатформне середовище Unity [12].

На першому етапі розробки необхідно створити новий проєкт у Unity, за основу для простоти та зручності було обрано 2D-шаблон. Також слід вказати ім’я проєкту (рис.14).

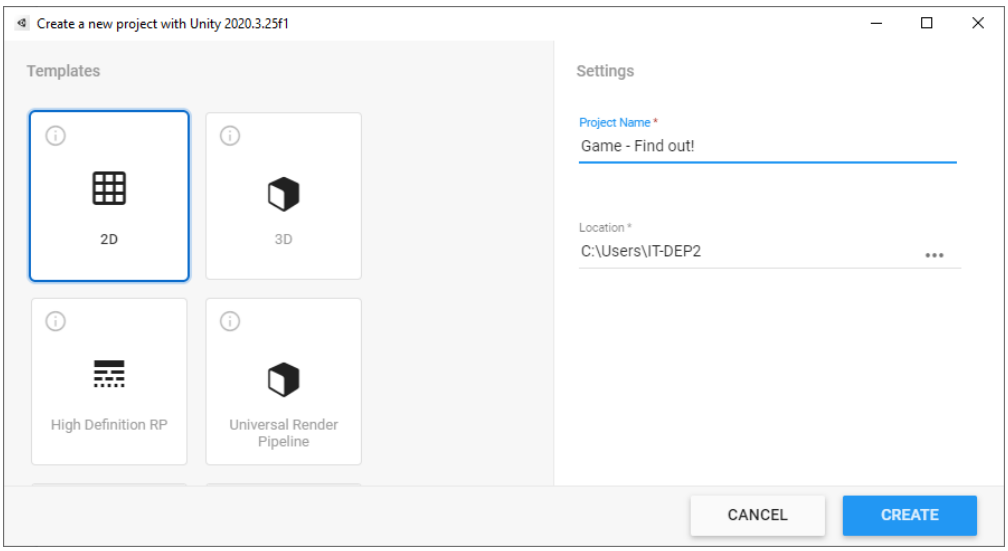

Рисунок 14 – Скріншот вікна для створення проєкту в Unity

Після натискання на кнопку Create автоматично відкривається безпосередньо середовище розробки, в якому необхідно створити папку Screen, де будуть зберігатися екрани майбутнього ігрового додатку. Також варто зазначити, що слід також обрати розміри екрану для якого буде розроблятися ігровий застосунок та платформа/ОС на якій в подальшому можна буде використовувати даний застосунок (див.рис.15).

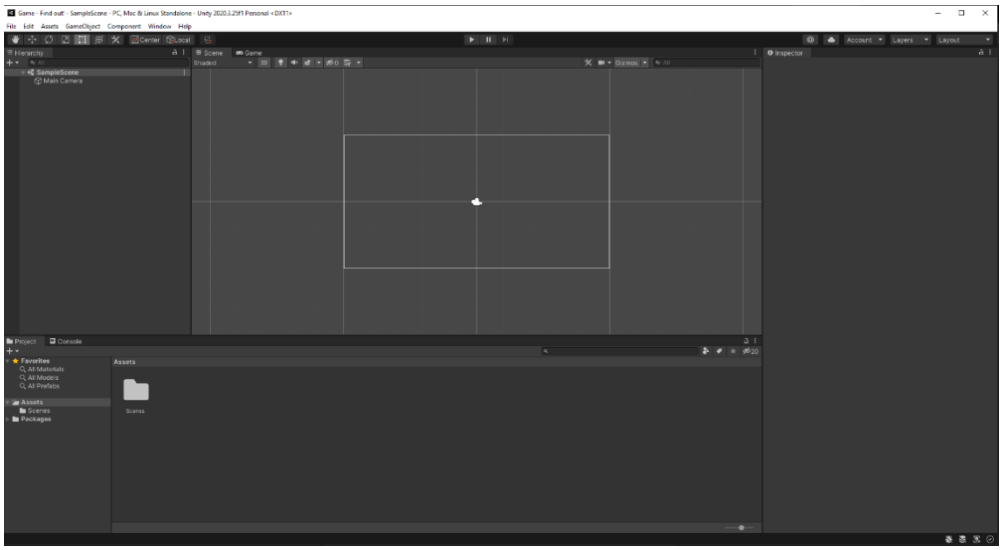

Рисунок 15 – Скріншот вікна для створення проєкту в Unity

Після створення проєкту, можна перейти до створення безпосередньо майбутнього ігрового застосунку.

В папку Sprites кореневої папки Assets додаються різні backgrounds, які можна використовувати у ігровому застосунку для створення різної складної рівнів проходження або ж варіантів локацій (див.рис.16).

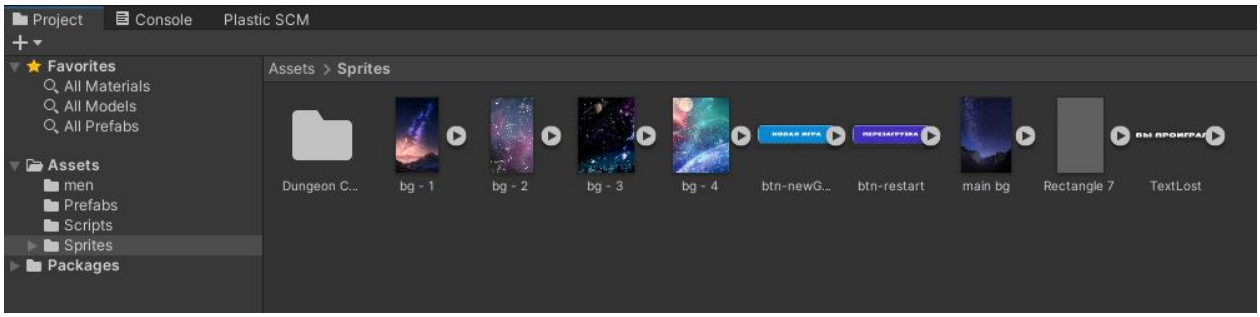

Рисунок 16 – Скріншот вмісту папки Sprites проєкту в Unity

Також було створено папку Prefabs, яка зберігає в собі безпосередньо об'єкти/персонажі гри. На даний момент в цій папці знаходиться один із персонажів, а саме літаючий монстр (рис.17)

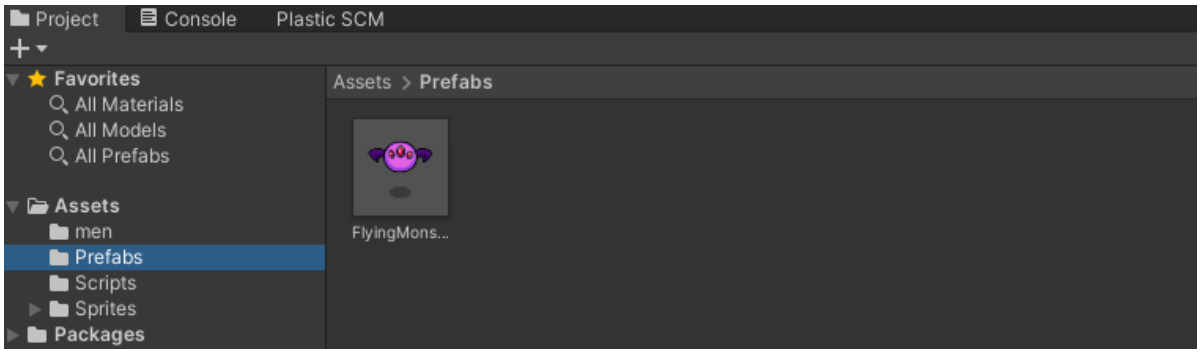

Рисунок 17 – Скріншот вмісту папки Sprites проєкту в Unity

Одна із локацій буде мати наступний вигляд (рис.18). Як можна побачити, на даному етапі проєктування на екрані користувача буде представлено

поле в якому відображається «Результат» (накопичені бали) та кнопка «Меню».

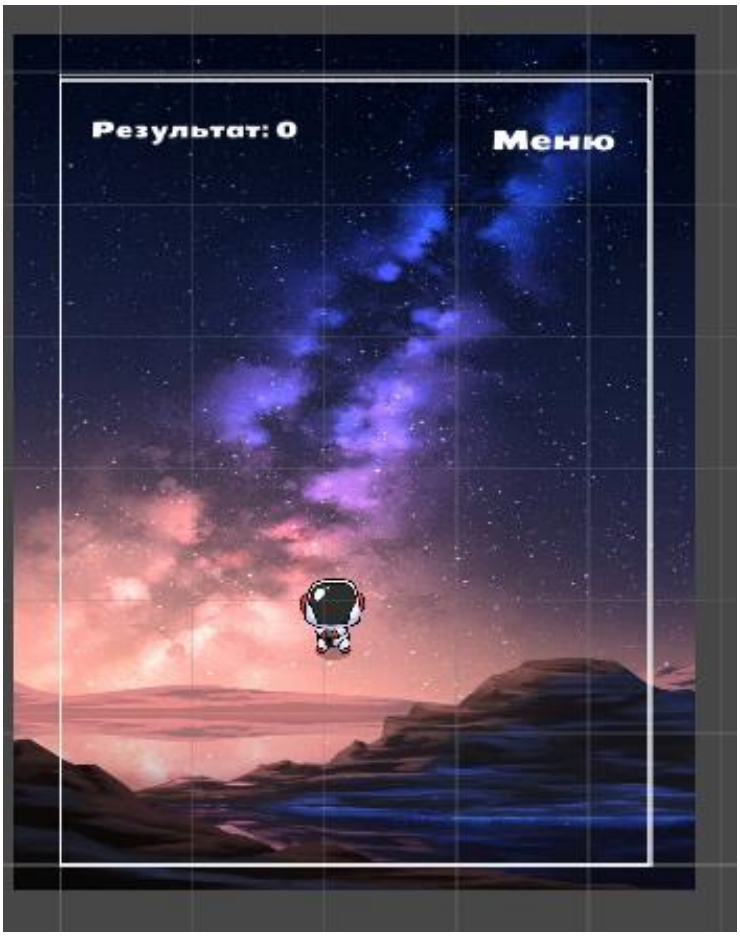

Рисунок 18 – Скріншот вмісту папки Sprites проєкту в Unity

Одним із кропітких завдань, було створення та пропрацювання можливих варіантів меню, кожен варіант відповідає своєму скрипту (див.рис.19). Передбачається, що меню має пункти які описують подальші дії в залежності від точки входу/виходу, а саме:

– головне меню;
– основне меню);
– меню програшу.

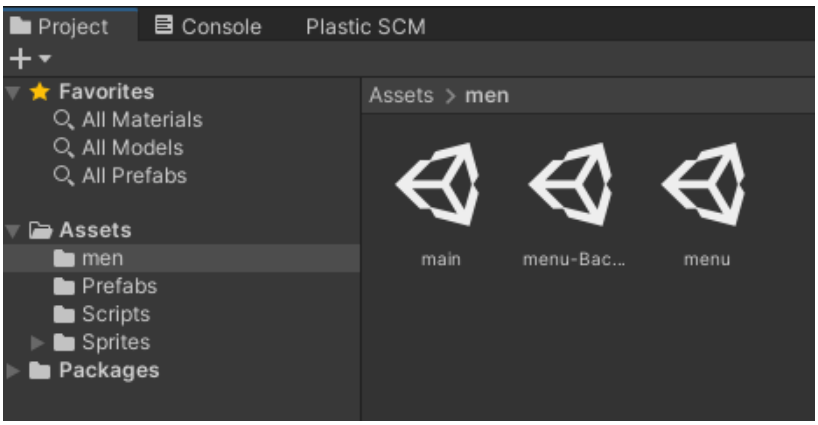

Рисунок 19 – Відображення існуючих трьох скриптів для реалізації різних екранів меню в ігровому додатку

Головне меню ігрового застосунку, що розробляється відображається у разі першого входу в дану гру (рис.20)

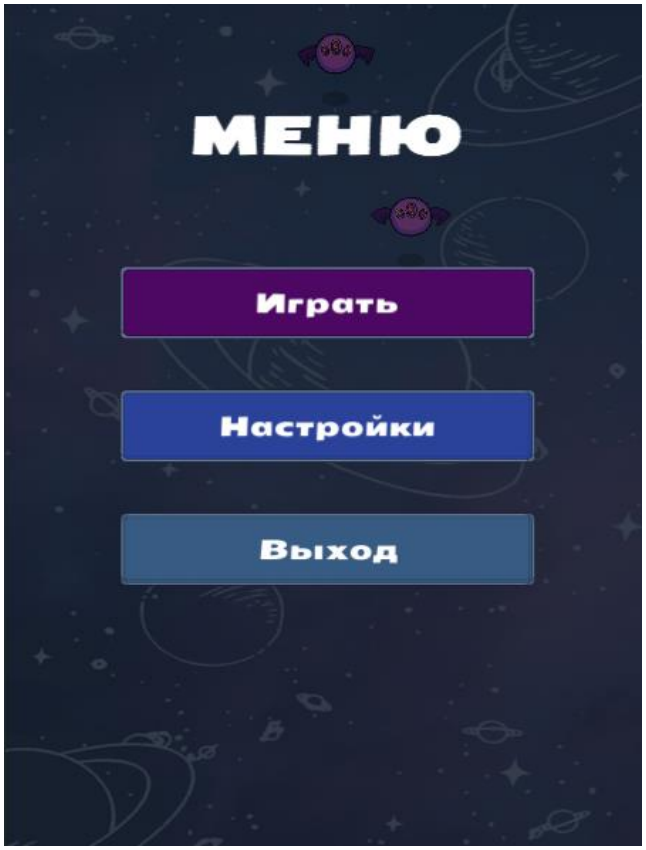

Рисунок 20 – Скріншот відображення головного меню в ігровому додатку

Основне меню (рис. 21) в свою чергу відображається користувачеві тільки у разі, якщо з головного екрану ігрового застосунку безпосередньо під час гри, натиснути на кнопку Меню в правому верхньому кутку.

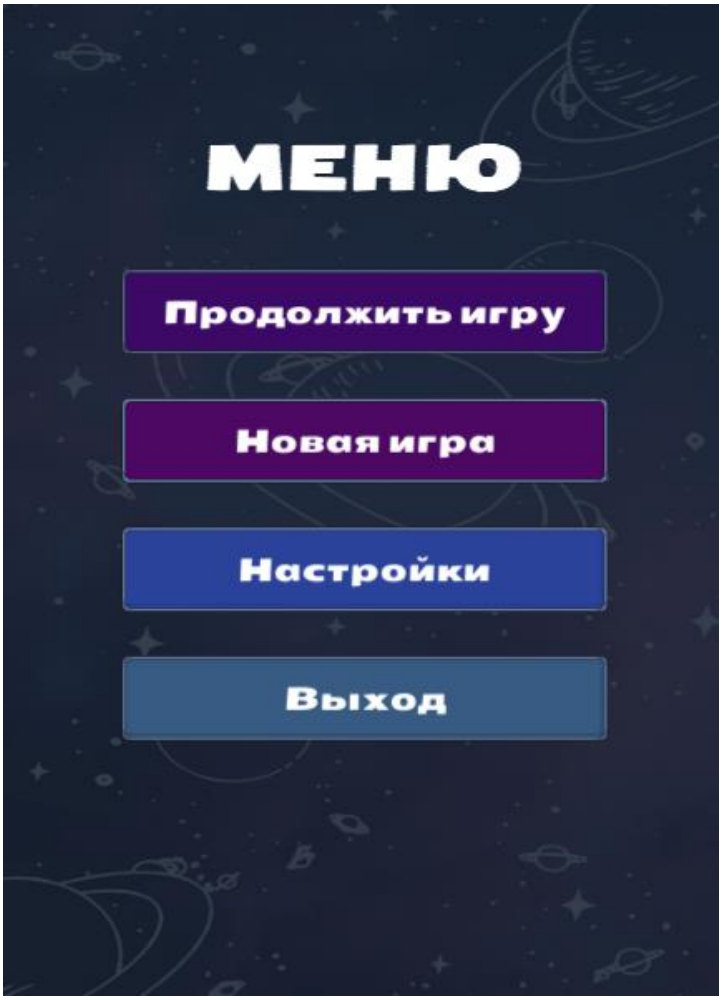

Рисунок 21 – Скріншот відображення основного меню в ігровому додатку

Третє меню, яке передбачено при розробці даного ігрового застосунку – це меню програшу, з назви зрозуміло, що дане меню користувач побачить на своєму екрані у разі програшу (див.рис. 22).

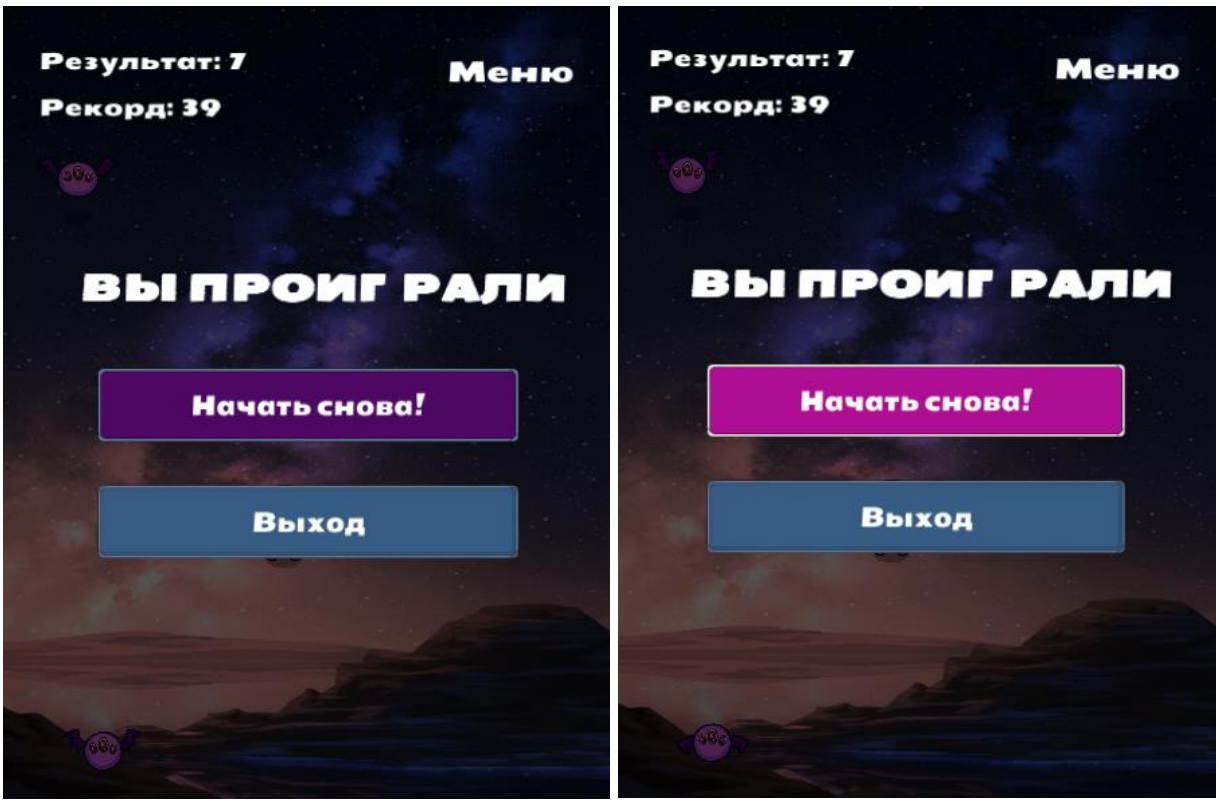

Рисунок 22 – Скріншот відображення меню програшу в ігровому додатку

Як можна побачити, з скріншотів для зручності, а також щоб здивувати та/або запам’ятатися користувачеві було вирішено додати динаміки в гру. А саме, це дуже добре можна побачити/прослідкувати під час вибору елементів будь-якого пункту меню. Також слід звернути увагу на те, що крім написання скриптів та налаштувань їх на таких кнопках меню як:

– грати;
– налаштування;
– продовжити гру;
– нова гра;
– почати знову;
– вийти.

Для пункту меню «Налаштування» було передбачено окремий екран, на якому можна буде обрати та змінити необхідні параметри налаштувань (див.рис. 23)

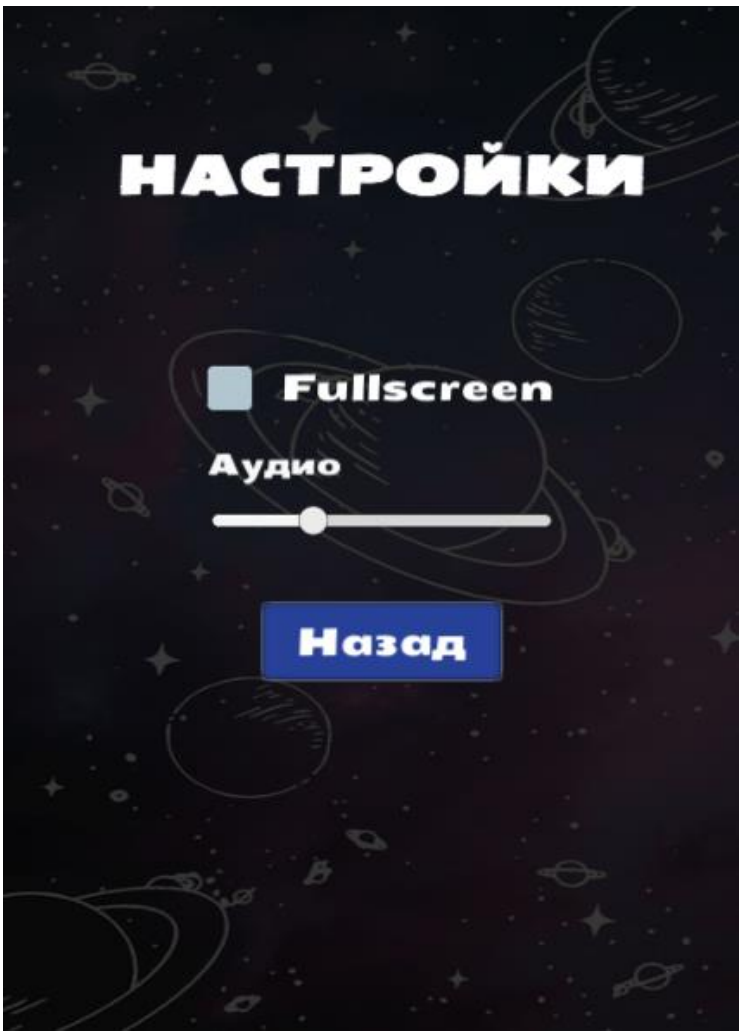

Рисунок 23 – Скріншот відображення окремого екрану Налаштувань в ігровому додатку

Як можна побачити, даний екран складається з таких елементів, як повномасштабний екран та гучність аудіо, також передбачена кнопка «Назад», яка повертає користувача до попереднього меню.

Для того, щоб підв’язати відповідні методи із скриптів до необхідних/конкретних кнопок меню, слід вказати вибрати потрібну кнопку і у вікні Inspector знайти поле On Click (). В ньому ж описати потрібні дії які будуть виконуватися при натисканні на кнопку (рис. 24).

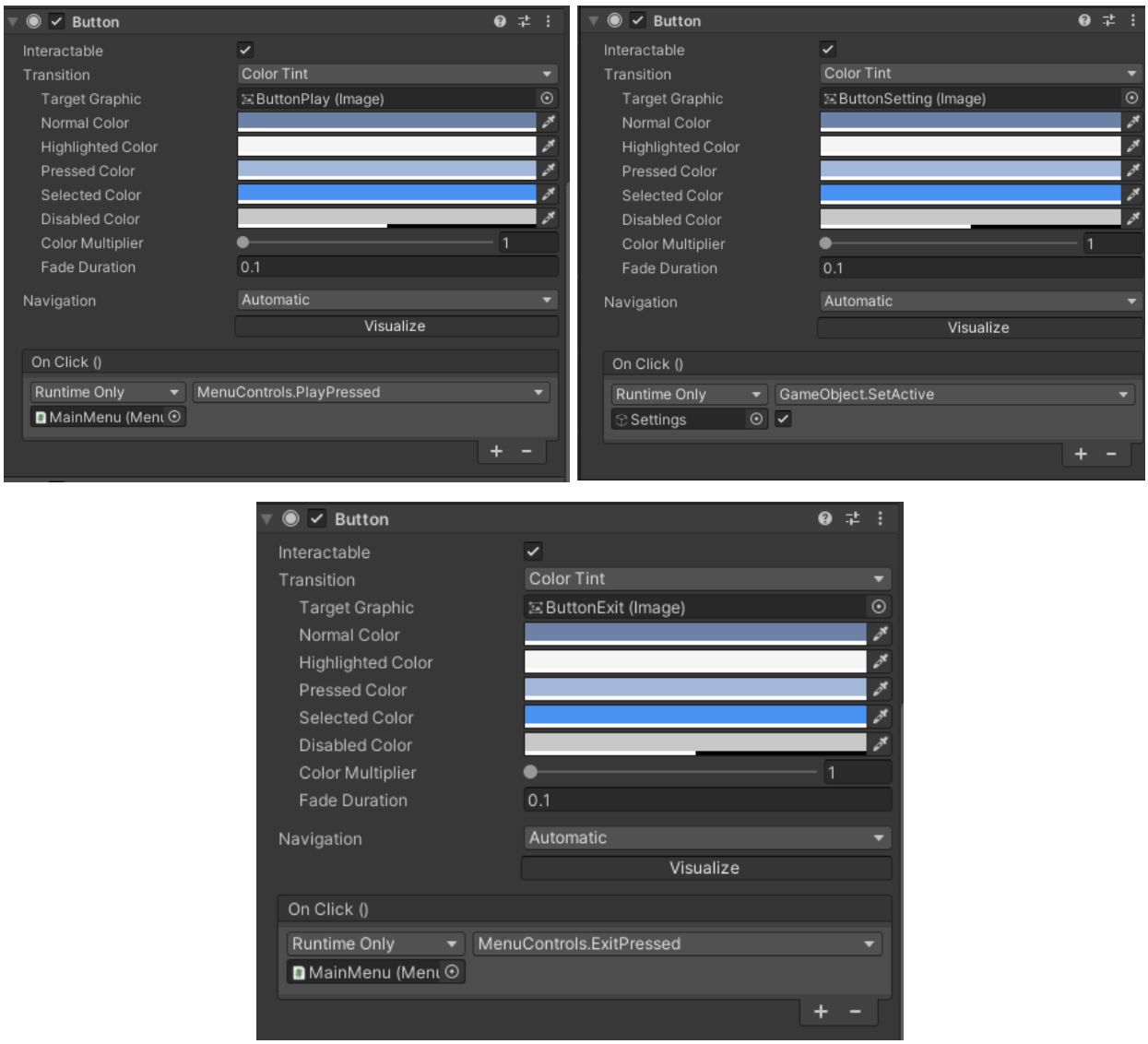

Рисунок 24 – Скріншот відображення окремого екрану Налаштувань в ігровому додатку

Просто заходити та грати в розроблений ігровий застосунок, при цьому ще й запам’ятовувати свій результат не є досить цікавим. Тому, на головному екрані було розміщено ще одне поле, яке в свою чергу зберігає дані про поставлений рекорд. Тому, навіть якщо на одному пристрої граються декілька осіб, можна буде змагатися між собою (рис.25).

Рисунок 25 – Скріншот головного екрану з доповненням в ігровому додатку

Unity пропонує відразу два способи зберігати ігрові дані – простіше та складніше. Найпростіший – вбудована система PlayerPrefs. Встановлюєте значення ключа, натискаєте Save – і все готово. Складніший – серіалізація даних та запис у файл для подальшого використання [12].

Обидва методи мають переваги і недоліки, тому для конкретного випадку важливо вибрати правильний варіант.

Тому, далі слід додати два скрипти – SavePrefs та SaveSerial – для реалізації двох методів. Щоб створити скрипт, варто клацнути правою кнопкою миші у вікні Assets і виберіть пункт Create -> C# Script (рис.26).

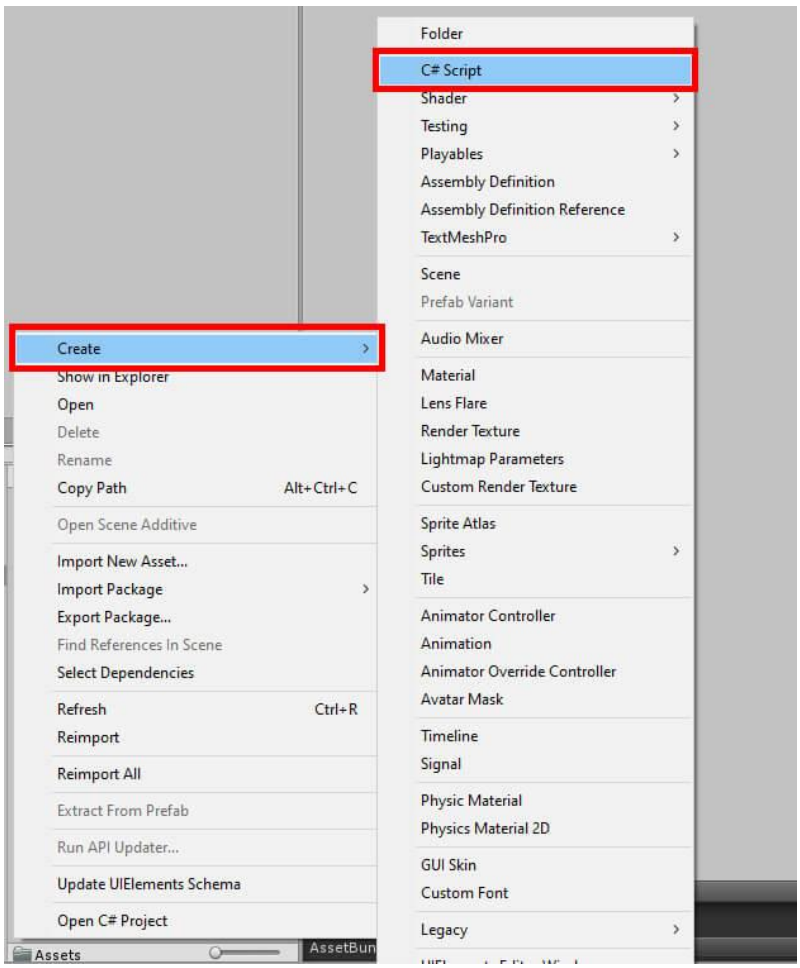

Рисунок 26 – Скріншот вікна для створення C# Script в Unity

Було вирішено розпочати з використання простого способу, а саме SavePrefs. Для початку слід оголосити змінні, які будуть використовуватися для збереження:

```
int intToSave;
float floatToSave;
string stringToSave = "";
```

За допомогою методу OnGui створюється інтерфейс користувача для візуального управління цими змінними.

Дві кнопки – для збільшення значень intToSave та floatToSave. Текстове поле – для змінної stringToSave. Декілька лейблів для відображення пото-

чних значень змінних. Три кнопки дій, щоб зберегти, завантажити та скинути дані.

Далі необхідно створити метод SaveGame, який безпосередньо буде відповідати за збереження даних

```
void SaveGame(){
  PlayerPrefs.SetInt("SavedInteger", intToSave);
  PlayerPrefs.SetFloat("SavedFloat", floatToSave);
  PlayerPrefs.SetString("SavedString", stringToSave);
  PlayerPrefs.Save();
  Debug.Log("Game data saved!");
}
```

Для збереження даних з PlayerPrefs потрібно лише кілька рядків коду. Тут слід встановити ключі налаштування ("SavedInteger" або "SavedFloat") та їх значення передати у відповідні методи об'єкта PlayerPrefs. Після того, як всі потрібні дані записані, зберегти їх, викликавши метод PlayerPrefs.Save. Для перевірки, можна вивести повідомлення в консоль налагодження, про те, що операція успішно виконана.

Завантаження збережених даних – це, власне, збереження навпаки. Необхідно взяти значення, що зберігаються в PlayerPrefs і записати їх у змінні.

Хорошою практикою є попередня перевірка наявності потрібних ключів. Для цього призначений метод HasKey. Достатньо перевірити хоча б один ключ із встановлених у методі SaveGame, щоб розуміти, чи є у користувача збережені дані. Якщо даних немає, можна вивести повідомлення про помилку в консоль.

```
void LoadGame() {
  if (PlayerPrefs.HasKey("SavedInteger"))   {
    intToSave = PlayerPrefs.GetInt("SavedInteger");
    floatToSave = PlayerPrefs.GetFloat("SavedFloat");
    stringToSave = PlayerPrefs.GetString("SavedString");
    Debug.Log("Game data loaded!");
  }
```

```
    else
      Debug.LogError("There is no save data!");
  }
```

Для видалення всіх даних, що зберігаються у PlayerPrefs, слід використовувати метод PlayerPrefs.DeleteAll.

```
  void ResetData() {
    PlayerPrefs.DeleteAll();
    intToSave = 0;
    floatToSave = 0.0f;
    stringToSave = "";
    Debug.Log("Data reset complete");
  }
```

У методі ResetData необхідно очистити сховище, а також обнулити всі змінні. Тепер можна перевіри весь код у справі. Для цього варто зберегти файл і повернутися до редактора Unity. Прикріпити скрипт SavePrefs до потрібного об'єкта, наприклад, до Main Camera (рис.27).

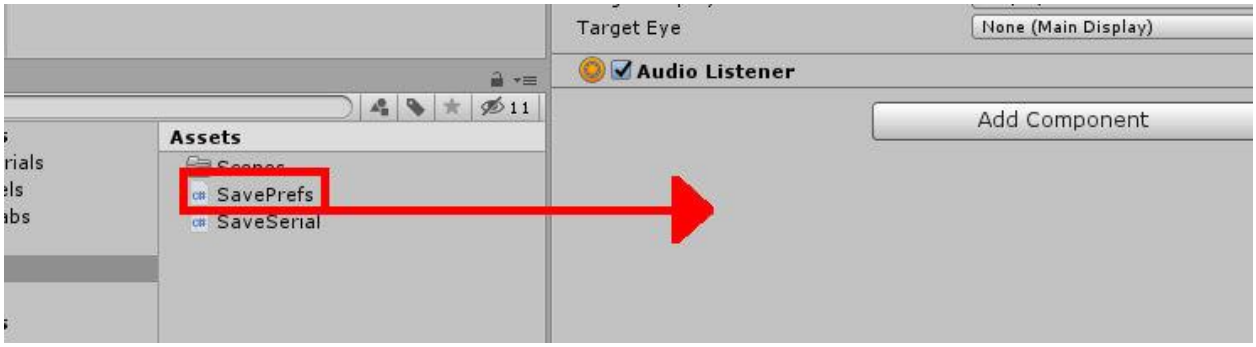

Рисунок 27 – Скріншот вікна з відображенням описаного процесу в Unity

Недоліками цього способу, самі по собі спосіб здається простим та ефективним. Але завжди використовувати PlayerPrefs для збереження даних користувача не вийде через його невисоку безпеку. Розробник точно не захо-

че зберігати таким способом дані, в які не повинен втручатися гравець: наприклад, кількість доступної гравцеві ігрової валюти або статистика.

З назви зрозуміло, що PlayerPrefs призначений для зберігання власних уподобань та інших неігрових даних. Наприклад, цей метод ідеально підходить для запису налаштувань інтерфейсу: колірної теми, розміру елементів тощо. Інша проблема – у недостатній гнучкості. У PlayerPrefs можна зберігати лише числа та рядки, тому він не підходить для даних, що мають складну структуру.

На щастя, є ще один спосіб, більш гнучкий та безпечний. Для демонстрації складного способу збереження даних у Unity слід відкрити скрипт SaveSerial.

Для серіалізації даних потрібно додати кілька директив using:

```
using System;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
```

Створений новий серіалізований клас SaveData, який в свою чергу буде містити дані, що зберігаються.

```
[Serializable]
class SaveData {
  public int savedInt;
  public float savedFloat;
  public bool savedBool;
}
```

Три змінні у класі SaveData відповідають змінним із класу SaveSerial. Для збереження необхідно передавати значення з SaveSerial до SaveData, а потім серіалізувати останній. Далі до класу SaveSerial додається метод SaveGame:

```
    void SaveGame() {
      BinaryFormatter bf = new BinaryFormatter();
      FileStream                    file                    =
File.Create(Application.persistentDataPath + "/MySaveData.dat");
      SaveData data = new SaveData();
      data.savedInt = intToSave;
      data.savedFloat = floatToSave;
      data.savedBool = boolToSave;
      bf.Serialize(file, data);
      file.Close();
      Debug.Log("Game data saved!");
    }
```

Об'єкт BinaryFormatter призначений для серіалізації та десеріалізації. При серіалізації він відповідає за перетворення інформації в потік бінарних даних (нулів та одиниць).

FileStream і File необхідні для створення файлу з розширенням .dat. Константа Application.persistentDataPath містить шлях до файлів проєкту: C:\Users\[user]\AppData\LocalLow\[company name].

У методі SaveGame створюється новий екземпляр класу SaveData. До нього записуються поточні дані із SaveSerial, які потрібно зберегти. BinaryFormatter серіалізує ці дані та записує їх у файл, створений FileStream. Потім файл закривається, у консоль виводиться повідомлення про успішне збереження.

Завантаження даних відбувається за допомогою методу LoadGame – який дозволяє відновити дані як і раніше, SaveGame навпаки:

```
    void LoadGame() {
      if  (File.Exists(Application.persistentDataPath          +
"/MySaveData.dat")){
        BinaryFormatter bf = new BinaryFormatter();
        FileStream file =
          File.Open(Application.persistentDataPath
          + "/MySaveData.dat", FileMode.Open);
        SaveData data = (SaveData)bf.Deserialize(file);
```

```
    file.Close();
    intToSave = data.savedInt;
    floatToSave = data.savedFloat;
    boolToSave = data.savedBool;
    Debug.Log("Game data loaded!");
  }
  else
    Debug.LogError("There is no save data!");
}
```

Спочатку відбувається пошук файлу із збереженими даними, який був створений у методі SaveGame. Якщо файл існує, слід відкрити його та десеріалізувати за допомогою BinaryFormatter. Далі, передати записані в ньому значення змінні класу SaveSerial. Потім вивести в консоль налагодження повідомлення про успішне завантаження. А якщо файл з даними не виявлено в папці проєкту, вивести повідомлення про помилку в консоль.

Далі необхідно, реалізувати метод для скидання збережених даних. Він схожий на ResetData, який був написаний для очищення в PlayerPrefs, але включає пару додаткових кроків. Спочатку потрібно переконатися, що файл, який необхідно видалити, існує. Після його видалення обнулити значення змінних класу SaveSerial до дефолтних та вивести повідомлення у консоль. Якщо ж файлу немає, вивести повідомлення про помилку.

```
void ResetData() {
  if (File.Exists(Application.persistentDataPath
    + "/MySaveData.dat")) {
    File.Delete(Application.persistentDataPath
      + "/MySaveData.dat");
    intToSave = 0;
    floatToSave = 0.0f;
    boolToSave = false;
    Debug.Log("Data reset complete!");
  }
  else
    Debug.LogError("No save data to delete.");
}
```

Окрім створення скриптів для збереження результатів, було розроблено і інші, а саме скрипт, який відповідає за меню, переміщення головного героя та інших об’єктів, опис головного персонажа, налаштування і тд (рис.28).

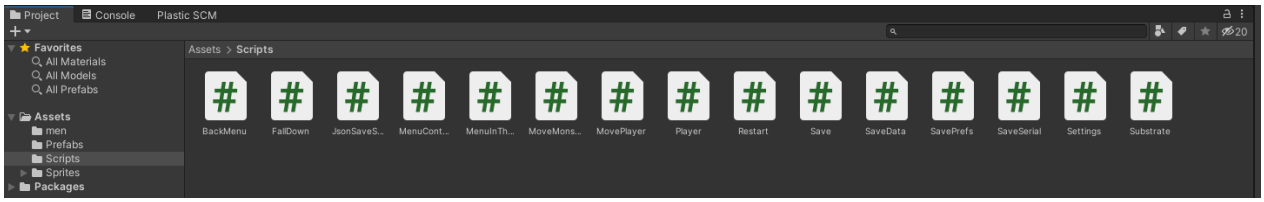

Рисунок 28 – Відображення існуючих скриптів що використовуються в ігровому додатку

Наступний етап (після збереження проєкту), який необхідно виконати для того щоб запустити даний проєкт в продакшн –це безпосередно перейти до Android Studio виконати певні маніпуляції та зберегти проєкт в форматі, який підтримують пристрої з ОС Android (APK-файл).

ВИСНОВКИ

Ігрова демографічна ситуація змінилася. Пересічний гравець більше не є стереотипним підлітком. Сьогодні в ігри грає майже кожна демографічна група суспільства. Середній вік гравця мобільних ігор близько 36 років. 51% – жінки, 49% – чоловіки. Одна третина всіх гравців мобільних ігор у віці від 35 до 50 років.

Процеси створення ігрового додатку геймдев-студією і інді-розробником-початківцем кардинально відрізняються. Такі етапи розробки ігрового додатку, як збір аналітики, плейтест та софт-лонч, початківцям в даному направленні будуть скасовані. Важливо знайти ідею, розвинути її і оформити в документ з легкою для сприйняття структурою; краще зробити кілька прототипів, щоб зайвий раз переконатися: механіка зрозуміла і все працює так само круто, як і у фантазії/уявленні розробника; використовувати бібліотеки готових безкоштовних об'єктів та звуків, а також інші допоміжні інструменти – нормально, особливо якщо це перша/друга/третя гра і тільки освоюється нова сфера; вивчення основ ігрової розробки на двигунах для візуального програмування, перш ніж занурюватися в нетрі коду.

Unity – це кросплатформний ігровий движок, який використовується багатьма іграми в Google Play Store. Модульні інструменти Unity допомагають створювати та створювати надзвичайно захоплюючі 2D або 3D мобільні ігри.

Одна з найважливіших речей у грі – збереження даних користувача: налаштувань та ігрових результатів. Колись ігри були короткими, і зберігати там було особливо нічого. У кращому випадку гра записувала найвищий бал для рейтингу. Але технології стрімко розвивалися, і геймдев не залишався осторонь. Зараз необхідно зберігати тонни даних, включаючи прогрес та статистику гравця. Ця потреба актуальна як ігор із величезним відкритим світом, так простих лінійних пригод. Найчастіше потрібно кілька сесій, щоб завершити гру, отже, потрібний спосіб збережеться і повернутися пізніше до

того моменту. Навіть для простих ігор може бути корисно записувати якусь інформацію.

Збереження даних – важлива частина більшості ігор, за винятком лише деяких специфічних винятків (наприклад, ігри з дуже короткими ігровими сесіями чи механікою перманентного завершення гри, у якій після кожного програшу вся гра скидається на початок). Варто пам'ятати, що PlayerPrefs – це простий спосіб зберегти налаштування та переваги користувача, але його не слід використовувати для запису важливих ігрових даних. Серіалізація та запис у файл – більш складніший шлях, але він в свою чергу дає більшу гнучкість та безпеку.

Під час виконання даної магістерської кваліфікаційної роботи, було проаналізовано існуючі та популярніші game engines (ігрові движки), які використовуються під час розробки ігрового додатку. Також було описано основні етапи проєктування ігрового застосунку. Було розроблено мобільний ігровий застосунок під ОС Android з використанням міжплатформного середовища Unity. Даний застосунок, простий у використанні, має зручний та зрозумілий інтерфейс для користувача. Ігровий застосунок, допоможе скоротати час поки користувач знаходиться в черзі, пробці або ж просто на обідній перерві хоче відпочити. Застосунок, орієнтований на людей будь якого віку, також він допоможе підвищити концентрацію та увагу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

[1] Статистика мобильных приложений 2021 года: популярность загрузок, тренды. URL: https://allretail.ua/ru/analytics/75763-statistika-mobilnyh-prilozheniy-2021-goda-populyarnost-zagruzok-trendy-nablyudenie-zakaz-ua (дата звернення 10.09.2022)

[2] Solar2D Game Engine. URL: https://solar2d.com/ (дата звернення 15.09.2022)

[3] Defold. Free and Developer-Friendly. URL: https://defold.com/open/ (дата звернення 02.10.2022)

[4] Ren'Py Documentation. URL: https://www.renpy.org/doc/html/language_basics.html (дата звернення 13.10.2022)

[5] O Scratch. URL: https://scratch.mit.edu/about (дата звернення 13.10.2022)

[6] Scratch (programming language). URL: https://en.wikipedia.org/wiki/Scratch_(programming_language) (дата звернення 13.10.2022)

[7] The world’s most open and advanced real-time 3D creation tool. Unreal Engine. URL: https://www.unrealengine.com/en-US (дата звернення 13.10.2022)

[8] Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. URL: https://unity.com/ (дата звернення 17.10.2022)

[9] Godot Engine - Free and open source 2D and 3D game engine. URL: https://godotengine.org/ (дата звернення 21.09.2022)

[9] Розробка ігор: 7 головних етапів створення мобільної free-to-play гри. URL: https://vokigames.com/ua/rozrobka-igor-7-golovnyh-etapiv-stvorennya-mobilnoyi-free-to-play-gry/ (дата звернення 27.10.2022)

[10] На чём создавать игры: что есть что в мире геймдева. URL: https://skillbox-ru.translate.goog/media/gamedev/na_chyem_sozdavat_igry/?_x_tr_sl=ru&_x_tr_tl=uk&_x_tr_hl=uk&_x_tr_pto=sc (дата звернення 04.11.2022)

[11] The 7 Stages of Game Development. URL: https://www.g2.com/articles/stages-of-game-development (дата звернення 25.09.2022)

[12] Сохранение игровых данных в Unity. URL: https://proglib.io/p/sohranenie-igrovyh-dannyh-v-unity-2020-04-17 (дата звернення 25.09.2022)

[13] Как создать меню для игры на Unity. URL: https://skillbox.ru/media/gamedev/kak_sozdat_menyu_dlya_igry_na_unity/ (звернення 24.10.2022)