

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

«Розробка гіперказуальної гри для Android-платформи»

Виконала студентка групи ЗКНк-21-1
МАЦЮК Анна Русланівна

Керівник роботи:
ЛЕВКІВСЬКИЙ Віталій Леонідович

Рецензент:
ПЕТРОСЯН Руслан Валерікович

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«25» січня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **МАЦЮК Анна Русланівна**

Керівник роботи: **ЛЕВКІВСЬКИЙ Віталій Леонідович**

Тема роботи: «Розробка гіперказуальної гри для Android-платформи»,
затверджена наказом закладу вищої освіти від «25» січня 2024 р., №23/с

Термін здачі закінченої роботи: «10» червня 2024 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): об'єктом дослідження є процес проектування та розробки гіперказуальної гри для Android платформи на ігровому рушії Unity. Предметом дослідження є використання рушія Unity для створення гіперказуальної гри.

Консультанти кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі, пошук та аналіз аналогічних розробок, опрацювання літературних джерел	25.01.2024 – 01.03.2023	Виконано
2	Планування структури програми	02.03.2024 – 20.03.2024	Виконано
3	Написання програмного коду	21.03.2024 – 01.05.2024	Виконано
4	Тестування роботи програми	02.05.2024 – 12.05.2024	Виконано
5	Оформлення пояснювальної записки	13.05.2024 – 31.05.2024	Виконано
6	Попередній захист	10.06.2024	Виконано

Здобувач вищої освіти **Анна МАЦЮК**
Керівник **Віталій ЛЕВКІВСЬКИЙ**

РЕФЕРАТ

Кваліфікаційна робота складається з ігрового додатку та пояснювальної записки.

Пояснювальна записка до кваліфікаційної роботи на тему «Розробка гіперказуальної гри для Android-платформи» складається з вступу, трьох розділів, висновків, списку використаної літератури та додатку. Текстова частина викладена на 46 сторінках друкованого тексту. Пояснювальна записка має 29 сторінок додатків. Список використаних джерел містить 24 найменувань і займає 2 сторінки. В роботі наведено 33 рисунки. Загальний обсяг роботи – 75 сторінок.

Під час розробки проєкту проведено аналіз ринку мобільних ігор, перевірено варіанти різних ігрових рушіїв та обрано оптимальний варіант для виконання завдання. Створено структуру програми та проведено тестування.

КЛЮЧОВІ СЛОВА: МОБІЛЬНА ГРА, ПЛАТФОРМЕР, ГІПЕРКАЗУАЛЬНА ГРА, UNITY, DRAGONBONES, ADOBE.

					КН.КР.Б – 122 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка гіперказуальної гри для Android-платформи Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.	А.Р. МАЦІЮК						4	87
Керівник	В.Л. ЛЕВКІВСЬКИЙ					Житомирська політехніка, група ЗКНк-21-1		
Рецензент	Р.В. ПЕТРОСЯН							
Н. контр.								
Зав. каф.	М.С. ГРАФ							

ABSTRACT

The qualification work consists of a game application and an explanatory note.

The explanatory note to the qualification work on the topic "Design of a hyper-casual game for the Android-platform" consists of an introduction, three sections, conclusions, a list of used literature and an appendix. The text part is laid out on 46 pages of printed text. The explanatory note has 29 pages of annexes. The list of used sources contains 24 names and occupies 2 pages. There are 33 drawings in the work. The total volume of work is 75 pages.

During the development of the project, an analysis of the mobile game market was carried out, the variants of the latest game pushers were checked, and the optimal variant for the task was determined. The program structure was created and testing was carried out.

KEY WORDS: MOBILE GPU, PLATFORM, HYPECASUAL GAME, UNITY, DRAGONBONES, ADOBE.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	7
1.1. Аналіз ігрової індустрії.....	7
1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи	14
1.3. Постановка задачі.....	20
1.4. Обґрунтування вибору інструментальних засобів	21
Висновки до першого розділу.....	25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ	26
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури додатку	26
2.2. Реалізація графічних елементів додатку.....	29
2.3. Розробка програмної частини додатку	31
Висновки до другого розділу	39
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ.....	40
3.1. Порядок встановлення та налаштування додатку	40
3.2. Інтерфейс та порядок роботи з програмним додатком	40
3.3. Тестування роботи програмного додатку.....	42
Висновки до третього розділу.....	43
ВИСНОВКИ.....	44
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	45
ДОДАТКИ.....	47

ВСТУП

Ринок мобільних ігор - це один із найбільш швидкозростаючих та конкурентних сегментів ігрової індустрії. За даними AppAnnie, у 2023 році обсяг ринку мобільних ігор склав 141,71 млрд доларів США, а до 2028 року він може досягти 300,47 млрд доларів США. Основними факторами, що сприяють зростанню ринку, є збільшення числа смартфонів майже у кожної людини він зараз під боком, покращення апаратних можливостей, поширення 5G, різноманітність жанрів та монетизаційних моделей, а також вплив пандемії COVID-19, яка посилила попит на мобільні розваги [1].

Сьогодні багато компаній зосередили свої зусилля на розробці мобільних ігор, через те, що вийти на цей ринок дуже просто багато молодих розробників намагаються досягти на ньому аудиторії. Навіть, є ігри в які заходять, щоб заповнити час “time killer” – найчастіше прості ігри з одним функціоналом, бігти вперед, стрибати вгору, “3 в ряд”.

На даний час є багато таких додатків, кожний з них в чомусь схожий на інший, і дуже важко “вистрелити”, але якщо слідкувати за трендами, або робити те що подобається, то все реально.

Актуальність проєкту полягає в тому, що в цей час все більше і більше розвивається світ мобільних пристроїв й з ним мобільних ігор. Зараз у кожної людини є мобільний телефон і потреба в мобільних додатках з кожним днем тільки збільшується.

Метою кваліфікаційної роботи є створення гіперказуальної гри жанру платформер, яка надасть гравцям можливість насолоджуватись і розважатись завдяки простоті та легкості гри.

Об’єктом дослідження є процес проєктування та розробки гіперказуальної гри для Android платформи на ігровому рушії Unity.

Предметом дослідження є використання рушія Unity для створення гіперказуальної гри.

Методи дослідження: в роботі було використано методи об’єктно-орієнтованого проєктування, програмування.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Аналіз ігрової індустрії

Мобільна гра – це відеогра, яку зазвичай грають на мобільному телефоні. Мобільні ігри варіюються від базових (як Snake на старих телефонах Nokia) до складних (3D-ігри та ігри з доповненою реальністю) [2]. Більшість мобільних пристроїв мають обмежені системні ресурси, тому для сильно складних ігор потрібний гарний девайс, і в ігор у яких гарна оптимізація, і менша завантаженість, теоретично більший обсяг аудиторії яка може зацікавитися.

Доходи від мобільних ігрових додатків за 2023 рік зросли — до \$89,9 млрд. Ринок комп'ютерних ігор для настільних персональних комп'ютерів зріс до \$94 млрд. Консольні ігри закінчили рік — на позначці у \$52,4 млрд (більше порівняно із 2022-м на 0,3%). Передовими у зростанні за підсумками 2023-го стали США і Китай — країни, де найбільше геймерів [3].

Багато світових корпорацій починають випускати свої проєкти та версії вже створених ігор на телефони, наприклад, серія “Grand Theft Auto” розробника Rockstar Games створила мобільні версії 5 ігор (GTA 3, GTA: San Andreas, GTA: Vice City, GTA: Liberty City Stories, GTA: Chinatown Wars) і не дивно буде якщо тенденція продовжиться, і багато старих ігор побачить себе на мобільних пристроях, як мріяли деякі діти коли вперше грали в них.

Сьогодні у нас є широкий вибір мобільних ігор, а також широкий вибір класів ігор. Проте немає єдиної системи класифікації мобільних ігор. Коріння цієї проблеми полягає у фрагментації та різноманітності ігрового простору загалом. З такою великою кількістю різних ігрових архетипів, гібридів та постійних інновацій, нелегко класифікувати ігри змістовно та послідовно.

Гнучкість. Це єдиний критерій, що вимагається від нової систематики. Оскільки є десятки ігрових архетипів і нових, які з'являються досить регулярно, занадто статичний підхід швидко застаріє, наприклад за останні роки з'явилися королівські битви, різноманітні AFK grpg проєкти які можна було б окремо систематизувати. Щоб все було динамічним, додавати нові моделі до моделі має

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

бути порівняно просто. І ця трирівнева ієрархія складається з: категорії, жанру і піджанру (рис. 1.1) [4].



Рисунок 1.1 – Трирівнева ієрархія [4]

Гіперказуальна гра – це мобільна відеогра, яка не потребує великих зусиль для ігрового процесу. Особливість гіперказуальних ігор у тому, що в основному вони безплатні, мають спрощений інтерфейс користувача, не вимагають спеціального навчання або інструкції для ігрового процесу. В основному застосовується 2D-дизайн з простою колірною схемою і легкі механіки, що не вимагають особливих зусиль, часто є нескінченно-зацикленими. Гіперказуальні ігри часто сприймаються як бізнес-модель, а не як повноцінний жанр мобільних ігор. Через відсутність стійкої внутрішньоігрової економіки та відсутність ціни більшості гіперказуальних ігор дохід в основному генерується шляхом реклами [5].

Першою гіперказуальною грою, яка набула широкої популярності, була Flappy Bird, яка була завантажена понад 50 мільйонів разів і приносила близько 50 000 доларів на день на піку своєї популярності.

Траєкторія розвитку гіперказуальних ігор віддзеркалює ширші тенденції в мобільних технологіях та поведінці користувачів. Спочатку мобільні ігри були простими через обмеження ранніх смартфонів. З удосконаленням пристроїв ігри ставали складнішими, зі складною графікою та сюжетними лініями. Однак, коли розробники помітили зростальний попит на ігри, в які легко грати, але складно опанувати, стався зсув. Це призвело до народження гіперказуальних ігор. З часом ці ігри перетворилися з простої ніші на домінуючу силу в магазинах додатків. Ця еволюція зумовлена зростанням кількості мобільних користувачів, які шукають швидких розваг без зобов'язань, що їх вимагають складніші ігрові програми. Гіперказуальні ігри змінили ландшафт мобільних ігор, створивши нову модель успіху, в якій пріоритетами є доступність та короткі ігрові сесії за продуманим ігровим процесом та сюжетами.

Гіперказуальні ігри набули поширення у 2017 році на мобільних пристроях, але часто розглядаються як жанр, схожий на відеоігри 1970-х років, у яких відсутній детально опрацьований дизайн та ігровий процес.

Після, гіперказуальні ігри лідирували в топчартах магазинів мобільних ігор Google Play Store й App Store (iOS). Згідно з EEDAR, більшість користувачів мобільних відеоігор грають у багатозадачному режимі, і через їхню простоту гіперказуальні ігри стають дедалі більше популярними серед цих користувачів. У 2016 році ігрова компанія Ubisoft придбала Ketchapp (одного з ініціаторів гіперказуальних ігор). У 2017 році Goldman Sachs інвестував 200 мільйонів доларів у компанію Voodoo, яка займається гіперказуальними іграми. Привабливість гіперказуальних ігор полягає в їхній простоті, яка задовольняє універсальне бажання простого і приємного ігрового процесу. Нескладний характер цих ігор часто приховує стратегічну глибину, яку вони можуть запропонувати, дозволяючи гравцям швидко відчувати почуття досягнення. Ця простота не лише приваблює гравців, але й приносить користь розробникам,

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

зменшуючи складність і час, пов'язані з процесом розробки гри. Як наслідок, такі ігри можна швидко створювати та оновлювати, щоб йти в ногу зі споживчими тенденціями. Мінімалістичний підхід до дизайну гіперказуальних ігор також дозволяє уникнути перевантаження нових гравців, що підвищує ймовірність того, що вони захочуть спробувати нові ігри. Крім того, легкість запуску та зупинки цих ігор у будь-який момент ідеально узгоджується зі стрімким стилем життя сучасного суспільства, де дозвілля фрагментоване. Ця відповідність пояснює, чому гіперказуальні ігри зайняли таке важливе і стійке місце на ринку мобільних ігор.

Ринок гіперказуальних ігор сформувався під впливом низки ключових гравців, які сприяли розвитку цієї тенденції. Великі студії мобільних ігор, такі як Voodoo, Ketchapp та SayGames, вирізняються великою кількістю гіперказуальних ігор. Ці компанії опанували мистецтво швидких циклів розробки й мають гостре відчуття того, що робить гру захопливою і прибутковою.

Вони часто використовують підходи на основі даних для оптимізації залучення та утримання користувачів, що допомогло їм домінувати в чартах завантажень з численними хітами. Крім того, ці лідери індустрії встановили успішні партнерські стосунки з незалежними розробниками, надаючи їм ресурси та платформи для втілення їхніх ігрових ідей у життя.

Розвиваючи таланти та заохочуючи інновації в гіперказуальному просторі, ці значні гравці відіграли ключову роль у популяризації жанру та підтримці його імпульсу в конкурентному середовищі мобільних ігор.

Якщо уважніше придивитися до найпопулярніших гіперказуальних ігор, то можна виявити спільні риси, які сприяють їхньому успіху. Flappy Bird, гра, що злетіла до небес завдяки своїй простій, але складній механіці, ілюструє вірусний потенціал гіперказуальних ігор. Інший хіт, "Helix Jump" від Voodoo, зачаровує гравців своїм рухом та простим і зрозумілим геймплеем. "Crossy Road" поєднує в собі привабливість ностальгічної аркади з сучасними можливостями мобільних ігор, що забезпечує широку популярність і значну прибутковість. Ці ігри не лише досягли високих показників завантажень, але й зберегли значну кількість

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів завдяки простому геймплею, мінімалістичному дизайну та інтуїтивно зрозумілому управлінню. Вони демонструють, як один хіт гіперказуальної гри може просунути розробника до вершини слави. Вивчення цих історій успіху дає цінну інформацію про вподобання користувачів та ринкові тенденції, що має вирішальне значення для всіх, хто хоче заявити про себе на ринку гіперказуальних ігор.

Критично важливим фактором успіху гіперказуальних ігор є швидкість їхньої розробки та виходу на ринок. На відміну від складних ігор, на розробку яких можуть піти роки, гіперказуальні ігри можна створити за кілька тижнів або місяців. Такий швидкий цикл розробки дозволяє розробникам швидко реагувати на ринкові тенденції та відгуки гравців, випускаючи ігри, які одразу стають актуальними для споживачів.

Короткий час розробки також зменшує фінансові ризики, дозволяючи розробникам експериментувати з різними концепціями та механіками геймплею без значних інвестицій. Така гнучкість у виробництві та випуску означає, що розробники можуть тестувати кілька ігор на ринку, вчитися на їхніх результатах і відповідно вдосконалювати свої стратегії. Акцент на мінімалістичному дизайні та простому геймплеї не лише приваблює гравців, але й спрощує процес розробки, дозволяючи виводити нові ігри на ринок у темпі, що відповідає вподобанням мобільних геймерів, які постійно змінюються.

Успішні розробники ігор на ринку гіперказуальних ігор поділяють стратегічний підхід, який балансує між творчістю та аналітикою. Вони зосереджуються на створенні ігор, які миттєво привертають увагу, часто тестують кілька прототипів і швидко ітерації на основі відгуків користувачів. Успішні студії також приділяють значну увагу аналізу даних, щоб зрозуміти поведінку та вподобання гравців, на основі яких приймаються рішення щодо дизайну та маркетингових стратегій. Здатність швидко адаптуватися до показників ефективності та тенденцій має вирішальне значення. Наприклад, вони можуть змінити рівень складності гри, якщо дані свідчать про те, що гравці занадто швидко проходять її. Крім того, такі розробники часто мають ефективні стратегії

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

залучення користувачів, використовуючи соціальні мережі та перехресне просування з іншими іграми для залучення та утримання гравців. Поєднуючи філософію гнучкої розробки з глибоким розумінням динаміки ринку, успішні розробники ігор можуть стабільно створювати та підтримувати популярність гіперказуальних ігор у висококонкурентному середовищі.

Майбутнє гіперказуальних ігор виглядає перспективним, а прогнозовані тенденції вказують на подальше зростання та інновації в цьому жанрі. Аналітики ринку передбачають стійкий інтерес до гіперказуальних ігор завдяки їхній широкій привабливості та зростальній глобальній базі користувачів смартфонів. Однією з тенденцій, яка може набути популярності, є поєднання механіки гіперказуальних ігор з елементами інших жанрів, створюючи гібридні ігри, які пропонують багатший досвід, зберігаючи при цьому простоту. Крім того, з розвитком технологій ми можемо очікувати, що гіперказуальні ігри будуть містити складніші функції, такі як доповнена реальність, що може відкрити нові ігрові можливості та залучити ширшу аудиторію. Зростання жанру також, ймовірно, стимулюватиме подальші інновації в монетизації, а розробники знаходитимуть нові способи отримання прибутку без шкоди для користувацького досвіду. Завдяки такому розвитку подій, гіперказуальні ігри залишатимуться важливою частиною мобільного ігрового ландшафту, що постійно розвивається.

Технологічний прогрес відіграватиме ключову роль у формуванні майбутнього гіперказуальних ігор. Оскільки смартфони стають потужнішими, з покращеною графікою та обчислювальними можливостями, у розробників з'являються нові можливості для інновацій у жанрі гіперказуальних ігор. Наприклад, покращена чутливість сенсорного екрана та тактильний зворотний зв'язок можуть призвести до більш захоплюючого та тактильного ігрового досвіду, навіть у простих іграх. Крім того, інтеграція штучного інтелекту може персоналізувати ігровий процес, адаптуючись до стилю та вподобань гравця, щоб зробити гру цікавою та захопл. Хмарні ігрові сервіси також можуть вплинути на ринок гіперказуальних ігор, надаючи можливість миттєвої гри без необхідності завантаження, що ще більше знизить бар'єр входу для потенційних гравців. Ці

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

технологічні досягнення в поєднанні з привабливістю гіперказуальних ігор свідчать про те, що цей жанр буде продовжувати розвиватися і захоплювати гравців по всьому світу.

Не зважаючи на зростання кількості гіперказуальних ігор, майбутнє несе як виклики, так і можливості для розробників. Одним із викликів є насиченість ринку; чим більше розробників виходить на нього, тим складніше стає виділитися на тлі інших. Це вимагає від розробників бути більш інноваційними та розуміти нюанси залучення та утримання гравців. Інший виклик - це потенційна втома гравців, якщо ігри стають занадто схожими або не розвиваються. Однак ці виклики також представляють можливості. Існує простір для інновацій в ігровій механіці та інтеграції з новими технологіями, такими як віртуальна та доповнена реальність. Зростає визнання ігор як мейнстримної форми розваг також відкриває можливості для охоплення нових аудиторій. Крім того, з розвитком рекламних моделей з'являються нові можливості для монетизації, які можуть підвищити прибутковість, зберігаючи при цьому позитивний досвід гравців. Вміння орієнтуватися в цих викликах і можливостях стане ключем до довгострокового успіху гіперказуальних ігор на ринку мобільних ігор.

Гіперказуальні ігри продовжують привертати увагу гравців та видавців завдяки своїй доступності, залученості та вірусності. Однак вони також стикаються з рядом викликів, таких як висока конкуренція, низька лояльність, зниження взаємодії з клієнтами й підвищення корупційних ризиків. Для того, щоб досягти успіху в цій ніші, розробникам необхідно стежити за трендами, тестувати ідеї, оптимізувати ігровий процес та монетизацію, а також співпрацювати з надійними партнерами [6].

Основні принципи розробки гіперказуальних ігор можна сформулювати так:

швидке прототипування та тестування. Розробники гіперказуальних ігор повинні постійно експериментувати з різними ідеями, механіками та фічами, щоб знайти ті, які будуть потрібні та залучають до аудиторії. Для цього вони

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

використовують різні інструменти та платформи, такі як Unity, Buildbox, Playable Studio та інші.

дотримання трендів і натхнення з різних джерел. Розробники гіперказуальних ігор повинні бути в курсі останніх тенденцій та мод у мобільній ігровій індустрії, а також черпати натхнення з різних джерел, таких як соціальні мережі, відео, меми, реальне життя та інші ігри

простота та доступність. Розробники гіперказуальних ігор повинні робити свої ігри максимально простими та зрозумілими для гравців, щоб вони могли почати грати одразу ж, без необхідності навчання чи інструкцій. Ігри повинні мати мінімалістичний дизайн, одноманітний колірний стиль, легку механіку та керування одним пальцем.

монетизація та аналітика. Розробники гіперказуальних ігор повинні оптимізувати свої ігри для отримання доходу від реклами, яка є основним джерелом прибутку у цьому жанрі. Одним із поширених підходів є модель freemium, для цього вони мають вибирати відповідні формати та позиції реклами, такі як відео з винагородою, банери або між сторінкові оголошення.

Ця модель добре узгоджується з казуальним характером ігор, оскільки гравці охочіше дивляться рекламу, якщо це означає, що вони можуть продовжувати грати безплатно. Крім того, деякі розробники використовують гібридну модель, поєднуючи внутрішні покупки та рекламу, щоб максимізувати дохід, залишаючи ігри доступними для всіх гравців. Також вони повинні аналізувати дані по поведінку та переваги гравців, щоб покращувати свої ігри та збільшувати їхню залученість та лояльність. Монетизація є критично важливим аспектом мобільних ігор, і розробники розробили різні стратегії для отримання доходу від своїх ігор [7].

1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи

Жанри гіперказуальних ігор - це спосіб класифікації ігор з їхньої ігрової механіки, мети, складності та стилю. Жанри гіперказуальних ігор не завжди

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

збігаються з жанрами комп'ютерних ігор, що базуються на сюжеті, тематиці, графіку та інших аспектах.

Існує безліч різних жанрів гіперказуальних ігор, які можуть комбінуватися та змінюватись. Деякі з найпопулярніших та типових жанрів гіперказуальних ігор такі:

Айдлери - ігри, в яких гравець не повинен брати активну участь в ігровому процесі, а лише спостерігати за розвитком свого персонажа, бізнесу, міста і т.д. Гравець може впливати на гру за допомогою простих дій, таких, як тапи, свайпи, апгрейди й т.д. Приклади: Idle Miner Tycoon, Cookie Clicker, AdVenture Capitalist (рис. 1.2) [8].



Рисунок 1.2 – Геймплей гри Idle Miner Tycoon [8].

Раннери - ігри, в яких гравець повинен керувати персонажем, який біжить, летить, їде або рухається іншим способом нескінченною або зацикленою трасою, уникаючи перешкод і збираючи бонуси. Гравець може впливати на гру за допомогою простих дій, таких як тапи, свайпи, тільти та ін. Приклади: Temple Run, Subway Surfers, Jetpack Joyride (рис. 1.3) [9].



Рисунок 1.3 – Геймплей гри Subway Surfers [9].

Пазли - ігри, у яких гравець має вирішувати логічні завдання, маніпулюючи різними об'єктами на екрані. Гравець може впливати на гру за допомогою простих дій, таких як тапи, свайпи, драг-енд-дропи тощо. Приклади: Candy Crush, 2048, Cut the Rope (рис. 1.4) [10].



Рисунок 1.4 – Геймплей гри Cut the Rope [10].

Аркади - ігри, в яких гравець повинен виконувати різні дії, що вимагають спритності, реакції, координації та точності. Гравець може впливати на гру за допомогою простих дій, таких як тапи, свайпи, жести тощо. Приклади: Flappy Bird, Fruit Ninja, Doodle Jump (рис. 1.5) [11].



Рисунок 1.5 – Геймплей гри Flappy Bird [11].

Розглянемо гіперказуальні ігри жанру платформер.

Doodle Jump [12]

Розробник: Limax Interactive

Платформи: iOS, Android, Windows Phone

Жанр: Платформер

Основні характеристики:

Простий і захоплюючий геймплей

Різноманітні рівні складності

Бонуси та посилення

Таблиці лідерів

Підтримка мікротранзакцій (рис. 1.6)

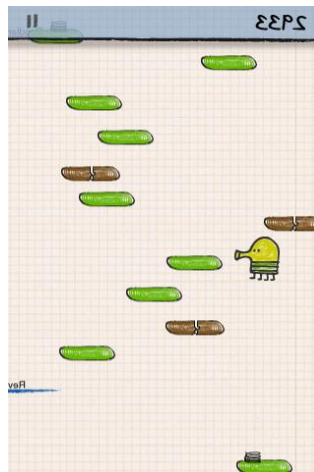


Рисунок 1.6 – Геймплей гри Doodle Jump [12]

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

Jetpack Joyride [13]

Розробник: Halfbrick

Платформи: iOS, Android, Windows Phone

Жанр: Платформер

Основні характеристики:

Раннер з елементами платформінгу

Безліч різноманітних реактивних ранців

Виконання місій

Таблиці лідерів

Підтримка мікротранзакцій (рис. 1.7)



Рисунок 1.7 – Геймплей гри Jetpack Joyride [13]

Crossy Road [14]

Розробник: Hipster Whale

Платформи: iOS, Android, Windows Phone

Жанр: Аркада

Основні характеристики:

Простий і захопливий геймплей

Різноманітні персонажі

Безліч небезпек

Таблиці лідерів

Підтримка мікротранзакцій (рис. 1.8)

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

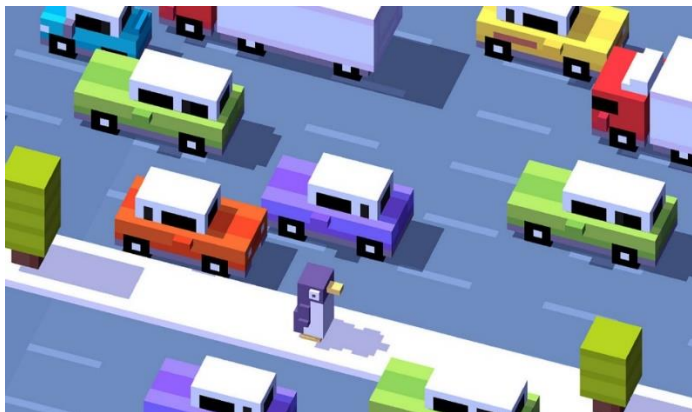


Рисунок 1.8 – Геймплей гри Crossy Road [14]

Alto's Adventure [15]

Розробник: Noodlecake Games

Платформи: iOS, Android

Жанр: Ендуро-серфінг

Основні характеристики:

Розслаблюючий і мальовничий геймплей

Різноманітні трюки

Безліч рівнів

Таблиці лідерів

Відсутність мікротранзакцій (рис. 1.9)

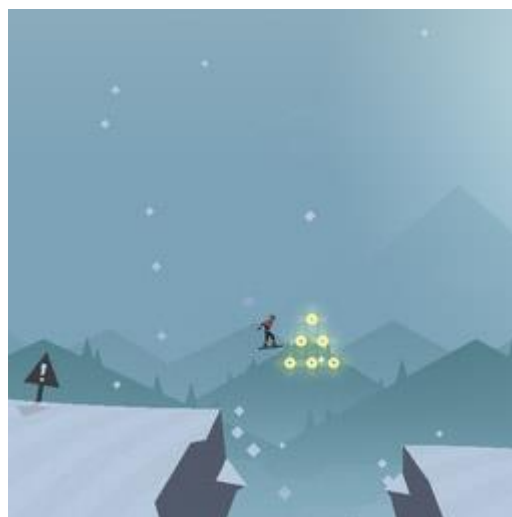


Рисунок 1.9 – Геймлей гри Alto's Adventure [15]

Leo's Fortune [16]

Розробник: Noodlecake Games

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Платформи: iOS, Android

Жанр: Платформер

Основні характеристики:

Красива візуальна складова

Унікальний геймплей

Захоплива історія

Відсутність мікротранзакцій (рис. 1.10)



Рисунок 1.10 – Геймплей гри Leo's Fortune [16]

1.3. Постановка задачі

Основною метою розробки гіперказуальної гри платформер є надання користувачеві можливість підняти настрій та можливість відволіктись від проблем під час користування додатком. Завдяки рейтингу зібраних очок, гра може стати змагальницькою у колі друзів, колег чи родичів.

У роботі потрібно розробити додаток у вигляді гри, в якій керуючи персонажем стрибати з платформи на платформу, збираючи допоміжні інструменти для здобуття найвищого рейтингу. Для реалізації додатка, основними етапами є:

1. Розробити дизайн користувацького інтерфейсу.

- а. Створити загальну концепцію дизайну користувацького інтерфейсу.
- б. За допомогою графічного редактора створити основні

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

елементи UI:

- Головне меню.
- Меню початок гри.
- Меню бали та досягнення.

2. Проектування та розробка основних ігрових модулів

а. Модуль керування персонажем.

- Стрибання персонажу.
- Акселерометре керування.

б. Модуль генерації платформ.

с. Модуль генерації бонусів.

д. Модуль звуків у грі.

Результатом виконання поставленого завдання є додаток у вигляді гри гіперказуального жанру платформер.

1.4. Обґрунтування вибору інструментальних засобів

Ігровий рушій - основна частина коду комп'ютерної гри, доступна з інтегрованим середовищем програмування, призначеним для людей та команд, що створюють комп'ютерні ігри. Він надає різноманітні функції від анімації до штучного інтелекту. Ігрові рушії відповідають за візуалізацію графіки, виявлення зіткнень, керування пам'яттю та багато інших опцій. Ігрові рушії надають розробникам інструменти для легшого створення своїх додатків.

Ігровий рушій містить п'ять компонентів: основна ігрова програма, яка містить ігрову логіку; механізм візуалізації, який можна використовувати для створення 3D-анімованої графіки; аудіосистема, яка складається з алгоритмів, що належать до звуків; фізичний рушій для реалізації “фізичних” законів у системі; та штучний інтелект.

Для успішного проекту, який зможе довго утримувати гравців, а також приносити прибуток кілька років, потрібно створити добрий продукт, а також здійснювати довготривалу підтримку продукту. Для всього цього потрібне використання найбільш придатних засобів розробки. На ринку існує велика

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

кількість рушіїв для реалізації відеоігор. Для розробника важливим завданням є вибір ігрового рушія, який йому більше підходить, кожен рушій є оптимальним вибором для конкретного жанру. Але є рушії, які підходять для більшості жанрів, і за допомогою яких можливо створити безліч різноманітних продуктів і навіть ігри “AAA” [17].

На ігровому ринку існують дуже багато рушіїв, але з них є три основні, а саме: Unity Engine, Unreal Engine, Godot.

Unity Engine – це мультиплатформенний ігровий рушій, який дозволяє легко створювати інтерактивний 3D-вміст. Цей ігровий рушій сьогодні є вибором багатьох великих організацій завдяки його чудовій функціональності, високоякісному контенту та можливості використовувати його для будь-якого типу гри. Він підтримує як 2D, так і 3D вміст, але більше підходить під 2D, але якщо добре з ним працювати з 3D, також можливо створити гарну картинку. Завдяки своєму універсальному редактору Unity сумісний з Windows, Mac, Linux, IOS, Android, Switch, Xbox, PS4, Tizen та іншими платформами. Зручний інтерфейс полегшує розробку та зменшує потребу в навчанні. Asset store Unity зберігає величезну колекцію інструментів та вмісту, що створюється щодня, й серед них є багато які розповсюджуються безплатно, або за гарною ціною. Використовує мову програмування C# [18].

Unreal Engine – один з найпопулярніших і широко використовуваних ігрових рушіїв, який належить Epic Games. Написаний на мові C++, рушій дозволяє створювати ігри для великих операційних систем і платформ, а також на різних портативних пристроях, наприклад, пристроїв Apple (iPad, iPhone), керованих системою iOS й інших. Для спрощення портування рушія використовує модульну систему залежних компонентів; підтримує різні системи рендерингу (Direct3D, OpenGL, Pixomatic), відтворення звуку (EAX, OpenAL, DirectSound3D), засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи із мережею та підтримки різних пристроїв введення. Для ігор мережею підтримуються технології Windows Live, Xbox Live, GameSpy й інші, включаючи до 64 гравців (клієнтів) одночасно. Таким чином, рушій адаптований і для

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

застосування в іграх жанру MMORPG (один із прикладів: Lineage II). Unreal Engine пропонує різноманітні навчальні посібники, які допоможуть новачкам почуватись комфортом з рушієм. Як і Unity, Unreal має власний ринок інструментів [19].

Godot – це рушій з відкритим кодом для крос-платформної розробки ігор. Це максимально інтегроване та самодостатнє середовище для розробки ігор. Середовище дозволяє розробникам створювати ігри з нуля, не користуючись більше ніякими інструментами окрім тих, що потрібні для створення ігрового контенту. Процес програмування також не потребує зовнішніх інструментів (хоча при необхідності використовувати зовнішній редактор досить легко). Загальна архітектура побудована навколо концепції дерева з наслідуваних "сцен". Кожен елемент сцени, в будь-який момент сам може стати повноцінною сценою. Всі ігрові ресурси, від скриптів до графічних ассетів та ігрових сцен, зберігаються в теці просто як звичайні файли, та не є частиною складної бази даних проекту. Ресурси, що не являють собою комплексних даних, зберігаються у простих текстових форматах (наприклад скрипти та сцени, на відміну від моделей та текстур). Ці рішення дозволяють значно спростити різним командам розробників працювати з системами керування версій [20].

Першим критерієм вибору ігрового рушія, є його ціна.

- 1) Unity Engine – безплатна модель поширення для проєктів з оборотом не більше 100 тис. доларів за останні 12 місяців;
- 2) Unreal Engine – перший 1 мільйон доларів США валового доходу за весь час, отриманий вашим продуктом, звільняється від роялті, а так 5% роялті від успіху продукту.
- 3) Godot – повністю безплатний;

Другим критерієм вибору ігрового рушія, є платформи котрі він підтримує. А саме операційні системи IOS та Android. Ці платформи підтримують всі.

Третім критерієм вибору ігрового рушія, є мова програмування яку він підтримує:

- 1) Unity Engine – C#;

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

2) Unreal Engine – C++;

3) Godot – GDScript;

Під поточні потреби було вирішено обрати ігровий рушій Unity Engine з таких причин:

розробка за допомогою мови програмування C#. Дана мова програмування добра тим, що вона має компонентно-орієнтований підхід до програмування, сприяє меншій машинно-архітектурній залежності програмного коду, гнучкості, мобільності й легкості повторного використання фрагментів коду;

1) вигідні умови ліцензування. Безоплатна модель поширення для проєктів з оборотом не більше 100 тис. доларів за останні 12 місяців;

2) рушій має зручний інтерфейс з можливістю розставляти об'єкти і в реальному часі тестувати результат;

3) також Unity Engine володіє великим товариством розробників, величезною бібліотекою готових рішень і плагінів, що прискорює процес розробки. За допомогою рушія можна розробляти ігри на всі актуальні ігрові платформи і перенести їх без особливої складності.

4) також склалося, що Unity більше підходить для розробки мобільних додатків, а UE для FPS ігор AAA рівня.

Для розробки левел дизайну ігри, потрібні засоби для роботи з графікою та анімацією.

Adobe Illustrator - дозволяє створювати векторні зображення. У відмінності від растрових картинок, які складаються з пікселів, у векторних зображеннях якість не змінюється при масштабуванні. Програму частіше всього використовують художники та дизайнери для створення логотипів, ілюстрацій, значків, макетів для друку, а також для дизайну мобільних додатків [21].

DragonBones — це безплатне рішення для скелетної анімації мобільних ігор з відкритим вихідним кодом з підтримкою кількох мов і одноразовим випуском функцій повного випуску платформи, яке в основному використовується для створення 2D-ігрової анімації та мультимедійного контенту, а також допомагає дизайнерам створювати більш яскраві анімаційні ефекти, з меншими витратами

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

на мистецтво. Підтримує кілька форматів виведення, таких як відео, вебсторінка, анімація. Основний ігровий рушій і підтримка мов програмування. Надання різноманітних бібліотек часу виконання платформ.

Visual Studio Code — редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки веб- та хмарних додатків. Містить в собі відладчик, інструменти для роботи з Git, підсвітку синтаксису та засоби для рефакторингу. Є широкі можливості для кастомізації: користувацькі теми, комбінації клавіш і файли конфігурації. Розповсюджується безплатно. Також за допомогою додатків в ньому, можливо полегшити собі написання коду для скриптів в Unity [22].

Отже, були обрані такі інструменти:

- 1) Unity Engine з мовою програмування C#;
- 2) Visual Studio Code – як редактор вихідного коду;
- 3) Adobe Illustrator;
- 4) DragonBones;

Висновки до першого розділу

Проведено аналіз ігрової індустрії, ознайомлено з поняттям гіперказуальна гра та визначено основні принципи розробки. Проведено аналіз чинного програмного забезпечення та різновиди за жанром.

Визначено основну задачу, на основі якої проаналізовано та вибрано ігровий рушій та інші компоненти для розв’язання поставленої задачі.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури додатку

Відповідно до визначеного завдання на розробку гри було побудовано діаграму активності користувача (рис. 2.1). Розглянемо основні дії детальніше.

Почати гру: гравець запускає гру.

Керувати персонажем: гравець керує персонажем, використовуючи повороти екраном.

Стрибати по платформах: персонаж стрибає по платформах, щоб уникнути падіння.

Впасти в безодню: якщо персонаж впаде в безодню, гра закінчується.

Закінчити гру: гра закінчується, коли персонаж досягає кінця рівня або втрачає всі життя.

Переглянути рахунок і рейтинг: після закінчення гри гравець може переглянути свій рахунок і рейтинг.

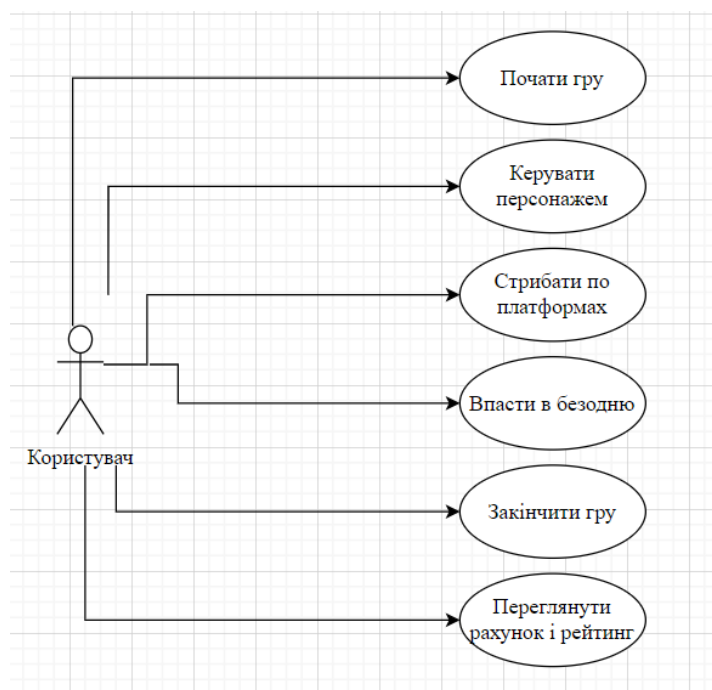


Рисунок 2.1 – Діаграма активності користувача

Відкривши додаток, гравець має можливість керувати персонажем, а саме стрибати по платформах. Під час гри на шляху героя трапляються бонуси, такі як,

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

батут, пружина та капелюх, за допомогою яких він може пришвидшити рух та скоротити дистанцію. Закінчити гру, можна натиснувши "пауза"-"вийти в головне меню"-"завершити гру" або Впавши з платформи, гравець може впасти в безодню, тим самим закінчити гру (рис. 2.2).

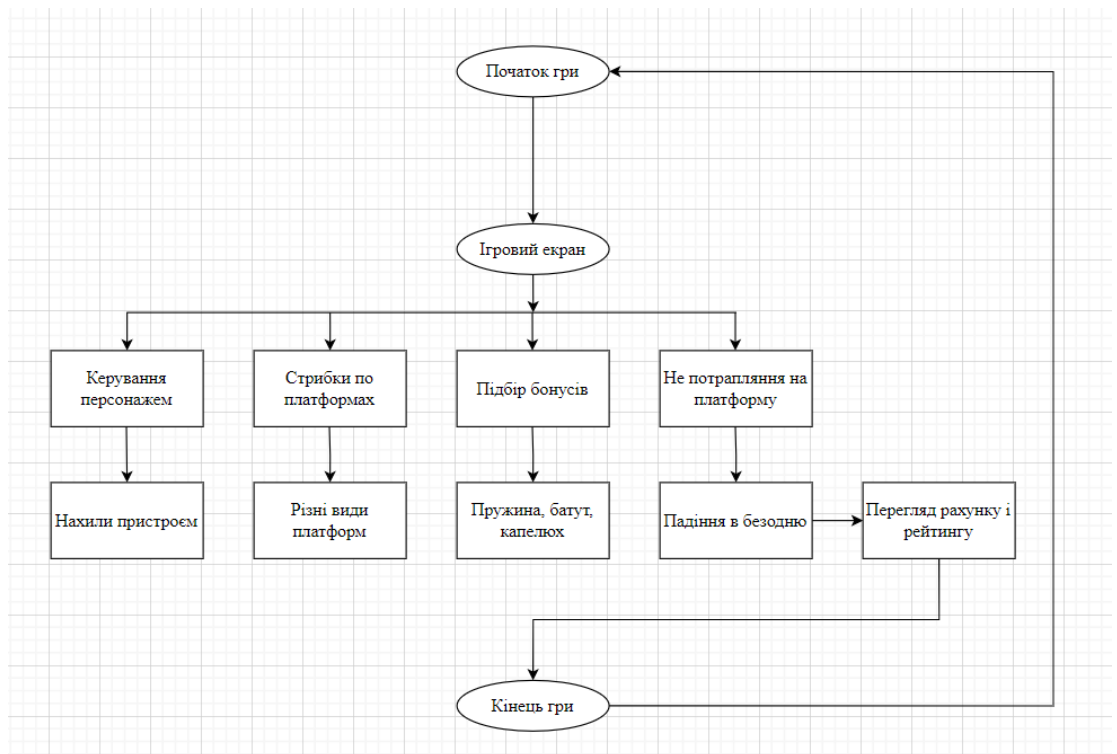


Рисунок 2.2 – Діаграма діяльності користувача системи

Проаналізувавши функціональні вимоги були побудовані сутності з яких складається гра. Для графічного зображення сутностей та зв'язків між ними використовують діаграму класів (рис. 2.3). Діаграма класів – діаграма на якій представлена сукупність декларативних, або статичних елементів моделі, таких як класи з атрибутами й операціями, а також відношення [23].

Основні класи:

Movement: відповідає за переміщення зі сторони в сторону, змінюється за допомогою акселерометра

Movement_Speed: швидкість пересування гравця.

Rigid: фізичний компонент Rigidbody, який використовується для керування рухом гравця.

Spr_Player: компонент в якому зберігається теперішній спрайт персонажу

FixedUpdate: метод в якому проходить зміна напрямку який дивиться спрайт персонажу, та описаний варп з одного краю екрану на інший.



Рисунок 2.3 – Діаграма класів

Player controller: цей клас описує дії гравця та має такі атрибути:

Update: основний метод класу який викликає кожну зміну фрейму, й додає до акселерометра швидкість руху.

Propeller_Fall: метод який видаляє з персонажа бустер пропелер

Platform: цей клас описує платформу. Він має такі атрибути:

PlatformType: метод який описує Тип платформи.

DestroyDistance: відстань, на якій платформа знищується.

Jump_Force: прискорення, яке дає персонаж платформа

Create_NewPlatform: створення нової платформи.

Діаграма класів взаємодії з персонажем (рис. 2.4) включає класи:

Propeller: клас описує взаємодію з ігровим бустером

Platform_White: клас який описує взаємодію з білою платформою, яка зникає

Submit_Score: клас для поширення результатів

Platform_Blue: клас в якому прописані особливості платформи яка рухається

UserRecord: клас в якому зберігається та оброблюється рекорд користувача

Platform_Brown: кпис взаємодії з платформою пасткою

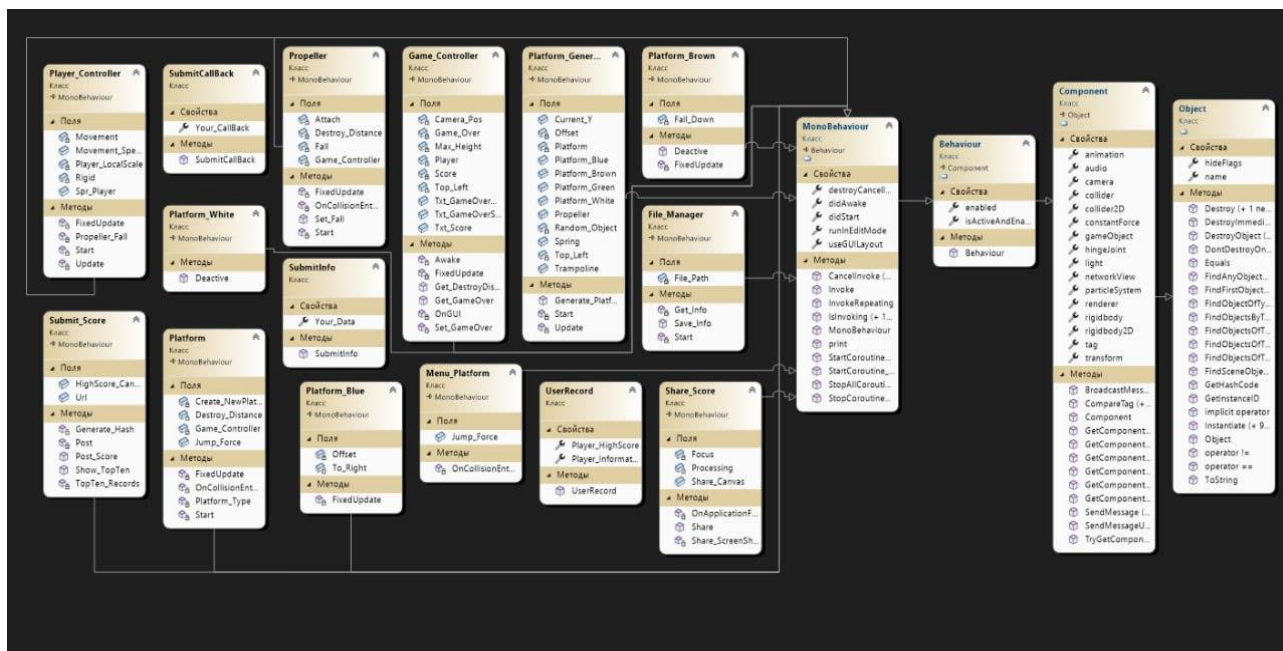


Рисунок 2.4 – Діаграма класів взаємодії з персонажем

2.2. Реалізація графічних елементів додатку

1. Персонаж:

Варіанти: можна додати різні костюми або аксесуари, щоб розширити кастомізацію.

Для створення персонажу гри скористаємося AdobeIllustrator, ним буде панда, що стрибає (рис. 2.5).

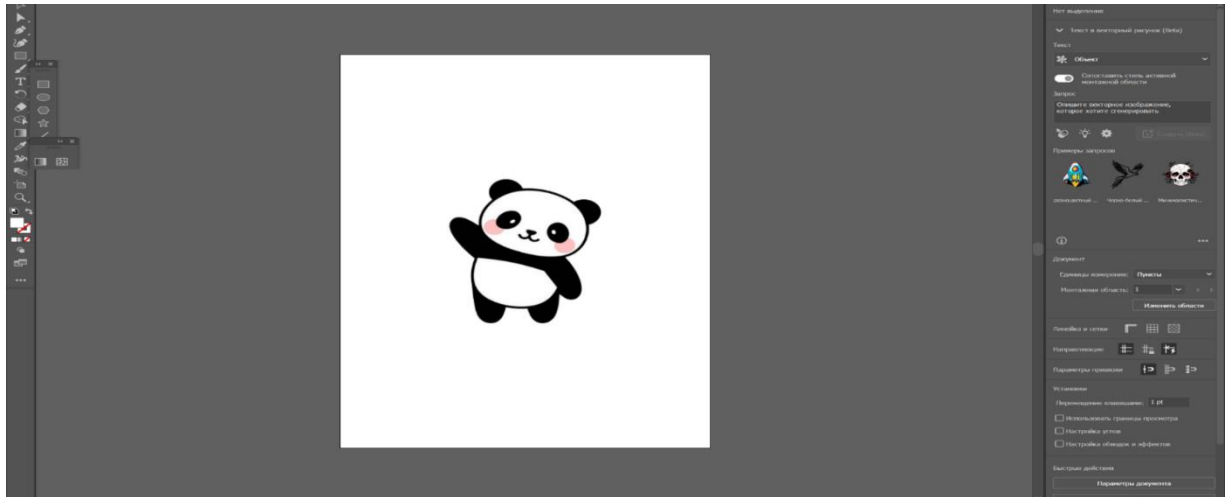


Рисунок 2.5 – Персонаж гри PandaJump в середовищі Adobe Illustrator

2. Платформи:

Різноманітні форми та розміри, щоб додати візуального інтересу та складності геймплею.

Текстури: трава, камінь, дерево, хмари тощо.

Можна додати анімацію, наприклад, рухомі платформи або платформи, що зникають (рис. 2.6).

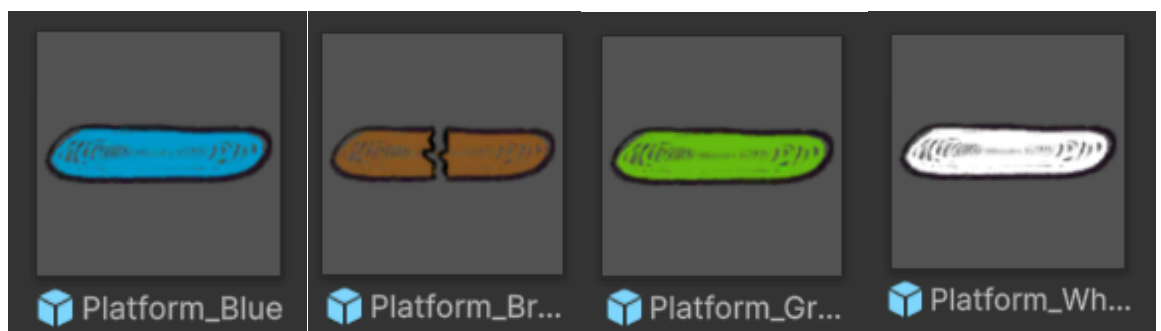


Рисунок 2.6 – Рухомі та зникаючі платформи гри PandaJump

3. Перешкоди:

Шипи, ями, інші небезпечні об'єкти.

Текстури мають відповідати платформам, на яких вони знаходяться. Можна додати анімовані перешкоди, наприклад, шипи, що висувуються, або ями, що розширюються.

4. Бонуси:

Монети, зірки, інші предмети, які дають очки або інші переваги

Яскраві та помітні текстури.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

Можна додати анімацію, наприклад, блискучі монети або зірки, що кружляють (рис. 2.7).

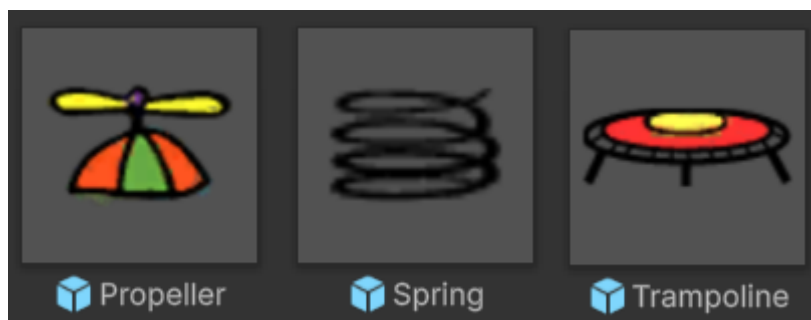


Рисунок 2.7 – Використанні бонуси гри PandaJump

5. Фони:

Різноманітні фони для різних рівнів, щоб додати візуального різноманіття.

Небо, хмари, гори, ліси тощо.

Можна використовувати градієнти, ефекти освітлення та інші візуальні прийоми для створення атмосфери.

6. Інтерфейс користувача (UI):

Простий та інтуїтивно зрозумілий.

Кнопки для початку гри, паузи та повернення в меню.

Рахунок очок (рис. 2.8).



Рисунок 2.8 – Активні кнопки меню початку гри

2.3. Розробка програмної частини додатку

Після того, як ми закінчили з візуальним оформленням нашої гри, переходимо в Unity так, як в нашій грі один нескінченний рівень, то нам потрібно буде створити дві сцени:

- In_Game - сцена в якій проходить основна суть нашої гри, гравець піднімається по платформах як можна високо.

- Main_Menu - головне меню гри.

Розробивши прототип персонажу, під'єднавши стрибання по платформа та акселерометр, виставивши персонажа на платформу (рис. 2.9).

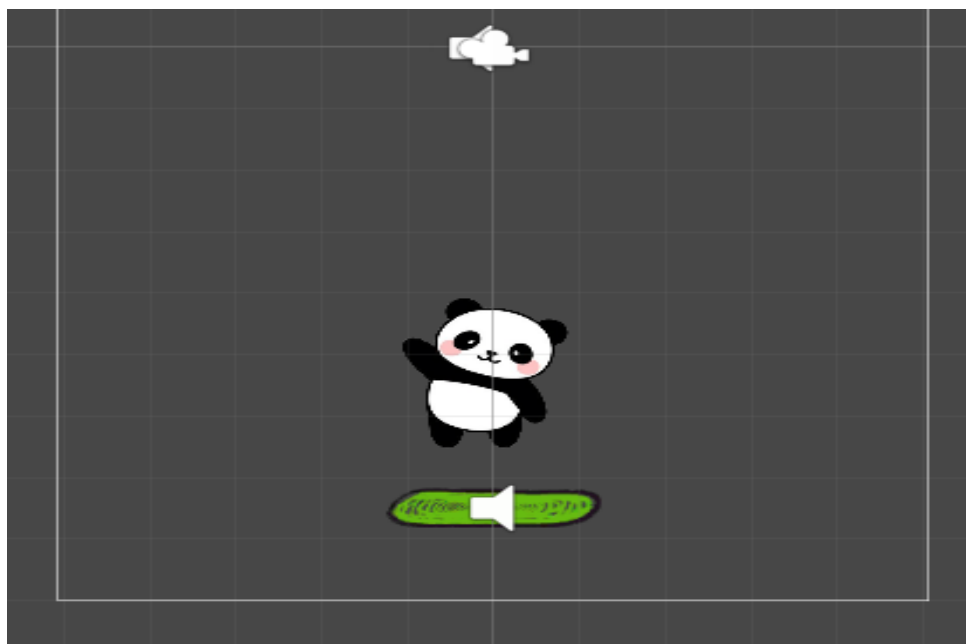


Рисунок 2.9 – Спрайти персонажу й платформи виставлені в робоче середовище

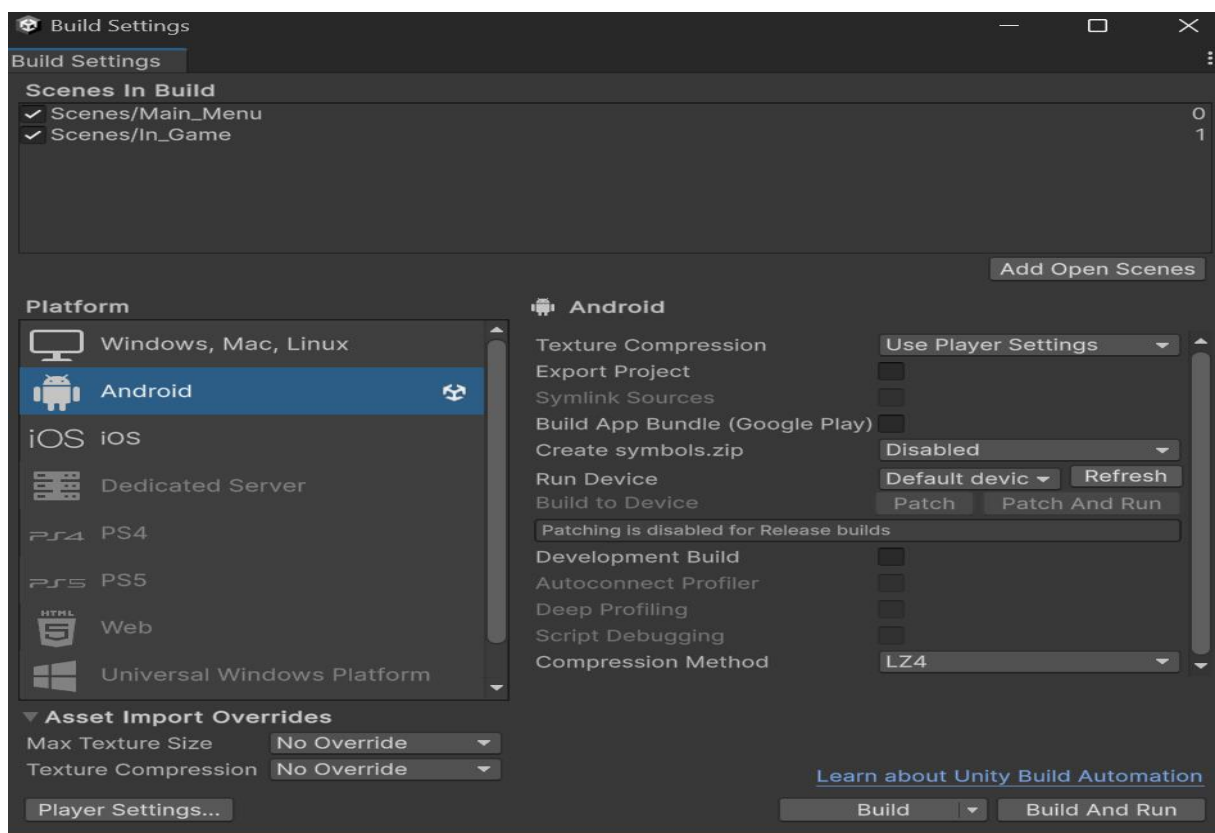


Рисунок 2.10 – Меню Build Settings

В меню Build Settings в параметрах змінено платформу на андроїд, згідно з тематикою завдання. (рис. 2.10). Додавання піктограм та інших необхідних параметрів для гри здійснюється в меню “Player Settings” (рис. 2.11).

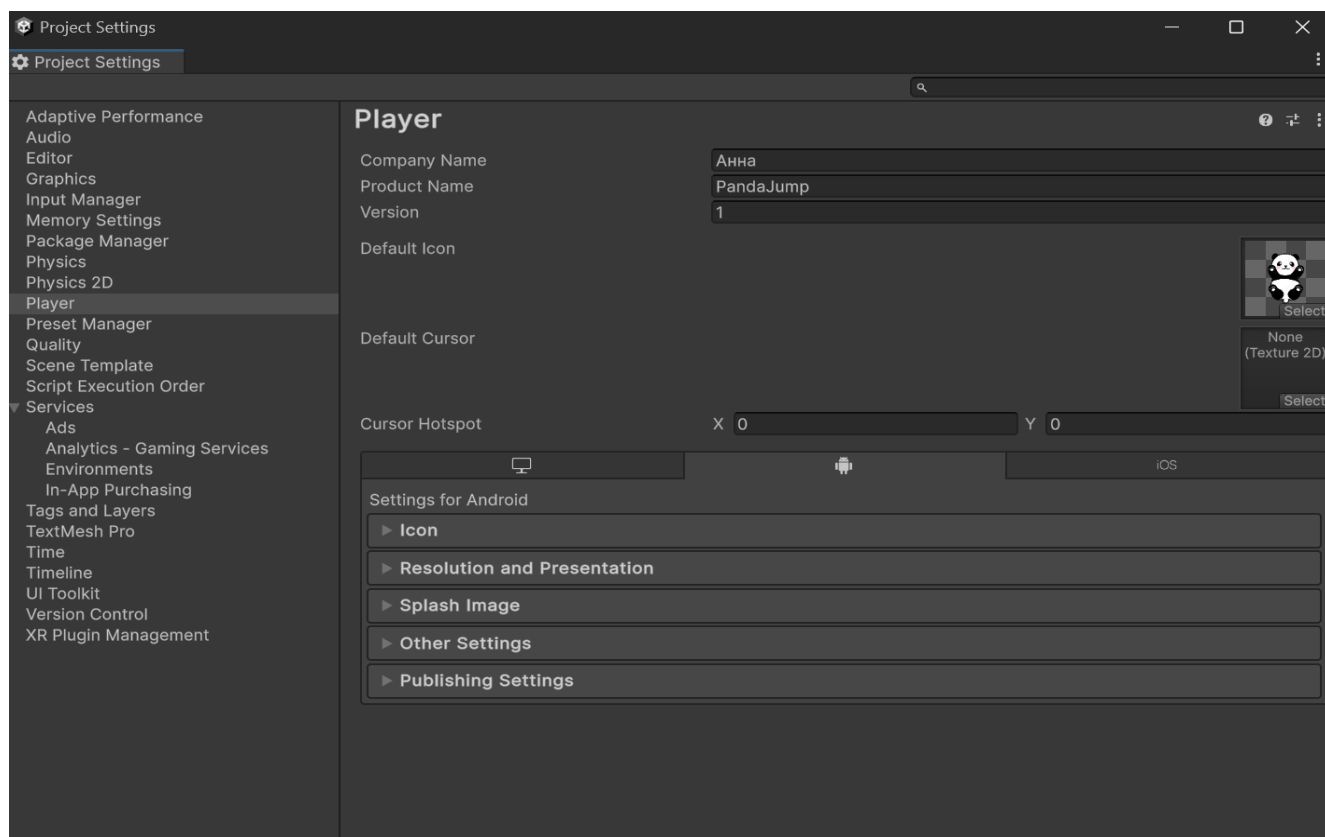


Рисунок 2.11 – Меню Project Settings

При розробці ігор використовують безліч об'єктів, які мають менші, скрипти, звуки та інші елементи. Інспектор зображає детальну інформацію про об'єкт, включаючи всі прикріплені компоненти й всі їх властивості. Будь-яка властивість яка зображається в інспекторі може бути модифікована без змін самого скрипта, а також дає змогу змінювати параметри змінних в режимі гри для експериментів. Також інспектор дає можливість якщо в скрипті ми створили публічну змінну типу об'єкт (GameObject або Transform), то ми можемо перенести об'єкт або префаб напряму в інспектор на цю зміну та дати їй значення (рис. 2.12).

Використаємо безліч інших компонентів які можуть нам знадобитися в роботі, наприклад, Button який дозволяє обробляти натискання. Додавши власні скрипти до об'єктів, це дає нам безліч можливостей для роботи.

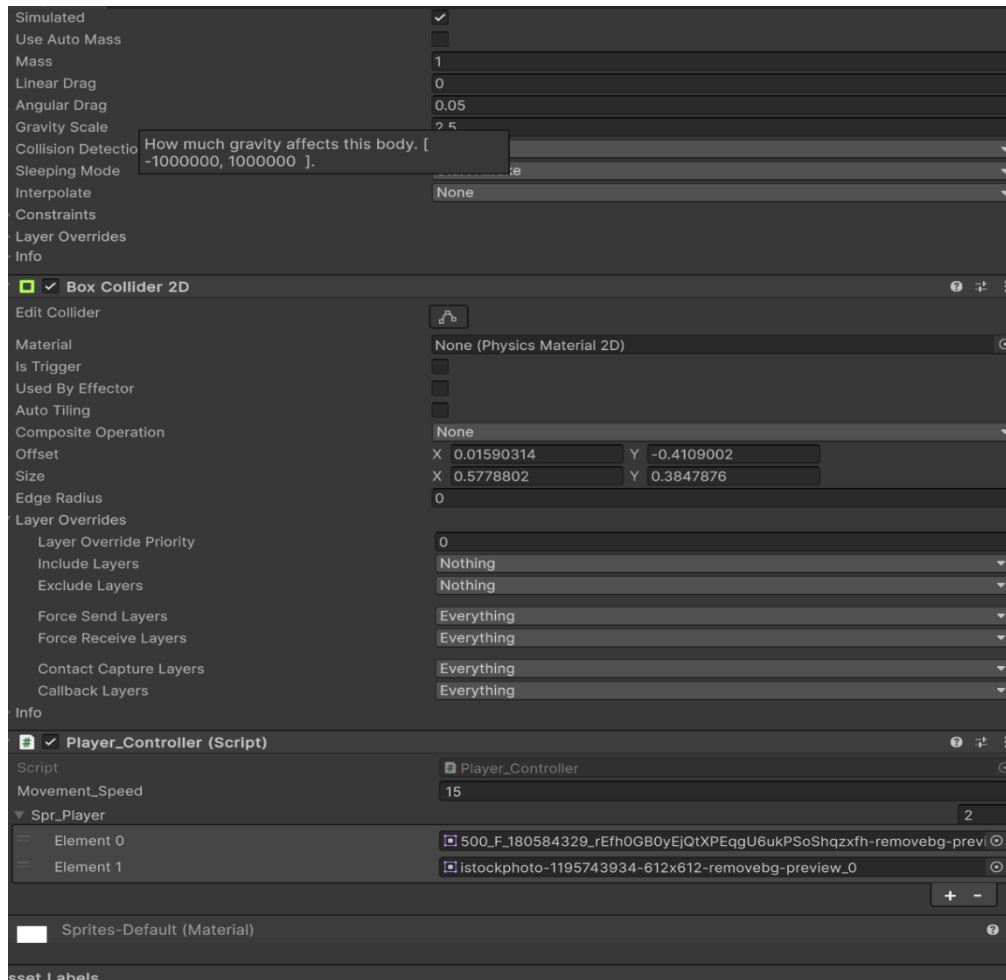


Рисунок 2.12 – Інспектор об'єкту персонажу

Досить зручно працювати з GameObject у сцені, додаючи компоненти та змінюючи їх значення на потрібні вам в інспекторі. Однак, це може створити низку проблем у випадках, коли ми працюємо над створенням неігрового персонажа, об'єктом або предметом, який багаторазово зустрічається у сцені. Звичайно, можемо просто скопіювати ці об'єкти для створення дублікатів, але всі вони редагуватимуться незалежно один від одного. Зазвичай ми бажаємо, щоб усі екземпляри окремо взятого об'єкта мали однакові значення властивостей, щоб при редагуванні одного такого об'єкта в сцені вам не довелося повторно вносити ті самі зміни та всі інші копії. В Unity можна створювати префаби. Це особливий тип ассетів, що дозволяє зберігати весь GameObject з усіма компонентами та значеннями властивостей. Префаб виступає в ролі шаблону для створення екземплярів об'єкта, що зберігається в сцені. Будь-які зміни в префабі негайно зображаються і на всіх його екземплярах, при цьому ви можете перевизначати

компоненти та налаштування для кожного екземпляра окремо. В нашій грі такий багаторазовий об'єкт це платформи тому створимо префаб для них (рис. 2.13).

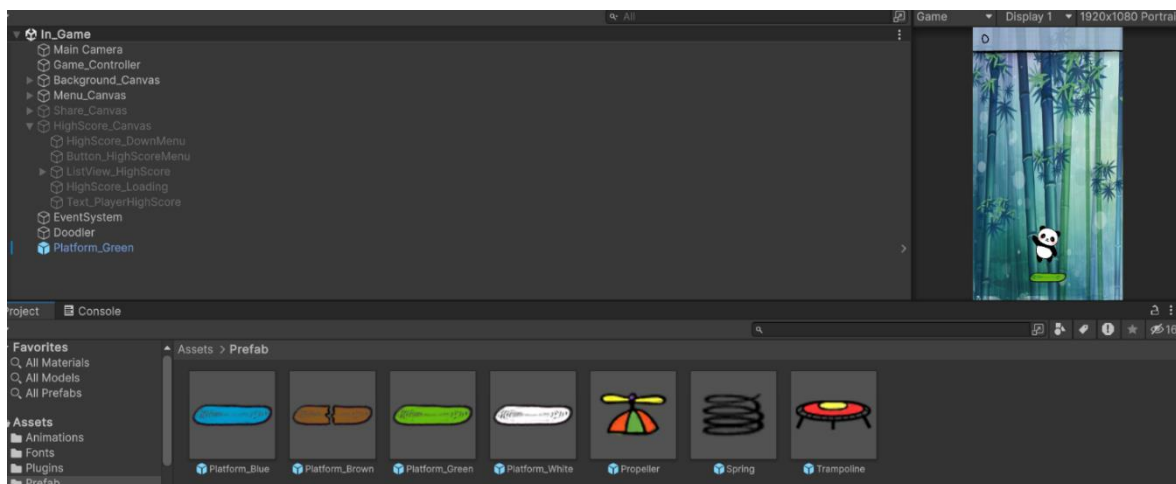


Рисунок 2.13 – Префаб платформи

В процесі гри відбувається генерація платформ, за це відповідає метод `Generate_Platform`.

```
public void Generate_Platform(int Num)
{
    for (int i = 0; i < Num; i++)
    {
        // Підрахунок платформ x, y
        float Dist_X = Random.Range(Top_Left.x + Offset, -Top_Left.x - Offset);
        float Dist_Y = Random.Range(2f, 5f);

        // Створити коричневі платформи з 1/8 вірогідністю
        int Rand_BrownPlatform = Random.Range(1, 8);

        if (Rand_BrownPlatform == 1)
        {
            float Brown_DistX = Random.Range(Top_Left.x + Offset, -Top_Left.x -
            Offset);
            float Brown_DistY = Random.Range(Current_Y + 1, Current_Y + Dist_Y
            - 1);
            Vector3 BrownPlatform_Pos = new Vector3(Brown_DistX, Brown_DistY,
            0);

            Instantiate(Platform_Brown,                                BrownPlatform_Pos,
            Quaternion.identity);
        }
        // створення інших платформ
        Current_Y += Dist_Y;
        Vector3 Platform_Pos = new Vector3(Dist_X, Current_Y, 0);
        int Rand_Platform = Random.Range(1, 10);

        if (Rand_Platform == 1) // Створити сині платформи
            Platform = Instantiate(Platform_Blue, Platform_Pos,
            Quaternion.identity);
    }
}
```

```

        else if (Rand_Platform == 2) // Створити білі платформи
            Platform = Instantiate(Platform_White, Platform_Pos,
Quaternion.identity);
        else // Створити зелену платформу
            Platform = Instantiate(Platform_Green, Platform_Pos,
Quaternion.identity);
        if (Rand_Platform != 2)
        {
            // Створення випадкових предметів
            int Rand_Object = Random.Range(1, 40);

            if (Rand_Object == 4)
            {
                // Створити пружину
                Vector3 Spring_Pos = new Vector3(Platform_Pos.x + 0.5f,
Platform_Pos.y + 0.27f, 0);
                Random_Object = Instantiate(Spring, Spring_Pos,
Quaternion.identity);

                // Встановлення батьківського об'єкту
                Random_Object.transform.parent = Platform.transform;
            }
            else if (Rand_Object == 7) // створити трамплін
            {
                Vector3 Trampoline_Pos = new Vector3(Platform_Pos.x + 0.13f,
Platform_Pos.y + 0.25f, 0);
                Random_Object = Instantiate(Trampoline, Trampoline_Pos,
Quaternion.identity);

                Random_Object.transform.parent = Platform.transform;
            }
            else if (Rand_Object == 15) // Створити пропелер
            {
                Vector3 Propeller_Pos = new Vector3(Platform_Pos.x + 0.13f,
Platform_Pos.y + 0.15f, 0);
                Random_Object = Instantiate(Propeller, Propeller_Pos, Quaternion.identity);

                Random_Object.transform.parent = Platform.transform;
            }
        }
    }
}
}
}

```

Система анімації Unity базується на концепції анімаційних кліпів, які містять інформацію про те, як певні об'єкти мають змінювати своє положення, обертання чи інші властивості з часом. Кожен кліп можна розглядати як один лінійний запис. Анімаційні кліпи із зовнішніх джерел створюються художниками

чи аніматорами за допомогою інструментів сторонніх розробників, таких як Мах або Мауа, або надходять зі студій захоплення руху чи інших джерел.

Потім анімаційні кліпи організовуються у структуровану систему, схожу на блок-схему, яка називається контролером анімації. Контролер Animator діє як «кінцевий автомат», який відстежує, який кліп зараз має відтворюватися, а також коли анімації мають змінитися або поєднати.

Дуже простий контролер анімації може містити лише один або два кліпи, наприклад, щоб керувати обертанням і підстрибуванням посилення або для анімації відкриття та закриття дверей у потрібний час. Досконаліший контролер анімації може містити десятки гуманоїдних анімацій для всіх дій головного героя та може поєднуватися між кількома кліпами одночасно, щоб забезпечити плавний рух, коли гравець рухається по сцені.

Система анімації Unity також має численні спеціальні функції для роботи з гуманоїдними персонажами, які дають вам можливість перенацілювати гуманоїдну анімацію з будь-якого джерела (наприклад, захоплення руху, сховище ресурсів або іншу сторонню бібліотеку анімації) на вашу власну модель персонажа як, а також коригування чіткості м'язів. Ці особливі функції доступні системи Аватар Unity, де персонажі-гуманоїди зображаються у загальному внутрішньому форматі.

Для створення гри, було розроблено ряд анімацій та контролерів, які продемонстровані на рисунку 2.14.



Рисунок 2.14 – Анімації проекту

Кожна з цих частин — анімаційні кліпи, контролер аніматора та аватар — об'єднані в GameObject за допомогою компонента аніматора. Цей компонент містить посилання на контролер аніматора та (за потреби) аватар для цієї моделі.

Контролер Animator, своєю чергою, містить посилання на анімаційні кліпи, які він використовує (рис. 2.15 - 2.19).

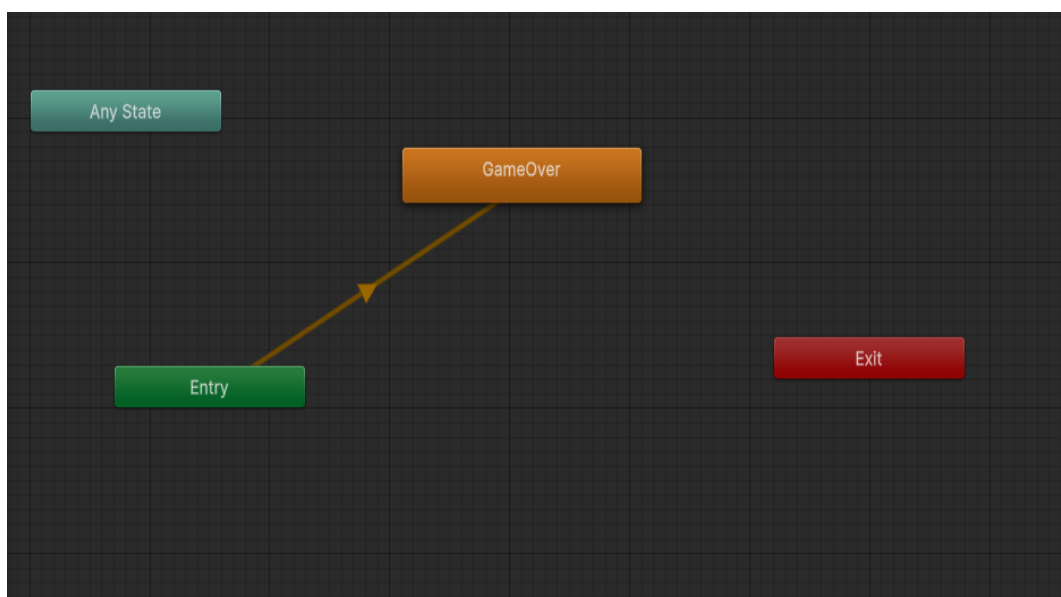


Рисунок 2.15 – Схема анімації BackgroundCanvas

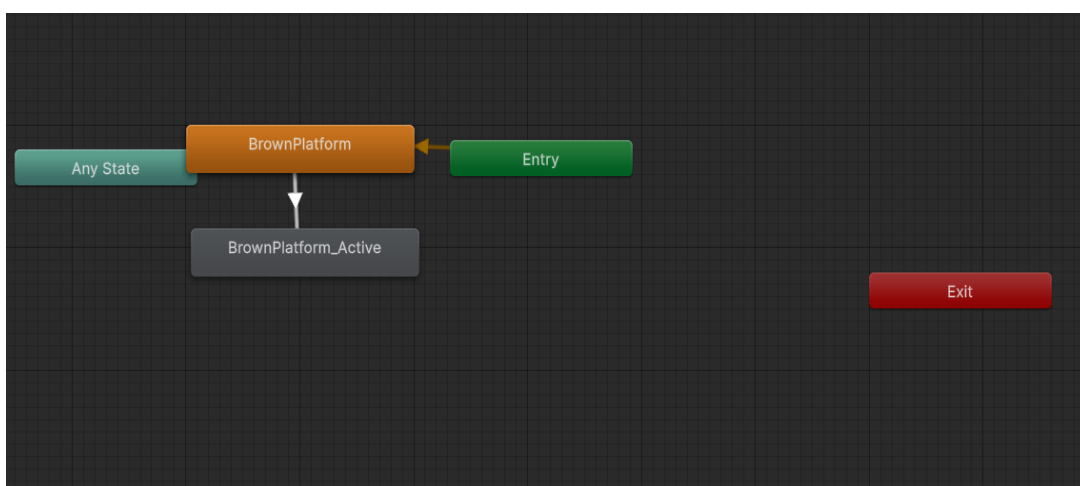


Рисунок 2.16 – Схема анімації Platform_Brown

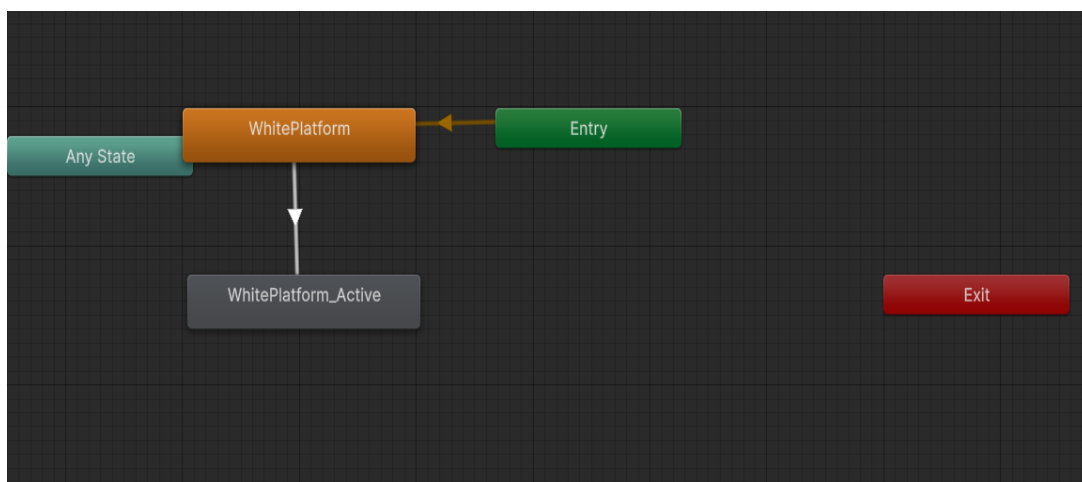


Рисунок 2.17 – Схема анімації Platform_White

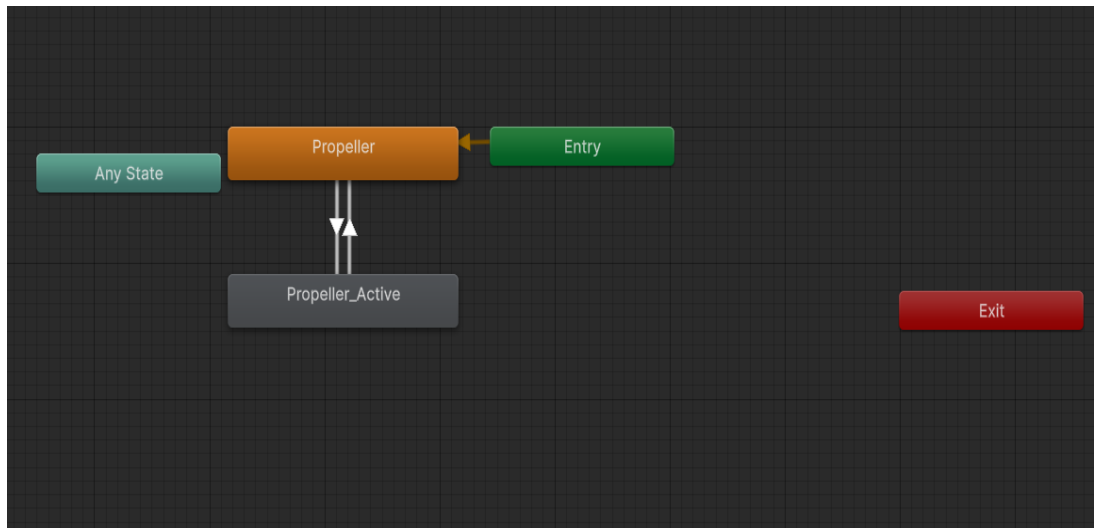


Рисунок 2.18 – Схема анімації Propeller

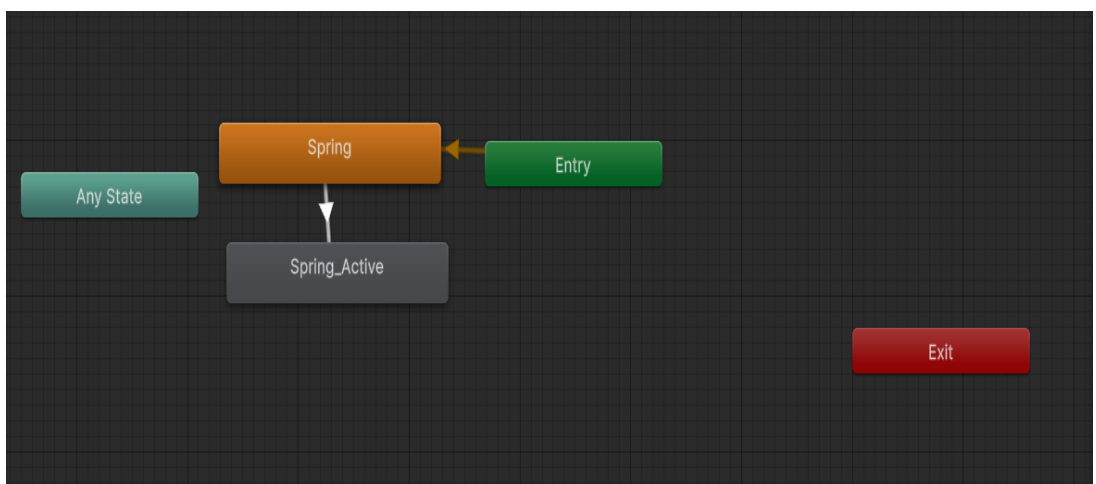


Рисунок 2.19 – Схема анімації Spring

Висновки до другого розділу

У даному розділі було проведено визначення варіантів використання та об'єктоорієнтованої структури додатка. Проаналізовано функціональні вимоги з яких складається сутність гри, здійснено розробку класів та реалізовано графічні елементи гри. Було здійснено розробку програмної частини гри.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ

3.1. Порядок встановлення та налаштування додатку

Протягом розробки проєкту, гра набула назву Panda Jump, що пов'язується з основним персонажем та стилем гри.

Збірку виконано для операційної системи Android. Для коректної роботи застосунку потрібні мінімальні вимоги до системи:

- операційна система: Android 9 (API рівень 28)
- процесор (CPU): MediaTek Helio G85
- оперативна пам'ять (RAM): 2 ГБ
- вільне місце на диску: 100 МБ (включаючи 51.5 МБ для інсталяції гри)
- графічний процесор (GPU): Mali-G52 MP2
- екран: роздільна здатність 720p (1280 x 720) або вище
- інтернет-з'єднання: працює без інтернет-з'єднання

Для користування додатком необхідно запустити арк-файл, гра буде встановлена та готова до роботи на вашому мобільному пристрої.

3.2. Інтерфейс та порядок роботи з програмним додатком

При відкритті гри зображається меню стрибучого персонажа зі звуковою імітацією (рис. 3.1). При натисканні на кнопку “play” – починається гра.



Рисунок 3.1 –Початкове меню гри PandaJump

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

В ігровому режимі де персонаж стрибає з платформи на платформу, гра супроводжується звуковим ефектом. З набором більшої кількості очок відбувається зміна платформ та з'являються допоміжні бонуси, підбирання яких супроводжується звуковим ефектом (рис. 3.2).



Рисунок 3.2 - Геймплей гри PandaJump

При переході через край екрана персонаж з'являвся з іншого боку та при падінні в нижню частину екрана, анімація супроводжується звуковим ефектом, і зображається меню кінця гри (рис. 3.4).



Рисунок 3.3 - Геймплей гри PandaJump

Меню кінця гри пропонує розпочати нову гру, перейти в головне меню та переглянути меню набраних та рекордних очок.



Рисунок 3.4 – Меню кінця гри PandaJump

3.3. Тестування роботи програмного додатку

Кожен готовий продукт перед реалізацією потребує перевірки основних компонентів та виявлення недоліків, які можуть негативно вплинути на реалізацію додатка. Перевірка також є важливим аспектом для подальшого майбутнього продукту, це доповнення, оновлення, апгрейди для збільшення аудиторії додатка.

Завершивши розробку проведено тестування готової гри.

Перевіримо кнопку “play” – при натисканні перекидає нас в ігровий режим (рис. 3.1-3.2).

Падіння персонажу в безодню – перекидає в меню кінця гри (рис. 3.3-3.4).

Меню кінця гри – можливість грати знову “play again”, переглянути кількість набраних очок, рейтинг та перехід у головне меню (рис. 3.4).

При тестуванні даного функціоналу не було виявлено критичних недоліків чи помилок у роботі даного застосунку.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до третього розділу

У даному розділі було наведено порядок встановлення та налаштування додатка. Наведено опис роботи з програмним додатком. Показано функціональні можливості гри. Наведено приклад використання бонусів, проведено тестування продукту.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У кваліфікаційній роботі метою було створення гіперказуальної гри для Android-платформи

Для досягнення мети було виконано наступні завдання:

- проведено аналіз ринку і трендів гіперказуальних ігор, визначена цільова аудиторія і конкурентні переваги продукту;
- досліджено різні середовища розробки ігор та ігрові рушії.
- встановлено і налаштовано ігровий рушій Unity, ознайомлено з його функціоналом і можливостями;
- спроектовано функціонал додатка.

На основі цього було реалізовано прототип двовимірного платформера. У ході розробки були написані скрипти, необхідні для роботи гри, та створенні або запозиченні графічні та звукові матеріали.

На основі проведеної роботи зробимо наступні висновки:

- розробка гіперказуальної гри на андроїд є актуальним і перспективним напрямком геймдеву, який вимагає від розробника креативності, аналітичності, уваги до деталей і постійного самовдосконалення;
- ігровий рушій Unity є потужним і зручним інструментом для розробки гіперказуальних ігор на андроїд, який надає розробнику широкі можливості для реалізації своїх ідей, інтеграції зовнішніх сервісів, оптимізації й публікації гри;
- для досягнення успіху на ринку гіперказуальних ігор на андроїд необхідно постійно вивчати попит і пропозицію, аналізувати конкурентів і тренди, тестувати та вдосконалювати гру, використовувати ефективні маркетингові стратегії й канали для просування гри.

У результаті отримано цінний досвід розробки гіперказуальної гри на андроїд, покращено навички програмування, дизайну, тестування і маркетингу.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Данні ринку мобільних ігор. 2023. [Електроний ресурс] – Режим доступу: <https://www.data.ai/en/insights/market-data/q3-2023-mobile-market-pulse/>
2. Що таке мобільна гра. 2024. [Електроний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Mobile_game
3. Світовий ринок ігор у 2023 році. 2023. [Електроний ресурс] – Режим доступу: <https://newzoo.com/resources/blog/last-looks-the-global-games-market-in-2023>
4. Таксономія нового жанру та навіщо вона нам потрібна. 2020. [Електроний ресурс] – Режим доступу: <https://www.gamerefinery.com/new-genre-taxonomy-and-why-we-need-it/>
5. Гіперказуальна гра. 2022. [Електроний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Гіперказуальна_гра
6. Розквіт гіперказуальних ігор: Тренд у мобільному геймінгу. 2024. [Електроний ресурс] – Режим доступу: <https://blog.lebara.co.uk/uk/поява-гіперказуальних-ігор-тренд-у-мо/>
7. Hyper Casual Games: Design, Monetization, and Market Insights. 2024. [Електроний ресурс] – Режим доступу: <https://www.blog.udonis.co/mobile-marketing/mobile-games/hyper-casual-game>
8. Гра Idle Miner Tycoon. 2024. [Електроний ресурс] – Режим доступу: <https://apps.apple.com/ie/app/idle-miner-tycoon-money-games/id1116645064>
9. Гра Subway Surfers. 2024. [Електроний ресурс] – Режим доступу: <https://medium.com/udonis/subway-surfers-lessons-from-the-record-breaking-mobile-game-8a0d26635945>
10. Гра Cut the Rope. 2024. [Електроний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.zeptolab.ctr.ads&hl=en_US
11. Гра Flapping Bird. 2023. [Електроний ресурс] – Режим доступу: <https://apkcombo.com/flapping-bird/com.ShinCom.FlappingBirdShin/>
12. Гра Doodle Jump. 2010. [Електроний ресурс] – Режим доступу: <https://www.invertty.com/2010/02/08/doodle-jump-iphone-3/index.html>

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Гра Jetpack Joyride. 2024. [Електроний ресурс] – Режим доступу:
<https://apps.apple.com/us/app/jetpack-joyride-2/id1598096399>
14. Гра Crossy Road. 2020. [Електроний ресурс] – Режим доступу:
https://www.iphonecake.com/app_924373886_.html#
15. Гра Alto`s Adventure. 2018. [Електроний ресурс] – Режим доступу:
<https://www.troopfun.com/>
16. Гра Leo`s Fortune. 2014. [Електроний ресурс] – Режим доступу:
<https://www.polygon.com/2014/4/9/5582614/leos-fortune-is-a-console-sized-game-in-a-mobile-sized-package>
17. Ігровий рушій. 2022. [Електроний ресурс] – Режим доступу:
https://uk.wikipedia.org/wiki/Ігровий_рушій
18. Unity (ігровий рушій). 2024. [Електроний ресурс] – Режим доступу:
[https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))
19. Unreal Engine. 2024. [Електроний ресурс] – Режим доступу:
https://en.wikipedia.org/wiki/Unreal_Engine
20. Godot. 2024. [Електроний ресурс] – Режим доступу:
<https://uk.wikipedia.org/wiki/Godot>
21. Adobe Illustrator. 2024. [Електроний ресурс] – Режим доступу:
https://uk.wikipedia.org/wiki/Adobe_Illustrator
22. Visual Studio Code. 2024. [Електроний ресурс] – Режим доступу:
https://uk.wikipedia.org/wiki/Visual_Studio_Code
23. Діаграми UML для моделювання процесів і архітектури проекту. 2023. [Електроний ресурс] – Режим доступу:
<https://evergreens.com.ua/ua/articles/uml-diagrams.html>
24. Асети гри Doodle-Jump. 2023. [Електроний ресурс] – Режим доступу:
<https://github.com/LunaTMT/Doodle-Jump>

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

Скрипт Button_OnClick.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class Button_OnClick : MonoBehaviour {
    // Button onclick functions
    public void Play_OnClick()
    {
        SceneManager.LoadScene("In_Game");
    }

    public void Scores_OnClick()
    {
        GameObject MainMenu_Canvas = GameObject.Find("Game_Controller");

        // Активація меню рекордів

MainMenu_Canvas.GetComponent<Submit_Score>().HighScore_Canvas.SetActive(true);
        GameObject Main_Canvas = GameObject.Find("Main_Canvas");

        for (int i = 1; i < 14; i++)
            Main_Canvas.transform.GetChild(i).gameObject.SetActive(false);

        // Активація завантаження

MainMenu_Canvas.GetComponent<Submit_Score>().HighScore_Canvas.transform.GetChild(5).gameObject.SetActive(true);

        // Показати результат
        MainMenu_Canvas.GetComponent<Submit_Score>().Show_TopTen();
    }

    public void Menu_OnClick()

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    SceneManager.LoadScene("Main_Menu");
}

public void TapToChange_OnClick()
{
    // Деактивація меню кінця гри
    Set_GameOverMenu(false);

    // Активація меню поширення результатів
    Set_SubmitScoreMenu(true);
}

public void SubmitConfirm_OnClick()
{
    GameObject Background_Canvas = GameObject.Find("Background_Canvas");
    Set_InteractableSubmitScore(false);
    Set_Error(false);
    // Відправити результати на сервер
    string Name =
Background_Canvas.transform.GetChild(14).GetComponent<InputField>().text;
    int High_Score = Data_Manager.Get_HighScore();

    // Перевірка вводу імені
    if (string.IsNullOrEmpty(Name.Replace(" ", "")))
    {
        // Name error
        Button_OnClick.Set_InteractableSubmitScore(true);
        Button_OnClick.Set_Error(true, "enter your name");
    }
    else if (Name.IndexOf("'") >= 0 || Name.IndexOf("|") >= 0)
    {
        // Перевірка коректності імені
        Button_OnClick.Set_InteractableSubmitScore(true);
        Button_OnClick.Set_Error(true, "enter correct name");
    }
    else
    {
        GameObject.Find("Game_Controller").GetComponent<Submit_Score>().Post_Score(Name,
        High_Score);
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}

public void SubmitCancel_OnClick()
{
    // Деактивація поширення результатів
    Set_SubmitScoreMenu(false);
    Set_Error(false);

    // активація меню кінця гри
    Set_GameOverMenu(true);
}

public void Share_OnClick()
{
    Set_GameOverMenu(false);

    // Поширити результат
    GameObject.Find("Game_Controller").GetComponent<Share_Score>().Share();

    Set_GameOverMenu(true);
}

public static void Set_GameOverMenu(bool State)
{
    GameObject Background_Canvas = GameObject.Find("Background_Canvas");

    for (int i = 1; i < 12; i++)
        Background_Canvas.transform.GetChild(i).gameObject.SetActive(State);

    Background_Canvas.transform.GetChild(5).GetComponent<InputField>().text =
Data_Manager.Get_PlayerName();
}

public static void Set_SubmitScoreMenu(bool State)
{
    GameObject Background_Canvas = GameObject.Find("Background_Canvas");
    for (int i = 12; i < 17; i++)
        Background_Canvas.transform.GetChild(i).gameObject.SetActive(State);
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Background_Canvas.transform.GetChild(14).GetComponent<InputField>().text =
Data_Manager.Get_PlayerName();

Background_Canvas.transform.GetChild(14).GetComponent<InputField>().Select();
    }
    public static void Set_InteractableSubmitScore(bool State)
    {
        GameObject Background_Canvas = GameObject.Find("Background_Canvas");

Background_Canvas.transform.GetChild(14).GetComponent<InputField>().interactable = State;
        Background_Canvas.transform.GetChild(15).GetComponent<Button>().interactable
= State;

Background_Canvas.transform.GetChild(16).GetComponent<Button>().interactable = State;
        if(State)
        {
            Background_Canvas.transform.GetChild(15).GetComponent<Image>().color =
new Color(1, 1, 1, 1);
            Background_Canvas.transform.GetChild(16).GetComponent<Image>().color =
new Color(1, 1, 1, 1);
        }
        else
        {
            Background_Canvas.transform.GetChild(15).GetComponent<Image>().color =
new Color(1, 1, 1, 0.5f);
            Background_Canvas.transform.GetChild(16).GetComponent<Image>().color =
new Color(1, 1, 1, 0.5f);
        }
    }

    public static void Set_TopTenMenu(bool State)
    {

        GameObject HighScore_Canvas =
GameObject.Find("Game_Controller").GetComponent<Submit_Score>().HighScore_Canvas;
        HighScore_Canvas.SetActive(State);

        HighScore_Canvas.transform.GetChild(5).gameObject.SetActive(State);
    }

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public static void Set_Error(bool State)
{

    GameObject Background_Canvas = GameObject.Find("Background_Canvas");
    Background_Canvas.transform.GetChild(17).gameObject.SetActive(State);
}

public static void Set_Error(bool State, string Message)
{
    GameObject Background_Canvas = GameObject.Find("Background_Canvas");
    Background_Canvas.transform.GetChild(17).gameObject.SetActive(State);
    Background_Canvas.transform.GetChild(17).GetComponent<Text>().text =
Message;
}

public static void Set_RecordListView(int Index, string Name, int High_Score)
{
    GameObject HighScore_Canvas =
GameObject.Find("Game_Controller").GetComponent<Submit_Score>().HighScore_Canvas;

    Set_RecordLoading(false);

    HighScore_Canvas.transform.GetChild(4).GetChild(0).GetChild(Index).GetChild(0).Get
Component<Text>().text = Name;

    HighScore_Canvas.transform.GetChild(4).GetChild(0).GetChild(Index).GetChild(1).GetCompon
ent<Text>().text = High_Score.ToString();
}

public static void Set_RecordLoading(bool State)
{
    GameObject HighScore_Canvas =
GameObject.Find("Game_Controller").GetComponent<Submit_Score>().HighScore_Canvas;

    HighScore_Canvas.transform.GetChild(5).gameObject.SetActive(false);
}

public static void Set_RecordMessage(string Record_Data)
{
    GameObject HighScore_Canvas =
GameObject.Find("Game_Controller").GetComponent<Submit_Score>().HighScore_Canvas;

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        HighScore_Canvas.transform.GetChild(6).GetComponent<Text>().text
Record_Data;
    }
}

```

Скрипт Camera_Follow.cs

```

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

public class Camera_Follow : MonoBehaviour {

    public Transform Target;

    private GameObject Game_Controller;
    private bool Game_Over = false;

    private float Time_ToDown = 0;

    // Ініціалізація
    void Start()
    {
        Game_Controller = GameObject.Find("Game_Controller");
    }
    void Update()
    {
        // Перевірка кінця гри
        Game_Over = Game_Controller.GetComponent<Game_Controller>().Get_GameOver();
    }

    void FixedUpdate()
    {
        // Рух камери вниз якщо гра закінчилась
        if (Game_Over)
        {
            if(Time.time < Time_ToDown + 4f)
                transform.position -= new Vector3(0, 1f, 0);
            else
            {
                // Видалення гравця та всіх об'єктів
                GameObject Player = GameObject.FindGameObjectWithTag("Player");
            }
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

```

        GameObject[] Objects = GameObject.FindGameObjectsWithTag("Object");

        Destroy(Player);
        foreach (GameObject Obj in Objects)
            Destroy(Obj);
    }
}

void LateUpdate ()
{
    if(!Game_Over)
    {
        // if target.y > camera.y + 2
        if (Target.position.y > transform.position.y + 2)
        {
            Vector3 New_Pos = new Vector3(transform.position.x,
Target.position.y - 2, transform.position.z);
            transform.position = New_Pos;
        }

        Time_ToDown = Time.time;
    }
}
}

```

Скрипт Data_Manager.cs

```

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

public class Data_Manager
{
    private static int High_Score = 0;
    private static string Player_Name = string.Empty;
    private static string Player_Hash = string.Empty;
    public static void Set_HighScore(int Score)
    {
        High_Score = Score;
    }
    public static int Get_HighScore()
    {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return High_Score;
}
public static void Set_PlayerName(string Name)
{
Player_Name = Name;
}

public static string Get_PlayerName()
{
return Player_Name;
}

    public static void Set_PlayerHash(string Hash)
    {
        Player_Hash = Hash;
    }

    public static string Get_PlayerHash()
    {
        return Player_Hash;
    }
}

```

Скрипт File_Manager.cs

```

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

using System.IO;
public class File_Manager : MonoBehaviour {

    private static string File_Path;

    void Start ()
    {

        Screen.sleepTimeout = SleepTimeout.NeverSleep;

        File_Path = Path.Combine(Application.persistentDataPath, "Info.DJP");
        Get_Info();
    }

    private void Get_Info()
    {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if(File.Exists(File_Path))
        {
            string[] Data = File.ReadAllLines(File_Path);

            if(Data.Length != 1)
            {
                Data_Manager.Set_PlayerName(string.Empty);
                Data_Manager.Set_PlayerHash(string.Empty);
                Data_Manager.Set_HighScore(0);
            }
            else
            {
                // Convert data to byte
                byte[] Byte_Data = System.Convert.FromBase64String(Data[0]);
                // Convert to string
                string[] Str_Data =
                System.Text.Encoding.UTF8.GetString(Byte_Data).Split('|');

                if (Str_Data.Length != 3)
                {
                    Data_Manager.Set_PlayerName(string.Empty);
                    Data_Manager.Set_PlayerHash(string.Empty);
                    Data_Manager.Set_HighScore(0);
                }
                else
                {
                    Data_Manager.Set_PlayerName(Str_Data[0]);
                    Data_Manager.Set_PlayerHash(Str_Data[1]);

                    Data_Manager.Set_HighScore(System.Convert.ToInt32(Str_Data[2]));
                }
            }
        }

        public static void Save_Info()
        {
            string Str_Data = Data_Manager.Get_PlayerName() + "|" +
            Data_Manager.Get_PlayerHash() + "|" + Data_Manager.Get_HighScore();
            byte[] Byte_Data = System.Text.Encoding.UTF8.GetBytes(Str_Data);
            string[] Data = new string[1];

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Data[0] = System.Convert.ToBase64String(Byte_Data);
        File.WriteAllLines(File_Path, Data);
    }
}

```

Скрипт Game_Controller.cs

```

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

using UnityEngine.UI;

public class Game_Controller : MonoBehaviour {

    private GameObject Player;

    private float Max_Height = 0;
    public Text Txt_Score;

    private int Score;
    private Vector3 Top_Left;
    private Vector3 Camera_Pos;
    private bool Game_Over = false;
    public Text Txt_GameOverScore;
    public Text Txt_GameOverHighscore;

    void Awake ()
    {
        Player = GameObject.Find("Doodler");
        Camera_Pos = Camera.main.transform.position;
        Top_Left = Camera.main.ScreenToWorldPoint(new Vector3(0, 0, 0));
    }

    void FixedUpdate ()
    {
        if(!Game_Over)
        {
            // Підрахунок максимальної висоти
            if (Player.transform.position.y > Max_Height)
            {
                Max_Height = Player.transform.position.y;
            }
            // Перевірка падіння

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (Player.transform.position.y - Camera.main.transform.position.y <
Get_DestroyDistance())
        {
            // Музика кінця гри
            GetComponent().Play();
            Set_GameOver();
            Game_Over = true;
        }
    }

    void OnGUI()
    {
        // Встановлення рахунку
        Score = (int)(Max_Height * 50);
        Txt_Score.text = Score.ToString();
    }

    public bool Get_GameOver()
    {
        return Game_Over;
    }

    public float Get_DestroyDistance()
    {
        return Camera_Pos.y + Top_Left.y;
    }

    void Set_GameOver()
    {
        if (Data_Manager.Get_HighScore() < Score)
            Data_Manager.Set_HighScore(Score);

        Txt_GameOverScore.text = Score.ToString();
        Txt_GameOverHighscore.text = Data_Manager.Get_HighScore().ToString();
        GameObject Background_Canvas = GameObject.Find("Background_Canvas");

        // Активація меню кінця гри
        Button_OnClick.Set_GameOverMenu(true);

        Background_Canvas.GetComponent<Animator>().enabled = true;

        // Сохранение
        File_Manager.Save_Info();
    }

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}
```

Скрипт Menu_Platform.cs

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

public class Menu_Platform : MonoBehaviour {

    public float Jump_Force;

    void OnCollisionEnter2D(Collision2D Other)
    {
        // Додавання прискорення для гравця
        if (Other.relativeVelocity.y <= 0f)
        {
            Rigidbody2D Rigid = Other.collider.GetComponent<Rigidbody2D>();

            if (Rigid != null)
            {
                Vector2 Force = Rigid.velocity;
                Force.y = Jump_Force;
                Rigid.velocity = Force;

                GetComponent<AudioSource>().Play();
            }
        }
    }
}
```

Скрипт Platform.cs

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

public class Platform : MonoBehaviour {

    public float Jump_Force = 10f;
    private float Destroy_Distance;
    private bool Create_NewPlatform = false;

    private GameObject Game_Controller;
```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void Start()
{
    Game_Controller = GameObject.Find("Game_Controller");

    // Встановлення дистанції для руйнування платформ
    Destroy_Distance =
Game_Controller.GetComponent<Game_Controller>().Get_DestroyDistance();
}

void FixedUpdate()
{
    // Платформа покинула екран
    if (transform.position.y - Camera.main.transform.position.y <
Destroy_Distance)
    {
        // Створити нову платформу
        if (name != "Platform_Brown(Clone)" && name != "Spring(Clone)" && name
!= "Trampoline(Clone)" && !Create_NewPlatform)
        {

Game_Controller.GetComponent<Platform_Generator>().Generate_Platform(1);
        Create_NewPlatform = true;
        }

        // Деактивація колайдера та ефектора
        GetComponent<EdgeCollider2D>().enabled = false;
        GetComponent<PlatformEffector2D>().enabled = false;
        GetComponent<SpriteRenderer>().enabled = false;

        if (transform.childCount > 0)
        {
            if(transform.GetChild(0).GetComponent<Platform>())
            {
                transform.GetChild(0).GetComponent<EdgeCollider2D>().enabled =
false;

transform.GetChild(0).GetComponent<PlatformEffector2D>().enabled = false;
                transform.GetChild(0).GetComponent<SpriteRenderer>().enabled =
false;
            }
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!GetComponent<AudioSource>().isPlaying &&
!transform.GetChild(0).GetComponent<AudioSource>().isPlaying)
            Destroy(gameObject);
    }
    else
    {
        if (!GetComponent<AudioSource>().isPlaying)
            Destroy(gameObject);
    }
}

void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.relativeVelocity.y <= 0f)
    {
        Rigidbody2D rb = collision.collider.GetComponent<Rigidbody2D>();
        if (rb != null)
        {
            Vector2 velocity = rb.velocity;
            velocity.y = Jump_Force;
            rb.velocity = velocity;
            GetComponent<AudioSource>().Play();

            if (GetComponent<Animator>())
                GetComponent<Animator>().SetBool("Active", true);

            Platform_Type();
        }
    }
}

void Platform_Type()
{
    if (GetComponent<Platform_White>())
        GetComponent<Platform_White>().Deactivate();
    else if (GetComponent<Platform_Brown>())
        GetComponent<Platform_Brown>().Deactivate();
}
}

```

Скрипт Platform_Blue.cs

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

public class Platform_Blue : MonoBehaviour
{
    // Платформа яка рухається
    private bool To_Right = true;
    private float Offset = 1.2f;

    void FixedUpdate()
    {
        // Рух платформи
        Vector3 Top_Left = Camera.main.ScreenToWorldPoint(new Vector3(0, 0, 0));

        if (To_Right) // Праворуч
        {
            if (transform.position.x < -Top_Left.x - Offset)
                transform.position += new Vector3(0.1f, 0, 0);
            else
                To_Right = false;
        }
        else // Ліворуч
        {
            if (transform.position.x > Top_Left.x + Offset)
                transform.position -= new Vector3(0.1f, 0, 0);
            else
                To_Right = true;
        }
    }
}
```

Скрипт Platform_Brown.cs

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

public class Platform_Brown : MonoBehaviour {

    private bool Fall_Down = false;
```

```

void FixedUpdate ()
{
    if (Fall_Down)
        transform.position -= new Vector3(0, 0.15f, 0);
}

public void Deactive()
{
    GetComponent<EdgeCollider2D>().enabled = false;
    GetComponent<PlatformEffector2D>().enabled = false;

    Fall_Down = true;
}
}

```

Скрипт Platform_Generator.cs

```

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

public class Platform_Generator : MonoBehaviour {

    public GameObject Platform_Green;
    public GameObject Platform_Blue;
    public GameObject Platform_White;
    public GameObject Platform_Brown;
    public GameObject Spring;
    public GameObject Trampoline;
    public GameObject Propeller;

    private GameObject Platform;
    private GameObject Random_Object;

    public float Current_Y = 0;
    float Offset;
    Vector3 Top_Left;

    void Start ()
    {
        Top_Left = Camera.main.ScreenToWorldPoint(new Vector3(0, 0, 0));
        Offset = 1.2f;
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        // Ініціалізація платформ
        Generate_Platform(10);
    }

    void Update () {

    }

    public void Generate_Platform(int Num)
    {
        for (int i = 0; i < Num; i++)
        {
            // Підрахунок платформ x, y
            float Dist_X = Random.Range(Top_Left.x + Offset, -Top_Left.x - Offset);
            float Dist_Y = Random.Range(2f, 5f);

            // Створити коричневі платформи з 1/8 вірогідністю
            int Rand_BrownPlatform = Random.Range(1, 8);

            if (Rand_BrownPlatform == 1)
            {
                float Brown_DistX = Random.Range(Top_Left.x + Offset, -Top_Left.x -
Offset);

                float Brown_DistY = Random.Range(Current_Y + 1, Current_Y + Dist_Y
- 1);

                Vector3 BrownPlatform_Pos = new Vector3(Brown_DistX, Brown_DistY,
0);

                Instantiate(Platform_Brown,                                BrownPlatform_Pos,
Quaternion.identity);
            }

            // створення інших платформ
            Current_Y += Dist_Y;
            Vector3 Platform_Pos = new Vector3(Dist_X, Current_Y, 0);
            int Rand_Platform = Random.Range(1, 10);

            if (Rand_Platform == 1) // Створити сині платформи
                Platform      =      Instantiate(Platform_Blue,      Platform_Pos,
Quaternion.identity);

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        else if (Rand_Platform == 2) // Створити білі платформи
            Platform = Instantiate(Platform_White, Platform_Pos,
Quaternion.identity);
        else // Створити зелену платформу
            Platform = Instantiate(Platform_Green, Platform_Pos,
Quaternion.identity);
        if (Rand_Platform != 2)
        {
            // Створення випадкових предметів
            int Rand_Object = Random.Range(1, 40);

            if (Rand_Object == 4)
            {
                // Створити пружину
                Vector3 Spring_Pos = new Vector3(Platform_Pos.x + 0.5f,
Platform_Pos.y + 0.27f, 0);
                Random_Object = Instantiate(Spring, Spring_Pos,
Quaternion.identity);

                // Встановлення батьківського об'єкту
                Random_Object.transform.parent = Platform.transform;
            }
            else if (Rand_Object == 7) // створити трамплін
            {
                Vector3 Trampoline_Pos = new Vector3(Platform_Pos.x + 0.13f,
Platform_Pos.y + 0.25f, 0);
                Random_Object = Instantiate(Trampoline, Trampoline_Pos,
Quaternion.identity);

                Random_Object.transform.parent = Platform.transform;
            }
            else if (Rand_Object == 15) // Створити пропелер
            {
                Vector3 Propeller_Pos = new Vector3(Platform_Pos.x + 0.13f,
Platform_Pos.y + 0.15f, 0);
                Random_Object = Instantiate(Propeller, Propeller_Pos, Quaternion.identity);

                Random_Object.transform.parent = Platform.transform;
            }
        }

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}
}
}

```

Скрипт Platform_White.cs

```

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

public class Platform_White : MonoBehaviour {

    public void Deactive()
    {
        GetComponent<EdgeCollider2D>().enabled = false;
        GetComponent<PlatformEffector2D>().enabled = false;
    }
}

```

Player_Controller.cs

```

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

public class Player_Controller : MonoBehaviour {
    Rigidbody2D Rigid;
    public float Movement_Speed = 10f;
    private float Movement = 0;
    private Vector3 Player_LocalScale;

    public Sprite[] Spr_Player = new Sprite[2];

    void Start ()
    {
        Rigid = GetComponent<Rigidbody2D>();
        Player_LocalScale = transform.localScale;
    }

    void Update ()
    {
        // Створити значення руху
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Movement = Input.acceleration.x * Movement_Speed;
//Input.GetAxis("Horizontal") * Movement_Speed; //Input.acceleration.x * Movement_Speed;

// Гравець дивиться на право чи ліво
if (Movement > 0)
    transform.localScale = new Vector3(Player_LocalScale.x,
Player_LocalScale.y, Player_LocalScale.z);
    else if (Movement < 0)
        transform.localScale = new Vector3(-Player_LocalScale.x,
Player_LocalScale.y, Player_LocalScale.z);
    }
void FixedUpdate()
{
    // Підрахунок швидкості гравця
    Vector2 Velocity = Rigid.velocity;
    Velocity.x = Movement;
    Rigid.velocity = Velocity;

    // Зміна спрайту гравця
    if (Velocity.y < 0)
    {
        GetComponent<SpriteRenderer>().sprite = Spr_Player[0];

        // Активація колайдеру
        GetComponent<BoxCollider2D>().enabled = true;

        // Падання пропеллеру
        Propeller_Fall();
    }
    else
    {
        GetComponent<SpriteRenderer>().sprite = Spr_Player[1];

        // Деактивація колайдеру
        GetComponent<BoxCollider2D>().enabled = false;
    }

    // перехід через стіну на інший бік
    Vector3 Top_Left = Camera.main.ScreenToWorldPoint(new Vector3(0, 0, 0));
    float Offset = 0.5f;
    if (transform.position.x > -Top_Left.x + Offset)

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        transform.position = new Vector3(Top_Left.x - Offset,
transform.position.y, transform.position.z);
        else if(transform.position.x < Top_Left.x - Offset)
            transform.position = new Vector3(-Top_Left.x + Offset,
transform.position.y, transform.position.z);
    }

    void Propeller_Fall()
    {
        if (transform.childCount > 0)
        {
            transform.GetChild(0).GetComponent<Animator>().SetBool("Active",
false);

            transform.GetChild(0).GetComponent<Propeller>().Set_Fall(gameObject);
            transform.GetChild(0).parent = null;
        }
    }
}

```

Скрипт Propeller.cs

```

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

public class Propeller : MonoBehaviour {

    private bool Attach = false;
    private bool Fall = false;
    private float Destroy_Distance;
    private GameObject Game_Controller;

    void Start()
    {
        Game_Controller = GameObject.Find("Game_Controller");

        // Дистанція знищення пропелеру
        Destroy_Distance =
Game_Controller.GetComponent<Game_Controller>().Get_DestroyDistance();
    }

    void FixedUpdate ()
    {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if(Fall)
        {
            GetComponent().Stop();
            transform.Rotate(new Vector3(0, 0, -3.5f));
            transform.position -= new Vector3(0, 0.3f, 0);

            if (transform.position.y - Camera.main.transform.position.y <
Destroy_Distance)

                Destroy(gameObject);

        }
    }

    void OnCollisionEnter2D(Collision2D Other)
    {
        if (Other.gameObject.tag == "Player" && !Attach)
        {
            if (Other.transform.childCount == 0)
            {

                transform.parent = Other.transform;
                transform.localPosition = new Vector3(0, -0.02f, 0);
                GetComponent<BoxCollider2D>().enabled = false;

                // Додавання прискорення вгору
                Rigidbody2D Rigid = Other.collider.GetComponent<Rigidbody2D>();

                if (Rigid != null)
                {
                    Vector2 Force = Rigid.velocity;
                    Force.y = 80f;
                    Rigid.velocity = Force;

                    GetComponent<AudioSource>().Play();

                    GetComponent<Animator>().SetBool("Active", true);

                    GetComponent<SpriteRenderer>().sortingOrder = 12;
                }

                Attach = true;
            }
        }
    }

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public void Set_Fall(GameObject Player)
    {
        Fall = true;
        Player.GetComponent<BoxCollider2D>().enabled = true;
    }
}

```

Скрипт Share_Score.cs

```

using System.Collections;
using UnityEngine;

using System.IO;
using UnityEngine.UI;

public class Share_Score : MonoBehaviour {

    public GameObject Share_Canvas;

    private bool Processing = false;
    private bool Focus = false;

    public void Share()
    {
        if(!Processing)
        {
            Share_Canvas.SetActive(true);
            Share_Canvas.transform.GetChild(4).GetComponent<Text>().text =
Data_Manager.Get_HighScore().ToString();

            StartCoroutine(Share_ScreenShot());
        }
    }

    IEnumerator Share_ScreenShot()
    {
        Processing = true;

        yield return new WaitForEndOfFrame();

        ScreenCapture.CaptureScreenshot("DoodleJump_Score.png", 2);
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        string    Dest_Path    =    Path.Combine(Application.persistentDataPath,
"DoodleJump_Score.png");

        yield return new WaitForSecondsRealtime(0.3f);

        if(!Application.isEditor)
        {
            AndroidJavaClass    Intent_Class    =    new
AndroidJavaClass("android.content.Intent");
            AndroidJavaObject    Intent_Object    =    new
AndroidJavaObject("android.content.Intent");
            Intent_Object.Call<AndroidJavaObject>("setAction",
Intent_Class.GetStatic<string>("ACTION_SEND"));

            AndroidJavaClass Uri_Class = new AndroidJavaClass("android.net.Uri");
            AndroidJavaObject    Uri_Object    =
Uri_Class.CallStatic<AndroidJavaObject>("parse", "file://" + Dest_Path);

            Intent_Object.Call<AndroidJavaObject>("putExtra",
Intent_Class.GetStatic<string>("EXTRA_STREAM"), Uri_Object);
            Intent_Object.Call<AndroidJavaObject>("putExtra",
Intent_Class.GetStatic<string>("EXTRA_TEXT"), "Can you beat my score?\n\n#Doodle_Jump");
            Intent_Object.Call<AndroidJavaObject>("setType", "image/jpeg");

            AndroidJavaClass    Unity_Class    =    new
AndroidJavaClass("com.unity3d.player.UnityPlayer");
            AndroidJavaObject    CurrentActivity_Object    =
Unity_Class.GetStatic<AndroidJavaObject>("currentActivity");

            AndroidJavaObject    Chooser_Object    =
Intent_Class.CallStatic<AndroidJavaObject>("createChooser", Intent_Object, "Share your
new score");

            CurrentActivity_Object.Call("startActivity", Chooser_Object);

            yield return new WaitForSecondsRealtime(1f);
        }

        yield return new WaitUntil(() => Focus);

        Share_Canvas.SetActive(false);
        Processing = false;
    }

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        private void OnApplicationFocus(bool State)
        {
            Focus = State;
        }
    }
}

```

Скрипт Submit_Score.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using Newtonsoft.Json;

public class SubmitCallBack
{
    public string Your_CallBack { get; set; }

    public SubmitCallBack(string CallBack)
    {
        Your_CallBack = CallBack;
    }
}

public class SubmitInfo
{
    public string Your_Data { get; set; }

    public SubmitInfo(string Data)
    {
        Your_Data = Data;
    }
}

public class UserRecord
{
    public string Player_Information { get; set; }
    public int Player_HighScore { get; set; }

    public UserRecord(string Player_Info, int Player_Score)
    {
        Player_Information = Player_Info;
        Player_HighScore = Player_Score;
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

public class Submit_Score : MonoBehaviour
{

    public GameObject HighScore_Canvas;
    public static string Url = "Your server URL";

    public void Post_Score(string Name, int High_Score)
    {
        if (Name != Data_Manager.Get_PlayerName())
        {
            Data_Manager.Set_PlayerName(Name);
            Data_Manager.Set_HighScore(High_Score);

            string Hash_Data = Generate_Hash("Your info");
            Data_Manager.Set_PlayerHash(Hash_Data);
        }

        SubmitInfo Info = new SubmitInfo("Your data");
        StartCoroutine(Post(Info));
    }

    public void Show_TopTen()
    {
        SubmitInfo Info = new SubmitInfo("Your data");
        StartCoroutine(TopTen_Records(Info));
    }
    private string Generate_Hash(string Hash_Data)
    {
        return "Your hash";
    }

    IEnumerator Post(SubmitInfo Info)
    {
        SubmitCallBack Call_Back;

        string Submit_Url = Url + "/YourWebPage";

        var Request = new UnityWebRequest(Submit_Url, "POST");

```

```

byte[] Body_Raw =
System.Text.Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(Info));
Request.uploadHandler = (UploadHandler)new UploadHandlerRaw(Body_Raw);
Request.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
Request.SetRequestHeader("Content-Type", "application/json");
yield return Request.Send();

if (Request.responseCode == 200)
{
    string Content = Request.downloadHandler.text;
    Call_Back = JsonConvert.DeserializeObject<SubmitCallBack>(Content);

    if (Call_Back.Your_CallBack == "Insert or Update your information")
    {
        File_Manager.Save_Info();
        Button_OnClick.Set_SubmitScoreMenu(false);
        Button_OnClick.Set_TopTenMenu(true);

        StartCoroutine(TopTen_Records(Info));
    }
    else if (Call_Back.Your_CallBack == "Name is taken")
    {
        Button_OnClick.Set_InteractableSubmitScore(true);
        Button_OnClick.Set_Error(true, "this name is taken");
    }
    else
    {
        Button_OnClick.Set_InteractableSubmitScore(true);
        Button_OnClick.Set_Error(true, "server error");
    }
}
else
{
    Button_OnClick.Set_InteractableSubmitScore(true);
    Button_OnClick.Set_Error(true, "server error");
}
}

IEnumerator TopTen_Records(SubmitInfo Info)
{
    List<UserRecord> Records = new List<UserRecord>();

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

string Records_Url = Url + "/YourWebPage";

var Request = new UnityWebRequest(Records_Url, "POST");

byte[] Body_Raw =
System.Text.Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(Info));
Request.uploadHandler = (UploadHandler)new UploadHandlerRaw(Body_Raw);
Request.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
Request.SetRequestHeader("Content-Type", "application/json");
yield return Request.Send();

if (Request.responseCode == 200)
{
    string Content = Request.downloadHandler.text;
    Records = JsonConvert.DeserializeObject<List<UserRecord>>(Content);

    if (Records[0].Player_Information == "your server has error")
    {
        Button_OnClick.Set_RecordMessage("server error");
        Button_OnClick.Set_RecordLoading(false);
    }
    else if (Records[0].Player_Information == "There are no player yet")
    {
        Button_OnClick.Set_RecordMessage("list is empty!");
        Button_OnClick.Set_RecordLoading(false);
    }
    else
    {
        for (int i = 0; i < Records.Count - 1; i++)
            Button_OnClick.Set_RecordListView(i, (i + 1) + ". " +
Records[i].Player_Information, Records[i].Player_HighScore);
        Button_OnClick.Set_RecordMessage(Records[Records.Count - 1].Player_Information + ":
" + Records[Records.Count - 1].Player_HighScore);
    }
}
else
{
    Button_OnClick.Set_RecordMessage("server error");
    Button_OnClick.Set_RecordLoading(false);
}
}
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		