МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «**РОЗРОБКА МОБІЛЬНОЇ ГРИ ADVENTURE/PUZZLE НА ПЛАТФОРМІ UNITY**»

Виконала: студентка 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
А.В. Кацура
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Мильцев О.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Кацурі Анастасії Володимирівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка мобільної гри Adventure/Puzzle на платформі Unity

керівник роботи Мильцев Олександр Михайлович, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка мобільної гри Adventure/Puzzle на платформі Unity.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	23.01.2024	
3.	Обробка методичних та теоретичних	08.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	16.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2024	

Студент _____ А.В. Кацура
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М. Мильцев
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка мобільної гри Adventure/Puzzle на платформі Unity»: 54 с., 23 рис., 8 джерел.

КОРИСТУВАЧ, ASSET STORE, C#, PHOTOSHOP, UNITY.

Об’єкт дослідження – розробка та створення мобільної гри в жанрі Adventure/Puzzle з використанням середовища розробки Unity.

Мета роботи: створення цікавої та унікальної мобільної гри, яка поєднує елементи жанрів Adventure та Puzzle на платформі Unity для задоволення користувачів та їхнього позитивного часовитратного досвіду.

Метод дослідження – методи програмної інженерії.

Результати та їх новизна: створена гра відповідає сучасним вимогам мобільних ігор, має цікавий геймплей та привабливий дизайн, що робить її унікальною серед інших ігор жанру.

Взаємозв’язок з іншими роботами: цей диплом базується на передових практиках розробки мобільних ігор з використанням платформи Unity та мови програмування C#. Вивчення та використання цих технологій ґрунтується на попередніх дослідженнях, що дозволило ефективно реалізувати гру в жанрі Adventure/Puzzle.

Рекомендації щодо використання результатів роботи: розроблена гра може бути використана як для особистого використання гравцями, так і для комерційних цілей у сфері мобільних додатків та розваг.

Значимість роботи та висновки: розробка мобільної гри жанру Adventure/Puzzle на платформі Unity є важливим кроком у вдосконаленні мобільних розваг через кілька ключових факторів. Ця гра поєднує в собі елементи пригодницького та головоломкового геймплею, що робить її більш привабливою для широкого кола гравців. Такий поєднаний підхід дозволяє користувачам отримати різноманітність від ігрового процесу, забезпечуючи не лише розвагу, а й виклик для інтелекту.

SUMMARY

Bachelor’s qualifying paper “Development of an Adventure/Puzzle Mobile Game on the Unity Platform”: 54 p., 23 figures, 8 references.

USER, ASSET STORE, C#, PHOTOSHOP, UNITY.

The object of the study is development and creation of a mobile game in the genre of Adventure/Puzzle using the Unity development environment.

The aim of the study is to create an interesting and unique mobile game that combines elements of the Adventure and Puzzle genres on the Unity platform for user satisfaction and positive time-consuming experience.

The method of research is software engineering methods.

Results and novelty: the created game meets the modern requirements of mobile games, has interesting gameplay and attractive design, which makes it unique among other games of the genre.

Relationship to other works: this thesis is based on the best practices of mobile game development using the Unity platform and the C# programming language. The study and use of these technologies is based on previous research, which allowed for the effective implementation of an Adventure/Puzzle game.

Recommendations for using the results of the work: the developed game can be used both for personal use by players and for commercial purposes in the field of mobile applications and entertainment.

Significance of the work and conclusions: the development of a mobile game of the Adventure/Puzzle genre on the Unity platform is an important step in improving mobile entertainment due to several key factors. This game combines elements of adventure and puzzle gameplay, which makes it more attractive to a wide range of players. This combined approach allows users to get variety from the gameplay, providing not only entertainment but also a challenge for the intellect.

ЗМІСТ

ВСТУП

Розвиток технологій у сучасному світі розширює можливості мобільних ігор, роблячи їх не тільки джерелом розваг, але й засобом для розвитку когнітивних навичок гравців. Нові можливості, що надає платформа Unity, дозволяють створювати інноваційні ігрові вирішення, що викликають інтерес та підвищують рівень взаємодії з аудиторією. Застосування передових технологій у створенні мобільних ігор є не лише тенденцією, але й стратегічним кроком у покращенні якості ігрового досвіду. Такі ігри стають не тільки важливим аспектом розваг, а й платформою для розвитку розумових здібностей та навичок у гравців.

Гра в жанрі Adventure/Puzzle, заснована на створенні комбінацій з числами. Це вирізняється як відмінний інструмент для стимуляції когнітивних функцій, тому процес допоможе не лише забавляти гравців, а й покращувати їхні когнітивні вміння під час приємного проведення часу, що є дуже важливим у нашому сучасному інформаційному суспільстві.

Процес гри полягатиме в розв’язанні головоломок, де гравець має створювати комбінації із чисел, що збігаються. Все це буде заохочуватися новими можливостями для гравця. Використання платформи Unity дозволить створити захоплююче інтерактивне середовище з приємною графікою та захоплюючим геймплеєм. Такий підхід до розробки ігор створює унікальний інтерфейс взаємодії, що сприяє глибокому поглибленню у гру та позитивному ігровому досвіду.

Мета цього проєкту – розробити цікаву гру, яка не тільки розважатиме, але й стимулюватиме увагу та мислення гравців. Використання сучасних технологій розробки дозволить створити інтерактивний продукт, що задовольняє вимоги сучасних гравців та сприяє їхньому розвитку. Це додає в гру аспекти, які не тільки цікаві для гравців, але й корисні з точки зору когнітивного розвитку та вирішення складних завдань.

1 ТЕОРЕТИЧНИЙ ОГЛЯД ТА ВИВЧЕННЯ UNITY

1.1 Огляд Unity, та його використання

Unity – це інтегроване середовище розробки для створення ігор та інших інтерактивних додатків. Воно володіє великим набором інструментів для розробки, від графіки та анімації до програмування та фізики. Unity дозволяє створювати ігри для різних платформ, включаючи на мобільні пристрої, комп’ютери та консолі [2].

Основними особливостями Unity є:

- Unity використовує мову програмування C# для розробки логіки гри та інтерактивних скриптів (це дозволяє розробникам створювати складні ігрові механіки та взаємодію з об’єктами у грі) [3];
- візуальний редактор: Unity має потужний візуальний редактор, який дозволяє створювати графічні ефекти, анімацію, розміщувати об’єкти у сцені та керувати компонентами гри без кодування [2];
- Unity дозволяє розробляти ігри для різних платформ, включаючи iOS, Android, Windows, macOS, Xbox, PlayStation та інші [2];
- Unity має велику та активну спільноту розробників, яка допомагає вирішувати проблеми, ділиться досвідом та розробляє різні розширення та ресурси для розробки.

Unity використовується для створення різних типів ігор та інтерактивних додатків, включаючи:

- Unity дозволяє створювати якісні та захоплюючі ігри для мобільних пристроїв з використанням різних функцій та можливостей, таких як ефекти, анімація та фізика;
- Unity також підтримує розробку ігор для консолей, таких як Xbox та PlayStation, що дозволяє розробникам створювати гри для широкої аудиторії гравців;

– Unity має підтримку для розробки ігор та додатків у сферах VR та AR, що робить його важливим інструментом для розробників, які цікавляться цими технологіями.

Unity надає широкі можливості для створення якісних та захоплюючих ігор та інтерактивних додатків для різних платформ та аудиторій, тому він є універсальним додатком для створення ігор на будь які платформи.

1.2 Можливості розробки мобільної гри на платформі Unity

Unity, як інструмент розробки ігор, надає широкі можливості для створення захоплюючих та ефективних проєктів. Основні переваги Unity для розробки мобільних ігор включають:

– Unity має інтуїтивний та зручний інтерфейс, що дозволяє створювати ігри без глибоких знань програмування;
– ігри, розроблені на Unity, можуть бути запущені на різних мобільних платформах, таких як iOS та Android, що спрощує їх розповсюдження та монетизацію;
– Unity має велику кількість готових ресурсів, таких як моделі, анімації та ефекти, що полегшує розробку ігор і дозволяє швидше випускати гру на ринок;
– Unity надає можливості для створення реалістичної графіки, ефектів світла та звуку, що робить ігри більш захоплюючими та привабливими для гравців;
– розробники можуть створювати різні режими гри, генерувати випадкові рівні, додавати нові функції та персонажів, що забезпечує гнучкість у вдосконаленні гри та збільшенні її привабливості для гравців;
– Unity має велику спільноту розробників та форуми підтримки, де можна знайти відповіді на питання, поради та рекомендації щодо

розробки [2].

Незважаючи на ці переваги, важливо враховувати і недоліки розробки мобільних ігор:

- складні графічні ефекти та анімація можуть вимагати значних ресурсів мобільного пристрою, що може вплинути на продуктивність гри та споживання енергії;
- збільшення кількості ресурсів та контенту може призвести до зростання розміру гри, що може бути проблематичним для завантаження та установки для деяких користувачів;
- для реалізації деяких унікальних функцій або ефектів можуть знадобитися додаткові плагіни або розробка власних модулів, що може вимагати додаткового часу та експертизи в розробці;
- наявність різних варіантів ліцензування та можливість оплати за додатковий контент чи функції може становити фінансові витрати для розробників гри.

1.3 Висновки до розділу 1

У данному розділі ми розібрали, чому Unity є потужним інструментом для розробки ігор та інтерактивних додатків на різних платформах. Він надає широкий спектр можливостей для творчості та гнучкість у вдосконаленні проєктів.

Розглянули основні переваги використання Unity, які включають зручний інтерфейс розробки, можливості мультиплатформенності, велику бібліотеку ресурсів, реалістичну графіку та ефекти, а також недоліки, такі як: висока вимогливість до ресурсів, великий розмір гри та обмеженість унікальних функцій.

Отже, можна сказати, що Unity – це не просто інструмент, а повноцінне середовище, яке надає розробникам безліч можливостей для творчості та

реалізації своїх ідей. Зберігаючи свою лідерську позицію в галузі, підтверджуючи свою ефективність через успішні проєкти, такі як «Angry Birds», «Pokémon GO», «Hearthstone» та багато інших, використання Unity виявляється відмінним вибором для розробки ігор та інтерактивних додатків, що сприяє досягненню значних результатів у цій сфері.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Стек технологій

C# використовується для розробки логіки гри та взаємодії різних об’єктів. За допомогою C# можна створювати складні ігрові механіки, об’єкти та реалізовувати взаємодію з користувачем [1].

Для створення графіки, текстур, анімацій та інших візуальних елементів гри було обрано Adobe Photoshop [4]. Графіка гри є ключовим елементом для створення атмосфери та привабливості гри для гравців.

Unity Asset Store – це платформа, де можна знайти готові ассети для Unity, які полегшують розробку гри. Тут можна знайти моделі персонажів, анімації, звукові ефекти, шаблони рівнів та інші корисні ресурси.

Додатковим інструментом для створення ряду графічних елементів, анімацій чи сцен є Paint. В нашому проєкті використовуються спрайти – двомірні графічні зображення, що використовуються у вебдизайні та розробці ігор для відображення об’єктів, персонажів, ефектів та інших елементів.

Завдяки Paint та іншим графічним редакторам ми зможемо створити ці графічні зображення з необхідними деталями та розмірами, які допомагають зробити гру чи додаток більш захоплюючими та привабливими для користувачів.

Unity Engine – це основний інструмент для створення гри. Він надає широкий набір інструментів для роботи з графікою, анімацією, фізикою та програмуванням.

Також Unity Engine підтримує складну фізичну модель, що дозволяє створювати реалістичні фізичні ефекти, такі як рух об’єктів, зіткнення та інші фізичні взаємодії.

У нашому проєкті Unity використовується для створення високоякісної

графіки, взаємодії об’єктів та реалізації ігрових механік, а також використовується для компіляції проєкту на мобільних пристроях, щоб гравці могли насолоджуватись грою на своїх телефонах чи планшетах.

Всіх цих інструментів буде достатньо для створення повноцінної мобільної гри, в якої буде реалізована логіка, графіка та звуковими ефектами.

2.2 Постанова задачі

До появи Unity та інших сучасних ігрових двигунів ігри розроблялися з використанням різних інструментів і технологій, залежно від того, на якій платформі передбачалося їх запуск. Багато великих студій та видавців створювали свої власні ігрові двигуни для кожної гри. Проблемою такого підходу була складність і витратність розробки. Кожен розробник чи студія повинні були вкласти значні зусилля в створення і оптимізацію власного двигуна або адаптацію гри під різні платформи. Це займало багато часу і ресурсів, що робило процес розробки повільним та витратним. Крім того, підтримка різних платформ вимагала додаткових зусиль та навичок.

З появою Unity та інших ігрових двигунів, ці проблеми були значно полегшені. Розробники тепер могли використовувати готові інструменти та рішення, що суттєво скоротило час і витрати на розробку ігор. Наприклад, створення гри, яка працює на різних платформах, стало набагато простіше завдяки можливостям кроссплатформенної розробки, які надає Unity.

Тому найкращим варіантом, було реалізувати нашу гру на Unity, оскільки цей ігровий двигун дозволяє нам зосередитися на творчості та інноваціях у процесі розробки, не турбуючись про складність адаптації гри під різні платформи, оптимізації коду та інших проблемах, з якими стикалися розробники раніше,. Також, будучи широко використовуваним у галузі ігрової розробки, Unity також надає доступ до великого співтовариства розробників, різноманітних ресурсів та плагінів, що полегшує процес

створення якісної та привабливої гри для наших користувачів.

Наша робота включає в себе розробку гри, яка акцентується на розвитку когнітивних навичок у користувача. Для цього потрібно спланувати логіку гри, гемплей, який повинен бути унікальним та цікавим, а також гарний дизайн.

Основними етапами нашої роботи є спланування логіки гри та створення унікального геймплею, який зацікавить гравців. Перед початком розробки гри ми детально працюємо над дизайном персонажів, об’єктів та фону, щоб створити візуально привабливу та логічно зв’язану гру.

Для реалізації візуальної частини ми використовуємо графічні редактори, такі як SAI, Paint та Adobe Photoshop, щоб створити спрайти у піксельному стилі. Ці спрайти виступають у ролі графічних елементів гри, які відображають об’єкти, персонажів та різноманітні декорації.

Після створення спрайтів ми переходимо до реалізації дизайну та об’єктів в самій грі. Ми розробляємо логіку гри, включаючи взаємодію користувача з об’єктами, зберігання даних, генерацію цифр.

Кінцевий результат нашої роботи має бути представлений у вигляді цікавої та захоплюючої гри, яка забезпечить користувачам не лише відпочинок, але й навчальний досвід. Ця гра повинна бути цікавою для широкого кола користувачів та спонукати їх до активної участі.

2.3 Створення діаграм

Діаграма прецедентів є важливим інструментом у проєктуванні програмного забезпечення, який показує, як система або програма взаємодіє зі своїми користувачами, відомими як актори. Ця діаграма складається з акторів, прецедентів і взаємодій між ними. Актори можуть бути користувачами або іншими системами, що взаємодіють з основною системою. Прецеденти описують дії, які система або її частини можуть

виконувати. Взаємодії визначають, як актори взаємодіють із системою. Діаграма прецедентів допомагає зрозуміти потреби користувачів і функціональність системи, а також служить ефективним засобом комунікації між розробниками і клієнтами завдяки чіткому візуальному представленню використання програмного забезпечення. На рисунку 2.1 наведено таку діаграмму для нашого проєкту.

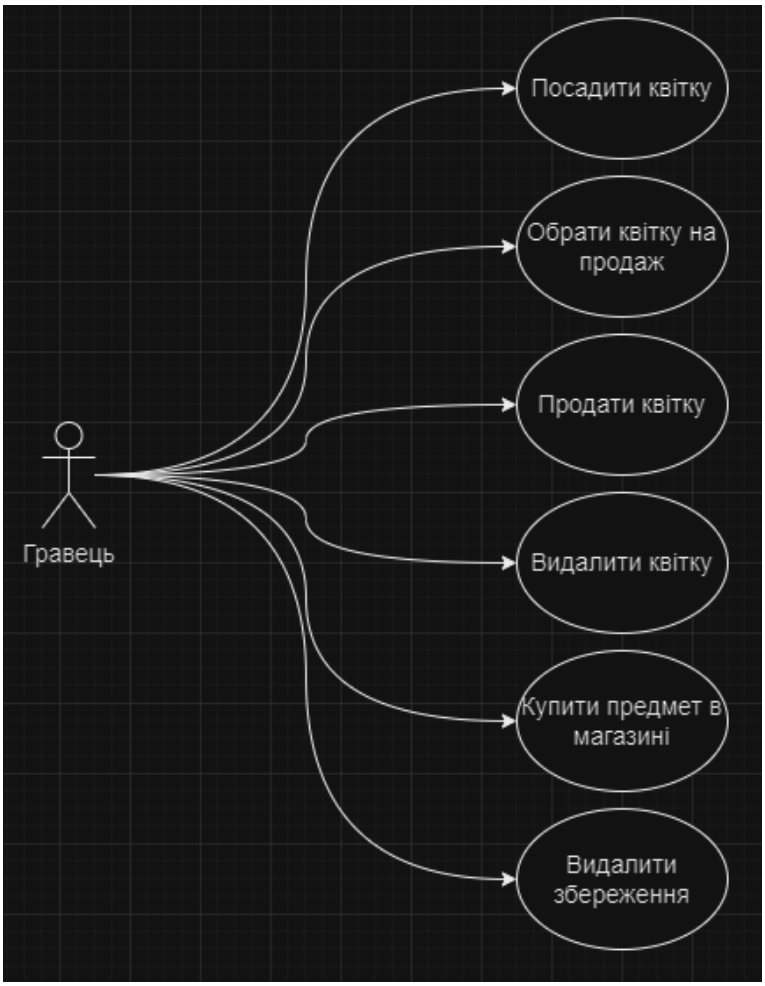

Рисунок 2.1 – Діаграма прецедентів

Діаграма послідовності відображає, як об’єкти взаємодіють один з одним в заданій послідовності. Вона показує, як об’єкти обмінюються повідомленнями (викликами методів) для виконання певної функції або процесу. Розберемо її основні компоненти. Об’єкти або актори (вертикальні лінії) представляють класи або компоненти системи, які взаємодіють між собою. Повідомлення (горизонтальні стрілки) вказують на обмін

інформацією між об’єктами. Часова шкала (вертикальна) показує послідовність подій зверху вниз.

Діаграма послідовності дозволяє візуально зрозуміти, як різні об’єкти гри взаємодіють між собою в певній послідовності. Це сприяє кращому розумінню логіки гри та виявленню можливих проблем. На рисунку 2.2 наведено таку діаграмму для нашого проєкту.

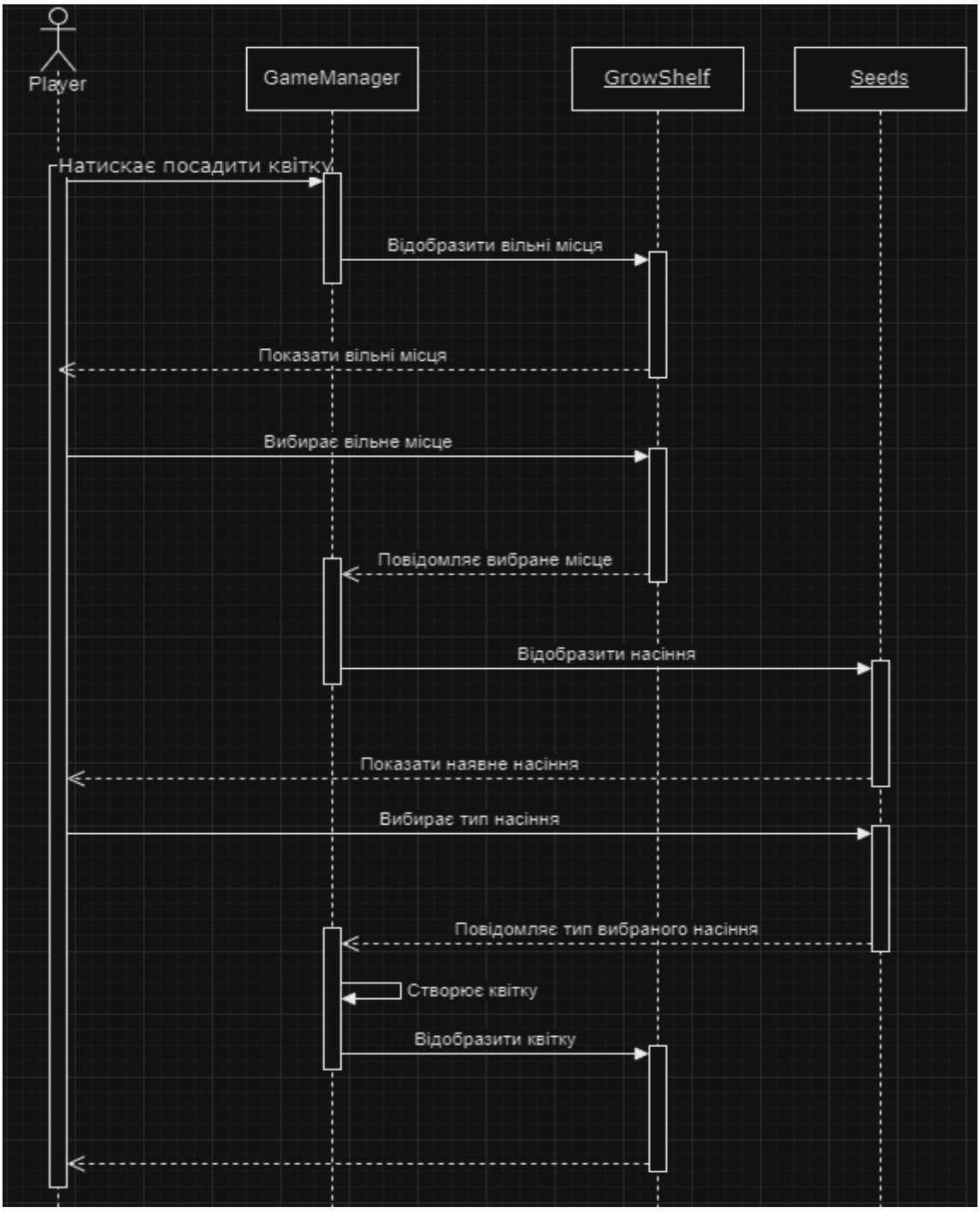

Рисунок 2.2 – Діаграма послідовності

Діаграма компонентів використовується для відображення високорівневої архітектури системи, зокрема для представлення програмних компонентів і їх взаємозв’язків. Вона показує, як різні компоненти системи співпрацюють між собою, які інтерфейси вони використовують і як вони залежні один від одного. Розберемо її основні компоненти. Компоненти представляють модулі або частини системи, які можуть бути повторно використані. Це можуть бути окремі класи, бібліотеки, модулі або підсистеми. На діаграмі компоненти зображені як прямокутники з іконкою у верхньому правому куті, що нагадує частину головоломки. Інтерфейси показують, як компоненти взаємодіють один з одним через визначені інтерфейси. Інтерфейси можуть бути реалізовані або використовувані компонентами. Взаємозв’язки показують залежності між компонентами, тобто які компоненти використовують інші компоненти або з якими взаємодіють. На рисунку 2.3 наведено таку діаграмму для нашого проєкту.

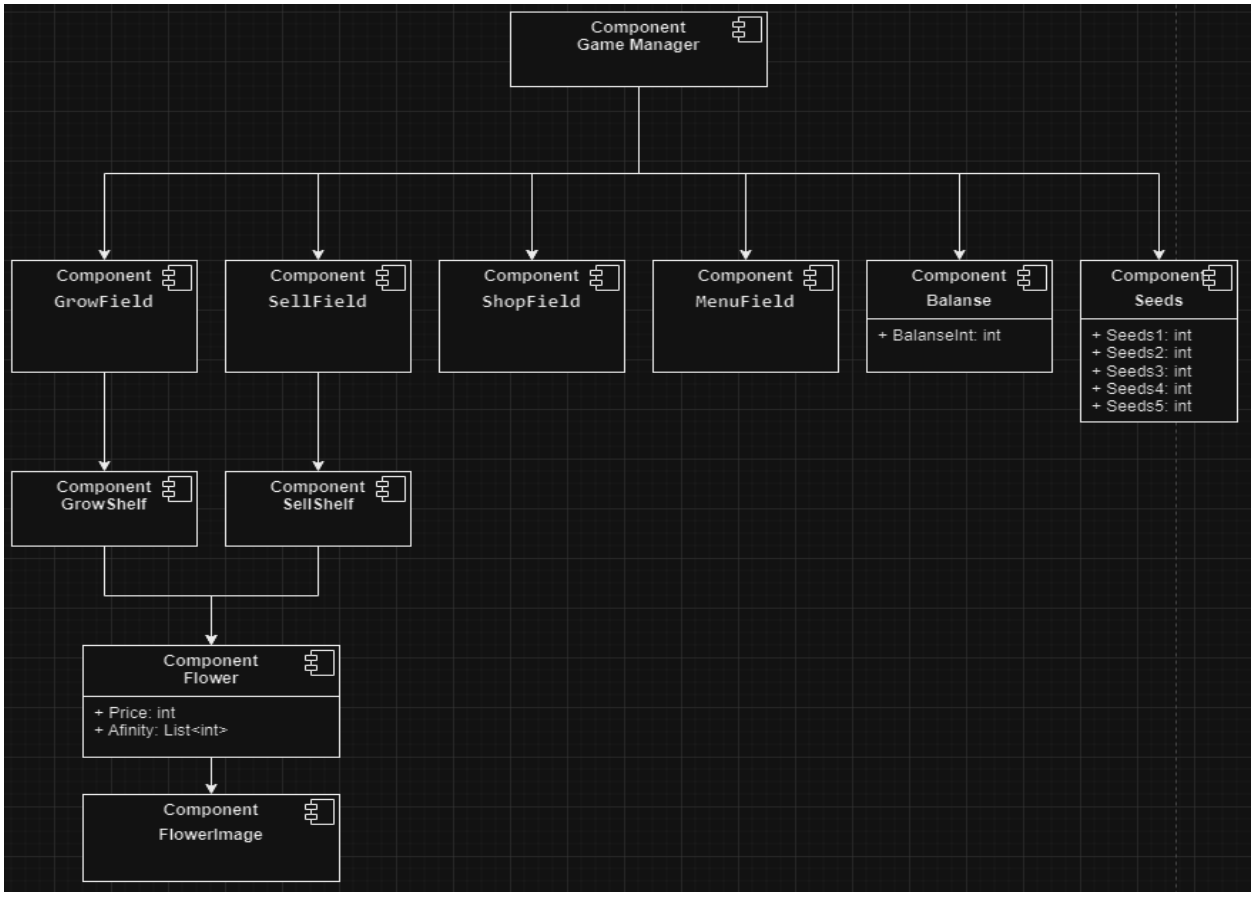

Рисунок 2.3 – Діаграма компонентів

Класова діаграма є однією з найважливіших діаграм, що використовується для моделювання статичної структури системи. Вона показує класи, їхні атрибути, методи, а також взаємозв’язки між класами. Розберемо її основні компоненти. Класи відображають сутності системи, включаючи їхні атрибути (поля) та методи (функції або операції). Атрибути – це поля або властивості класу, що описують характеристики об’єктів класу. Методи – функції або операції класу, що визначають поведінку об’єктів класу. Взаємозв’язки – відносини між класами, які можуть бути різних типів: асоціація, агрегація, композиція, наслідування. Множинність – вказує кількість об’єктів, які можуть брати участь у відносинах між класами.

Класова діаграма допомагає зрозуміти, як організовані класи в проєкті, які атрибути та методи вони містять, і як вони взаємодіють один з одним. За допомогою класової діаграми можна чітко спланувати розробку гри, визначити, які класи необхідно створити.

Продивитись классову діаграму можна за посиланням [8].

2.4 Опис ігрового процессу

Основним завданням гравця є вирощування та продаж квітів, з метою набору максимальної кількості ігрових очок. Система очок ретельно розроблена, щоб стимулювати стратегічне мислення та оптимальне розміщення квітів на спеціальних полях.

Ігровий процес складається з декількох етапів. Спочатку гравець садить квіти на ігровому полі для росту. Через певний час квіти виростають, і їх можна переносити на поле для продажу. Кожна квітка має свою базову вартість, яка може бути збільшена залежно від її характеристик і сусідніх квітів.

Щоб отримати максимальну кількість очок, гравець має стратегічно розставити вирощені квіти по комірках на полі для продажу. Якщо квітка має

характеристику, що впливає на сусідні квіти, її вартість збільшується, що безпосередньо впливає на загальний результат. Гравець може бачити зміни в сумарній вартості квітів під час переміщення їх по комірках, що допомагає в ухваленні рішень.

Перетягувати квіти з полиці на полицю на одному ігровому полі можна не тільки на вільне місце, а й на зайняте місце іншою квіткою. У такому разі ці дві квітки поміняються місцями.

Залежно від рівня насіння, з нього може вирости квітка з такою вартістю:

- 1 рівень – вартість 1;
- 2 рівень – вартість 2 або 3 (обирається випадково);
- 3 рівень – вартість 4 або 5 (обирається випадково);
- 4 рівень – вартість 6 або 7 (обирається випадково);
- 5 рівень – вартість 8 або 9 (обирається випадково).

Після завершення розставляння квітів по комірках, гравець може продати їх одним натисканням кнопки. Основна мета – зібрати якомога більше очок під час продажу квітів, постійно знаходячи нові вигідні комбінації їх розміщення. Оскільки квіти виростають із випадковими характеристиками, щоразу потрібна нова стратегія для максимізації очок.

Зароблені очки можна витратити в магазині на придбання поліпшень.

Предмети, які є в магазині:

- полиці, що збільшують кількість місць для посадки;
- полиці, що збільшують кількість місць для розміщення квітів під час продажу;
- насіння 1 рівня;
- квіти в підвісних горщиках, наявність кожного з яких на ігровому полі grow скорочує час росту квітів на значний відсоток.

Так само в окремій вкладці є вже зрощені квіти 1 рівня, випадкові характеристики яких, визначаються в момент придбання. Ці квіти гравцеві обійдуться дорожче, ніж якби він самостійно їх вирощував із насіння, але

іноді їхнє придбання може бути виправдане ігровими моментами.

Кількість придбаних предметів у магазині обмежена, окрім квітів 1 рівня та насіння, які гравець може купувати необмежену кількість разів.

Таким чином, система набору та підрахунку очок у грі не тільки робить ігровий процес захопливим, а й розвиває у гравця навички стратегічного планування та оптимізації.

2.5 Висновки до розділу 2

В результаті аналізу та проєктування гри була розроблена детальна архітектура та компоненти, необхідні для її реалізації. Були визначені взаємозв’язки між цими компонентами та способи їх взаємодії.

Проєктування гри допомагає забезпечити ефективне та структуроване виконання розробки, зменшує ризики та забезпечує високу якість кінцевого продукту. Цей процес визначає основні принципи, архітектурні рішення та структуру, які використовуються для створення гри.

Результатом проєктування гри є докладний план, що включає опис компонентів гри, їх функціональність та взаємозв’язки, а також детальні технічні специфікації. Цей план є основою для подальшої розробки та впровадження гри з використанням відповідних технологій та інструментів.

У цьому розділі представлені результати проєктування гри, які будуть використані для подальшої розробки та реалізації. Вони включають діаграми архітектури, опис компонентів, взаємодію між ними та інші важливі аспекти, що допоможуть зрозуміти та втілити задуману гру.

3 РЕАЛІЗАЦІЯ ТА ТЕСУВАННЯ

3.1 Створення сцени

Під час створення проєкту, Unity дозволяє обрати шаблон попередньо встановлених пакетів. Обираємо «2D Mobile» так, як нашою метою є створення мобільної 2D гри, це продемонстровано на рисунку 3.1.

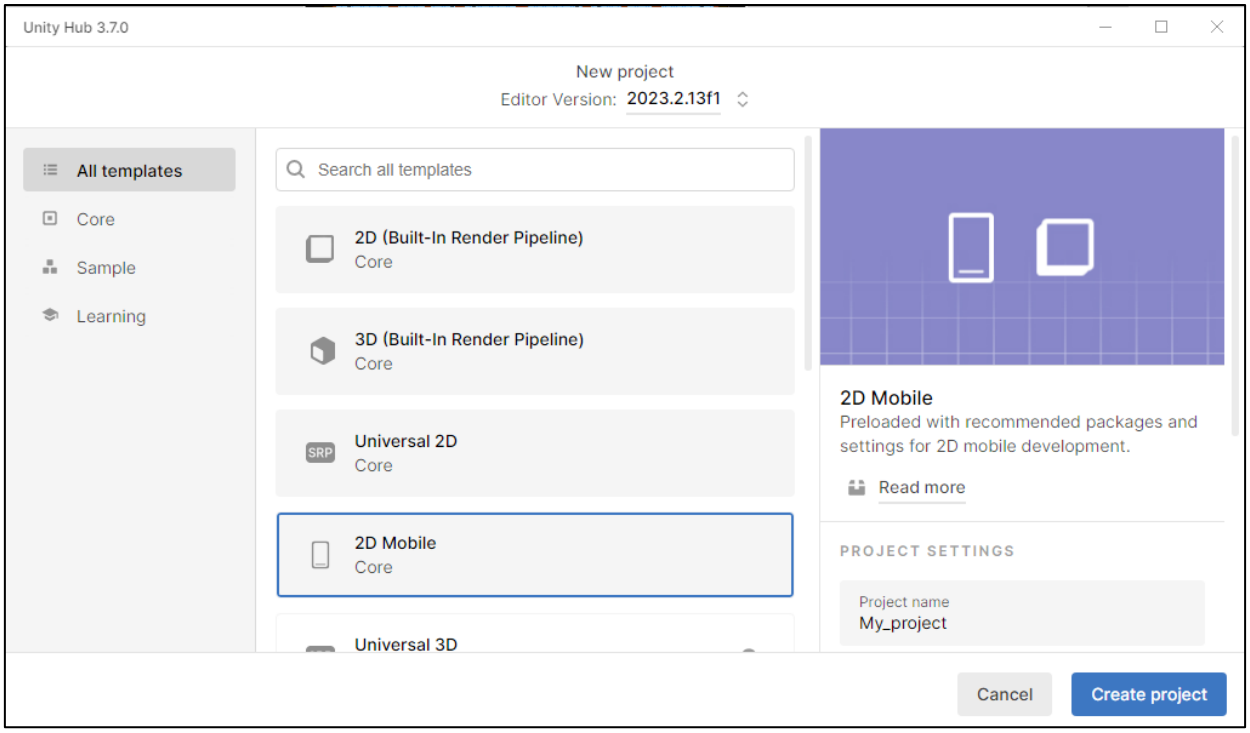

Рисунок 3.1 – Шаблон проєкту

Першим об’єктом, що існує після створення сцени в редакторі Unity, є об’єкт “Main Camera”.

Відкривши його в інспекторі, координати об’єкта по осі X та осі Y встановлені в 0, 0. Це буде центром екрана, показано на рисунку 3.2.

Для зручності розробки даного проєкту було бажано, щоб одній одиниці в системі вимірювання UnityEngine відповідав один піксель на екрані. Цьому відповідає вибраний масштаб «1» у полі Scale по всіх осях.

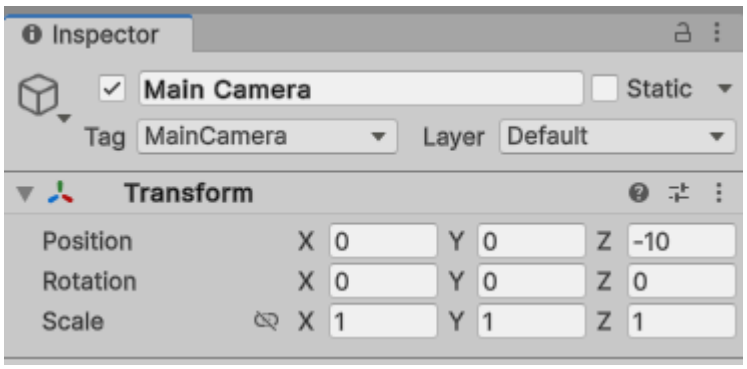

Рисунок 3.2 – Налаштування камери

Надалі всі рішення за розмірами та координатами об’єктів ґрунтуються на цьому виборі.

3.1.1 Налаштування камери, вибір роздільної здатності

Під час розробки гри на смартфон необхідно врахувати системні можливості цільових пристроїв та їх особливості. В даний час у більшості смартфонів під керуванням операційної системи Android співвідношення екрану варіюється від 16:9 до 20:9. Необхідно розглянути найпопулярніші та зупинитися на одному з них відповідно до рисунку 3.3 [5].

Тож оберемо найпопулярнішу роздільну здатність «1080*2400». Необхідно відзначити, що при використанні відмінної від передбаченої роздільної здатності в рушії Unity буде задіяний механізм масштабування при рендрінгу зображення, що цілком допустимо.

Оскільки ігровий світ є 2D ігровим полем, то в налаштуваннях камери режим проекції обрано “Orthographic”, що забезпечує відсутність перспективного спотворення та фіксований масштаб об’єктів на екрані. Розмір екрана встановлюється у параметрі Size, який у редакторі Unity визначається як половина висоти екрана в одиницях координат. Для роздільної здатності 1080 x 2400 необхідно було встановити цей параметр 2400/2, тобто 1200 одиниць, так як обраний масштаб 1 одиниця на 1 піксель екрану.

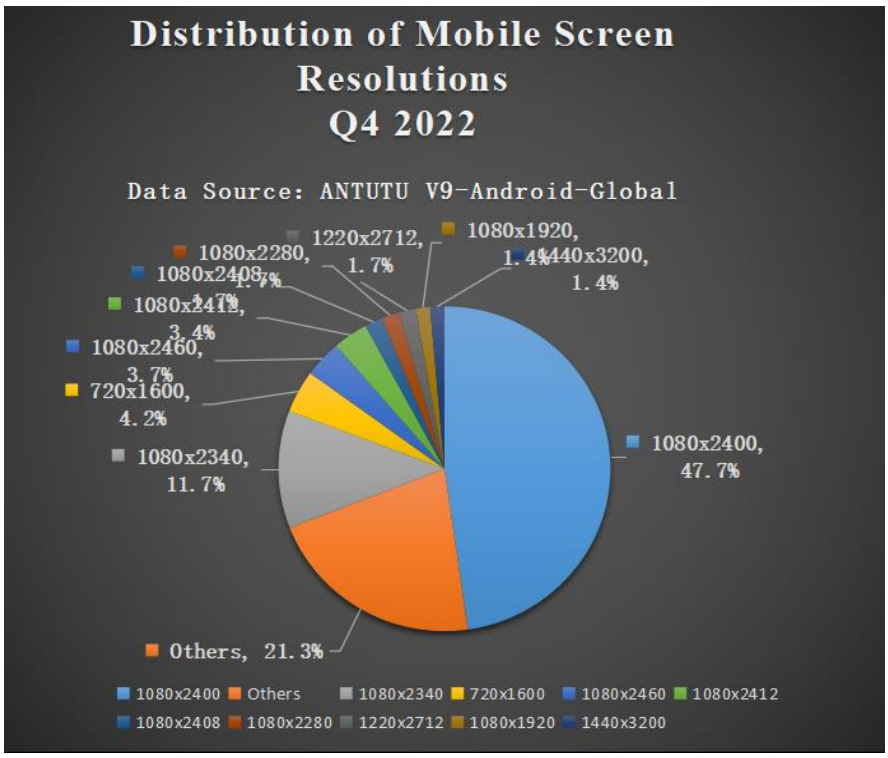

Рисунок 3.3 – Статистика роздільної здатності

Параметр в Clipping Planes встановлена: Far – 1000 одиниць, Near – 0.3. Це вказує камері не малювати об’єкти, що знаходяться за цими межами, наведено на рисунку 3.4.

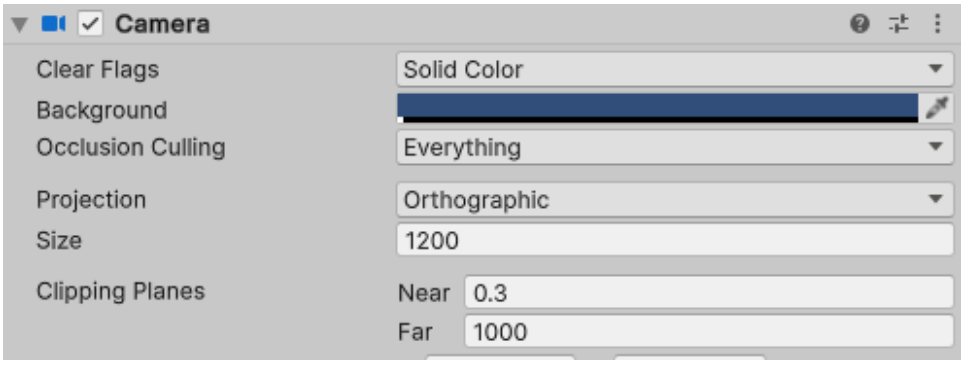

Рисунок 3.4 – Параметри малювання об’єктів

3.1.2 Ігрові об’єкти

Метод створення ігрового світу було обрано через інтерфейсні елементи, зокрема, з використанням полотна (Canvas) і розташування на

ньому елементів. Це зумовлено специфікою ігрового процесу та вимогами до візуального оформлення. За такого підходу ми отримали не тільки зручний засіб керування та розташування об’єктів, а й змогли забезпечити естетично привабливий дизайн, що поєднує в собі функціональність.

Оскільки обрано механізм ігрової взаємодії через UI-елементи, насамперед на сцені створено об’єкт «Background», що містить компонент Canvas. Цей компонент являє собою полотно заданого розміру, на якому можна розмістити інші UI-елементи і вибрати, яка камера буде відображати. Розмір цього елемента було встановлено 1080 x 2400. Render Mode обрано в координатах простору світу (World Space), що не прив’язує до координат камери, а дає змогу вільно переміщати Canvas відносно неї, за допомогою власних координат.

Координати об’єкта встановлені в 0, 0 по осях X і Y, що цілком обґрунтовано, оскільки дочірні об’єкти на Canvas матимуть свої координати відносно цих, а в цьому варіанті ще й відносно центру екрана. Координата Z за ортогональної проекції в нашому випадку визначає відстань від об’єкта до камери, і вона дорівнює 20, щоб цей об’єкт був трохи далі від камери, ніж інші, у яких координата Z буде меншою.

Поле Камера подій (Event Camera) визначає, через яку камеру гравець взаємодіє з UI-елементами полотна, а оскільки камера на сцені цього проєкту тільки одна, то її й обрано. Налаштування полотна наведено на рисунку 3.5.

Всередині об’єкта “Background” через дерево ієрархії об’єктів створено дочірні об’єкти.

Об’єкт «Back», що містить спрайт заднього фону, який заповнює весь простір на екрані позаду інших об’єктів і створює приємний візерунок. Розмір об’єкта фону обрано 1920 x 2560, що значно більше за цільову роздільну здатність екрана смартфона, зважаючи на міркування, що під час можливого запуску на інших пристроях із ширшим екраном не буде прогалин у заповненні всього простору екрана візерунком.

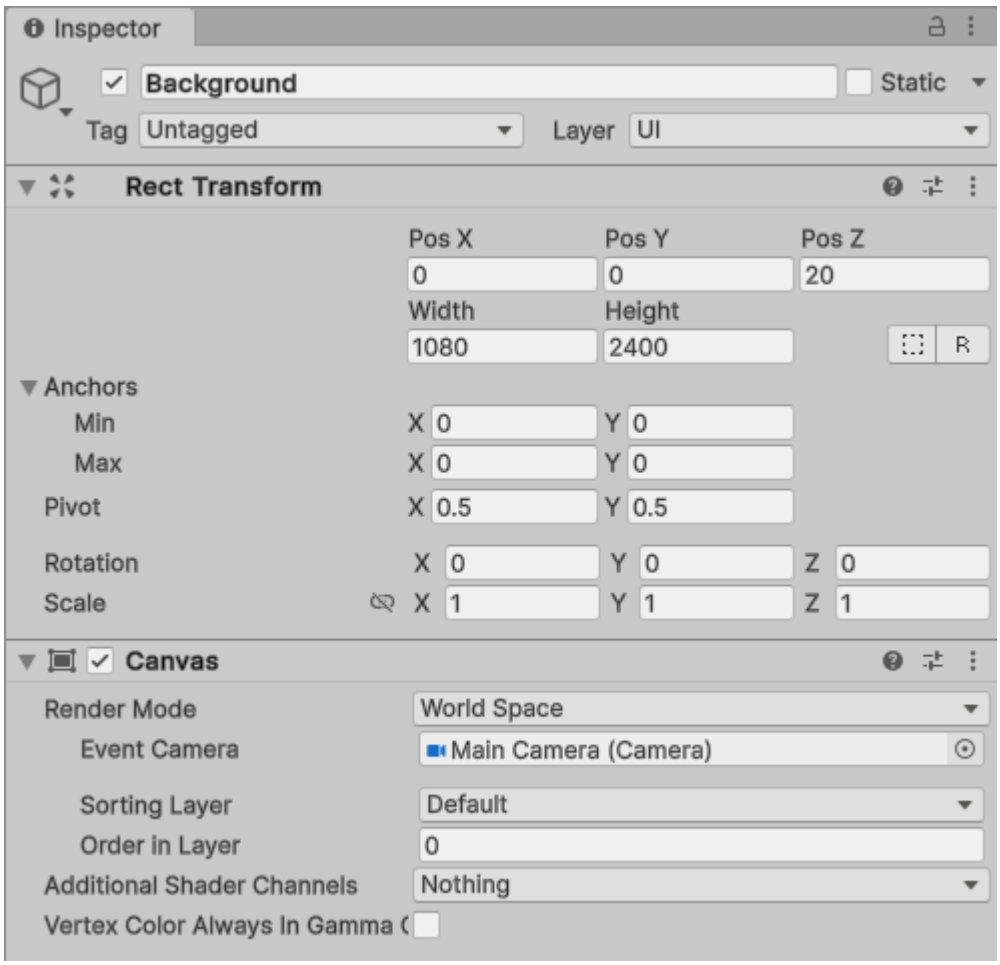

Рисунок 3.5 – Налаштування полотна

Два рівнозначних об’єкти «Balanse» і «Seeds», що містять відображувану на екрані інформацію про зароблені під час продажу ігрові очки і насіння. Ці об’єкти містять спрайти заднього фону, «BackgroundSprite», а також зображення валюти і текстові елементи з кількістю цієї валюти. Розташовуються вони внизу екрана під основним ігровим полем.

Чотири рівнозначних об’єкти з UI-компонентом Button: «ButtonGrowLayer», «ButtonSellLayer», «ButtonShopLayer», «ButtonMenuLayer». Об’єкт, що містить Button, виконує функцію кнопки – він має зображення кнопки та метод OnClick(), який викликається оброблювачем подій під час натискання гравцем на кнопку на екрані. Також Button може мати напис у вигляді дочірнього UI-елемента «Text».

Розмір об’єктів Button обраний наступний: ширина дорівнює 250

одиниць, висота 155. Вибір ширини обумовлений тим, що при ширині екрана 1080 пікселів для розташування чотирьох елементів поряд, на кожний припадає чверть ширини, яка становить 270 пікселів. Залишивши ліворуч та праворуч по 10 пікселів для відступу, залишається 250 на елемент. Виходячи з ширини 250 пікселів і застосувавши співвідношення пропорції золотого перерізу 1,618 приходимо до висоті 155 пікселя. При такому підході координати X у центрів об’єктів будуть -405, -135, 135, 405 відповідно щодо центру екрану. Координата Y встановлена в 1066, що розміщує об’єкти над основним ігровим полем.

На сцені створено чотири рівнозначні та взаємозамінні об’єкти “Grow”, “Sell”, “Shop”, “Menu”. Ці об’єкти перебувають на місці основного ігрового поля і мають у собі дочірній об’єкт “Canvas” розміром 1080 x 1920 одиниць. Розташовуються в центрі екрану за координатами X, Y, Z рівними 0, 0, 0 відповідно. Вибір обумовлений таким міркуванням, що ширина не повинна перевищувати ширину цільової роздільної здатності екрана, а висота відповідає звичному для очей співвідношення 16:9. Основна взаємодія гравця з ігровим світом запланована за допомогою елементів цих об’єктів. Кожне полотно цих чотирьох об’єктів має власне стилізоване зображення заднього фону, на якому розташовуються UI-елементи, що підтримують взаємодію, що містять компонент Button. Керування елементами здійснюється в C# скрипті MainScript.cs, який має доступ до цих елементів.

Розглянемо дочірні елементи об’єкта «Grow».

Об’єкт “BackgroundGrow”, що містить неінтерактивні об’єкти зображень заднього фону, що складається з декількох шарів зображень: “Layer1Image_0”, “Layer1Image_1”, “Layer1Image_2” – містять стилізовані піксельні зображення полиць для вирощування рослин, “Layer2Image”, “Layer3Image” – зображення неба за вікном. Об’єкти зображення полиць розташовуються на трьох горизонтальних рівнях і в залежності від ігрового прогресу деякі можуть бути неактивними. Координата X у центрів всіх об’єктів дорівнює 0 так, як всі об’єкти вирівняні по центру екрану,

координата Y центрів об’єктів “Layer2Image” та “Layer3Image” так само дорівнює 0, а у вершин об’єктів “Layer1Image_2”, “Layer1Image_0”, “Layer1Image_1”, дорівнює 195, -255, -705, відповідно. Координати зображень полиць для вирощування спеціально підібрані так, щоб можна було розташовувати зображення рослин, що вирощуються між ними і не перекривати їх (висота зображення квітки 400 пікселів, 50 пікселів залишено на висоту полиці і відстань до неї, виходить необхідна відстані між координатами полиць 450 одиниць). Координата -705 обрана з таким розрахунком, що до низу ігрового поля залишається 255 одиниць, що достатньо для розміщення там необхідних об’єктів (705+255=960=1920/2). Масштаб зображення кожної полиці по осях X і Y обраний рівним 4, щоб пікселі виглядали більшими і доводилося 4 одиниці довжини на 1 піксель зображення. З урахуванням масштабу та положення на екрані розмір зображення полиці підібрано 250 x 135, 250 x 177, 250 x 62 відповідно. У полі “Pivot” значення сторони прив’язки цих об’єктів по осі Y змінено з «0,5» на «1», по осі X залишено «0,5», що означає перенесення прив’язки до координат з центру об’єкта на середину його верхньої межі. Налаштування заднього фону наведено на рисунку 3.6.

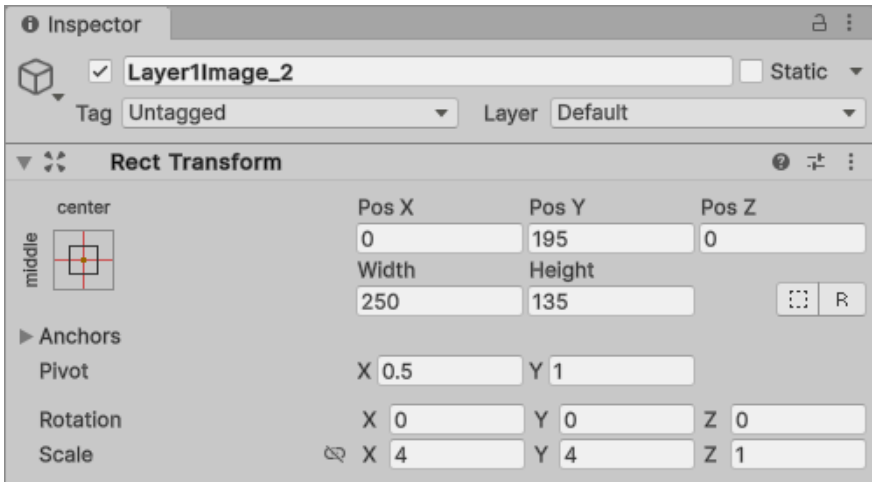

Рисунок 3.6 – Налаштування заднього фону

Рівнозначні об’єкти «GrowShelf1», «GrowShelf2», ... «GrowShelf9», які містять об’єкти вирощуваних квіток. Ці об’єкти різняться тільки

координатами, дочірні об’єкти повністю ідентичні.

Координати підібрані таким чином, що відстань між об’єктами дорівнює 450 по вертикалі і 350 по горизонталі. 450 по висоті відповідає розміру зображення квітки 400 пікселів і 50 пікселів припадає на висоту полиці та відстань до неї. 350 по горизонталі відповідає ширині зображення квітки 300 пікселів і 50 пікселів на відстань між ними.

Об’єкти «ButtonPlant», «ButtonBox», що виконують функції кнопки, яку натискають, і мають дочірній об’єкт «Image», що містить стилізоване піксельне зображення. Самі ці кнопки після натискання створюють ігрову подію.

Об’єкт «ButtonBack» виконує функції кнопки, та має текстовий напис «Back» над зображенням кнопки. Ця кнопка виконує функції скасування в певних ігрових подіях. За замовчуванням об’єкт перебуває в неактивному, невидимому стані й активується під час початку певних ігрових подій. Розмір об’єкта з естетичних міркувань установлено в 250 одиниць за шириною і 155 одиниць за висотою, щоб відповідати кнопкам у верхній частині екрана. Координати розташування обрано так, щоб розташовуватися праворуч унизу основного ігрового поля, і вони дорівнюють (0, 0, 0) із прив’язкою до нижнього правого кута батьківського об’єкта (Canvas) своїм нижнім правим кутом, що відповідає значенням «1» за X та «0» за Y у полі «Anchors» і значенням «1» за X та «0» за Y у полі «Pivot» відповідно.

3.1.3 Неігрові об’єкти

У редакторі Unity використання C#-скриптів можливе тільки шляхом прикріплення їх до об’єктів сцени. На сцені створено порожній об’єкт «GameManager», до якого можна буде прикріпити головний скрипт, а також ресурси, що використовуються надалі в грі та не були прикріплені до ігрових об’єктів, зокрема аудіо кліпи.

Обробник подій, викликаних діями користувача, «EventSystem», автоматично створюється на сцені під час додавання об’єктів, що містять UI, як canvas, button тощо. Оскільки цей об’єкт присутній номінально, його налаштування не були змінені.

Об’єкт «SwipeManager» додано на сцену для подальшого прикріплення до нього колайдера Box Collider 2D і C#-скрипта, що розпізнає та обробляє свайпи по екрану. Розмір колайдера встановлено 1920 x 2560, що значно відрізняється від цільової роздільної здатності екрана в більший бік. Цей вибір зроблено для того, щоб під час запуску на пристроях з набагато більшою роздільною здатністю, ніж у смартфона, наприклад на планшеті, свайпи розпізнавалися на більшій площі екрана.

3.2 Оформлення елементів гри

Під час розробки візуального стилю та оформлення елементів гри було приділено особливу увагу вибору зображень і спрайтів, які не тільки ефективно передають концепцію гри, а й забезпечують приємний візуальний досвід для гравців.

Для створення спрайтів та зображень у проєкті використовувався графічний редактор Adobe Photoshop. З його допомогою були розроблені всі графічні елементи гри, включаючи фони, персонажів, об’єкти оточення та інтерфейсні елементи. Photoshop дозволив ретельно опрацьовувати кожен елемент, забезпечуючи високу якість та естетичну відповідність загальному стилю гри. Різноманітні інструменти та функції програми дозволили досягти потрібної деталізації та виразності графічних елементів, а також оптимізувати зображення для ефективного завантаження ресурсів і підвищення продуктивності гри.

Для додавання спрайтів до об’єктів ігрового світу в проєкті було виконано наступні дії.

Створення папки «Sprites». У папці «Assets» було створено підпапку з назвою «Sprites», призначену для зберігання всіх зображень і атласів спрайтів.

Імпорт зображень. Зображення та атласи спрайтів були імпортовані в створену папку «Sprites» шляхом перетягування файлів із файлового менеджера у відкриту папку в редакторі Unity.

Налаштування зображень для коректного відображення в ігровому світі, шляхом застосування до них таких налаштувань, як зміна режиму фільтрації спрайта, щоб зберегти чіткість пікселів на кордонах зображення. Для цього було застосовано режим фільтрації «Point» (точковий), який унеможливлює розмиття пікселів під час масштабування зображення. Це дало змогу зберегти піксельну стилістику гри та забезпечити чітке відображення всіх елементів в ігровому світі.

Під час вилучення спрайтів з атласу в Unity вручну було вказано межі спрайта за допомогою Sprite Editor, щоб точно дотриматися розмірів зображень. Цей процес давав змогу точніше відтворити початкові розміри спрайтів з атласу, відповідаючи оригінальному дизайну гри.

Для відображення спрайтів і текстур на об’єктах у сцені використовується компонент Image. Він надає інтерфейс для встановлення спрайтів, кольорів та інших параметрів візуалізації. Поле ‘Source Image’ у компоненті Image визначає, яке зображення буде відображатися цим компонентом. Тут обирається зображення або спрайт, який буде відображатися.

У разі використання компонента Image для відображення зображень на інтерфейсі, в Unity автоматично додається до об’єкта компонент Canvas Renderer. Canvas Renderer займається рендерингом (відображенням) елементів інтерфейсу на полотні (Canvas). Він визначає порядок відображення елементів, їхню прозорість, а також інші атрибути, пов’язані з відображенням. У всіх об’єктів у проєкті цей компонент не налаштовувався явно, оскільки Unity автоматично керував цим компонентом, забезпечуючи

коректне відображення зображень на інтерфейсі.

Після додавання зображення до елемента на екрані, проконтрольовано розміри та масштаб зображення, щоб гарантувати його відповідність дизайну інтерфейсу та задуму проєкту.

Надалі, для динамічного керування зображеннями під час гри можна отримати доступ до компонента Image зі скриптів. Маючи в коді посилання на об’єкти, що містять компонент Image, за допомогою методу GetComponent<Image>() можна отримати до нього доступ і змінити його властивості. Наприклад, через властивість sprite можна змінити зображення, що відображається на об’єкті. Також можна змінити колір зображення, використовуючи властивість color. Ці можливості дають змогу динамічно змінювати зовнішній вигляд об’єктів в ігровому світі, підлаштовуючись під різні ігрові ситуації.

3.3 Реалізація взаємодій з ігровими об’єктами

Під час створення гри було обрано підхід, за якого всі дії та події обробляються в єдиному місці через один головний клас. Це забезпечує простоту і зрозумілість коду, централізоване управління ігровим процесом і ефективність розробки. Оскільки не планується додавання нових функцій і елементів у майбутньому, було вирішено відмовитися від модульного підходу. У простих іграх, де немає потреби в складній архітектурі або високій оптимізації, одноцільовий клас може бути більш ефективним рішенням.

У редакторі Unity у вікні проєкту створено папку Scripts, у ній створено файл C#-скрипта з назвою «MainScript.cs» [6].

В Unity скрипти є частиною ігрових об’єктів і визначають їхню поведінку. Коли скрипт прикріплюється до об’єкта, він отримує доступ до компонентів цього об’єкта і може керувати ними. Це необхідно, тому що без

прив’язки до об’єкта скрипт не має контексту виконання і не може взаємодіяти з іншими елементами гри. Прикріплення скрипта до об’єкта створює зв’язок між кодом і конкретним ігровим об’єктом, що дає змогу скрипту взаємодіяти з ігровим світом і реагувати на дії гравця або інших об’єктів.

Виходячи з цього, на сцені було створено порожній об’єкт без компонентів, який названо «GameManager», потім до якого було прикріплено скрипт MainScript.cs.

Відкривши скрипт у редакторі Visual Studio Code на рисунку 3.7 видно, що він уже містить клас, який успадковується від MonoBehaviour. Це базовий клас для всіх скриптів в Unity, який надає доступ до життєвого циклу об’єкта і різних функціональностей, таких як управління поведінкою, обробка вхідних даних і взаємодія з іншими об’єктами. Успадкування від MonoBehaviour дає змогу скрипту взаємодіяти з Unity Engine і реалізовувати необхідну логіку для роботи з об’єктом в ігровому світі. Це важливо для створення інтерактивних і функціональних ігрових елементів.

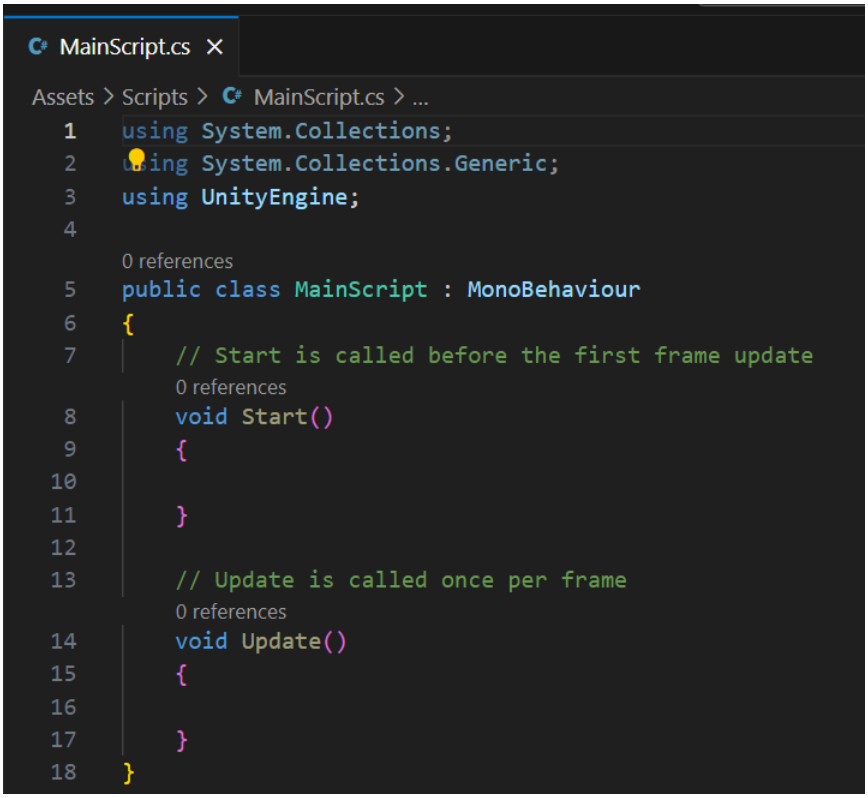

Рисунок 3.7 – Файл MainScript.cs

У межах цього класу виконано основний код проєкту гри. Цей клас, успадкований від MonoBehaviour, містить методи і змінні, які визначають поведінку об’єкта в ігровому світі. Тут реалізується логіка гри, управління взаємодією з іншими об’єктами, обробка користувацького введення та інші важливі аспекти ігрового процесу. Весь код, написаний усередині цього класу, є основою для функціонування гри та визначає її ігровий досвід.

Після прикріплення скрипта до об’єкта, методи, визначені в цьому скрипті, стають доступними для виклику з методів onClick кнопок на сцені. Це відкриває можливість обробки дій гравця, таких як натискання на кнопки інтерфейсу. Під час натискання на кнопки ігрового інтерфейсу викликаються відповідні методи скрипта, які реагують на дії гравця та взаємодіють з іншими елементами гри, забезпечуючи інтерактивність і керування ігровим процесом.

Для того щоб методи скрипта були доступні ззовні, вони мають бути оголошені з модифікатором доступу public. Таким чином, методи, призначені для опрацювання дій гравця, також мають бути відкритими для виклику ззовні. Оголошення методів із модифікатором public забезпечує доступ до них із кнопок інтерфейсу та інших компонентів гри, що дає змогу коректно реагувати на дії гравця та керувати ігровим процесом.

Метод стає доступним для вибору як обробник події для кнопок в інтерфейсі редактора Unity. Приклад методу наведений на рисунку 3.8.

```
public static void PressedExit()
{
    // Завершаем приложение
    Application.Quit();
}
```

Рисунок 3.8 – Метод, що викликається при натисканні на кнопку ButtonExit

Далі потрібно в панелі інспектора в об’єкта ButtonExit у методі OnClick() додано дію, в якій обрано об’єкт GameManager, обрано скрипт MainScript, обрано метод PressedExit(), це наведено на рисунку 3.9.

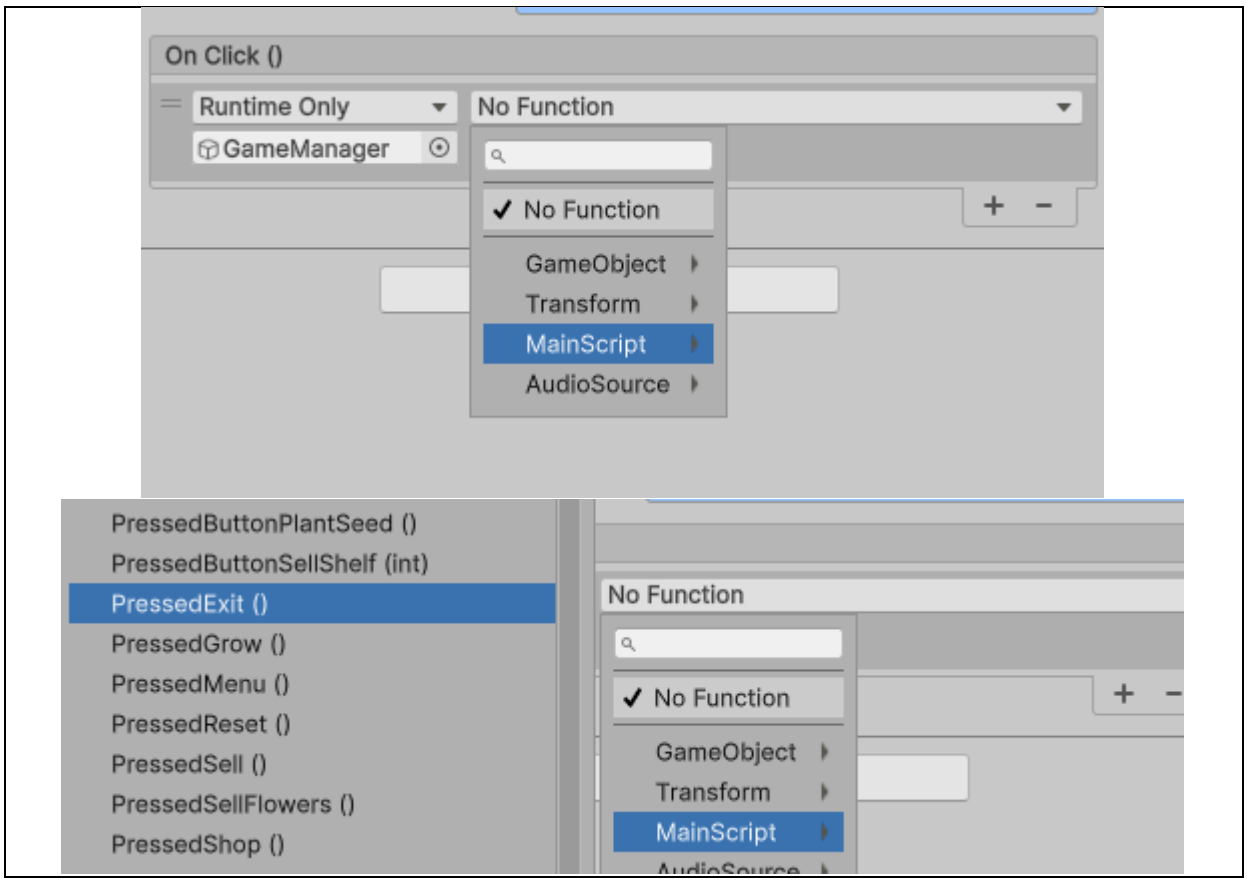

Рисунок 3.9 – Встановлення обробника події OnClick

Використання методів обробників натискань на кнопки є основним механізмом взаємодії гравця з ігровим світом у грі.

Цей підхід було обрано з огляду на його простоту й ефективність в управлінні взаємодією гравця з ігровим світом, а також завдяки можливості використання стандартних компонентів Unity для опрацювання подій під час натискання на кнопки, що скорочує час розробки та спрощує процес створення інтерактивних елементів інтерфейсу для користувача, що важливо для забезпечення зручності та привабливості ігрового процесу; такий підхід дає змогу також легко налаштовувати поведінку кнопок безпосередньо в редакторі Unity, що спрощує роботу з інтерфейсом і сприяє підвищенню ефективності роботи з ним.

Для реалізації функціональності перетягування об’єктів по екрану було створено окремий скрипт під назвою DragAndSwap.cs [6]. Цей скрипт прикріплюється до об’єктів, які мають бути перетягуваними, відповідно до

вимог Unity Engine. DragAndSwap обробляє такі дії, як перетягування квіток, а також здійснює обмін місцями квіток у разі, якщо одна квітка переміщується на місце іншої. Такий підхід дає змогу реалізувати зручну та інтуїтивно зрозумілу взаємодію користувача з об’єктами на екрані.

Взаємодія з об’єктом, який передбачається перетягувати по екрану пальцем, здійснюється з використанням у об’єкта компонента BoxCollider2D. Він визначає прямокутну область навколо об’єкта, яка реагує на взаємодію з іншими об’єктами, включно з дотиком пальцем на екрані пристрою. Таким чином, BoxCollider2D використано для визначення зони, у якій користувач може почати перетягування об’єкта.

У скрипті, прикріпленому до перетягуваного об’єкта, реалізовано деякі функції. Розберемо їх детальніше.

Обробка початку перетягування: коли користувач починає перетягувати об’єкт, скрипт запам’ятовує початкові координати об’єкта й ініціює процес перетягування.

Обробка переміщення об’єкта: під час перетягування скрипт оновлює координати об’єкта відповідно до поточного положення пальця користувача на екрані.

Обробка завершення перетягування: коли користувач відпускає об’єкт, скрипт визначає, чи перебуває об’єкт у зоні, призначеній для перетягування. У разі успішного перетягування об’єкта на потрібне місце, скрипт виконує необхідні дії, тобто обмінює об’єкти місцями.

Обробка помилкового перетягування: якщо об’єкт не було переміщено на правильне місце, скрипт повертає його назад, щоб запобігти помилковим діям користувача.

Таким чином, скрипт повинен забезпечує коректне та інтуїтивно зрозуміле перетягування об’єкта по екрану, враховуючи різні сценарії взаємодії користувача з об’єктом.

Оскільки на екрані присутні кілька об’єктів, призначених для перетягування, до кожного з яких було прикріплено цей скрипт.

Для зміни стану об’єктів на екрані зі скрипта, прикріпленого до перетягуваних об’єктів, викликаються методи основного скрипта.

Виходячи з того, що є два ігрових поля, на яких є об’єкти, що перетягуються, поля Grow і Sell, було вирішено створити два скрипти, що відповідають кожен за своє поле, скрипти DragAndSwapGrow і DragAndSwapSell. Принцип реалізації ідентичний, однак для врахування відмінності у внутрішніх даних цих полів, методи, що викликаються в основному скрипті для поновлення об’єктів на екрані, у кожному свої.

Щоб додати трохи динаміки та інтерактивності в гру, були використані анімації, такі як рух.

В Unity анімації реалізуються за допомогою компонента Animator. Цей компонент дає змогу створювати та керувати анімаціями об’єктів просто в редакторі Unity. Можна створювати ключові кадри, визначати переходи між анімаціями, налаштовувати параметри анімацій і багато чого іншого, все це без необхідності писати складний код.

Крім того, в Unity також доступні механізми керування анімаціями через код, що дає змогу динамічно керувати ними залежно від умов ігрового процесу.

У цій грі використовувалися анімації для створення ефекту зміни ігрового поля. Під час вибору нового поля старе переміщується вбік за екран, а нове переміщується з іншого боку на його місце. Цей ефект досягається шляхом запуску відповідних анімацій для переміщення об’єктів ігрового поля з використанням компонента Animation.

Для створення анімацій використовувалося вікно Animation в Unity. Це вікно дає змогу записувати рух та інші зміни стану об’єктів прямо в редакторі Unity. Однак, управління анімаціями в грі здійснюється через код. Це означає, що анімації, створені за допомогою вікна Animation, активуються зі скрипта, що дало змогу створити динамічний ефект зміни ігрового поля під час вибору нового.

Для програвання анімацій на ігрових об’єктах в Unity призначений

компонент Animation. Цей компонент було прикріплено до об’єктів, які мають програвати анімацію, створену у вікні Animation.

Для додавання анімації, створеної у вікні Animation, до ігрового об’єкта виконано такі кроки:

– прикріплення компонента Animation до ігрового об’єкта, який необхідно анімувати;
– прив’язка створеної анімації до компонента Animation, яка має бути програна на цьому об’єкті;
– керування анімацією через код за допомогою скрипта, який активує анімації, прикріплені до певних об’єктів.

На рисунку 3.10 наведено приклад коду з використанням анімації після натискання кнопки Grow при активному ігровому полі Sell.

```
GrowObject.GetComponent<Animation>().Play("FromLeft");
SellObject.GetComponent<Animation>().Play("ToRight");
```

Рисунок 3.10

Ці дві анімації починають програватися одночасно, один об’єкт переміщується з центру праворуч за екран, і водночас на його місце в центр з іншого боку переміщується інший.

Варто зазначити, що в цьому проєкті відсутній явний ігровий цикл (Game Loop), який традиційно використовується для оновлення стану гри та відтворення кадрів. Це зроблено з кількох причин. По-перше, ігровий рушій, що використовується, автоматично керує цим процесом, забезпечуючи стабільну роботу гри й оптимальне використання ресурсів. По-друге, для спрощення розробки і скорочення часу розробки, використані функції і події, обробка яких відбувається в міру необхідності, замість традиційного ігрового циклу. Цей підхід дає змогу гнучкіше керувати поведінкою гри та покращує її масштабованість і підтримуваність.

3.4 Система набору та підрахунку ігрових очок

Для підрахунку ігрових очок у головному керуючому скрипті реалізовано та використовуються два основні методи: CalculateAllBonus() і CalculateTotalSellValue().

Метод CalculateAllBonus() призначений для перевірки всіх комірок, на яких можна розмістити квітку для продажу. Він аналізує сусідні комірки кожної квітки на предмет можливого збільшення її вартості від впливу сусідніх квіток. Отримані бонуси додаються до базової вартості кожної квітки, а зміни одразу відображаються поруч із базовою вартістю кожної квітки на екрані. Цей метод викликається після кожної зміни розташування квіток, щоб забезпечити актуальний підрахунок бонусів.

Метод CalculateTotalSellValue() відповідає за підрахунок підсумкової суми за можливий продаж усіх квіток на ігровому полі. Після кожної зміни розташування квіток він перераховує цю суму і оновлює її на екрані, забезпечуючи гравцеві актуальну інформацію про потенційні доходи від продажу квітів.

Завдяки цьому, гравець може стратегічно планувати розміщення квітів на ігровому полі, максимізуючи свій набір ігрових очок і загальний дохід від продажу квітів.

3.5 Механізм збереження ігрового прогресу

Для забезпечення збереження та завантаження прогресу гравця в грі було розроблено спеціальний клас SaveManager і структуру GameData. Ці компоненти працюють разом для збереження поточного стану ігрового світу і відновлення його під час наступних запусків гри.

Структура GameData була створена для зберігання всіх необхідних даних, що відображають поточний стан ігрового світу.

Дана структура містить такі поля:

– Money, Seeds1, Seeds2, Seeds3, Seeds4, Seeds5, SellShelfPrice: значення, що представляють кількість набраних очок, кількість придбаного та зароблено насіння, поточну вартість полиці в магазині;
– ListDateTimeString: список терміну росту рослин, які ще ростуть;
– ListGrowFlowers1, ListGrowFlowers2, ... ListGrowFlowers9: списки даних за рослинами, які перебувають на полі Grow;
– ListSellFlowers1, ListSellFlowers2, ... ListSellFlowers9: списки даних за рослинами, що знаходяться на полі Sell.

Структура GameData наведена на рисунку 3.11.

```
public struct GameData
{
    public int Money, Seeds1, Seeds2, Seeds3, Seeds4, Seeds5, SellShelfPrice;
    public int[] ArrayGrowShelfs, ArraySellShelfs, ArrayPlantFlowers, ArrayBuyedItems;
    public List<System.String> ListDateTimeString;
    public List<int> ListGrowFlowers1, ListGrowFlowers2, ListGrowFlowers3, ListGrowFlowers4,
ListGrowFlowers5, ListGrowFlowers6, ListGrowFlowers7, ListGrowFlowers8, ListGrowFlowers9;
    public List<int> ListSellFlowers1, ListSellFlowers2, ListSellFlowers3, ListSellFlowers4, ListSellFlowers5,
ListSellFlowers6, ListSellFlowers7, ListSellFlowers8, ListSellFlowers9;
}
```

Рисунок 3.11 – Приклад коду структури GameData

Клас SaveManager було розроблено для керування процесами збереження та завантаження даних ігрового прогресу. Він містить методи, які забезпечують перетворення даних структури GameData у рядок формату JSON та їхнє збереження з використанням класу PlayerPrefs, а також методи для завантаження даних з PlayerPrefs і їхнього перетворення назад у структуру GameData.

Основні методи класу SaveManager. SaveGameData – метод для збереження поточного стану гри. LoadGameData – метод для завантаження збереженого стану гри. ClearGameData – метод для видалення всіх збережених даних, він наведений на рисунку 3.12.

```
public class SaveManager
{
    private const string dataKey = "GameData";

    public void ClearGameData()
    {
        PlayerPrefs.DeleteAll();
    }
    public void SaveGameData(GameData _gameData)
    {
        string json = JsonUtility.ToJson(_gameData);
        PlayerPrefs.SetString(dataKey, json);
        PlayerPrefs.Save();
    }
    public GameData LoadGameData()
    {
        string json = PlayerPrefs.GetString(dataKey, "");
        if (!string.IsNullOrEmpty(json))
        {
            return JsonUtility.FromJson<GameData>(json);
        }
        else
        {
            return new GameData();
        }
    }
}
```

Рисунок 3.12 – Метод ClearGameData

Для переведення даних стану ігрового світу у формат структури GameData для подальшого збереження в головному керуючому класі MainScript створено приватний метод saveAllData(). Для відновлення даних зі структури GameData назад в ігровий світ створено метод loadAllData().

Коли гравець досягає певної точки збереження (наприклад, згортає додаток або виходить з гри), викликається метод saveAllData класу MainScript. Цей метод збирає поточні дані з ігрового світу, заповнює структуру GameData, і потім одразу викликається метод SaveGameData класу SaveManager, який приймає структуру GameData, перетворює її на рядок формату JSON і зберігає цей рядок із використанням PlayerPrefs.

Під час запуску гри викликається метод LoadGameData класу SaveManager. Цей метод витягує рядок JSON з PlayerPrefs і десеріалізує його

назад у структуру GameData, потім відразу викликається метод loadAllData класу MainScript, відновлюючи стан ігрового світу.

Таким чином, використання класу SaveManager і структури GameData дає змогу ефективно зберігати і завантажувати прогрес гравця, забезпечуючи цілісність даних і безперервність ігрового процесу.

3.6 Ініціалізація та завантаження гри

Під час запуску гри в головному керуючому класі MainScript викликається метод Start(), у якому послідовно виконуються кроки підготовки ігрового світу до стану готовності. Ці дії включають ініціалізацію змінних, завантаження ресурсів і відновлення стану гри зі збережених даних.

Насамперед викликається метод Initialise(), який виконує ініціалізацію всіх необхідних змінних і компонентів, що відображають поточний стан ігрового світу. Цей метод відповідає за налаштування початкових параметрів гри, забезпечуючи коректне функціонування всіх ігрових елементів. Зокрема, відбувається створення списків спрайтів рослин, списків часу росту рослин, відбувається вибір спрайта неба на задньому фоні залежно від реального часу доби, створюються посилання на об’єкти ігрового світу, до яких необхідно мати доступ у процесі гри для управління ними.

Потім відбувається завантаження збереженого стану – викликається метод LoadGameData класу SaveManager, заповнюючи структуру GameData. Перед витяганням даних ігрового світу з GameData виконується метод resetDataToDefaults, який приводить ігровий світ у стан за замовчуванням.

Це необхідно для коректної ініціалізації всіх змінних і об’єктів, якщо збережені дані відсутні (наприклад, гру запущено вперше). Після цього, якщо існують збережені дані в структурі GameData, вони завантажуються і відновлюються з використанням loadAllData класу MainScript. Це дає змогу гравцеві продовжити гру з того місця, де він зупинився.

Наступним кроком відбувається оновлення об’єктів ігрового світу для приведення їх у відповідність із внутрішніми змінними. Викликаються методи updateBalanse, updateSeeds, updateAllShelf, CalculateTotalSellValue, updateShop, checkGrow. Зазвичай у процесі гри вони викликаються після зміни відповідних внутрішніх змінних, а тут вони викликаються всі разом, що і приводить ігровий світ в актуальний стан.

Наприкінці методу Start() викликається ігрове поле, яке гравець має бачити після запуску, і гра готова до взаємодії з гравцем.

3.7 Оптимізація та налаштування зовнішнього вигляду

Оптимізація ігрового проєкту – важливий етап розробки, особливо коли йдеться про мобільні ігри. Мобільні пристрої, на відміну від стаціонарних комп’ютерів, мають обмежені ресурси, як процесор і графічний чип, що вимагає особливої уваги до оптимізації ігрового процесу. Розглянемо кілька ключових аспектів оптимізації, які допоможуть грі працювати більш ефективно і плавно на мобільних пристроях.

Завдяки статичній природі фону та відсутності динамічних елементів гра потребує мінімальної кількості ресурсів, що робить її більш ефективною для запуску на мобільних пристроях.

Простота рівнів і мінімальна кількість деталей сприяють швидкому завантаженню гри і плавному її виконанню на мобільних пристроях.

Статичне заднє тло не тільки додає глибини й атмосфери в гру, а й зменшує навантаження на процесор і відеокарту, оскільки не потребує оновлення або анімації в кожному кадрі.

Для зниження споживаних ресурсів відсутність викликаних методів Update() в об’єктів зменшує навантаження на процесор, оскільки об’єкти не постійно оновлюються в кожному кадрі. Відсутність методів Update() в об’єктів також означає, що здебільшого жодні дії не відбуваються, коли гра перебуває у згорнутому стані, і ігровий процес фактично зупиняється. Це

сприяє додатковій економії енергії. Відсутність явного ігрового циклу (Game Loop) також впливає на енергоефективність вашої гри, хоча в меншій мірі, ніж відсутність викликаних методів Update() у об’єктів. Коли гра не має явного ігрового циклу, вона може заморожуватися або переходити в стан пасивного очікування, коли немає активності з боку користувача або коли гра перебуває в згорнутому стані. Це означає, що процесор не витрачає ресурси на постійне оновлення ігрового світу, що сприяє додатковій економії енергії.

Ці аспекти дозволяють грі зберегти високу продуктивність і оптимізацію навіть на слабких мобільних пристроях, що робить її привабливою для широкої аудиторії користувачів.

Що стосується налаштування зовнішнього вигляду, використовувалися такі підходи:

– увімкнення режиму Pixel Perfect дає змогу уникнути розмиття спрайтів, що робить зображення чітким і зберігає візуальну цілісність піксельної графіки;
– уникнення розмиття спрайтів дає змогу зберегти чіткість зображення навіть на пристроях із низькою роздільною здатністю екрана, що важливо для збереження візуальної цілісності піксельної стилістики гри;
– підтримання правильних розмірів об’єктів зі спрайтами для уникнення нечіткого масштабування та викривлення в піксельній стилістиці, де кожна деталь має бути чітко помітною і сприйматися саме так, як задумано.

3.8 Тестування

Під час першого запуску гри або після скидання ігрових налаштувань на екрані з’являється привітальне повідомлення, яке коротко пояснює, що потрібно робити, це наведено на рисунку 3.13.

Рисунок 3.13 – Стартовий напис

На ігровому екрані вгорі розміщені 4 кнопки grow, sell, shop і menu, натискання на які перемикає ігрове поле в центрі екрана на відповідний режим вирощування, продажу, магазину або меню. Внизу екрана знаходяться два елементи інтерфейсу з інформацією про кількість наявних ігрових очок і кількість наявного насіння 2, 3, 4 і 5 рівнів.

На початку ігрового прогресу гравець уже має 20 насінин 1 рівня, які він може посадити на ігровому полі grow, що являє собою полиці для вирощування квітів. Для цього потрібно натиснути на зображення пакета з насінням і потім вибрати одне з вільних місць на полиці (див. рис. 3.14).

Після вибору вільного місця витрачається одна насінинка, і на цьому місці з’являється горщик із паростком квітки, що виросте через деякий час. Приблизний час зростання, що залишився, буде написано під горщиком і змінюватиметься залежно від часу, що залишився.

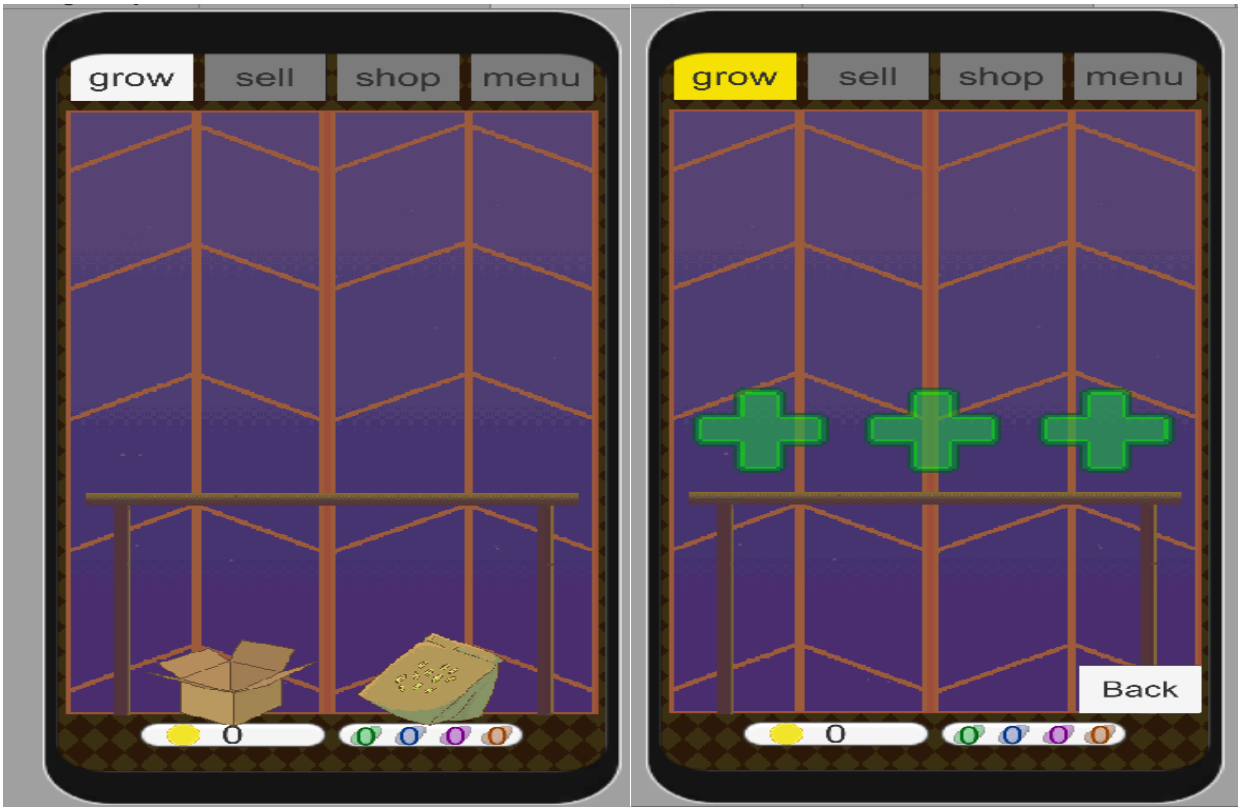

Рисунок 3.14 – Посадка квітів

Надалі, коли гравець заробить насіння починаючи з 2 рівня, після натискання на пакет із насінням і вибору вільного місця гравцеві буде надано вибір, яке з них посадити у відповідному зображенні інтерфейсу. Рівні насіння позначаються його кольором: 1 рівень – сірий, 2 рівень – зелений, 3 рівень – синій, 4 рівень – фіолетовий, 5 рівень – помаранчевий (див. рис. 3.15).

Вибір насінини здійснюється натисканням на кнопку з її зображенням і кількістю. Гравець може закрити цей елемент інтерфейсу, натиснувши на «X», якщо він передумав, або якщо в нього більше немає насіння, яке можна вибрати для посадки. Після того як рослина закінчила рости, вона набуває свого фінального вигляду і стає доступною для продажу. Зовнішній вигляд рослини визначається її вартістю та іншими характеристиками, які впливають на вартість під час продажу.

На екрані «grow» є кнопка коробки, натиснувши на яку гравець може швидко перемістити квіти, що виросли, з поля grow на поле sell, якщо там є відповідні вільні місця.

Рисунок 3.15 – Час росту квітки і рівні насіння

Зображення квітів, які вирощує гравець, різняться не тільки залежно від вартості, а й можуть відрізнятися в межах однієї вартості, залежно від інших внутрішніх характеристик квітки, вносячи різноманітність в ігровий процес.

Натиснувши на рослину, гравець може подивитися інформацію про її вартість, навколо якої розташовані інші характиристики, що впливають на вартість під час продажу. Також є кнопка «Delete», після натискання на яку квітка видалиться, звільнивши місце для посадки. Меню рослини наведено на рисунку 3.16.

Гравець може натиснути на кнопку «To sell», після чого відкриється ігрове поле sell для продажу, даючи гравцеві вибрати вільне місце. Натискання на доступне вільне місце над наявною полицею переміщує обрану квітку на це місце. Після чого гра назад повертається до ігрового поля grow.

Рисунок 3.16 – Меню квітки

Якщо відкрити ігрове поле sell, можна побачити полиці та квіти, що знаходяться на них (див. рис. 3.17). Під кожною полицею буде зображуватися вартість квітки на ній, а внизу ігрового поля – сумарна вартість усіх квітів на полі, а також зображення насіння починаючи з 2 рівня, яке гравець отримає під час продажу квітів за цією вартістю. Продаж здійснюється натисканням на кнопку «SELL» внизу ігрового поля.

Квітки можна переміщати з однієї полиці на іншу, шляхом їх перетягування. У момент перетягування гравець може побачити всі характеристики квіток на ігровому полі sell. Залежно від характеристик самої квітки та вартості сусідніх із нею квіток ця квітка може збільшити свою вартість на бонусну величину. Вона відображається поруч з основною вартістю квітки на полиці, де знаходиться квітка, у вигляді значка «+» та її величини. Механіка продажу квітів наведена на рисунку 3.18.

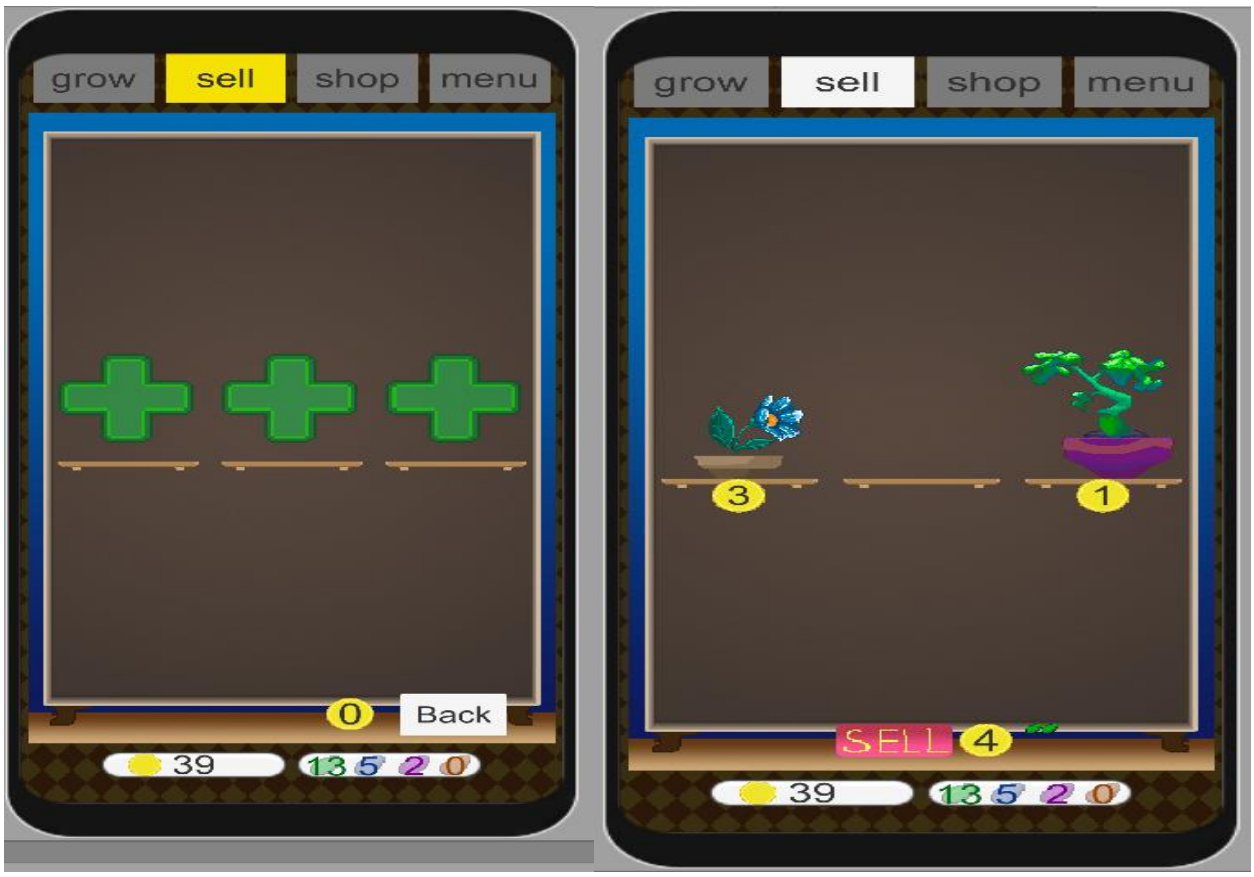

Рисунок 3.17 – Поле Sell

Рисунок 3.18 – Продаж квітів

Підбираючи різні комбінації розміщення квіток з різною вартістю на основі їхніх характеристик, можна значно збільшити підсумкову суму продажу квіток, що і є основною задумкою цієї гри як представника жанру головоломок (Puzzle). Крім цього, гравець має можливість витратити набрані за продаж квітів бали в магазині, купуючи там ігрові предмети, що поліпшують його ігровий процес (рис. 3.19). Відкрити магазин можна натиснувши на кнопку «shop». При першому отриманні ігрових очок гравець побачить інформаційне повідомлення про це, після закриття якого автоматично відкриється магазин. Інформація про наявні в магазині предмети гравцеві не розкривається, даючи можливість визначити її або по зображенню предмета, або залишатися у вигляді загадки, поки гравець не придбає предмет і не дізнається його призначення.

Рисунок 3.19 – Меню магазину

Ігрове поле «MENU» (рис. 3.20), має низку кнопок, натиснувши на які гравець може виконати такі дії:

– кнопка «i» – на екрані з’явиться інформаційне повідомлення, що нагадує основний сюжет гри;
– кнопка «Reset» – скидається весь ігровий прогрес і гравець починає грати з самого початку;
– кнопка «Exit» – додаток гри закривається, зберігаючи весь ігровий прогрес. Для продовження достатньо запустити гру знову.

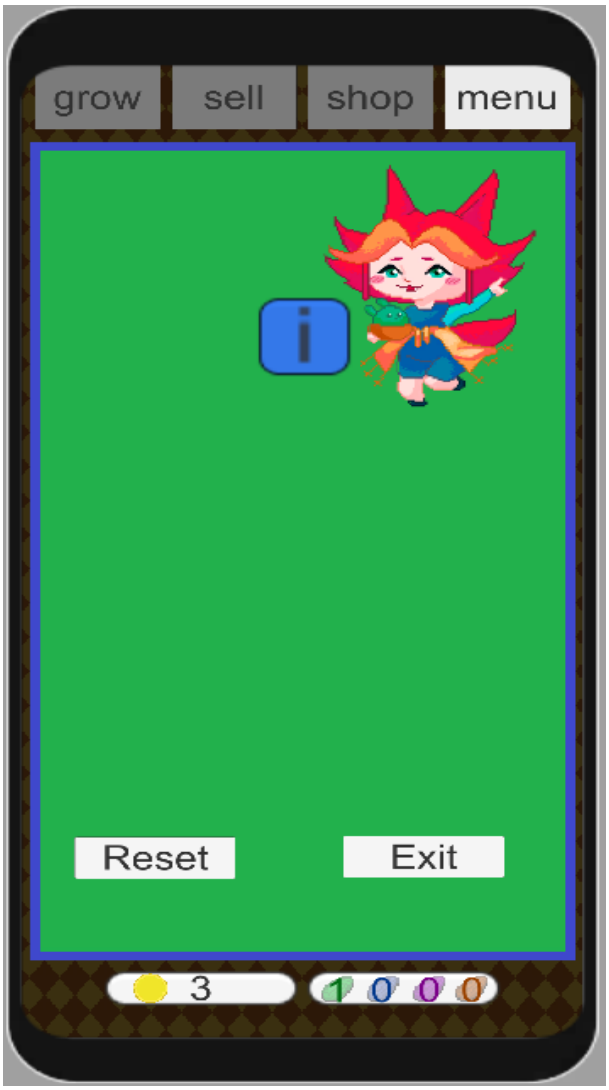

Рисунок 3.20 – Поле Menu

Додатково варто відзначити наявність у грі підтримки свайпів по екрану по вільних місцях, які перемикають поточне ігрове поле на сусіднє.

3.9 Висновки до розділу 3

У розділі «Реалізація та тестування» була проведена розробка та тестування мобільної гри жанру adventure/puzzle на платформі Unity. Процес реалізації включав створення основних механік гри, розробку графічного інтерфейсу, анімацій та ігрових сцен.

Під час розробки були реалізовані ключові функціональні можливості гри, включаючи вирішення головоломок, систему збереження прогресу та обробку подій. Був створений графічний інтерфейс включаючи спрайти та анімації, що покращує ігровий досвід гравця.

У результаті реалізації та тестування була отримана функціональна мобільна гра жанру adventure/puzzle, яка задовольняє поставлені вимоги та надає захоплюючий ігровий досвід для користувачів. Гра готова до подальшого розгортання на різних мобільних платформах та використання в реальних умовах [7].

ВИСНОВКИ

В рамках роботи була проведена розробка та реалізація мобільної гри жанру adventure/puzzle на платформі Unity. Робота включала аналіз вимог, проєктування гри, реалізацію функціональності, тестування та оцінку якості.

Було розроблено функціональну мобільну гру жанру adventure/puzzle, яка надає користувачам захоплюючий ігровий досвід та можливість вирішення головоломок.

Гра була реалізована з використанням платформи Unity, що дозволило забезпечити високоякісну графіку, зручний користувацький інтерфейс та плавний геймплей.

Були застосовані функціональні тести для перевірки коректності роботи основних механік гри та взаємодії між різними компонентами.

Робота була успішно завершена, мета проєкту була досягнута, а розроблена мобільна гра готова до подальшого розгортання на різних мобільних платформах та використання в реальних умовах. Результати роботи можуть бути використані для подальшого вдосконалення гри та її запуску.

ПЕРЕЛІК ПОСИЛАНЬ

1. Офіційна документація C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 15.01.2024).

2. Офіційна документація Unity. URL: https://docs.unity3d.com/Manual/index.html (дата звернення: 16.01.2024).

3. Офіційна документація юніті для скриптів. URL: https://docs.unity3d.com/ScriptReference/index.html (дата звернення: 21.03.2024).

4. Офіційна документація Adobe Photoshop. URL: https://helpx.adobe.com/photoshop/user-guide.html (дата звернення: 25.04.2024).

5. Статистика налаштувань Android користувачів. URL: https://www.antutu.com/en/doc/128924.html (дата звернення: 26.04.2024).

6. Код скриптів. URL: https://github.com/JIaIIyX/project2/tree/main/Assets/Scripts (дата звернення: 02.06.2024).

7. Посилання на мобільну гру. URL: https://drive.google.com/file/d/1SWLqufWmWZ2kAarXilLjIi7hufRkCRyt/view?usp=sharing (дата звернення: 02.06.2024).

8. Класова діаграма. URL: https://viewer.diagrams.net (дата звернення: 22.03.2024).