Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Шевченко Кирила Миколайовича*
(ПІБ)

академічної групи 122-18-3
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка кросплатформеної комп'ютерної гри в середовищі Unity 3D*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В*			
розділів:				
спеціальний	*доц. Спірінцев В.В*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-3* *Шевченко К.М.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка кросплатформеної комп'ютерної гри в середовищі Unity 3D*

затверджена наказом ректора НТУ «ДП» від 18.05.2022 № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проектно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ доц. Спірінцев. В.В.
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ Шевченко К.М.
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *16.06.2020 р.*

РЕФЕРАТ

Пояснювальна записка: 86 с., 24 рис., 2 табл., 3 дод., 20 джерел.

Об'єкт розробки: комп’ютерна гра жанру Action/RPG.

Мета кваліфікаційної роботи: реалізація стратегічної відеогри на платформі Unity 3D.

У вступі розглядається аналіз та сучасний стан проблеми, уточнюється мета та область застосування кваліфікаційної роботи, обґрунтовується тематика теми та з’ясовується проблема.

У першому розділі проводиться аналіз предметної області, визначається актуальність роботи та мета її розробки, опрацьовується завдання роботи, встановлюються вимоги до програмного забезпечення, технології та програмної реалізації. У другій частині аналізується існуюче рішення, вибирається платформа для розробки, проектується і розробляється програма, описується алгоритм і структура програми, визначаються вхідні та вихідні дані, надається параметрична характеристика технічних засобів. , а також опис виклику і завантаження додатку.

Економічна частина визначає складність розробленої інформаційної системи, розраховує вартість робіт зі створення програми, розраховує час, необхідний для створення програми.

Практичний сенс полягає у створенні програмного додатку, в якому реалізовано на основі алгоритму пошуку найкоротшого шляху для NPC [12] і ворогів, Action/RPG гру на платформі Unity які забезпечують функціональний кінцевий продукт для використання в розважальних цілях.

Актуальність цього програмного продукту продиктована високим попитом на таку розробку, оскільки зараз доцільно використовувати ігри для розваги людей. Користувачі можуть використовувати цю програму як гру для розваги та відпочинку.

Список ключових слів: UNITY, C#, КОМПОНЕНТИ, ІНТЕРФЕЙС, ПРОГРАМА, ДИЗАЙН, АЛГОРИТМ.

ABSTRACT

Explanatory note: 84 pp., 24 fig., 2 table., 3 app., 20 sources.

Object of development: computer game of the Action / RPG genre.

The purpose of the qualification work: the implementation of a strategic video game on the Unity 3D platform.

The introduction examines the analysis and current state of the problem, clarifies the purpose and scope of the qualification work, substantiates the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the work and the purpose of its development are determined, the tasks of the work are worked out, the requirements to the software, technology and software implementation are established. The second part analyzes the existing solution, selects a platform for development, designs and develops a program, describes the algorithm and structure of the program, determines the input and output data, provides parametric characteristics of technical means. , as well as a description of the call and download the application.

The economic part determines the complexity of the developed information system, calculates the cost of work to create a program, calculates the time required to create a program.

The practical meaning is to create a software application, which is implemented on the basis of the shortest path search algorithm for NPC [12] and enemies, Action / RPG game on the Unity platform, and which provide a functional end product for use for entertainment purposes.

The relevance of this software product is dictated by the high demand for such development, as it is now advisable to use games to entertain people. Users can use this program as a game for entertainment and recreation.

List of keywords: UNITY, C #, COMPONENTS, INTERFACE, PROGRAM, DESIGN, ALGORITHM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ГІК - Графічний інтерфейс користувача;

ІС - Інформаційна система;

ОС - Операційна система;

ООП - Об'єктно-орієнтоване програмування;

ПЗ - Програмне забезпечення;

UI/UX - UX (user experience) and UI (user interface);

ПК - Персональний комп’ютер.

ВСТУП

На сьогоднішній день ринок розваг є одним з найприбутковіших. З моменту початку інформаційно-технічної революції світ стрімко рухається в майбутнє, створюючи все більш досконалі комп'ютери.

З появою персональних комп'ютерів, з кожним роком, їх роль в житті людей постійно зростає. Комп'ютер став незамінним помічником не тільки в сфері економічних розрахунків, а й став потужним центром розваг.

Комп'ютерні розваги роблять життя людини більш насиченим і, як наслідок – це потужна економічна сфера, яка приносить величезні доходи.

Тому не випадково, що особливу роль у житі сучасної людини відводять комп'ютерним іграм, перші з яких існували на самій зорі комп'ютерної техніки.

У світі існує надзвичайно велика кількість любителів комп'ютерних ігор з різним досвідом в цій сфері. Є початківці, є навчені досвідом гравці, тому на цьому ринку затребувані ігрові проекти різної спрямованості.

Багато програмістів-любителів створюють невеликі ігрові програми, які не володіють сучасною дорогою висококласною графікою і звуковим супроводом, як ігри популярних компаній, але вони часто несуть якість

інновації в гемплеї, або володіють цікавим сюжетом, і в підсумку вони так само користуються великою популярністю і приносять творцям достатні доходи.

Метою кваліфікаційної роботи є реалізація стратегічної відеогри на платформі Unity.

Актуальність розробленої програми полягає в тому, що в наш час є доцільним використання ігор для розваг людей.

Дана програма - гра може використовуватись користувачами для розважальних заходів та приємного відпочинку.

В результаті виконання кваліфікаційної роботи створено додаток, в якому реалізовано Action/RPG гру на платформі Unity та запропоновано працеспроможний кінцевий продукт для його використання в розважальних цілях.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

1.1.1. Історія створення комп'ютерних ігор

У 1940 році американський фізик Едвард Кондон на Всесвітній виставці в Нью-Йорку продемонстрував пристрій під назвою Nimatron - першу в історії комп'ютерну ігрову електронно-релейну машину для гри в ньому (китайська гра, в якій два гравці по черзі беруть предмети, розкладені на кілька купок). Незважаючи на приголомшливий успіх на виставці, Nimatron незабаром був забутий — головним чином тому, що навіть батько-винахідник Едвард Кондон не усвідомив потенціал своєї гри.

Наступною важливою віхою в історії комп'ютерних ігор став 1952, коли британець Олександр Дуглас придумав OXO, також відома як Noughts And Crosses - комп'ютерні хрестики-нулики. Але й OXO залишилася недоступною широкому загалу, оскільки єдиний у світі комп'ютер, на якому можна було зіграти, стояв у Кембриджському університеті. Знадобилося ще 10 років, щоб комп'ютерні ігри вийшли у маси.

У квітні 1962 року компанія DEC розпочала продаж комп'ютерів PDP-1. У базову комплектацію як тестова програма входила гра SpaceWar, яка стала першою грою, випущеною у тираж.

З середини 1970-х років з'явилося багато класичних аркад, таких як Space Invaders, які пізніше були розроблені і для консолей. Онлайн-ігри, особливо текстові пригоди та онлайн-версії добре відомих настільних ігор, таких як шахи або шашки. З 1984 року онлайн-ігри продовжували поширюватися з появою нових комп'ютерів та переходом до Інтернету.В

На даний час усім комп'ютерним іграм призачаються класифікації. Вони не затверджені жодною організацією зі стандартизації. Критеріями віднесення ігор того чи іншого класу є її загальний бюджет, що включає витрати на розробку та рекламну кампанію, розмах самої рекламної кампанії, масштабність видавця гри. Основних класів 3: AAA (Triple-A); AA (Double-A); Інді-ігри.

1.1.2. Класифікація ігор

AAA (Triple-A) – це неофіційний клас високобюджетних відеоігор (найвищий клас). Це ігри масштабні, опрацьовані, складні, розраховані на аудиторію. З гарною технологічною графікою.

І це ігри з величезними фінансовими вкладеннями у розробку, над ними працюють сотні працівників, яких залучає видавець. І це ігри з величезними фінансовими вкладеннями у популяризацію, насамперед у рекламну кампанію. Часто бюджет на рекламну компанію гри порівняємо з бюджетом її розробки. Чим більший бюджет гри, тим теоретично більше можливостей зробити її висококласною – підібрати цікавий сюжет, опрацювати геймплей, постаратися із графікою. Клас ігор ААА порівняний із терміном «блокбастер» у кіноіндустрії.

Ці критерії застосовуються до ігор ААА: такі ігри повинні бути обов'язково успішні і нести окупність вкладених видавцем коштів. Бюджет ігор ААА стартує від $50 млн і може сягати неймовірних розмахів.

Клас ААА найвищий, проте можна зустріти застосування до окремих ігор якогось класу ААА+ (АААА або Quad-A). Це прийом, що виділяє гру з інших ААА. Формально для відповідності класу ААА+ гра має бути масштабною, підтримуватись оновленнями, передбачати доповнення, мати власний магазин. Цим критеріям відповідають багато ігор ААА, яких не зараховують до ААА+.

АА (Double-A або B) — це клас відеоігор нижче за ААА, середній клас, і це ігри з істотно меншими бюджетами, до $50 млн. У таких ігор немає достатку ігрових механік. І вони не можуть похвалитися бездоганною графікою.

Межі між цими двома класами ігор розмиті, і іноді грі АА не вистачає якоїсь дещиці, щоб опинитися в класі ААА. У якомусь із аспектів реалізації вона програє, і це – вже другий сорт. З відомих ігор АА класу: S.T.A.L.K.E.R., Metro 2033, Deus Ex, Dishonored, Lost Planet, Fallout 76, Call of Juarez, Total Overdose, Burnout Paradise, Singularity, Sleeping Dogs, Spec Ops: The Line, Alan Wa, BioShock, Wolfenstein.

Ризикуючи меншими сумами, що вкладаються в розробку, творці ігор АА більш схильні до різних експериментів, можуть реалізувати найсміливіші творчі задуми, привносити новизну. Не вкладаючись особливо у графіку, розробники ігор можуть зосередити зусилля на геймплеї та вигадати незвичайний захоплюючий сюжет. До речі, сюжети зазвичай цікавіші якраз у ігор АА. Раніше такі ігри виходили частіше, ніж ААА. Зараз же АА – це клас, що вмирає. Не маючи грошей на великомасштабну рекламну кампанію, вони губляться у тіні ігор ААА. І користувач просто не може дізнатися про них. Успішним інді-проектам гри АА програють необхідністю наявності достатнього бюджету на розробку.

Інді-ігри – це незалежні відеоігри, створені без участі видавця, одним розробником або вузьким колом розробників. Це немасштабні ігри, з малим бюджетом чи його відсутністю зовсім. З мізерною графікою, але гарним геймплеєм, часто унікальним. Відсутність бігбосів дає можливість розробникам максимально відійти від шаблонів ігрової індустрії та створити самородок. І іноді такий самородок стає золотим. Найяскравіший приклад – Minecraft.

Інші інді-ігри: Timberborn, Journey, Braid, World of Goo, Super Meat Boy, Limbo, Inside.

Інді-ігри мають свій підклас - III (Triple-I), це ігри з великими бюджетами, з амбіціями ігор ААА. Але, як і звичайні інді-ігри, вони незалежні, розробляються у вузькому колі, фінансуються рахунок краудфандинга (народного фінансування) чи коштів угруповань розробки ігор. Це проекти, які пропонують інновації та креативні ідеї. Приклади таких ігор: The Witness, Hellblade: Senua's Sacrifice, Ancestors: The Humankind Odyssey.

1.1.3. Жанри комп'ютерних ігор

Внаслідок того, що критерії приналежності гри до того чи іншого жанру не визначені однозначно, класифікація жанрів комп'ютерних ігор недостатньо систематизована, і в різних джерелах дані про жанр конкретного проекту можуть відрізнятися. Проте, існує консенсус, якого прийшли розробники ігор, і належність гри до одного з основних жанрів майже завжди можна визначити однозначно.

Найпопулярніші жанри ігор:

- симулятори;
- екшен;
- стратегії;

- спортивні;
- пригоди;
- рольові ігри (RPG);
- головоломки;
- текстові.

Існують ігри з елементами кількох жанрів, які можуть належати кожному з них (наприклад, серія Grand Theft Auto, Космічні Рейнджери, Rome: Total War та багато інших). Такі проекти зараховують або до одного з жанрів, який у грі є основним, або одразу до всіх присутніх у грі, якщо вони однаково складають геймплей проекту.

1.1.4. Опис жанру Action/RPG

Жанр Action/RPG поширений насамперед на приставках. Мотивується це хоча б тим, що A/RPG запозичує детальне опрацювання сюжету.

Жанр досить сильно відходить від канонів рольової гри у її настільному варіанті. В Action/RPG бої йдуть у реальному часі, і ваш успіх залежить не тільки від характеристик героя, а й від того, наскільки швидко ви натиснете кнопку. Відповідно, не тільки відсутній відіграш ролі та нелінійність сюжету, а й знижується значення екіпіровки та характеристик героя. Відрізняються вони від бойовиків з елементами RPG (наприклад, Diablo або Dark Stone на PSX) такі ігри зазвичай тим, що є сюжет, майже не поступається сюжету класичних RPG. Крім того, часто ігри поділяються на Action/RPG та «Action з елементами RPG» чисто умовно.

Наприклад, Beyond Oasis aka The Story of Thor перша та друга частини (Megadrive і Sega Saturn відповідно) прийнято відносити саме до Action/RPG, хоча зовні гра виглядає на подобі аркад Streets of Rage або Final Fight, лише трохи в іншій проекції. Одночасно Diablo незважаючи на його

систему зброї прийнято відносити до hack'n'slash [3] або rogue (аркада з елементами RPG, слабким сюжетом та рівнями, що створюються випадковим чином), а не Action/RPG. Плюс, ще одна відмінність Action/RPG – зазвичай у таких іграх завжди є кілька головоломок, нехай і не таких, як у жанрі Quest. Вони допомагають розбавити бойовик спокійним пошуком рішення нехай і простеньких, але все ж таки не завжди очевидних завдань. Саме тому любителі PC часто називають Action/RPG «адвентюрами», хоча класичний Adventure – це все ж таки дещо не те.

1.2. Призначення розробки та галузь застосування

Комп'ютерні ігри є невід'ємною частиною дозвілля більшості сучасних молодих людей як в Україні, так і у всьому світі.

Сучасна комп'ютерна гра – це багатофункціональна програма, яку використовують не тільки з розважальними, а і з навчальними та пропагандистськими цілями.

В результаті виконання кваліфікаційної роботи буде створено кросплатформенну гру в жанрі Action/RPG на платформі Unity 3D та запропоновано працеспроможний продукт для його використання в розважальних цілях.

Буде використано середовище Unity 3D та мову програмування C#.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випусковою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 122 «Комп'ютерні технології»
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» No _____від____.____.2022 р;
- завдання на кваліфікаційну роботу на тему «Розробка кросплатформеної комп’ютерної гри в середовищі Unity 3D».

1.4. Постановка завдання

Метою кваліфікаційної роботи є реалізація стратегічної відеогри жанру Action/RPG на платформі Unity.

Правила гри наступні: У кожному матчі (бою) гравець має перемогти всіх ворогів.

Під час бою головний герой отримує винагороду з кожного переможеного ним юніта. Ця винагорода є важливим лутом для гравця: їжа, зброя, окуляри досвіду.

Гравець програє у разі якщо юніти завдали йому шкоди перевищує кількістю життів або якщо минув рівень заряду головного героя.

Для виконання даного завдання необхідно розробити програму – комп'ютерну стратегічну гру. В результаті виконання даної роботи розробник засвоює прийоми практичного використання об'єктно-орієнтованого підходу до створення кінцевого програмного продукту:

- реалізує обрану комп'ютерну гру;
- володіє методами побудови графічного інтерфейсу користувача;
- задовольняє вимогу переносимості на рівні вихідного коду, простого в установці і обслуговуванні.

Для вирішення цих задач при розробці ігрового додатку повинні бути використані наступні рішення:

1. Розробка логічної схеми додатку.
2. Розробка алгоритму роботи додатку.
3. Розробка графіки додатку.
4. Розробка інтерфейсу користувача.
5. Розробка ігрового процесу.
6. Тестування розробленої програми.

1.5. Вимоги до програми або програмного виробу

Вимоги до гри - забезпечення безперервної праці системи, недопустимості збоїв у обробці запросів чи роботі коду, візуалізація даних та створення даних, на основі яких формується статистика для гравця.

Базовими критеріями для ефективної роботи коду є безперервна обробка запитів, захист даних користувача та захист ігрових даних від користувача.

Основний ряд вимог до додатка:

- додаток має виконувати головну функцію створення, редагування та подальше відображення даних на головному екрані

- код має будет гнучкий для компіляції гри під різні платформи (Android, Ios, Mac, Windows, Linux, PS, Xbox)

- код гри має бути виконаний в найкращому стилі для подальшої підтримки, мати ефективну та чисту архітектуру та інтуїтивно легкий для використання інтерфейс (UI/UX [19])

1.5.1. Вимоги до функціональних характеристик

До розроблюваного програмного забезпечення виставляються наступні функціональні характеристики:

1. Головний герой. Переміщення [7] головного герою відбуватиметься у двомірному просторі, можна рухатися вліво та вправо.

Для головного героя необхідно розробити:

- 2D спрайт;
- анімації простою та ходьби;
- додаткові ефекти;
- звуки;
- унікальні атаки.

2. Поле бою. Необхідно розробити карту, яка буде відображати зміст назви гри, а саме демонструвати жанр гри Action/RPG в якій гравцю потрібно воювати з різними юнітами. Також продумати контролер камери, щоб гравець зміг знайти для себе оптимальне положення, для більш чіткого сприйняття інформації.

3. Графічний інтерфейс [19]. При закінчені гри, необхідно відобразити панель перемоги або програшу та статистику проведеного гравцем часу в бою. В будь-який момент гра може бути призупинена для редагування користувачем управління переміщення та гучності звуків.

4. Головне меню міститиме кнопки виходу з гри та ігрових налаштувань. Додатково, для ознайомлення з правилами гри та особливостями додатку, в головному меню міститиметься кнопка інструкції, яка повинна включити підказки для гравця.

1.5.2. Вимоги до інформаційної безпеки

Для надійної роботи системи необхідно:

- використовувати ліцензійне програмне забезпечення;
- здійснювати захист від несанкціонованого доступу;

- застосовувати джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні;

- здійснювати контроль даних, що вводяться користувачем.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для запуску відеогри необхідна конфігурація ПК не гірше мінімальної.

Мінімальна конфігурація для запуску гри:

- операційна система: Windows 7 або більше;
- процесор: 1.2 ghz;
- оперативна пам'ять: 4 GB ОЗУ;
- відеокарта: 512 Мб відеопам'яті;
- DirectX: Версії 10;
- місце на диску: Не менше 1 ГБ.

Такий вибір компонентів є оправданим тому, що:

- процесори підтримують інструкції SSE2, необхідні для запуску;
- відеокарти підтримують версії DirectX, які використовуються;
- кількості і швидкості ОЗП вистачає для процесів;
- пам’яті ЖД вистачає для зберігання ігрових файлів.

Також для вводу даних необхідна мишка та клавіатура. Для виводу картинки потрібен монітор.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для коректної роботи відеогри потрібні наступні програмні засоби:

- DirectX – це набір компонентів Windows, який дає змогу програмному забезпеченню, в основному й особливо іграм, працювати напряму з відео і аудіо устаткуванням. Він допомагає ефективніше використовувати функції мультимедійного акселератора, вбудовані в устаткування, що покращує якість відтворення мультимедійного вмісту в цілому. Для запуску гри потрібна 10 версія набору;

- платформа .NET Framework – це середовище виконання, яке керує додатками, воно складається з середовища CLR, яке надає інструменти управління пам'яттю, інші служби системи, та велику бібліотеку класів, що дозволяє програмістам використовувати стійкий, надійний код у всіх основних областях розробки додатків;

- ОС Windows (або Mac OS) забезпечує комп’ютер усіма необхідними драйверами, але драйвера відеокарти необхідно оновити до останньої версії для покращення продуктивності.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Завданням кваліфікаційної роботи є спроектувати та розробити відео гру в жанрі Action/RPG на платформі Unity, метою якої є створення оптимальних умов праці, які створюють у людини відчуття задоволення виконаної роботи.

- переміщення головного героя відбувається в двомірному просторі, можна рухатися вліво та вправо.
- на полі бою головному герою потрібно воювати з різними юнітами та добувати предмети які гравець зможе використати для відновлення своїх характеристик (Кількість життів, Витривалість, Голод, Спрагу).
- при закінчені гри відображається панель перемоги або програшу та статистику проведеного гравцем часу в бою. В будь-який момент гра може бути призупинена для редагування користувачем управління переміщення та гучності звуків.
- головне меню містить кнопки виходу з гри та ігрових налаштувань. Додатково, для ознайомлення з правилами гри та особливостями додатку, в головному меню міститься кнопка інструкції, яка повинна включити підказки для гравця.

2.2. Опис застосованих математичних методів

Під час проектування та розробки додатку використовувався математичний метод пошуку найкоротшого шляху - Дейкстри.

Алгоритм Дейкстри будує найкоротші шляхи, які ведуть вихідної вершини графа до інших вершин цього графа (якщо такі є).

У процесі роботи алгоритму послідовно відзначаються розглянуті вершини графа. Причому вершина, позначена останньою розташована ближче до вихідної вершини, ніж усі непомічені, але далі, ніж усі помічені.

Спочатку позначається вихідна вершина; наступною, очевидно, буде позначено вершину, найближчу до вихідної, і суміжну з нею.

Нехай на якомусь кроці вже помічено кілька вершин. Відомі найкоротші шляхи, що ведуть з вихідної вершини до помічених. Для кожної з непомічених вершин виконаємо наступне:

Розглянемо всі дуги, що ведуть із помічених вершин в одну непомічену. Кожна така дуга є останньою дугою на шляху з вихідної вершини до цієї непоміченої.

Виберемо з цих шляхів найкоротший. А потім виберемо серед них найкоротший до всіх непомічених вершин і позначимо вершину, до якої він веде.

Алгоритм завершиться, коли будуть помічені всі досяжні вершини.

Внаслідок роботи алгоритму Дейкстри будується Дерево найкоротших шляхів.

2.3. Опис використаних технологій та мов програмування

2.3.1. Опис використаних засобів програмування

Для запуску гри на пк необхідна версія ОС Windows не нижче 7 SP1, але для кращої стабільності необхідно використовувати більш сучасні версії.

Unity3d [1] є сучасним крос-платформним движком для створення ігор і додатків, розроблений Unity Technologies. За допомогою даного движка можна розробляти не тільки додатки для комп'ютерів, але і для мобільних пристроїв, ігрових приставок і інших девайсів. Інтерфейс платформи зображено на рис. 2.1.

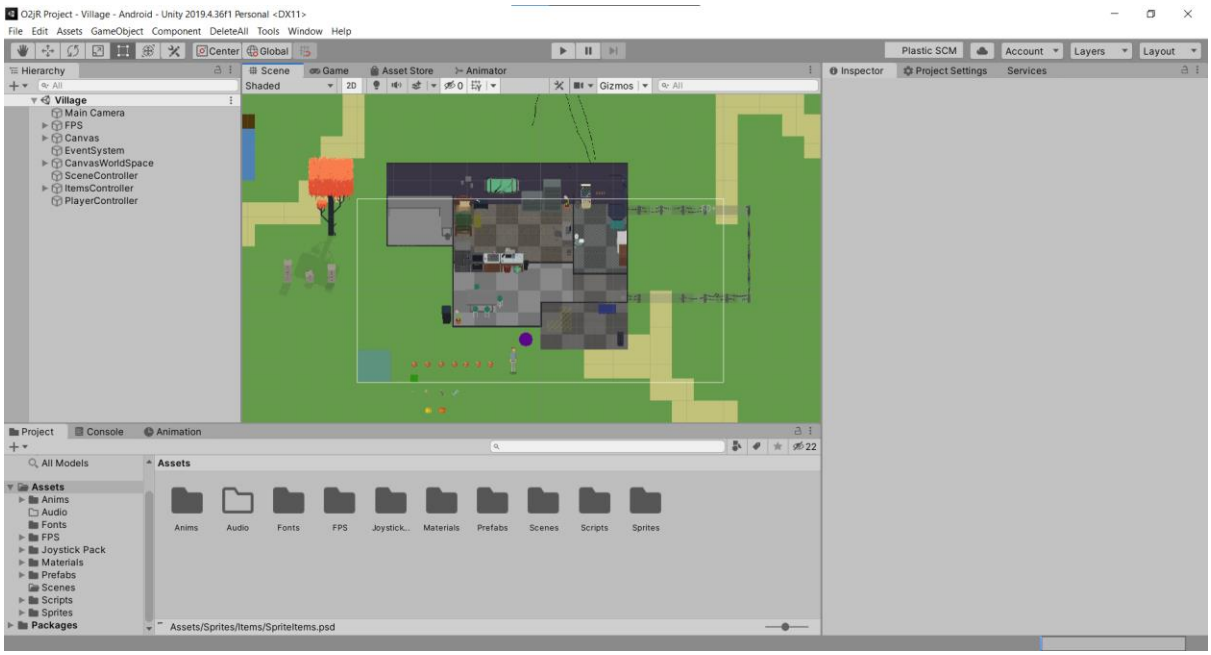

Рис. 2.1. Інтерфейс платформи Unity

Характеристики платформи:

- по-перше, в середовище розробки Unity інтегрований ігровий движок, за допомогою якого можна протестувати свою гру не виходячи з редактора;

- по-друге, Unity підтримує імпорт величезної кількості різних форматів, що дозволяє розробнику гри конструювати самі моделі в більш зручному додатку, а Unity використовувати за прямим призначенням - розробки продукту;

- по-третє, написання сценаріїв (скриптів) здійснюється на найбільш популярних мовах програмування - C# і JavaScript.

Таким чином, Unity3d є актуальною платформою, за допомогою якої можна створювати свої власні додатки і експортувати їх на різні пристрої, будь то мобільний телефон або приставка Nintendo Wii.

Для того щоб створити свою гру, потрібно володіти однією з доступних (на Unity) мов програмування: C #, JavaScript або Boo.

Їх компоненти мають ряд властивостей або змінних, які можна налаштувати як у вікні Inspector редактора Unity, так і за допомогою скрипта.

Unity надає багато влаштованих компонентів, але можна створити власний для реалізації специфічних алгоритмів. Це можна зробити за допомогою створення користувацького скрипта та прикріплення його до необхідного об’єкта. Кожен скрипт зв'язується з внутрішніми механізмами Unity шляхом реалізації класу, похідного від вбудованого класу MonoBehaviour.

Скриптовий компонент дозволяє запускати ігрові події. Об’єкти тестування на зіткнення використовують фізичні властивості, програмовані відповіді на елементи керування користувача тощо.

C# [2] є об'єктно-орієнтованою мовою, але підтримує також і компонентноорієнтоване програмування. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних і самоописувальних пакетів, що реалізують окремі функціональні можливості. Головна особливість таких компонентів в тому, що вони являють собою модель програмування з властивостями, методами і подіями. У них є атрибути, що надають декларативні відомості про компоненті. Вони включають в себе власну документацію. C # надає мовні конструкції, безпосередньо підтримують, таку концепцію роботи. Завдяки цьому C # підходить для створення і застосування програмних компонентів.

Ось лише кілька функцій мови C #, що забезпечують надійність і стійкість додатків. Прибирання сміття автоматично звільняє пам'ять, зайняту недосяжними невикористовуваними об'єктами. Обробка винятків надає структурований і розширюваний підхід до виявлення помилок і їх відновлення. Типобезпечна структура мови унеможливлює читання з не ініціалізованих змінних, індексацію масивів за межами їх кордонів або виконання неперевірених привидів типів.

У C# всі типи даних, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Таким чином, всі типи використовують загальний набір операцій і значення будь якого типу можна зберігати, передавати і обробляти схожим чином. Крім того, C# підтримує призначені для користувача посилальні типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стек.

Мова програмування забороняє звернення до змінних, що не були ініційовані, що виключає можливість виконання безконтрольного приведення типів або виходу за межі певного масиву даних.

Для роботи додатків на C# необхідно встановити і налаштувати платформу NET Framework. Платформа вбудована в інсталяційний пакет Windows, при необхідності її також можна скачати та інсталювати окремо. Існують також версії для Linux і MAC.

В рамках платформи до обробки виконуваного коду під’єднується середовище CLR – єдиний об'єднаний набір бібліотек і класів, який був розроблений Майкрософт і є реалізацією світового стандарту Common Language Infrastructure (CLI).

Основний механізм CLR має вигляд бібліотеки під назвою mscoree.dll (називається загальним механізмом виконання виконуваного коду об'єктів – Common Object Runtime Execution Engine). При додаванні посилання на збірку для її використання, завантаження бібліотеки mscoree.dll

здійснюється автоматично і потім, в свою чергу, призводить до завантаження необхідної збірки в пам'ять.

Щоб забезпечити сумісність програм і бібліотек C # при подальшому розвитку, при розробці C # багато уваги було приділено управлінню версіями. Багато мови програмування обходять увагою це питання. В результаті програми на цих мовах ламаються частіше, ніж хотілося б, при виході нових версій залежних бібліотек. Питання управління версіями істотно вплинули на такі аспекти розробки C#, як роздільні модифікатори virtual і override, правила вирішення перевантаження методів і підтримка явного оголошення членів інтерфейсу.

У недавніх версіях C# були використані інші парадигми програмування. C# включає функції, що підтримують прийоми функціонального програмування, такі як лямбда-вирази. Інші нові можливості підтримують поділ даних і алгоритмів, наприклад зіставлення шаблонів.

Організаційна структура C# ґрунтується на таких поняттях, як типи і змінні;

- класи є найважливішим типом в мові C#. Об'єкти представляють собою екземпляри класів. Класи створюються описом їх членів;

- масив - це структура даних, що містить кілька змінних, доступ до яких здійснюється за який обчислюється індексам;

- інтерфейс визначає контракт, який може бути реалізований класами і структурами. Інтерфейс може містити методи, властивості, події і індексатори. Інтерфейс не надає реалізацію членів, які в ньому визначені. Він лише перераховує члени, які повинні бути визначені в класах або структурах, що реалізують цей інтерфейс;

- delegate представляє посилання на методи з конкретним списком параметрів і типом значення, що повертається. Делегати дозволяють використовувати методи як сутності, зберігаючи їх у змінні і передаючи в

якості параметрів. Принцип роботи делегатів близький до покажчиків функцій з деяких мов, але на відміну від покажчиків функцій делегати є об'єктно-орієнтованими і строго типізовані;

- атрибути дозволяють програмам вказувати додаткові описові да-ні про типи, членах та інших сутності.

Інтегроване середовище розробки Visual Studio [11] - це стартовий майданчик для написання, налагодження і складання коду, а також подальшої публікації додатків. Інтегроване середовище розробки (IDE) являє собою багатофункціональну програму, яку можна використовувати для різних аспектів розробки програмного забезпечення. Крім стандартного редактора і відладчика, які існують в більшості середовищ IDE, Visual Studio включає в себе компілятори, засоби авто-завершення коду, графічні конструктори і багато інших функцій для спрощення процесу розробки. Інтерфейс середи (рис.2.2).

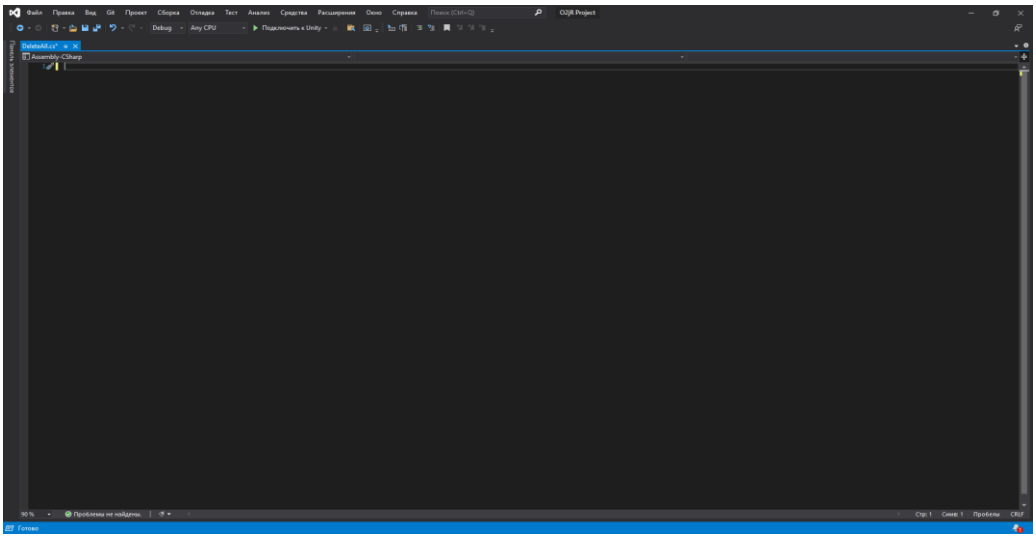

Рис. 2.2. Інтерфейс середовище Visual Studio

На рисунку показано середовище Visual Studio з відкритим проектом і декількома вікнами основних інструментів.

Оглядач рішень (вгорі праворуч) дозволяє переглядати файли коду, пересуватися по ньому і управляти ними. Оглядач рішень дозволяє впорядкувати код шляхом об'єднання файлів в рішення і проекти.

У вікні редактора (центр), відображається вміст файлу. Тут можна редагувати код або розробляти призначений для користувача інтерфейс, наприклад вікно з кнопками або текстові поля.

Team Explorer (правий нижній кут) дозволяє відслідковувати робочі елементи і використовувати код спільно з іншими користувачами за допомогою технологій управління версіями, таких як Git і система управління версіями Team Foundation (TFVC).

Функціональність Visual Studio охоплює всі етапи розробки програмного забезпечення, надаючи сучасні інструменти для написання коду, проектування графічних інтерфейсів, збірки, налагодження і тестування програм. Можливості Visual Studio можуть бути доповнені шляхом підключення необхідних розширень.

Хвилясті лінії позначають помилки або потенційні проблеми коду прямо під час введення. Ці візуальні підказки дозволяють усувати проблеми негайно і не чекати, поки помилка буде виявлена під час збирання або запуску програми. Якщо навести курсор миші на хвилясту лінію, на екран будуть виведені додаткові відомості про помилку. Крім того, в полі зліва може з'являтися значок лампочки з швидкими діями щодо усунення помилки.

Можна одним натисканням кнопки відформатувати код і застосувати до нього виправлення, запропоновані параметрами стилю коду, які вказані в файлі EditorConfig. Очищення коду допомагає усунути багато проблем в коді ще до перевірки.

Рефакторинг включає в себе такі операції, як інтелектуальне перейменування змінних, витягування однієї або декількох рядків коду в новий метод, зміна порядку параметрів методів і багато іншого.

Редактор коду Visual Studio підтримує підсвічування синтаксису, вставку фрагментів коду, відображення структури і пов'язаних функцій. Істотно прискорити роботу допомагає технологія IntelliSense [13] – автозавершення коду під час введення.

Вбудований відладчик Visual Studio використовується для пошуку і виправлення помилок у вихідному коді, в тому числі на низькому апаратному рівні. Інструменти діагностики дозволяють оцінити якість коду з точки зору продуктивності і використання пам'яті.

Visual Studio надає комплекс інструментів для автоматизації тестування додатків в частині перевірки роботи інтерфейсів, модульного і навантажувального тестування.

Система налагодження Visual Studio дозволяє переглядати код з кроком в одну інструкцію, перевіряючи значення змінних. Можна задати точки зупину, які зупиняють виконання коду на певному рядку. Таким чином можна побачити як значення змінної змінюється по мірі виконання коду.

Adobe Photoshop — багатофункціональний графічний редактор, що розробляється та розповсюджується компанією Adobe Systems. Здебільшого працює з растровими зображеннями, однак має деякі векторні інструменти. Продукт є лідером ринку в області комерційних засобів редагування растрових зображень та найвідомішою програмою розробника.

В даний час Photoshop (рис. 2.3) доступний на платформах macOS, Windows та iPadOS. Для Windows Phone та Android доступна спрощена версія програми під назвою Adobe Photoshop Touch. Також існує версія Photoshop Express для Windows Phone 8 та 8.1. У 2014 році у США проходило бета-тестування потокової версії продукту для Chrome OS. Ранні версії редактора були портовані під SGI IRIX, але офіційна підтримка припинена з третьої версії продукту. Для версій 8.0 та CS6 можливий запуск під Linux за допомогою альтернативи Windows API – Wine.

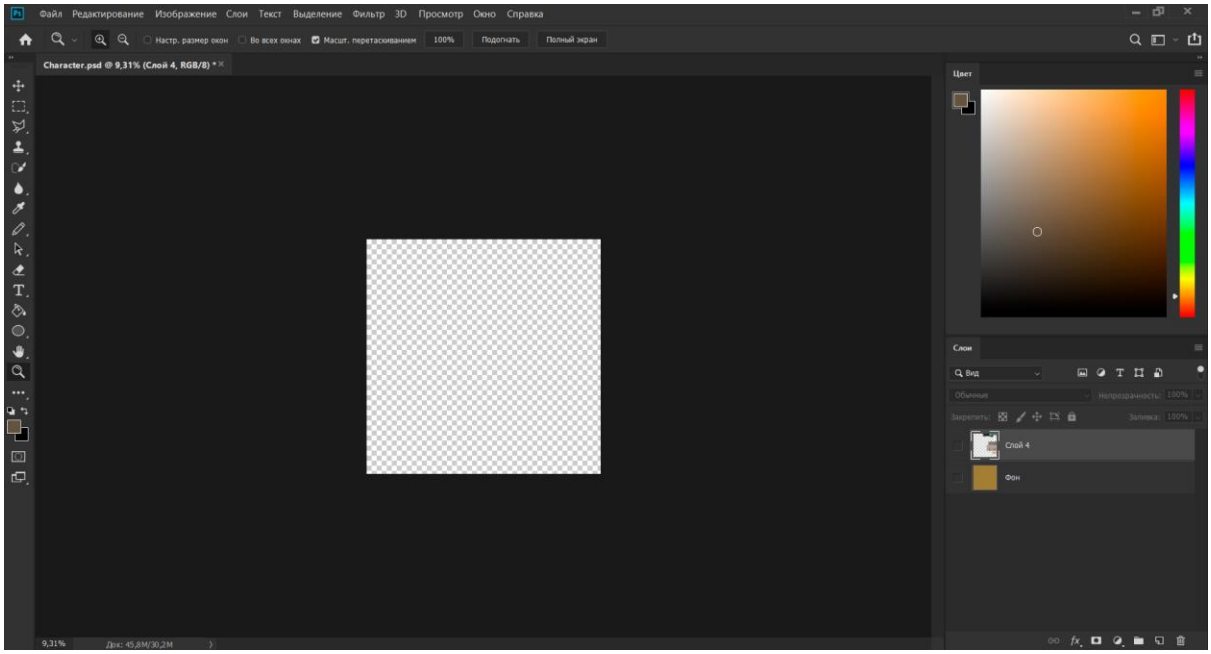

Рис. 2.3. Вікно програми Photoshop

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Загальний опис структури програми

Гра розрахована на одного гравця. Гравець грає за добру сторону, ворог за ворожу. Є карта на якій гравець та ворог [12] вільно переміщаються. Основна мета гри полягає в тому, щоб перемогти всіх запропонованих ворогів.

Взаємодію користувача з системою графічно відображено на діаграмі використання (рис.2.4 та 2.5).

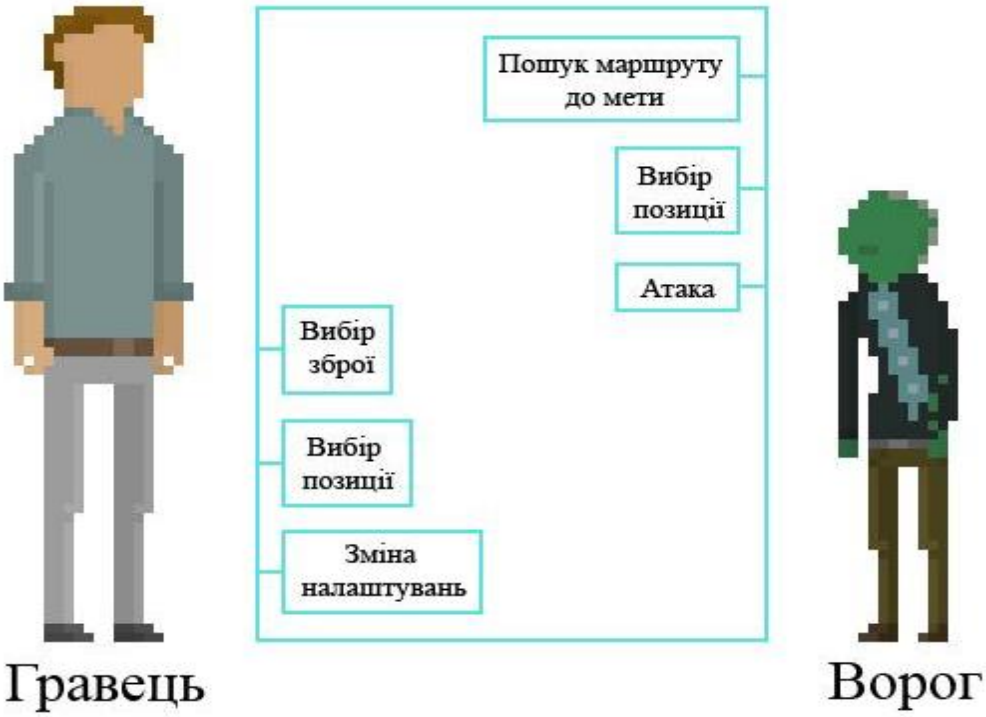

Рис. 2.4. Діаграма використання

Рис. 2.5. Діаграма діяльності

Склад файлової системи розробленого проєкту зображено на рис. 2.6:

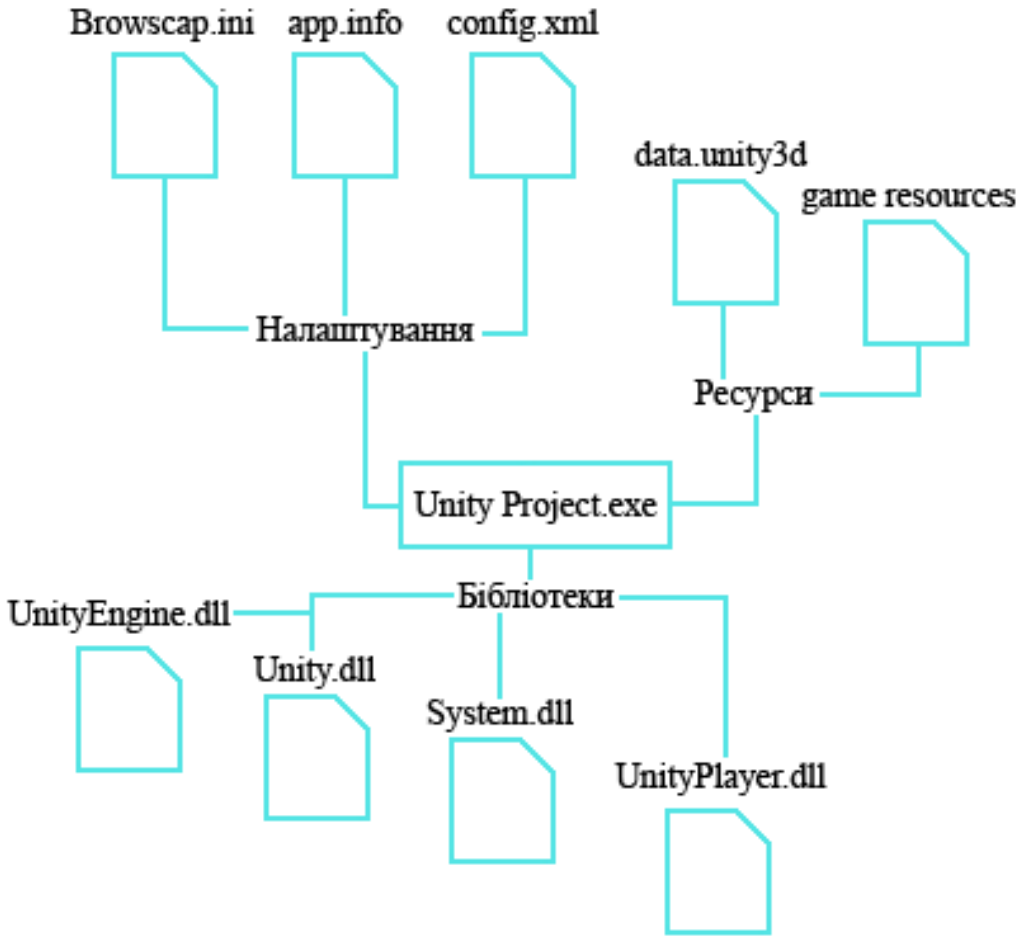

Рис. 2.6. Файлова система проекту

2.4.2. Опис основних компонентів

В платформі Unity кожен об’єкт представляє собою GameObject, який зберігає атрибути користувацьких та влаштованих компонентів. Так, наприклад, стандартним компонентом GameObject, який не можливо видалити є «Transform» (рис. 2.7). Він зберігає параметри Position, Rotation, та Scale відповідно позиція, поворот і розмір за трьома осями (X,Y,Z) відносно початку координат.

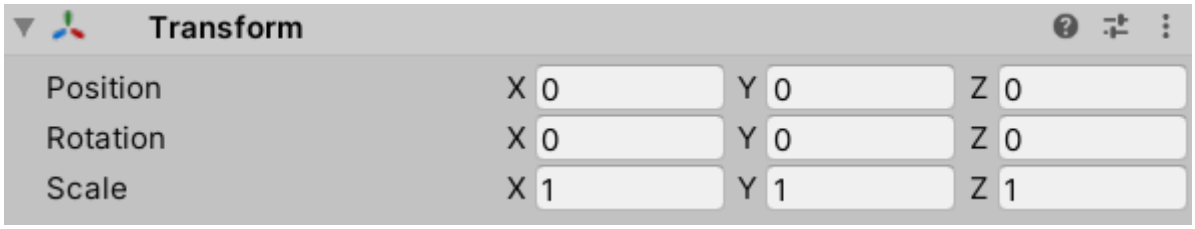

Рис. 2.7. Компонент «Transform»

Щоб відобразити двомірні зображення в тривимірному ігровому просторі використовується компонент «Sprite renderer» (рис. 2.8) Після вказівки у властивості Sprite посилання на імпортоване в проект гри над зображення можна проводити різні маніпуляції, такі як обертання, переміщення, зміна кольору тощо.

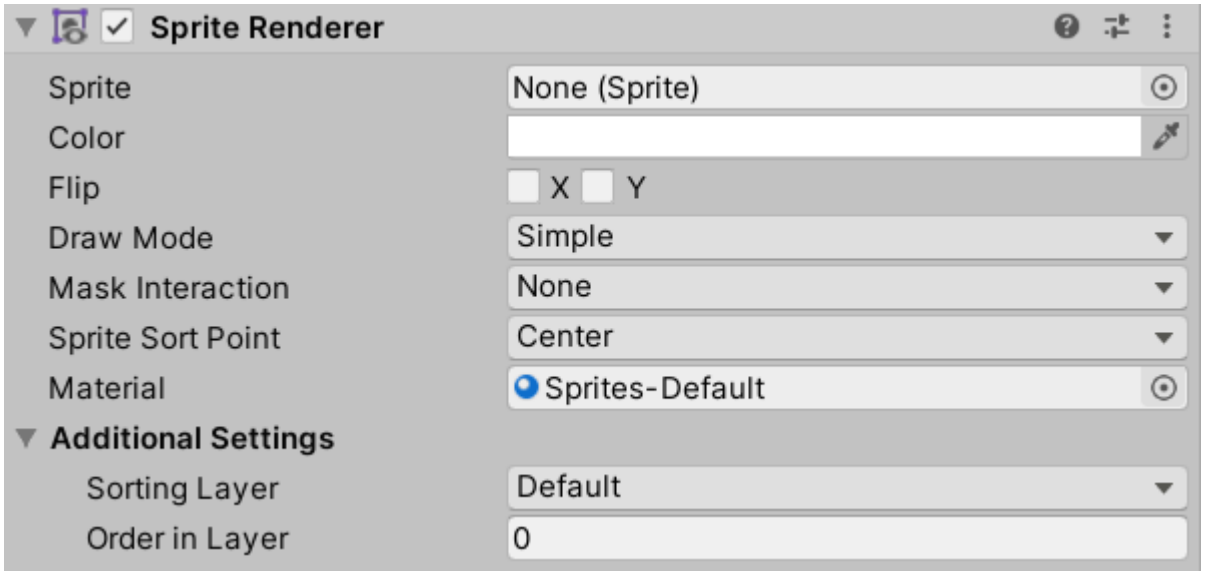

Рис. 2.8. Компонент «Sprite renderer»

Спроектовану анімацію гри розглянемо на прикладі фігури, для якої в проекті застосовано компонент «Animator» (рис. 2.9). Він зв’язує об’єкт з файлами анімацій (.anim), що створюються у вікні «Animation».

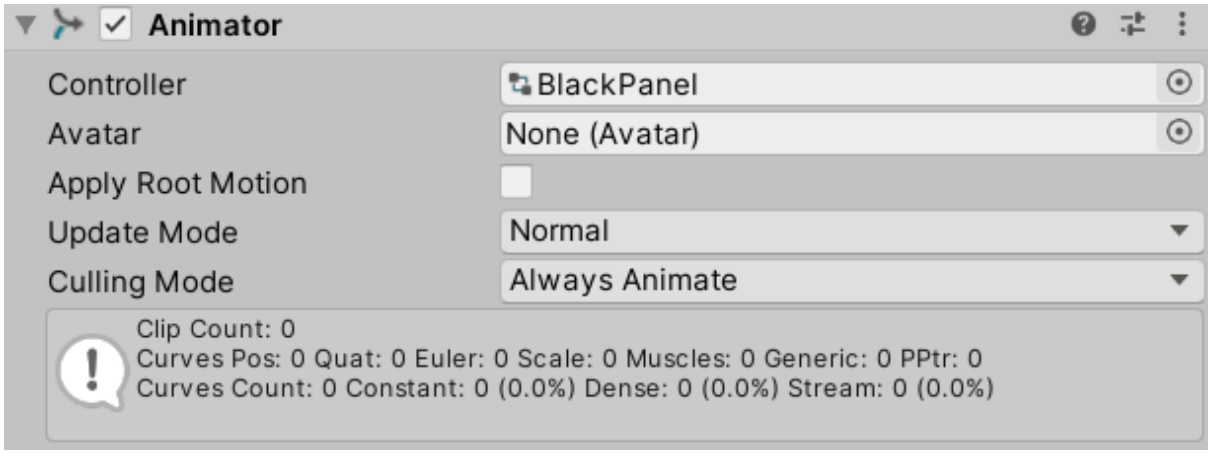

Рис. 2.9. Компонент «Animator»

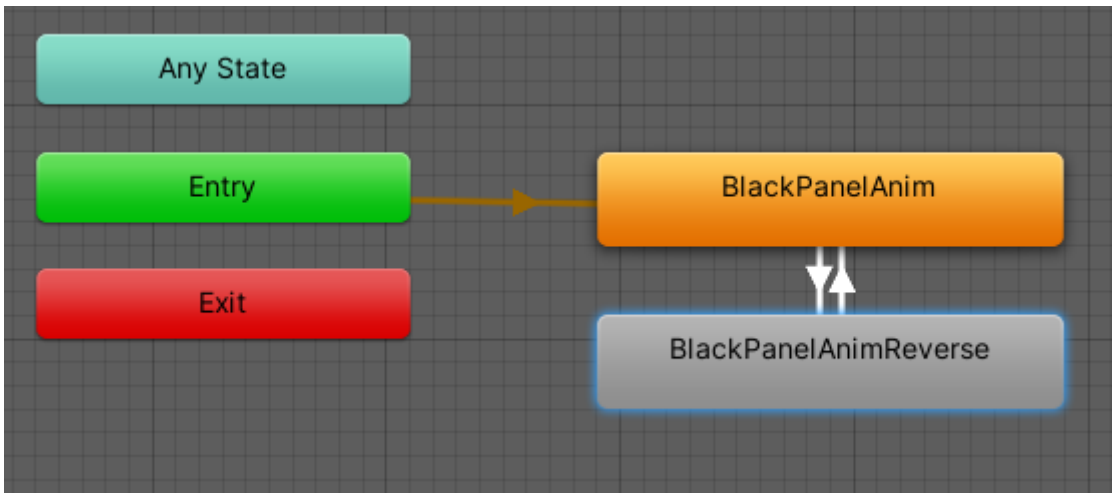

Рис. 2.10. Вікно «Animator»

Група компонентів «Collider» додає колізію, тобто фізичну взаємодію з іншими об’єктами. Розглянемо компонент «Capsule Collider» (рис. 2.11) – він не дає головному героя або іншому об'єкту пройти крізь інший твердий об'єкт.

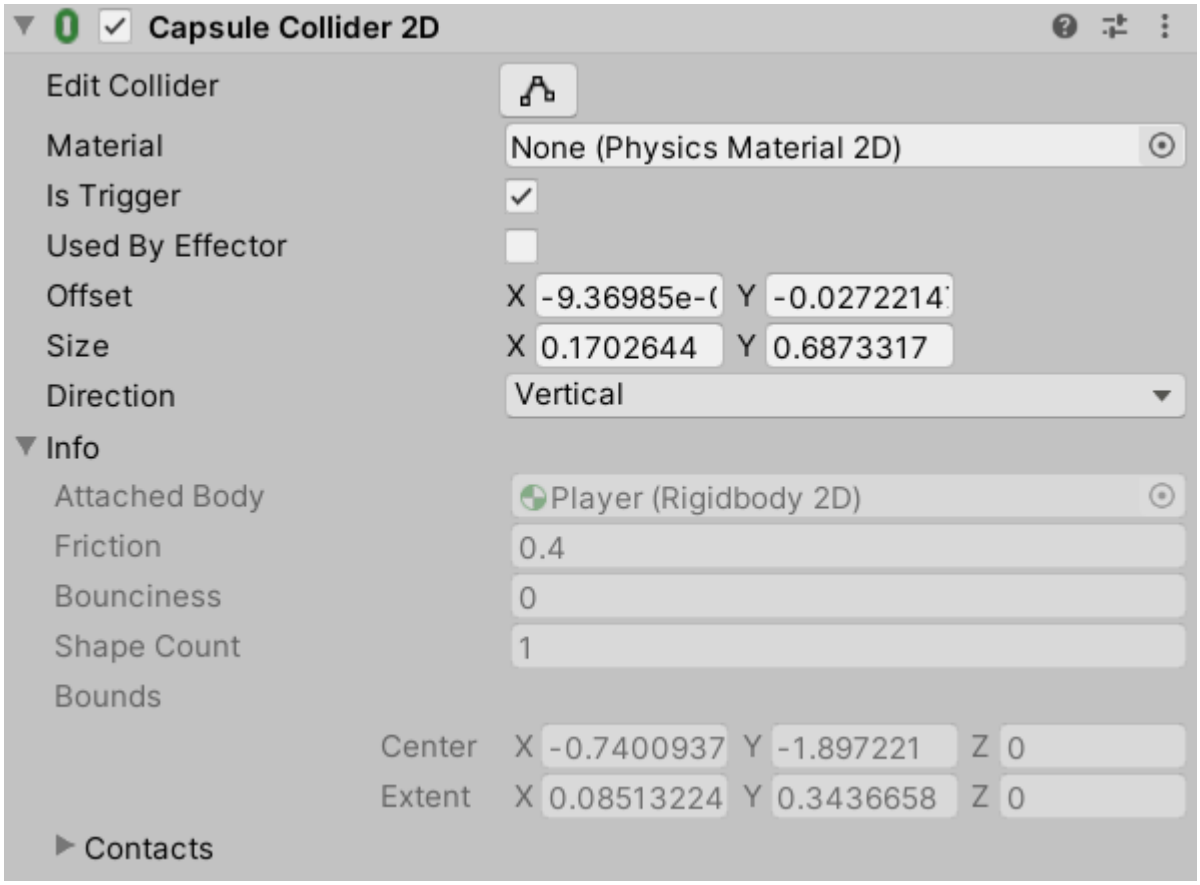

Рис. 2.11. Компонент «Capsule Collider»

Створений користувачем компонент може керувати властивостями інших компонентів, додавати нові, змінювати ігрову сцену та використовувати всі можливості мови C#. Лістинг всіх користувацьких скриптів знаходиться у додатку А.

2.4.3. Реалізація ігрових елементів

Дизайн гри створений у стилі pixel-art [10], що набирає чималої популярності у сучасній ігровій індустрії. Серед використаних засобів обрамлення містяться:

- привабливі декорації жанру sci-fi;
- цікаві та стилізовані персонажі.

Зовнішній вигляд інтерфейсу тісно пов’язаний із самою грою та сутністю ігрового процесу. В основному він асоціюється з графікою відеоігор 80-х та 90-х років. Тоді художникам доводилося враховувати обмеження пам'яті та низького дозволу. Персонажі, що оточують гравця, створені та стилізовані відповідно до жанру. Приклади персонажів наведено на рис. 2.12.

Рис. 2.12. Персонажі гри

Саме ігрове поле, що буде знаходитись перед гравцем більшу частину ігрового процесу, також стилізоване під заданий стиль. Ліворуч та праворуч від поля наявні панелі з основним функціоналом. Усе графічне оформлення було створене спеціально для гри.

Для локації було створено наступні об’єкти і компоненти (табл. 2.1). Суть цієї локації полягає в тому, щоб гравець на ній воював з ворогами. Вигляд локації відображено на рис. 2.13.

Таблиця 2.1

Компоненти сцени

Об’єкт	Компонент	Призначення
Земля	Sprite Renderer	Візуальна картинка
	Box Collider 2D	Колізія
	Script (Ground)	Контроль усіма об’єктами у домі
Фон	Sprite Renderer	Візуальна картинка
	Script (Parallax)	Контроль взаємодії з гравцем
Coor X		Позиція по координаті X
Coor Y	Transform	Позиція по координаті Y
Coor Z		Позиція по координаті Z

Рис. 2.13. Концепт сцени

Для елементу камера був створений скрипт «SlowCamera». Суть камери полягає у тому, щоб постійно слідкувати за головним героєм,

відображати ігрову сцену і зміни які на ній відбуваються. Скрипт «SlowCamera» дає можливість плавного стеження за гравцем, обмеження за вказаними координатами щоб камера не вийшла за умовні межі та зміна розміру камери для повного захоплення різних об'єктів у полі зору гравця. Камера та її кордони на сцені відображені на рис. 2.14.

Рис. 2.14. Камера на сцені

Для головного героя було створено наступні об’єкти і компоненти (табл. 2.2). Вигляд і дія головного героя відображено на рис. 2.15.

Таблиця 2.2

Компоненти головного героя

Об’єкт	Компонент	Призначення
Player	Sprite Renderer	Візуальна картинка
	Capsule Collider 2D	Колізія
	Trigger	Колізія з об'єктами
	Script (Player)	Контроль головним героєм
	Shadow	Тінь
	PlayerHighlight	Підсвічування
	Rigidbody 2D	Контроль фізики
	Animator	Контроллер анімаціями

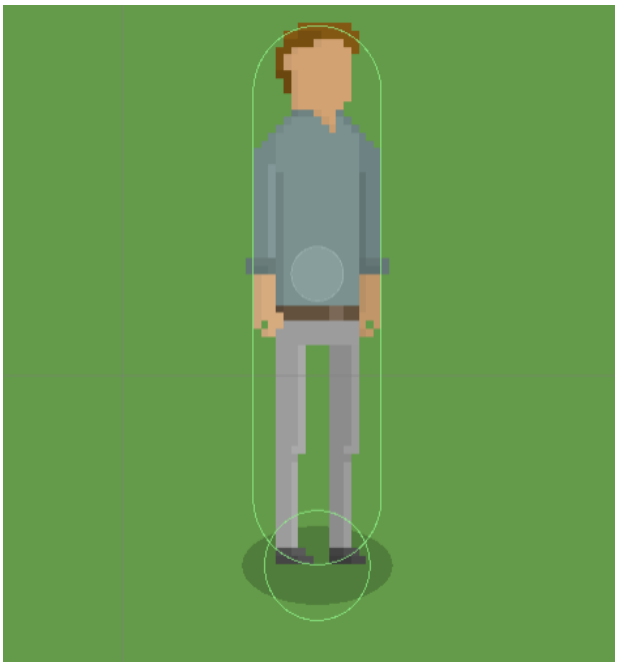

Рис. 2.15. Концепт головного героя

Скрипт «Player» призначений для переміщення головного героя гравцем, взаємодію з предметами та контролю всіх характеристик гравця.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Згідно з постановкою завдання у якості вхідних даних програми виступають:

- переміщення персонажу;
- взаємодія з різними об'єктами;
- налаштування гри.

Вихідними даними є:

- відображення положення головного героя;
- програвання звуків;
- відображення анімацій;
- вивід результату гри.

2.6. Опис роботи розробленого програмного продукту

2.6.1. Використані технічні засоби

Для розробки даної гри використовувалися наступні технічні засоби:

- Процесор: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz;
- Відеокарта: Nvidia GeForce 1080 Ti;
- Оперативна пам’ять: 8 гб;
- ОС: Windows 10.

2.6.2. Використані програмні засоби

Реалізація виконана на платформі Unity, мова програмування для написання скриптів та ігрової логіки – C#. Об’єктно-орієнтована концепція мови програмування C# найліпше підходить для описання ігрової логіки об’єктів за допомогою системи класів.

Вибране програмне забезпечення Photoshop [6] підійшло для створення двовимірних об’єктів.

2.6.3. Виклик та завантаження програми

На початку роботи з грою, необхідно завантажити її у вигляді rar/zip архіву. Після завантаження файлу, необхідно розпакувати весь його зміст в комфортне для користувача місце в документах пристрою. Після розпакування архіву, відкрити .exe файл (рис. 2.16). Після цього відбудеться запуск гри. Для андроїд [9] потрібно лише встановити на смартфон файл .apk та встановити гру (рис. 2.17).

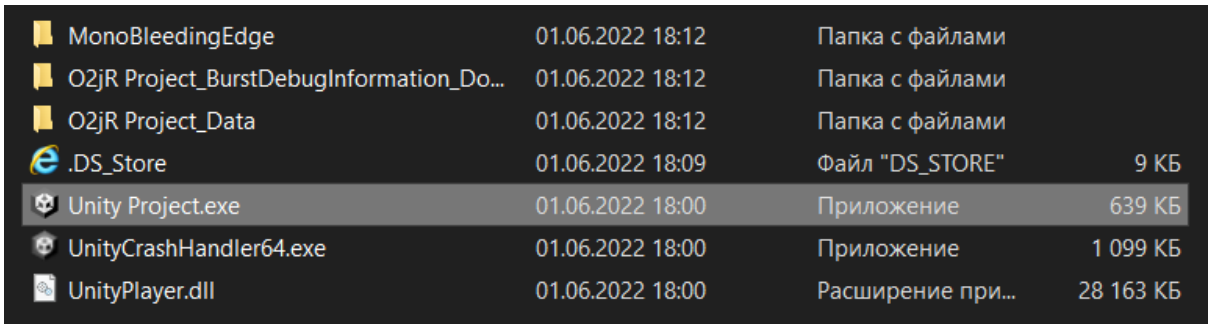

Рис. 2.16. Встановлення гри на пк

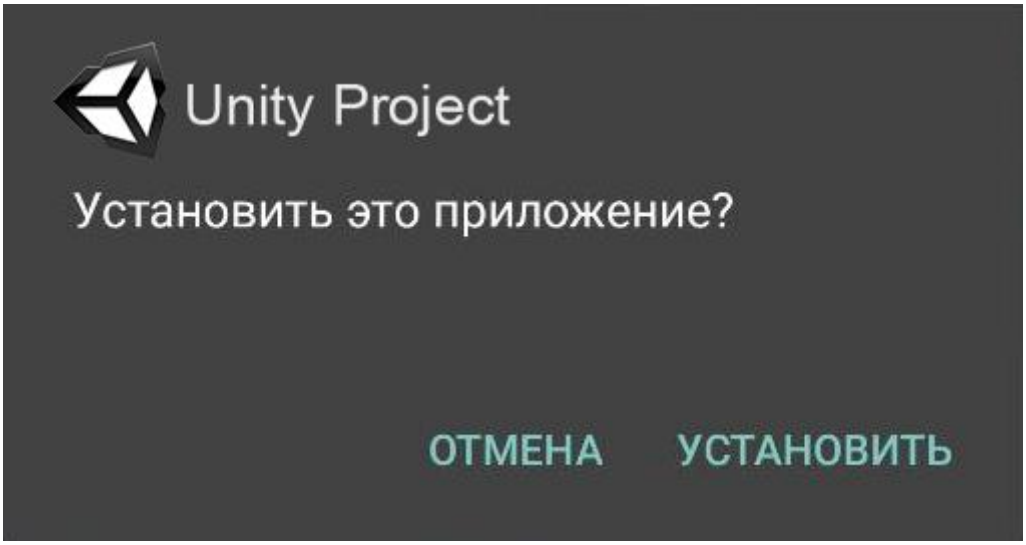

Рис. 2.17. Встановлення гри на андроїд

2.7. Опис інтерфейсу користувача

Після запуску гри, перед користувачем з’являється Головне меню (рис. 2.18), яке представлене декількома кнопками взаємодії з проектом: грати, додаткові опції та вихід.

Рис. 2.18. Головне меню

При виборі кнопки Options користувач потрапляє в вікно редагування графіки та звуку. Усі зміни зберігаються автоматично методом Player Prefs [4] (рис. 2.19).

Рис. 2.19. Вікно налаштування музики та графіки

При завантаженні головної сцени, за допомогою написаних скриптів, виконується скрипт віддалення камери від сцени. Камера стабілізується на головному герої (рис. 2.20). Завантаження всіх об'єктів на сцені відбувалося в перші секунди після запуску гри під час відтворення анімації згасання екрану.

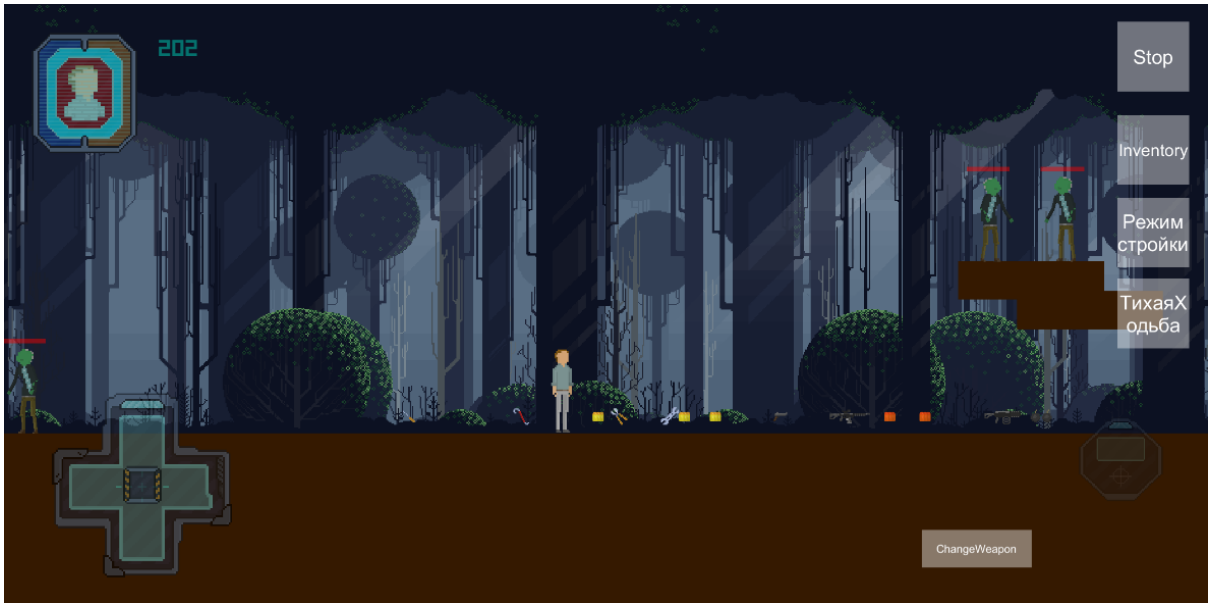

Рис. 2.20. Положення камери після завантаження сцени

На сцені у лівому верхньому куті разташована панель з характеристиками головного героя (рис. 2.21) Ця панель містить у собі такі показники: кількість життів, витривалість головного героя, шкала спраги, шкала голоду.

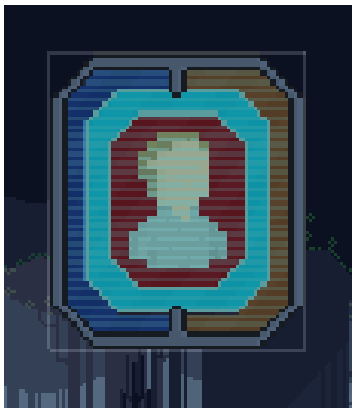

Рис. 2.21. Панель з характеристиками головного героя

У правому верхньому куті розташовані кнопки: зупинити гру, відкрити інвентар, режим огляду локації та тиха ходьба. Натиснувши на кнопку - зупиниту гру, відкривається панель (рис. 2.22) із кнопками: продовжити гру, налаштування, вийти з гри.

Рис. 2.22. Панель паузи гри

При натисканні на кнопку інвентарю, відкривається панель з інструментами та предметами гравця (рис. 2.23). Усі об'єкти в інвентарі інтерактивні. Їх можна перетягувати та викидати із інвентарю.

Рис. 2.23. Панель інвентаря гравця

У нижній частині екрану, розташовуються елементи управління головним героєм. Зліва - джойстик управління ходьбою. Справа - приціл та кнопка зміни зброї в руках. При натисканні на кнопку прицілу гравець вистрілює (рис. 2.24) у бік напряму зброї за умови, що в руках у гравця є зброя, а в інвентарі є патрони для діючої зброї.

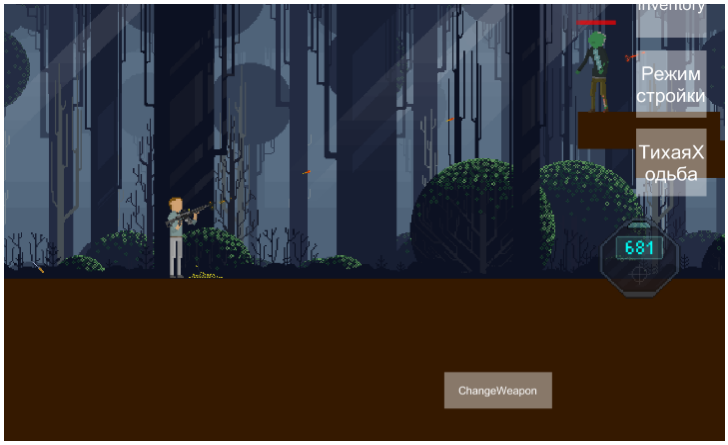

Рис. 2.24. Прицілювання та стрільба

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ 3.1.

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

- вихідні дані розробки програмного забезпечення:
- передбачуване число операторів – 860;
- коефіцієнт складності програми – 1,2;
- коефіцієнт кореляції програми в ході її розробки - 0,1;
- середня годинна заробітна плата програміста, грн/год – 40;
- вартість машино-години ЕОМ, грн/год – 5.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = to + ти + та + тп + тотл + тд \text{ , людино-годин,} \quad (3.1)$$

де to – витрати праці на підготовку й опис поставленої задачі (приймається 50);

ти – витрати праці на дослідження алгоритму рішення задачі;

та– витрати праці на розробку блок-схеми алгоритму;

тп – витрати праці на програмування по готовій блок-схемі;

тотл – витрати праці на налагодження програми на ЕОМ;

тд – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C(1 + p), \quad (3.2)$$

де q – передбачуване число операторів;

С– коефіцієнт складності програми;

р – коефіцієнт кореляції програми в ході її розробки.

$$Q = 860 * 1,2 * (1 + 0,1) = 1135;$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин}, \quad (3.3)$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В = 1.2 ... 1.5;

К – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. до 2 – 0,8;

$$t_u = \frac{1135 \cdot 1,2}{85 \cdot 0,8} = 20, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}; \quad (3.4)$$

$$t_a = \frac{1135}{25 \cdot 0,8} = 57, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20...25)K}; \quad (3.5)$$

$$t_a = \frac{1135}{25 \cdot 0,8} = 57, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K};$$

$$t_{отл} = \frac{1135}{5 \cdot 0{,}8} = 284, \text{ людино-годин,} \quad (3.6)$$

- за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл};$$

$$t^{к}_{отл} = 1{,}2 \cdot 284 = 340, \text{ людино-годин} \quad (3.7)$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до}; \quad (3.8)$$

де tдр – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K}; \quad (3.9)$$

$$t_{др} = \frac{1135}{20 \cdot 0{,}8} = 71, \text{ людино-годин.}$$

tдо – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}; \quad (3.10)$$

$$t_{до} = 0{,}75 \cdot 71 = 53, \text{ людино-годин.}$$

$$t_{д} = 71 + 53 = 124, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 +20 + 57 +57 + 284 +124 = 592, людино-годин.

В результаті розраховано, що в загальній складності необхідно 592 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн}, \tag{3.11}$$

де ЗЗП – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн}, \tag{3.12}$$

де t – загальна трудомісткість, людино-годин; СПР – середня годинна заробітна плата програміста, грн/година

$$Ç_{ÇÏ} = 592 \cdot 40 = 23680, \text{грн.}$$

ЗМВ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн}, \tag{3.13}$$

де tотл – трудомісткість налагодження програми на ЕОМ, год. СМЧ – вартість машино-години ЕОМ, грн/год

$$З_{МВ} = 283 \cdot 5 = 1419, \text{ грн.}$$

$$\hat{E}_{Ï} = 23680 + 1419 = 25100, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.} \tag{3.14}$$

де Вк - число виконавців; Fр - місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{592}{1 \cdot 176} = 3{,}4 \text{ міс.}$$

Таким чином, очікувана тривалість розробки складе 3,4 місяця, а витрати на створення програмного забезпечення 25100 грн.

Висновок: практична цінність програми полягає в тому, що користувачі можуть використовувати систему для веселого та приємного відпочинку та тренування реакції. Вартість даного програмного забезпечення становить 25 100 грн. Для впровадження та роботи програми не потрібно додаткових витрат. Очікуваний період розробки становить приблизно 3,4 місяці. Термін пов'язаний з багатьма операторами і включає час на дослідження та розробку алгоритмів вирішення проблем, програмування завершених алгоритмів, налагодження програми та підготовку документації.

ВИСНОВКИ

Завданням кваліфікаційної роботи є проектування та розробка відеоігри в жанрі Action/RPG на платформі Unity 3D, метою якої є створення оптимальних умов роботи, що створюють задоволення людини від виконуваної роботи.

Актуальність розробленого проекту полягає в тому, щоб реалізувати Action/RPG гру на платформі Unity, а також надати функціональний продукт для використання в розважальних цілях.

Реалізація здійснюється на платформі Unity 3D, мові програмування для написання скриптів та ігрової логіки. Об'єктно-орієнтована концепція мови програмування C# найкраще використовувати для опису логіки гри за допомогою системи класів.

Головний герой рухається в двовимірному просторі і може рухатися вліво та вправо.

На полі бою головний герой повинен битися з різними ворогами, щоб отримати предмети, які гравець може використовувати для відновлення своїх показників (життя, витривалість, голод, спрага).

В кінці гри відображається панель перемоги/програшу і статистика часу гравця. Гра може бути призупинена в будь-який момент, щоб користувач міг налаштувати регулятори руху та гучності.

У головному меню є кнопка виходу та налаштування гри. Крім ознайомлення з правилами гри та особливостями програми, в головному меню є кнопка з інструкціями з підказками для гравців.

Практична цінність програми полягає в тому, що система може бути використана користувачем для веселого та приємного відпочинку, а також для тренування реакції гравця.

На основі проведених розрахунків з економічної сторони показано, що на розробку цього програмного забезпечення потрібно всього 592 людино-години, орієнтовний час розробки складе 3,4 місяця, а вартість створення цього програмного забезпечення складе 25 100 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Unity 3D [Електронний ресурс] // URL: https://docs.unity3d.com/ru/530/Manual/ (дата звернення 17.05.2022)
2. Документація C# Docs [Електронний ресурс] // URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата звернення 17.05.2022)
3. Жанр гри Hack'n'slash [Електронний ресурс] // URL: https://cubiq.ru/hack-and-slash/ (дата звернення 18.05.2022)
4. Збереження даних у Unity за допомогою Player Prefs [Електронний ресурс] // URL: https://docs.unity3d.com/ru/2019.4/ScriptReference/PlayerPrefs.html (дата звернення 26.05.2022)
5. Застосування алгоритму Дейкстри [Електронний ресурс] // URL: https://habr.com/ru/post/111361/ (дата звернення 01.06.2022)
6. Навчання роботи в Photoshop [Електронний ресурс] // URL: https://www.youtube.com/watch?v=KXOMHopxIl4 (дата звернення 01.06.2022)
7. Пояснення як робити управління 2D персонажем [Електронний ресурс] // URL: https://www.youtube.com/watch?v=tywt9tOubEY&ab_channel=Unity (дата звернення 18.05.2022)
8. Пояснення роботи алгоритму пошуку шляху юнітам [Електронний ресурс] // URL: https://www.red-gate.com/simple-talk/development/dotnet-development/pathfinding-unity-c/ (дата звернення 01.06.2022)
9. Пояснення технічних особливостей двигуну Unity 3D для компіляції проекту під андроїд платформу [Електронний ресурс] // URL: https://www.youtube.com/watch?v=0NA6UAT-cAo&ab_channel=loftblog (дата звернення 24.05.2022)
10. Стиль Pixel Art [Електронний ресурс] // URL: https://diskettelounge.com/statie/istoriya-pixel-art/ (дата звернення 13.05.2022)

11. Що таке Visual Studio [Електронний ресурс] // URL: https://wiki.archlinux.org/title/Visual_Studio_Code_(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9) (дата звернення 19.05.2022)
12. Що таке NPC у грі, та як його зробити у Unity [Електронний ресурс] // URL: https://hub.packtpub.com/how-to-create-non-player-characters-npc-with-unity-2018/ (дата звернення 03.06.2022)
13. C# IntelliSense [Електронний ресурс] // URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-csharp-intellisense?view=vs-2022 (дата звернення 02.06.2022)
14. Github [Електронний ресурс] // URL: https://github.com/microsoft/vscode (дата звернення 25.05.2022)
15. Joe Hocking Unity in Action 2015. - 47c
16. Mark J. Price C# 7 and .NET Core 2017. - 33c
17. Mark Reed C#: The Ultimate Advanced Guide To Master C# Programming 2022. - 138c
18. Paul Schroeder Visual Studio 2019 Tricks and Techniques 2021. - 54c
19. Ui/Ux Unity 3d [Електронний ресурс] // URL: https://blog.unity.com/technology/evolving-the-unity-editor-ux (дата звернення 01.06.2022)
20. Will Goldstone Unity Game Development Essentials 2009. - 177c

ДОДАТОК А. Код програми

Item.cs - Контроль предметів на сцені

```
using UnityEngine;
```

```
public class Item : MonoBehaviour
{
    public ItemsController itemsController;
    public ControllerScenes controllerScenes;
    public SavingDroppedItems savingDroppedItems;
    [Space]
    public Transform Player;
    public Sprite IconItemForBtn;
    [Space]
    public int IndexClass;
    public int MyIndex;
    public int AmountOfResourses = 1;
    public int LvLItem = 1;

    public int Index_TrownItem;
    [Space]
    public bool SingleItem;
    public bool ItemCanBeInHands;
    public bool Seeds;
    [Header("Взаимодействие с наркотиками")]
    public string PlayerParameter;
    public int PlayerParameterIncrease;
    public float HowLongWillItTakeWork;
    [Space]

    bool canPickUp = false;
    int i = 0;

    void Start()
    {
        Invoke("CanPickUp", 1);

        if (controllerScenes.SceneDownload)
        {
            for (int i = 1; i < savingDroppedItems.Index_ThrownItems.Length; i++)
            {
                if (savingDroppedItems.Index_ThrownItems[i] == 0)
                {
                    savingDroppedItems.Index_ThrownItems[i] = MyIndex;
                    savingDroppedItems.AmountOfResources_ThrownItems[i] = AmountOfResourses;
                    savingDroppedItems.LvL_ThrownItems[i] = LvLItem;

                    savingDroppedItems.CoorX_ThrownItems[i] = transform.position.x;
                    savingDroppedItems.CoorY_ThrownItems[i] = transform.position.y;

                    PlayerPrefs.SetInt(controllerScenes.IndexThisScene + "Key_Index_ThrownItems" + i, MyIndex);
                    PlayerPrefs.SetInt(controllerScenes.IndexThisScene + "Key_AmountOfResources_ThrownItems" +
i, AmountOfResourses);
                    PlayerPrefs.SetInt(controllerScenes.IndexThisScene + "Key_LvL_ThrownItems" + i, LvLItem);

                    PlayerPrefs.SetFloat(controllerScenes.IndexThisScene + "Key_CoorX_ThrownItems" + i,
transform.position.x);
```

```
                PlayerPrefs.SetFloat(controllerScenes.IndexThisScene + "Key_CoorY_ThrownItems" + i,
transform.position.y);

                Index_TrownItem = i;

                break;
            }
            else if (i == savingDroppedItems.Index_ThrownItems.Length - 1)
            {
                Debug.Log("Новые выкинутые предметы не будут сохраняться");

                break;
            }
        }
    }
}

public void SpawnNearPlayer()
{
    transform.localPosition = new Vector3(Player.transform.position.x + Random.Range(-0.3f,0.3f),
Player.transform.position.y - 0.35f + Random.Range(-0.3f, 0.3f), 0);
}

void CanPickUp()
{
    canPickUp = true;
}

private void OnTriggerEnter2D(Collider2D other)
{
    if(other.tag == "Player")
    {
        if(canPickUp)
        {
            canPickUp = false;
            Act1();
        }
    }
}

void Act1()
{
    if (!SingleItem)
    {
        for (i = 0; i < 24; i++)
        {
            if (itemsController.IndexItemOnCell[i] == MyIndex)
            {
                itemsController.AmountOfResourcesOnCell[i] = AmountOfResourses +
itemsController.AmountOfResourcesOnCell[i];

                Act3();
```

```
                break;
            }
            else if (i == 23)
            {
                Act2();
                break;
            }
        }
    }
    else
    {
        Act2();
    }
}

void Act2()
{
    i = 0;

    for (i = 0; i < 24; i++)
    {
        if (itemsController.IndexItemOnCell[i] == 0)
        {
            itemsController.AmountOfResourcesOnCell[i] = AmountOfResourses;

            Act3();
            break;
        }
        else if (i == 23)
        {
            i = 0;
            canPickUp = true;
            break;
        }
    }
}

void Act3()
{
    itemsController.IndexItemOnCell[i] = MyIndex;

    itemsController.LvLItemOnCell[i] = LvLItem;
    itemsController.IndexCell_NewItem = i;
    itemsController.Index_NewItem = MyIndex;
    itemsController.ItemAddedOnInventory();

    InvokeRepeating("Attraction", 0.01f, 0.01f);
    //Sprite.GetComponent<Animator>().Play("Decrease");
    Invoke("Destroy", 0.4f);
}

void Attraction()
```

```
    {
        transform.position = Vector3.Lerp(transform.position, Player.position, 10 * Time.deltaTime);
    }

    void Destroy()
    {
        savingDroppedItems.DeletingValues(Index_TrownItem);
        Destroy(gameObject);
    }
}
```

InventorySlot.cs - Контроль блоків інвентарю

```
using UnityEngine;
using UnityEngine.UI;

public class InventorySlot : MonoBehaviour
{
    public ItemsController itemsController;
    [Space]
    public Image image;
    public Text text;
    [Space]
    public Color[] color;
    public Sprite[] AddCell_Sprites;
    [Space]
    public int Index_Cell;
    [Space]
    public int PouchFullness = 0;
    public int[] AddCells_IndexItems;
    public int[] AddCells_AmountOfResourcesItems;
    public int[] AddCells_LvLItems;


    void Awake()
    {
        if(itemsController.IndexItemOnCell[Index_Cell] == 300)
        {
            for (int i = 0; i < 9; i++)
            {
                AddCells_IndexItems[i] = PlayerPrefs.GetInt(Index_Cell + "Key_AddCells_IndexItems" + i, 0);
                AddCells_AmountOfResourcesItems[i] = PlayerPrefs.GetInt(Index_Cell +
"Key_AddCells_AmountOfResourcesItems" + i, 0);
                AddCells_LvLItems[i] = PlayerPrefs.GetInt(Index_Cell + "Key_AddCells_LvLItems" + i, 0);
            }

            CheckPouchFullness();
        }
    }
```

```
    public void InfoItem()
    {
        if(itemsController.IndexItemOnCell[Index_Cell] > 0)
        {
            itemsController.IndexActiveCell = Index_Cell;
            //itemsController.OutlineBtn.transform.localPosition = new Vector3(transform.localPosition.x,
transform.localPosition.y, 0);

            if (itemsController.IndexItemOnCell[Index_Cell] == 300 && Index_Cell < 24 || Index_Cell > 32)
            {
                itemsController.CloseAdditionalSlot();
                itemsController.OpenAdditionalSlots();
            }
            else
            {
                if (Index_Cell < 24 || Index_Cell > 32)
                {
                    itemsController.CloseAdditionalSlot();
                }
            }
        }

        if (itemsController.IndexItemOnCell[Index_Cell] == 1600)
        {

        }
    }

    //-------------------------------------------------------------------------------------------------------------------
    public void Down()
    {
        InvokeRepeating("CheckMousePos", 0.12f, 0.02f);

        image.color = color[1];
    }

    public void Up()
    {
        itemsController.ItemToChange1 = Index_Cell;

        if (!itemsController.ThrowItem)
        {
            itemsController.SwapObjects();
        }
        else
        {
            itemsController.ThrowItemFromInventory();
        }

        CancelInvoke("CheckMousePos");

        image.color = color[0];
```

```
        image.transform.localPosition = new Vector3(0,0,0);
        image.transform.localScale = new Vector2(1f, 1f);
    }

    public void Enter()
    {
        itemsController.ItemToChange2 = Index_Cell;

        if (itemsController.IndexItemOnCell[Index_Cell] > 0)
        {
            itemsController.PanelInfoItem.SetActive(true);

            InvokeRepeating("EnterAct2", 0f, 0.02f);
        }
    }

    void EnterAct2()
    {
        //itemsController.PanelInfoItem.transform.localPosition = new Vector3(transform.localPosition.x + 94.5f,
transform.localPosition.y - 47f, 0);
        //Vector3 newPosition = new Vector3(Input.mousePosition.x + 170, Input.mousePosition.y - 120f, 10.0f);
        //itemsController.PanelInfoItem.transform.position = Camera.main.ScreenToWorldPoint(newPosition);
    }

    public void Exit()
    {
        itemsController.PanelInfoItem.SetActive(false);
        CancelInvoke("EnterAct2");
    }

    void CheckMousePos()
    {
        Vector3 newPosition = new Vector3(Input.mousePosition.x, Input.mousePosition.y, 10.0f);
        image.transform.position = Camera.main.ScreenToWorldPoint(newPosition);
        image.transform.localScale = new Vector2(0.75f, 0.75f);
    }
    //-------------------------------------------------------------------------------------------------------------------
    public void ActivateThisSlot()
    {
        if(itemsController.LvLItemOnCell[itemsController.IndexActiveCell] > Index_Cell - 26)
        {
            GetComponent<Image>().sprite = AddCell_Sprites[0];
            GetComponent<Image>().raycastTarget = true;
        }
        else
        {
            GetComponent<Image>().sprite = AddCell_Sprites[2];
            GetComponent<Image>().raycastTarget = false;
        }
    }
    public void DeactivateThisSlot() //
    {
```

```
        GetComponent<Image>().sprite = AddCell_Sprites[1];
        GetComponent<Image>().raycastTarget = false;
    }

    public void CheckPouchFullness()
    {
        PouchFullness = 0;

        for (int i = 0; i < 9; i++)
        {
            if (AddCells_IndexItems[i] > 0)
            {
                PouchFullness++;
            }

            int T = itemsController.LvLItemOnCell[Index_Cell] + 2;
            text.text = PouchFullness.ToString() + "/" + T.ToString();
        }
    }

    public void SaveAddCells()
    {
        for(int i = 0; i < 9; i++)
        {
            PlayerPrefs.SetInt(Index_Cell + "Key_AddCells_IndexItems" + i, AddCells_IndexItems[i]);
            PlayerPrefs.SetInt(Index_Cell + "Key_AddCells_AmountOfResourcesItems" + i,
AddCells_AmountOfResourcesItems[i]);
            PlayerPrefs.SetInt(Index_Cell + "Key_AddCells_LvLItems" + i, AddCells_LvLItems[i]);
        }
    }
}
```

ItemsController.cs - Контроль інвентарю

```
using UnityEngine;
using UnityEngine.UI;

public class ItemsController : MonoBehaviour
{

    public PlayerController playerController;
    [Space]
    public InventorySlot[] inventorySlot;
    [Space]
    public Item[] item;
    public GameObject PanelInfoItem;
    public GameObject[] PrefabItems;
    public GameObject[] AdditionalPanels; // Дополнительные панели от обьектов в доме. Мусорки, Шкафы
и тд
    public Sprite NoneSprite;
```

```
    [Space]
    public int[] IndexItemOnCell; // Если 0 то нету предмета, если 1 или более то в ячейке какой то предмет
    public int[] AmountOfResourcesOnCell; // Количество ресурсов в каждой ячейке
    public int[] LvLItemOnCell; // Уровни каждого предмета в ячейке
    [Space]
    public int IndexCell_NewItem = 0; // Обозначение ячейки инвентаря в которую попадет новый подобранный предмет
    public int Index_NewItem = 0; // Обьект который попадет в ячейку инвентаря.
    [Space]
    public int ItemToChange1; // Индекс ячейки над которой провели в последний раз ячейку перед тем как отпустить мышь/палец тем самым поменяв местами две ячейки
    public int ItemToChange2; // Индекс ячейки который будут менять на ItemToChange1
    [Space]
    public bool ThrowItem; // Выкинуть обьект из инвентаря
    bool ItemToTrash; // Уничтожить предмет

    public int IndexActiveCell; // Индекс ячейки в которой находится обводка

    void Awake()
    {
        for (int i = 0; i < IndexItemOnCell.Length; i++)
        {
            if(i < 24 || i > 32)
            {
                IndexItemOnCell[i] = PlayerPrefs.GetInt("Key_IndexItemOnCell" + i, 0);

                if (IndexItemOnCell[i] != 0)
                {
                    AmountOfResourcesOnCell[i] = PlayerPrefs.GetInt("Key_AmountOfResourcesOnCell" + i, 0);
                    LvLItemOnCell[i] = PlayerPrefs.GetInt("Key_LvLItemOnCell" + i, 0);

                    inventorySlot[i].image.GetComponent<Image>().sprite = item[IndexItemOnCell[i]].IconItemForBtn;

                    if (item[IndexItemOnCell[i]].SingleItem && item[IndexItemOnCell[i]].MyIndex != 300) // Если предмет единичный и не с индексом 300 (мешочек)
                    {
                        inventorySlot[i].text.text = null;
                    }
                    else
                    {
                        if (item[IndexItemOnCell[i]].MyIndex == 300)
                        {
                            int T = LvLItemOnCell[i] + 2;
                            inventorySlot[i].text.text = inventorySlot[i].PouchFullness.ToString() + "/" + T.ToString();
                        }
                        else
                        {
                            inventorySlot[i].text.text = AmountOfResourcesOnCell[i].ToString();
                        }
                    }
                }
```

```
            }
        }

        playerController.IndexItemInHand = IndexItemOnCell[41];
        playerController.IndexItemBehindBack = IndexItemOnCell[42];
    }

    public void ItemAddedOnInventory() // Новый предмет в инвентаре
    {
        inventorySlot[IndexCell_NewItem].image.GetComponent<Image>().sprite =
item[Index_NewItem].IconItemForBtn;

        if (item[Index_NewItem].SingleItem && item[Index_NewItem].MyIndex != 300) // Если предмет
единичный и не с индексом 300 (мешочек)
        {
            inventorySlot[IndexCell_NewItem].text.text = null;
        }
        else
        {
            if(item[Index_NewItem].MyIndex == 300)
            {
                int T = LvLItemOnCell[IndexCell_NewItem] + 2;
                inventorySlot[IndexCell_NewItem].text.text =
inventorySlot[IndexCell_NewItem].PouchFullness.ToString() + "/" + T.ToString();
            }
            else
            {
                inventorySlot[IndexCell_NewItem].text.text =
AmountOfResourcesOnCell[IndexCell_NewItem].ToString();
            }
        }

        PlayerPrefs.SetInt("Key_IndexItemOnCell" + IndexCell_NewItem, Index_NewItem);
        PlayerPrefs.SetInt("Key_AmountOfResourcesOnCell" + IndexCell_NewItem,
AmountOfResourcesOnCell[IndexCell_NewItem]);
        PlayerPrefs.SetInt("Key_LvLItemOnCell" + IndexCell_NewItem, LvLItemOnCell[IndexCell_NewItem]);
    }

    public void SwapObjects() // Метод вызывается только из скрипта InventorySlot при смене местами
ячеек
    {
        if (ItemToChange2 >= 41 && ItemToChange2 <= 42)
        {
            if (item[IndexItemOnCell[ItemToChange1]].ItemCanBeInHands)
            {
                SwapObjectsAct2();
            }
        }
        else if (ItemToChange2 >= 24 && ItemToChange2 <= 32)
        {
            if(IndexItemOnCell[ItemToChange1] != 300) // Проверка: Перекидываем ли мы в дополнительные
ячейки мешочек
```

```
        {
            SwapObjectsAct2();
        }
    }
    else
    {
        if (IndexItemOnCell[IndexActiveCell] == 300 && ItemToChange1 == IndexActiveCell ||
ItemToChange2 == IndexActiveCell)
        {
            CloseAdditionalSlot();
        }

        SwapObjectsAct2();
    }
}

void SwapObjectsAct2()
{
    if (IndexItemOnCell[ItemToChange1] != 0 && ItemToChange1 != ItemToChange2)
    {
        if (!item[IndexItemOnCell[ItemToChange1]].SingleItem && IndexItemOnCell[ItemToChange1] ==
IndexItemOnCell[ItemToChange2]) // Проверка перед тем как буду соединять одинаковые предметы если
они не соединились как только попали в инвентарь
        {
            AmountOfResourcesOnCell[ItemToChange1] = AmountOfResourcesOnCell[ItemToChange1] +
AmountOfResourcesOnCell[ItemToChange2];

            IndexItemOnCell[ItemToChange2] = 0;
            AmountOfResourcesOnCell[ItemToChange2] = 0;
            LvLItemOnCell[ItemToChange2] = 0;
        }

        int T1 = IndexItemOnCell[ItemToChange1];            IndexItemOnCell[ItemToChange1] =
IndexItemOnCell[ItemToChange2];                IndexItemOnCell[ItemToChange2] = T1;
        int T2 = AmountOfResourcesOnCell[ItemToChange1];
AmountOfResourcesOnCell[ItemToChange1] = AmountOfResourcesOnCell[ItemToChange2];
AmountOfResourcesOnCell[ItemToChange2] = T2;
        int T3 = LvLItemOnCell[ItemToChange1];              LvLItemOnCell[ItemToChange1] =
LvLItemOnCell[ItemToChange2];                  LvLItemOnCell[ItemToChange2] = T3;

        if (IndexItemOnCell[ItemToChange1] == 300 || IndexItemOnCell[ItemToChange2] == 300)
        {
            int PouchFullness = inventorySlot[ItemToChange1].PouchFullness;
inventorySlot[ItemToChange1].PouchFullness = inventorySlot[ItemToChange2].PouchFullness;
inventorySlot[ItemToChange2].PouchFullness = PouchFullness;

            int[] AddCells_IndexItems = new int[9];
            int[] AddCells_AmountOfResourcesItems = new int[9];
            int[] AddCells_LvLItems = new int[9];

            for (int i = 0; i < 9; i++)
            {
```

```
                AddCells_IndexItems[i] = inventorySlot[ItemToChange1].AddCells_IndexItems[i];
                AddCells_AmountOfResourcesItems[i] =
inventorySlot[ItemToChange1].AddCells_AmountOfResourcesItems[i];
                AddCells_LvLItems[i] = inventorySlot[ItemToChange1].AddCells_LvLItems[i];

                inventorySlot[ItemToChange1].AddCells_IndexItems[i] =
inventorySlot[ItemToChange2].AddCells_IndexItems[i];
                inventorySlot[ItemToChange1].AddCells_AmountOfResourcesItems[i] =
inventorySlot[ItemToChange2].AddCells_AmountOfResourcesItems[i];
                inventorySlot[ItemToChange1].AddCells_LvLItems[i] =
inventorySlot[ItemToChange2].AddCells_LvLItems[i];

                inventorySlot[ItemToChange2].AddCells_IndexItems[i] = AddCells_IndexItems[i];
                inventorySlot[ItemToChange2].AddCells_AmountOfResourcesItems[i] =
AddCells_AmountOfResourcesItems[i];
                inventorySlot[ItemToChange2].AddCells_LvLItems[i] = AddCells_LvLItems[i];
            }
        }

        if (IndexItemOnCell[IndexActiveCell] == 300 && ItemToChange2 >= 24 && ItemToChange2 <= 32) //
Перемещение обьектов в дополнительные ячейки при условии что открыт мешочек
        {
            inventorySlot[IndexActiveCell].AddCells_IndexItems[ItemToChange2 - 24] =
IndexItemOnCell[ItemToChange2];
            inventorySlot[IndexActiveCell].AddCells_AmountOfResourcesItems[ItemToChange2 - 24] =
AmountOfResourcesOnCell[ItemToChange2];
            inventorySlot[IndexActiveCell].AddCells_LvLItems[ItemToChange2 - 24] =
LvLItemOnCell[ItemToChange2];

            inventorySlot[IndexActiveCell].CheckPouchFullness();
            inventorySlot[IndexActiveCell].SaveAddCells();
        }

        if (IndexItemOnCell[IndexActiveCell] == 300 && ItemToChange1 >= 24 && ItemToChange1 <= 32) //
Перемещение обьектов в дополнительные ячейки при условии что открыт мешочек
        {
            inventorySlot[IndexActiveCell].AddCells_IndexItems[ItemToChange1 - 24] =
IndexItemOnCell[ItemToChange1];
            inventorySlot[IndexActiveCell].AddCells_AmountOfResourcesItems[ItemToChange1 - 24] =
AmountOfResourcesOnCell[ItemToChange1];
            inventorySlot[IndexActiveCell].AddCells_LvLItems[ItemToChange1 - 24] =
LvLItemOnCell[ItemToChange1];

            inventorySlot[IndexActiveCell].CheckPouchFullness();
            inventorySlot[IndexActiveCell].SaveAddCells();
        }

        if (IndexItemOnCell[ItemToChange1] > 0)
        {
            inventorySlot[ItemToChange1].image.GetComponent<Image>().sprite =
item[IndexItemOnCell[ItemToChange1]].IconItemForBtn;
        }
```

```
        else
        {
            inventorySlot[ItemToChange1].image.GetComponent<Image>().sprite = NoneSprite;
        }
        if (IndexItemOnCell[ItemToChange2] > 0)
        {
            inventorySlot[ItemToChange2].image.GetComponent<Image>().sprite =
item[IndexItemOnCell[ItemToChange2]].IconItemForBtn;
        }
        else
        {
            inventorySlot[ItemToChange2].image.GetComponent<Image>().sprite = NoneSprite;
        }

        if (IndexItemOnCell[ItemToChange1] == 0 || item[IndexItemOnCell[ItemToChange1]].SingleItem &&
item[IndexItemOnCell[ItemToChange1]].MyIndex != 300)
        {
            inventorySlot[ItemToChange1].text.text = null;
        }
        else
        {
            if (item[IndexItemOnCell[ItemToChange1]].MyIndex == 300)
            {
                int T = LvLItemOnCell[ItemToChange1] + 2;
                inventorySlot[ItemToChange1].text.text =
inventorySlot[ItemToChange1].PouchFullness.ToString() + "/" + T.ToString();
            }
            else
            {
                inventorySlot[ItemToChange1].text.text =
AmountOfResourcesOnCell[ItemToChange1].ToString();
            }
        }
        if (IndexItemOnCell[ItemToChange2] == 0 || item[IndexItemOnCell[ItemToChange2]].SingleItem &&
item[IndexItemOnCell[ItemToChange2]].MyIndex != 300)
        {
            inventorySlot[ItemToChange2].text.text = null;
        }
        else
        {
            if (item[IndexItemOnCell[ItemToChange2]].MyIndex == 300)
            {
                int T = LvLItemOnCell[ItemToChange2] + 2;
                inventorySlot[ItemToChange2].text.text =
inventorySlot[ItemToChange2].PouchFullness.ToString() + "/" + T.ToString();
            }
            else
            {
                inventorySlot[ItemToChange2].text.text =
AmountOfResourcesOnCell[ItemToChange2].ToString();
            }
        }
```

```
        }

        if (ItemToChange2 >= 41 && ItemToChange2 <= 42 || ItemToChange1 >= 41 && ItemToChange1 <= 42)
        {
            playerController.IndexItemInHand = IndexItemOnCell[41];
            playerController.IndexItemBehindBack = IndexItemOnCell[42];
        }

        PlayerPrefs.SetInt("Key_IndexItemOnCell" + ItemToChange1, IndexItemOnCell[ItemToChange1]);
        PlayerPrefs.SetInt("Key_AmountOfResourcesOnCell" + ItemToChange1,
AmountOfResourcesOnCell[ItemToChange1]);
        PlayerPrefs.SetInt("Key_LvLItemOnCell" + ItemToChange1, LvLItemOnCell[ItemToChange1]);

        PlayerPrefs.SetInt("Key_IndexItemOnCell" + ItemToChange2, IndexItemOnCell[ItemToChange2]);
        PlayerPrefs.SetInt("Key_AmountOfResourcesOnCell" + ItemToChange2,
AmountOfResourcesOnCell[ItemToChange2]);
        PlayerPrefs.SetInt("Key_LvLItemOnCell" + ItemToChange2, LvLItemOnCell[ItemToChange2]);
    }

    public void ThrowItemFromInventory() // Метод вызывается только из скрипта InventorySlot для того
что бы выкинуть предмет
    {
        // А еще нужно добавить мусорку для предметов

        if (IndexItemOnCell[ItemToChange1] == 300)
        {
            for (int i = 0; i < 9; i++)
            {
                if (inventorySlot[ItemToChange1].AddCells_IndexItems[i] != 0)
                {
                    GameObject New_Item =
Instantiate(PrefabItems[inventorySlot[ItemToChange1].AddCells_IndexItems[i]]) as GameObject; // Создание
выкинутого обьекта из префаба
                    New_Item.transform.SetParent(transform, false);
                    New_Item.GetComponent<Item>().AmountOfResourses =
inventorySlot[ItemToChange1].AddCells_AmountOfResourcesItems[i];
                    New_Item.GetComponent<Item>().LvLItem =
inventorySlot[ItemToChange1].AddCells_LvLItems[i];
                    New_Item.GetComponent<Item>().SpawnNearPlayer();

                    inventorySlot[ItemToChange1].AddCells_IndexItems[i] = 0;
                    inventorySlot[ItemToChange1].AddCells_AmountOfResourcesItems[i] = 0;
                    inventorySlot[ItemToChange1].AddCells_LvLItems[i] = 0;

                    inventorySlot[i + 24].image.GetComponent<Image>().sprite = NoneSprite;
                    inventorySlot[i + 24].text.text = null;

                    // Проверить выкидывание предметов на низком fps
                }
            }
        }
```

```
        GameObject NewItem = Instantiate(PrefabItems[IndexItemOnCell[ItemToChange1]]) as GameObject; // Создание выкинутого обьекта из префаба
        NewItem.transform.SetParent(transform, false);
        NewItem.GetComponent<Item>().AmountOfResourses = AmountOfResourcesOnCell[ItemToChange1];
        NewItem.GetComponent<Item>().LvLItem = LvLItemOnCell[ItemToChange1];
        NewItem.GetComponent<Item>().SpawnNearPlayer();

        if (IndexItemOnCell[ItemToChange1] == 300) // Закрываем доп. ячейки если выкинули открытый мешочек
        {
            CloseAdditionalSlot();

            inventorySlot[ItemToChange1].PouchFullness = 0; // Пересчет заполненности ячейки при его выкидывании
        }

        if (IndexItemOnCell[IndexActiveCell] == 300 && ItemToChange1 >= 24 && ItemToChange1 <= 32)
        {
            inventorySlot[IndexActiveCell].AddCells_IndexItems[ItemToChange1 - 24] = 0;
            inventorySlot[IndexActiveCell].AddCells_AmountOfResourcesItems[ItemToChange1 - 24] = 0;
            inventorySlot[IndexActiveCell].AddCells_LvLItems[ItemToChange1 - 24] = 0;

            inventorySlot[IndexActiveCell].CheckPouchFullness();
            inventorySlot[IndexActiveCell].SaveAddCells();
        }

        IndexItemOnCell[ItemToChange1] = 0;
        AmountOfResourcesOnCell[ItemToChange1] = 0;
        LvLItemOnCell[ItemToChange1] = 0;

        inventorySlot[ItemToChange1].image.GetComponent<Image>().sprite = NoneSprite;
        inventorySlot[ItemToChange1].text.text = null;

        PlayerPrefs.SetInt("Key_IndexItemOnCell" + ItemToChange1, IndexItemOnCell[ItemToChange1]);
        PlayerPrefs.SetInt("Key_AmountOfResourcesOnCell" + ItemToChange1, AmountOfResourcesOnCell[ItemToChange1]);
        PlayerPrefs.SetInt("Key_LvLItemOnCell" + ItemToChange1, LvLItemOnCell[ItemToChange1]);
    }
    //-------------------------------------------------------------------------------------------------------------------------------------------
    public void Enter() // Этот метод выполняется из обьекта BtnClosePanelInventory. Если перетащить предмет из инвентаря на эту панель, то предмет выпадет из инвентаря и будет лежать на полу
    {
        ThrowItem = true;
        // Я думаю что стоит высвечивать на обьекте который выкидываю иконку с обозначением что этот предмет выкидывают
    }
    public void Exit() // Этот метод выполняется из обьекта BtnClosePanelInventory.
    {
        ThrowItem = false;
    }
    public void ClosePanelInventory()
```

```
    {
        CloseAdditionalSlot();

        for(int i = 0; i < AdditionalPanels.Length; i++) // Закрытие дополнительных панелей от суорок, шкафов и тд
        {
            AdditionalPanels[i].SetActive(false);
        }
    }
    //-------------------------------------------------------------------------------------------------------------------------------------------------------
    public void OpenAdditionalSlots()
    {
        for(int i = 24; i < 33; i++)
        {
            inventorySlot[i].ActivateThisSlot();

            IndexItemOnCell[i] = inventorySlot[IndexActiveCell].AddCells_IndexItems[i-24];
            AmountOfResourcesOnCell[i] = inventorySlot[IndexActiveCell].AddCells_AmountOfResourcesItems[i - 24];
            LvLItemOnCell[i] = inventorySlot[IndexActiveCell].AddCells_LvLItems[i - 24];

            if (IndexItemOnCell[i] > 0)
            {
                inventorySlot[i].image.GetComponent<Image>().sprite = item[IndexItemOnCell[i]].IconItemForBtn;
            }
            else
            {
                inventorySlot[i].image.GetComponent<Image>().sprite = NoneSprite;
            }

            if (IndexItemOnCell[i] == 0 || item[IndexItemOnCell[i]].SingleItem)
            {
                inventorySlot[i].text.text = null;
            }
            else
            {
                inventorySlot[i].text.text = AmountOfResourcesOnCell[i].ToString();
            }
        }

        // Сделать текст красным если мешочек переполнен
    }
    public void CloseAdditionalSlot()
    {
        for (int i = 24; i < 33; i++)
        {
            inventorySlot[i].DeactivateThisSlot();

            IndexItemOnCell[i] = 0;
            AmountOfResourcesOnCell[i] = 0;
            LvLItemOnCell[i] = 0;
```

```
            inventorySlot[i].image.GetComponent<Image>().sprite = NoneSprite;

            inventorySlot[i].text.text = null;
        }
    }
}
```

Player.cs - Скрипт управління головним героєм

```
using UnityEngine;
using UnityEngine.UI;

public class Player : MonoBehaviour
{
    public PlayerController playerController;

    public bool Androind;
    public bool Pc;
    public bool Ios;

    public Joystick joystickWalk;
    public Joystick joystickAttack;

    Rigidbody2D rb;

    public float Speed = 1;
    float SpeedX;
    float SpeedY;

    bool playerHighlight = false;
    public GameObject PlayerHighlight;

    public GameObject Shadow;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();

        SpeedX = Speed;
        SpeedY = Speed;
    }

    void FixedUpdate()
    {
        if (Androind)
        {
            ControlForAndroid();
        }
        else if (Pc)
        {
```

```
            ControlForPc();
        }
        else if (Ios)
        {
            ControlForIos();
        }

        if (Physics2D.OverlapCircle(Shadow.transform.position, 0.1f, 1 << 9))
        {
            if (!playerHighlight)
            {
                playerHighlight = true;
                PlayerHighlight.SetActive(true);
            }
        }
        else
        {
            if (playerHighlight)
            {
                playerHighlight = false;
                PlayerHighlight.SetActive(false);
            }
        }

        transform.localPosition = new Vector3(transform.position.x, transform.position.y, transform.position.y / 6f
- 0.05f);
    }
    //------------------------------------------------------------------------------------------------------------------------
------------------
    void ControlForAndroid()
    {
        rb.velocity = new Vector2(joystickWalk.Horizontal * SpeedX, joystickWalk.Vertical * SpeedY);
    }

    void ControlForPc()
    {
        if (Input.GetKey(KeyCode.W))
        {
            SpeedY = Speed;
        }
        else if (Input.GetKey(KeyCode.S))
        {
            SpeedY = -Speed;
        }
        else
        {
            SpeedY = 0;
        }

        if (Input.GetKey(KeyCode.A))
        {
            SpeedX = -Speed;
```

```
        }
        else if (Input.GetKey(KeyCode.D))
        {
            SpeedX = Speed;
        }
        else
        {
            SpeedX = 0;
        }

        rb.velocity = new Vector2(SpeedX, SpeedY);
    }

    void ControlForIos()
    {

    }
}
```

Garden.cs - Контроль рослин у саду

```
using UnityEngine;
using UnityEngine.UI;

public class Garden : MonoBehaviour
{
    public PlayerController playerController;
    public ItemsController itemsController;
    public GameObject ItemsController_Object;

    public int IndexGarden; // Индекс грядки
    public int PlantedBeds; // засаженые грядки


    public int[] GrowingSeeds; // Семена которые растут в 4 грядках
    public bool[] CanBeCollected; // Грядка выросла и можно её собрать

    public Color[] BlueColor;

    public GameObject[] TestGardenBed;
    public Color[] ColorTestGardenBed;

    public bool CanInteract;

    void Start()
    {

    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.tag == "Player")
```

```
        {
            if (CanBeCollected[0] || CanBeCollected[1] || CanBeCollected[2] || CanBeCollected[3])
            {
                // Включается подсветка обьекта
                GetComponent<SpriteRenderer>().color = BlueColor[1];
                CanInteract = true;
            }
            else if (playerController.IndexItemInHand != 0 &&
itemsController.item[playerController.IndexItemInHand].Seeds && PlantedBeds < 4)
            {
                // Включается подсветка обьекта
                GetComponent<SpriteRenderer>().color = BlueColor[1];
                CanInteract = true;
            }
        }
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.gameObject.tag == "Player")
        {
            // Выключается подсветка обьекта
            GetComponent<SpriteRenderer>().color = BlueColor[0];
            CanInteract = false;
        }
    }

    void Update()
    {
        if (CanInteract)
        {
            if (Input.GetKeyDown(KeyCode.E))
            {
                if (CanBeCollected[0] || CanBeCollected[1] || CanBeCollected[2] || CanBeCollected[3])
                {
                    PlantHarvested();
                }
                else if (itemsController.AmountOfResourcesOnCell[41] > 0 && PlantedBeds < 4)
                {
                    Planting();
                }
            }
        }
    }

    void Planting() // Растение посажено
    {
        if (PlantedBeds < 4)
        {
            int r = Random.Range(0, 4);

            if (GrowingSeeds[r] == 0)
```

```
        {
            itemsController.AmountOfResourcesOnCell[41] -= 1;
            itemsController.inventorySlot[41].text.text =
itemsController.AmountOfResourcesOnCell[41].ToString();

            PlantedBeds++;
            TestGardenBed[r].SetActive(true);
            GrowingSeeds[r] = playerController.IndexItemInHand;

            Invoke("PlantInGardenHasGrown_" + r, 5);

            PlayerPrefs.SetInt("Key_AmountOfResourcesOnCell" + 41,
itemsController.AmountOfResourcesOnCell[41]);

            if (PlantedBeds == 4)
            {
                GetComponent<SpriteRenderer>().color = BlueColor[0];
                CanInteract = false;
            }

            if (itemsController.AmountOfResourcesOnCell[41] <= 0)
            {
                GetComponent<SpriteRenderer>().color = BlueColor[0];
                CanInteract = false;

                itemsController.IndexItemOnCell[41] = 0;
                itemsController.AmountOfResourcesOnCell[41] = 0;
                itemsController.LvLItemOnCell[41] = 0;

                itemsController.inventorySlot[41].image.GetComponent<Image>().sprite =
itemsController.NoneSprite;
                itemsController.inventorySlot[41].text.text = null;

                PlayerPrefs.SetInt("Key_IndexItemOnCell" + 41, itemsController.IndexItemOnCell[41]);
                PlayerPrefs.SetInt("Key_AmountOfResourcesOnCell" + 41,
itemsController.AmountOfResourcesOnCell[41]);
                PlayerPrefs.SetInt("Key_LvLItemOnCell" + 41, itemsController.LvLItemOnCell[41]);

                playerController.IndexItemInHand = 0;
            }
        }
        else
        {
            Planting();
        }
    }
}

void PlantHasGrown() // Растение выросло
{

}
```

```
    void PlantHarvested() // Растение собрано
    {
        for (int i = 0; i < 4; i++)
        {
            if (CanBeCollected[i])
            {
                TestGardenBed[i].GetComponent<SpriteRenderer>().color = ColorTestGardenBed[0];
                CanBeCollected[i] = false;
                TestGardenBed[i].SetActive(false);

                GameObject NewItem = Instantiate(itemsController.PrefabItems[GrowingSeeds[i] + 1000]) as
GameObject; // Создание выкинутого обьекта из префаба
                NewItem.transform.SetParent(ItemsController_Object.transform, false);
                NewItem.GetComponent<Item>().AmountOfResourses = 1;
                NewItem.GetComponent<Item>().LvLItem = 1;
                NewItem.GetComponent<Item>().SpawnNearPlayer();

                GrowingSeeds[i] = 0;
                PlantedBeds--;
            }
        }
    }


    void PlantInGardenHasGrown_0()
    {
        TestGardenBed[0].GetComponent<SpriteRenderer>().color = ColorTestGardenBed[1];
        CanBeCollected[0] = true;
    }

    void PlantInGardenHasGrown_1()
    {
        TestGardenBed[1].GetComponent<SpriteRenderer>().color = ColorTestGardenBed[1];
        CanBeCollected[1] = true;
    }

    void PlantInGardenHasGrown_2()
    {
        TestGardenBed[2].GetComponent<SpriteRenderer>().color = ColorTestGardenBed[1];
        CanBeCollected[2] = true;
    }

    void PlantInGardenHasGrown_3()
    {
        TestGardenBed[3].GetComponent<SpriteRenderer>().color = ColorTestGardenBed[1];
        CanBeCollected[3] = true;
    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

на кваліфікаційну роботу бакалавру

на тему:

“Розробка кросплатформеної комп’ютерної гри в середовищі Unity 3D”

студента групи 122-18-3 Шевченка Кирила Миколайовича

Керівник кваліфікаційної роботи
доцент, каф. ПЕП та ПУ, к.е.и **Л.В. Касьяненко**

ДОДАТОК В. Перелік файлів на диску

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_Шевченко.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Диплом_Шевченко.pdf	Пояснювальна записка до дипломного проекту в форматі PDF

Програма	
Project.rar	Архів. Містить код програми і откомпільовану програму
Презентація	
Презентація_Шевченко.pptx	Презентація дипломного проекту