

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Мобільна гра «Космонавт-золотошукач»»

Виконав студент групи ПЗ-20-2
СИДОРЧУК Олександр Сергійович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ДУНЄВ Сергій Сергійович

Житомир – 2024

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

«ЗАТВЕРДЖУЮ»

Зав. каф. інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СИДОРЧУК Олександр Сергійович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Мобільна гра «Космонавт-золотошукач»»,**

затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №74/с

Вихідні дані для роботи: **розробка мобільної гри про видобуток корисних
копалин та будівництво космічної бази на інших планетах**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М.	27.02.2024р.	27.02.2024р.
2	Єфремов Ю.М.	24.04.2024р.	24.04.2024р.
3	Єфремов Ю.М.	02.05.2024р.	02.05.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу розробленої мобільної гри про видобуток ресурсів на позаземних цивілізаціях та пояснювальної записки. Пояснювальна записка до дипломної роботи містить 83 сторінки, 43 ілюстрації та 7 таблиць. Головною метою є створення мобільної гри, яка дозволить користувачам провести з цікавістю вільний час офлайн, відчувши себе космонавтом, що досліджує позаземні світи у пошуках корисних копалин.

В роботі визначено ключові завдання, що пов'язані з розробкою ігрового додатку, проведено аналіз існуючих програм, подібних до розроблюваної, а також наведено сценарії роботи програмного продукту. Представлено огляд загальної структури додатку, описано алгоритми взаємодії між окремими компонентами, а також наведено опис об'єктної структури (діаграму класів) додатку.

У рамках випускної роботи були проведені експерименти по оптимізації геймплею, в результаті яких були визначені оптимальні налаштування для більшості існуючих девайсів.

КЛЮЧОВІ СЛОВА: ГРА, UNITY, МОБІЛЬНИЙ, СИСТЕМА, МЕХАНІКА, ГЕЙМПЛЕЙ, РУШІЙ, СКРИПТ, СИМПЛЕКС-МЕТОД.

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Сидорчук О.С.			Мобільна гра «Космонавт-золотошукач»	Літ.	Арк.	Аркушів
Керівник		Єфремов Ю.М.					3	100
Н. контр.					Пояснювальна записка	Житомирська політехніка, група ІПЗ-20-2		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification thesis consists of the software complex of the developed mobile game about the extraction of resources on extraterrestrial civilizations and an explanatory note. The explanatory note to the thesis contains 83 pages, 43 illustrations and 7 tables.

The main goal is to create a mobile game that will allow users to spend their free time offline with interest, feeling like an astronaut exploring extraterrestrial worlds in search of minerals.

The work defines the key tasks related to the development of a game application, analyzes existing programs similar to the one under development, and also provides scenarios for the operation of the software product. An overview of the general structure of the application is presented, the algorithms of interaction between individual components are described, and a description of the object structure (class diagram) of the application is also given.

As part of the final work, experiments were conducted to optimize the gameplay, as a result of which the optimal settings for most existing devices were determined.

KEYWORDS: GAME, UNITY, MOBILE, SYSTEM, MECHANICS, GAMEPLAY, ENGINE, SCRIPT, SIMPLEX METHOD

		Сидорчук О.С.			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ.....	8
1.1 Постановка задачі	8
1.2 Аналіз аналогів програмного продукту	9
1.3 Вибір архітектури системи	15
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	18
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНОЇ ГРИ «КОСМОНАВТ- ЗОЛОТОШУКАЧ»	21
2.1 Визначення варіантів використання та структури системи	21
2.2 Розробка алгоритмів та ігрових механік	29
2.3 Реалізація функціоналу додатку	37
Висновки до другого розділу	60
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ «КОСМОНАВТ-ЗОЛОТОШУКАЧ».....	61
3.1 Порядок встановлення та налаштування параметрів додатку	61
3.2 Структура інтерфейсу додатку та порядок роботи	62
3.3 Тестування роботи додатку.....	74
Висновки до третього розділу	76
ВИСНОВКИ.....	77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	79
ДОДАТКИ	82
Додаток А.....	83

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення;

ОС – операційна система;

UI – інтерфейс користувача;

ЦП – центральний процесор;

ОЗП – оперативна пам'ять;

AI – штучний інтелект;

FPS – кількість кадрів в секунду.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми полягає в історії розробки ігор, що має дуже давній початок - на сьогодні існує безліч жанрів, в наслідок чого це дозволило виокремити геймдев як окрему гілку мистецтва. Тим не менше, ігровий ринок щороку приносить чималі прибутки власникам, тому ця ніша все ще має величезні можливості для нових продуктів. Одним із найпопулярніших нині жанрів є мобільні відеоігри, що означає мінімум складних завдань для гравця, високий рівень залучення, а також мінімалістичний дизайн, що забезпечує кращу продуктивність на повільних пристроях. Саме тому наша гра також відноситиметься до цього напрямку.

Метою даної роботи є розробка програми гри та реалізація механік мобільних ігор. Встановлена мета обумовлює наступні завдання:

- проведення аналізу залучення гравців та їх подальшої зацікавленості у геймплеї;
- визначення архітектури та узагальненої структури системи;
- обґрунтування та вибір засобів реалізації системи;
- розробка інформаційного та математичного забезпечення системи.

Об'єктом дослідження є процес взаємодії користувача із грою для мобільних пристроїв.

Предметом дослідження є використання ключових механік та підходів, що практикуються у мобільних іграх та способи їх реалізації.

В роботі було використано методи об'єктно-орієнтованого програмування та дизайну інтерфейсів.

Розроблений додаток може бути використаний гравцями для розваги та проведення вільного часу з цікавістю, досліджуючи екзопланети та відкриваючи нові джерела видобутку ресурсів.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Механіки мобільних ігор представляють собою інноваційний підхід до розробки, спрямований на створення унікального та емоційно насиченого геймплею [1]. Мета цієї роботи полягає в розробці програми гри, яка використовує різноманітні сучасні механіки для створення неповторного досвіду гравця.

Необхідна реалізація таких функцій:

1) Аналіз проекту та збір вимог:

- Аналіз конкурентних додатків на ринку;
- Детальний опис ключових механік, з якими відбуватиметься взаємодія;
- Оцінювання майбутньої гри з точки зору гравця, що використовуватиме продукт;

2) Реалізація та впровадження системи:

- Реалізація локального сховища на базі ScriptableObjects;
- Створення моделей для зберігання у сховищі;
- Розробка системи шифрування та серіалізації даних;
- Створення і конфігурація зв'язків між головними сервісами системи;

3) Реалізація головної механіки гри:

- Створення модулів керування для гравця та ботів;
- Реалізація інвентарю та синхронного збереження добутих матеріалів у сховищі;
- Розробка системи взаємодії між юнітами та зонами керування, розташованими на поверхні землі;
- Створення системи поступового розблокування нових шахт та станцій;
- Додавання нових світів, доступних для дослідження;

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

- Реалізація щоденних місій з автопідбором необхідних завдань достатньої складності та відповідністю місії до поточного рівня розвитку гравця;
- Створення окремих міні-ігор, що доступні користувачу раз на день;
- Реалізація гнучкої системи досягнень гравця, збереження поточного результату та отримання винагороди за виконанні завдання

4) Тестування та забезпечення якості:

- Проведення ретельного тестування щоб гарантувати, що програмне рішення працює за призначенням і відповідає вимогам;
- Тестування адаптивності програми до різних режисів роздільної здатності, а також мінімальних потреб в обчислювальних потужностях пристроїв;

5) Реалізація бізнес-логіки:

- Додавання внутрішньо-ігрових покупок та продуктів для продажу;
- Імпорт та реалізація бібліотек мобільної реклами;

1.2 Аналіз аналогів програмного продукту

Нараз мобільні ігри користуються чималою популярністю через примітивність механік, простоту графіки та можливість грати без інтернету, через що багато студій зосередило свої сили саме на них. Було знайдено декілька популярних ігор, подібних по стилістиці до запланованої:

Stickmans of Wars: RPG Shooter

Подана гра є реалізацією шутера із спрощеним керуванням та легкістю в освоєнні. У нашій грі не планується система бою, однак буде запозичена така механіка, як збір ресурсів – достатньо підвести гравця на необхідну відстань і збір руди відбудеться автоматично [2]. Також Stickman Of Wars є чудовим прикладом застосування зон керування, на які необхідно стати гравцем для подальшої взаємодії з певним об'єктом чи явищем (рис. 1.1-1.2).

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Посилання на гру:

<https://play.google.com/store/apps/details?id=com.multicastgames.sow3>



Рис. 1.1-1.2. Stickmans of Wars: RPG Shooter

Переваги:

- Великий світ, прямолінійна сюжетна історія;
- Поступове прокачування персонажа, як наслідок відкриття нових локацій;
- Легко добувати ресурси, гравець після втрати життів нічого не губить і може продовжити знову.

Недоліки:

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

- Однотипність: незважаючи на різноманіття ворогів, гра швидко набридає, так як мало простору для власних ідей щодо дій у грі;
- Замкнутість: кожна локація представлена у вигляді невеликої кімнати із 4 стінами, тому гравець легко може передбачити наступну локацію і втрачає інтерес;
- Циклічне повторення: незважаючи на перемогу над ворогами, вони з'являться через декілька хвилин на тому ж місці, через що гравець вимушено стикається з постійним гріндом.
- Велика кількість незрозумілих речей, що випадають з ворогів, і гравцеві важко зрозуміти, що він має з ними робити.

Circle the Fish

Наступна гра для мобільних пристроїв є спрощеним симулятором риболова, у якій гравцеві пропонується ловити рибу у спеціальних місцях за допомогою сітки. У подальшому риба відвозиться на переробку або ж одразу продається у магазині, а на отримані кошти можна зайнятися покращенням судна, наймом нових робітників або зайнятися відкриттям нових рибних локацій. Кожна з локацій має власний час перезарядки риби, а кожна рибина має свою вагу та вартість. Також паралельно гравець може здійснювати замовлення в порту, що виконують роль щоденних завдань і будуть повторно доступні лише через деякий час.

У грі також побудована система інтерактивних зон, завдяки яким здійснюється керування за допомогою одного пальця. На додачу ми бачимо вдале планування поступового розвитку рибного бізнесу і розвинений баланс гри, мінімалістичний дизайн та можливість змінювати кораблі (рис. 1.3-1.4).

Посилання на гру: <https://apps.apple.com/us/app/circle-the-fish/id1624758267>

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

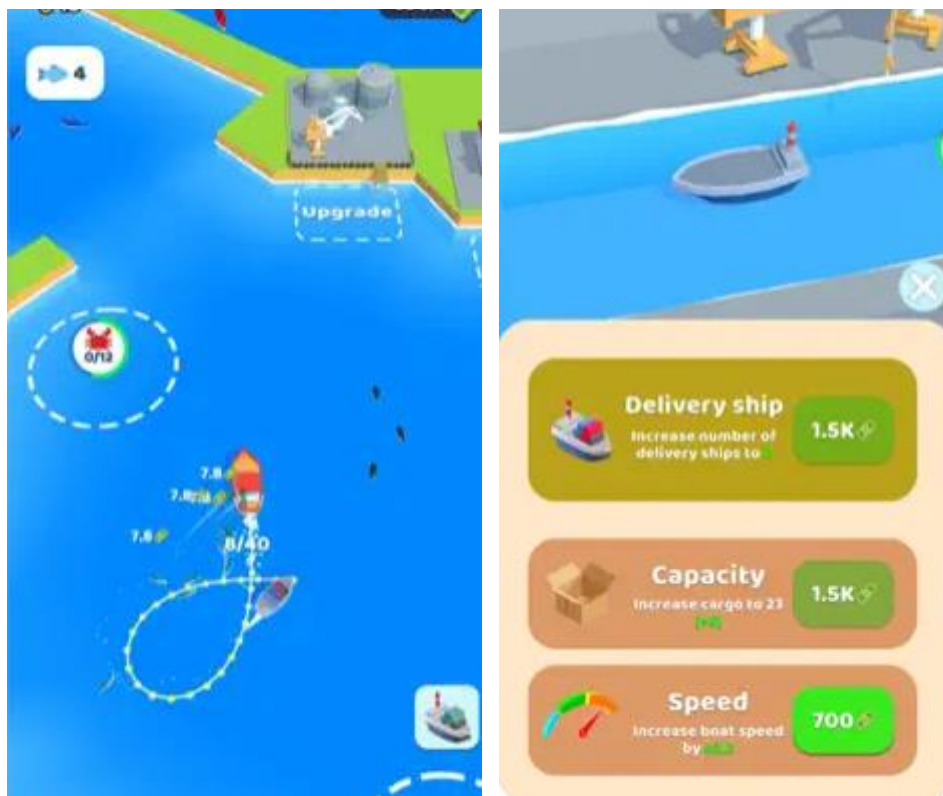


Рис. 1.3-1.4. Circle the Fish

Переваги:

- Помітний розвиток під час проходження гри, унікальні рибні локації
- Можливість як прокачувати персонажа, так і купити кращий корабель із новими характеристиками;

Недоліки:

- Дуже малий простір для плавання;
- Лише один вид фабрики, який виготовляє рибні консерви
- Після проходження першої локації відкривається наступна, яка нічим візуально не відрізняється – це і є точка втрати уваги гравця.

ExoMiner - Idle Miner Universe

Наступна гра виходить за межі гіперказуальних ігор та представляє собою суміш гіперказуальної та стратегічної гри. Також вона є дуже близькою по задумці – гравець повинен добувати різноманітні корисні копалини на інших планетах, удосконалюючи спорядження та розвиваючи індустрію видобутку

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

мінералів. Кожна планета має свою унікальність, а чудовий дизайн вдало приховує рутинність у грі (рис. 1.5-1.6).

Посилання на гру:

<https://play.google.com/store/apps/details?id=com.eldring.exominer>



Рис. 1.5-1.6. ExoMiner

Переваги:

- Постійний розвиток у грі, нові планети
- Гарний дизайн, що відповідає тематиці
- Постійне відкриття нових матеріалів і фабрик

Недоліки:

- Нема головного персонажа
- Відсутній сюжет
- Важко одразу зрозуміти сутність усіх налаштувань та можливостей

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Таблиця 1.1

Характеристика	Stickmans of Wars	Circle the Fish	ExoMiner
Гра від імені головного героя	+	+	-
Покращення	+	+	+
Видимість розвитку	-	+	+
Різноманіття світів	+	-	+
Різноманіття добутих речей	-	-	+
Можливість офлайн-гри	-	-	+
Легкість в освоєнні	+	+	-
Популярність серед цільової аудиторії	+	+	-
Розмір завантаження	-	+	+

При розгляді трьох мобільних ігор, що були взяті за орієнтир, можна виявити, що кожна з них сильна у своїй стороні - Stickmans of Wars пропонує просте і зрозуміле керування та велику кількість підказок у вигляді знаків та прямих доріг, що не дають гравцеві заблукати. Circle the Fish має більшу кількість покращень, дозволяє автоматизувати процеси, що дає користувачу відчуття власний розвиток у грі. ExoMiner натомість гарантує можливість грати офлайн, а також продовжувати видобуток, поки гравець вийшов з гри.

Кожна з ігор налаштована на власну аудиторію, що зацікавлена у тій чи іншій тематиці, і користується чималою популярністю. Проте вибравши найкращі характеристики кожної з них, ми можемо отримати новий додаток, що неодмінно знайде свою частку на ринку [3].

1.3 Вибір архітектури системи

У сучасному світі за стрімкого розвитку інформаційних технологій програмістами було даліджено чимало видів зв'язків між окремими компонентами програми. Це призвело до виникнення поняття архітектури та правильності планування побудови коду. Правильно обраний підхід дозволяє зменшити обсяг роботи у майбутньому, наприклад збільшити перевикористовуваність класів, розбивши великі скрипти на дрібніші із конкретною задачею.

Слово "моноліт" має стародавнє походження і вказує на величезний камінь. Навіть у сучасному вживанні цей термін залишається відомим, і його сутність не змінюється, незалежно від контексту. У сфері програмного забезпечення під монолітним підходом розуміється ідея створення неподільної єдиної одиниці. Концепція монолітного програмного забезпечення передбачає об'єднання різних компонентів програми в одну єдину структуру на одній платформі. Зазвичай монолітний додаток складається з бази даних, інтерфейсу користувача та серверної частини. Всі частини програми є уніфікованими, а їх функції централізовано керуються в одному місці.

Архітектура моноліту є зручною для роботи з невеликими командами, тому що багато стартапів обирають цей підхід при створенні програм. Компоненти монолітного програмного забезпечення взаємопов'язані та взаємозалежні, що дозволяє програмному забезпеченню функціонувати автономно. Ця архітектура є традиційним рішенням для побудови додатків, але деякі розробники вважають її застарілою. Однак монолітна архітектура може бути ідеальним рішенням у певних обставинах [4].

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Для визначення корисності цього архітектурного рішення для вашого бізнесу, розглянемо його переваги та недоліки.

Переваги:

- Простіша розробка та розгортання: Використання багатьох інтегрованих інструментів полегшує процес розробки, а єдиний каталог сприяє простоті розгортання. Монолітне ядро дозволяє розробникам вносити зміни та виконувати оновлення одночасно, що ефективно заощаджує час.
- Легка інтеграція систем моніторингу: Монолітні програми полегшують аналіз трафіку, логування та інші аспекти завдяки своїй єдиній кодовій базі. Підключення компонентів стає простішим, оскільки все функціонує в межах одного додатку.
- Краща продуктивність: Правильно побудовані монолітні програми зазвичай є ефективнішими, порівняно з програмами на основі мікросервісів. Здатність ефективно спілкуватися між компонентами програмного забезпечення через спільний код і пам'ять дозволяє досягти вищої продуктивності.

Недоліки:

- Нагромадження коду: розширення продукту призводить до збільшення обсягу коду, і структура стає менш чіткою. Кодова база стає важкою для розуміння та редагування, особливо для нових розробників, і стає важкою для виявлення побічних ефектів та залежностей.
- Важкість внесення змін: через суцільну структуру згодом може бути важко, а іноді й неможливо змінити структуру базових операцій, що були внесені на попередніх етапах розробки.

Модульна архітектура - це парадигма програмування, в якій весь проект розбивається на окремі, самостійні модулі, кожен із яких відповідає за конкретну функцію або завдання. Основна ідея модульної архітектури полягає

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

в тому, щоб зробити код більш роздільним, ефективним у відносинах до розробки та підтримки [5].

Основні переваги модульного програмування включають в себе:

- Гнучкість та масштабованість: Зміни в одному модулі не впливають на решту системи, що полегшує внесення змін і підтримку проекту. Крім того, можливість використання готових модулів полегшує масштабування проектів.
- Самостійність модулів: Кожен модуль має свою власну функціональність та може бути розроблений та тестований незалежно від інших. Це робить робочий процес більш ефективним та прозорим.
- Стійкість до помилок: Якщо один модуль не працює належним чином, інші можуть продовжувати функціонувати, що полегшує виявлення та виправлення помилок.
- Повторне використання коду: Модулі можуть бути використані в інших проектах або в тому ж самому проекті, що зменшує час розробки та сприяє консистентності.
- Підтримка паралельної розробки: Різні команди можуть працювати над різними модулями паралельно, що прискорює процес розробки.

Недоліки:

- Витрати на комунікацію: Якщо модулі недостатньо добре визначені або недостатньо документовані, може виникнути проблема з розумінням взаємозв'язків між ними. Це може вимагати додаткових зусиль для забезпечення ефективної комунікації між командами розробників.
- Ускладнення контролю версій: При збільшенні кількості модулів може ускладнитися контроль версій. Керування різними версіями різних модулів та їх сумісність може стати викликом.
- Втрата продуктивності через зайві зв'язки: Якщо модулі мають зайві зв'язки, це може призвести до зайвих витрат на продуктивність, оскільки система повинна керувати більшою кількістю взаємозв'язків.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Серед запропонованих рішень було обрано монолітну архітектуру. Незважаючи на можливі втрати гнучкості та масштабованості, ми отримуємо перевагу у швидкості, що є дуже важливим у світі розробки ігор. Також гра не потребуватиме значних видозмін коду, так як її ціль, головна механіка та пропонувані користувачу можливості заздалегідь чітко визначені та не будуть змінені.

Додаток можна логічно розділити на декілька сегментів. Розглянемо основні частини програми:

- Точка входу: Початкова сцена, що завантажує весь поточний прогрес та ініціює роботу інших модулів;
- Моделі: дані певних юнітів повинні бути винесені за межі його структури, щоб полегшити їх зміну та надати властивість зберігати внесені правки.
- Керування: всі об'єкти, що рухаються, слідує однаковій логіці пересування, тому ці елементи також можуть існувати окремо.
- Збереження: окремі структури, що забезпечують спілкування із сховищем та синхронізацію із змінами у грі, мають існувати окремими компонентами, що виконують лише свою роботу.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

При розробці гри необхідно обрати певний рушій, на базі якого створюватиметься майбутня програма. Серед наявних потужностей сфери ІТ щороку з'являються нові версії та види ігрових рушіїв, і кожен з них фокусується на вирішенні власного спектру проблем, що робить їх відмінними для ігор обраного жанру.

Одним із найвідоміших ігрових рушіїв є Unity, що здобув свою славу завдяки багаторічній надійній підтримці, широкій аудиторії, а також тисячам прикладів успішних відеоігор, що були створені без використання великого бюджету на її просування. Відомий своєю дружелюбністю до початківців, Unity пропонує інтуїтивно зрозумілий інтерфейс і багату документацію. Користувачі

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

можуть швидко почати працювати завдяки численним туторіалам та великій кількості ресурсів для навчання. Unity підтримує величезну кількість платформ, включаючи мобільні пристрої (iOS, Android), консолі (PlayStation, Xbox, Nintendo Switch), ПК, веб-браузери, та навіть AR/VR пристрої. Це робить його дуже універсальним інструментом для розробників.

Щодо використання ліцензій - пропонує безкоштовну версію (Unity Personal) з усіма основними функціями, що дозволяє початківцям розробникам легко розпочати роботу без значних витрат. Професійні версії (Unity Plus і Unity Pro) доступні за відносно доступними цінами.

Наступним на розгляді був ігровий рушій Unreal Engine. Це великий та потужний інструмент, що використовується великими компаніями для розробки ігор високої якості. Також підтримує багато платформ, їх фірмовий Unreal Marketplace пропонує багато асетів, які містять набагато якісніші текстури та моделі для роботи, причому багато з них є безкоштовними. Також має активну спільноту, але розміри і активність можуть трохи поступатися Unity. Хоча рушій також має потужну документацію і ресурси для навчання, Unreal Engine часто вважається трохи складнішим для освоєння. Однак найсуттєвішим недоліком є вкрай висока важкість і вартість розробки мобільних проектів, так як рушій заточений під розробку ігор для ПК.

Отже, було обрано рушій Unity [6], що вже давно зарекомендував себе на ринку як надійна, потужна та сучасна оболонка для створення різножанрових ігор. Одним із найбільших його плюсів є наявність додаткових пакетів у магазині Asset Store, що дозволяє використовувати уже готові рішення та зекономити час на певних етапах розробки. Також компонентно-орієнтована система дозволяє розбивати скрипти на декілька елементів, що одночасно використовуються певним об'єктом на сцені.

Як мова програмування застосовується C# [7]. Це об'єктно-орієнтована мова програмування зі строгою типізацією, що покращує взаємодію між окремими частинами коду та забезпечує уникнення базових помилок, таких як невідповідність типів та неможливість передбачити значення із параметра.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Мінімальні системні вимоги до пристрою користувача:

- 100 MB вільного простору на диску;
- Операційна система Android 5.1 (API 22);
- 2GB RAM і більше;
- процесор з тактовою частотою не нижче 2GHz;
- архітектура ARMv7.

Висновки до першого розділу

У першому розділі було проаналізовано завдання та аналогічні мобільні ігри. При порівнянні з такими додатками, як Stickman of wars, Circle the fish та Ehominer, було визначено ключові переваги, що будуть додані у наш продукт.

Було розглянуто різні типи архітектури та обрано найбільш оптимальний варіант. Для нашого проекту найбільше пасуватиме модульна архітектура, що забезпечить гнучкість та адаптивність.

Під час вибору інструменту реалізації було обрано ігровий рушій Unity, що є чудовою оболонкою як для 3D, так і 2D проектів різного масштабу. Для програмування використовуватиметься мова C#, що зарекомендувала себе як надійна мова високого рівня для розробки додатків.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНОЇ ГРИ

«КОСМОНАВТ-ЗОЛОТОШУКАЧ»

2.1 Визначення варіантів використання та структури системи

Для забезпечення неперервного виконання необхідних функцій і відповідності встановленим стандартам та специфікаціям, наведено високорівневі вимоги, які повинна задовольняти наша програма. Ці вимоги допомагають забезпечити безперебійну роботу системи та виконання всіх необхідних стандартів і специфікацій.

Бізнес вимоги

1. Основні цілі: проект створюється з метою залучення гравців до простої розслабляючої гри із подальшим отриманням доходу за допомогою внутрішньо ігрових покупок та реклами.

2. Представлення проекту: проект буде реалізовано у вигляді додатку на мобільні пристрої Android [8] та IOS [9], який можна буде встановити із офіційних магазинів додатків для поданих платформ.

Вимоги користувачів

1. Можливість керувати головним героєм за допомогою простої системи управління;

2. Можливість кастомізації персонажа;

3. Проста механіка видобутку корисних копалин та подальших дій із ними - продаж, переробка тощо.

4. Велика кількість контенту у вигляді локацій, прихованих призів та нових можливостей, що поступово розкриваються при проходженні гри;

5. Можливість виконувати щоденні місії;

6. Система досягнень та отримання призів за кожне виконане досягнення.

7. Можливість придбати певні ресурси за реальні гроші, прискорити процес проходження гри за рекламу.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Вимоги власників

1. Можливість переглядати статистику доходів за продажами внутрішньо ігрових покупок та переглядів реклами;
2. Можливість перегляду графіків використання додатку, перегляду активних користувачів за країнами та групами пристроїв;

Характеристика об'єкта комп'ютеризації

Користувач додатку буде мати можливість виконувати прості дії в грі, яка не потребує багато часу та зусиль на навчання, а також отримає гарне враження про позаземні локації та їх особливості. Гра не потребуватиме обов'язкового з'єднання з мережею Інтернет, що дозволить запускати додаток у будь-якій точці світу, однак для отримання усіх можливостей, таких як щоденні місії, досягнення та магазин, необхідне з'єднання [10].

Функціональні вимоги:

1. Завантаження попереднього збереження: гра повинна мати можливість завантажити попередній прогрес, якщо такий був;
2. Можливість збереження прогресу: програма повинна зберігати будь-які дії користувача, такі як статистика, поточне місцезнаходження персонажа, перелік видобутих ресурсів, кількість ігрової валюти тощо.
3. Нескінченність ігрового процесу: після проходження усіх етапів гравець може продовжити грати або ж скинути прогрес в обмін на бонуси, які допоможуть йому при наступному проходженні гри.

Нефункціональні вимоги:

1. Сприйняття

- час, необхідний для навчання звичайних користувачів – 10-15 хв;
- час відгуку для типових задач – не більше 2 секунд, для складних завдань – не більше 5 секунд.
- інтерфейс має бути зрозумілим та з відповідними підказками.

2. Надійність

- Доступність – час, що витрачається на обслуговування додатку, не повинен перевищувати 5% від загального часу роботи.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

- середній час безвідмовної роботи – 15 робочих днів.
- максимальна норма помилок та дефектів в роботі системи – 1 помилка на десять тисяч рядків коду.

3. Продуктивність

- Гра повинна бути адаптованою під різні розширення екранів та передбачати можливість працювати на повільних пристроях завдяки зниженню якості графіки.

4. Можливість експлуатації

- Масштабування – система повинна мати можливість розвиватись у майбутньому та легко розширятись у наступних оновленнях.

Аналіз вимог користувачів дав підстави для визначення варіантів використання гіперказуальної гри про видобуток корисних копалин. На рис. 1.1 наведено діаграму використання програмної системи (рис. 2.1):



Рис. 2.1. Діаграма використання системи

У результаті аналізу функціональних вимог було створено діаграми активності, що відображають ймовірні сценарії розвитку подій у грі (рис. 2.2):

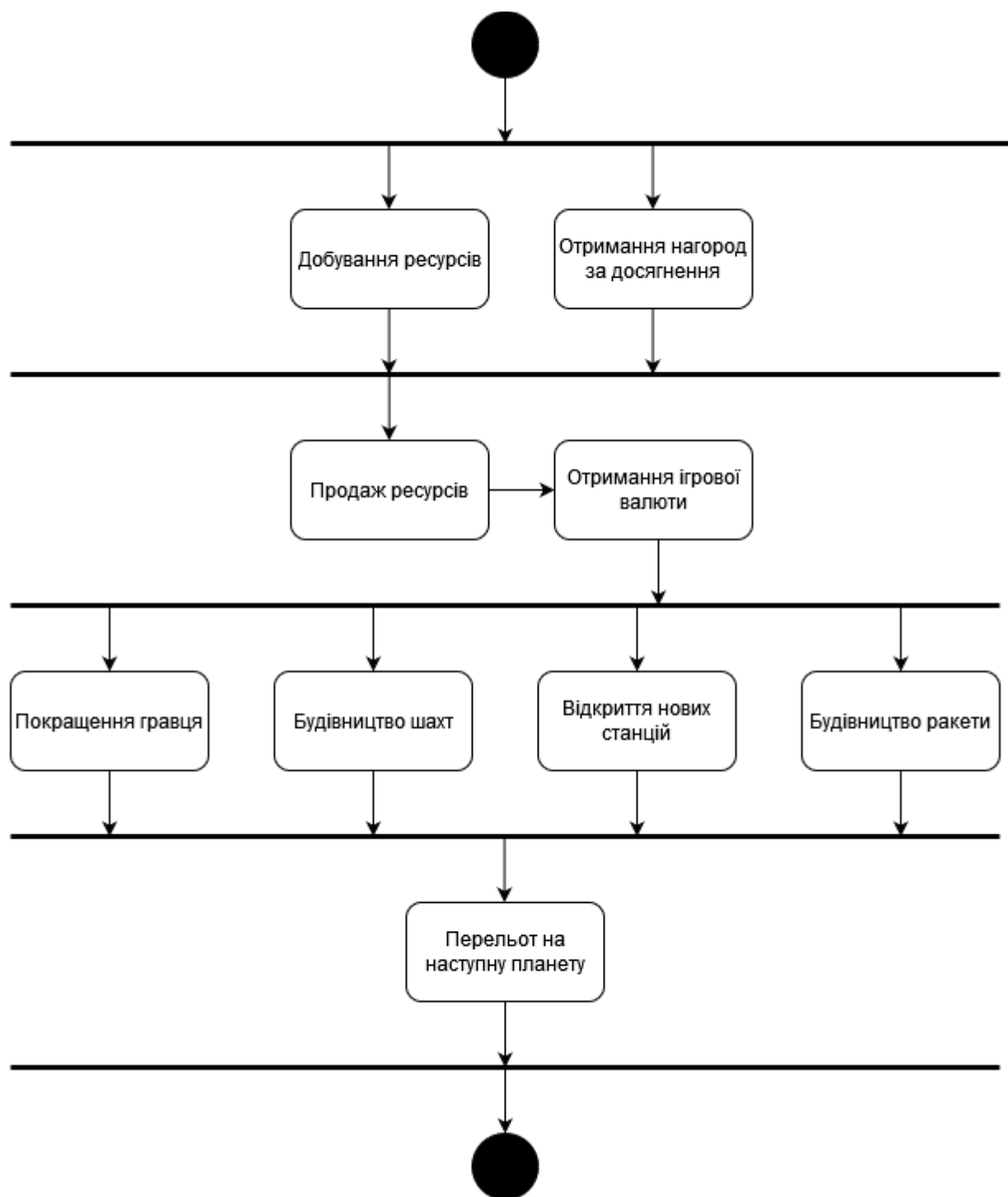


Рис. 2.2. Діаграма активностей взаємодії користувача з основними процесами гри

Розглянемо умовні позначення на діаграмі активності [11]:

- Початковий вузол діяльності (круг): початок потоку виконання діяльності.
- Контроль потоку (стрілка): послідовність виконання між діями.

- Дія (заокруглений прямокутник): конкретна дія, що виконується в системі.
- Вузол-рішення (ромб з розгалуженням): створює певну умову, в якій є два варіанти дій.
- Вузол-злиття (ромб після рішення): об'єднує різні шляхи виконання.
- Вузол-розгалуження (зафарбований прямокутник із стрілками): розділяє процес на паралельні потоки без умови.
- Об'єднуючий вузол (зафарбований прямокутник із стрілками): об'єднує паралельні потоки.
- Фінальний вузол діяльності (круг з контуром): завершує діаграму.
- Доріжка: групує дії, виконувані одним актором або об'єднує дії в одному потоці.

Було побудовано діаграми, що відображають основний процес гри з боку користувача, а також схема роботи показу оголошення з винагородою. Діаграма основних процесів гри відображає перелік можливостей користувача в ігровому світі, а також послідовність певних глобальних процесів, які необхідні для проходження. Пояснення:

- Початковий вузол діяльності: момент початку ігрового процесу
- Вузол-розгалуження;
 - Дія: Добування ресурсів;
 - Дія: Отримання нагород за досягнення;
- Вузол-розгалуження;
 - Дія: Продаж ресурсів;
 - Дія: отримання ігрової валюти за ресурси;
- Вузол-розгалуження;
 - Дія: Покращення персонажа;
 - Дія: Будівництво шахт;
 - Дія: Відкриття нових станцій;

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

- Дія: Будівництво ракети;
- Вузол-розгалуження;
 - Дія: Переліт на іншу планету;
- Кінцевий вузол: завершення гри.

Об'єктно-орієнтована модель системи зображується у вигляді діаграми класів. В ній знаходиться вся структура системи, що може містити різноманітні класи, перерахування, а також зв'язки між ними різних типів.

Розглянемо список загальноприйнятих позначень для побудови діаграм класів. Взаємодія між класами та іншими об'єктами на діаграмі відображається через наступні типи зв'язків:

- Наслідування: Коли один клас успадковує властивості та методи іншого (батьківського) класу.
- Асоціація: Вказує на зв'язок, коли об'єкти одного класу пов'язані з об'єктами іншого класу.
- Агрегація: Виражає зв'язок, при якому один клас може містити об'єкти іншого класу, які можуть існувати незалежно.
- Композиція: Описує зв'язок, коли один клас є невід'ємною частиною іншого класу і не може існувати окремо від нього.
- Реалізація: Вказує на те, що клас забезпечує реалізацію всіх методів, описаних у інтерфейсі, який він реалізує.

Завдяки вищезгаданим умовним позначенням можна відобразити будь-яку сплановану схему сутностей, їх детальну структуру та потенційні шляхи використання, тому ця методика також буде використана і у нашій діаграмі класів. Гра також потребує детального планування процесів та їх виконавців перед розробкою, тому після ознайомлення з усіма правилами створення подібних діаграм було спроектовано власну, що відповідає структурі майбутнього проекту і описує найважливіші аспекти розробки програмного забезпечення(рис. 2.3):

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

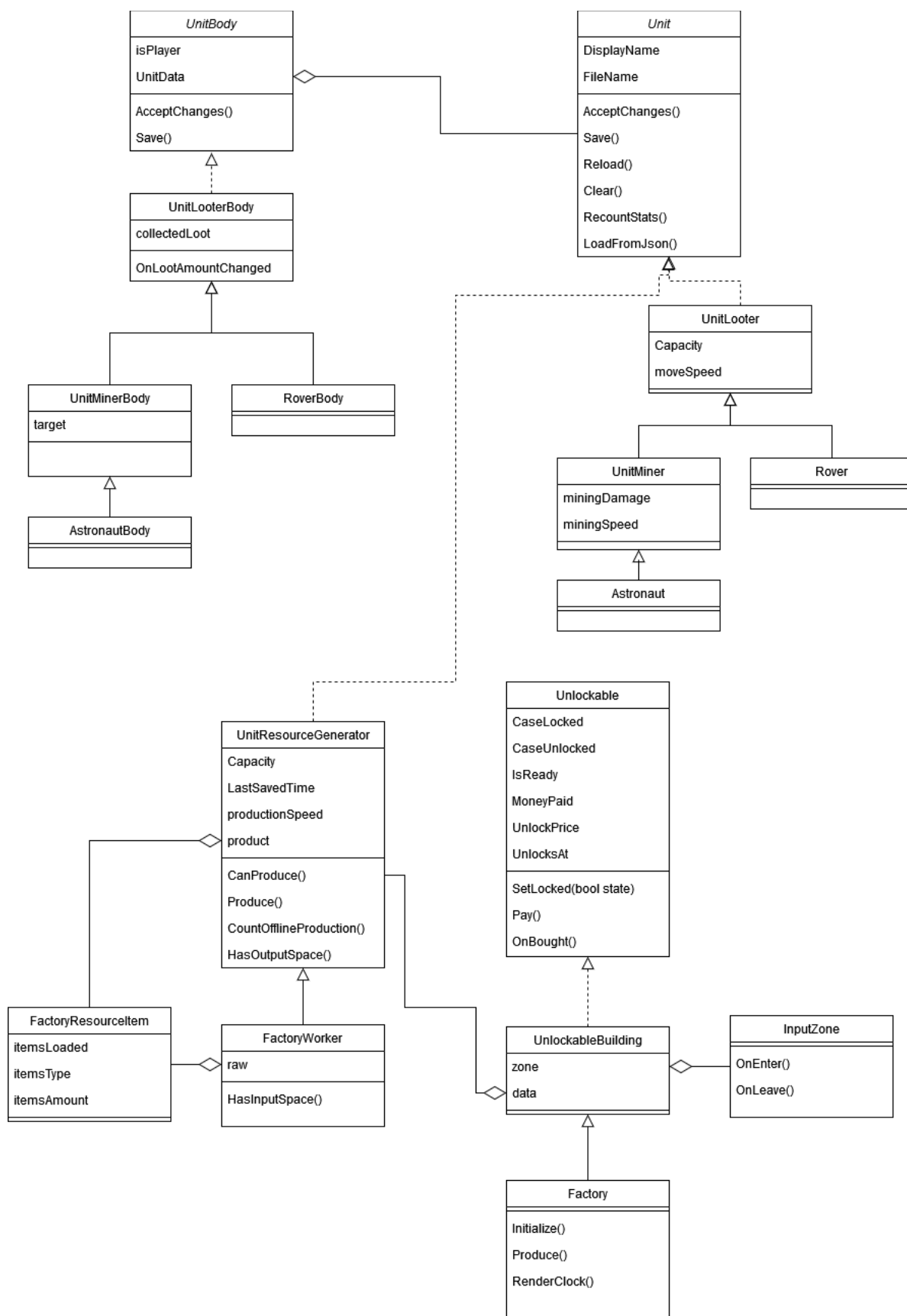


Рис. 2.3. Діаграма класів

Проект складатиметься з набагато більшої кількості класів, тому було прийнято рішення зобразити лише найголовніші, що створюють основи структури гри [12].

Опис класів:

- Unit – абстрактний клас моделі юнітів, який унаслідкується та дає декілька методів для реалізації, наприклад збереження та завантаження;
- UnitLooter, UnitMiner, Rover, Astronaut – класи-юніти, що наслідуються від Unit і представляють конкретні зразки даних;
- UnitBody – абстрактний клас тіла реального ігрового об’єкта, що використовує дані Unit. Містить логіку роботи юнітів;
- UnitLooterBody, UnitMinerBody, RoverBody, AstronautBody – тіла конкретних юнітів, кожен з яких містить методи їх поведінки;
- UnitResourceGenerator – також є юнітом, проте керує їх іншою гілкою – фабриками. З точки зору логіки роботи, кожна фабрика також може покращуватись, тож повинна мати певний набір даних для збереження. Представляє собою станції, що здатні генерувати ресурси.
- FactoryWorker – доповнює клас UnitResourceGenerator і представляє заводи, що виготовляють певні ресурси із інших.
- Unlockable – абстрактний клас будь-якої станції, що може бути побудована.
- UnlockableBuilding – клас будівель. Приймає дані із UnitResourceGenerator та InputZone.
- Factory – клас фабрик. Має методи старту виробництва, видачі продукції та відмальовування прогресу.
- InputZone – клас зони, в яку входить юніт. Має два тригери, що спрацьовують при вході та виході із колайдера. Надалі ці методи можуть бути перевизначені задля власних потреб, наприклад, для викидання ресурсів чи продажу майна.
- FactoryResourceItem – клас ресурсу, що приймає тип, кількість продукції, а також кількість таких партій, що вже були завантажені.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

кожен файл відповідає певний клас, і PlayerDataSynchronizer один із них. Він виконує збереження, після чого сповіщає про зміни інформації. Про це довідуються усі класи UI, у тому числі і PlayerDataViewer, який підписаний на подію зміни даних користувача і відмальовує у інвентарі іконку та кількість кристалів.

Усі матеріали, видобуті гравцем, можуть бути використані для отримання нових ресурсів. Для цього існують станції-фабрики, де здійснюється переробка матеріалів на новий товар, який можна продати дорожче або ж використати як матеріал для інших дорожчих ресурсів, якщо такі є (рис. 2.5):

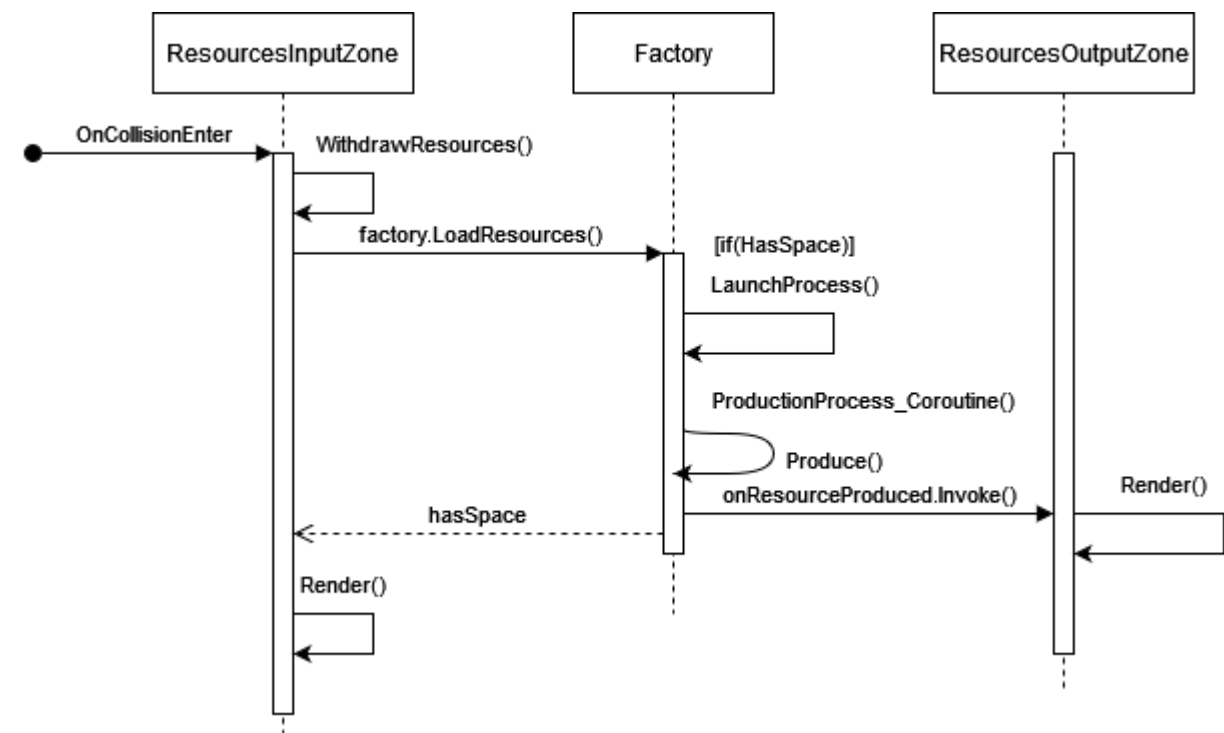


Рис. 2.5. Діаграма послідовності переробки кристалів

Діаграма демонструє наступну логіку:

При зіткненні гравця із зоною взаємодії для завантаження ресурсів зона намагається зібрати необхідні ресурси у гравця у методі WithdrawResoures(). Якщо вони є, ці дані направляються до фабрики, до якої ця зона була приєднана. Фабрика запускає процес виробництва, який через деякий час виготовляє новий товар. Ця інформація поступає у зону для вивантаження виготовлених ресурсів, і вона відповідно перемальовується, щоб відобразити

зміни користувачу. Аналогічно рендерінг виконується і для зони входу ресурсів, де показується кількість поточних завантажених матеріалів.

Так як у грі існує велика кількість фабрик, заводів та станцій, і гравець не завжди може об'єктивно оцінити цінність добутих мінералів, було прийнято рішення додати бота-помічника, що зможе підрахувати найбільш оптимальний варіант використання ресурсів. Він здійснює аналіз наявних ресурсів у інвентарі та доступних на карті заводів, що можуть переробити сировину. Пошук оптимального плану здійснюється шляхом застосування математичного симплекс-методу [14]. Математичні методи дозволяють застосувати більше енергозощадливий та надійний метод до знаходження необхідної пропорції на виробництві, визначення найпростішого шляху до досягнення певної бізнес-цілі, побудувати моделі для передавання подій на ринку тощо. Цей метод має досить широке застосування, тому він також знадобиться для розв'язання нашої поточної задачі. За його допомогою ми зможемо перетворити існуючі дані на таблицю, відносно якої будуть проводитися розрахунки найоптимальнішого способу отримання максимального прибутку із доступних кристалів.

Спроекуємо уявну задачу, що може виникнути під час гри:

Гравець має 3 відкритих заводи, кожен з яких виготовляє свій вид продукції. Кожен продукт складається із певної пропорції мінералів і має свою вартість. Гравцеві доступно два види мінералів – Кварц і Смарагд, що мають різні степені рідкості, а відповідно і отримана продукція із заводів матиме різну вартість. Знайти такий спосіб переробки кристалів, що наявні у інвентарі гравця, щоб сумарний дохід був максимальний. Дані представлено у таблиці 2.1:

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Таблиця 2.1

	Завод 1	Завод 2	Завод 3	Запаси мінералів
Кварц	2	0	2	12
Смарагд	0	2	1	20
Ціна	3	5	6	

Як видно із умови, кожна станція потребує свою пропорцію кристалів для виготовлення нового продукту, ціна якого вказана у нижньому ряду.

Складемо обмеження.

Нехай x_1 – кількість виготовлених продуктів на заводі №1, x_2 – кількість виготовлених продуктів на заводі №2, а x_3 – кількість виготовлених продуктів на заводі №3.

Виходячи з поставлених умов, ми маємо обмежену кількість кристалів кварцу – 12 штук, а також інформацію про їх використання на заводах. Отже, обмеження матиме такий вигляд [15]:

$$2x_1 + 0x_2 + 2x_3 \leq 12$$

Аналогічно, обмежимо використання ресурсу Смарагд, кількість якого становить 20 штук:

$$0x_1 + 2x_2 + 1x_3 \leq 20$$

Виходячи з логічних роздумів, на змінні повинна накладатись умова невід'ємності:

$$x_1, x_2, x_3 \geq 0$$

Складемо цільову функцію до задачі. Якщо завдання полягає у пошуку максимального прибутку, то це свідчить про функцію на $\max$ [16]:

$$F(x_1, x_2, x_3) = 3x_1 + 5x_2 + 6x_3 \rightarrow \max$$

Отримана система обмежень:

$$\begin{cases} 2x_1 + 0x_2 + 2x_3 \leq 12 \\ 0x_1 + 2x_2 + 1x_3 \leq 20 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Можна приступити до розв'язання задачі.

$$F(x_1, x_2, x_3) = 3x_1 + 5x_2 + 6x_3 \rightarrow \max$$

$$\begin{cases} 2x_1 + 0x_2 + 2x_3 \leq 12 \\ 0x_1 + 2x_2 + 1x_3 \leq 20 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Використовуємо метод симплекс-таблиць:

$$F(x_1, \dots, x_5) = 3x_1 + 5x_2 + 6x_3 + 0x_4 + 0x_5 \rightarrow \max$$

$$\begin{cases} 2x_1 + 0x_2 + 2x_3 + 1x_4 \leq 12 \\ 0x_1 + 2x_2 + 1x_3 + 1x_5 \leq 20 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

$$x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 12, x_5 = 20.$$

Складається симплекс-таблиця:

Таблиця 2.2

	С	–	3	5	6	0	0
	В	A_0	A_1	A_2	A_3	A_4	A_5
0	x_4	12	2	0	2	1	0
0	x_5	20	0	2	1	0	1
	Δ	0	-3	-5	-6	0	0

Розв'язуємо задачу ЛП на *max*, маємо в індексному рядку одну з найвід'ємніших оцінок $\Delta_3 = -6$, а у відповідному стовпці додатні елементи, тому можливий перехід до іншого, більш кращого розв'язку задачі [17].

Визначаємо напрямний стовпець – A_3 (за найвід'ємнішою оцінкою), який вказує на змінну, що вводиться в базис для покращення розв'язку задачі – x_3 .

Напрямний рядок.

$$\text{Min} \left\{ \frac{12}{2}; \frac{20}{1} \right\} = \frac{12}{2} = 6$$

Отже, напрямний рядок – x_4 . Змінна x_4 виводиться з базису.

На перетині напрямного стовпця та напрямного рядка знаходиться напрямний елемент – x_4 .

Таблиця 2.3

	C	–	3	5	6	0	0
	B	A_0	A_1	A_2	A_3	A_4	A_5
0	x_3	12 : 2	2 : 2	0 : 2	2 : 2	1 : 2	0 : 2
0	x_5	20- (12*1):2	0-(2*1):2	2-(0*1):2	1-(2*1):2	0-(1*1):2	1-(0*1):2
	Δ						

Складається симплекс-таблиця:

Таблиця 2.4

	C	–	3	5	6	0	0
	B	A_0	A_1	A_2	A_3	A_4	A_5
6	x_3	6	1	0	1	1/2	0
0	x_5	14	-1	2	0	-1/2	1
	Δ	36	3	-5	0	3	0

Розв'язуємо задачу ЛП на *max*, маємо в індексному рядку одну з найвід'ємніших оцінок $\Delta_2 = -5$, а у відповідному стовпці додатні елементи, тому можливий перехід до іншого, більш кращого розв'язку задачі.

Визначаємо напрямний стовпець – A_2 (за найвід’ємнішою оцінкою), який вказує на змінну, що вводиться в базис для покращення розв’язку задачі – x_2 .

Напрямний рядок.

$$\text{Min} \left\{ \frac{14}{2} \right\} = \frac{14}{2} = 7$$

Отже, напрямний рядок – x_5 . Змінна x_5 виводиться з базису.

На перетині напрямного стовпця та напрямного рядка знаходиться напрямний елемент – x_5 .

Таблиця 2.5

	C	–	3	5	6	0	0
	B	A_0	A_1	A_2	A_3	A_4	A_5
0	x_3	$6 - (14 \cdot 0) : 2$	$1 - (1 \cdot 0) : 2$	$0 - (2 \cdot 0) : 2$	$1 - (0 \cdot 0) : 2$	$1/2 - (1/2 \cdot 0) : 2$	$0 - (1 \cdot 0) : 2$
0	x_2	$14 : 2$	$-1 : 2$	$2 : 2$	$0 : 2$	$-1/2 : 2$	$1 : 2$
	Δ						

Складається симплекс-таблиця:

Таблиця 2.6

	C	–	3	5	6	0	0
	B	A_0	A_1	A_2	A_3	A_4	A_5
6	x_3	6	1	0	1	1/2	0
5	x_2	7	-1/2	1	0	-1/4	1/2
	Δ	71	1/2	0	0	7/4	5/2

У стовпці A_0 немає від’ємних елементів – отже, отримано оптимальний цілочисельний розв’язок задачі:

$$x_1 = 0, x_2 = 7, x_3 = 6, F_{\max} = 71.$$

Отже, найоптимальнішим рішенням для гравця буде виготовити із наявних у інвентарі кристалів 7 одиниць продукції із заводу №2 та 6 одиниць

продукції на заводі №3. Отримані товари становитимуть найбільший можливий прибуток.

Під час гри можуть траплятися безліч аналогічних задач, заснованих на кількості відкритих фабрик, їх сировини та наявних ресурсів у гравця, тому спроектований алгоритм має побудувати обмеження, зважаючи на вищевказані параметри, та запустити виконання програмного симплекс-методу для отримання кінцевої відповіді:

Схема роботи програмного алгоритму симплекс-методу [18] (рис. 2.6):

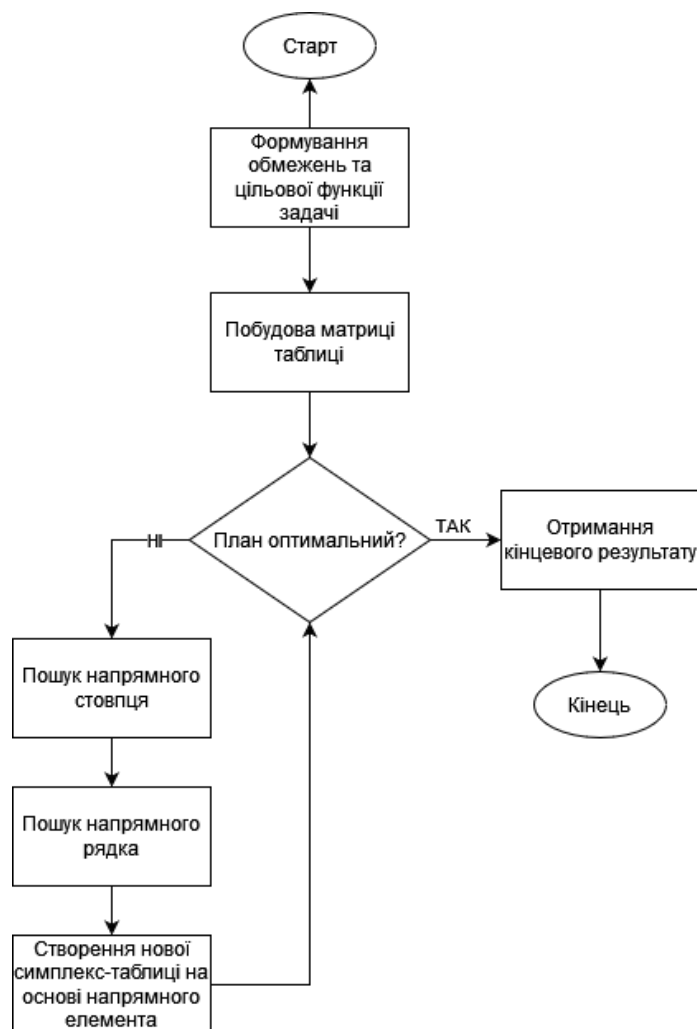


Рис. 2.6. Схема роботи алгоритму симплекс-методу

2.3 Реалізація функціоналу додатку

Основним ядром [19] системи буде клас Kernel, що міститиме посилання на решту основних модулів, що можуть використовуватись іншими класами:

```
public static class Kernel
{
    public static readonly IRouter RootManager = new RootManager();

    public static readonly ISaver SaveManager = new SaveManager();

    public static readonly IDisplayManager DisplayManager = new DisplayManager();

    public static readonly ITagsManager TagsManager = new TagsManager();

    public static readonly PoolDictionary PoolDictionary = new PoolDictionary();

    public static readonly VibrationController VibrationController = new
VibrationController();

    public static readonly ScenesDictionary ScenesDictionary = new ScenesDictionary();

    public static readonly Env Env = new Env();

    public static readonly AstronautFactory AstronautFactory = new
AstronautFactory(PrefabsAssistant.Instance.AstronautBody);
    public static readonly RoverFactory RoverFactory = new
RoverFactory(PrefabsAssistant.Instance.RoverBody);
    public static readonly RoverFactory DroneFactory = new
RoverFactory(PrefabsAssistant.Instance.DroneBody);
    public static readonly OreFactory OreFactory = new
OreFactory(PrefabsAssistant.Instance.Ore1);

    public static readonly LootSynchronizer LootSynchronizer = new LootSynchronizer();
    public static readonly PlayerDataSynchronizer PlayerDataSynchronizer = new
PlayerDataSynchronizer();
    public static readonly UnitDataSynchronizer UnitDataSynchronizer = new
UnitDataSynchronizer();
    public static readonly UnlocksDataSynchronizer UnlocksDataSynchronizer = new
UnlocksDataSynchronizer();
    public static readonly CutsceneSynchronizer CutsceneSynchronizer = new
CutsceneSynchronizer();
    public static readonly TimeSynchronizer TimeSynchronizer = new TimeSynchronizer();
    public static readonly PlanetSynchronizer PlanetSynchronizer = new
PlanetSynchronizer();
    public static readonly SettingsSynchronizer SettingsSynchronizer = new
SettingsSynchronizer();
    public static readonly AchievementSynchronizer AchievementSynchronizer = new
AchievementSynchronizer();}
```

Серед них можна виділити декілька типів глобальних класів – списки даних, синхронізатори, фабрики та сервіси.

Списки даних зберігають певну глобальну інформацію, що може бути доступною для читання для решти класів, що її потребують. Такими є RootManager – містить посилання на внутрішні та зовнішні шляхи до файлів, а

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

також TagsManager, що зберігає назви тегів і дозволяє позбутись прив'язок до рядкових типів даних.

Синхронізатори дозволяють виконувати збереження гри, маючи конструкцію, що дозволяє керувати одному класу лише одним файлом або ж однією групою файлів у певній теці. Такими є LootSynchronizer, PlayerDataSynchronizer тощо. Кожен з них виконує лише функцію менеджменту свого виду даних та налагоджує спілкування між сховищем та грою.

Фабрики містять логіку створення нових екземплярів об'єктів з різними характеристиками, маючи лише один збережений зразок. Для прикладу, AstronautFactory описує етапи побудови астронавтів-ботів та астронавтів-гравців в залежності від отриманих даних, використовуючи лише одну заготовку-префаб.

Сервіси – мають чітке значення в проєкті у вигляді потенційного використання для певної задачі. Зміни коду всередині них неодмінно вплинуть на весь процес всередині гри. Одним із таких модулів є клас SaveManager, що містить логіку завантаження та збереження гри:

```
public class SaveManager : ISaver
{
    private Cipher CipherType = new EmptyEncoding();

    public void SetCipher(Cipher cipher)
    {
        CipherType = cipher;
    }

    public string Save(object objToSave, string path)
    {
        string objToJson = JsonUtility.ToJson(objToSave, true);
        objToJson = Encrypt(objToJson);

        File.WriteAllText(path, objToJson);
        return path;
    }

    public string SaveDefault(object objToSave, string path)
    {
        string objToJson = JsonUtility.ToJson(objToSave, true);
        objToJson = Encrypt(objToJson);

        File.WriteAllText(path, objToJson);
        return path;
    }

    public string Load(string path)
```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    string objJson = File.ReadAllText(path);
    objJson = Decrypt(objJson);

    return objJson;
}

public T Load<T>(string path)
{
    return JsonUtility.FromJson<T>( Load(path) );
}

public string Encrypt(string strToEncrypt)
{
    string decryptedStr = CipherType.Encrypt(strToEncrypt);
    return decryptedStr;
}

public string Decrypt(string strToDecrypt)
{
    string encryptedStr = CipherType.Decrypt(strToDecrypt);
    return encryptedStr;
}
}

```

SaveManager не виконує збереження конкретного файлу, а лише представляє інструмент для використання в інших класах, щоб система збереження інформації завжди мала спільну логіку та була гнучкою до потенційних змін. Також він виконує і шифрування даних, логіка шифрування [20] описується у нащадках класу Cipher:

```

public abstract class Cipher
{
    public abstract string Encrypt(string strToEncrypt);
    public abstract string Decrypt(string strToDecrypt);
}

```

Усі посилання на зовнішні та внутрішні ресурси містяться у класі у вигляді шляхів:

```

public class RootManager : IRouter
{
    public string BotsDirectory => "Bots/";
    public string SaveFormat => ".ktefx";
    public string FakeSaveFormat => ".png";

    public string TimeDataExternalPath => Application.persistentDataPath + "/dt" +
    FakeSaveFormat;
    //...
}

```

Далі, вищезгадані класи будуть використані разом для зручного менеджменту збережень [21]. Для цього було розроблено систему синхронізації даних – на кожен файл з даними у зовнішньому сховищі є свій клас-

синхронізатор, який відповідає за нього. Одним із них є клас
PlayerDataSynchronizer:

```
public class PlayerDataSynchronizer : AbstractFileSynchronizer
{
    protected override string savePath => Kernel.RootManager.PlayerDataExternalPath;
    protected PlayerData playerData = new PlayerData();

    #region GET-SET
    public string GetSkin()
    {
        return playerData.Skin;
    }

    public void SetSkin(string skin)
    {
        playerData.Skin = skin;
        SaveData();
    }

    public List<MiningLoot> GetCollectedGems()
    {
        return playerData.CollectectedGems;
    }

    public int GetLootCount()
    {
        return playerData.CollectectedGems.Count;
    }

    public void SetCollectedGems(List<MiningLoot> value)
    {
        playerData.CollectectedGems = value;
        SaveData();
    }

    public int GetMoney()
    {
        return playerData.Money;
    }

    public void AddMoney(int amount)
    {
        playerData.Money += amount;

        InterSceneManager.Instance?.AchievementSystem.OnCoinsCollected?.Invoke(amount);

        SaveData();
    }

    public bool SubstractMoney(int amount)
    {
        if (playerData.Money >= amount)
        {
            playerData.Money -= amount;
            SaveData();

            return true;
        }
        else
        {
            return false;
        }
    }
}
```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    #endregion

    public override void Reload()
    {
        playerData = Kernel.SaveManager.Load<PlayerData>(savePath);
        Invoke();
    }

    protected override void SaveData()
    {
        Kernel.SaveManager.SaveAsynchronously(playerData, savePath);
        Invoke();
    }
}

```

Як ми бачимо, він містить власний перелік методів, що дозволяють редагувати зарання визначений перелік даних – кількість монет, скін персонажа, інвентар тощо. Кожна з таких опцій представлена у вигляді власного блоку get-set методів, що здійснюють додаткове фільтрування даних та не дозволяють введення помилкових. Клас містить зв'язок із класом серіалізації, із якого потім буде сконструйовано дані для збереження у файл. Також для отримання інформації по те, куди потрібно виконувати зберігання, за допомогою змінної savePath відбувається звернення до вищезгаданого класу RootManager.

Завантаження файлу та перезаписування локальних змінних на його основі відбувається при старті гри у функції Reload(). Збереження відбувається у методі SaveData(), який викликається щоразу при зміні даних через set-метод. Таким чином, гра завжди має актуальні кешовані дані, що можуть бути зміненими у будь-який момент

Предок класу – AbstractSynchronizer – містить певну загальну логіку:

```

public abstract class AbstractSynchronizer
{
    protected abstract string savePath { get; }

    public AbstractSynchronizer(bool autoInitialization = false)
    {
        if (autoInitialization)
        {
            Init();
        }
    }

    public abstract void Init();
}

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public abstract void Reload();

protected abstract void SaveData();

public event UnityAction OnDataChanged;

protected void Invoke()
{
    OnDataChanged?.Invoke();
}
}

```

Як і кожному іншому синхронізатору, при запуску гри потрібно знайти файл, і закешувати дані в грі для подальшого використання. При зміні даних змінюються локальні кешовані дані, відразу спрацьовує подія, що слухає будь-які зміни в даних. Відразу виконується збереження найновіших змін у зовнішнє сховище. Таким чином, налагоджується постійна неперервна лінія зв'язку із файлами в ОС, що дозволяє полегшити подальшу розробку та необмежено масштабувати систему збереження.

Розглянемо систему взаємодії ігрових об'єктів. Найголовнішим класом для всіх персонажів є Unit [22], із якого в подальшому створюються всі екземпляри даних юнітів:

```

public abstract class Unit : ScriptableObject
{
    public string DisplayName = "";
    public string FileName = "";

    public bool IsBought = false;

    private void OnValidate()
    {
        FileName = this.name;
    }

    protected virtual void Save()
    {
        RecountStats();
        Kernel.UnitDataSynchronizer.SaveUnit(this);
    }

    public void ApplyChanges()
    {
        Save();
    }

    public abstract void LoadFromJson(string json);
    public abstract UpgradableValue[] GetUpgradableValues();
    public abstract void UpdateUnitValues(Unit loadedUnit);

    public void RecountStats()
    {
        if (GetUpgradableValues().Length > 0)
        {
            foreach (UpgradableValue variable in GetUpgradableValues())

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        variable.RecountValue();
    }
}

public virtual void Clear()
{
    foreach (UpgradableValue variable in GetUpgradableValues())
        variable.Clear();

    IsBought = false;
}

public abstract float GetIncomePerMinute();
}

```

Клас описує базову логіку у вигляді методів для збереження, завантаження, покращень та скидання налаштувань персонажа. Unit є закритим для редагування, але відкритим для розширення – він побудований таким чином, щоб можна було легко наслідуватись від нього та додати певні особливості новому класу, що уособлюватиме інформацію про нового юніта.

Однак для використання даних потрібен клас-тіло, який зможе взаємодіяти з ігровим світом та комунікувати із іншими компонентами, що будуть додані до певного ігрового об'єкта. На допомогу приходить клас із відповідною назвою – UnitBody [23]:

```

[RequireComponent(typeof(Rigidbody))]
public abstract class UnitBody : MonoBehaviour
{
    public abstract Unit UnitData { get; set; }
    public Collider2D[] BodyColliders { get; private set; }

    protected Rigidbody rb;

    protected bool isPlayer = false;
    public void SetIsPlayer()
    {
        isPlayer = true;
    }

    protected virtual void Start()
    {
        BodyColliders = GetComponentsInChildren<Collider2D>();
        rb = GetComponent<Rigidbody>();

        Kernel.UnitDataSynchronizer.OnUnitUpgraded += OnUnitUpgraded;
    }

    public abstract Type GetRefClass();

    public abstract void OnUnitUpgraded(Type type);

    private void OnDestroy()
    {
        Kernel.UnitDataSynchronizer.OnUnitUpgraded -= OnUnitUpgraded;
    }
}

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Базові потреби кожного юніта – знати, які дані йому відповідають, підписуватися на їх зміни при створенні для вчасної їх демонстрації гравцю, і відповідно відписуватись при знищенні.

Розширення потреб також відбувається у класах, що його наслідують – одним із них є AstronautUnitBody, який також є і головним персонажем. Розглянемо принцип його основної роботи – видобутку руди:

```
protected override void Start()
{
    base.Start();

    OnOreCollided += CrystalOreCollision;
    OnOreCollisionExit += CrystalOreUncollided;
    OnLootAmountChanged += UiUpdate;

    SetPickaxeActive(false);
    UiUpdate();
}
```

На початку виконується підписка на внутрішні події [24] для відслідковування зіткнень із рудою та оновлення лейблу з інформацією про інвентар.

```
private void CrystalOreCollision(PreciousOre ore)
{
    if (GetTarget() != null && HasPlaceInBag())
    {
        if (CanInteractWithUI())
        {
            StartMining();
        }
        else
        {
            StopMining();
        }
    }
}

private void CrystalOreUncollided()
{
    StopMining();
}
```

При зіткненні із мінералом виконується перевірка, чи об’єкт не був знищений і чи є місце у рюкзаку. Відповідно, виконується виклик методу StartMining або StopMining:

```
public override void StopMining()
{
    SetIsMining(false);
    if (currentAnimaton != AnimMove)
        ChangeAnimation(AnimIdle);
    SetPickaxeActive(false);
}

public override void StartMining()
```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```
{
    SetIsMining(true);
    ChangeAnimation(AnimMine);
    SetPickaxeActive(true);
}
```

Вище згадані методи контролюють анімацію удару по каменю, яка запустить у певний момент функцію Mine:

```
public void Mine(IMineable ore)
{
    MiningLoot gem = ore.Mine(_data.MiningDamage.Value, transform.position, isPlayer);
    AddToLoot(gem);

    OnLootAmountChanged?.Invoke();

    if (!HasPlaceInBag())
        StopMining();

    StartCoroutine(CheckIfNull_Coroutine());
}
```

Так як гравець знає, з яким каменем від зіткнувся, буде викликаний метод Mine всередині об'єкта-каменя, по якому був нанесений удар:

```
public MiningLoot Mine(float damage, Vector3 minerPos, bool isPlayer)
{
    int knockedGems = 0;
    if (damage >= hp)
    {
        knockedGems = resourceAmount;
    }
    else
    {
        knockedGems = (int)(1.0f * damage * resourceAmount / hp);
    }

    hp -= damage;
    resourceAmount -= knockedGems;

    if (hp <= 0)
    {
        if(isPlayer)
            GlobalManagerHolder.Instance.BoosterManager.TryChanceBooster(transform);

        OnHpEnd();
        Destroy(gameObject);
    }

    MiningAnim(minerPos);

    return new MiningLoot(OreData.Type, knockedGems);
}
```

Згідно формули, яка залежить від співвідношення наявного та знятого здоров'я у мінерала, гравцю нараховується відповідна кількість добутих кристалів. При здоров'ї у вигляді менше чи дорівнює нулю мінерал самознищиться.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Наступною не менш важливою механікою є взаємодія користувача із навколишнім світом через інтерактивні зони – саме завдяки ним відбувається купівля, продаж, обмін ресурсів тощо.

GroundUI – це базовий клас для всіх наступних видів інтерактивних зон:

```
public abstract class GroundUI : MonoBehaviour
{
    public abstract void OnObjectCollision(Collider collider);
    public abstract void OnObjectCollisionExit(Collider collider);

    private void OnTriggerEnter(Collider other)
    {
        OnObjectCollision(other);
    }

    private void OnTriggerExit(Collider other)
    {
        OnObjectCollisionExit(other);
    }
}
```

При зіткненні [25] із об’єктом спрацьовує тригер, аналогічно і при виході з колізії. За таким принципом відбуватимуться різноманітні кастомні події. Наприклад, такі налаштування містяться у класі SellZone:

```
public class SellZone : GroundUI
{
    List<UnitLooterBody> sellers = new List<UnitLooterBody>();
    public event UnityAction<int> OnItemsSold;

    public override void OnObjectCollision(Collider collider)
    {
        if (collider.TryGetComponent(out UnitLooterBody unitMiner))
        {
            SellLootOf(unitMiner);
        }
    }

    private void SellLootOf(UnitLooterBody unitMiner)
    {
        int sum = 0;

        var list = unitMiner.GetLoot().List;
        if (list != null)
        {
            foreach (MiningLoot item in list)
            {
                Gem g = Kernel.LootSynchronizer.Type2Loot(item.type);
                sum += g.Price * item.amount;
            }
        }

        unitMiner.ClearLoot(true);

        if (sum != 0)
```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            Kernel.PlayerDataSynchronizer.AddMoney(sum);
            SpawnIncomePopup(sum, unitMiner.transform.position);

            OnItemsSold?.Invoke(sum);
        }
    }

    private void SpawnIncomePopup(float income, Vector3 pos)
    {
        GameObject g = GlobalManagerHolder.Instance.ControllerPool
            .Spawn(Kernel.PoolDictionary.IncomeItem, pos, Quaternion.identity);
        g.GetComponent<IncomeUiItem>().Setup(income);

        GlobalManagerHolder.Instance.ControllerPool.PlaySound(
            SoundAssistant.Instance.Cash,
            transform.position, 0.5f);
    }

    public override void OnObjectCollisionExit(Collider collider)
    {
        if (collider.TryGetComponent(out UnitMinerBody unitMiner))
        {
            sellers.Remove(unitMiner);
        }
    }
}

```

Перевизначення базових методів додало необхідний функціонал – продавати ресурси можуть лише юніти, похідні від класу UnitLooter, що є батьківським для всіх персонажів-добувачів руди. При зіткненні виконується алгоритм продажу у вигляді обнулення інвентаря та нарахування доходу гравцеві. Для отримання заробітку необхідно проітеруватися по всьому списку матеріалів у інвентарі, обрахувати їх суму, після чого відбудеться його повне скидання.

Також відмальовується анімація отримання грошей – у місці зіткнення із інтерактивною зоною з'явиться невелика іконка валюти із позначкою її кількості, що повільно підніматиметься і потім поступово зникне. Перевизначені методи колізі дозволяють додавати власний й функціонал та зважди відкриті до розширення.

Потенційна кастомізація та розробка нових видів інтерактивних зон не має меж – наприклад, аналогічним способом було розроблено плитку, при ставанні на яку повинно з'явитися вікно покращення заводу:

```

public class FactoryUpgradeZone : AbstractBuildingUpgradeZone
{

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        protected override AbstractPopup Popup { get => popup; set => popup =
(PopupFactoryUpgrade)value; }

        [SerializeField] PopupFactoryUpgrade popup;

        protected override void Start()
        {
            popup =
GlobalManagerHolder.Instance.PanelManager.CreatePopup<PopupFactoryUpgrade>(popup);

            UnitResourceGenerator refUnitData =
(UnitResourceGenerator)building.GetUnitData();

            popup.SetRefUnitData(refUnitData);
            popup.SetHeader(refUnitData.DisplayName);

            base.Start();
        }
    }
}

```

Об'єкту потрібно лише надати посилання на фабрику, яку буде покращено, а також вікно, яке потрібно відкрити.

Будівлі займають чи не найважливішу частину геймплею – неможливо уявити розвиток гравця без будівництва нових шахт та станцій. За принципом відкриття, шахти та будівлі мають спільні риси – місце під забудову повинно позначитись лише при відкритті цієї локації, після до інтерактивної зони потрібно принести вказану кількість грошей. Після погашення всієї суми буде зведено нову станцію. Тому було прийнято рішення винести спільне у загальний клас – Unlockable:

```

public abstract class Unlockable : MonoBehaviour
{
    [SerializeField] public int UnlocksAt = 1;

    [SerializeField] protected int UnlockPrice = 500;
    public int GetUnlockPrice() => UnlockPrice;
    public void SetTestUnlockPrice(int i)
    {
        UnlockPrice = i;
    }

    [SerializeField] protected int MoneyPaid = 0;
    public int GetMoneyPaid() => MoneyPaid;
    public int GetMoneyLeft() => UnlockPrice - MoneyPaid;
    public void SetMoneyPaid(int a) => MoneyPaid = a;
    public abstract void Pay(int a);

    protected bool IsReady = false;

    public void SetUnlocksAtEditor(int val)
    {
        UnlocksAt = val;
    }
}

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public virtual void SetLocked(bool state)
{
    CaseLocked.SetActive(state);
    CaseUnlocked.SetActive(!state);

    IsReady = !state;
}

public virtual void SetUnavaliabile()
{
    CaseLocked.SetActive(false);
    CaseUnlocked.SetActive(false);

    IsReady = false;
}

public abstract void OnBought();
}

```

Клас владіє усіма можливостями, що були згадані раніше, а також містить ідентифікатор, так як усі станції можуть відкриватися лише у вказаному порядку. На кожному етапі гравець може відкрити лише одну станцію, тому дане обмеження було враховано. Для збереження коштів, які були внесені на пбудову даної станції, клас повинен містити методи GetMoneyPaid() та GetMoneyLeft(). Також оплата здійснюється через функцію void Pay(int a), яка доступна для перевизначення завдяки своїй абстрактності. Щоб відкрити станцію, потрібно пройти декілька фаз – її доступність до відкривання (попередня по черзі станція повинна бути придбаною), далі сплатити повну ціну, яка вказана користувачу на інерактивній зоні, і після цього станція буде розблокованою і готовою до використання, що в свою чергу дозволить купівлю наступної конструкції на цій планеті, якщо така є.

Розгалуження шахт та будівель відбувається у похідних класах.

Такий вигляд має клас UnlockableBuilding, що використовується для створення станцій:

```

public abstract class UnlockableBuilding : Unlockable
{
    [Space(5)]
    [SerializeField] internal BuyBuildingZone zone;

    public abstract Unit GetUnitData();

    public override void Pay(int a)
    {
        if (!IsReady)
        {
            MoneyPaid += a;

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Kernel.UnlocksDataSynchronizer.SetBuildingMoneyPaid(MoneyPaid);

        if (GetMoneyLeft() <= 0)
        {
            SetLocked(false);
            GlobalManagerHolder.Instance.BuildingUnlockManager.BuyBuilding();
            OnBought();
        }
    }

protected virtual void Start()
{
    zone?.SetBuilding(this);

    if(IsReady)
        OnBought();
}
}

```

А клас UnlockableMine, який відповідає за шахти, описує таку логіку – шахта аналогічно купується, однак розблоковує інший об’єкт, що не приймає ніяких ресурсів та не має інтерактивних зон для подальшої взаємодії – на місці розблокованої шахти з’явиться пустота, де будуть з часом створюватися об’єкти руди:

```

public abstract class UnlockableMine : Unlockable
{
    [Space(5)]
    [SerializeField] protected BuyMineZone zone;
    [SerializeField] int minPickaxeLevel;

    protected virtual void Awake()
    {
        GlobalManagerHolder.Instance.MineUnlockManager.Add(this);
    }

    public override void Pay(int a)
    {
        if (!IsReady)
        {
            MoneyPaid += a;
            Kernel.UnlocksDataSynchronizer.SetMineMoneyPaid(MoneyPaid);

            if (GetMoneyLeft() <= 0)
            {
                SetLocked(false);
                GlobalManagerHolder.Instance.MineUnlockManager.BuyBuilding();
                OnBought();
            }
        }
    }

    protected virtual void Start()
    {
        zone?.SetBuilding(this);
        zone?.SetMinPickaxeLevel(minPickaxeLevel);
    }
}

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (IsReady)
            OnBought();
    }
}

```

Як ми бачимо, відмінності між класами полягають у налаштування менеджерів порядку відкриття, а також і певними власними параметрами, наприклад, `UnlockableMine` додатково містить обмеження на мінімальний рівень покращення кирки, з яким буде можливо відкрити шахту.

Розглянемо одну із будівель – фабрику `MineralProcessingFactory`. Клас було побудовано таким чином, щоб в неї можна було підставити дані, які ресурси вона приймає та які виготовляться у поле `refData`:

```
[SerializeField] UnitFactoryWorker refData;
```

Також клас ніяк не залежить від зовнішнього вигляду фабрики – за необхідності, ми можемо змінити вигляд одного із підприємства з метою урізноманітнення геймплею та при цьому ніяк не змінювати код.

При запуску гри (або при покупці фабрики) одразу запускається така функція:

```

public override void OnBought()
{
    OnStatsChanged += Render;
    SubscribeToUnitUpgrades();
    Invoke();

    SetSmoke(false);

    if (CanProduce())
        LaunchProcess(true);
    else
    {
        ClearClock();
        StopProcess();
    }

    UnlockResource();
}

```

Вищеописана логіка описує запуск роботи заводу, який через певний час виконує переробку. Для цього використовується корутина:

```

public IEnumerator ProductionProcess_Coroutine()
{
    yield return new WaitForSeconds(refData.ProductionSpeed.Value);

    Produce();
    Invoke();

    refData.ApplyChanges();

    StopProcess();
}

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (CanProduce())
    LaunchProcess();
}

```

Фабрика віднімає від завантажених ресурсів необхідний їй обсяг сировини, після чого нараховує одиницю виготовленого продукту. Ці зміни одразу зберігаються завдяки спроектованій системі синхронізаторів, що були згадані на початку. Якщо фабрика має місце на складі, а також містить сировину на ще один продукт – цикл повторюється. Таким чином, завод працюватиме автоматично при наявних завантажених мінералах.

Додатково, відбувається рендерінг годинника, що демонструє залишковий час на ітерацію виробництва:

```

void LaunchClock()
{
    SetClockActive(true);

    imgClock.fillAmount = 0;
    imgClock.DOFillAmount(1, refData.ProductionSpeed.Value).SetEase(Ease.Linear);
}

```

Логіка роботи шахт дещо відрізняється. Шахта – це чітко визначена зона із певними кордонами, де періодично з’являтиметься об’єкт-руда, яку можна добувати. За активного видобутку мінералів зона може залишитися пустою, тому гравець також повинен зосереджувати сили на відкритті нових родовищ задля постійного джерела прибутку.

Клас містить такі поля:

```

[Space]
[SerializeField] float radius = 5;
[SerializeField] int maxOresCount = 5;
[SerializeField] public float averageHpPerGem = 5;

[Space]
[SerializeField] float minGems = 5;
[SerializeField] float maxGems = 25;

```

Таким чином, записується інформація про радіус шахти, максимальну кількість об’єктів руди, середню кількість урону, яку потрібно нанести для отримання одного кристалу, а також максимальну та мінімальну кількість кристалів всередині об’єкта руди.

Для запуску фабрики потрібен вдубований метод Start(), який одразу ініціалізує її роботу при успішному будівництві:

```

private void Start()

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    refGem = Kernel.LootSynchronizer.Type2Gem(type);
    UnlockResource();

    Init();

    MineZoneManager.Instance?.Add(this);
    StartCoroutine(SpawnOre_Coroutine());
}

```

```

IEnumerator SpawnOre_Coroutine()
{
    yield return new WaitForSeconds(generationTime);

    if (oreList.Count < maxOresCount)
    {
        SpawnOre();
    }

    StartCoroutine(SpawnOre_Coroutine());
}

```

Як ми бачимо, при створенні шахта зчитує інформацію, за який саме тип мінералів вона відповідає, та запускає цикл створення об'єктів руди:

```

public void SpawnOre()
{
    Vector3 spawnPoint = GetSpawnPoint();
    GameObject ore = Kernel.OreFactory.Build(refGem, this);
    float collisionCheckRadius = 1;

    for (int i = 0; i < 3; i++)
    {
        if (Physics.CheckSphere(spawnPoint, collisionCheckRadius))
            spawnPoint = GetSpawnPoint();
    }

    ore.transform.position = spawnPoint;
    ore.transform.parent = this.transform;

    ore.name = $"ore{oreList.Count}";
    ore.isStatic = true;

    oreList.Add(ore.GetComponent<PreciousOre>());
}

```

Ініціалізація нових одиниць мінералів на карті виконується рандомно у межах зони шахти. Далі об'єкт створюється, отримує ім'я та додається в загальний список руд. Далі по цьому списку шахта знатиме, чи був якийсь із мінералів знищений, і якщо так, то з часом вона створить новий.

Після того, як гравець добуває ресурси з шахти, він має можливість продати їх, або принести на відповідний завод для подальшої переробки, після чого продати за більшу ціну. Однак важко розрахувати найоптимальніший

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

варіант заробітку, для чого була додана можливість отримати найкращий план використання добутих ресурсів від помічника, що розраховує поставлену задачу симплекс-методом.

Спочатку здійснюється підготовка обмежень. Для цього цикл має обійти всі присутні заводи, отримати ті, що були розблоковані, та стягнути перелік «рецептів» кожного продукту, що виготовляється:

```
private static List<IngredientsTuple> GetUnlockedRecipes(UnitFactoryWorker[] workers)
{
    List<IngredientsTuple> recipesList = GetAllRecipesList(workers);
    List<IngredientsTuple> unlockedRecipesList = new List<IngredientsTuple>();

    foreach (var item in recipesList)
    {
        if (Kernel.LootSynchronizer.Type2Loot(item.resource).IsUnlocked)
        {
            unlockedRecipesList.Add(item);
        }
    }

    return unlockedRecipesList;
}
```

Так як багато заводів можуть потребувати один і той самий вид мінералів, наступна функція прибирає дублікати:

```
private static List<Gem> GetUniqueIngridientsList(List<IngredientsTuple> recipesList)
{
    var listOfIngridients = new List<MiningLoot>();

    foreach (IngredientsTuple item in recipesList)
    {
        listOfIngridients.AddRange(item.ingredients);
    }

    var uniqueIngridients = listOfIngridients.Select(r => r.type).Distinct().ToList();

    List<Gem> gems = new List<Gem>();
    foreach (var item in uniqueIngridients)
    {
        gems.Add(Kernel.LootSynchronizer.Type2Loot(item));
    }

    gems.OrderBy(i => i.Price);

    return gems;
}
```

Тепер можна приступити до формування системи обмежень [26]:

```
private static double[,] GetMatrixA(List<IngredientsTuple> unlockedRecipesList,
List<Gem> uniqueIngridients)
{
    List<List<double>> matrix = new List<List<double>>();

    List<MiningLoot> allGems = Kernel.PlayerDataSynchronizer.GetCollectedGems();

    foreach (var ingridient in uniqueIngridients)
```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Єфремов Ю.М.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {
        List<double> variables = new List<double>();

        foreach (var product in unlockedRecipesList)
        {
            int ingridientAmount = GetIngredientAmount(product.ingredients,
ingridient.Type);
            variables.Add(ingridientAmount);
        }

        matrix.Add(variables);
    }

    return ConvertListToArray(matrix);
}

private static double[] GetMatrixB(List<IngredientsTuple> unlockedRecipesList,
List<Gem> uniqueIngridients)
{
    List<double> matrix = new List<double>();
    List<MiningLoot> allGems = Kernel.PlayerDataSynchronizer.GetCollectedGems();

    foreach (var ingridient in uniqueIngridients)
    {
        int inBag = GetLootAmount(allGems, ingridient.Type);

        matrix.Add(inBag);
    }

    return matrix.ToArray();
}

```

Обмеження формуються за наступною логікою: зпочатку збирається вся інформація про існуючі заводи у грі, після чого відбувається фільтрація і вибираються лише ті фабрики, що доступні гравцеві. З них дістається інформація про виробництво, а саме кінцевий ресурс та його сировина. Усі види сировини будуть конкатеновані в один масив із уникненням дублікатів, щоб отримати весь список сировини, який присутній у задачі. Далі кожне обмеження заповнюється у вигляді кількості одиниць кожного ресурсу для виробництва певного продукту у запланованому порядку – усі умови повинні містити такий самий порядок ресурсів.

Також формується і цільова функція на основі цін кожного з кінцевих продуктів. При формуванні головної функції обов’язково виконується відповідність послідовності аргументів у обмеженнях:

```

private static double[] GetFunction(List<IngredientsTuple> unlockedRecipesList)
{
    List<double> variables = new List<double>();

    foreach (var item in unlockedRecipesList)
    {
        Gem g = Kernel.LootSynchronizer.Type2Loot(item.resource);
        variables.Add(g.Price);
    }
}

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Єфремов Ю.М.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    return variables.ToArray();
}

```

Отримана побудована задача потрапляє в алгоритм симплекс-методу [27], де відбувається спершу перевірка на успішність побудови рівнянь, а згодом і отримується відповідь до задачі:

```

private static SimpleSimplex PrepareSimplex()
{
    var collectedLootItems = Kernel.PlayerDataSynchronizer.GetCollectedGems();

    if (collectedLootItems.Count > 0)
    {
        UnitFactoryWorker[] workers =
            Resources.LoadAll<UnitFactoryWorker>(Kernel.RootManager.UnitsLocalDataPath);

        UnlockedRecipesList = GetUnlockedRecipes(workers);

        if (UnlockedRecipesList.Count > 0)
        {
            List<Gem> UniqueIngridients =
                GetUniqueIngridientsList(UnlockedRecipesList);

            var matrixA = GetMatrixA(UnlockedRecipesList, UniqueIngridients);
            var matrixB = GetMatrixB(UnlockedRecipesList, UniqueIngridients);

            var func = GetFunction(UnlockedRecipesList);

            return new SimpleSimplex(func, matrixA, matrixB);
        }
        else
        {
            }
        }
    }
    return null;
}

```

Клас, що відповідає за реалізацію програмного симплекс-методу, починає підготовку до побудови першої таблиці:

```

public SimplexMethod(double[] c, double[,] A, double[] b)
{
    int vars = c.Length, constraints = b.Length;

    this.c = new double[vars + constraints];
    Array.Copy(c, this.c, vars);

    this.A = new double[vars + constraints, vars + constraints];
    for (int i = 0; i < constraints; i++)
    {
        for (int j = 0; j < vars; j++)
        {
            this.A[i + vars, j] = A[i, j];
        }
    }

    this.b = new double[vars + constraints];
    Array.Copy(b, 0, this.b, vars, constraints);
}

```

```

for (int i = 0; i < vars; i++)
{
    N.Add(i);
}

for (int i = 0; i < constraints; i++)
{
    B.Add(vars + i);
}
}

```

Обмеження приводяться до канонічного вигляду [28], відбувається заповнення всіх пустих клітинок матриці Таким чином, утворюється початкова таблиця, що готова до наступних ітерацій.

Після цього стартує цикл максимізації отриманої задачі:

```

public Tuple<double, double[]> Maximize()
{
    while (true)
    {
        int e = -1;
        double ce = 0;
        foreach (var _e in N)
        {
            if (c[_e] > ce)
            {
                ce = c[_e];
                e = _e;
            }
        }

        if (e == -1) break;

        double minRatio = double.PositiveInfinity;
        int l = -1;
        foreach (var i in B)
        {
            if (A[i, e] > 0)
            {
                double r = b[i] / A[i, e];
                if (r < minRatio)
                {
                    minRatio = r;
                    l = i;
                }
            }
        }

        if (double.IsInfinity(minRatio))
        {
            return Tuple.Create<double, double[]>(double.PositiveInfinity, null);
        }

        NextIteration(e, l);
    }

    double[] x = new double[b.Length];
    int n = b.Length;
    for (var i = 0; i < n; i++)
    {
        x[i] = B.Contains(i) ? b[i] : 0;
    }
}

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return Tuple.Create<double, double[]>(v, x);
}

```

Всередині знаходяться декілька послідовних циклів. Перший шукає найменше від’ємне число в індексному рядку, що означатиме напрямний стовпець. За його допомогою буде знайдено напрямний рядок, що дасть відповідь на питання про наявність оптимальнішого плану.

Далі необхідно знайти найменше додатне відношення нульового стовпця до напрямного – це і є напрямний рядок. На передині напрямних рядка і стовпця з’являється напрямний елемент, що використовується для обчислення наступної симплекс-таблиці.

Якщо вищеописані етапи було пройдено – отже, існує можливість переходу до більш оптимального розв’язку задачі і відбувається обчислення наступної ітерації:

```

private void NextIteration(int e, int l)
{
    N.Remove(e);
    B.Remove(l);

    b[e] = b[l] / A[l, e];

    foreach (var j in N)
    {
        A[e, j] = A[l, j] / A[l, e];
    }

    A[e, l] = 1 / A[l, e];

    foreach (var i in B)
    {
        b[i] = b[i] - A[i, e] * b[e];

        foreach (var j in N)
        {
            A[i, j] = A[i, j] - A[i, e] * A[e, j];
        }

        A[i, l] = -1 * A[i, e] * A[e, l];
    }

    v = v + c[e] * b[e];

    foreach (var j in N)
    {
        c[j] = c[j] - c[e] * A[e, j];
    }

    c[l] = -1 * c[e] * A[e, l];

    N.Add(l);
    B.Add(e);
}

```

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

В залежності від того, який рядок та який елемент знаходиться на обрахуванні, використовується формула перетворення таблиці до більш оптимального плану. В результаті отримується нова симплекс-таблиця, що знову перевіряється на оптимальність. Цикл відбувається доти, доки не буде знайдено оптимальний розв'язок або ж не буде можливості продовжувати розв'язання, в такому разі задача буде визнана як така, що не має розв'язків.

Після цього метод повертає результат із даними, що містять записаний оптимальний план. Далі клас вікна RorupHint виводить результат у читабельному вигляді – для цього відбувається ітерація по отриманих значеннях функції, відбувається підстановка об'єктів руди та викликання їх назв у тому ж порядку, в якому було передано аргументи функції:

```
private string GetMessage(Tuple<double, double[]> data)
{
    for (int i = 0; i < data.Item2.Length; i++)
    {
        Debug.Log($"X{i + 1} = " + data.Item2[i]);
    }

    string text = "";

    if (data != null)
    {
        text = "You have enough resources to produce some stuff, I recommend to manufacture ";

        int length = data.Item2.Length;

        var allGems = AiHelperGenerator.UnlockedRecipesList;

        for (int i = 0; i < length; i++)
        {
            int x = (int)data.Item2[i];

            if (x > 0) //ignore zero values
            {
                IngredientsTuple t = allGems[i];
                LootType type = t.resource;
                Gem gem = Kernel.LootSynchronizer.Type2Loot(type);

                sum += x * gem.Price;

                if (i == length - 1)
                {
                    text += $"{x} {gem.Title} ";
                }
                else
                {
                    text += $"{x} {gem.Title}, ";
                }
            }
        }
    }
}
```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    text += "and you will earn: ";
}
else
{
    text = "There is nothing to do with your resources, " +
        "so I recommend you to sell them at the base or complete station order.";
}

return text;
}

```

Якщо ж відповідь до задачі є такою, що не містить ніяких допустимих розв’язків або ж гравець не має жодного кристалу у інвентарі – виведеться відповідне повідомлення.

Висновки до другого розділу

Під час проектування та розробки мобільної гри були визначені функціональні та нефункціональні вимоги, бізнес вимоги, а також необхідний функціонал для користувачів. Відповідно до врахованих особливостей програми було спроектовано діаграму класів. Розроблена структура додатку відповідає тій, що була спланована на початку.

		Сидорчук О.С.			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ «КОСМОНАВТ-ЗОЛОТОШУКАЧ»

3.1 Порядок встановлення та налаштування параметрів додатку

Додаток має бути встановлений на девайсі користувача, однак шлях до мобільного пристрою простягається крізь декілька важливих етапів. Ці кроки описані на діаграмі розгортання [29] (рис. 3.1):

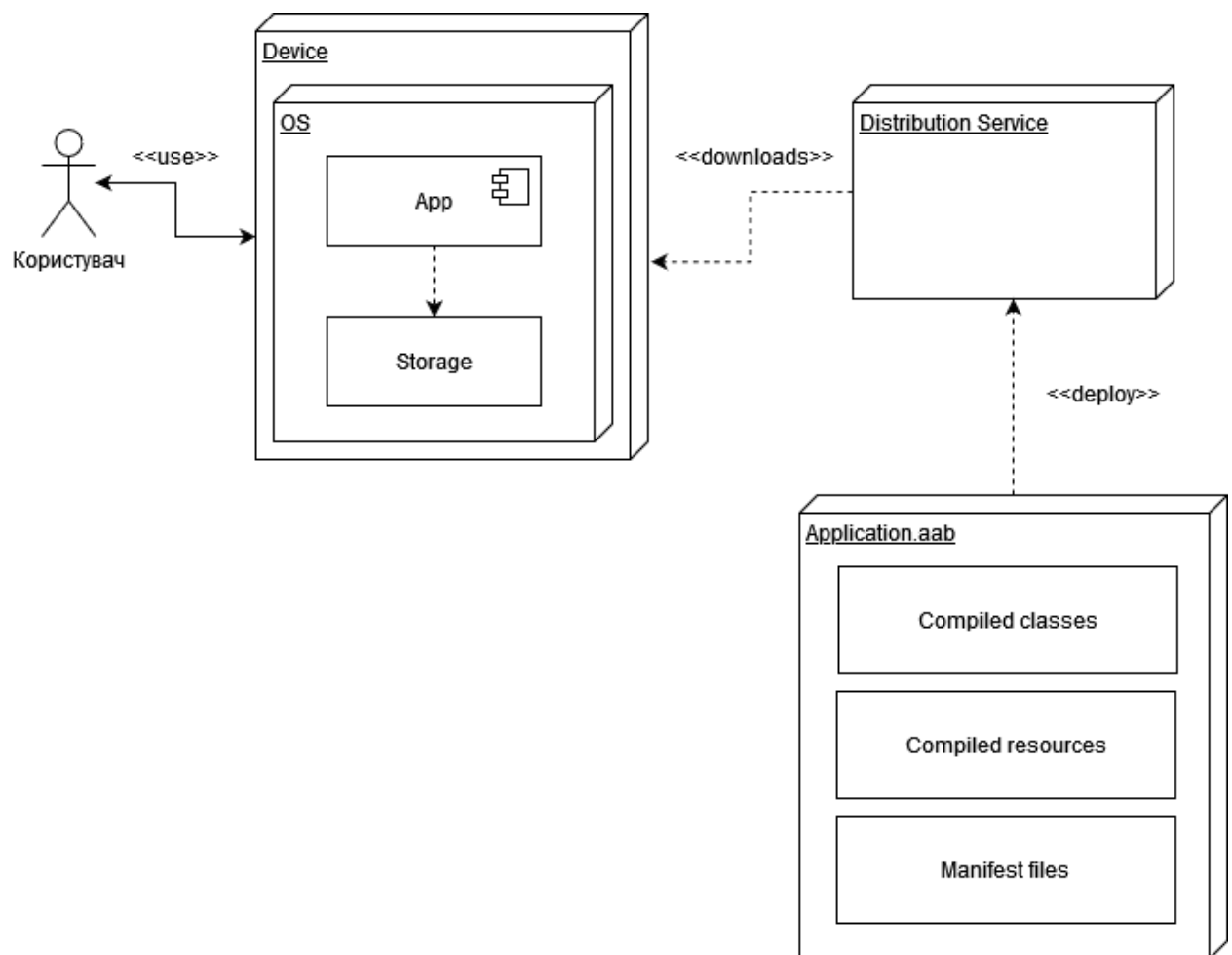


Рис. 3.1. Діаграма розгортання

На діаграмі зображена система, що складається з трьох вузлів: зібраний інсталятор проекту, сервіс для розподілу та сам кінцевий пристрій користувача, на який здійснюється установка гри.

Спочатку розробник повинен зібрати інсталяційний пакет додатку, що здійснюється вбудованими засобами рушія Unity. Згідно з останніми вимогами

магазинів мобільних ігор, файл повинен мати розширення .aab, що сприяє додатковому захисту та зменшенню розміру на диску.

Далі цей файл вивантажується в певний сервіс для розподілу – одним із таких може бути магазин додатків Google Play, де користувач матиме змогу його завантажити [30].

В кінці потенційний гравець скачує інсталятор, що встановлюється на його ОС. Додаток не має жодних залежностей, крім потреби доступу до внутрішнього сховища пристрою, без якого використання гри буде неможливе.

3.2 Структура інтерфейсу додатку та порядок роботи

Так як гра має націлення на саме мобільні пристрої, користувач повинен встановити гру на свій девайс. Гра розрахована на портретну орієнтацію екрану, однак враховує різноманітність роздільної здатності пристроїв та містить адаптивний інтерфейс.

При запуску користувачеві відображається екран завантаження гри (рис. 3.2):

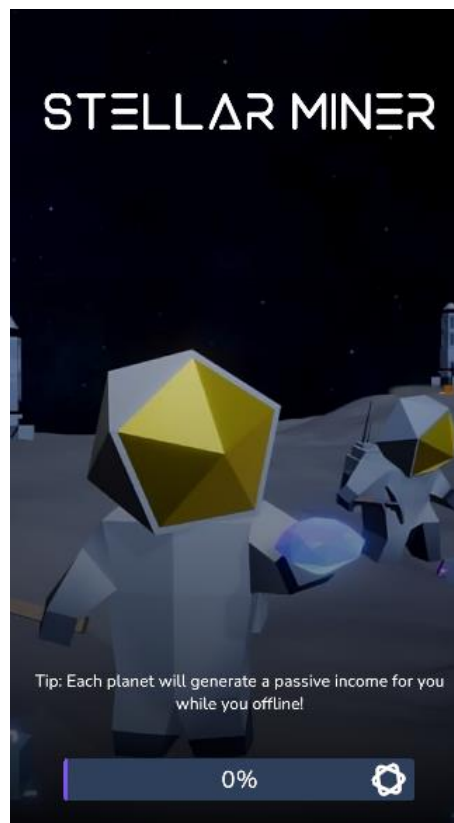


Рис. 3.2. Завантаження

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо це перший запуск – гравцеві відкриється міні-комікс, в якому демонструється сутність гри (рис. 3.3):



Рис. 3.3. Міні-розповідь про персонажа

Далі пропонується пройти навчання у вигляді проходження гри на планеті Земля із підказками, щоб користувач міг швидко розібратись із геймплеєм. Підказки зображуються у вигляді напівпрозорої панелі у верхній частині екрану, що робить поточну задачу помітною, а також не заважає гравцю орієнтуватися на відкритій місцевості. Приклад стартової планети зображено на рис. 3.4:

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

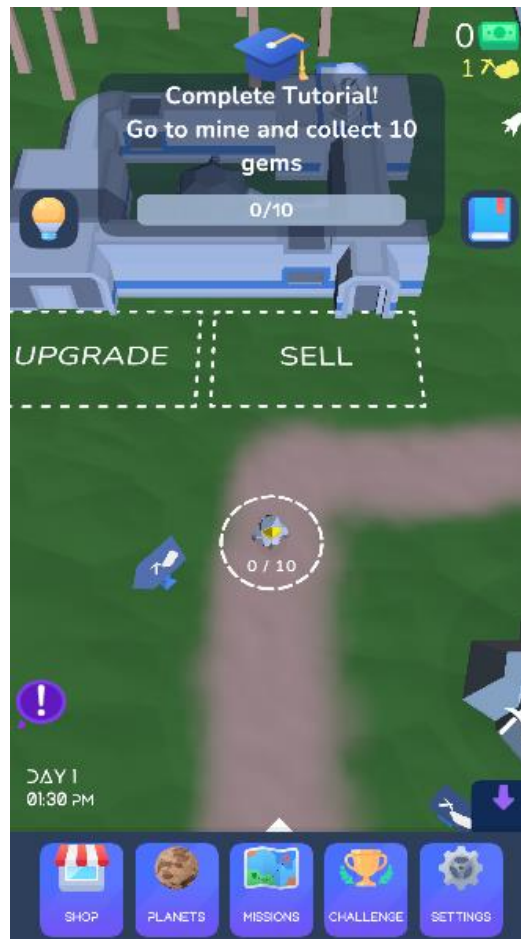


Рис. 3.4. Навчання

У нижній частині екрану є перелік із 5 кнопок (рис. 3.5):

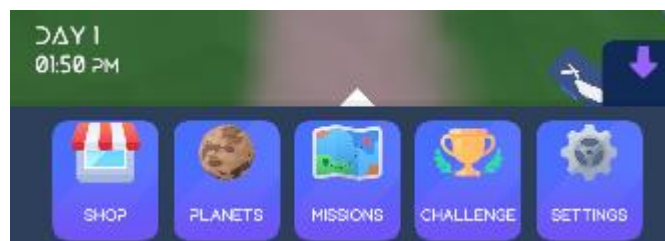


Рис. 3.5. Кнопки меню

Магазин: відкриває панель, де можна здійснити покупку шкінів та ігрової валюти, також за реальні гроші можливо відключити рекламу;

Планети: відкриває вікно із планетами-рівнями, на які планується здійснити політ. Гравець може переглянути, на якій планеті він знаходиться зараз та які будуть доступні далі;

Місії: міні-ігри, які можна проходити раз на день як щоденний бонус та отримувати призи;

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Досягнення: протягом ігрового процесу гравець іноді виконує поставлені завдання, за які можна отримати додаткові ресурси;

Налаштування: можна змінити налаштування гри – вимкнути звуки, музику, змінити якість графіки тощо.

Усі вищезгадані вікна у відповідному порядку (рис. 3.6 – 3.10):

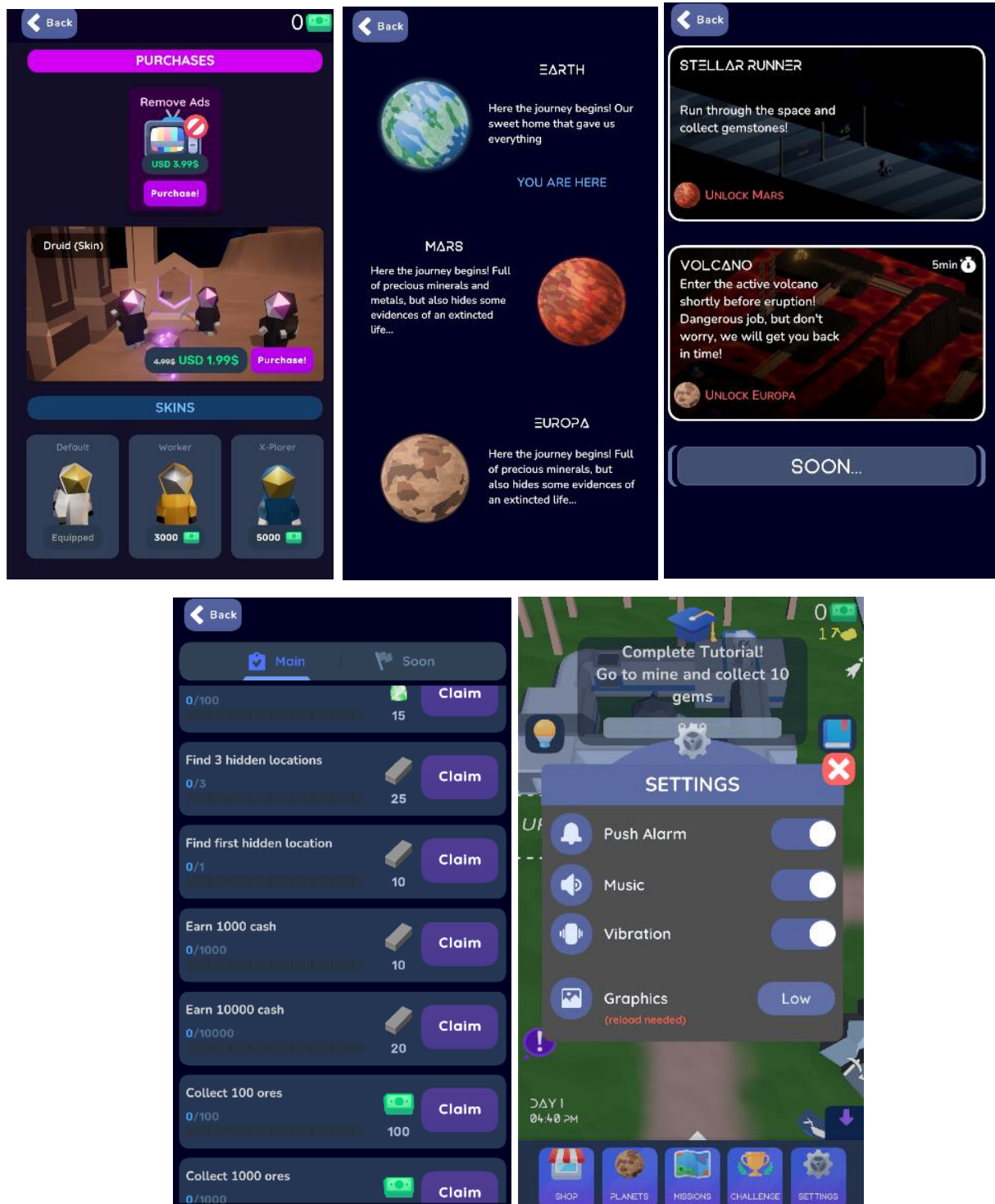


Рис. 3.6-3.10. Вікна нижньої панелі

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Єфремов Ю.М.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Нижню панель можна сховати за необхідності.

Керування гравцем здійснюється одним пальцем за допомогою вбудованого джойстика (рис. 3.11):



Рис. 3.11. Керування

Якщо користувач не відпустив джойстик – гравець не буде взаємодіяти з жодним із навколишніх об'єктів (рис. 3.12):



Рис. 3.12. Персонаж слухає дії гравця

В даному випадку, у зоні досяжності знаходиться об'єкт руди, що може бути видобутий. Також показник заповненості рюкзака має позначку 0/10, що означає «Інвентар має 0 предметів із 10 можливих». Однак при відпусканні джойстика персонаж сам почне добувати руду (рис. 3.13):

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66



Рис. 3.13. Головний герой добуває мінерали

Аналогічно, персонаж зупиниться, якщо його інвентар повний. Якщо ми хочемо дізнатись детальніше про поточний вміст рюкзака, можна глянути у верхній лівий кут екрану (рис. 3.14-3.15):



Рис. 3.14-3.15. Повний інвентар та його вміст

Усі ресурси можуть бути продані на базі. Під час проходження гравець буде відкривати нові мінерали, кожен з яких має власні характеристики важкості видобутку, рідкості та ціни. Гра має чималий спектр можливостей збуту отриманих ресурсів, однак найпростіший та найпопулярніший для гравця варіант є їх продаж. Для продажу потрібно повернутись до бази та зайти на зону «Sell», одразу відмалюється анімація прибутку (рис. 3.16):

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67



Рис. 3.16. Продаж ресурсів

З отриманою ігровою валютою можна зробити чимало вкладень – покращити характеристики персонажа, придбати скін, відкрити нову станцію тощо. На даному етапі туторіал пропонує побудувати шахту. Будівництво також відбувається за допомогою інтерактивних зон – при зіткненні відбудеться зняття грошей та з’явиться нова відкрита ділянка (рис. 3.17-3.18):



Рис. 3.17-3.18. Будівництво шахти

Аналогічно будується і станція – наприклад, в туторіалі пропонується спробувати купити нового робітника. Для цього потрібно побудувати йому житло на цій планеті (рис. 3.19-3.20):



Рис. 3.19-3.20. Будівництво станції для робітника

Тепер ми можемо полегшити собі процес гри – робітник сам буде іти в шахту, добувати ресурси та продавати їх на базі. За потреби, ми можемо покращувати бота.

Однак головною метою гри є будівництво ракети, що дозволить нам перелетіти на наступну планету і почати розвиватись на ній. Аналогічним чином будуються і ступені космічного корабля (рис. 3.21-3.22):



Рис. 3.21-3.22. Етапи будівництва ракети

Після цього навчання завершено, про що свідчить наступне вікрите вікно з відповідним повідомленням (рис. 3.23):

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		69



Рис. 3.23. Кінець туторіалу

Якщо гравець стане на зону запуску ракети, програма запитас його, чи бажає він здійснити політ, і за згоди відбудеться невелика кат-сцена, протягом якої персонажа автоматично перемістить на наступну планету (рис. 3.24):

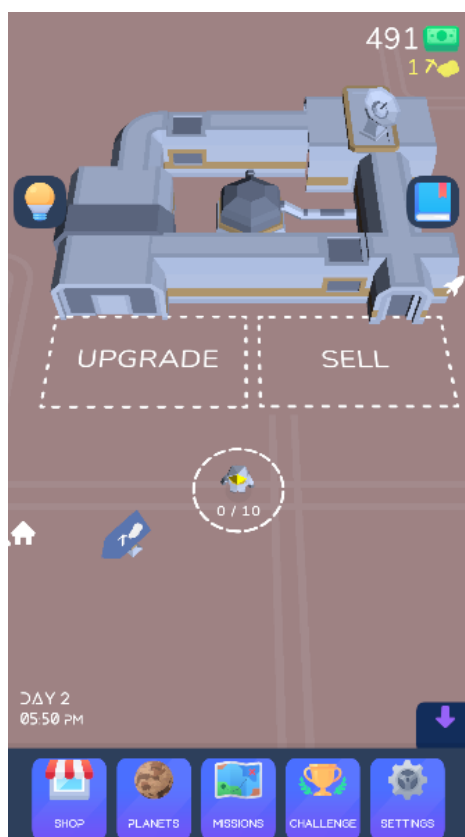


Рис. 3.24. Гравець на Марсі

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

На початку розвитку на новій планеті, гравець мусить покладатися на власні характеристики, так як всі дії в грі йому доведеться робити вручну. Однак при розумному розподіленню коштів, у персонажа буде більше помічників-ботів, що автоматизовуватимуть рутинні задачі. За продуктивної гри користувач вже скоро може отримати перших ботів та доволі розвинену базу (рис. 3.25):



Рис. 3.25. Приклад розвиненої цивілізації

Також за активної гри та кропіткому дослідженню планети і подорожах до найвіддаленіших її куточків гравець періодично знаходитиме сховані міні-сюжетні лінії, де йому доводиться зробити один із двох виборів. Кожн з них дасть певну винагороду, однак варіант, що передбачає рекламне оголошення, має більший приз (рис. 3.26):

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		71



Рис. 3.26. Схований бонус – кістки динозаврів

При подальшому розвитку та великій кількості фабрик може бути важко оцінювати справжню вартість мінералів та їх оптимальний варіант використання. Тому гравець може натиснути на кнопку з емблемою лампи, щоб отримати ідею оптимального виробництва нових товарів з наявних ресурсів. Для цього буде використано програмний симплекс-метод, що проаналізує наявні заводи та їх потреби (рис. 3.27-3.29):

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Єфремов Ю.М.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

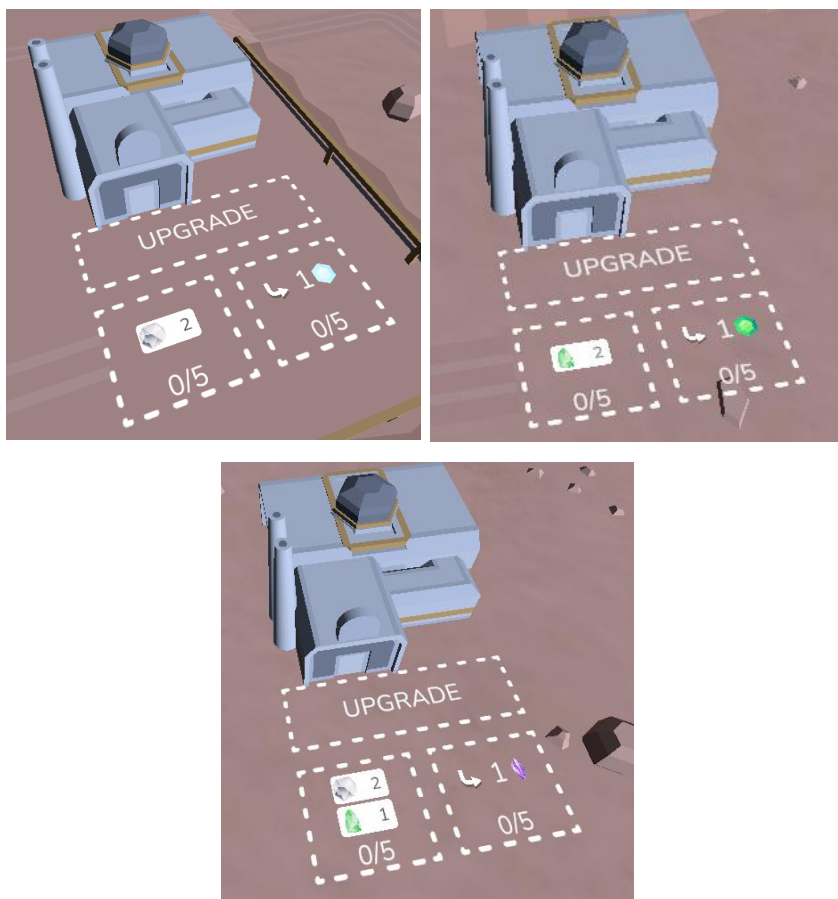


Рис. 3.27-3.29. Присутні відкриті фабрики на планеті

Одна із фабрик потребує 2 кристали кварцу, наступна – 2 кристали смарагдів та третя – 2 шт. кварцу і 1 смарагд.

Наразі користувач добув 12 шт. кварцу та 20 шт. смарагдів, про що свідчить інформація у верхньому лівому куті екрану (рис. 3.30):



Рис. 3.30. Інвентар користувача

Як бачимо, було спроектовано ситуацію, що повністю ідентична задачі із 2 розділу. Протестуємо програмний симплекс-метод(рис. 3.31):

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

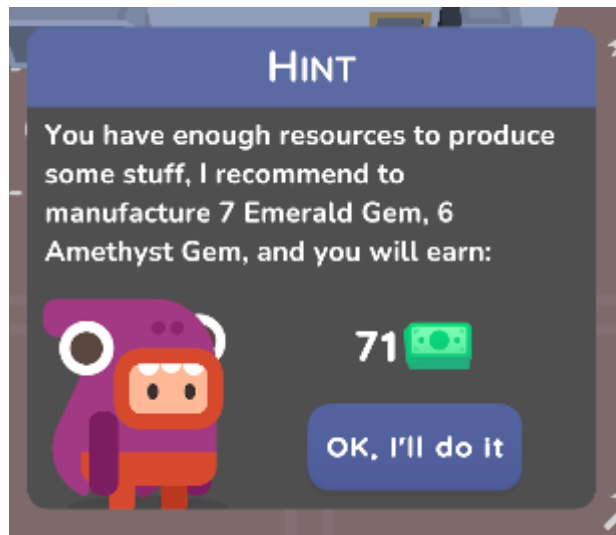


Рис. 3.31. Порада користувачу

Бот радить використати наявні мінерали для виробництва 7 оброблених смарагдів (на заводі №2) та 6 кристалів аметисту (на заводі №3). Також виконано підрахунок кінцевої вартості – 71 од. ігрової валюти. Отже, відповідь співпадає із тією, що було отримано при ручних обчисленнях.

3.3 Тестування роботи додатку

Тестування має вкрай важливу роль при розробці надійного додатку та його справності на пристроях користувачів. Крім того, тестування дозволяє перевірити, чи задовольняє додаток вимогам та очікуванням користувачів.

Тестування повинно проводитись на кожному етапі реалізації програми, починаючи від базових функцій і закінчуючи кінцевим оглядом отриманого ПЗ, так як при великій нагромадженості різноманітних задач на тестування неможливо охопити весь обсяг, не впустивши деякі аспекти чи умови виникнення тих чи інших помилок.

Таким чином, протягом розробки програми було виявлено такі дефекти у її роботі:

1. Гравець натискає на кнопку меню «Досягнення», маючи місткість рюкзака 10.
2. Система винагороджує гравця і дає йому 20 ресурсів Заліза.
3. Гравець має 20 ресурсів Заліза у своєму рюкзаку
4. Гравець підходить до об'єкта руди.

		Сидорчук О.С.			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

Очікуваний результат: персонаж нічого не робитиме, так як має заповнений інвентар.

Фактичний результат: персонаж почав добувати руду.

Проблему було виправлено.

Також було протестовано перельоти гравця на наступні планети:

1. Персонаж заходить в зону запуску ракети.
2. Гравець натискає кнопку «Пуск» у відкритому вікні.
3. Гравець переміщається у ракету і спостерігає за кат-сценою зльоту ракети.
4. Відбувається перехід гри у вікно завантаження.

Очікуваний результат: гравець спостерігає за кат-сценою приземлення на новій планеті, із приземленого зонда з'являється персонаж.

Фактичний результат: помилка виконання – не було знайдено планету з таким іменем.

Даний баг гри було виправлено, а якщо це остання доступна планета – гравець не матиме змоги будувати ракету.

Під кінець розробки алгоритму розрахунку оптимального плану за допомогою симплекс-методу було протестовано правильність відображення результатів користувачеві, і отримано таку помилку:

1. Гравець продає всі товари з інвентарю.
2. Гравець натискає на кнопку отримання підказки у вигляді розрахованого найоптимальнішого плану використання наявних ресурсів.
3. Виводиться вікно з оптимальним планом.

Очікуваний результат: користувач отримує повідомлення про те, що його рюкзак пустий.

Фактичний результат: користувач отримує оптимальний план, за якого його прибуток становитиме 0.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Вищеописаний баг демонструє правильність побудови алгоритму розв’язання задачі симплекс-методом, однак таке повідомлення навряд чи матиме практичне значення для гравця, тому помилку було виправлено.

Висновки до третього розділу

Під час проектування та розробки мобільної гри було продемонстровано отримані можливості програми з огляду користувача. Відповідно до врахованих особливостей програми було спроектовано діаграму варіантів використання системи. Також було протестовано програму на помилки та виявлено деякі несправності, які згодом були виправлені.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В ході розробки кваліфікаційної роботи у вигляді мобільної гри було отримано різноманітні теоретичні навички, які згодом було засвоєно і використано на практиці. На сьогодні сфера розробки ігор розвивається з неймовірною швидкістю, і у даній роботі було продемонстровано вміння розробляти гру, що відноситься до одного з найпопулярніших жанрів. Також в ході розробки було доведено, що створення мобільної гри може потребувати великих обсягів різноманітних ресурсів, а також подекуди і використання нетрадиційних підходів для вирішення певних задач. Націлювання на мобільні пристрої та завчасне планування публікувати отриманий продукт на відповідних платформах забезпечить успішність та прибуток у майбутньому.

Також було продемонстровано можливість застосування симплекс-методу у ігрових проектах як можливість для аналізу поточного рівня розвитку гравця та вираховування оптимального вектору розвитку. Методи математичного програмування є актуальними і мають широке застосування у різних сферах – науці, економіці, а також розробці планів на підприємствах з метою оптимізації виробництва. У даній кваліфікаційній роботі симплекс-метод був застосований задля оптимізації переробки мінералів, що неодмінно збільшить потенційний прибуток гравця.

При виборі інструменту реалізації обрано ігровий рушій Unity, який є чудовим рішенням для 3D та 2D проектів різного масштабу. Для програмування буде використовуватися мова C#, яка зарекомендувала себе як надійна мова високого рівня для розробки додатків.

У першому розділі було проаналізовано завдання та аналогічні мобільні ігри. Порівнюючи зі схожими по стилістиці та геймплею іграми, визначено ключові переваги, які будуть додані у наш продукт. Було розглянуто різні типи архітектури і вибрано найбільш оптимальний варіант. Для нашого проекту найбільш підходящою є монолітна архітектура, що забезпечить стабільність і швидкість.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

У другому розділі було визначено функціональні та нефункціональні вимоги, бізнес вимоги, а також необхідний функціонал для користувачів. Відповідно до врахованих особливостей програми було спроектовано діаграму варіантів використання системи. Було складено алгоритм симплекс-методу, а також розв'язано теоретичну задачу, що цілком можлива для реалізації у додатку. Розроблена структура класів відповідає тій, що була спланована на початку.

У третьому розділі було продемонстровано отримані можливості програми з огляду користувача. Відповідно до врахованих особливостей програми було спроектовано діаграму варіантів використання системи. Також було протестовано програму на помилки та виявлено деякі несправності, які згодом були виправлені.

У підсумку, було спроектовано, розроблено та протестовано готовий програмний продукт, що цілком і повністю може задовольнити потреби потенційних користувачів.

		Сидорчук О.С.			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Art of Game Design: A Book of Lenses / Jesse Schell. - A K Peters/CRC Press, 2019. – 600 с.
2. Game Feel: A Game Designer's Guide to Virtual Sensation / Steve Swink. - Morgan Kaufmann, 2008. – 376 с.
3. Level Up! The Guide to Great Video Game Design / Scott Rogers. - Wiley, 2014. – 552 с.
4. C# 8.0 Pocket Reference: Instant Help for C# 8.0 Programmers / Joseph Albahari, Ben Albahari. - O'Reilly Media, 2019. – 240 с.
5. Refactoring.Guru [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/>
6. Документація по .NET [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/uk-ua/dotnet/>
7. Документація по Unity [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unity.com/>
8. Документація по Android [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/>
9. Документація по IOS [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.apple.com/>
10. Casual Game Design Designing Play for the Gamer in ALL of Us / G. Trefry, 2010. - 325 с.
11. Sparx Systems [Електронний ресурс] – Режим доступу до ресурсу: <https://sparxsystems.com/resources/tutorials/uml2/activity-diagram.html>
12. Visual Paradigm [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/>
13. Creately [Електронний ресурс] – Режим доступу до ресурсу: <https://creately.com/guides/sequence-diagram-tutorial/>
14. Operations Research: Applications and Algorithms / Wayne L. Winston. - Cengage Learning, 2018. - 912 с.

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

- 15.Дослідження операцій. Збірник задач / Зайченко О. Ю., Зайченко Ю. П.. – Київ: Видавничий Дім «Слово», 2007. – 472 с.
- 16.Introduction to Operations Research / Frederick S. Hillier, Gerald J. Lieberman. - McGraw-Hill Education, 2014. - 1088 с.
- 17.Operations Research: Principles and Practice / Ravindran, Phillips, and Solberg. - Wiley, 2006. - 848 с.
- 18.Operations Research: An Introduction / Hamdy A. Taha. - Pearson Education, 2016. - 864 с.
- 19.C# 9 and .NET 5 - Modern Cross-Platform Development / Mark J. Price, 2011. - 822 с.
- 20.C# 9.0 in a Nutshell: The Definitive Reference / Joseph Albahari, Ben Albahari, - O'reilly Media , 2021. – 1060 с.
- 21.Sourcemaking [Електронний ресурс] – Режим доступу до ресурсу: <https://sourcemaking.com/>
- 22.Unity in Action: Multiplatform Game Development in C# / Joe Hocking. - Manning Publications, 2022. – 416 с.
- 23.Unity 2021 Cookbook: Over 140 Recipes to Take Your Unity Game Development Skills to the Next Level / Matt Smith, Shaun Ferns. - Packt Publishing, 2021. – 820 с.
- 24.Learning C# by Developing Games with Unity 2021: Kickstart Your Game Development Journey with Unity / Harrison Ferrone. - Packt Publishing, 2021. – 426 с.
- 25.Unity 2021 Shaders and Effects Cookbook / John P. Doran, Alan Zucconi. - Packt Publishing, 2021. – 464 с.
- 26.C# 8 Quick Syntax Reference / Mikael Olsson. - Apress, 2019. – 234 с.
- 27.Essential C# 8.0 / Mark Michaelis, Eric Lippert. - Addison-Wesley Professional, 2020. – 1056 с.
- 28.AlgLib [Електронний ресурс] – Режим доступу до ресурсу: <https://www.alglib.net/>

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Єфремов Ю.М.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

- 29.Guru99 [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.guru99.com/uk/deployment-diagram-uml-example.html>
- 30.GDevelop [Електронний ресурс] – Режим доступу до ресурсу:
<https://wiki.gdevelop.io/>

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Сидорчук О.С.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

public static class Kernel
{
    public static readonly IRouter RootManager = new RootManager();

    public static readonly ISaver SaveManager = new SaveManager();

    public static readonly IDisplayManager DisplayManager = new DisplayManager();

    public static readonly ITagsManager TagsManager = new TagsManager();

    public static readonly PoolDictionary PoolDictionary = new PoolDictionary();

    public static readonly VibrationController VibrationController = new
VibrationController();

    public static readonly ScenesDictionary ScenesDictionary = new ScenesDictionary();

    public static readonly Env Env = new Env();

    public static readonly AstronautFactory AstronautFactory = new
AstronautFactory(PrefabsAssistant.Instance.AstronautBody);
    public static readonly RoverFactory RoverFactory = new
RoverFactory(PrefabsAssistant.Instance.RoverBody);
    public static readonly RoverFactory DroneFactory = new
RoverFactory(PrefabsAssistant.Instance.DroneBody);
    public static readonly OreFactory OreFactory = new
OreFactory(PrefabsAssistant.Instance.Ore1);

    public static readonly LootSynchronizer LootSynchronizer = new LootSynchronizer();
    public static readonly PlayerDataSynchronizer PlayerDataSynchronizer = new
PlayerDataSynchronizer();
    public static readonly UnitDataSynchronizer UnitDataSynchronizer = new
UnitDataSynchronizer();
    public static readonly UnlocksDataSynchronizer UnlocksDataSynchronizer = new
UnlocksDataSynchronizer();
    public static readonly CutsceneSynchronizer CutsceneSynchronizer = new
CutsceneSynchronizer();
    public static readonly TimeSynchronizer TimeSynchronizer = new TimeSynchronizer();
    public static readonly PlanetSynchronizer PlanetSynchronizer = new
PlanetSynchronizer();
    public static readonly MinigameSynchronizer MinigameSynchronizer = new
MinigameSynchronizer();
    public static readonly TutorialSynchronizer TutorialSynchronizer = new
TutorialSynchronizer();
    public static readonly SettingsSynchronizer SettingsSynchronizer = new
SettingsSynchronizer();
    public static readonly AchievementSynchronizer AchievementSynchronizer = new
AchievementSynchronizer();}
public class SaveManager : ISaver
{

    private Cipher CipherType = new EmptyEncoding();

    public void SetCipher(Cipher cipher)
    {
        CipherType = cipher;
    }

    public string Save(object objToSave, string path)
    {
        string objToJson = JsonUtility.ToJson(objToSave, true);
        objToJson = Encrypt(objToJson);
    }
}

```

```

        File.WriteAllText(path, objToJson);
        return path;
    }

    public string SaveDefault(object objToSave, string path)
    {
        string objToJson = JsonUtility.ToJson(objToSave, true);
        objToJson = Encrypt(objToJson);

        File.WriteAllText(path, objToJson);
        return path;
    }

    public string Load(string path)
    {
        string objJson = File.ReadAllText(path);
        objJson = Decrypt(objJson);

        return objJson;
    }

    public T Load<T>(string path)
    {
        return JsonUtility.FromJson<T>( Load(path) );
    }

    public string Encrypt(string strToEncrypt)
    {
        string decryptedStr = CipherType.Encrypt(strToEncrypt);
        return decryptedStr;
    }

    public string Decrypt(string strToDecrypt)
    {
        string encryptedStr = CipherType.Decrypt(strToDecrypt);
        return encryptedStr;
    }
}

public abstract class Cipher
{
    public abstract string Encrypt(string strToEncrypt);
    public abstract string Decrypt(string strToDecrypt);
}

public class RootManager : IRouter
{
    public string BotsDirectory => "Bots/";
    public string SaveFormat => ".ktefx";
    public string FakeSaveFormat => ".png";

    public string TimeDataExternalPath => Application.persistentDataPath + "/dt" +
    FakeSaveFormat;

    //...
}

public class PlayerDataSynchronizer : AbstractFileSynchronizer
{
    protected override string savePath => Kernel.RootManager.PlayerDataExternalPath;
}

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

//Data
protected PlayerData playerData = new PlayerData();

#region GET-SET
public string GetSkin()
{
    return playerData.Skin;
}

public void SetSkin(string skin)
{
    playerData.Skin = skin;
    SaveData();
}

public List<MiningLoot> GetCollectedGems()
{
    return playerData.CollectedList;
}

public int GetLootCount()
{
    return playerData.CollectedList.Count;
}

public void SetCollectedGems(List<MiningLoot> value)
{
    playerData.CollectedList = value;
    SaveData();
}

public int GetMoney()
{
    return playerData.Money;
}

public void AddMoney(int amount)
{
    playerData.Money += amount;

    InterSceneManager.Instance?.AchievementSystem.OnCoinsCollected?.Invoke(amount);

    SaveData();
}

public bool SubtractMoney(int amount)
{
    if (playerData.Money >= amount)
    {
        playerData.Money -= amount;
        SaveData();

        return true;
    }
    else
    {
        return false;
    }
}

public bool IsRoversOn() => playerData.RoversOn;
public void SetRoversOn(bool state)
{

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        playerData.RoversOn = state;
        SaveData();
    }

    public bool IsRoverUnlocked() => playerData.RoverUnlocked;
    public void UnlockRover()
    {
        if (!playerData.RoverUnlocked)
        {
            playerData.RoverUnlocked = true;
            SaveData();
        }
    }
    public void BlockRover()
    {
        if (playerData.RoverUnlocked)
        {
            playerData.RoverUnlocked = false;
            SaveData();
        }
    }
}
#endregion

public override void Reload()
{
    playerData = Kernel.SaveManager.Load<PlayerData>(savePath);
    Invoke();
}

protected override void SaveData()
{
    Kernel.SaveManager.SaveAsynchronously(playerData, savePath);
    Invoke();
}
}

public abstract class AbstractSynchronizer
{
    protected abstract string savePath { get; }

    public AbstractSynchronizer(bool autoInitialization = false)
    {
        if (autoInitialization)
        {
            Init();
        }
    }

    public abstract void Init();

    public abstract void Reload();

    protected abstract void SaveData();

    public event UnityAction OnDataChanged;

    protected void Invoke()
    {
        OnDataChanged?.Invoke();
    }
}

```

```

public abstract class Unit : ScriptableObject
{
    public string DisplayName = "";
    public string FileName = "";

    public bool IsBought = false;

    private void OnValidate()
    {
        FileName = this.name;
    }

    protected virtual void Save()
    {
        RecountStats();
        Kernel.UnitDataSynchronizer.SaveUnit(this);
    }
    public void ApplyChanges()
    {
        Save();
    }

    public abstract void LoadFromJson(string json);
    public abstract UpgradableValue[] GetUpgradableValues();
    public abstract void UpdateUnitValues(Unit loadedUnit);

    public void RecountStats()
    {
        if (GetUpgradableValues().Length > 0)
        {
            foreach (UpgradableValue variable in GetUpgradableValues())
                variable.RecountValue();
        }
    }

    public virtual void Clear()
    {
        foreach (UpgradableValue variable in GetUpgradableValues())
            variable.Clear();

        IsBought = false;
    }

    public abstract float GetIncomePerMinute();
}

[RequireComponent(typeof(Rigidbody))]
public abstract class UnitBody : MonoBehaviour
{
    public abstract Unit UnitData { get; set; }
    public Collider2D[] BodyColliders { get; private set; }

    protected Rigidbody rb;

    protected bool isPlayer = false;
    public void SetIsPlayer()
    {
        isPlayer = true;
    }

    protected virtual void Start()
    {
        BodyColliders = GetComponentsInChildren<Collider2D>();
        rb = GetComponent<Rigidbody>();
    }
}

```

```

        Kernel.UnitDataSynchronizer.OnUnitUpgraded += OnUnitUpgraded;
    }

    public abstract Type GetRefClass();

    public abstract void OnUnitUpgraded(Type type);

    private void OnDestroy()
    {
        Kernel.UnitDataSynchronizer.OnUnitUpgraded -= OnUnitUpgraded;
    }
}

protected override void Start()
{
    base.Start();

    OnOreCollided += CrystalOreCollision;
    OnOreCollisionExit += CrystalOreUncollided;
    OnLootAmountChanged += UiUpdate;

    SetPickaxeActive(false);
    UiUpdate();
}

private void CrystalOreCollision(PreciousOre ore)
{
    if (GetTarget() != null && HasPlaceInBag())
    {
        if (CanInteractWithUI())
        {
            StartMining();
        }
        else
        {
            StopMining();
        }
    }
}

private void CrystalOreUncollided()
{
    StopMining();
}

public override void StopMining()
{
    SetIsMining(false);
    if (currentAnimaton != AnimMove)
        ChangeAnimation(AnimIdle);
    SetPickaxeActive(false);
}

public override void StartMining()
{
    SetIsMining(true);
    ChangeAnimation(AnimMine);
    SetPickaxeActive(true);
}

public void Mine(IMineable ore)
{
    MiningLoot gem = ore.Mine(_data.MiningDamage.Value, transform.position, isPlayer);
    AddToLoot(gem);
}

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

        OnLootAmountChanged?.Invoke();

        if (!HasPlaceInBag())
            StopMining();

        StartCoroutine(CheckIfNull_Coroutine());
    }

    public MiningLoot Mine(float damage, Vector3 minerPos, bool isPlayer)
    {
        int knockedGems = 0;
        if (damage >= hp)
        {
            knockedGems = resourceAmount;
        }
        else
        {
            knockedGems = (int)(1.0f * damage * resourceAmount / hp);
        }

        hp -= damage;
        resourceAmount -= knockedGems;

        if (hp <= 0)
        {
            if(isPlayer)
                GlobalManagerHolder.Instance.BoosterManager.TryChanceBooster(transform);

            OnHpEnd();
            Destroy(gameObject);
        }

        MiningAnim(minerPos);

        return new MiningLoot(OreData.Type, knockedGems);
    }

    public abstract class GroundUI : MonoBehaviour
    {
        public abstract void OnObjectCollision(Collider collider);
        public abstract void OnObjectCollisionExit(Collider collider);

        private void OnTriggerEnter(Collider other)
        {
            OnObjectCollision(other);
        }

        private void OnTriggerExit(Collider other)
        {
            OnObjectCollisionExit(other);
        }

        public virtual void Render()
        {
        }
    }
}

```

```

public class SellZone : GroundUI
{
    List<UnitLooterBody> sellers = new List<UnitLooterBody>();
    public event UnityAction<int> OnItemsSold;

    public override void OnObjectCollision(Collider collider)
    {
        if (collider.TryGetComponent(out UnitLooterBody unitMiner))
        {
            SellLootOf(unitMiner);
        }
    }

    private void SellLootOf(UnitLooterBody unitMiner)
    {
        int sum = 0;

        var list = unitMiner.GetLoot().List;
        if (list != null)
        {
            foreach (MiningLoot item in list)
            {
                Gem g = Kernel.LootSynchronizer.Type2Loot(item.type);
                sum += g.Price * item.amount;
            }
        }

        unitMiner.ClearLoot(true);

        if (sum != 0)
        {
            Kernel.PlayerDataSynchronizer.AddMoney(sum);
            SpawnIncomePopup(sum, unitMiner.transform.position);

            OnItemsSold?.Invoke(sum);
        }
    }

    private void SpawnIncomePopup(float income, Vector3 pos)
    {
        GameObject g = GlobalManagerHolder.Instance.ControllerPool
            .Spawn(Kernel.PoolDictionary.IncomeItem, pos, Quaternion.identity);
        g.GetComponent<IncomeUiItem>().Setup(income);

        GlobalManagerHolder.Instance.ControllerPool.PlaySound(
            SoundAssistant.Instance.Cash,
            transform.position, 0.5f);
    }

    public override void OnObjectCollisionExit(Collider collider)
    {
        if (collider.TryGetComponent(out UnitMinerBody unitMiner))
        {
            sellers.Remove(unitMiner);
        }
    }
}

public class FactoryUpgradeZone : AbstractBuildingUpgradeZone
{

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        protected override AbstractPopup Popup { get => popup; set => popup =
(PopupFactoryUpgrade)value; }

        [SerializeField] PopupFactoryUpgrade popup;

        protected override void Start()
        {
            popup =
GlobalManagerHolder.Instance.PanelManager.CreatePopup<PopupFactoryUpgrade>(popup);

            UnitResourceGenerator refUnitData =
(UnitResourceGenerator)building.GetUnitData();

            popup.SetRefUnitData(refUnitData);
            popup.SetHeader(refUnitData.DisplayName);

            base.Start();
        }
    }

    public abstract class Unlockable : MonoBehaviour
    {
        [SerializeField] public int UnlocksAt = 1;

        [SerializeField] protected int UnlockPrice = 500;
        public int GetUnlockPrice() => UnlockPrice;
        public void SetTestUnlockPrice(int i)
        {
            UnlockPrice = i;
        }

        [SerializeField] protected int MoneyPaid = 0;
        public int GetMoneyPaid() => MoneyPaid;
        public int GetMoneyLeft() => UnlockPrice - MoneyPaid;
        public void SetMoneyPaid(int a) => MoneyPaid = a;
        public abstract void Pay(int a);

        [Header("States")]
        [SerializeField] internal GameObject CaseLocked;
        [SerializeField] internal GameObject CaseUnlocked;

        protected bool IsReady = false;

        public void SetUnlocksAtEditor(int val)
        {
            UnlocksAt = val;
        }

        public virtual void SetLocked(bool state)
        {
            CaseLocked.SetActive(state);
            CaseUnlocked.SetActive(!state);

            IsReady = !state;
        }

        public virtual void SetUnavaliable()
        {
            CaseLocked.SetActive(false);
            CaseUnlocked.SetActive(false);

            IsReady = false;
        }
    }

```

```

    public abstract void OnBought();
}

public abstract class UnlockableBuilding : Unlockable
{
    [Space(5)]
    [SerializeField] internal BuyBuildingZone zone;

    public abstract Unit GetUnitData();

    public override void Pay(int a)
    {
        if (!IsReady)
        {
            MoneyPaid += a;
            Kernel.UnlocksDataSynchronizer.SetBuildingMoneyPaid(MoneyPaid);

            if (GetMoneyLeft() <= 0)
            {
                SetLocked(false);
                GlobalManagerHolder.Instance.BuildingUnlockManager.BuyBuilding();
                OnBought();
            }
        }
    }

    protected virtual void Awake()
    {
        GlobalManagerHolder.Instance.BuildingUnlockManager.Add(this);
    }

    protected virtual void Start()
    {
        zone?.SetBuilding(this);

        if(IsReady)
            OnBought();
    }
}

public abstract class UnlockableMine : Unlockable
{
    [Space(5)]
    [SerializeField] protected BuyMineZone zone;
    [SerializeField] int minPickaxeLevel;

    protected virtual void Awake()
    {
        GlobalManagerHolder.Instance.MineUnlockManager.Add(this);
    }

    public override void Pay(int a)
    {
        if (!IsReady)
        {
            MoneyPaid += a;
            Kernel.UnlocksDataSynchronizer.SetMineMoneyPaid(MoneyPaid);

```

```

        if (GetMoneyLeft() <= 0)
        {
            SetLocked(false);
            GlobalManagerHolder.Instance.MineUnlockManager.BuyBuilding();
            OnBought();
        }
    }

protected virtual void Start()
{
    zone?.SetBuilding(this);
    zone?.SetMinPickaxeLevel(minPickaxeLevel);

    if (IsReady)
        OnBought();
}

[SerializeField] UnitFactoryWorker refData;
public override void OnBought()
{
    OnStatsChanged += Render;
    SubscribeToUnitUpgrades();
    Invoke();

    SetSmoke(false);

    if (CanProduce())
        LaunchProcess(true);
    else
    {
        ClearClock();
        StopProcess();
    }

    UnlockResource();
}

public IEnumerator ProductionProcess_Coroutine()
{
    yield return new WaitForSeconds(refData.ProductionSpeed.Value);

    Produce();
    Invoke();

    refData.ApplyChanges();

    StopProcess();

    if (CanProduce())
        LaunchProcess();
}

void LaunchClock()
{
    SetClockActive(true);

    imgClock.fillAmount = 0;

```

		Сидорчук О.С.			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Єфремов Ю.М.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        imgClock.DOFillAmount(1, refData.ProductionSpeed.Value).SetEase(Ease.Linear);
    }

    [Space]
    [SerializeField] float radius = 5;
    [SerializeField] int maxOresCount = 5;
    [SerializeField] public float averageHpPerGem = 5;

    [Space]
    [SerializeField] float minGems = 5;
    [SerializeField] float maxGems = 25;

    private void Start()
    {
        refGem = Kernel.LootSynchronizer.Type2Gem(type);
        UnlockResource();

        Init();

        MineZoneManager.Instance?.Add(this);
        StartCoroutine(SpawnOre_Coroutine());
    }

    IEnumerator SpawnOre_Coroutine()
    {
        yield return new WaitForSeconds(generationTime);

        if (oreList.Count < maxOresCount)
        {
            SpawnOre();
        }

        StartCoroutine(SpawnOre_Coroutine());
    }

    public void SpawnOre()
    {
        Vector3 spawnPoint = GetSpawnPoint();
        GameObject ore = Kernel.OreFactory.Build(refGem, this);
        float collisionCheckRadius = 1;

        for (int i = 0; i < 3; i++)
        {
            if (Physics.CheckSphere(spawnPoint, collisionCheckRadius))
                spawnPoint = GetSpawnPoint();
        }

        ore.transform.position = spawnPoint;
        ore.transform.parent = this.transform;

        ore.name = $"ore{oreList.Count}";
        ore.isStatic = true;

        oreList.Add(ore.GetComponent<PreciousOre>());
    }

```

```

private static List<IngredientsTuple> GetUnlockedRecipes(UnitFactoryWorker[] workers)
{
    List<IngredientsTuple> recipesList = GetAllRecipesList(workers);
    List<IngredientsTuple> unlockedRecipesList = new List<IngredientsTuple>();

    foreach (var item in recipesList)
    {
        if (Kernel.LootSynchronizer.Type2Loot(item.resource).IsUnlocked)
        {
            unlockedRecipesList.Add(item);
        }
    }

    return unlockedRecipesList;
}

static double[,] ConvertListToArray(List<List<double>> list)
{
    int rows = list.Count;
    int cols = list[0].Count;

    double[,] array = new double[rows, cols];

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            array[i, j] = list[i][j];
        }
    }

    return array;
}

private static double[,] GetMatrixA(List<IngredientsTuple> unlockedRecipesList,
List<Gem> uniqueIngridients)
{
    List<List<double>> matrix = new List<List<double>>();

    List<MiningLoot> allGems = Kernel.PlayerDataSynchronizer.GetCollectedGems();

    foreach (var ingridient in uniqueIngridients)
    {
        List<double> variables = new List<double>();

        foreach (var product in unlockedRecipesList)
        {
            int ingridientAmount = GetIngredientAmount(product.ingredients,
ingridient.Type);
            variables.Add(ingridientAmount);
        }

        matrix.Add(variables);
    }

    return ConvertListToArray(matrix);
}

private static double[] GetMatrixB(List<IngredientsTuple> unlockedRecipesList,
List<Gem> uniqueIngridients)
{
    List<double> matrix = new List<double>();

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        List<MiningLoot> allGems = Kernel.PlayerDataSynchronizer.GetCollectedGems();

        foreach (var ingradient in uniqueIngridients)
        {
            int inBag = GetLootAmount(allGems, ingradient.Type);

            matrix.Add(inBag);
        }

        return matrix.ToArray();
    }

private static double[] GetFunction(List<IngredientsTuple> unlockedRecipesList)
{
    List<double> variables = new List<double>();

    foreach (var item in unlockedRecipesList)
    {
        Gem g = Kernel.LootSynchronizer.Type2Loot(item.resource);
        variables.Add(g.Price);
    }

    return variables.ToArray();
}

private static SimpleSimplex PrepareSimplex()
{
    var collectedLootItems = Kernel.PlayerDataSynchronizer.GetCollectedGems();

    if (collectedLootItems.Count > 0)
    {
        UnitFactoryWorker[] workers =
Resources.LoadAll<UnitFactoryWorker>(Kernel.RootManager.UnitsLocalDataPath);

        UnlockedRecipesList = GetUnlockedRecipes(workers);

        if (UnlockedRecipesList.Count > 0)
        {
            List<Gem> UniqueIngridients =
GetUniqueIngridientsList(UnlockedRecipesList);

            var matrixA = GetMatrixA(UnlockedRecipesList, UniqueIngridients);
            var matrixB = GetMatrixB(UnlockedRecipesList, UniqueIngridients);

            var func = GetFunction(UnlockedRecipesList);

            return new SimpleSimplex(func, matrixA, matrixB);
        }
        else
        {
        }
    }

    return null;
}

public SimplexMethod(double[] c, double[,] A, double[] b)
{
    int vars = c.Length, constraints = b.Length;

    if (vars != A.GetLength(1))
    {
        throw new Exception("Number of variables in c doesn't match number in A.");
    }
}

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

if (constraints != A.GetLength(0))
{
    throw new Exception("Number of constraints in A doesn't match number in b.");
}

this.c = new double[vars + constraints];
Array.Copy(c, this.c, vars);

this.A = new double[vars + constraints, vars + constraints];
for (int i = 0; i < constraints; i++)
{
    for (int j = 0; j < vars; j++)
    {
        this.A[i + vars, j] = A[i, j];
    }
}

this.b = new double[vars + constraints];
Array.Copy(b, 0, this.b, vars, constraints);

for (int i = 0; i < vars; i++)
{
    N.Add(i);
}

for (int i = 0; i < constraints; i++)
{
    B.Add(vars + i);
}
}

public Tuple<double, double[]> Maximize()
{
    while (true)
    {
        int e = -1;
        double ce = 0;
        foreach (var _e in N)
        {
            if (c[_e] > ce)
            {
                ce = c[_e];
                e = _e;
            }
        }

        if (e == -1) break;

        double minRatio = double.PositiveInfinity;
        int l = -1;
        foreach (var i in B)
        {
            if (A[i, e] > 0)
            {
                double r = b[i] / A[i, e];
                if (r < minRatio)
                {
                    minRatio = r;
                    l = i;
                }
            }
        }

        if (double.IsInfinity(minRatio))

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            return Tuple.Create<double, double[]>(double.PositiveInfinity, null);
        }

        NextIteration(e, l);
    }

    double[] x = new double[b.Length];
    int n = b.Length;
    for (var i = 0; i < n; i++)
    {
        x[i] = B.Contains(i) ? b[i] : 0;
    }

    return Tuple.Create<double, double[]>(v, x);
}

```

```

private void NextIteration(int e, int l)
{
    N.Remove(e);
    B.Remove(l);

    b[e] = b[l] / A[l, e];

    foreach (var j in N)
    {
        A[e, j] = A[l, j] / A[l, e];
    }

    A[e, l] = 1 / A[l, e];

    foreach (var i in B)
    {
        b[i] = b[i] - A[i, e] * b[e];

        foreach (var j in N)
        {
            A[i, j] = A[i, j] - A[i, e] * A[e, j];
        }

        A[i, l] = -1 * A[i, e] * A[e, l];
    }

    v = v + c[e] * b[e];

    foreach (var j in N)
    {
        c[j] = c[j] - c[e] * A[e, j];
    }

    c[l] = -1 * c[e] * A[e, l];

    N.Add(l);
    B.Add(e);
}

```

```

private string GetMessage(Tuple<double, double[]> data)
{
    for (int i = 0; i < data.Item2.Length; i++)
    {
        Debug.Log($"X{i + 1} = " + data.Item2[i]);
    }
}

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		98

```

        string text = "";

        if (data != null)
        {
            text = "You have enough resources to produce some stuff, I recommend to
manufacture ";

            int length = data.Item2.Length;

            var allGems = AiHelperGenerator.UnlockedRecipesList;

            for (int i = 0; i < length; i++)
            {
                int x = (int)data.Item2[i];

                if (x > 0)
                {
                    IngredientsTuple t = allGems[i];
                    LootType type = t.resource;
                    Gem gem = Kernel.LootSynchronizer.Type2Loot(type);

                    sum += x * gem.Price;

                    if (i == length - 1)
                    {
                        text += $"{x} {gem.Title} ";
                    }
                    else
                    {
                        text += $"{x} {gem.Title}, ";
                    }
                }
            }

            text += "and you will earn:";
        }
        else
        {
            text = "There is nothing to do with your resources, " +
                "so I recommend you to sell them at the base or complete station order.";
        }

        return text;
    }

    public class AstronautPlayerInput : AbstractInput
    {
        private PlayerDefaultView View;

        internal override UnitBody Body { get => body; set => body = (AstronautBody)value; }

        private AstronautBody body;

        public void SetView(PlayerDefaultView View)
        {
            this.View = View;
            View.SetTarget(body.GetFollowPoint());
        }
    }

```

```

public override ControlType GetControlType()
{
    return ControlType.Player;
}

protected override void Start()
{
    base.Start();
    SetView((PlayerDefaultView)Kernel.DisplayManager.GetPlayerView());

    if (IsViewExists())
    {
        //Events
        View.joystick.OnPressed.AddListener(OnJoystickPress);
        View.joystick.OnReleased.AddListener(OnJoystickRelease);
    }
}
}

```

		Сидорчук О.С.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		100