МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна інженерія»

Група: 2БКС-27

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

здобувача освіти денної форми навчання

БКС.27.04.000.КРБ

БІЛОКАМЕНСЬКОГО СЕРГІЯ ОЛЕГОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітня програма: **«Комп'ютерна інженерія»**

Група: **2БКС-27**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему:

«Створення ігрового проекту – квесту на платформі Unity: розробка гейм дизайну, системного дизайну та дизайну інтерфейсу»

Проектний матеріал складається з пояснювальної записки на 80 сторінках та графічного (презентаційного) матеріалу на 12 аркушах (слайдах)

Виконавець _____ (Білокаменський С.О.)

Керівник проекту _____ (Іванова Л.В.)

Консультанти:

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувачка кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «20» 06 2023 р. Протокол ДКК № 1

Оцінка ЕК 5 (відмінно)

Секретар ДКК _____

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Створення ігрового проекту – квесту на платформі Unity: розробка гейм дизайну, системного дизайну та дизайну інтерфейсу».

У цій роботі описаний принцип розробки ігрового проекту – квесту на платформі Unity.

У технологічній частині аналізуються існуючі ігрові жанри та ігрові двигуни, створюється технічне завдання до проекту, також реалізується головне меню, сцена з розповіддю сюжету та дизайн головної сцени, проводиться тестування ігрового проекту. Останнім розділом стала охорона праці під час розробки та тестування проекту.

Ключові слова: Unity, дизайн, інтерфейс, ігрові жанри, ефекти, C#.

ANNOTATION

The topic of the bachelor's thesis is "Creating a game project - a quest on the Unity platform: development of game design, system design and interface design".

This paper describes the principle of developing a game project - a quest on the Unity platform.

The technological part analyzes the existing game genres and game engines, creates a technical task for the project, implements the main menu, a scene with a story and the design of the main scene, and tests the game project. The last section is occupational health and safety during project development and testing.

Keywords: Unity, design, interface, game genres, effects, C#.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Кафедра *комп'ютерної інженерії*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР ***Беркань І.В.***

"_____" _____ 202___р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

Здобувачеві (здобувачці) освіти ***Білокаменський Сергій Олегович***

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи

Створення ігрового проекту – квесту на платформі Unity: розробка гейм дизайну, системного дизайну та дизайну інтерфейсу

затверджена наказом по коледжу від "17" жовтня 202__р. № 235-АД-ОД

2. Термін здачі кваліфікаційної роботи 20.06.2023р.

3. Вихідні дані до роботи

1. Вимоги щодо реалізації ігрових механік

2. Вимоги до системного дизайну

3. Вимоги до ігрового дизайну

4. Вимоги до дизайну інтерфейсу

5. Мінімальні системні вимоги

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Передмова. Аналіз ігрових жанрів. Аналіз ігрових двигунів. Розробка технічного завдання для ігрового проекту. Створення ігрового проекту. Тестування ігрового проекту. Охорона праці. Висновки. Перелік використаних джерел.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

1. Реалізація головного меню

2. Реалізація початкової кат-сцени

3. Реалізація головної сцени

4. Реалізація системи заводів та складів

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів роботи, що стосується їх

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний	*Іванова Л.В.*		
Охорона праці	*Чорновол В.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання *30.11.2023*

Керівник роботи *Іванова Л.В.*
(підпис)

Завдання прийняв до виконання *Білокаменський С.О.*
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Передмова*	*4.05.2023*	Виконав
2.	*Аналіз ігрових жанрів*	*8.05.2023*	Виконав
3.	*Аналіз ігрових двигунів*	*10.05.2023*	Виконав
4.	*Розробка технічного завдання для ігрового проекту*	*15.05.2023*	Виконав
5.	*Створення ігрового проекту*	*17.05.2023*	Виконав
6.	*Тестування ігрового проекту*	*28.05.2023*	Виконав
7.	*Охорона праці*	*30.05.2023*	Виконав
8.	*Висновки*	*1.06.2023*	Виконав
9.	*Перелік використаних джерел*	*2.06.2023*	Виконав
10.	*Оформлення пояснювальної записки*	*3.06.2023*	Виконав
11.	*Оформлення презентаційного матеріалу*	*12.06.2023*	Виконав
12.	*Малий захист кваліфікаційної роботи*	*15.06.2023*	Виконав

Виконавець *Білокаменський С.О.*
(підпис)

Керівник роботи *Іванова Л.В.*
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 27. 04 000. 00 КРБ	*Кваліфікаційна робота*		
А4			*БКС 27. 04 000. 00 КРБ ПЗ*	*Пояснювальна*	*1*	
				записка		

Зм.	Арк.	№ докум.	Підпис	Дата	*БКС 27. 04 000. 00 КРБ ПЗ*		
Розробив		Білокаменський С.О.			Створення ігрового проекту – квесту на платформі Unity: розробка гейм дизайну, системного дизайну та дизайну інтерфейсу	Літ.	Аркуш 6 / Аркушів 80
Перевірив		Іванова Л.В.					
Н. Контр.		Петрашова В.І.				ВСП «ОТФК ОНТУ» гр.2БКС-27	
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

За останні кілька десятиліть ігрова індустрія зазнала значного зростання і трансформації, перетворившись на багатомільярдну глобальну індустрію, яка конкурує і навіть перевершує за доходами інші форми розваг, такі як кіно і музика. Таке зростання можна пояснити різними факторами, зокрема розвитком технологій, появою платформ цифрової дистрибуції та зростаючою популярністю ігор серед різноманітної аудиторії.

Технологічний прогрес відіграв вирішальну роль у формуванні ігрової індустрії. Розвиток потужного обладнання, такого як ігрові консолі, високопродуктивні комп'ютери та мобільні пристрої, дозволив розробникам створювати більш візуально приголомшливі та захоплюючі ігри. Крім того, наявність сучасних ігрових двигунів, таких як Unity та Unreal Engine, дала змогу розробникам будь-якого масштабу створювати високоякісні ігри без потреби в глибоких знаннях програмування.

Поява платформ цифрової дистрибуції, таких як Steam, PlayStation Network, Xbox Live та магазини мобільних додатків, докорінно змінила спосіб розповсюдження та споживання ігор. Ці платформи полегшили незалежним розробникам вихід їхніх ігор на широку аудиторію, оминаючи традиційні бар'єри, пов'язані з фізичними каналами дистрибуції.

Ігрова індустрія стала свідком значного розширення своєї цільової аудиторії. Ігри більше не обмежуються нішевою групою ентузіастів, а стали масовою формою розваг, якою користуються люди різного віку, статі та походження. Доступність і різноманітність ігрового досвіду сприяли такому широкому розповсюдженню. Від казуальних мобільних ігор до складних і захоплюючих ААА-тайтлів, існує гра для будь-якого інтересу та рівня навичок.

Одним із популярних жанрів в ігровій індустрії є квест-жанр, який фокусується на дослідженні, вирішенні головоломок та розповіді історій. Ігри-квести пропонують гравцям захопливу розповідь і спонукають їх виконувати різноманітні завдання чи цілі для проходження гри. Ці ігри часто мають

складний дизайн рівнів, цікавих персонажів і головоломки, що спонукають до роздумів, пропонуючи гравцям унікальний ігровий досвід із повним зануренням.

Жанр квестів значно еволюціонував з моменту свого зародження, і розробники постійно прагнуть розширити межі можливого в цьому жанрі. Однак, незважаючи на свою популярність, жанр квесту все ще стикається з певними викликами та проблемами.

Однією з головних проблем в ігровій індустрії є створення привабливого ігрового дизайну, який би захоплював гравців і утримував їх впродовж усього ігрового процесу. Ігровий дизайн передбачає розробку загального бачення та механіки гри, гарантуючи, що вона пропонує баланс між викликом та задоволенням. Дизайнери повинні створювати інноваційні та захопливі квести, які дають гравцям відчуття прогресу та винагороди.

Інший виклик полягає в системному дизайні, який фокусується на основних механіках і технічних аспектах гри. Розробники повинні створювати ефективні та оптимізовані системи, які забезпечують безперебійний ігровий процес, без технічних збоїв або проблем з продуктивністю. Це включає проектування систем штучного інтелекту, фізичне моделювання та реалізацію інтуїтивно зрозумілих елементів керування.

Дизайн інтерфейсу - ще один важливий аспект у розробці ігор. Він передбачає створення зручних інтерфейсів, які дозволяють гравцям взаємодіяти з грою без особливих зусиль. Добре продуманий інтерфейс гарантує, що гравці зможуть легко орієнтуватися в меню, отримувати доступ до ігрових функцій та отримувати важливий зворотній зв'язок під час гри. Дизайн інтерфейсу відіграє важливу роль у покращенні загального користувацького досвіду та зануренні гравців у ігровий світ.

У даній кваліфікаційній роботі буде розроблений ігровий проект – квест на платформі Unity. Також реалізовану гру можна поширювати серед гравців, виклавши її до платформ цифрової дистрибуції.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Аналіз ігрових жанрів

У світі існує широкий спектр ігрових жанрів, які задовольняють різні людські інтереси та ігровий досвід. Необхідно розглянути існуючі жанри ігор, виявити їх унікальні характеристики, переваги та недоліки.

1.1.1 Ігровий жанр Action

Жанр екшену характеризується динамічним ігровим процесом, напруженою боротьбою та зосередженістю на рефлексах і зорово-моторній координації людини. Гравців часто занурюють у захоплюючі адреналінові сценарії, де вони повинні долати складні рівні та перемагати ворогів. Прикладами популярних екшенів є «Devil May Cry», «God of War» та «Uncharted». До переваг жанру екшену відносяться захопливий ігровий процес, захоплююче навколишнє середовище та великий акцент на завданнях, що базуються на навичках. Однак у цьому жанрі може бракувати глибокого сюжету та елементів дослідження.

Рисунок 1.1. Скріншот з гри «Uncharted»

1.1.2 Ігровий жанр Adventure

Пригодницькі ігри ставлять на перше місце дослідження, розв'язання головоломок та розповідь історій. Гравці здійснюють подорожі, розгадують містичні таємниці, взаємодіють з різними персонажами та оточенням. Яскравими прикладами є такі ігри, як «Legend of Zelda», «Tomb Raider» та «Witcher».

Пригодницький жанр пропонує багатий сюжет, захоплюючі світи та інтелектуальні виклики. З іншого боку, йому може бракувати орієнтованого на дію геймплею та швидкого розвитку подій.

Рисунок 1.2. Скріншот з гри «Witcher»

1.1.3 Ігровий жанр RPG

Рольові ігри дозволяють гравцям відчути себе ролі персонажів у детально розробленому світі, часто заснованому на фентезі. Зазвичай вони передбачають розвиток персонажа, підвищення рівня та прийняття рішень, які впливають на сюжет. Серед відомих рольових ігор – «The Elder Scrolls: Skyrim», «Dark Souls» та «Mass Effect». RPG пропонують глибоку кастомізацію, розгорнуту історію та стратегічний геймплей. Однак вони можуть потребувати багато часу і бути надто складними для гравців, які прагнуть негайного задоволення.

Рисунок 1.3. Скріншот з гри «Dark Souls»

1.1.4 Ігровий жанр Strategy

Стратегічні ігри вимагають від гравців тактичного мислення, планування та управління ресурсами для досягнення конкретних цілей. Вони часто передбачають контроль юнітів, будівництво споруд та прийняття стратегічних рішень. Прикладами популярних стратегічних ігор є «Civilization», «StarCraft» та «XCOM». Жанр стратегії пропонує інтелектуальні виклики, довгострокове планування та задоволення від перемог. Проте він може здатися дуже складним для гравців-початківців і не передбачати швидкого розвитку подій.

Рисунок 1.4. Скріншот з гри «StarCraft»

1.1.5 Ігровий жанр Puzzle

Ігри жанру головоломки зосереджені на вирішенні логічних, просторових або математичних завдань. Гравці отримують набір перешкод або викликів, які вони повинні подолати, використовуючи свою кмітливість, логіку та аналітичні навички. Цей жанр часто вимагає творчого мислення та вміння підходити до проблем з різних боків. Головоломки можуть варіюватися від простих і зрозумілих до неймовірно складних, розрахованих на гравців з різним рівнем навичок та інтелектуальних здібностей. Такі ігри, як «Tetris», «Portal» та «The Witness» є відомими прикладами. Головоломки забезпечують цікаві розумові вправи, сприяють розвитку критичного мислення та дарують відчуття досягнення після вирішення складних головоломок. Однак цьому жанру може бракувати розгорнутих сюжетів та захоплюючих світів.

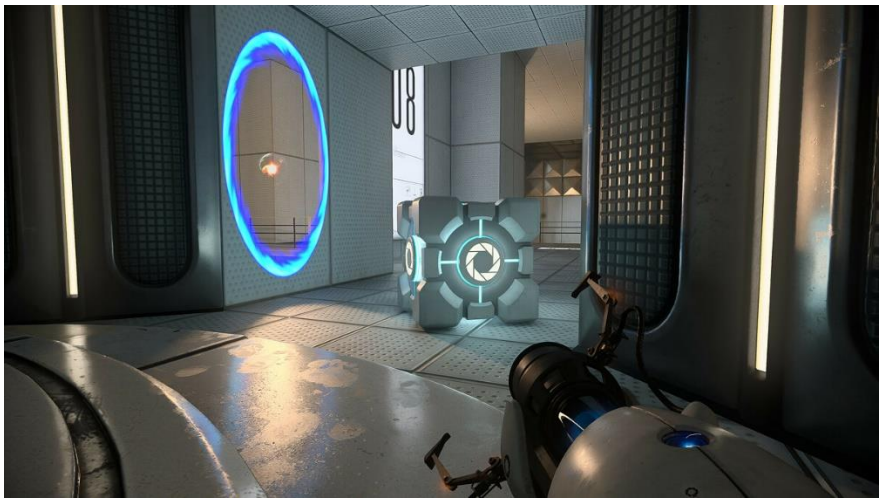


Рисунок 1.5. Скріншот з гри «Portal»

1.1.6 Ігровий жанр Simulation

Ігри жанру симулятори мають на меті відтворити реальний досвід або діяльність. Вони можуть імітувати все - від будівництва міст і фермерства до симуляторів польотів або водіння. До популярних симуляторів належать «The Sims», «Farming Simulator» та «Euro Truck Simulator». Жанр симулятора дозволяє гравцям зануритися в реалістичні сценарії, пропонуючи відчуття свободи та можливість опанувати складні системи. Тим не менш, цьому жанру може не вистачати екшену та динамічного ігрового процесу.

Рисунок 1.6. Скріншот з гри «Euro Truck Simulator»

1.1.7 Ігровий жанр Quest

Жанр квесту фокусується на залученні гравців у керовану розповіддю подорож, яка часто включає розв'язання головоломок, дослідження та розвиток.

Ігри-квести заохочують гравців до виконання завдань, досягнення цілей і розкриття історії. Прикладами таких ігор є «Syberia» та "Machinarium". Ігри-квести - це захопливий сюжет, різноманітні механіки геймплею та відчуття завершеності після виконання поставлених завдань. Вони пропонують баланс між екшеном, пригодами та елементами розв'язання головоломок. Однак цьому жанру може бракувати екшену та ігрового процесу [1].

Рисунок 1.7. Скріншот з гри «Machinarium»

Вибір жанру квесту для ігрового проекту виправданий його захопливою ігровою механікою, наративною структурою, світобудівним потенціалом, гнучкістю в дизайні квесту, активністю гравця та балансом виклику і винагороди. Орієнтуючись на жанр квесту, ігровий проект має на меті створити захоплюючий досвід, який спонукає до дослідження, вирішення проблем та розвитку сюжету.

1.2 Аналіз ігрових двигунів

У середовищі розробки ігор з'явилися численні ігрові двигуни, які пропонують широкий спектр функцій та можливостей для полегшення створення інтерактивного цифрового досвіду. Кожен двигун має свої унікальні переваги, особливості та недоліки, тому потрібно проаналізувати існуючі платформи для розробки ігрового проекту.

1.2.1 Ігровий двигун Unreal Engine

Unreal Engine, розроблений Epic Games, є потужним і широко використовуваним ігровим двигуном, відомим своїми передовими можливостями рендерингу графіки. Він пропонує візуально приголомшливе та реалістичне середовище, що робить його особливо придатним для створення ігор рівня AAA. Unreal Engine надає цілий ряд можливостей, включаючи надійну фізичну систему, вдосконалене освітлення та ефекти частинок, а також гнучку систему візуальних сценаріїв blueprint. Він також підтримує розробку віртуальної реальності (VR) та доповненої реальності (AR). Сильною стороною двигуна є його здатність створювати високоякісні візуальні ефекти та реалістичний ігровий процес. Однак Unreal Engine вимагає більш глибоких знань програмування і може бути більш ресурсоємним порівняно з іншими двигунами [2].

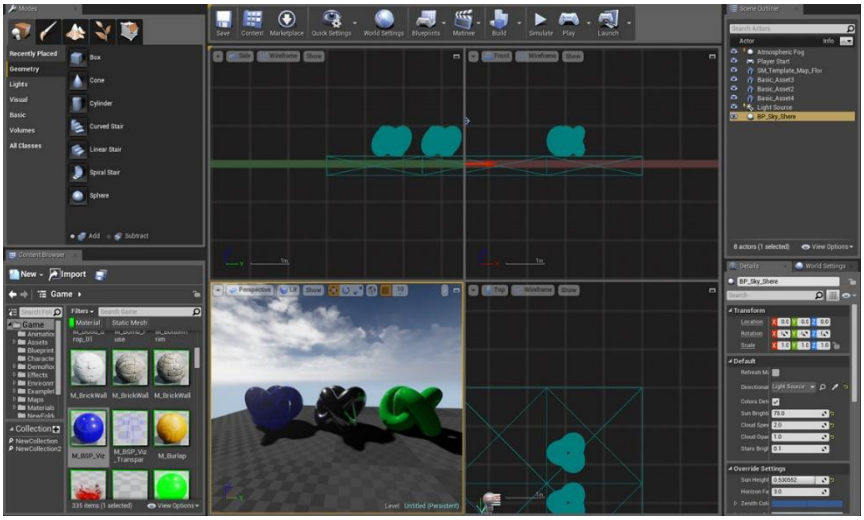


Рисунок 1.8. Інтерфейс ігрового двигуна «Unreal Engine»

1.2.2 Ігровий двигун Unity

Unity – це широко відомий та універсальний ігровий двигун, який підходить як для початківців, так і для досвідчених розробників. Він пропонує крос-платформну підтримку, що дозволяє розробляти ігри для різних платформ, таких як комп’ютерні, мобільні та консольні. Unity має зручний інтерфейс та інтуїтивно зрозумілу візуальну систему написання сценаріїв, відому як

Unity Editor. Ця візуальна система сценаріїв, разом з великою документацією та великою спільнотою, сприяє швидкому створенню прототипів та ітераціям.

Unity підтримує розробку як 2D, так і 3D ігор і пропонує широкий спектр можливостей, включаючи потужний фізичний двигун, інструменти анімації, аудіосистеми та магазин ресурсів з величезним вибором готових ресурсів і плагінів. Доступність, універсальність, кросплатформеність та широка підтримка спільноти роблять Unity популярним вибором як для інді-розробників, так і для великих ігрових студій.

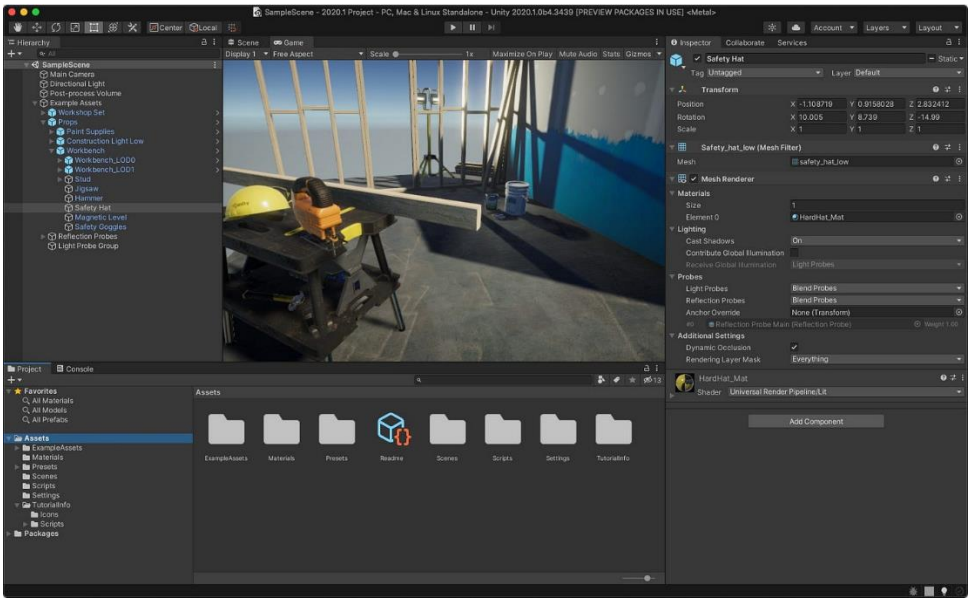


Рисунок 1.9. Інтерфейс ігрового двигуна «Unity»

1.2.3 Ігровий двигун CryEngine

CryEngine, розроблений компанією Crytek, відомий своєю винятковою графічною точністю та можливостями рендерингу. Він використовувався для створення візуально приголомшливих ігор, таких як серія Crysis. CryEngine пропонує передові ефекти освітлення та частинок, динамічні погодні системи та реалістичні фізичні симуляції. Він також надає візуальну систему написання сценаріїв під назвою Flowgraph, яка дозволяє швидко створювати прототипи. Сила CryEngine полягає в його здатності створювати приголомшливі візуальні ефекти та середовища з високим рівнем занурення. Однак, він має складніше навчання і вимагає більш глибоких знань з програмування та технічної експертизи [3].

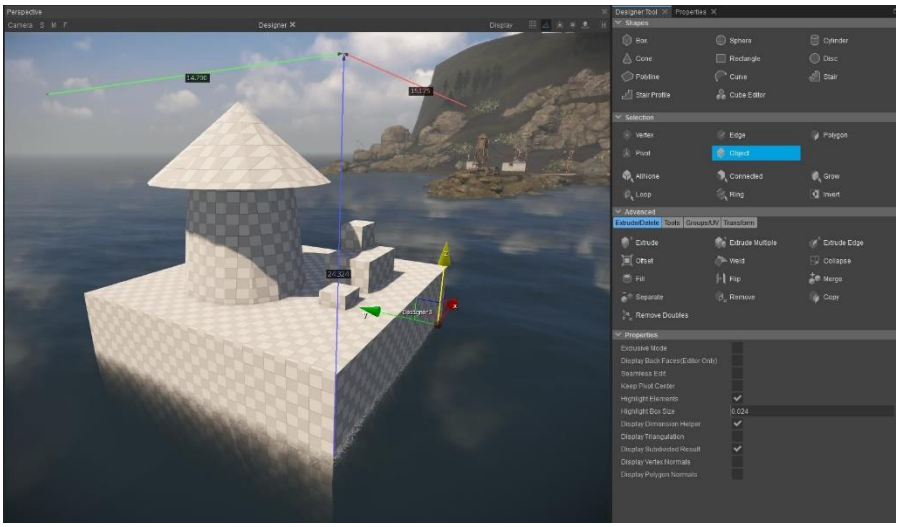


Рисунок 1.10. Інтерфейс ігрового двигуна «CryEngine»

1.2.4 Ігровий двигун Godot

Godot - це ігровий двигун з відкритим вихідним кодом, відомий своєю легкістю, гнучкістю та простотою використання. Він підтримує розробку як 2D, так і 3D ігор і пропонує цілий ряд функцій, включаючи візуальну систему сценаріїв, редактор анімації, фізичний двигун і спеціальний 2D-двигун. Розробкою Godot керує спільнота, що призвело до постійних вдосконалень та великої кількості доступних ресурсів. Він також має спеціальну бібліотеку ресурсів для обміну та завантаження ігрових ресурсів. Легка природа Godot робить його придатним для мобільних і невибагливих платформ. Однак, у порівнянні з більш відомими двигунами, такими як Unity та Unreal Engine, Godot може мати меншу спільноту та більш обмежений вибір готових ресурсів і плагінів [4].

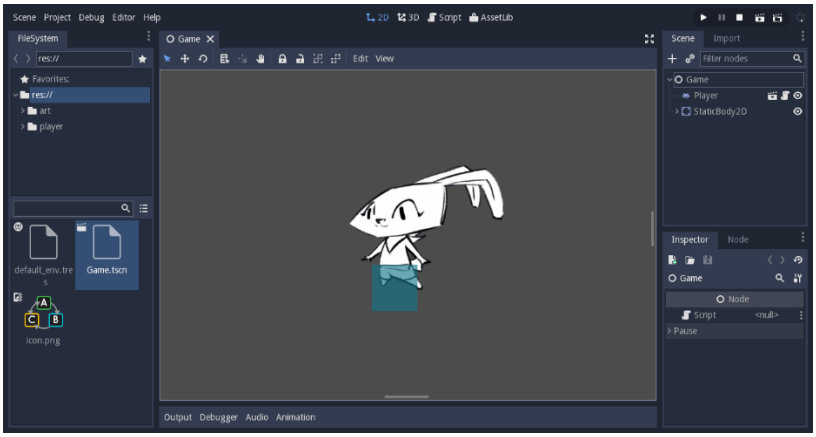


Рисунок 1.11. Інтерфейс ігрового двигуна «Godot»

Вибір Unity як ігрового двигуна для цього проекту виправданий його доступністю, зручним інтерфейсом, крос-платформенною підтримкою, великою спільнотою та сховищем ресурсів, універсальністю та індустріальною адаптацією. Можливості Unity відповідають практичним цілям проекту, надаючи необхідні інструменти та ресурси для створення захоплюючого ігрового досвіду.

1.3 Розробка технічного завдання для ігрового проекту

Переді мною постало завдання створити ігровий проект, розроблений на платформі Unity, використовуючи мову програмування C#, для операційної системи Windows, в жанрі квест. Гравець, граючи за космонавта, повинен будувати та апгрейдити заводи, збирати та продавати ресурси і, зрештою, відремонтувати свою ракету, щоб повернутися на Землю. Гра буде зосереджена на сюжеті, ігрових механіках, ігровому дизайні, системному дизайні та дизайні інтерфейсу для створення цікавого та захоплюючого ігрового процесу.

- Ігрова механіка:

1. Збір ресурсів - гравці повинні збирати матеріали, отриманих від заводів, такі як мікрочіпи, мікросхеми, балони з паливом, пласкогубці та ізоляційну стрічку, щоб виготовляти інші матеріали.

2. Продаж ресурсів – гравці можуть продавати зайві ресурси, щоб отримувати ігрову валюту.

3. Купівля та модернізація заводів - гравці можуть купувати різні типи заводів для отримання необхідних матеріалів. Також гравці можуть апгрейдити заводи за ігрову валюту.

4. Виконання завдань – гравці можуть виконувати завдання, здаючи необхідні матеріали до заводу, щоб отримувати інші ресурси.

5. Ремонт ракети - гравці повинні відремонтувати ракету, збираючи необхідні матеріали та виконуючи завдання, щоб пройти гру.

- Ігровий дизайн:

1. Поверхня Місяця - гра матиме поверхню Місяця, щоб забезпечити унікальний досвід для кожного гравця.

2. Космічний дизайн рівня – гра матиме відтворений космос у якості навколишнього середовища рівня.

3. Космічні об’єкти – гра матиме космічні бази та заводи.

4. Ефекти – гра матиме різноманітні візуальні та звукові ефекти для отримання захоплюючого ігрового досвіду.

5. Сюжет – гра матиме візуальне представлення сюжету у вигляді перемикання слайдів.

6. Головне меню

- Системний дизайн:

1. Налаштування гри - гравці можуть налаштовувати налаштування гри, такі як аудіо та графічні опції.

2. Локалізація - гра буде підтримувати російську мову у якості озвучення сюжету гри, та англійську мову у якості стандартної мови.

- Дизайн інтерфейсу:

1. Інтуїтивно зрозумілий інтерфейс - користувальницький інтерфейс буде розроблений таким чином, щоб його було легко зрозуміти та орієнтуватися в ньому.

2. Кольорова палітра - гра буде використовувати кольорову палітру, натхненну космічними дослідженнями, з відтінками сірого, синього та чорного кольорів.

3. Сумісність з Windows - гра буде оптимізована для операційних систем Windows, що забезпечить плавний і приємний досвід для гравців.

- Мінімальні системні вимоги:

Процесор: Intel Core i3 чи його аналоги

Графічний процесор: інтегрована графіка (наприклад, Intel Graphics 4000) або недорогий спеціалізований графічний процесор

1.4 Створення ігрового проекту

Для більшої атмосферності та реалізму було прийняте рішення використати онлайн-ринок ресурсів Unity Asset Store. Це офіційний магазин Unity, що пропонує широкий вибір готових рішень, які допомагають прискорити процес розроблення, додати функціональність і поліпшити візуальний стиль гри. Було завантажено моделі планети Земля, зірки Сонце, космічних заводів, ракет, а також візуальні та графічні елементи.

1.4.1 Реалізація головного меню

Щоб зробити інтерфейс необхідно застосовувати компонент Canvas. Це компонент, який представляє 2D простір, на якому можна відображати користувацький інтерфейс (UI) в ігровій сцені. Він слугує контейнером для різних елементів інтерфейсу, таких як кнопки, текстові поля, зображення, панелі та інші UI-елементи.

Перше що необхідно зробити – додати фон головного меню. Для цього потрібно клацнути правою кнопкою миші на Canvas, далі UI – Raw Image. У компоненті Raw Image у поле Texture потрібно обрати необхідну текстуру для бекграунду.

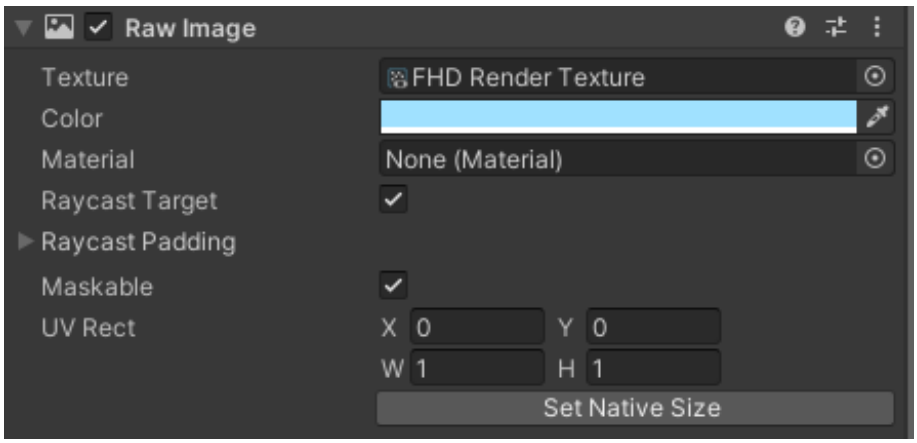


Рисунок 1.12. Компонент Raw Image

Рисунок 1.13. Створений фон для головного меню

Particle System в Unity дає змогу створювати реалістичні та динамічні ефекти, які пожвавлюють ігрову сцену та надають їй атмосфери. Він може бути використаний для створення атмосферних ефектів, додавання деталей і руху в оточення, створення вибухів і спеціальних ефектів, а також для створення інтерактивних елементів, таких як сліди від руху персонажа або сліди кулі від використання зброї. Додаємо на задній фон візуальні ефекти, за допомогою Particle System, для поглиблення у атмосферу головного меню.

Для цього клацаємо правою кнопкою миші по ієрархії сцени та обираємо Effects – Particle System. У розділі Shape вибираємо параметр Box, щоб ефекти створювалися усередині прямокутної області.

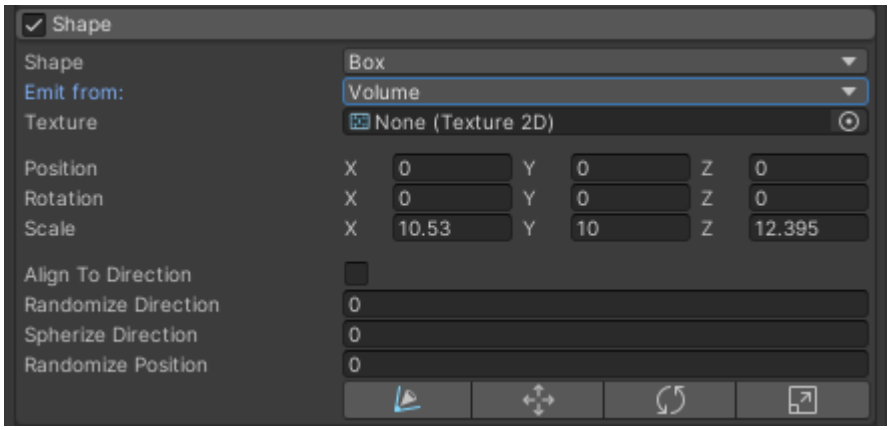


Рисунок 1.14. Налаштування області створення
візуальних ефектів

У розділі Emission – Rate over Time обираємо значення 100. Це буде кількість частинок випромінених за одиницю часу.

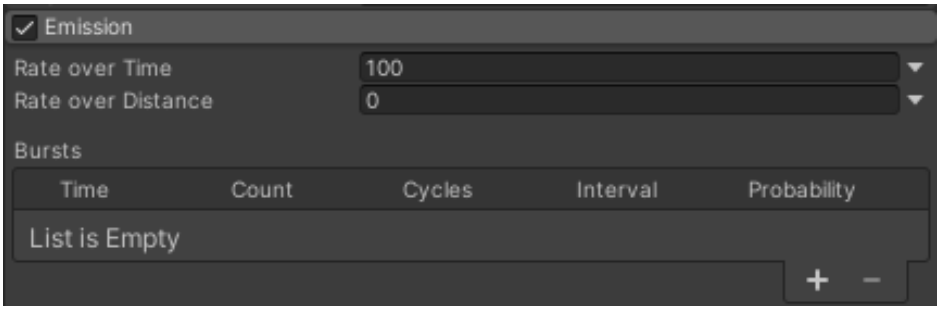


Рисунок 1.15. Налаштування частоти випромінювання частинок

Налаштовуємо параметри тривалості ефекту, обираємо повторення циклу створення частинок (Looping), тривалість життя кожної частинки, розмір, колір та швидкість.

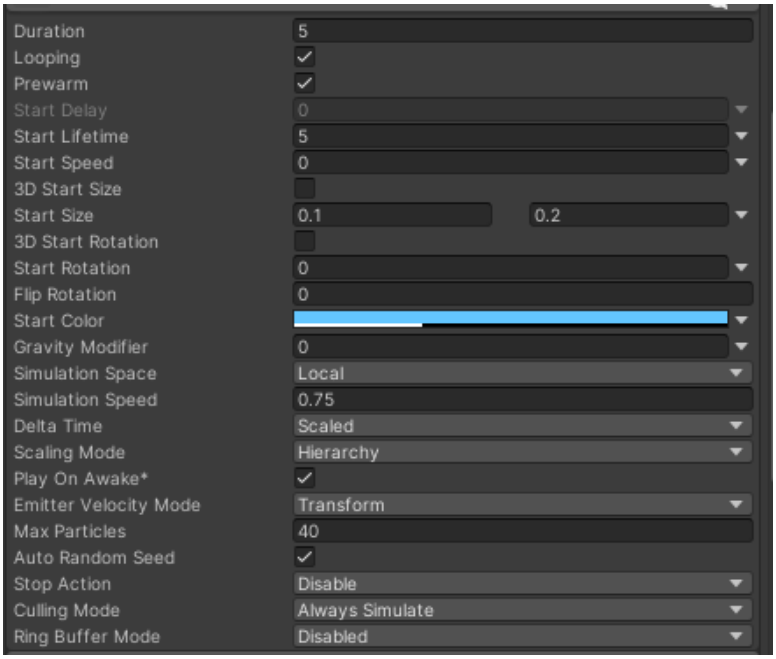


Рисунок 1.16. Налаштування параметрів частинок

Налаштовуємо модуль Шум (Noise). Він додає турбулентності до руху частинок. Крива, яка визначає, наскільки сильним є вплив шуму на частинку (Strength) має значення 0.1. Значення частоти (Frequency) обираємо 0.5, низькі значення створюють м’який, плавний шум. Цей параметр визначає, як часто частинки змінюють напрямок руху. Вмикаємо параметр Damping, сила шуму буде пропорційна частоті.

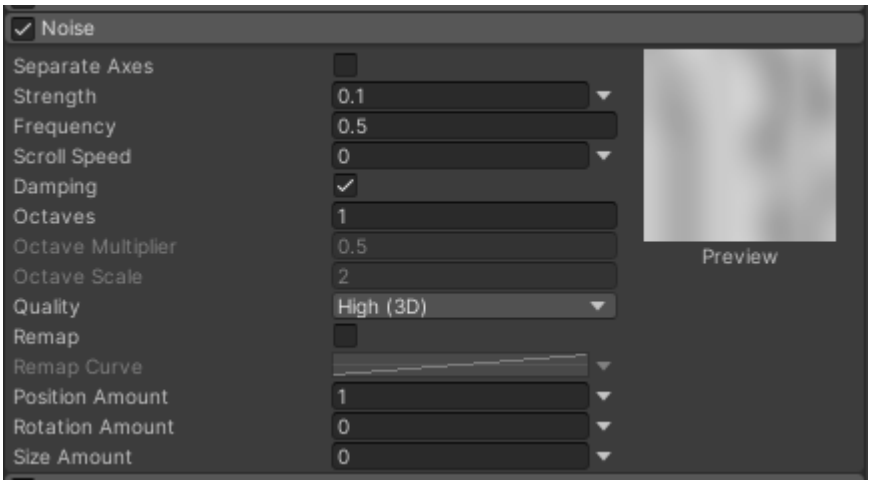


Рисунок 1.17. Налаштування параметру шуму

Рисунок 1.18. Створений візуальний ефект для фону головного меню

Для отримання більш красивого ефекту, додаємо тінь у середину екрану. Створюємо компонент Image (Canvas – UI – Image). Обираємо необхідне зображання (розпливчасте коло по краях), чорного кольору, з прозорістю кольору 125.

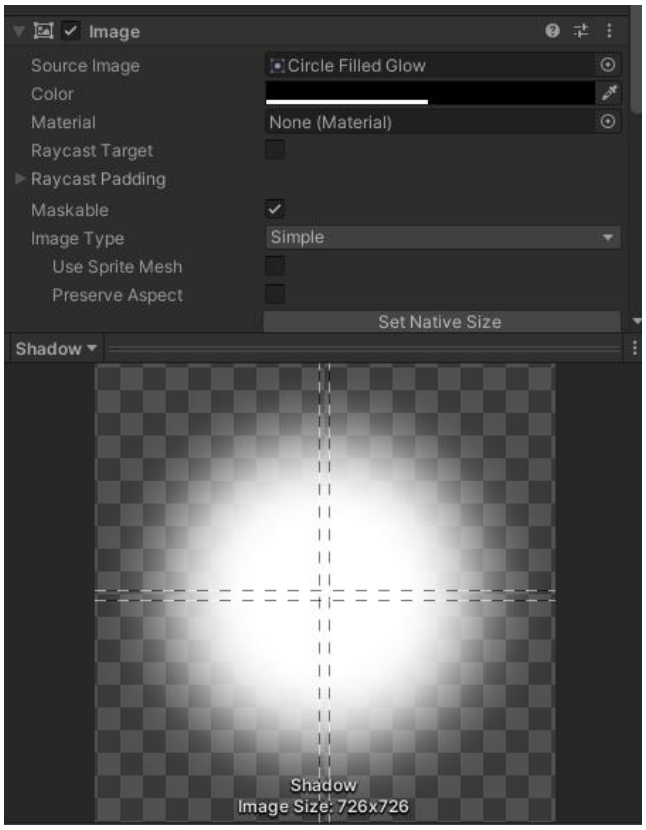


Рисунок 1.19. Створення тіні у середині заднього фону

Рисунок 1.20. Створена тінь у середині заднього фону

Додаємо назву гри та текст, який буде підказкою для гравця, що потрібно натиснути Пробіл для продовження.

Щоб додати текст клацаємо правою кнопкою миші по компоненту Canvas, далі UI – Text TextMeshPro. Обираємо необхідний шрифт, вирівнювання та в полі Text Input вводимо назву Moon Escape.

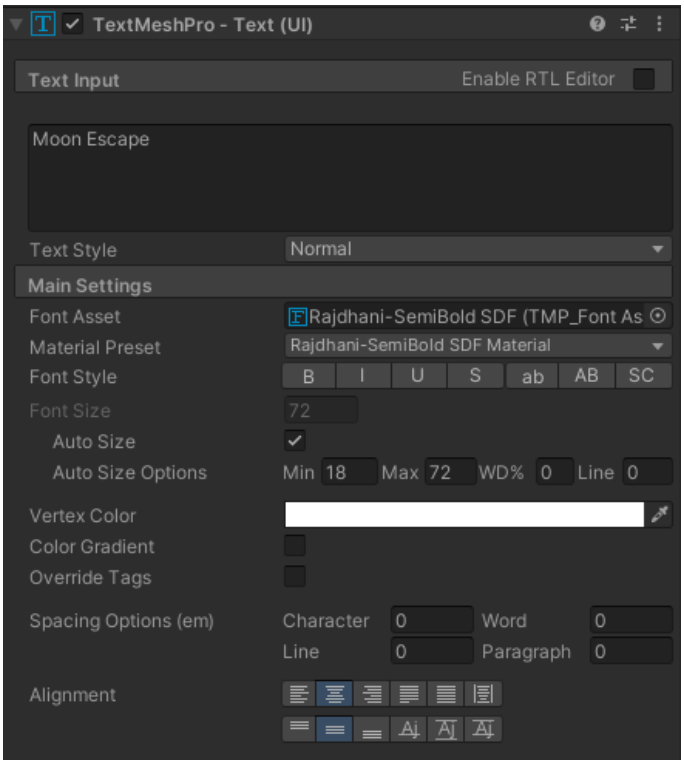


Рисунок 1.21. Налаштування тексту з назвою гри

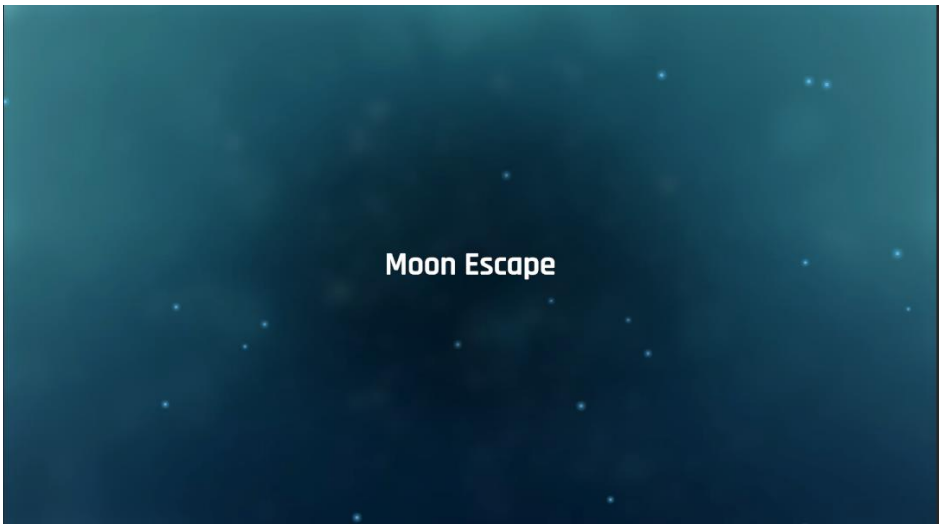


Рисунок 1.22. Створений текст з назвою гри

Таким самим чином додаємо текст-підказку «PRESS SPACE TO CONTINUE». Створюємо рамки навколо слова SPACE, за допомогою компонентів Image.

Рисунок 1.23. Створена рамка для тексту

Додаємо ще один компонент Image, для якого буде зроблена зациклена анімація пересування з лівого краю до правого. Щоб зробити анімацію, виділяємо необхідний об’єкт, на вкладці Animation натискаємо кнопку Create New Clip, створюємо анімацію з назвою «Highlight Anim Loop». Натискаємо на червоне коло, яке почне записувати анімацію. Ставимо повзунок на 0 секунд, обираємо потрібну стартову позицію. Далі ставимо повзунок на 1:30 секунд та обираємо кінцеву позицію.

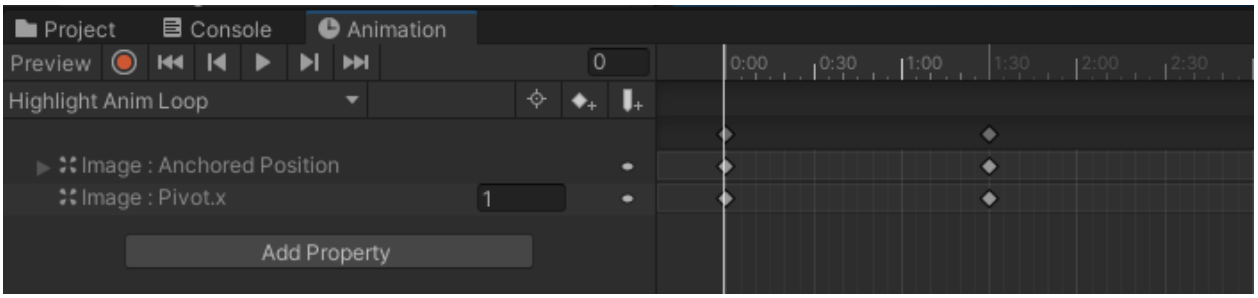


Рисунок 1.24. Створення анімації пересування зображення

Створюємо у папці Animator Controller з назвою «Highlight Anim». Для цього натискаємо правої кнопкою миші на пусте місце, далі Create – Animation Controller, вписуємо необхідну назву. Додаємо до об’єкту компонент Animator, та перетягуємо створений контролер до компоненту Animator.

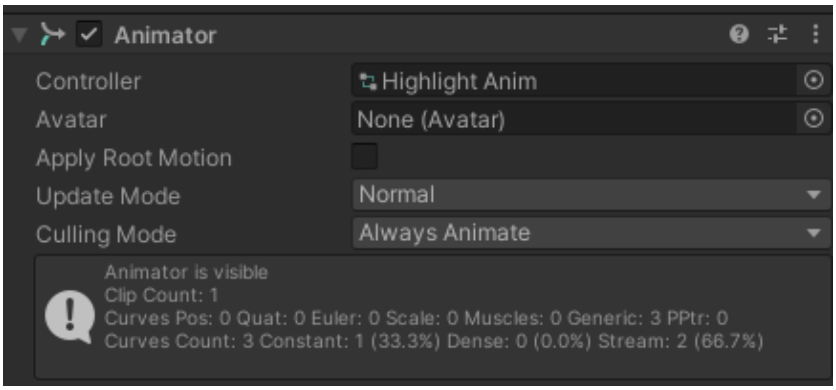


Рисунок 1.25. Компонент Animator з Animator Controller

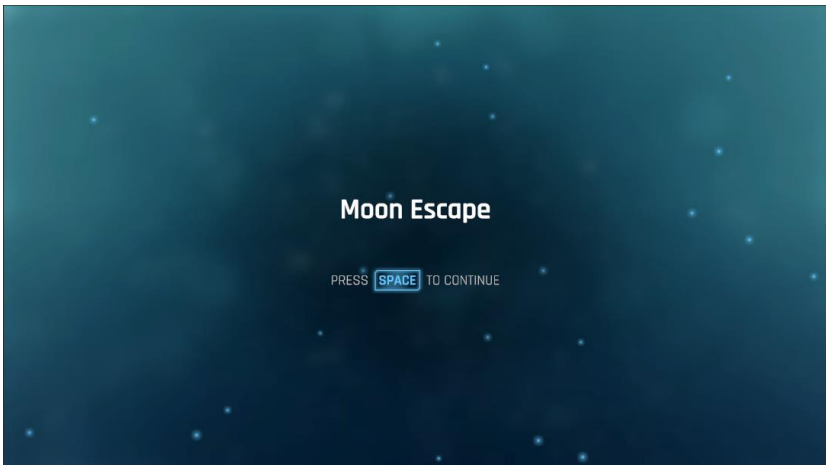


Рисунок 1.26. Створена анімація для тексту

Далі потрібно створити скрипт, який перенесе гравця на наступний етап головного меню при натисканні на Пробіл. Створюємо пустий об’єкт з назвою «Continue». Створюємо C# скрипт з назвою «Press Key Event». Відкриваємо скрипт за допомогою IDE Visual Studio 2019.

```
public class PressKeyEvent : MonoBehaviour
{
    [Header("Key")]
    [SerializeField]
    private KeyCode hotkey;

    [Header("Action")]
    [SerializeField]
    private UnityEvent pressAction;

    Сообщение Unity | Ссылок: 0
    void Update()
    {
        if (Input.GetKeyDown(hotkey))
            pressAction.Invoke();
    }
}
```

Рисунок 1.27. Скрипт відстеження натискання кнопки

public class PressKeyEvent : MonoBehaviour: цей рядок описує визначення класу PressKeyEvent, який є похідним від класу MonoBehaviour. MonoBehaviour - це базовий клас, який використовується в Unity для створення компонентів, які можуть взаємодіяти зі сценою та ігровим об'єктом.

Атрибут [Header("Key")] використовується для відображення заголовка в інспекторі Unity для секції властивостей "Key" (Клавіша).

[SerializeField] private KeyCode hotKey; Цей рядок оголошує приватну змінну hotKey типу KeyCode. KeyCode - це перерахування, що представляє клавіші на клавіатурі. Атрибут [SerializeField] дає змогу серіалізувати це поле, щоб воно відображалося в інспекторі Unity і було доступне для налаштування.

[SerializeField] private UnityEvent pressAction; Цей рядок оголошує приватну змінну pressAction типу UnityEvent. UnityEvent - це тип події Unity, яка може містити набір методів-обробників, що викликатимуться під час активації події.

Void Update() - цей метод викликається кожен кадр Unity і використовується для оновлення стану компонента.

if (Input.GetKeyDown(hotkey)) pressAction.Invoke(); У цьому умовному виразі перевіряється, чи була натиснута клавіша, визначена у змінній hotkey. Input.GetKeyDown(hotkey) повертає true, якщо клавіша була натиснута в цьому кадрі. Якщо ця умова виконується, то викликається метод Invoke() для події pressAction. Це призведе до виконання всіх методів-обробників, які були додані до події pressAction.

Додаємо новий текст «SYNCING DATA», створюємо анімацію для мерехтіння фону тексту, створюємо анімацію для компоненту Image, яка буде відображати процес загрузки. Анімація процесу загрузки створена за допомогою швидкого перемикання зображень.

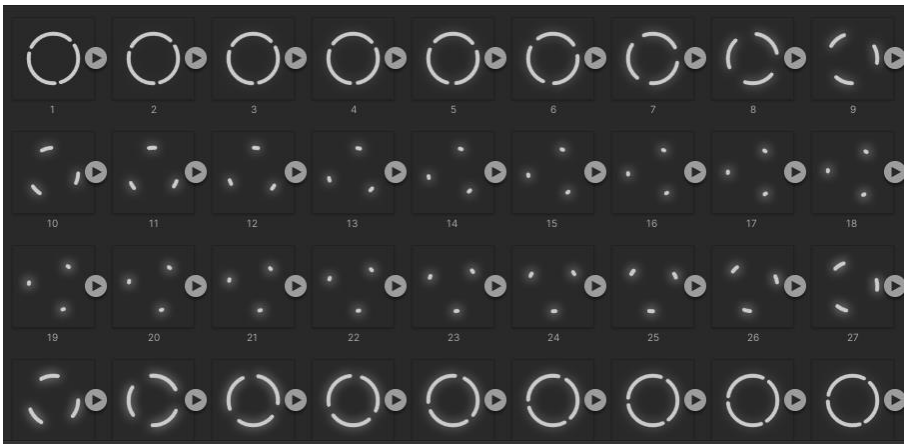


Рисунок 1.28. Зображення загрузки

Додаємо необхідні події, які будуть викликатися при натисканні на Пробіл у скрипті «Press Key Event». Це загрузка наступного зображення та програвання звуку.

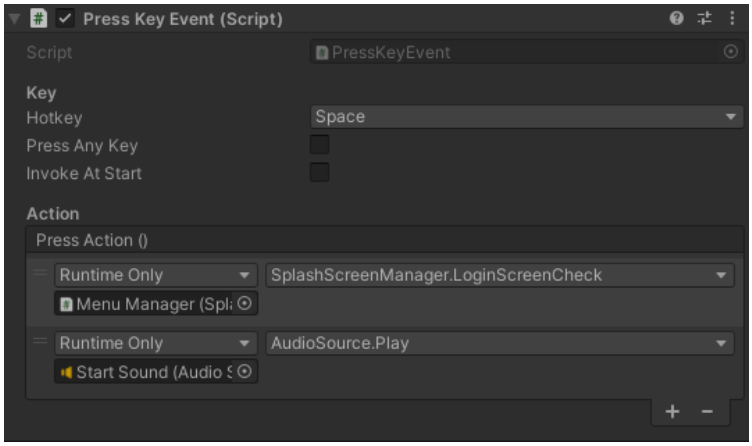


Рисунок 1.29. Події при натисканні на Пробіл

Щоб загрузка не була моментальною, було створено скрипт з назвою TimedEvent, який запускає необхідні події через 4 секунди.

Корутини в Unity - простий і зручний спосіб запускати функції, які повинні працювати паралельно протягом деякого часу. Корутини являють собою прості C# ітератори, що повертають IEnumerator і використовують ключове слово yield. У Unity корутини реєструються і виконуються до першого yield за допомогою методу StartCoroutine. Далі Unity опитує зареєстровані корутини після кожного виклику Update і перед викликом LateUpdate, визначаючи за значенням, що повертається в yield, коли потрібно переходити до наступного блоку коду.

```
public class TimedEvent : MonoBehaviour
{
    [Header("Timing (seconds)")]
    public float timer = 4;
    public bool enableAtStart;

    [Header("Timer Event")]
    public UnityEvent timerAction;

    Сообщение Unity | Ссылок: 0
    void Start()
    {
        if(enableAtStart == true)
        {
            StartCoroutine("TimedEventStart");
        }
    }

    Ссылок: 0
    IEnumerator TimedEventStart()
    {
        yield return new WaitForSeconds(timer);
        timerAction.Invoke();
    }

    Ссылок: 4
    public void StartIEnumerator ()
    {
        StartCoroutine("TimedEventStart");
    }

    Ссылок: 0
    public void StopIEnumerator ()
    {
        StopCoroutine("TimedEventStart");
    }
}
```

Рисунок 1.30. Скрипт TimedEvent

public float timer = 4; це публічна змінна timer, яка визначає час у секундах, через який має відбутися подія. Значення за замовчуванням дорівнює 4, але його можна змінити у інспекторі.

public bool enableAtStart; це публічна змінна enableAtStart, яка визначає, чи повинна подія бути запущена під час старту компонента. Якщо значення дорівнює true, то буде викликано метод StartIEnumerator() для запуску події.

if(enableAtStart == true) { StartCoroutine("TimedEventStart"); } У цьому умовному виразі перевіряється значення змінної enableAtStart. Якщо воно дорівнює true, то викликається метод StartIEnumerator() для запуску події через зазначений час.

IEnumerator TimedEventStart() – метод TimedEventStart() є корутиною (IEnumerator), який затримує виконання наступного рядка коду на вказану кількість секунд, а потім викликає подію timerAction.Invoke().

yield return new WaitForSeconds(timer); цей рядок коду призупиняє виконання корутини на вказану кількість секунд (timer). Після цього виконання продовжується з наступного рядка timerAction.Invoke(), який викликає подію timerAction та призводить до виконання всіх методів-обробників, які були додані до події.

public void StartIEnumerator () та public void StopIEnumerator () – це публічні метои, які надають змогу вручну запустити чи зупинити подію через зазначений час. Він викликає метод StartCoroutine("TimedEventStart"), щоб почати виконання корутини або метод StopCoroutine("TimedEventStart"), щоб зупинити виконання корутини.

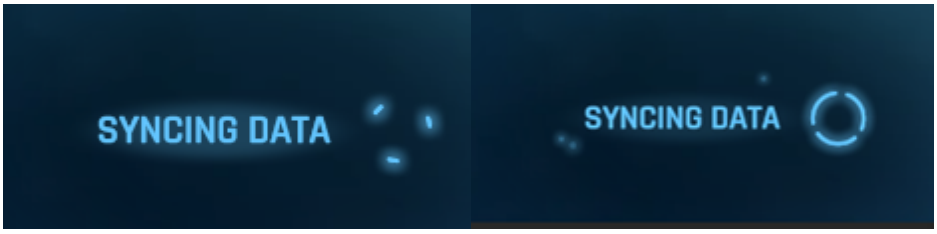


Рисунок 1.31. Анімація загрузки та мерехтіння фону тексту

Було створено наступний розділ головного меню, воно включає у себе різні анімації, текст, зображення, можливість налаштування звуку, фонова музика, звукові ефекти при наведенні на кнопки (SFX), можливість виходу з гри.

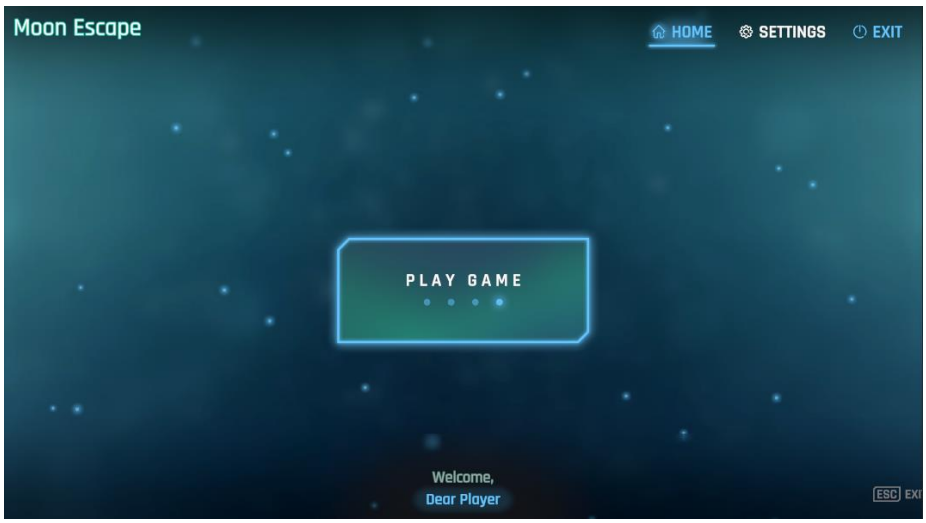


Рисунок 1.32. Другий розділ головного меню

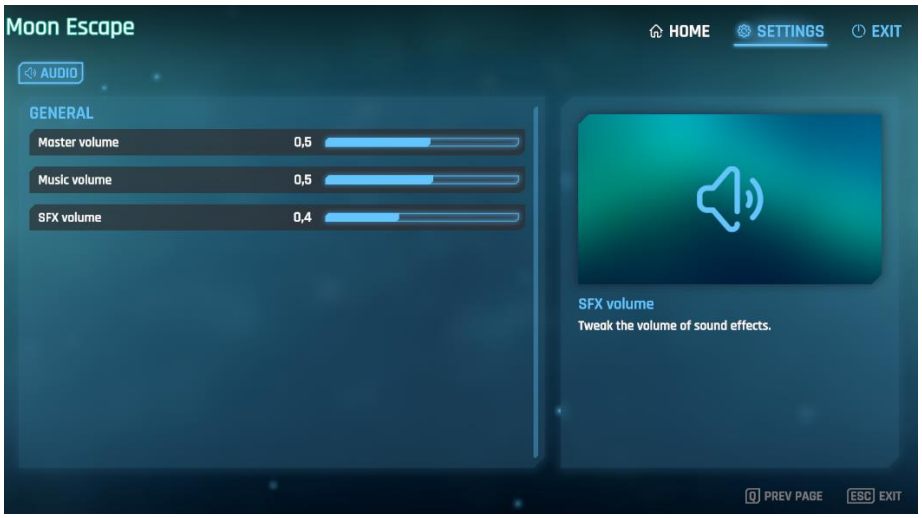


Рисунок 1.33. Меню налаштування гучності звуку та звукових ефектів

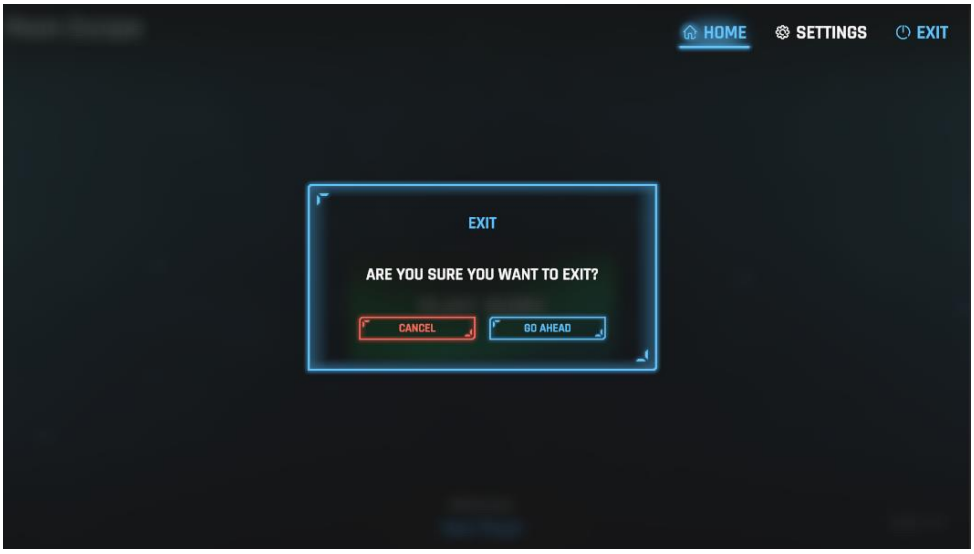


Рисунок 1.34. Спливаюче меню виходу з гри

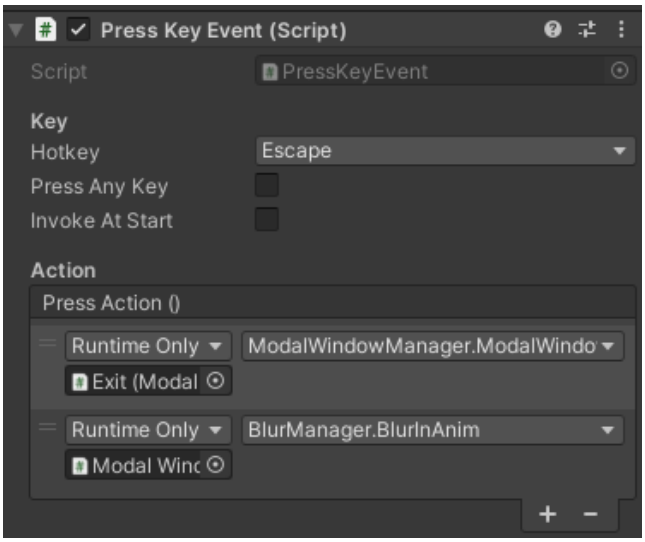


Рисунок 1.35. Події при натисканні на Escape

```
public class ExitToSystem : MonoBehaviour
{
    Ссылок: 0
    public void ExitGame()
    {
        Debug.Log("Exit function is working on build mode.");
        Application.Quit();
        UnityEditor.EditorApplication.isPlaying = false;
    }
}
```

Рисунок 1.36. Скрипт виходу з гри ExitToSystem

Метод ExitGame() призначений для завершення роботи гри або програми. Модифікатор доступу public дозволяє викликати метод ExitGame будь-де. Application.Quit() використовується для завершення роботи гри. Він виходить із гри та закриває її. UnityEditor.EditorApplication.isPlaying = false; цей рядок коду встановлює властивість isPlaying у редакторі Unity у значення false. Це використовується для зупинки програвання сцени в редакторі Unity. Додаємо подію OnClick() на кнопку «Go Ahead», щоб при натисканні кнопки ми виходили з гри.

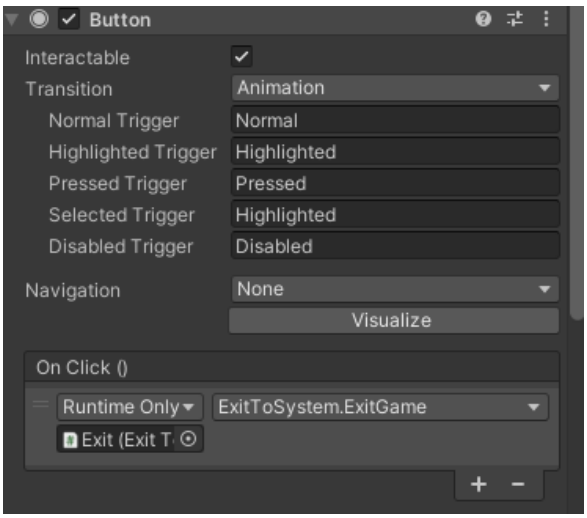


Рисунок 1.37. Події при натисканні на кнопку «GoAhead»

1.4.2 Реалізація сцени з розповіддю сюжету

Створюємо Canvas. У ієрархі в компоненті Canvas Scale обираємо мод Scale With Screen Size щоб забезпечити масштабованість інтерфейсу на різних пристроях і роздільній здатності екрана. У полі Reference Resolution обираємо роздільну здатність екрану 1080 по X та 1920 по Y. Значення Match ставимо 0.5.

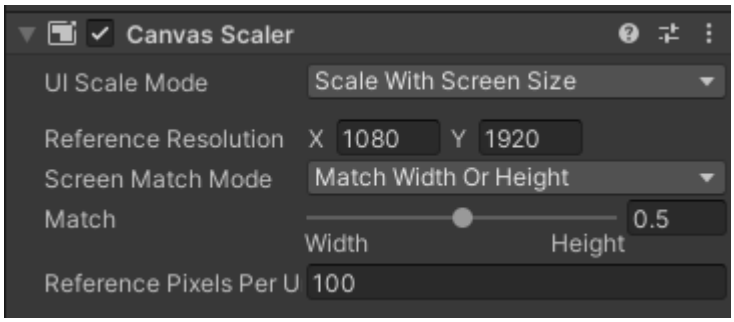


Рисунок 1.38. Налаштування компоненту Canvas Scaler

Створюємо фон сцени за допомогою компоненту Image. Обираємо необхідний 2D Sprite.

Рисунок 1.39. Фон сцени з розповіддю сюжету

Далі створюємо затінення фонового зображення за допомогою компоненту Panel. Для цього натискаємо правою кнопкою миші по компоненту Canvas, далі обираємо UI – Panel. Робимо прозорість кольору зі значенням 220.

Рисунок 1.40. Фон сцени з затіненням

Додаємо до компоненту Panel текстове поле. Встановлюємо потрібний шрифт, розмір, колір та вирівнювання тексту.

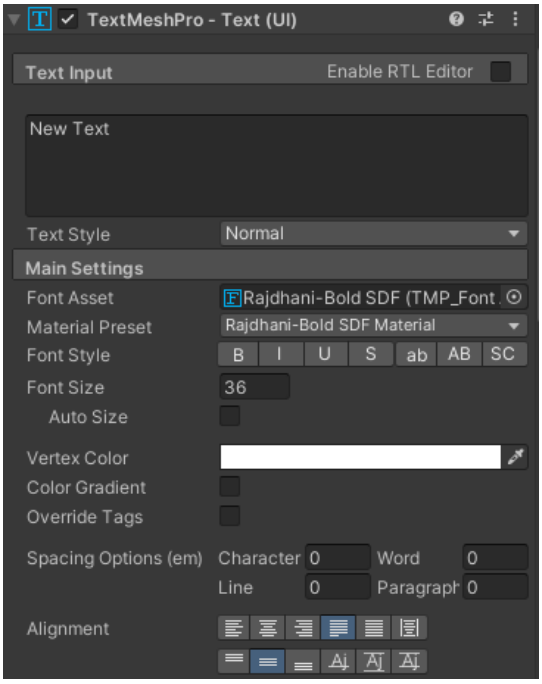


Рисунок 1.41. Налаштування текстового компоненту

Налаштовуємо границі тексту за допомогою інструменту Rect Tool.

Рисунок 1.42. Границі тексту

Далі потрібно створити сховище даних для тексту. Для цього скористаємося класом Scriptable Object. Scriptable Object в Unity - це клас, який

дає змогу створювати та зберігати дані, які можуть бути використані в різних частинах проєкту. Він являє собою контейнер для даних, який може бути збережений у вигляді активу (asset) у проєкті Unity.

Створюємо скрипт з назвою StoryText.

```
[CreateAssetMenu(fileName = "Story", menuName = "Data/Story")]
[System.Serializable]
Скрипт Unity | Ссылок 6
public class StoryText : ScriptableObject
{
    public List<Sentence> sentences;
    public StoryText nextText;

    [System.Serializable]
    Ссылок 1
    public struct Sentence
    {
        [TextArea(1, 10)]
        public string text;
    }
}
```

Рисунок 1.43. Скрипт StoryText

Цей код описує клас StoryText, який успадковується від класу ScriptableObject в Unity.

Атрибут [CreateAssetMenu] вказує Unity, що цей клас може бути створений як актив (asset) через контекстне меню проекту. У цьому рядку визначено ім'я файлу ("Story") і шлях до створення активу ("Data/Story").

Публічне поле sentences типу List<Sentence>, яке являє собою список речень (Sentence). Список sentences містить текстові речення, які складають сюжетний текст.

Публічне поле nextText типу StoryText, яке представляє наступний сюжетний текст після поточного. Воно використовується для встановлення зв'язку між різними сюжетними елементами та переходів у грі.

Атрибут [TextArea] вказує Unity, що поле text має бути відображено в інспекторі з можливістю багаторядкового текстового введення. Параметри (1, 10) визначають мінімальну та максимальну кількість рядків для текстового поля.

Публічне поле text типу string, яке містить текст речення в сюжетному тексті.

Створюємо ScriptableObject за допомогою натискання правої кнопки миші по порожньому місці у папці, далі Create – Data – Story. Вписуємо необхідний текст та обираємо наступний ScriptableObject.

Рисунок 1.44. Створений ScriptableObject

Створюємо скрипт з назвою «StoryController» (Додаток А), який буде виконувати роль контролера сюжету. Щоб створити анімацію друкованого тексту пишемо наступний код.

```
Ссылок: 2
public void PlayNextSentence()
{
    StartCoroutine(TypeText(currentText.sentences[++_sentenceIndex].text));
}

Ссылок: 1
private IEnumerator TypeText(string text)
{
    yield return new WaitForSeconds(0.1f);
    storyText.text = "";
    state = State.PLAYING;
    int wordIndex = 0;

    while (state != State.COMPLETED)
    {
        storyText.text += text[wordIndex];
        yield return new WaitForSeconds(textSpeed);
        if(++wordIndex == text.Length)
        {
            state = State.COMPLETED;
            audioSource.Stop();
            StartCoroutine(EnableSpaceBarCanvas());
            _textBlinkIndex = 1;
            break;
        }
    }
}
```

Рисунок 1.45. Код для створення анімації друкованого тексту

Метод PlayNextSentence() відтворює наступне речення в поточній сцені. Збільшує значення _sentenceIndex на 1 і викликає корутину TypeText() з текстом із масиву sentences об'єкта currentText за вказаним індексом _sentenceIndex.

Корутина для посимвольного відображення тексту TypeText(string text). Приймає як аргумент текст, який потрібно відобразити. Спочатку затримує виконання на 0.1 секунди. Потім встановлює storyText.text у порожній рядок і встановлює state у значення State.PLAYING. Створює змінну wordIndex і встановлює її значення в 0. Потім у циклі while, поки state не стане рівним State.COMPLETED, додає символи тексту в storyText.text, затримуючи виконання на інтервал textSpeed між символами. Якщо wordIndex досягає кінця тексту, встановлює state у значення State.COMPLETED, зупиняє відтворення звуку в audioSource, запускає корутину EnableSpaceBarCanvas() і встановлює _textBlinkIndex в 1.

Далі створюємо скрипт з назвою «ControllerStoryScene» (Додаток Б), за допомогою якого, при натисканні клавіши Пробіл, ми будемо перемикати текст та фонове зображення.

```
private enum State
{
    IDLE, ANIMATE
}

Сообщение Unity | Ссылок 0
void Start()
{
    currentTextIndex = -1;
    storyControllerScript.PlayScene(currentScene);
}

Сообщение Unity | Ссылок 0
void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        if (state == State.IDLE && storyControllerScript.IsCompleted())
        {
            if (storyControllerScript.IsLastSentence())
            {
                {
                    PlayScene(currentScene.nextText);
                }
            }
            else
            {
                storyControllerScript.PlayNextSentence();
            }
        }
    }
}
```

Рисунок 1.46. Перша частина коду скрипта ControllerStoryScene

Метод Start() виконується під час запуску сцени. Ініціалізує змінну currentTextIndex зі значенням -1 і викликає метод PlayScene() скрипта StoryController для запуску першої сцени.

Метод Update() виконується кожен кадр. Перевіряє, чи була натиснута клавіша пробілу. Якщо контролер перебуває у стані IDLE і сцену завершено, то відбувається перевірка, чи є поточне речення останнім у сцені. Якщо так, то викликається метод PlayScene() з посиланням на наступну сцену. В іншому разі викликається метод PlayNextSentence() скрипта storyControllerScript, щоб продовжити показувати наступне речення.

```
public void ChangeBackgroundImage()
{
    background.sprite = backgrounds[currentTextIndex];
}

Ссылок: 1
private void PlayScene(StoryText scene)
{
    StartCoroutine(SwitchScene(scene));
}
Ссылок: 1
private IEnumerator SwitchScene(StoryText scene)
{
    state = State.ANIMATE;
    currentScene = scene;
    storyControllerScript.Hide();
    yield return new WaitForSeconds(0.4f);
    ChangeBackgroundImage();
    yield return new WaitForSeconds(0.4f);
    storyControllerScript.ClearText();
    storyControllerScript.Show();
    yield return new WaitForSeconds(0.6f);
    if (currentTextIndex != 8)
    {
        storyControllerScript.PlayScene(scene);
    }
    state = State.IDLE;
}
```

Рисунок 1.47. Друга частина коду скрипта ControllerStoryScene

Метод ChangeBackgroundImage() змінює спрайт фонового зображення на основі значення currentTextIndex.

Метод PlayScene() запускає перемикання на нову сцену. Викликає корутину SwitchScene().

Метод SwitchScene() виконує перемикання між сценами. Встановлює стан контролера в ANIMATE і зберігає посилання на нову сцену. Приховує текст і запускає анімацію зміни фону. Після завершення анімації показує новий текст і відображає його. Якщо поточний currentTextIndex не дорівнює 8, то викликається

метод PlayScene() скрипта storyControllerScript, щоб запустити наступну сцену. Потім стан контролера встановлюється в IDLE. Значення 8 потрібно для того, щоб уникнути появи помилки «Вихід за границі масиву».

Були створені анімації Hide та Show, які змінюють прозорість компоненту Panel. Hide змінює прозорість с значення 1 до 0. Show змінює значення прозористі з 0 до 1.

Також був створений текст-підказка «PRESS SPACEBAR TO CONTINUE…», для того щоб гравець розумів, яким чином можна перемикнути на наступний текст.

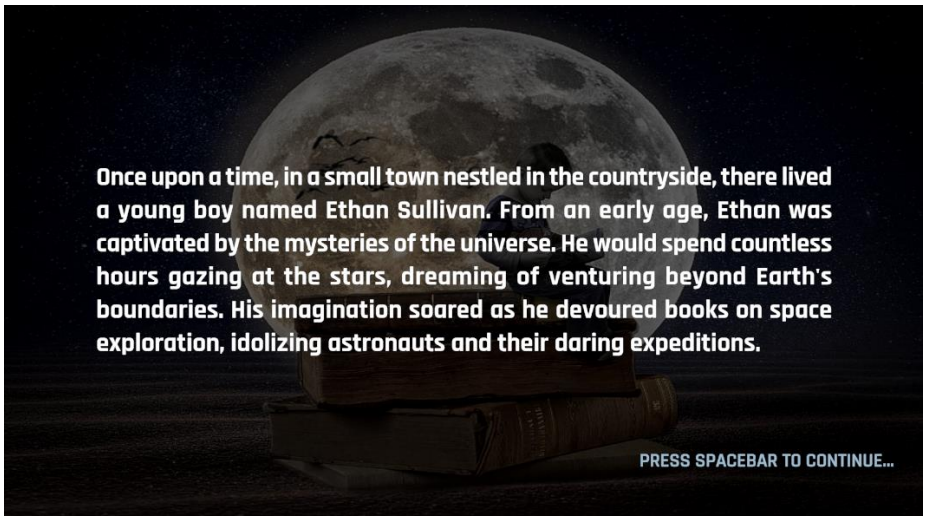


Рисунок 1.48. Сцена розповіді сюжету з текстом-підказкою

Для тексту-підказки було створено анімації блимання з назвою «Blinking Text» та «Stop Blinking», які змінюють прозорість тексту.

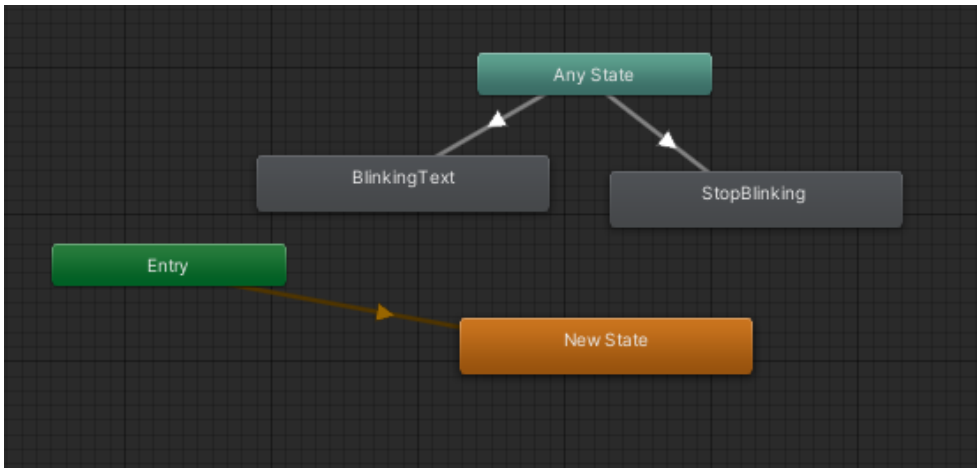


Рисунок 1.49. Вікно Animator для тексту-підказки

Було створено тригери, при включенні яких відбувався початок або зупинення анімації.

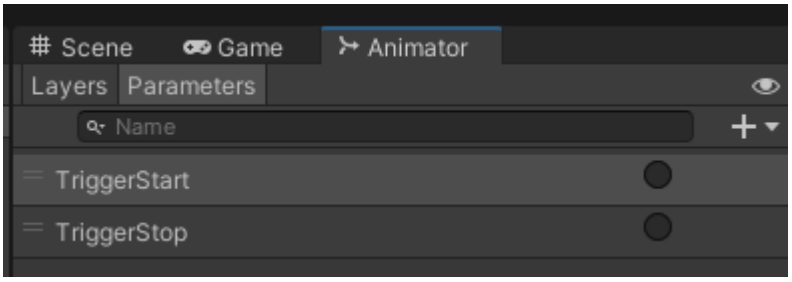


Рисунок 1.50. Створені тригери для анімації

Далі потрібно було зробити озвучення поточного тексту. Для цього було завантажено спеціальний набір даних RTVoice. Додаємо до сцени компонент SpeechText, який буде програвати озвучення тексту.

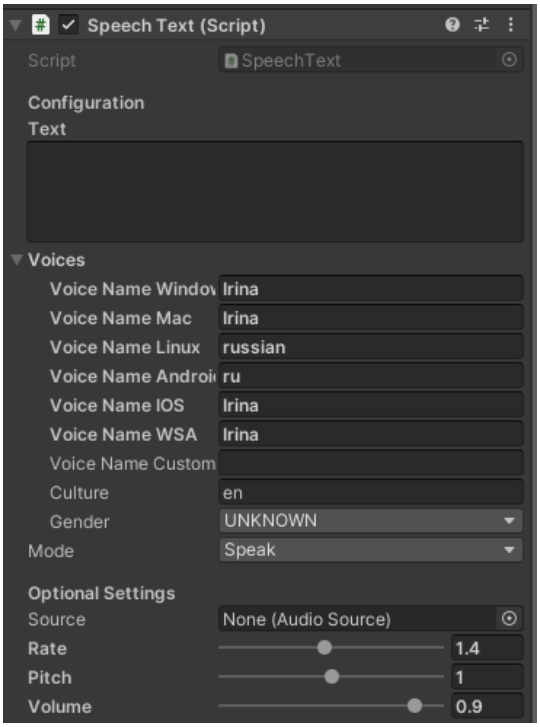


Рисунок 1.51. Компонент SpeechText

У скрипті StoryController створюємо метод Speak(), для відтворення озвучення за вказаним індексом index тексту.

```
Ссылок: 1
private void Speak(int index)
{
    speakerScript.text = text[index];
    speakerScript.Speak();
}
```

Рисунок 1.52. Метод Speak

Встановлюється текст у компоненті SpeechText таким, що дорівнює тексту, який міститься в масиві text за вказаним індексом, а потім викликає метод Speak() компонента SpeechText, щоб відтворити озвучення.

Також додаємо фоновий звук печатання за допомогою компоненту AudioSource.

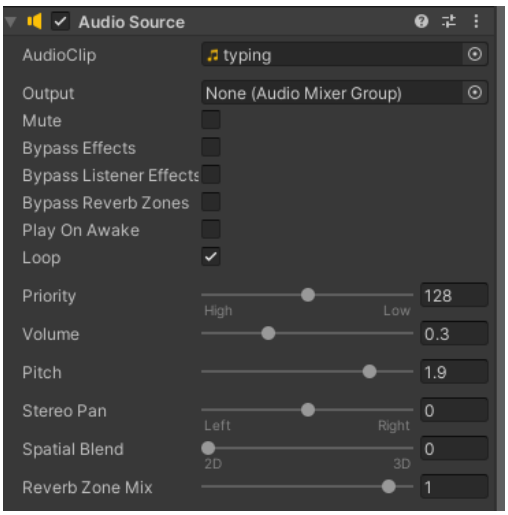


Рисунок 1.53. Компонент AudioSource

Створюємо скрипт з назвою «Start Game Scene», який буде запускати головну сцену.

```
public class StartGameScene : MonoBehaviour
{
    [SerializeField] private string sceneName;
    Ссылок: 1
    public void StartGame()
    {
        StartCoroutine(LoadSceneAsyncCoroutine(sceneName));
    }

    Ссылок: 1
    private IEnumerator LoadSceneAsyncCoroutine(string sceneName)
    {
        AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(sceneName, LoadSceneMode.Single);
        asyncOperation.allowSceneActivation = false;

        while (!asyncOperation.isDone)
        {
            // Check if the scene has finished loading.
            if (asyncOperation.progress >= 0.9f)
            {
                asyncOperation.allowSceneActivation = true;
            }

            yield return null;
        }
    }

}
```

Рисунок 1.54. Код скрипту StartGameScene

[SerializeField] private string sceneName це текстова змінна, яка буде містити назву сцени, яку необхідно завантажити.

Метод StartGame() запускає корутину LoadSceneAsyncCoroutine() із зазначеним ім'ям сцени.

Метод LoadSceneAsyncCoroutine() виконує асинхронне завантаження сцени. Створює екземпляр AsyncOperation і запускає завантаження зазначеної сцени в режимі LoadSceneMode.Single. Встановлює прапор allowSceneActivation у false, щоб запобігти автоматичній активації завантаженої сцени.

У циклі while перевіряється, чи завершено завантаження сцени. Якщо прогрес завантаження досягає 90% (asyncOperation.progress >= 0.9f), то встановлюється прапор allowSceneActivation в true, щоб дозволити активацію завантаженої сцени.

Цикл триває доти, доки завантаження сцени не буде завершено повністю (asyncOperation.isDone == true).

Корутина завершується, коли завантаження сцени закінчено.

Асинхронне завантаження використовується щоб не було ніяких затримок перед завантаженням головної сціни.

Створюємо текст завантаження сцени та створюємо анімацію для нього.

Рисунок 1.55. Створений текст завантаження сцени

У скрипті StoryController створюємо умову, доки індекс тексту не дорівнює значенню 8, буде програватися озвучення, звук печатання та буде програватися анімація печатання тексту. Якщо індекс тексту дорівнює іншому значенню, з’являється текст загрузки та викликається метод StartIEnumerator() з скрипту Timed Event, щоб завантаження головної сцени тривало 4 секунди.

```
if (_textIndex != 8)
{
    Speak(_textIndex);
    PlayNextSentence();
    StartCoroutine(TypeSound());
}
else
{
    loading.alpha = 1;
    timedEventScript.StartIEnumerator();
}
```

Рисунок 1.56. Код умови у скрипті StoryController

У компоненті Timed Event викликаємо подію, яка буде звертатися до скрипту Start Game Scene та викликати метод Start Game.

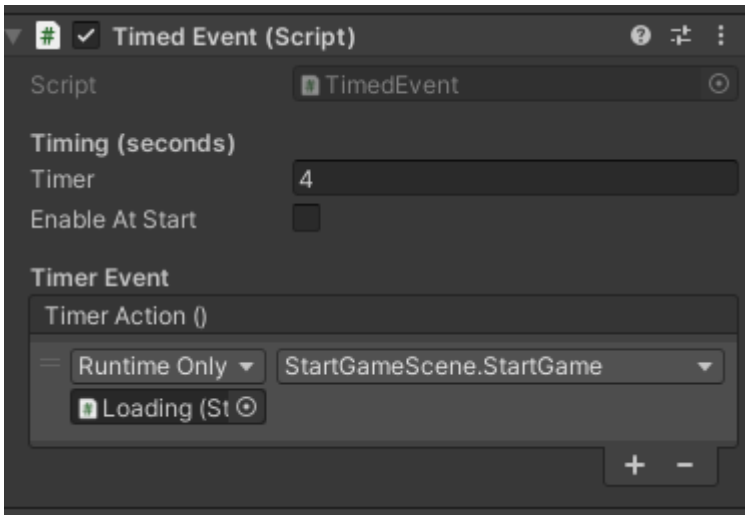


Рисунок 1.57. Налаштування компоненту TimedEvent

1.4.3 Реалізація дизайну головної сцени

Terrain - це компонент і система, призначена для створення та візуалізації природних ландшафтів і поверхонь у 3D-просторі. Terrain дає змогу створювати реалістичні та різноманітні місцевості, такі як гори, пагорби, рівнини, долини тощо, з використанням різних текстур і деталей.

За допомогою інструменту Paint Terrain можна обрати необхідні пензли, їх розмір та непрозорість для створення ігрового ландшафту.

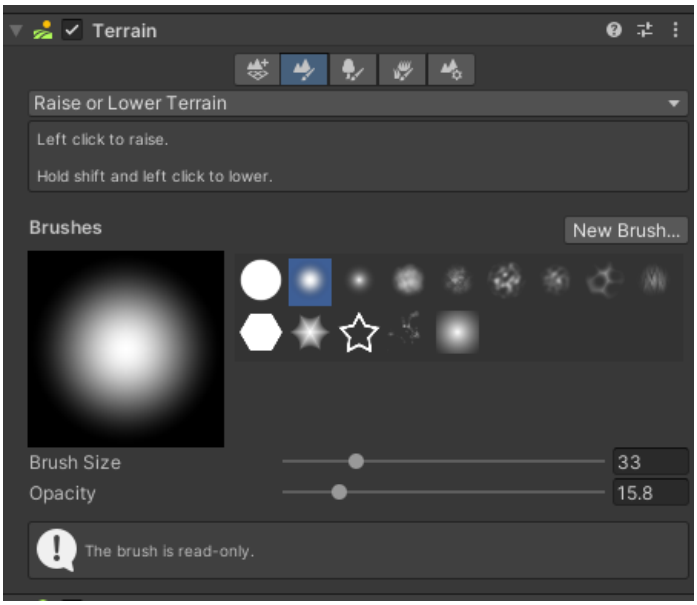


Рисунок 1.58. Інструмент Paint Terrain

Щоб створити необхідну текстуру ландшафту, потрібно обрати функцію Paint Texture та обрати текстуру для майбутньої поверхні.

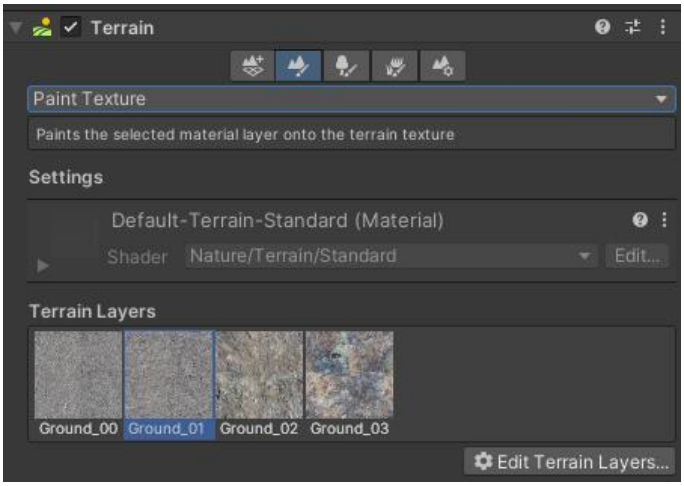


Рисунок 1.59. Функція Paint Texture

Обравши потрібну форму пензлю, за допомогою лівої кнопки миші можна наносити обрану текстуру на поверхню.

Рисунок 1.60. Змінена текстура поверхні ландшафту

За допомогою функції Raise Or Lower Terrain можна змінювати висоту або форму ландшафту на сцені, створювати пагорби, гори, западини та інші елементи. Підняття поверхні здійснюється натисканням лівої кнопки миші, що збільшує висоту обраної області, в той час як зменшення висоти здійснюється натисканням клавіши Shift та лівої кнопки миші.

Рисунок 1.61. Створення пагорбів та западин у ландшафті

Інструмент Paint Details використовується для розміщення деталей на поверхні ландшафту, таких як трава, каміння, квіти та інші об'єкти. Він дає змогу додати необхідні деталі та різноманітність до створеної поверхні, роблячи його більш реалістичним та цікавим. За бажанням можна змінити параметри розміру

області, на яку будуть добавлені деталі; ступінь прозорості, з якою деталі відображатимуться та кількість об’єктів, які додаються, у певній області.

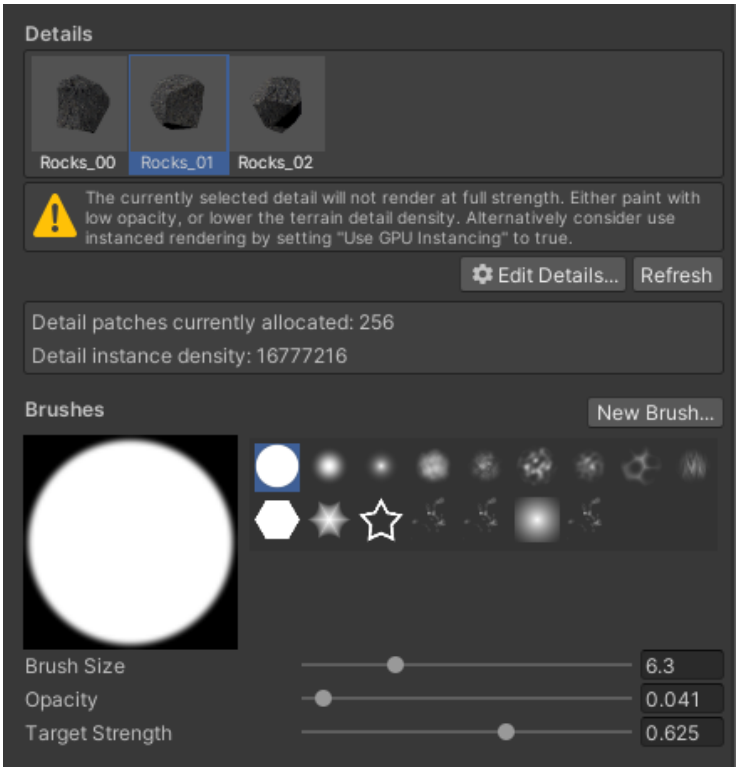


Рисунок 1.62. Інструмент Paint Details

Рисунок 1.63. Створення каменів за допомогою інструменту Paint Details

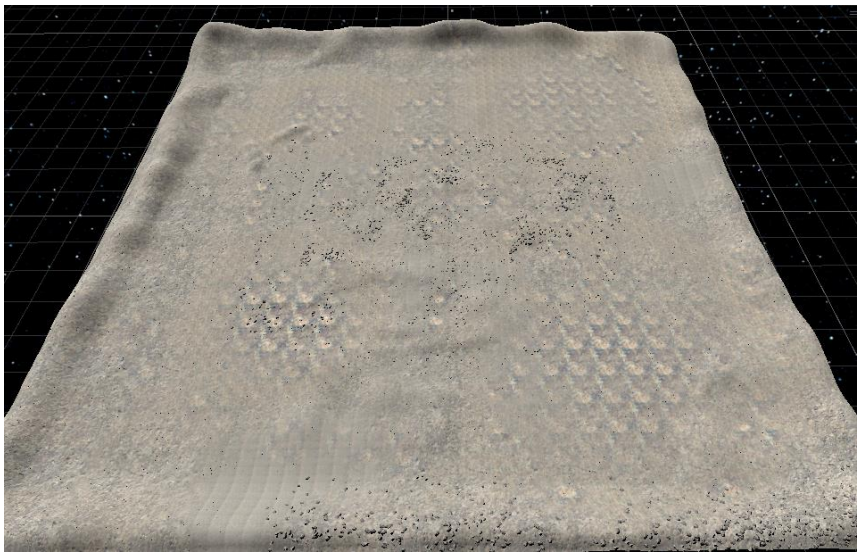


Рисунок 1.64. Створений ландшафт поверхні Місяця

Після створення ігрової поверхні, потрібно створити космічну атмосферу, яка додасть реалізму й атмосферності в ігрове середовище. Вона може створити відчуття перебування у відкритому космосі та підкреслити космічну тематику гри. Щоб це зробити можна використовувати Skybox.

Skybox - це матеріал або набір матеріалів, які оточують ігрову сцену і створюють ілюзію нескінченного неба або навколишнього середовища. Він використовується для створення заднього фону або навколишнього простору в сцені та може включати текстури неба, хмар, гір, зірок та інших елементів.

Для того, щоб обрати матеріал Skybox необхідно натиснути на вкладку Window – Rendering – Lightning – Environment, та обрати Skybox Material. У вкладці Environment можна змінити колір освітлення неба, поверхні землі, інтенсивність світла, вимкнути/увімкнути туман, обрати якість відображення та інше. Натиснувши на кнопку Generate Lightning, Unity автоматично згенерує освітлення для ігрової сцени на основі налаштувань і параметрів, заданих користувачем. Це включає розрахунок і застосування освітлення, тіней, віддзеркалень та інших візуальних ефектів, щоб створити бажану атмосферу і зовнішній вигляд сцени.

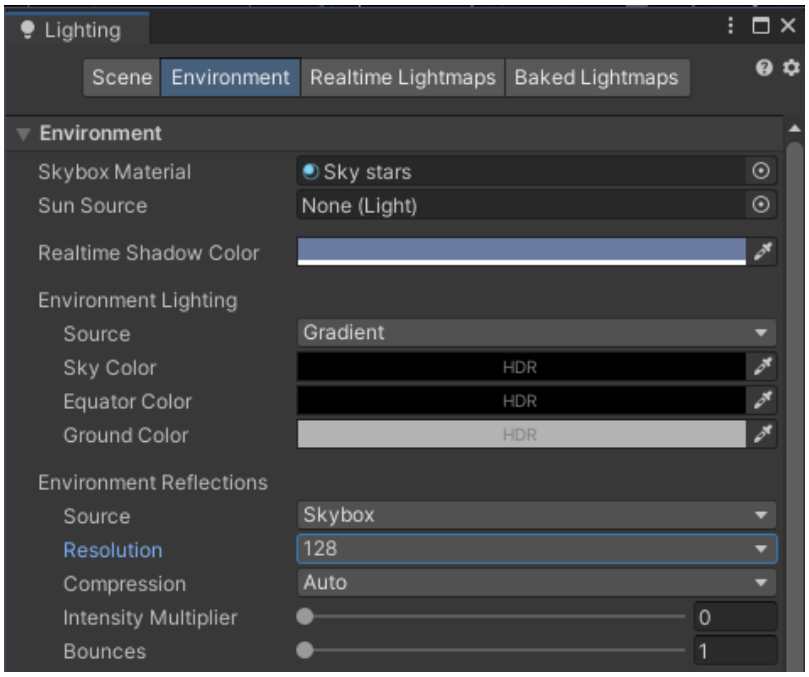


Рисунок 1.65. Налаштування зовнішнього середовища

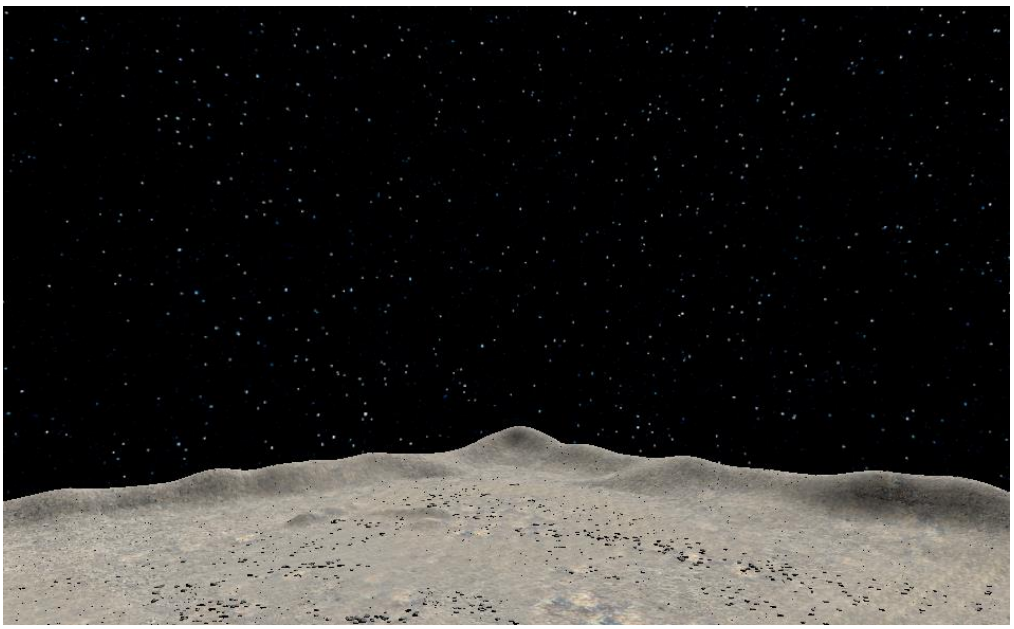


Рисунок 1.66. Створений Skybox для головної сцени

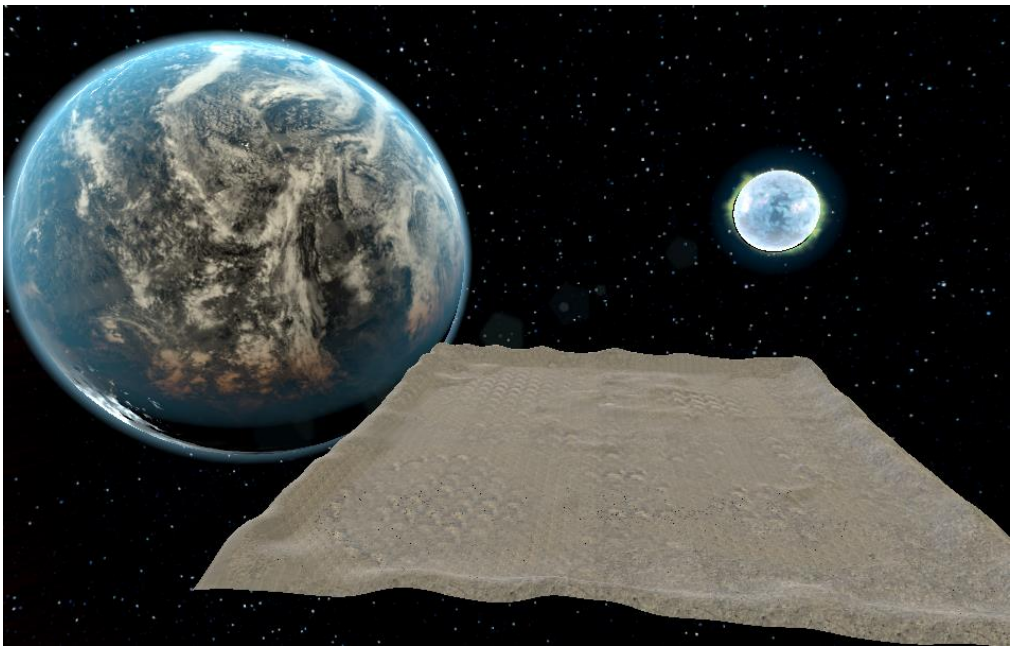


Рисунок 1.67. Вигляд ігрової сцени після
додавання Землі та Сонця

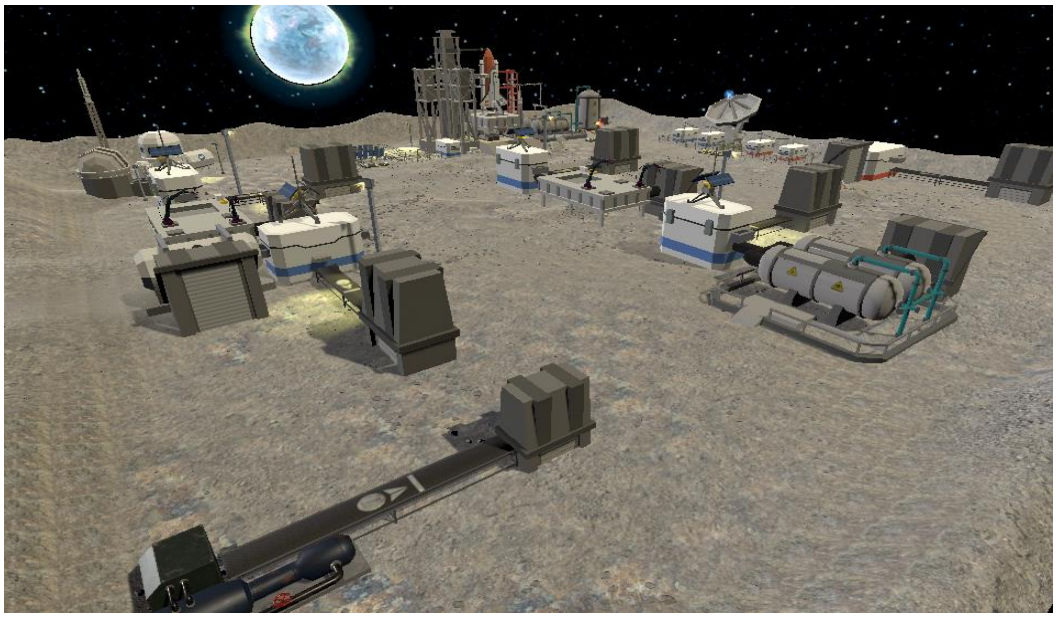


Рисунок 1.68. Вигляд ігрової сцени після
додавання космічних заводів та ракети

Щоб зімітувати ліхтарне світло, я використовував джерело світла Spot Light, яке створює зосереджений промінь світла в певному напрямку.

Змінюючи параметри світла, можна регулювати наскільки далеко від центру об’єкта буде розповсюджуватись світло, кут у градусах при основі прожекторного конуса, колір світла та яскравість.

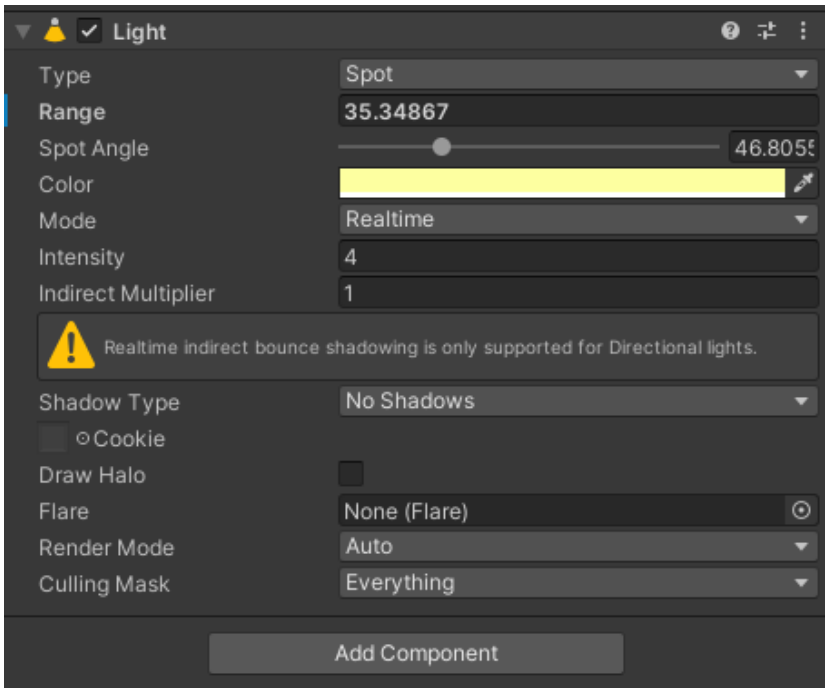


Рисунок 1.69. Налаштування ліхтарного світла

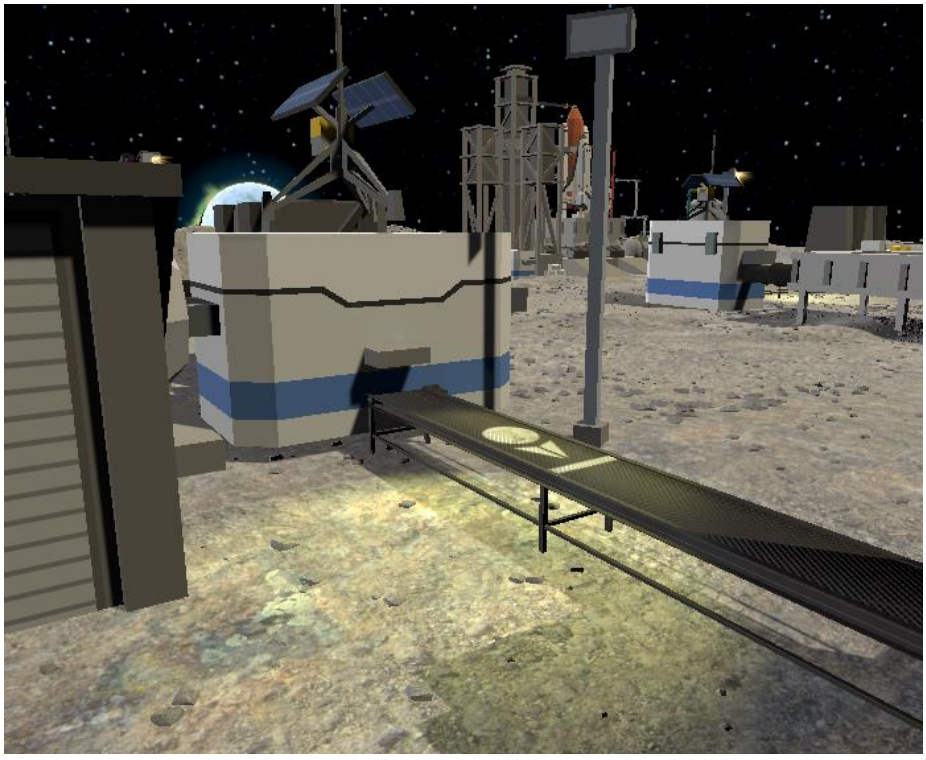


Рисунок 1.70. Створене ліхтарне світло

Для досягнення більшого задоволення від ігрового процесу, було прийнято рішення завантажити візуальні ефекти (Particle System).

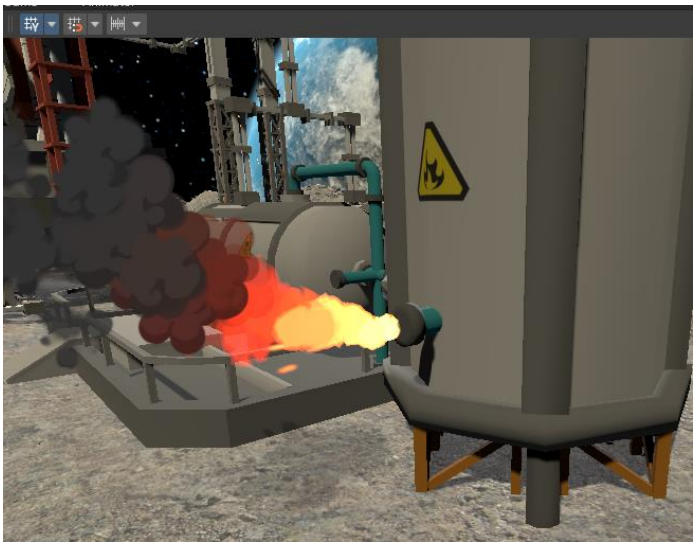


Рисунок 1.71. Візуальний ефект горіння

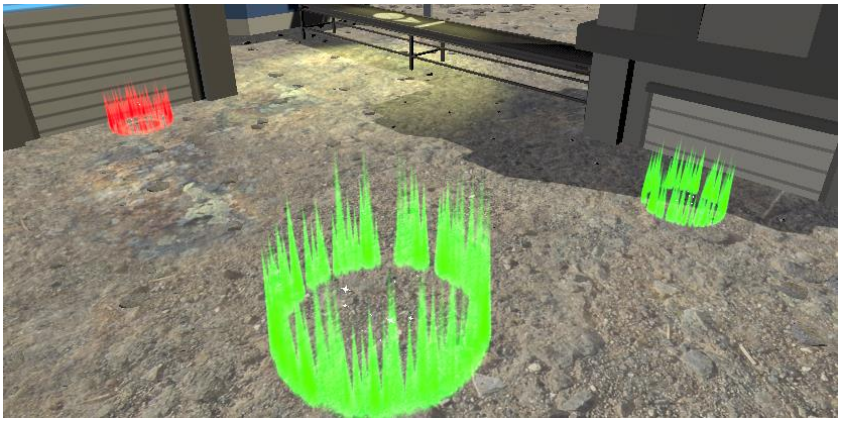


Рисунок 1.72. Візуальний ефект місця
для взаємодії з оточенням

Створюємо інтерфейс головного меню, за допомогою компонентів Image.

Рисунок 1.73. Інтерфейс головної сцени гри

1.5 Тестування ігрового проекту

Короткий огляд етапів створення ігрового проекту з скріншотами можна подивитися у Додаток В.

1.5.1 Тестування сцени головного меню

При старті сцени головного меню з’являється екран з назвою гри та текстом підказкою. Створені анімації працюють без помилок. Візуальні ефекти на фоні працюють без затримок, виконують рухи плавно.

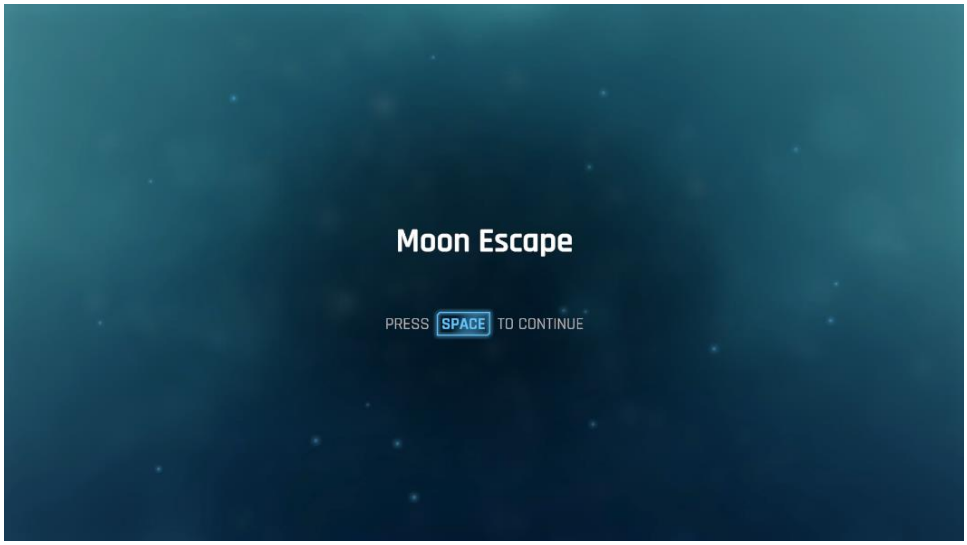


Рисунок 1.74. Екран з назвою гри та текстом підказкою

При натисканні на кнопку Пробіл скриваються непотрібні компоненти та з’являється текст загрузки і анімація завантаження.

Рисунок 1.75. Завантаження другого розділу головного меню

Компоненти другого розділу головного меню з’являються плавно, відсутні помилки у налаштуванні анімацій. Фонова музика грається неперервно, без зупинок. Візуальні ефекти також працюють без помилок.

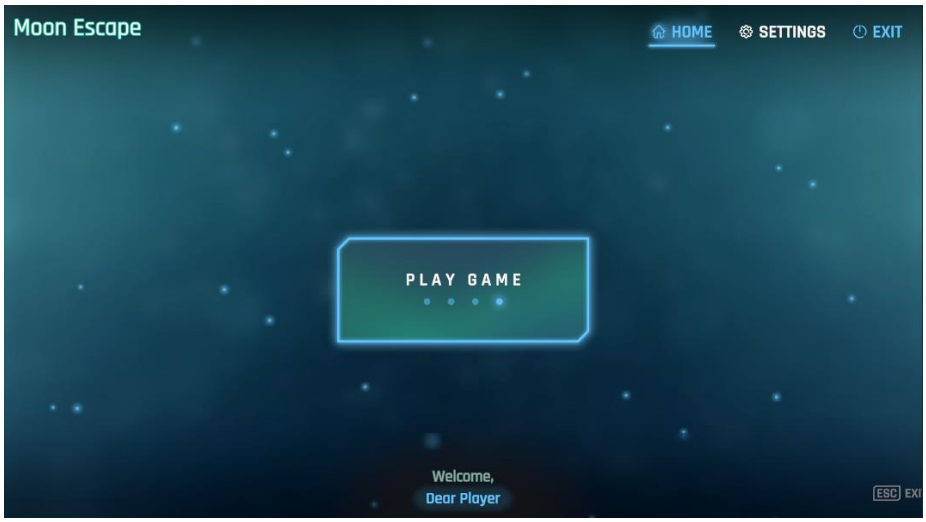


Рисунок 1.76. Другий розділ головного меню

Вкладки Home, Settings та Exit працюють належним чином. Анімація переходу з одної вкладки до іншою виконується плавно.

Зміна гучності загального звуку, звуку музики та звукових ефектів працює без помилок.

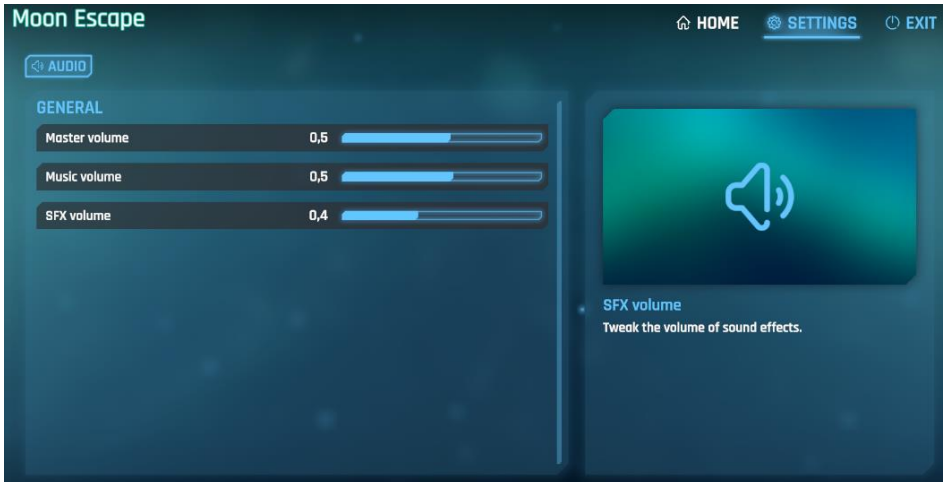


Рисунок 1.77. Вкладка Settings

При натисканні на клавішу Escape або на кнопку Exit, анімація появлення діалогового вікна виходу з гри працює без помилок. Задній фон розмивається, для створення уваги на діалогове вікно. При натисканні на кнопку Cancel,

діалогове вікно згортається, та з’являється головний екран. При натисканні на кнопку Go Ahead помилок не виявлено, гра закривається.

Рисунок 1.78. Діалогове вікно виходу з гри

1.5.2 Тестування сцени з розповіддю сюжету

Після начала сцени, з’являється перша картинка заднього фону, починається анімація друкування тексту, також можна почути озвучення тексту на російській мові та фонове звучання друкування тексту на печатній машинці. Після закінчення написання та озвучення тексту, з’являється текст-підказка з анімацією блимання, зайвих звуків не чутно.

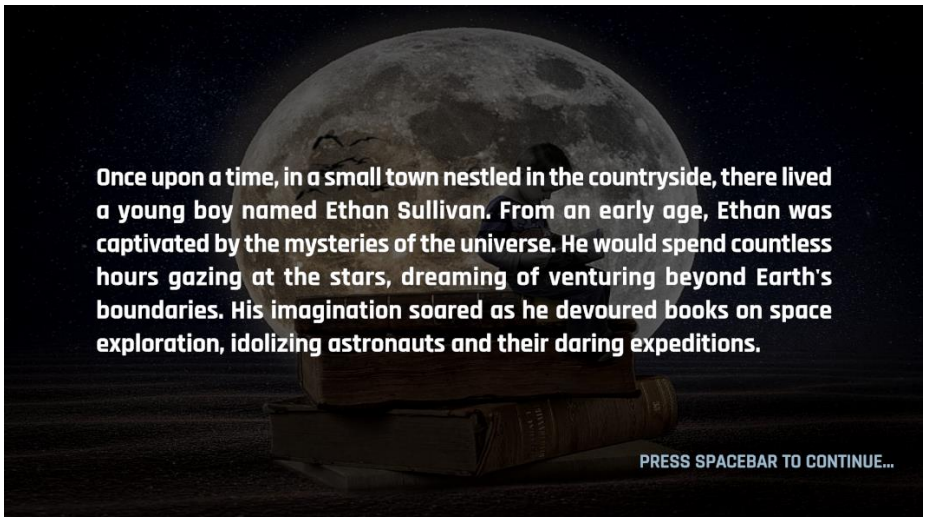


Рисунок 1.79. Сцена розповіді сюжету з першим текстом

При натисканні на клавішу Пробіл, анімація переходу з одного тексту на інший працює плавно.

Усі послідуючі картинки з текстом, озвучення та анімації працюють належним чином, без помилок. Роздільна здатність екрану відповідає параметрам екрану.

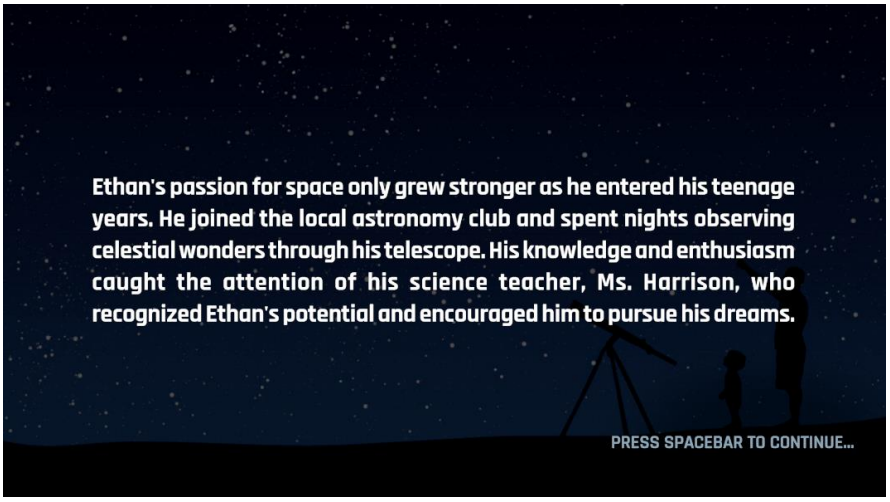


Рисунок 1.80. Сцена розповіді сюжету з другим текстом

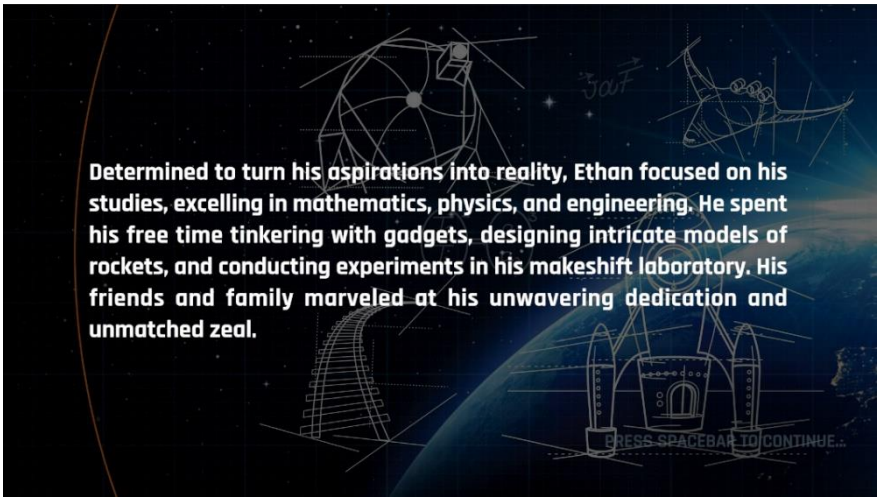


Рисунок 1.81. Сцена розповіді сюжету з третім текстом

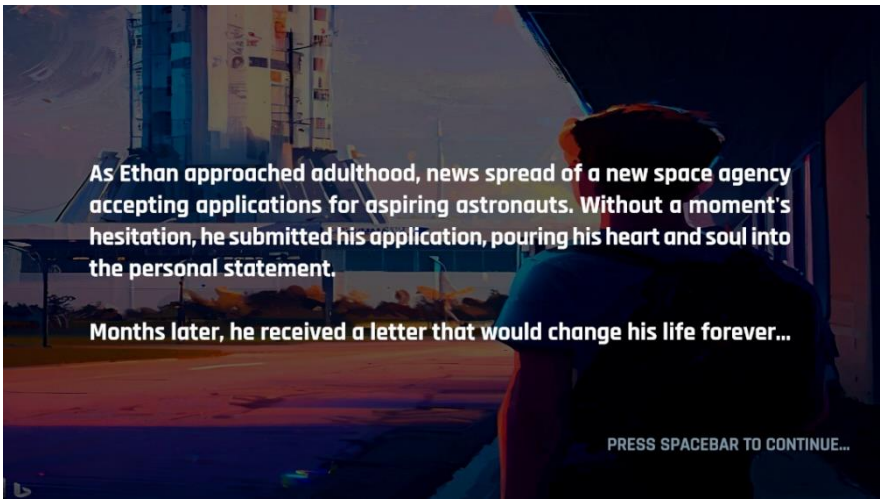


Рисунок 1.82. Сцена розповіді сюжету з четвертим текстом

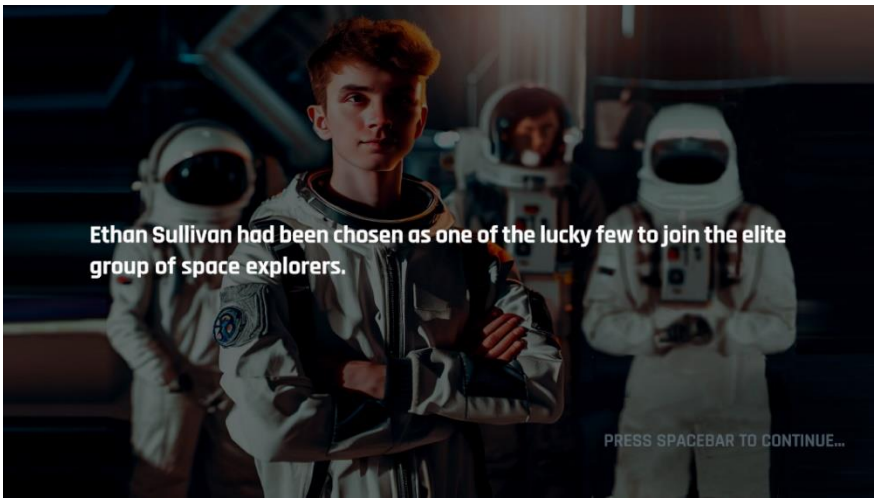


Рисунок 1.83. Сцена розповіді сюжету з п’ятим текстом

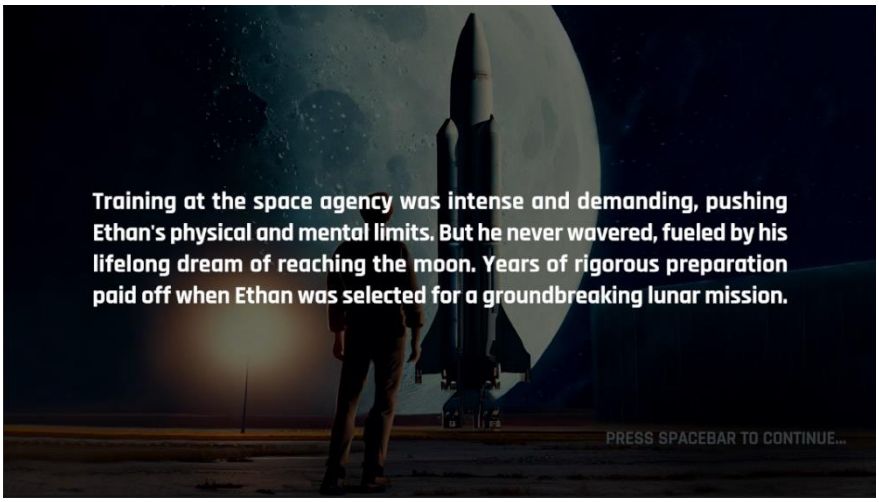


Рисунок 1.84. Сцена розповіді сюжету з шостим текстом

Рисунок 1.85. Сцена розповіді сюжету з сьомим текстом

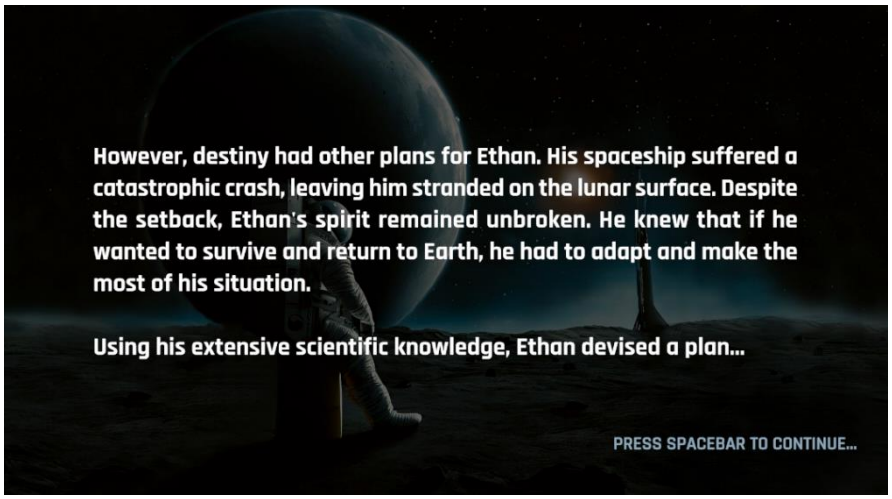


Рисунок 1.86. Сцена розповіді сюжету з восьмим текстом

Після закінчення усього тексту, при натисканні на клавішу Пробіл, текст зникає, озвучення вимикається, програється анімація переходу, з’являється текст та анімація завантаження.

Рисунок 1.87. Завантаження головної сцени

Після чотирьох секунд, гра переходить на головну сцену. Сцена завантажується без затримок завдяки асинхронній загрузці.

1.5.3 Тестування головної сцени

Після завантаження головної сцени, з’являється рівень гри. Космічна атмосфера створює відчуття перебування у космосі та підкреслює тематику гри. Елементи інтерфейсу розташовані належним чином та не заважають при грі. Інтуїтивно зрозуміло що для чого потрібно. Поверхня місяця не має візуальних

недоліків, місць, де можна провалитися під поверхню, немає та добре гармонується з ігровим оточенням.

Рисунок 1.88. Головна сцена гри

Візуальні ефекти працюють добре, без помилок, доповнюють атмосферу гри ті створюють різноманіття.

Рисунок 1.89. Візуальний ефект місця для взаємодії з оточенням

Рисунок 1.90. Візуальний електричний ефект

2 ОХОРОНА ПРАЦІ

Охорона праці є невід'ємною складовою сучасного виробництва і має вирішальне значення для забезпечення безпеки та здоров'я працівників. Розробка заходів з охорони праці у кваліфікаційній роботі бакалавра є необхідною, оскільки вона спрямована на покращення умов праці та запобігання можливим ризикам та нещасним випадкам.

Основні завдання з охорони праці випливають з Конституції України, Закону України про охорону праці, основ трудового законодавства та законодавчих актів. Вони включають забезпечення безпеки та здоров'я працівників, запобігання професійним захворюванням, визначення нормативів і вимог щодо умов праці, регулювання питань щодо організації та здійснення контролю за дотриманням правил охорони праці.

Для аналізу умов праці відповідно до кваліфікаційної роботи бакалавра було обрано робоче місце оператора ПК на підприємстві. Воно є актуальним, оскільки все більше людей працюють з використанням комп'ютерів та інших електронних пристроїв.

2.1 Аналіз та безпека умов праці працівника на робочому місці

2.1.1 Організація робочого місця

Для аналізу умов праці та оцінки шкідливих та небезпечних факторів виробничого середовища, що можуть мати місце під час експлуатації устаткування, необхідно враховувати фізичні, хімічні, біологічні та психофізіологічні аспекти.

Фізичні фактори:

- Неправильна організація робочого місця може призводити до незручної позиції тіла, неправильної постави, що сприяє розвитку м'язово-скелетних захворювань.
- Відсутність адекватного освітлення може спричинити зорове напруження та погіршення зору.

- Недостатня аерація та провітрювання приміщення можуть спричинити збільшення концентрації шкідливих речовин у повітрі та погіршення якості вдихуваного повітря.

Психофізіологічні фактори:

- Довга робота за комп'ютером може призводити до перенапруження зору та розвитку синдрому комп'ютерного зору.
- Відсутність відповідного відпочинку та перерви на роботу може призвести до збільшення психологічного напруження, стресу та виснаження.

Всі елементи робочого місця та їх взаємне розташування повинні відповідати ергономічним вимогам, які враховують характер і особливості трудової діяльності (згідно з ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Робочі місця краще розташовувати так, щоб природне світло падало збоку, переважно зліва. Екран має бути розміщений на оптимальній відстані від очей користувача, що становить 600-700 мм, але не ближче, ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів. Монітор має бути розташований прямо перед працівником, на рівні очей або трохи нижче.

Щоб працівник мав достатньо простору для комфортної роботи, ширина робочого столу повинна бути не менше 120 см, а глибина - не менше 80 см. Висота столу повинна бути від 70 до 75 см, а висота стільця - від 40 до 50 см. Важливо, щоб працівник міг сидіти зі спиною прямою, а підлокітники стільця забезпечували підтримку для плечей та зап'ястків.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до працюючого. Конструкція клавіатури повинна передбачати опорний пристрій, який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-150.

Комп’ютерна миша має бути розташована поруч з клавіатурою на такій висоті, щоб плечі були розслаблені, а зап'ястя не перебували під натиском або в

неприродному положенні. Рекомендована кутова постановка мишки - близько 20-30 градусів.

Працівник повинен мати підтримку для нижньої спини та дотримуватися рівної постави. Рекомендується робити перерви на розтяжку та фізичні вправи кожні 30-60 хвилин роботи за комп'ютером. Під час перерв можна робити прості фізичні вправи, розтяжку м'язів та відпочивання очей.

На один комп'ютер повинно відводиться не менше 6 м^2 площі, при цьому об'єм приміщення повинен бути не менше 24 м^3.

Фактичні значення: площа робочого місця 7 м^2. Об’єм приміщення 120 м^3.

2.1.2 Вимоги безпеки до мікроклімату виробничих приміщень, освітлення та шуму

Основними документами, які визначають параметри мікроклімату виробничих приміщень, є ДСН 3.3.6.042-99 та ГОСТ 12.1.005-88. Ці параметри регулюються для робочої зони - визначеного простору, де знаходяться робочі місця, згідно з нормативним документом ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Температура повітря в приміщенні має бути 21-26 °С. Рекомендована вологість повітря становить 40-60%.

Робочі приміщення повинні бути оснащені достатнім природним та штучним освітленням, щоб забезпечити достатню якість та кількість світла на робочих місцях. Природне освітлення здійснюється через вікна, орієнтовані переважно на схід. Штучне освітлення в приміщені здійснюється системою загального рівномірного освітлення. Рівень освітлення на робочому місці за комп’ютером має бути приблизно в діапазоні від 300 до 500 люкс для забезпечення комфортного та ефективного робочого середовища. Освітлення повинно бути розташоване таким чином, щоб уникнути блискіття на робочій поверхні та утворення непотрібних тіней.

Основне джерело шуму є кондиціонер влітку. Шум має 38 дБ, при роботі за комп’ютером шум не заважає.

Для запобігання виникнення інших шумів у відповідності з ГОСТ 12.1.029-80 зниження шуму й вібрації в приміщенні передбаченні звукоізоляція вікон та дверей.

2.2 Пожежна безпека

Вибухо- та пожежонебезпечні речовини і матеріали на робочому місці та в приміщенні: електричні розетки та перемикачі світла, комп'ютер та його компоненти, кондиціонер, принтер, різні електричні пристрої та зарядні пристрої.

Вибухо- та пожежонебезпечні характеристики:

Електричні розетки та перемикачі світла: коротке замикання може виникнути через пошкодження проводки, вологу або перевантаження мережі. Загорання може виникнути внаслідок перегріву проводки або іскріння.

Комп'ютер та його компоненти: перегрів може виникнути через несправність системи охолодження, пил або перевантаження компонентів. Загорання може виникнути внаслідок короткого замикання або перегріву компонентів.

Кондиціонер: перегрів може виникнути через несправність системи охолодження, забруднення фільтрів або неправильне використання. Загорання може виникнути внаслідок короткого замикання або перегріву компресора.

Принтер: погане підключення, несправність електричного кабелю або живлення, перегрів джерела живлення принтера можуть спричинити коротке замикання і виникнення пожежі.

Різні електричні пристрої та зарядні пристрої: перегрів може виникнути через несправність або неправильне використання. Загорання може виникнути внаслідок короткого замикання або перегріву пристроїв.

Аналіз можливих місць і причин загорань і вибухів у приміщенні:

- Неправильне використання або несправність електроприладів
- Перевантаження електричної мережі
- Несправність або пошкодження електропроводки

- Використання неякісних або несумісних зарядних пристроїв
- Недотримання правил експлуатації електронної техніки

Засоби пожежогасіння:

Пінні вогнегасники використовують для гасіння тліючих матеріалів, горючих рідин. Однак, їх не можна застосовувати для гасіння устаткування, що знаходиться під напругою, для гасіння сильно нагрітих або розплавлених речовин, а також речовин, які вступають з водою в хімічну реакцію, що супроводжується інтенсивним виділенням тепла і розбризкуванням горючої суміші. Також пінні вогнегасники не підходять для гасіння речовин, горіння яких проходить без доступу повітря (бавовна, піроксилін і тому подібне), горючих металів (натрій, магній). Ще один недолік пінних вогнегасників — вузький температурний діапазон: від +5°С до +50°С.

Порошкові вогнегасники – найпоширеніший тип вогнегасників. Вони оснащені порошком, який при попаданні на джерело пожежі заглушує його. Порошкові вогнегасники підходять для гасіння пожеж класу А, В і С, тобто для гасіння твердих речовин, що горять, рідин і газів. Їх можна використовувати у приміщеннях, на вулиці, на автотранспорті та в інших місцях. Використовується для гасіння пожеж від електроприладів, комп'ютерів, кондиціонерів та інших електроустановок.

Вуглекислотні вогнегасники гасять загорання за рахунок значного охолодження зони горіння струминою вуглекислоти CO_2, яка, випаровуючись, перетворюється на вуглекислий газ, який не підтримує горіння. До недоліків вуглекислотних вогнегасників можна віднести можливість обмороження вуглекислотою при необережному їх використанні. Вуглекислотний вогнегасник ефективний для гасіння пожеж класів В та С (рідини та електроустановки). Використовується для гасіння пожеж від електроприладів, комп'ютерів, кондиціонерів та інших електроустановок без пошкодження обладнання.

ВИСНОВКИ

На початку розробки ігрового проекту був проведений детальний аналіз існуючих ігрових жанрів та ігрових двигунів.

Було обрано ігровий жанр квест для ігрового проекту, це рішення виправдано його захопливою ігровою механікою, сюжетною структурою, світобудівним потенціалом, гнучкістю в дизайні квесту, активністю гравця та балансом виклику і винагороди.

Вибір Unity як ігрового двигуна для цього проекту виправданий його доступністю, зручним інтерфейсом, крос-платформенною підтримкою, великою спільнотою та сховищем ресурсів, універсальністю та індустріальною адаптацією. Можливості Unity відповідають практичним цілям проекту, надаючи необхідні інструменти та ресурси для створення захоплюючого ігрового досвіду.

Кінцевим результатом кваліфікаційної роботи бакалавра є розроблений ігровий проект – квест на платформі Unity.

Сутність задачі полягає у розробці гейм дизайну, системного дизайну і дизайну інтерфейсу.

В ході виконання роботи було використано набори готових рішень з онлайн-ринку ресурсів Unity, які допомогли прискорити процес розробки та поліпшити візуальну складову ігрового проекту.

При розробці дизайну було створено багато різноманітних анімацій та візуальних ефектів, які покращують загальний користувальницький досвід, занурює гравців у ігровий світ та надають атмосферу гри.

Розробку ігрового інтерфейсу було проведено згідно зі стандартними роздільними здатностями комп’ютерних моніторів, що дозволяє всьому інтерфейсу бути гнучким для різних форматів моніторів.

Увесь проект був підключений до системи управління версіями GitHub, це створило можливість працювати декільком розробникам одночасно, швидко вносити зміни до проекту, які одразу ж будуть доступні для іншого розробника.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вікіпедія [Електронне джерело] – Ігрові жанри;

URL – https://uk.wikipedia.org/wiki/Жанри_відеоігор

2. Unreal Engine [Електронне джерело] – Ігровий двигун Unreal Engine

URL – https://www.unrealengine.com/en-US

3. CryEngine [Електронне джерело] – Ігровий двигун CryEngine

URL – https://www.cryengine.com/

4. Godot [Електронне джерело] – Ігровий двигун Godot

URL – https://godotengine.org/

5. Unity: [Електронне джерело] – Unity документація;

URL – https://docs.unity3d.com/Manual/index.html

6. Unity Asset Store: [Електронне джерело] – Онлайн-сховище ресурів;

URL – https://assetstore.unity.com/

7. Tracy Fullerton: «Game Design Workshop: A Playcentric Approach to Creating Innovative Games».

8. Jesse Schell: «The Art of Game Design: A Book of Lenses».

9. Kevin Saunders: «Game Development Essentials: Game Interface Design».

10. Joe Hocking: «Unity in action: Multiplatform game development with C#».

11. Alan Thorn: «Unity Animation Essentials».

12. Harrison Ferrone: «Learning C# by Developing Games with Unity».

ДОДАТОК А

Лістинг скрипту StoryController для створення анімації друкування тексту, анімації тексту-підказки та програвання звуку печатання

```
public class StoryController : MonoBehaviour
{
    [Header ("Story texts")]
    [TextArea(1, 10)]
    [SerializeField] private string[] text;

    [Header ("Story")]
    [SerializeField] private TextMeshProUGUI storyText;
    [SerializeField] private ControllerStoryScene controller;
    [SerializeField] private StoryText currentText;

    [Header ("Text Speed")]
    [SerializeField] private float textSpeed;

    [Header ("Speaker")]
    [SerializeField] private SpeechText speakerScript;

    [Header ("Audio")]
    [SerializeField] private AudioSource audioSource;

    [Header("Spacebar To Continue Animation")]
    [SerializeField] private CanvasGroup spaceBarCanvas;
    [SerializeField] private Animator textAnim;

    [Header("Loading")]
    [SerializeField] private CanvasGroup loading;

    [Header("TimedEvent")]
    [SerializeField] private TimedEvent timedEventScript;

    [Header("Start Game Script")]
    [SerializeField] private StartGameScene startGameScript;


    private int _sentenceIndex = -1;
    private int _textIndex = -1;
    private int _textBlinkIndex = 0;

    private bool _isHidden = false;

    private State state = State.COMPLETED;
    private Animator _animator;
```

```
private enum State
{
   PLAYING, COMPLETED
}

private void Start()
{
   _animator = GetComponent<Animator>();
}


private void Update()
{
   if(Input.GetKeyDown(KeyCode.Escape))
   {
      startGameScript.StartGame();
   }
}
private void Speak(int index)
{
   speakerScript.text = text[index];
   speakerScript.Speak();
}

public void Hide()
{
   if(!_isHidden)
   {
      _animator.SetTrigger("Hide");
      _isHidden = true;
   }
}

public void Show()
{
   _animator.SetTrigger("Show");
   _isHidden = false;
}

public void ClearText()
{
   storyText.text = "";
}
public void PlayScene(StoryText text)
{
   currentText = text;
   _sentenceIndex = -1;
   controller.currentTextIndex += 1;
   _textIndex += 1;
   controller.ChangeBackgroundImage();
```

```
    if (_textBlinkIndex == 1)
    {
        StartCoroutine(DisableSpaceBarCanvas());
    }
    else
    {
        spaceBarCanvas.alpha = 0;
    }
    if (_textIndex != 8)
    {
        Speak(_textIndex);
        PlayNextSentence();
        StartCoroutine(TypeSound());
    }
    else
    {
        loading.alpha = 1;
        timedEventScript.StartIEnumerator();
    }
}

public void PlayNextSentence()
{
    StartCoroutine(TypeText(currentText.sentences[++_sentenceIndex].text));
}

public bool IsLastSentence()
{
    return _sentenceIndex + 1 == currentText.sentences.Count;
}

public bool IsCompleted()
{
    return state == State.COMPLETED;
}

private IEnumerator TypeText(string text)
{
    yield return new WaitForSeconds(0.1f);
    storyText.text = "";
    state = State.PLAYING;
    int wordIndex = 0;

    while (state != State.COMPLETED)
    {
        storyText.text += text[wordIndex];
        yield return new WaitForSeconds(textSpeed);
        if(++wordIndex == text.Length)
        {
            state = State.COMPLETED;
            audioSource.Stop();
```

```
                StartCoroutine(EnableSpaceBarCanvas());
                _textBlinkIndex = 1;
                break;
            }
        }
    }

    private IEnumerator TypeSound()
    {
        yield return new WaitForSeconds(0.1f);
        audioSource.Play();
    }

    private IEnumerator EnableSpaceBarCanvas()
    {
        spaceBarCanvas.alpha = 0f;
        yield return new WaitForSeconds(0.15f);
        spaceBarCanvas.alpha = 0.3f;
        yield return new WaitForSeconds(0.15f);
        spaceBarCanvas.alpha = 0.6f;
        yield return new WaitForSeconds(0.15f);
        spaceBarCanvas.alpha = 1f;
        textAnim.SetTrigger("TriggerStart");
    }

    private IEnumerator DisableSpaceBarCanvas()
    {
        spaceBarCanvas.alpha = 1f;
        yield return new WaitForSeconds(0.03f);
        spaceBarCanvas.alpha = 0.6f;
        yield return new WaitForSeconds(0.03f);
        spaceBarCanvas.alpha = 0.3f;
        yield return new WaitForSeconds(0.03f);
        spaceBarCanvas.alpha = 0f;
        textAnim.SetTrigger("TriggerStop");
    }
}
```

ДОДАТОК Б

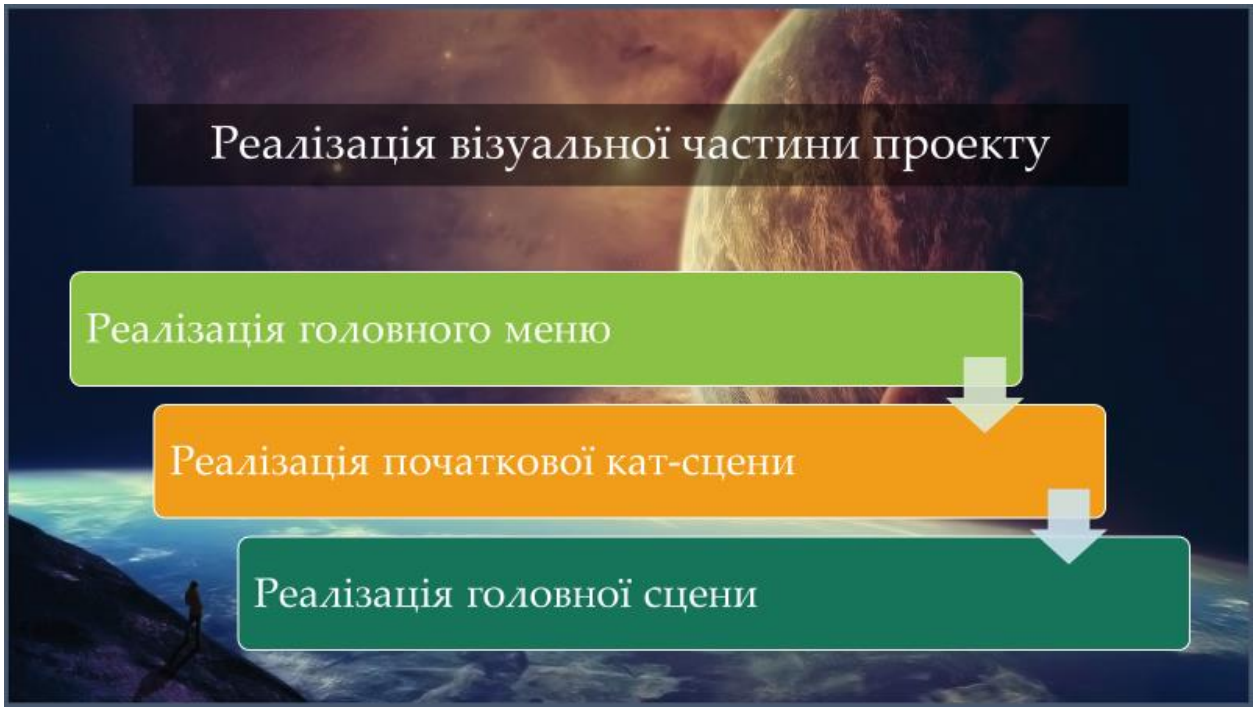
Лістинг скрипту ControllerStoryScene для створення тексту після натискання клавіши Пробіл, а також для зміни фонового зображення

```
public class ControllerStoryScene : MonoBehaviour
{
   [Header ("Background")]
   [SerializeField] private Image background;
   [SerializeField] private Sprite[] backgrounds;

   [Header ("Animator")]
   [SerializeField] private Animator animator;

   [Header ("Index")]
   public int currentTextIndex;

   [Header ("Story")]
   public StoryText currentScene;
   public StoryController storyControllerScript;

   private State state = State.IDLE;
   private enum State
   {
      IDLE, ANIMATE
   }

   void Start()
   {
      currentTextIndex = -1;
      storyControllerScript.PlayScene(currentScene);
   }

   void Update()
   {
      if (Input.GetKeyDown(KeyCode.Space))
      {
         if (state == State.IDLE && storyControllerScript.IsCompleted())
         {
            if (storyControllerScript.IsLastSentence())
            {
               {
                  PlayScene(currentScene.nextText);
               }
            }
            else
            {
               storyControllerScript.PlayNextSentence();
```

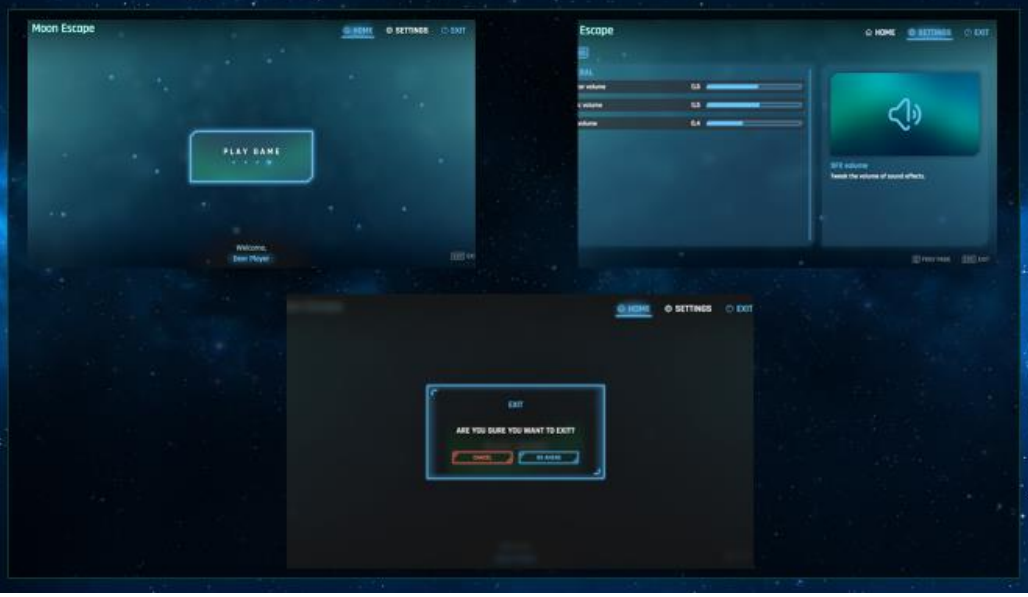
```
            }
        }
    }
}

public void ChangeBackgroundImage()
{
    background.sprite = backgrounds[currentTextIndex];
}

private void PlayScene(StoryText scene)
{
    StartCoroutine(SwitchScene(scene));
}
private IEnumerator SwitchScene(StoryText scene)
{
    state = State.ANIMATE;
    currentScene = scene;
    storyControllerScript.Hide();
    yield return new WaitForSeconds(0.4f);
    ChangeBackgroundImage();
    yield return new WaitForSeconds(0.4f);
    storyControllerScript.ClearText();
    storyControllerScript.Show();
    yield return new WaitForSeconds(0.6f);
    if (currentTextIndex != 8)
    {
        storyControllerScript.PlayScene(scene);
    }
    state = State.IDLE;
}
}
```

ДОДАТОК В

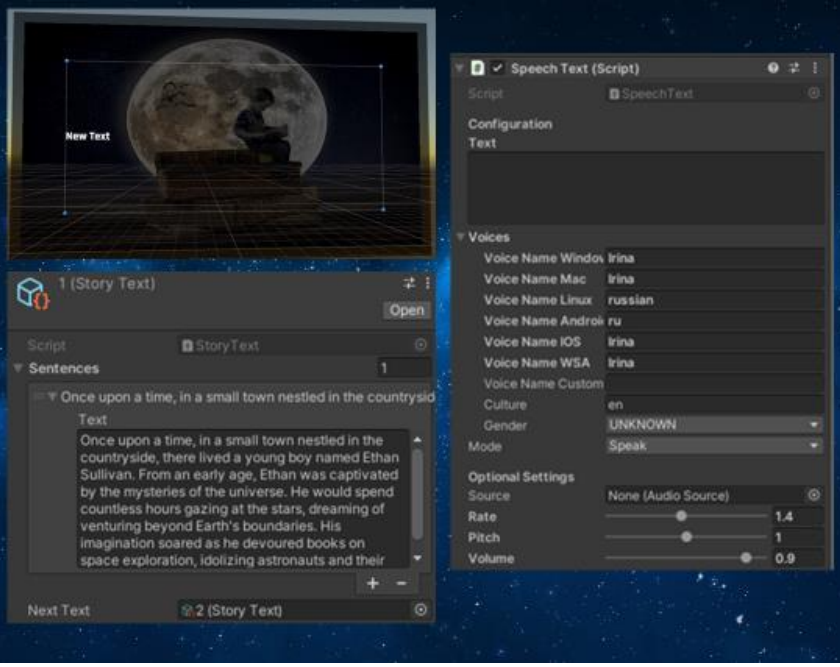
Слайди мультимедійної презентації

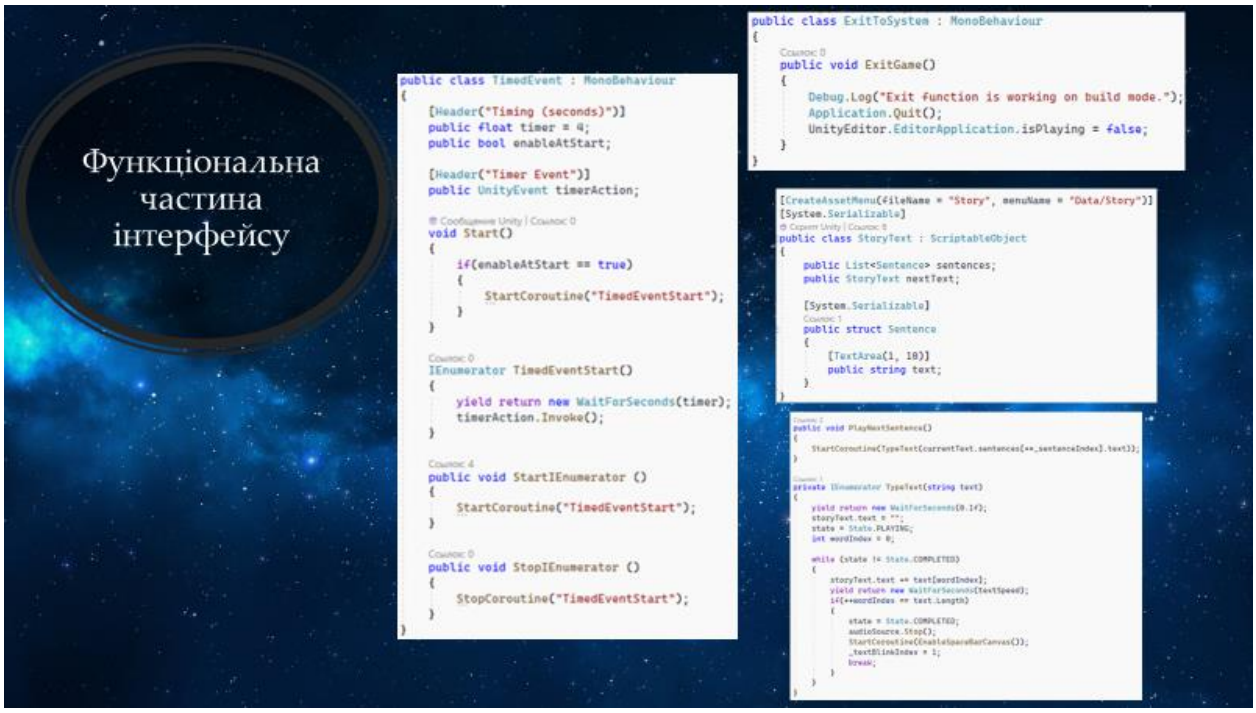


Реалізація головного меню



Реалізація початкової кат-сцени

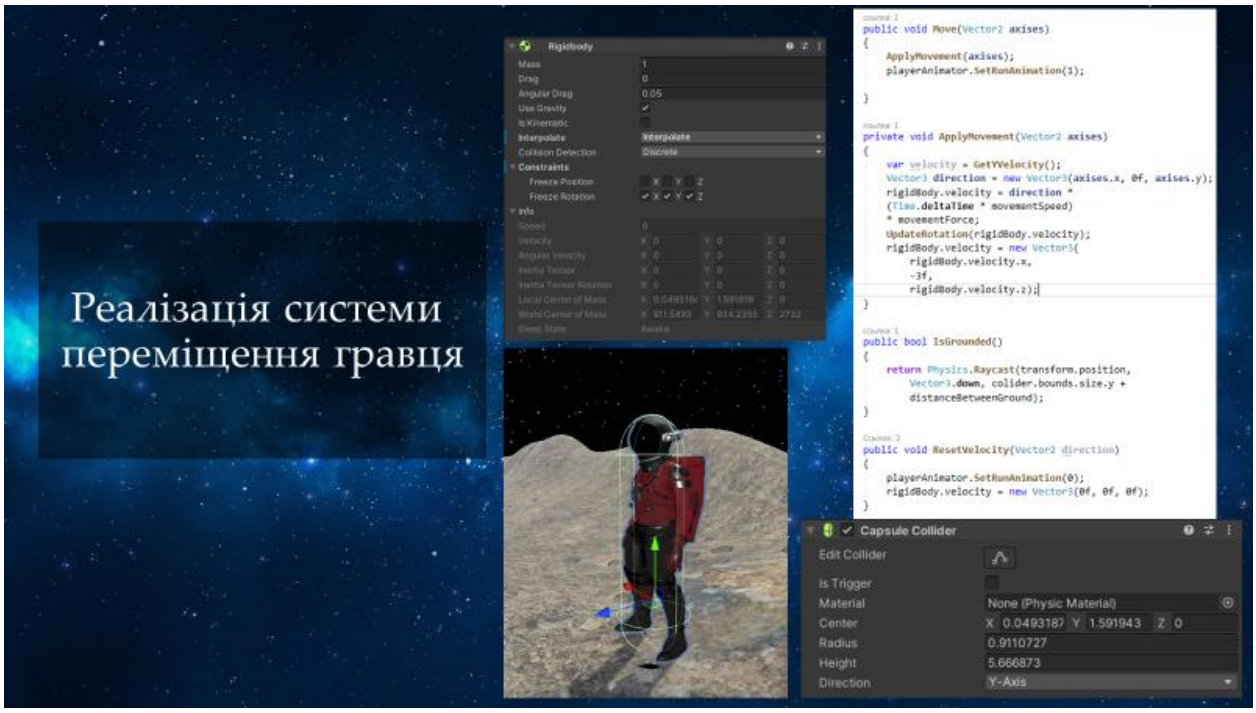



Функціональна частина інтерфейсу
public class TimedEvent : MonoBehaviour
{
[Header("Timing (seconds)")]
public float timer = 4;
public bool enableAtStart;
[Header("Timer Event")]
public UnityEvent timerAction;
void Start()
{
if(enableAtStart == true)
{
StartCoroutine("TimedEventStart");
}
}
IEnumerator TimedEventStart()
{
yield return new WaitForSeconds(timer);
timerAction.Invoke();
}
public void StartIEnumerator ()
{
StartCoroutine("TimedEventStart");
}
public void StopIEnumerator ()
{
StopCoroutine("TimedEventStart");
}
}
public class ExitToSystem : MonoBehaviour
{
public void ExitGame()
{
Debug.Log("Exit function is working on build mode.");
Application.Quit();
UnityEditor.EditorApplication.isPlaying = false;
}
}
[CreateAssetMenu(fileName = "Story", menuName = "Data/Story")]
[System.Serializable]
public class StoryText : ScriptableObject
{
public List<Sentence> sentences;
public StoryText nextText;
[System.Serializable]
public struct Sentence
{
[TextArea(1, 10)]
public string text;
}
}
public void PlayNextSentence()
{
StartCoroutine(TypeText(currentText.sentences[++_sentenceIndex].text));
}
private IEnumerator TypeText(string text)
{
yield return new WaitForSeconds(0.1f);
storyText.text = "";
state = State.PLAYING;
int wordIndex = 0;
while (state != State.COMPLETED)
{
storyText.text += text[wordIndex];
yield return new WaitForSeconds(textSpeed);
if(++wordIndex == text.Length)
{
state = State.COMPLETED;
audioSource.Stop();
StartCoroutine(EnableSpaceBarCanvas());
_textBlinkIndex = 1;
break;
}
}
}


Реалізація головної сцени

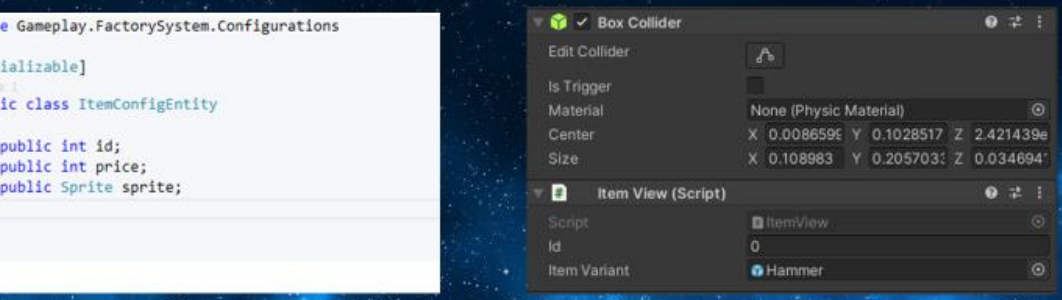
Реалізація функціональної частини проекту

- Реалізація системи переміщення гравця
- Реалізація предметів
- Реалізація системи заводів та складів
- Реалізація системи квестів

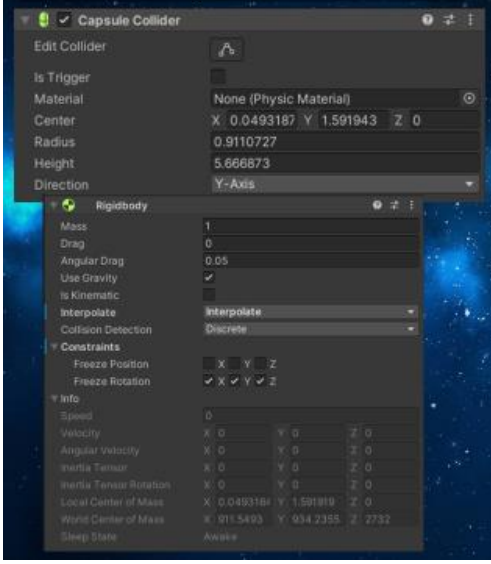


Реалізація предметів

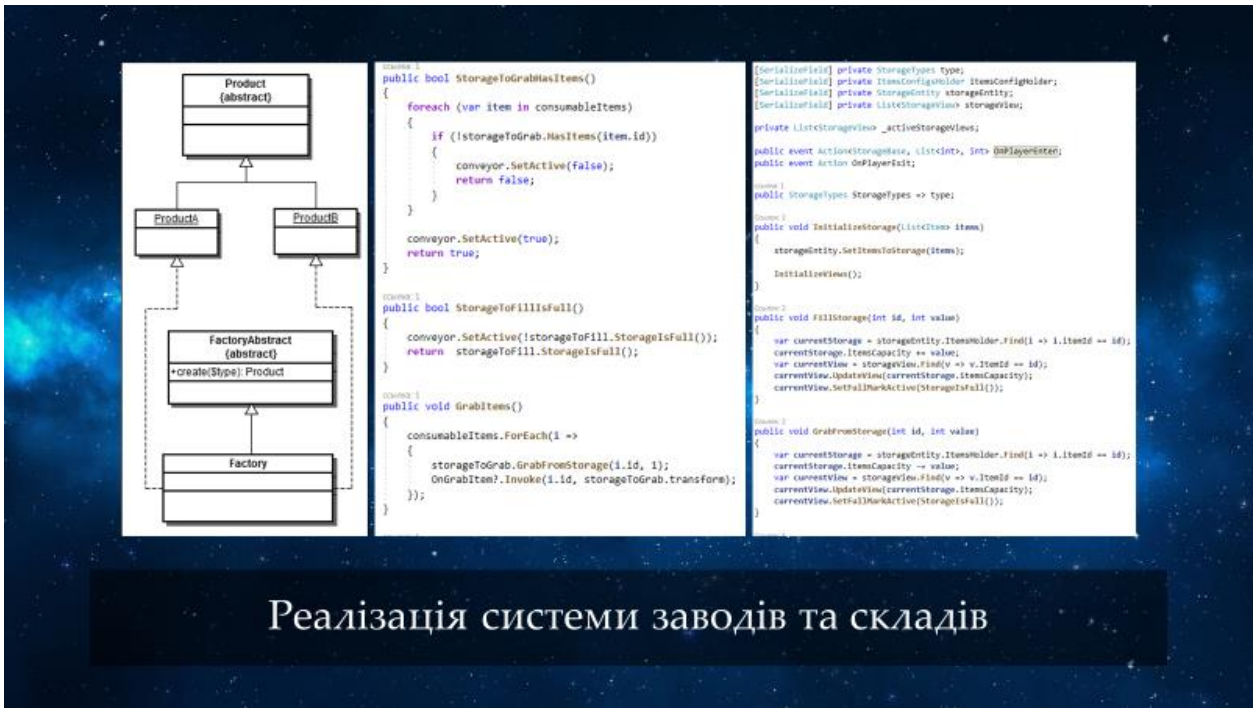
```
namespace Gameplay.FactorySystem.Configurations
{
    [Serializable]
    public class ItemConfigEntity
    {
        public int id;
        public int price;
        public Sprite sprite;
    }
}
```

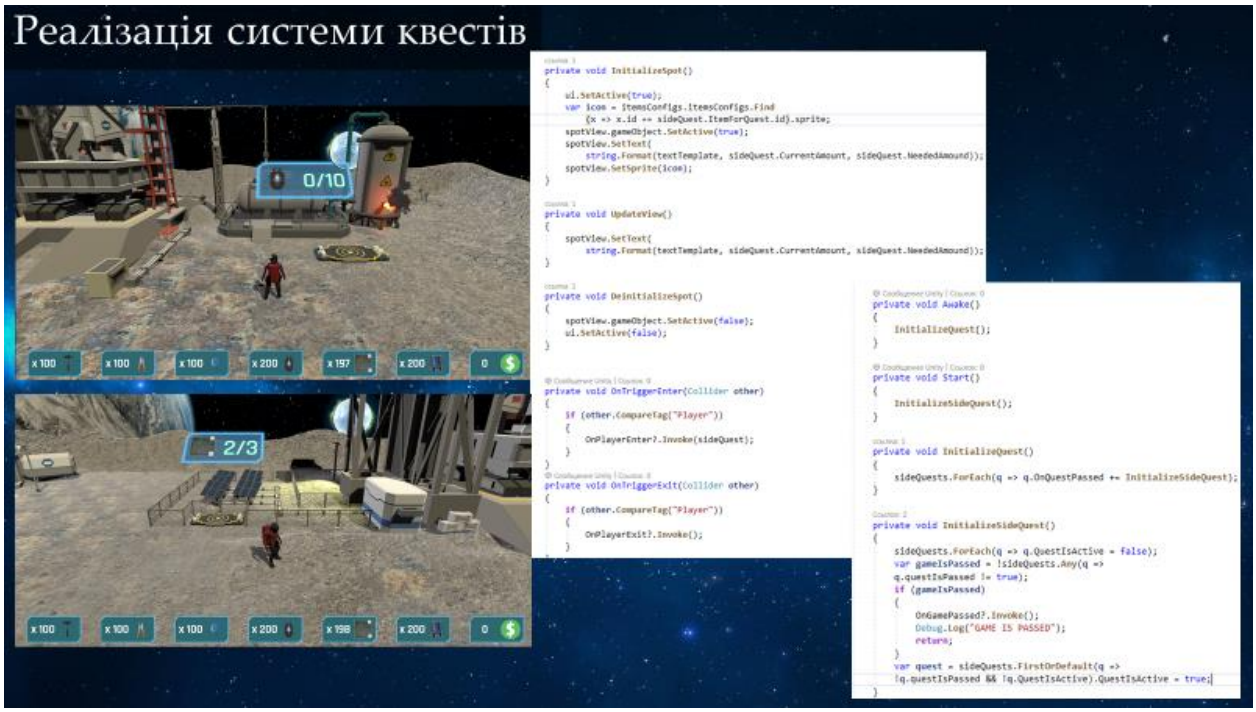


Реалізація системи переміщування гравця




```
[SerializeField] private float movementSpeed;
[SerializeField] private float movementForce;
[SerializeField] private float rotationSpeed;
[SerializeField] private float distanceBetweenGround;
[SerializeField] private Rigidbody rigidBody;
[SerializeField] private Collider colider;
[SerializeField] private PlayerAnimator playerAnimator;


public void Move(Vector2 axises)
{
    ApplyMovement(axises);
    playerAnimator.SetRunAnimation(1);

}

private void ApplyMovement(Vector2 axises)
{
    var velocity = GetYVelocity();
    Vector3 direction = new Vector3(axises.x, 0f, axises.y);
    rigidBody.velocity = direction * (Time.deltaTime * movementSpeed) * movementForce;
    UpdateRotation(rigidBody.velocity);
    rigidBody.velocity = new Vector3(rigidBody.velocity.x, -3f, rigidBody.velocity.z);
}

public bool IsGrounded()
{
    return Physics.Raycast(transform.position, Vector3.down, colider.bounds.size.y + distanceBetweenGround);
}

public void ResetVelocity(Vector2 direction)
{
    playerAnimator.SetRunAnimation(0);
    rigidBody.velocity = new Vector3(0f, 0f, 0f);
}
```


Product
(abstract)
ProductA
ProductB
FactoryAbstract
(abstract)
+create($type): Product
Factory
public bool StorageToGrabHasItems()
{
foreach (var item in consumableItems)
{
if (!storageToGrab.HasItems(item.id))
{
conveyor.SetActive(false);
return false;
}
}
conveyor.SetActive(true);
return true;
}
public bool StorageToFillIsFull()
{
conveyor.SetActive(!storageToFill.StorageIsFull());
return storageToFill.StorageIsFull();
}
public void GrabItems()
{
consumableItems.ForEach(i =>
{
storageToGrab.GrabFromStorage(i.id, 1);
OnGrabItem?.Invoke(i.id, storageToGrab.transform);
});
}
[SerializeField] private StorageTypes type;
[SerializeField] private ItemsConfigsHolder itemsConfigHolder;
[SerializeField] private StorageEntity storageEntity;
[SerializeField] private List<StorageView> storageViews;
private List<StorageView> _activeStorageViews;
public event Action<StorageBase, List<Int>, Int> OnPlayerEnter;
public event Action OnPlayerExit;
public StorageTypes StorageTypes => type;
public void InitializeStorage(List<Item> items)
{
storageEntity.SetItemsToStorage(items);
InitializeViews();
}
public void FillStorage(int id, int value)
{
var currentStorage = storageEntity.ItemsHolder.Find(i => i.itemId == id);
currentStorage.ItemsCapacity += value;
var currentView = storageViews.Find(v => v.ItemId == id);
currentView.UpdateView(currentStorage.ItemsCapacity);
currentView.SetFullMarkActive(StorageIsFull());
}
public void GrabFromStorage(int id, int value)
{
var currentStorage = storageEntity.ItemsHolder.Find(i => i.itemId == id);
currentStorage.ItemsCapacity -= value;
var currentView = storageViews.Find(v => v.ItemId == id);
currentView.UpdateView(currentStorage.ItemsCapacity);
currentView.SetFullMarkActive(StorageIsFull());
}
Реалізація системи заводів та складів


Реалізація системи квестів
0/10
2/3
x 100
x 100
x 100
x 200
x 197
x 200
0
x 100
x 100
x 100
x 200
x 198
x 200
0
private void InitializeSpot()
{
ui.SetActive(true);
var icon = itemsConfigs.itemsConfigs.Find
(x => x.id == sideQuest.ItemForQuest.id).sprite;
spotView.gameObject.SetActive(true);
spotView.SetText(
string.Format(textTemplate, sideQuest.CurrentAmount, sideQuest.NeededAmount));
spotView.SetSprite(icon);
}
private void UpdateView()
{
spotView.SetText(
string.Format(textTemplate, sideQuest.CurrentAmount, sideQuest.NeededAmount));
}
private void DeinitializeSpot()
{
spotView.gameObject.SetActive(false);
ui.SetActive(false);
}
private void OnTriggerEnter(Collider other)
{
if (other.CompareTag("Player"))
{
OnPlayerEnter?.Invoke(sideQuest);
}
}
private void OnTriggerExit(Collider other)
{
if (other.CompareTag("Player"))
{
OnPlayerExit?.Invoke();
}
}
private void Awake()
{
InitializeQuest();
}
private void Start()
{
InitializeSideQuest();
}
private void InitializeQuest()
{
sideQuests.ForEach(q => q.OnQuestPassed += InitializeSideQuest);
}
private void InitializeSideQuest()
{
sideQuests.ForEach(q => q.QuestIsActive = false);
var gameIsPassed = !sideQuests.Any(q =>
q.questIsPassed != true);
if (gameIsPassed)
{
OnGamePassed?.Invoke();
Debug.Log("GAME IS PASSED");
return;
}
var quest = sideQuests.FirstOrDefault(q =>
!q.questIsPassed && !q.QuestIsActive).QuestIsActive = true;
}


Дякуємо за увагу!

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015609350

Дата перевірки:
15.06.2023 10:30:06 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2023 10:31:27 EEST

ID користувача:
100011688

Назва документа: 2БКС-27 Білокаменський С.О.

Кількість сторінок: 67 Кількість слів: 8721 Кількість символів: 70501 Розмір файлу: 21.38 MB ID файлу: 1015257129

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.95%
Схожість

Найбільша схожість: 1.41% з Інтернет-джерелом (http://ir.nmu.org.ua/bitstream/handle/123456789/155750/%2b141-16-3.

5.95% Джерела з Інтернету | 764 ..... Сторінка 69

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | 14 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Білокаменський Сергій Олегович,
здобувачка освіти гр. 2БКС-27, та

Іванова Лілія Вікторівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи молодшого спеціаліста на тему:

«Створення ігрового проекту – квесту на платформі Unity: розробка гейм дизайну, системного дизайну та дизайну інтерфейсу» (автор роботи – Білокаменський С.О., керівник роботи – Іванова Л.В..)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Білокаменський С.О. /

Керівник _____ / Іванова Л.В./

«15» 06 2023 р.

Відокремлений структурний підрозділ

Одеський технічний фаховий коледж ОНАХТ

ВІДГУК

Керівника про кваліфікаційну роботу бакалавра

Білокаменського Сергія Олеговича

(прізвище, ім`я та по батькові)

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Спеціальність *123 «Комп'ютерна інженерія»*

Тема кваліфікаційної роботи

«Створення ігрового проекту – квесту на платформі Unity: розробка гейм дизайну, системного дизайну та дизайну інтерфейсу»

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) Обсяг і якість виконання роботи (розрахунково-пояснювальної записки)

Пояснювальна записка виконана якісно, у достатньому обсязі, відповідно до індивідуального завдання та теми дипломного проекту, розділи пояснювальної записки відповідають етапам рішення завдання, поставленого у дипломному проекті

Презентація виконана якісно, у достатньому обсязі. Презентація наочно демонструє результати роботи.

б) Самостійність роботи над кваліфікаційною роботою

Студент самостійно обрала напрям та тематику кваліфікаційної роботи. Провів аналіз існуючих рішень і зробив необхідні висновки для реалізації проекту. Виявив навички самостійно опрацьовувати новий матеріал та виконувати пошук необхідної літератури та інших джерел інформації

в) Теоретична підготовка бакалавра_____

відповідає вимогам, що надаються до бакалавра зі спеціальності «Комп'ютерна інженерія»

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва_____

У дипломному проекті розглянута та реалізована цікава тема створення ігрової платформи на Unity на основі якої розроблена демонстраційна казуальна гра у жанрі у жанрі квест, айдт, пригода, створений з використанням Unity та мови програмування C#, яка може зацікавити комерційні структури ринку комп'ютерних розваг. Застосовані сучасні програмні засоби для реалізації програмного забезпечення.

Загальна оцінка_____5(відмінно)_____

Прізвище, ім`я, по батькові_____Іванова Лілія Вікторівна_____

Місце роботи і посада керівника проекту_ВСП «Одеський технічний фаховий_____коледж__ОНТУ» к.т.н., зав. кафедрою Комп'ютерної інженерії_____

Підпис _____

«_15_»_____06_____2023р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти

відділення комп'ютерних систем

Білокаменського Сергія Олеговича

(прізвище, ім`я та по батькові)

Спеціальність 123 Комп'ютерна інженерія

Освітня програма Комп'ютерна інженерія

Керівник кваліфікаційної роботи Іванова Лілія Вікторівна

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *«Створення ігрового проекту – квесту на платформі Unity: розробка гейм дизайну, системного дизайну та дизайну інтерфейсу»*

Обсяг розрахунково-пояснювальної записки 61 сторінок

Обсяг графічної (презентаційної) частини 10 аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

кваліфікаційна робота у повному обсязі відповідає темі та завданню

б) характеристика виконання кожного розділу кваліфікаційної роботи

Кваліфікаційна робота складається з розділів: Передмова. Аналіз ігрових жанрів. Аналіз ігрових двигунів. Розробка технічного завдання для ігрового проекту. Створення ігрового проекту. Тестування ігрового проекту. Розділ охорони праці. Висновок. Перелік використаних джерел інформації. Кожен розділ присвячено одному з етапів виконання завдання кваліфікаційної роботи та містить необхідну інформацію щодо результатів виконаної роботи.

в) оцінка якості виконання пояснювальної записки та графічної частини кваліфікаційної роботи

Пояснювальна записка виконана якісно, у достатньому обсязі, відповідно до індивідуального завдання та теми дипломного проекту, розділи пояснювальної записки відповідають етапам рішення завдання, поставленого у дипломному проекті

Презентація виконана якісно, у достатньому обсязі. Презентація наочно демонструє результати роботи.

г) перелік позитивних якостей кваліфікаційної роботи

1. Актуальна тематика
2. Сучасні технології реалізації програмного продукту
3. Якісне подання результатів роботи

д) основні недоліки кваліфікаційної роботи Етапи виконання роботи варто було подати більш ілюстративно

Оцінка розрахункової частини відмінно

Оцінка графічної частини відмінно

Загальна оцінка відмінно

Прізвище, ім'я, по батькові рецензента Царьов Роман Юрійович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, старший викладач кафедри комп'ютерної інженерії та інформаційних систем

Підпис:

« 23 » червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ * 43997335 * ВІДДІЛ КАДРІВ