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

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Обслуговування комп'ютерних систем і мереж»

Група: 4КС-57

Дипломний проект

здобувача освіти денної форми навчання
КС.57.16.000.ДП

ПАЛАДІ
СЕРГІЯ ВАЛЕРІЙОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Освітньо-професійна програма: «**Обслуговування комп'ютерних систем і мереж»**

Група: **4КС-57**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity

Проектний матеріал складається з пояснювальної записки на 71 сторінках та графічного (презентаційного) матеріалу на 12 аркушах (слайдах)

Дипломник _____ (Паладі С.В.)

Керівник _____ (Шувалова І.О.)

Консультанти:

з економічного розділу _____ (Іванченков В.С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «20» 06 2024 р. Протокол ЕК № 4

Оцінка ЕК 3 (задовільно) 70.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Обслуговування комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"15" 11 2024 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Паладі Сергію Валерійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity*

затверджена наказом по коледжу від "02" 11 2023 р. № 244-А2-ОД

2. Термін здачі закінченого проекту 10.06.2024

3. Вихідні данні до проекту *Використання програмного рушія Unity; Використання мови програмування C# та бібліотек Unity для розробки гри; Використання принципів модульності у проектуванні та розробці ігрових проектів; Реалізація системи контролю та передачі даних між рівнями комп'ютерної гри в жанрі адвенчура; Гра повинна мати основні признаки жанру адвенчура; Виконана частина повинна давати змогу гравцю зберігати пройдений ігровий прогрес та характеристики гравця у наступні рівні..*

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

Аналіз ігрового жанру адвенчура; Аналітичний огляд програмних рушіїв з умовно безкоштовним доступом; Вибір інструментів розробки; Формування концепту контролю та передачі даних між рівнями; Проектування системи контролю та передачі даних між рівнями; Реалізація елементів системи контролю та передачі даних між рівнями; Тестування працездатності реалізованих елементів.

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

Особливості ігрового жанру адвенчура; Особливості програмного рушія Unity та причини його вибору для розробки проекту; Особливості механіки системи контролю та передачі даних між рівнями; Огляд елементів системи контролю та передачі даних між рівнями; Принцип роботи елементів системи контролю та передачі даних між рівнями; Етапи реалізації елементів системи контролю та передачі даних між рівнями; Хід тестування гри; Скріншоти реалізованої гри.

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Шувалова І.О.*		
Економічний розділ	*Іванченков В.С.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7 Дата видачі завдання 15.01.2024

Керівник *Шувалова І.О.* _____
(підпис)

Завдання прийняв до виконання *Паладі С.В.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	*20.05.2024*	виконав
2	*Аналіз ігрового жанру 2D адвенчура*	*23.05.2024*	виконав
3	*Аналітичний огляд та вибір програмного рушія*	*25.05.2024*	виконав
4	*Формування концепту системи контролю та передачі даних*	*28.05.2024*	виконав
5	*Проектування системи контролю та передачі даних*	*30.05.2024*	виконав
6	*Проектування графічної частини гри*	*01.06.2024*	виконав
7	*Реалізація графічної частини гри*	*03.06.2024*	виконав
8	*Реалізація системи контролю та передачі даних*	*05.06.2024*	виконав
9	*Імплементація та відлагодження ігровіх елементів*	*07.06.2024*	виконав
10	*Тестування працездатності елементів гри*	*09.06.2024*	виконав
11	*Виправлення виявлених помилок*	*10.06.2024*	виконав
12	*Аналіз результатів, підготовка слайдів презентації*	*11.06.2024*	виконав
13	*Економічні розрахунки та питання з охорони праці*	*12.06.2024*	виконав
14	*Підготовка графічної частини проекту*	*13.06.2024*	виконав
15	*Підготовка проекту до захисту та тестування ПП*	*14.06.2024*	виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КС 57. 16 000. 00 ДП	Дипломний проект		
А4			КС 57. 16 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 57. 16 000. 00 ДП		
Розробив		Паладі С.В.		10.06	Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity	Літ.	Аркуш 4 / Аркушів 71
Перевірив		Шувалова І.О.		10.06		НДП	
Н. контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4КС-57	
Затв.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

Важливою частиною будь-якого ігрового продукту є збереження та передача ігрових даних між ігровими сесіями, або рівнями гри. На початку ігрової індустрії ігри не мали такого функціоналу, оскільки не було можливості зберігати інформацію у носії інформації, а самі ігрові продукти формувались таким чином, що ігровий процес намагались умістити в одну ігрову сесію, тому багато старих ігор для ігрових приставок можна пройти за вечір.

У наш час складно уявити собі гру, яка б не зберігала ігровий процес гравця, хіба що це була б основна особливість проекту. Втім, процес збереження та передачі інформації у ігрових проектах не є тривіальною задачею.

Темою мого дипломного проекту є «Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity». Для гри в жанрі адвенчура важливо передавати інформацію між рівнями, оскільки вони можуть бути різної довжини, а ігровий процес може включати в собі розгалуження, які необхідно брати до уваги під час завантаження нового рівня.

Те, яким чином буде виконуватись контроль та передача даних між рівнями в адвенчурі, залежить від того, яку саме інформацію необхідно буде передавати. Окрім того свою особливість у цей процес додає використання ігрового програмного рушія Unity, оскільки він використовую систему сцен, які самі по собі відокремлені одна від одної.

Оскільки виконання проекту буде виконуватись на ігровому програмному рушії Unity, виконання коду виконуватиметься на мові програмування C#, що дає змогу виконувати розробку із використанням принципів ООП. Також, є доцільним розробляти систему таким чином, щоби у подальшому її можна було використовувати у інших ігрових проектах.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз ігрового жанру адвенчура

1.1.1 Загальний огляд питання

Для вирішення поставленої мети роботи є необхідність визначитись із початковими питаннями, які будуть основою для кінцевої розробки. Оскільки темою мого дипломного проекту є «Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity», одним з таких питань є визначення жанру адвенчура. Це, можливо, дасть змогу визначитись із вирішенням деяких питань проектування та реалізації проекту.

Ігрові жанри в ігровій індустрії представлені дуже широким переліком, яких не менше, а в деякому сенсі навіть більший ніж перелік кінематографічних жанрів. Ігровий жанр визначається ігровими механіками, прийомами зовнішнього відображення, елементами ігрового процесу, розташування ігрової камери. До ігрових жанрів відносять багато найменувать, але існують основні ігрові жанри, а також похідні, які можуть виділятись своєю особливістю, чи гібридні, які поєднують особливості декількох жанрів.

Для отримання відповіді на питання визначення жанру, потрібно розкласти його на основні характеристики, притаманні йому. Адвенчура, або пригодницькі ігри, – це відеоігри, які акцентують на сюжеті та дослідженні світу гри, часто з акцентом на розгадуванні головоломок, взаємодії з персонажами та розв'язанні завдань. Цей жанр відомий своєю здатністю занурювати гравця в багатий та деталізований віртуальний світ.

Ці ігри зазвичай не містять швидких бойових дій або високих вимог до реакції гравця, натомість пропонуючи більш розмірений та роздумувальний ігровий досвід. Для більш детального вивчення питання, потрібно провести аналіз аналогічних ігрових проектів, що реалізовані в жанрі адвенчура.

1.1.2 Огляд прикладів ігор в жанрі адвенчура

Щоб виконати аналіз ігрового жанру опираючись на практичні дані, потрібно провести аналіз існуючих ігрових проектів з метою перейняти від них

інформацію про ігровий жанр з точки зору теми мого проекту.

Myst – гра, створена у 1993 році компанією Cyan Inc, для цільової платформи Macintosh. Після цього гра була багато разів портована на інші платформи, в тому числі на ОС Windows. Гра була орієнтована на гру від першого лиця та головною метою гри було вивчення загадкового острову Мист, розгадуючи загадки, гравець відкривав для себе нові сторінки історії острову та частини ігрового сюжету.

Завдяки візуальному стилю та антуржу, а також кількості загадок для гравця, гра створювала правильну атмосферу загадки, яка спонукала гравця рухатись далі і далі по острову, розгадуючи все нові і нові загадки.

Отже до основних характеристик гри Myst можна віднести сюжет в якому гравець починає гру як безіменний персонаж, який виявляє книгу з назвою Myst. Відкривши її, він потрапляє на таємничий острів з такою ж назвою. Головна мета полягає в тому, щоб дослідити острів та розгадати його таємниці.

Гра включає безліч головоломок, які гравці повинні вирішити, щоб розкрити секрети острова та його історію. Myst дозволяє гравцям вільно досліджувати світ і вирішувати головоломки в будь-якому порядку, що було нововведенням для жанру в той час.

З графічної точки зору, гра використовувала передові на той час технології рендерингу для створення високоякісних статичних зображень, які створювали відчуття реалістичності та іммерсії. Звуковий дизайн та музика Myst також відіграють важливу роль, створюючи атмосферу та допомагаючи гравцям зануритися в світ гри. Myst мав великий вплив на розвиток жанру пригодницьких ігор і сприяв популяризації CD-ROM як носія для відеоігор. Гра залишається класикою, яка цінується за свою атмосферу, інновації та глибокий іммерсивний досвід. На рисунку 1.1 зображено скріншот із гри Myst:

Рисунок 1.1. Скріншот з гри Myst

Крім зазначених ігор були й інші проекти, такі як Full Throttle, що також отримала культовий статус у ігровій індустрії, розширяючи ігровий жанр.

The Secret of Monkey Island – це класична гра, де гравці вирішують головоломки та взаємодіють з персонажами в гумористичному піратському світі. Вона була розроблена у 1990 році компанією Lucasfilm Games, та отримала перевидання у 2009.

В цій грі потрібно було виконувати ряд завдань у декораціях піратського сетінгу. Більшу частину ігрового процесу гравцю потрібно було переміщуватись по рівням, досліджуючи локації, збирати предмети, розмовляти з персонажами, інколи вступаючи у бій.

The Secret of Monkey Island стала культовою для свого часу. Гра є першою в серії Monkey Island і вважається однією з найвеличніших адвенчура ігор усіх часів. Основні аспекти гри можна поділити на декілька складових.

З точки зору сюжету, гравці керують Гайбрашем Тріпвудом, молодим чоловіком, який мріє стати піратом. Він прибуває на острів Мелі, де починає

свою пригоду, шукаючи способи довести свою піратську майстерність. Гра відома своїм іронічним гумором, остроумними діалогами та незабутніми персонажами.

Отже, The Secret of Monkey Island пропонує різноманітні головоломки, які гравці повинні розгадати, щоб просунутися в грі. Для свого часу гра мала вражаючу графіку та якісний саундтрек, які допомагали створити неповторну атмосферу.

Гра використовує графічний інтерфейс SCUMM, який дозволяє гравцям взаємодіяти з об'єктами та персонажами за допомогою простих команд. The Secret of Monkey Island мала великий вплив на розвиток жанру адвенчура і до сьогодні залишається популярною серед фанатів. Завдяки своїй оригінальності, вона стала еталоном для багатьох наступних ігор. У тому ж графічному стилі було створено гру Full Throttle, яка не безрезультатно намагалась повторити успіх гри The Secret of Monkey Island. На рисунку 1.2 зображено скріншот цієї гри:

Рисунок 1.2. Скріншот гри The Secret of Monkey Island

Life is Strange – серія відносно нових ігор в жанрі адвенчура. Перша гра із серії вийшла у 2015 році компанією Dontnod Entertainment. У всіх іграх цієї серії

основою ігрового процесу було вивчення локацій гри, а також увага до деталей оточення. Гра представляла собою дуже сильну розповідь про відношення двох старих подруг, які зустрілись напередодні великого лиха. Перед гравцем є можливість приймати рішення, які впливатимуть на персонажів гри та на кінцівку сюжету.

Основний опір гра робила на візуальний та музикальний супровід, створюючи потрібну атмосферу для гравця, яка сприяла до занурення у ігровий процес. Також, важливу частину занурення відігравала наративна складова проекту, через блокнот головної героїні. Бойової системи, як такої, не було. Замість неї використовувалась система Quick Time Event яка при не виконанні умов створювала негативні наслідки для персанажу гри. Ця ж система відповідала за ключові моменти в сюжети, коли гравцю потрібно було дуже швидко приймати рішення.

Серед особливостей гри Life is Strange можна виділити сильний сюжет. Гра розповідає історію Макс Колфілд, студентки фотографії, яка виявляє, що може повертати час назад. Використовуючи свої нові здібності, вона намагається розкрити таємниці зникнення своєї однокласниці Рейчел Ембер. Гравці стикаються з рішеннями, які впливають на подальший розвиток сюжету та взаємини з іншими персонажами.

Гра отримала досить неординарну систему продажу, оскільки була поділена на епізоди, що дозволяло розповідати історію з розвитком персонажів та сюжетних ліній.

Life is Strange має унікальний художній стиль та саундтрек, які допомагають створити емоційну атмосферу. Гра отримала визнання за свою здатність залучити гравців до глибоких тем, таких як дружба, сім'я, відносини, самовизначення та наслідки вибору. Life is Strange вплинула на багатьох гравців своєю реалістичністю та емоційною глибиною, ставши однією з найбільш значущих ігор свого часу.

Основний опір у реалізації гри робився на передачу емоцій та почуттів. Важливу роль у реалізації цього задуму відігравав постановник сцен, що

режисирував сцени в середині гри та заставки, які майже більшу частину ігрового процесу супроводжували гравця. На рисунку 1.3 зображено скріншот з гри Life is Strange:

Рисунок 1.3. Скріншот з гри Life is Strange

1.1.3 Результати аналізу ігрового жанру адвенчура

Під час аналізу було розглянуто три ігри одного ігрового жанру. Незважаючи на це, ігрові продукти дають різні варіанти ігрового процесу, але їх суть завжди зосереджена на завданнях або головоломках, які вирішуються в грі. Однак кожна гра вирішує цю проблему по-своєму. The Secret of Monkey Island робить це за допомогою контекстних меню, Life is Strange створює ситуації, які потрібно швидко вирішити, а Myst фокусується на головоломках. Однак у цих проектів є одна спільна риса – гравець завжди вільний у виборі рішення проблеми з доступних на даний момент варіантів.

Іншим важливим компонентом цих ігор пригодницького типу є дослідження рівня. У будь-якій подібній грі дослідження рівня є основою ігрового процесу, тому що, переміщаючись по ігровому простору, гравець знайде предмети або інформацію, які допоможуть йому в подальшому опрацювати діалог або ігрові ситуації, які можуть з ним трапитися. Знаходження того чи іншого предмета може допомогти гравцеві пройти гру або, навпаки,

затримати його надовго.

Тому, з точки зору моєї теми, необхідно створити таку систему контролю та передачі даних між рівнями, щоби її можна було використовувати для покращення якості ігрового процесу. Відповідь на це питання дасть змогу почати розробку у правильному напрямку. Виходячи з аналізу жанру, можна зробити висновок, що в адвенчурах існує багато подій та завдань. Спочатку потрібно визначитись із тим, що таке рівень. У розглянутих ігор це різні величини. У The Secret of Monkey Island таким можна назвати кожний екран на якому проходить гравець. Для гри Life is Strange – це геймплейні моменти між катсценами. У обох випадках, між ними передавалась та чи інша інформація, про персонажів, умови, завдання, або розв'язані головоломки. Банальним чином, наявність ключу є важливою інформацією для ігрового процесу, адже його зникнення із інвентарю гравця може викликати дуже негативну реакцію зі сторони гравця і це можна буде зрозуміти. Тому потрібно спробувати реалізувати систему, яка буде зберігати такі дані. На рисунку 1.4 показано основні компоненти жанру пригодницької гри.

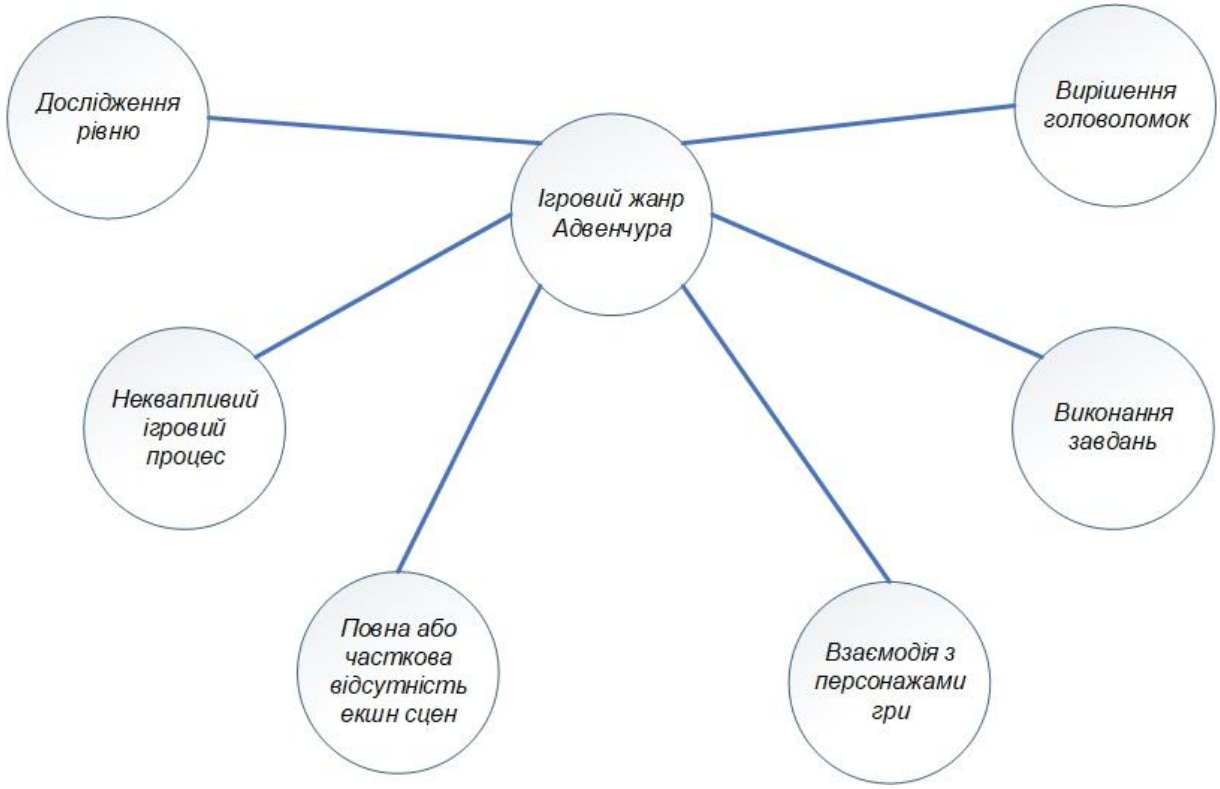


Рисунок 1.4. Складові ігрового жанру Адвенчура

1.2 Аналітичний огляд програмних рушіїв з умовно безкоштовним доступом

1.2.1 Огляд ігрового програмного рушія Unity

Розробка будь-якого ігрового проекту базується на виборі програмного забезпечення для вирішення питань технічної реалізації та підтримки. Вибір движка ігрового програмного забезпечення є дуже важливим кроком на початку розробки. Він визначає використання технології розробки, принципи архітектури гри та методи реалізації механізмів, серед багатьох інших важливих питань. Багато разів движки програмного забезпечення для ігор повідомляють розробникам, як і чи працюватиме їхній проект.

Наприклад, однією з головних причин невдачі Anthem були його технічні аспекти, які були дуже поганими, оскільки це була рольова гра з елементами налаштування та керування, відкритим світом і глибокою багатокористувацькою взаємодією, і вона використовувала свого роду електронні Власний механізм ігрового програмного забезпечення компанії Arts, спочатку створений як механізм ігрового програмного забезпечення в жанрі шутера, без наміру використовувати ресурси, глибоке налаштування та багаторівневу багатокористувацьку взаємодію. В результаті розробникам довелося витратити багато часу і сил на створення окремого інструментарію для розробки рольових ігор і залучення розробників з інших підрозділів Electronic Arts.

Драйвери ігрового програмного забезпечення можна розділити на безкоштовні, умовно-безкоштовні та платні. Останні часто включають ігрові програмні механізми, створені у великих ігрових компаніях для внутрішніх проектів, і в більшості випадків ці компанії не діляться своїми рішеннями з конкурентами, лише в контексті партнерства чи фінансових відносин. Безкоштовні ігрові програмні движки зазвичай поширюються у вигляді бібліотек з мінімалістичними засобами розробки, що створює досить серйозні проблеми для розробки. Ігрові програмні движки, які умовно поширюються безкоштовно, ідеально підходять за ціною і якістю. Ці програмні механізми мають широкі

набори інструментів для розробки і не вимагають оплати, якщо виконуються певні умови.

Тема мого дипломного проекту «Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity». Не зважаючи на те, що в темі прописано обраний програмний рушій, важливим кроком у пояснювальній записці буде проілюструвати, чому мій вибір зупинився саме на ігровому програмному рушії Unity. Для цього я проведу аналіз його конкурентів.

Unity — це популярний кросплатформний ігровий програмний рушій і середовище розробки, створений Unity Technologies. Він надає розробникам інструменти для створення 2D і 3D ігор, додатків віртуальної та доповненої реальності, моделювання, візуалізації тощо. Простота використання та доступність Unity роблять його привабливим вибором як для новачків, так і для досвідчених розробників. Він забезпечує зручний графічний інтерфейс, багато готових ресурсів і компонентів, а також використання потужної мови програмування C# для створення взаємодії та логіки гри. На рисунку 1.5 показано приклад роботи у редакторі ігрового програмного рушія Unity:

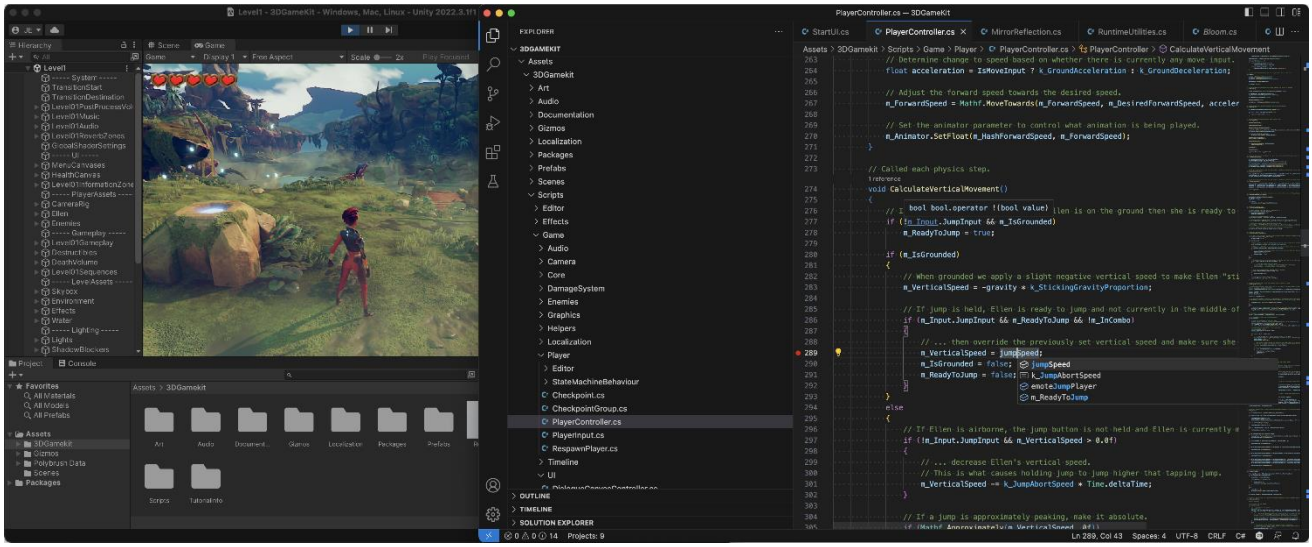


Рисунок 1.5. Приклад роботи у ігровому програмному рушії Unity

Однією з ключових особливостей Unity є його кросплатформенність. Він підтримує кілька платформ, включаючи ПК, консолі, мобільні пристрої, віртуальну реальність і доповнену реальність, що дозволяє розробникам

охоплювати більшу аудиторію та запускати свої продукти на кількох пристроях.

З точки зору ліцензування, механізм ігрового програмного рушія Unity надає повний доступ до своїх функцій безкоштовно для створення ігор будь-якого рівня. Купівля ліцензії потрібна лише тоді, коли сума, зароблена на одного користувача або його власну команду чи компанію, досягає двохсот тисяч доларів.

1.2.2 Огляд ігрового програмного рушія CryEngine

Для того, щоби дати розуміння причини обрання ігрового рушія Unity, розглянемо його двох найближчих конкурентів. Почнемо з ігрового програмного рушія CryEngine який є потужним ігровим програмним рушієм, розробленим німецькою компанією Crytek. Сам ігровий програмний рушій відомий своїми вражаючими фізикою та графікою, а також широкими можливостями для створення відкритих багаторівневих та реалістичних ігрових світів. CryEngine був використаний у різних популярних іграх, таких як серія Crysis, Ryse: Son of Rome та Hunt: Showdown. Початково отримав відомість як ігровий програмний рушій, що відтворює настільки неймовірно реалістичну графіку, що навантажує найсильніші системи. Це стало в ті часи справжньою проблемою для розробників, оскільки використання цього програмного ігрового рушія потребувало сильну систему, так само, як і використання кінцевого продукту. Його правдоподібна симуляція фізики та руйнування довкілля також дуже довго була основною рисою цього ігрового програмного рушія. Через свої неймовірні візуальні та фізичні показники отримав відомість, як бєнчмарк для комп’ютерів того часу, а в деяких проектах сьогодення він таким продовжує залишатись. Одною з останніх крупних ігрових релізів на цьому ігровому програмному рушії став Ryse: Son of Rome, який був запускаючим ексклюзивом для приставки XBOX. На рисунку 1.6 можна побачити приклад графіки, що відтворює програмний рушій CryEngine.

Рисунок 1.6. Приклад графіки, що відтворює програмний рушій CryEngine

З точки зору можливостей, то однією з ключових особливостей CryEngine є його підтримка технології трасування променів (ray tracing), яка дозволяє досягти фотореалістичних ефектів освітлення та відображення поверхонь на різних матеріалах, як метал або водна поверхня. Крім того, CryEngine має високий рівень масштабованості, що дозволяє створювати як невеликі інді-проекти, так і великі AAA-ігри. Програмний рушій також забезпечує розробників інструментами для створення інтерактивного та живого ігрового середовища, включаючи систему штучного інтелекту, керування анімацією персонажів, а також інструменти для створення звукової атмосфери, що дуже часто використовують для створення досить якісних ігор в жанрі хоррор. CryEngine також пропонує гнучку систему багатоплатформної розробки, що дозволяє випускати ігри на різних платформах, включаючи ПК, консолі та мобільні пристрої. На рисунку 1.7 можна побачити приклад роботи з вбудованим аніматором рушія CryEngine.

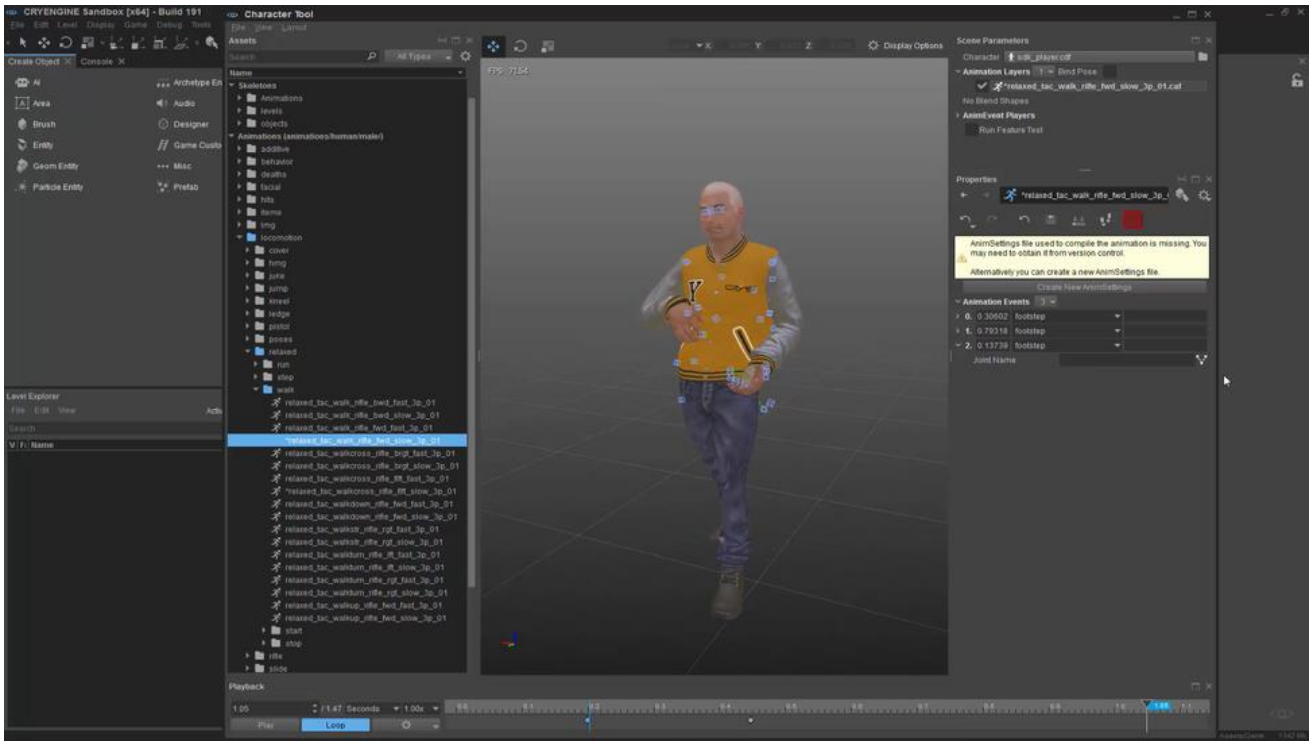


Рисунок 1.7. Приклад роботи з вбудованим аніматором рушія CryEngine

Загалом ігровий програмний рушій CryEngine є безкоштовним, втім система побудована таким чином, що використовувати індивідуальну підтримку, бібліотеку ассетів, можна лише з покупкою ліцензії. Окрім того, її все одно необхідно буде купувати, якщо кількість фінансового обороту користувача програмного рушія перевищить визначеного рівня.

1.2.3 Огляд ігрового програмного рушія Unreal Engine

Другим відомим конкурентом Unity є ігровий програмний рушій Unreal Engine. Це потужний і широко використовуваний ігровий програмний рушій який був розроблений компанією Epic Games та вперше був використаний для створення гри Unreal Turnament звідки й взяв своє ім’я. Він надає розробникам інструменти для створення високоякісних ігор та візуалізацій, а також розробки віртуальної реальності (VR), доповненої реальності (AR) та багатьох інших інтерактивних проектів, різність та глибина яких залежить від вмінь розробника. Ігровий програмний рушій Unreal Engine пропонує широкий набір функцій, включаючи потужний рушій графіки, підтримку власного рушія фізіки, інструменти для роботи зі штучним інтелектом персонажів та середовища, глибоку анімацією персонажів, звуком та багатьом іншим. Він також включає

графічний інтерфейс користувача, який спрощує процес розробки та створення контенту. На рисунку 1.8 можна побачити приклад роботи із анімацією у ігровому рушії Unreal Engine.

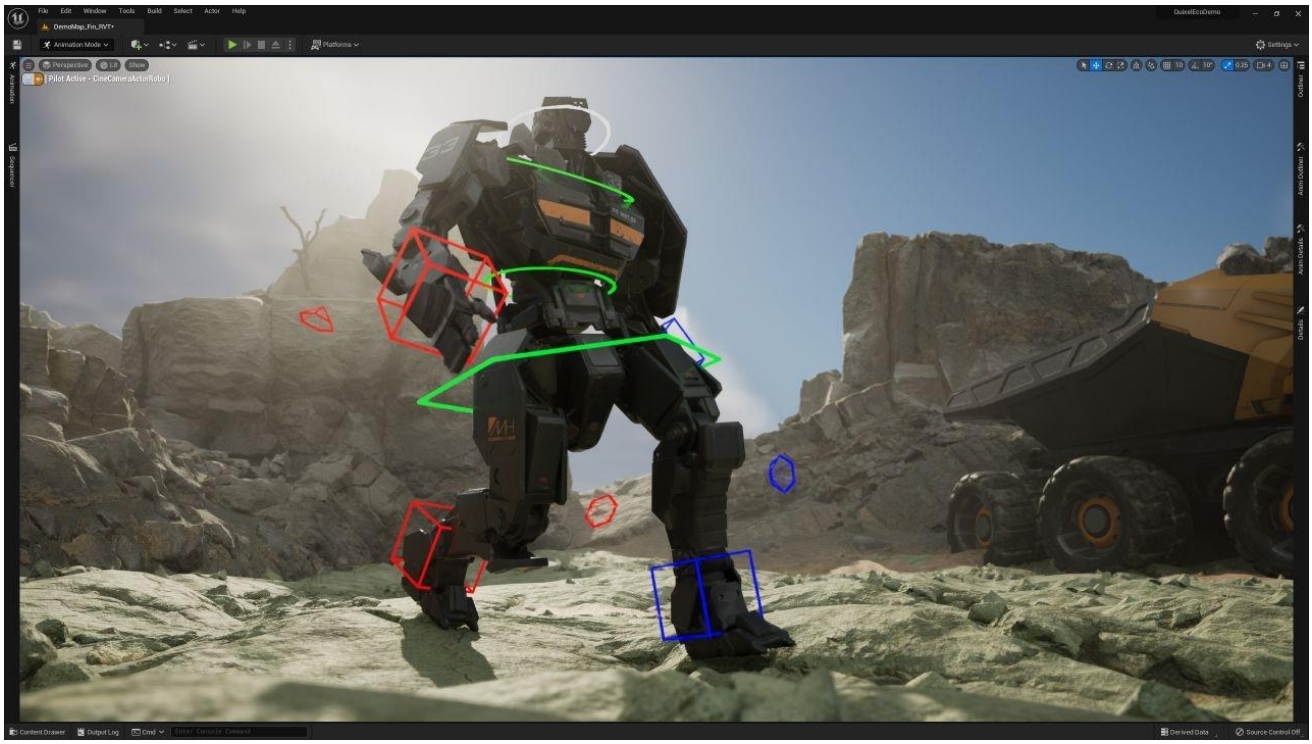


Рисунок 1.8. Приклад роботи із анімацією у ігровому рушії Unreal Engine

Однією з ключових особливостей Unreal Engine є його графічний рушій, який підтримує передові технології, такі як трасування променів (ray tracing), що дозволяє створювати неймовірно реалістичні графічні ефекти відображення та бліків на водній, або металічних поверхнях. Це робить Unreal Engine популярним вибором для розробників ААА-ігор, а також для створення вражаючих візуалізацій та симуляцій в інших галузях. Також, цей програмний рушій використовують для створення фільмів за допомогою спеціального інструментарію, який за допомогою штучного інтелекту будує кадр навколо людини. Ігровий програмний рушій Unreak Engine лежить в основі багатомільярдної гри Fortnite, яка є лідером у жанрі королівської битви з елементами платформеру, крафтингу та пісочниці. На рисунку 1.9 можна побачити приклад графіки яку відтворює Unreal Engine.

З точки зору ліцензування, то ігровий програмний рушій Unreal Engine є безкоштовним, проте не дає доступу до коду самого ігрового програмного рушія,

деякі платформи розробки та можливості заблоковані, а також закритий доступ до виділеного середовища розробників та підтримки.

Рисунок 1.9. Приклад графіки яку відтворює Unreal Engine

1.2.4 Обґрунтування обрання ігрового програмного рушія Unity

Проаналізувавши ігрові програмні рушії було вирішено використовувати рушій Unity. Рішення було зумовлено в першу чергу доцільністю використання можливостей ігрових рушіїв. Його конкуренти надаються передову графіку дуже високого рівня, яка не потрібна у проекті в 2D жанрі. Темою мого дипломного проекту є «Реалізація системи характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity», тому використовувати настільки потужні рушії не має сенсу. Крім того гра в жанрі адвенчура розробляється у виконанні 2D графіки. Також, для Unity існує багато навчального матеріалу та відкритий сайт із документацією, яку можна вивчити у разі виникнення проблем. Також ком'юніті цього ігрового програмного рушія надає допомогу всім бажаючим безкоштовно. Ще одним важливим плюсем у виборі цього ігрового програмного рушія є можливість використовувати AssetStore, в якому вільно розповсюджуються графічні, і не тільки, матеріали для розробників ігор. Через вказані причини, вибір програмного рушія був зроблений на користь Unity.

1.3 Вибір інструментів розробки

Наступним важливим кроком у підготовці до розробки ігрового проекту є вибір інструменту. Розробка програмного забезпечення відіграє дуже важливу роль у процесі роботи, оскільки кожне рішення має свої особливості, а також важливо, щоб у користувача був свій улюблений редактор, в якому він має навички роботи та розуміє всі нюанси. Більшість великих компаній-розробників досить вільно називають інструменти розробки програм, такі як редактори коду. Під час роботи над дипломною роботою мені потрібно було реалізувати деякі графічні частини, тому вибір графічного редактора повинен був бути зроблений дуже відповідно.

Редактор коду – це спеціалізоване програмне забезпечення, яке використовується для створення, зміни та організації вихідного коду програми. Такі редактори зазвичай мають інтуїтивно зрозумілі інтерфейси, які підтримують підсвічування синтаксису, автоматичне доповнення коду, пошук і заміну тексту, а також інтеграцію з іншими інструментами розробки, включаючи системи контролю версій, налагоджувачі та компілятори.

Коли справа доходить до вибору з багатьох редакторів коду, я віддаю перевагу Microsoft Visual Studio. Цей вибір був зроблений тому, що ігровий движок Unity використовує мову програмування C#, яка вимагає, щоб середовище розробки не лише підтримувало форматування коду, але й могло взаємодіяти з іншими класами, методами, об’єктами гри та бібліотеками. Крім того, Microsoft Visual Studio є рекомендованим вибором для розробників і персоналу служби підтримки Unity. На рисунку 1.10 показано вікно розробки проекту Visual Studio в Unity:

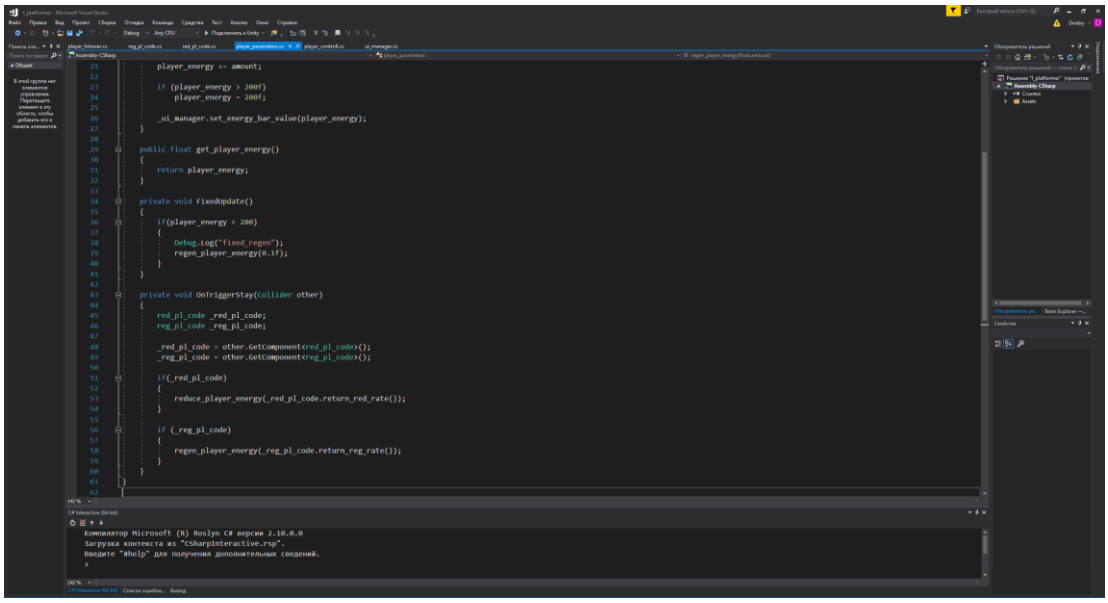


Рисунок 1.10. Вікно розробки Visual Studio проекту у Unity

Для роботи з графікою потрібен графічний редактор. Графічні редактори – це програмне забезпечення, призначене для створення, модифікації та оптимізації графічних зображень. Він надає користувачам інструменти для створення ілюстрацій, редагування фотографій та інших типів графіки, включаючи пензлі, олівці, спреї, штампи, інструменти виділення тощо, а також параметри для налаштування кольору, розміру, прозорості та інших параметрів зображення. До популярних графічних редакторів належать Adobe Photoshop, GIMP, CorelDRAW і Adobe Illustrator.

У контексті розробки цієї гри я вирішив використовувати Adobe Photoshop, оскільки у мене вже є досвід роботи з цим інструментом. На даному етапі розробки гра не вимагає створення складної графіки, а лише потребує розробки кількох спрайтів та редагування зображень, включаючи зміну пропорцій та розмірів. Під час роботи над проектом потрібно буде створити велику кількість кнопок, панелей на зображень для відміток, тому графічний редактор є важливим інструментарієм розробки. На рисунку 1.11 зображено використання Adobe Photoshop для створення спрайту для гри:

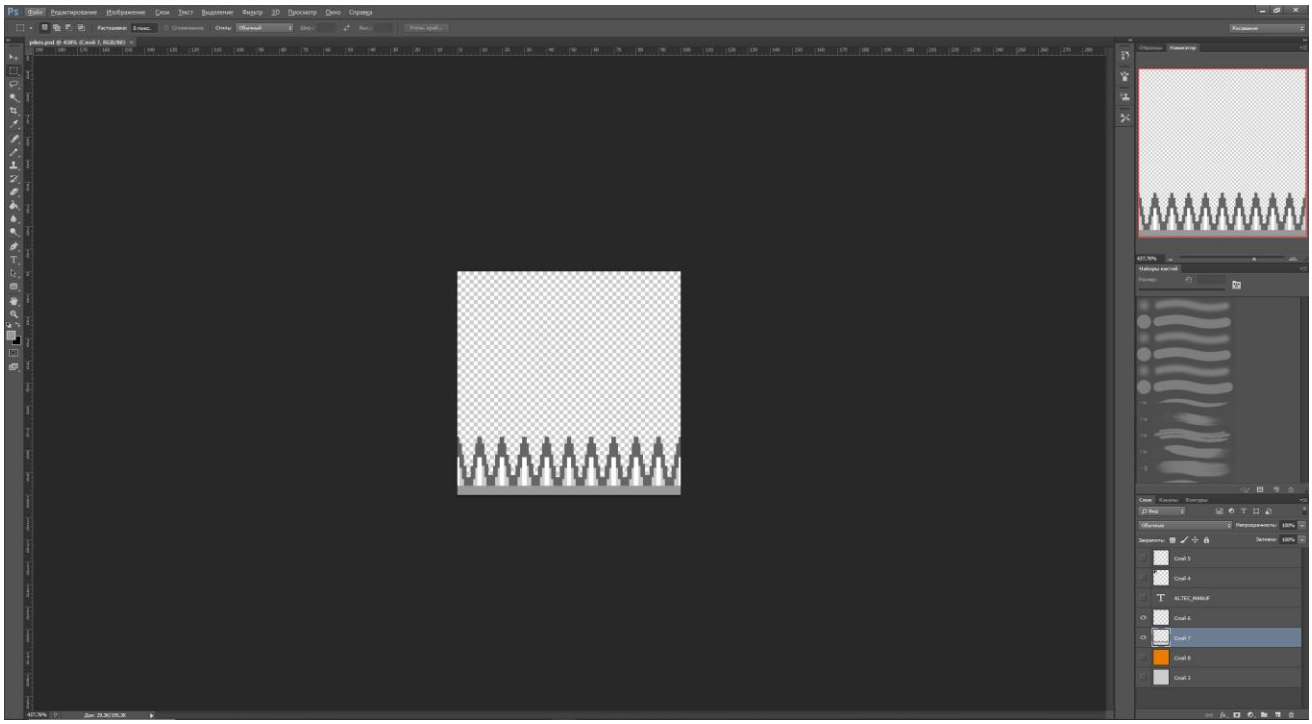


Рисунок 1.11. Використання Adobe Photoshop для створення спрайту для гри

Однією з найважливіших частин розробки є система управління версіями. Це інструмент, який дозволяє розробникам відслідковувати зміни у вихідному коді та інших файлах проекту, порівнювати версії, повертатися до попередніх станів та об'єднувати роботу різних учасників. Вона також зберігає історію змін, що сприяє кращому розумінню процесу розробки та спільної роботи над проектом. До відомих систем управління версіями відносяться Git, Subversion та Mercurial.

Для керування версіями в онлайн-середовищі я використовую Bitbucket та Sourcetree. Bitbucket є веб-платформою контролю версій, яка безкоштовна та інтегрована з різними сервісами в рамках екосистеми Atlassian. Sourcetree, у свою чергу, є клієнтським додатком, який надає графічний інтерфейс для роботи із системами контролю версій, роблячи процес більш зручним та інтуїтивно зрозумілим. Ці інструменти необхідні для збереження результатів розробки онлайн-середовища.

В цілому програмні рішення були обрані із розрахунку на працю із кодом, тому основна робота буде виконуватись у редакторі ігрового програмного рушія Unity. Код для виконання поставленої мети дипломного проекту буде писатись у Microsoft Visual Studio. Не зважаючи на те, що в темі мого проекту не зазначено

створення гри, потрібно буде реалізувати деякі спрайти, для проектування та тестування роботи написаного коду. Графічна частина для проекту буде виконуватись у Adobe Photoshop.

1.4 Формування концепту системи характеристик та вмінь гравця

1.4.1 Загальні відомості про концепт-документ

На виконання задачі поставленої темою дипломного проектування, потрібно виконати основні елементи циклу розробки будь-якої гри, який також складається з формування ігрового концепту. Ігровим концептом можна назвати формування переліку потреб, які сформовані на етапі проробки ідей гри. А взагалі концепт-документ гри – це документ, який містить основні ідеї, концепції, механіки геймплею, сюжетні елементи та інші ключові аспекти, пов’язані з розробкою гри. Цей документ зазвичай є основою для її подальшого розвитку. У концепт-документі гри частіше за все розглядають такі питання:

- Опис гри: Загальний опис ігрового світу, сюжету, атмосфери та основної мети гри;
- Сюжет та персонажі: Опис сюжету гри, персонажів, їх характеристик, мотивацій та відносин між ними;
- Графіка та аудіо: Концепції та ідеї з візуального стилю, арт-дизайну, анімації та звукового оформлення гри;
- Світогляд і фон: Додаткові деталі про світ гри, його історію, культуру, технології та інші аспекти, які можуть впливати на ігровий досвід.
- Технічні вимоги та можливості: Обговорення технічних аспектів розробки гри, включаючи платформи, програмні рушії, інструменти та інше;
- Механіка геймплею: Опис ігрових механік, механіки управління, фізики ігрового світу, системи прогресії, системи бою та інших ігрових аспектів.

Як вже зрозуміло з опису, такий документ допомагає розробникам чітко

визначити напрям роботи та розділити гру на чіткі зони відповідальності між командою, та частини розробки гри, яку можна потім розподіляти між командою у часі та етапах розробки. Завдяки цьому, коли є концепт документ, уникається розповсюджена проблема розробників, коли проект набуває ігрових елементів, або механік, які не передбачені концепт документом та призводять до створення хаосу у розробці.

1.4.2 Розробка концепції системи контролю та передачі даних між рівнями гри

Якщо роздивлятись концепт-документ з точки зору теми дипломного проектування, то у мене немає потреби реалізовувати повний спектр концепції, потрібно лише загострити увагу на системах, які потрібно реалізувати.

Темою мого дипломного проекту є «Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity».

Виходячи з теми проекту, а також особливостей ігрового жанру, потрібно реалізувати механізм отримання, збереження та передачі інформації між рівнями гри. При цьому ігрова система має виконувати контроль тих даних, які передаються між рівнями. Спочатку потрібно розкласти задачу на складові і роздивитись ці питання окремо.

Отже, для отримання більш різноманітного ігрового досвіду, ігровий процес може складатись з багатьох складових. Якщо б була мова про гру жанру гонок, або стратегій, то в основах цих жанрів немає потреби передавати дані між рівнями, оскільки кожний рівень не має зв’язку один з одним. Звісно, це справедливо в тих випадках, коли ігровий жанр гри не було вирішено розширити. У випадку із іграми в жанрі адвенчура, потрібно зберігати різні дані. Наприклад статус завдань, знаходження предметів у інвентарі гравця, можливо відношення персонажів до головного герою. Всі ці дані потрібно передавати справно, без втрати інформації між рівнями, починаючи із самого початку.

Виходячи із написаного вище, є очевидним той факт, що така система має зберігати у собі повний перелік всіх даних, які потрібно зберегти. Тому в першу чергу мені потрібно визначити такі дані для мого проекту. Оскільки система

реалізовується на базі розробляємої гри, буде розумно звернутись до ігрових механік, які реалізовуються в цьому проекту. Одною із важливіших механік гри є характеристики гравця, які відіграють велику роль у якості ігрового процесу, тому потрібно передбачити їх передачу від моменту розподілення цих характеристик до початку рівня і супроводжувати ними гравця весь ігровий проект. Отже, перелік даних для передачі я визначим наступний:

- Параметр сили;
- Параметр спритності;
- Параметр сприйняття;
- Параметр вдачі.

Ці чотири основних параметри гравця мають вплив на самого персонажа та на те, як виглядає сцена, тому можна буде відразу проводити тестування того, на скільки система реалізована вірним чином.

Наступним кроком є визначення з тим, яким чином виконувати перевірку коректності передачі інформації. Найпростішим для цього способом буде створення коду, який буде знімати для себе копію ігрових даних, які отримує гравець. Виходячи з того, що він буде лише приймати інформацію та виконувати перевірку, можна не боятись випадкової зміни значень.

Сама передача буде виконуватись між сценами ігрового проекту. Серед сцен будуть представлені сцени рівнів, а також меню розподілу характеристик гравця. Оскільки це меню окрема сцена, потрібно буде зберігати та передавати інформацію від неї до інших сцен, які будуть завантажуватись після її використання.

1.5 Проектування системи контролю та передачі даних між рівнями

1.5.1 Загальне проектування

Перш ніж розпочати впровадження розроблених систем, відповідно до теми дипломного проекту, необхідно здійснити їх попереднє проектування. Це критично важлива частина процесу розробки, оскільки вона закладає фундамент

впровадження та допомагає уникнути численних проблем на етапі реалізації. Професійні розробницькі команди проводять проектування будь-яких функціональних елементів програмного забезпечення перед впровадженням не лише для того, щоб мати уявлення про процес створення модулів програми, а й для орієнтації співробітників команди та підтримки коду у разі виникнення проблем з ним.

Для реалізації системи потрібно виконати загальне проектування – визначитись із тим, яка буде програмна будова проекту. Це буде зручною основою для виконання проектування по принципу «Від великого до маленького». На рисунку 1.12 зображено загальну схему ігрового проекту:

Рисунок 1.12. Загальна схема ігрового проекту

Можна побачити, що в поточному проекті доволі багато різних модулів, кожний з котрих зберігає який-небудь код для виконання тих, чи інших завдань під час ігрового процесу, або роботі у меню гри.

Не дивлячись на те, що ці модулі ніби існують окремо один від одного, але це не так. Більшість модулів гри буде взаємодіяти між собою. Так код гравця безпосередньо буде взаємодіяти із розробляємою системою. Код інтерфейсу приймає участь у розподілені характеристик гравця, тобто теж є частиною роботи системи контролю та передачі даних між рівнями гри.

1.5.2 Розробка системи контролю та передачі даних між рівнями

Але в розрізі теми дипломного проектування я акцентую увагу на модулях «Коду для контролю та передачі даних», «Код скриптів рівнів» та «Код для

гравця». Саме ці три модуля є основою для розробляємої системи. Не дивлячись на те, що код гравця не вказаний у темі дипломного проектування, але він відіграє важливу роль у роботі системи. Тому для більшого розуміння того, як і що я буду передавати та контролювати при передачі, потрібно спроектувати модуль гравця, його схема зображена на рисунку 1.13:

Рисунок 1.13. Модуль гравця

Стисло опишу цей модуль. Гра в жанрі адвенчура має на увазі переміщення та взаємодію із оточенням. Для цього персонаж має вміти ходити та взаємодіяти із оточенням, тому в модуль гравця входять відповідний код. Оскільки для реалізації розробляємої системи пов’язана із характеристиками гравця, слід окремо зупинитись на коді параметрів гравця, його схема зображена на рисунку 1.14:

Рисунок 1.14. Схема коду параметрів гравця

Тут можна приділити увагу таким блокам як отримання характеристик, самознищення та методи для передачі даних. Характеристики гравець буде отримувати саме від системи передачі даних, роботу якої буде спроектовано нижче. Самознищення потрібно для геймдизайнерських цілей, як з боку гравця, так і з боку розробника. Річ у тім, що характеристики гравця відіграють важливу роль у ігровому процесі. Тому в разі невдалого розподілення очок характеристик і неможливості подальшого проходження рівня, можна самознищитись та повернутись до меню розподілу характеристик. Методи для передачі даних виконують чисто функційну роль для передачі характеристик гравця до зовнішніх скриптів гри, щоби не відкривати доступ до цих показників у коді гравця.

Тепер потрібно перейди до проектування безпосередньо системи передачі

даних. Пед тим як описати спроектований модуль, потрібно розкрити сутність сцен у ігровому програмному рушії Unity. Річ у тім, що сцени є файлами, які зберігають в собі дані про: де, як, із якими компонентами, в яких конфігураціях, знаходяться ігрові об’єкти на цій сцені. Проте кожний раз, при завантажені сцени всі ігрові об’єкти створюються знову. Тому, наприклад якщо розглядати поточний ігровий проект, при поверненні до меню розподілу характеристик, у ігровому об’єкт гравця всі дані будуть знищені і к моменту повернення до рівня гравця буде зустрічати нова копія ігрового об’єкту гравця із початковими налаштуваннями. Це є одною із причин, чому потрібно реалізовувати систему передачі даних між рівнями (сценами) гри.

Втім, виходячи з інформації, написаної вище, можна прийти до запитання, яким чином, це можна реалізувати. На рисунку 1.15 зображено схему того, як буде працювати система передачі даних між рівнями.

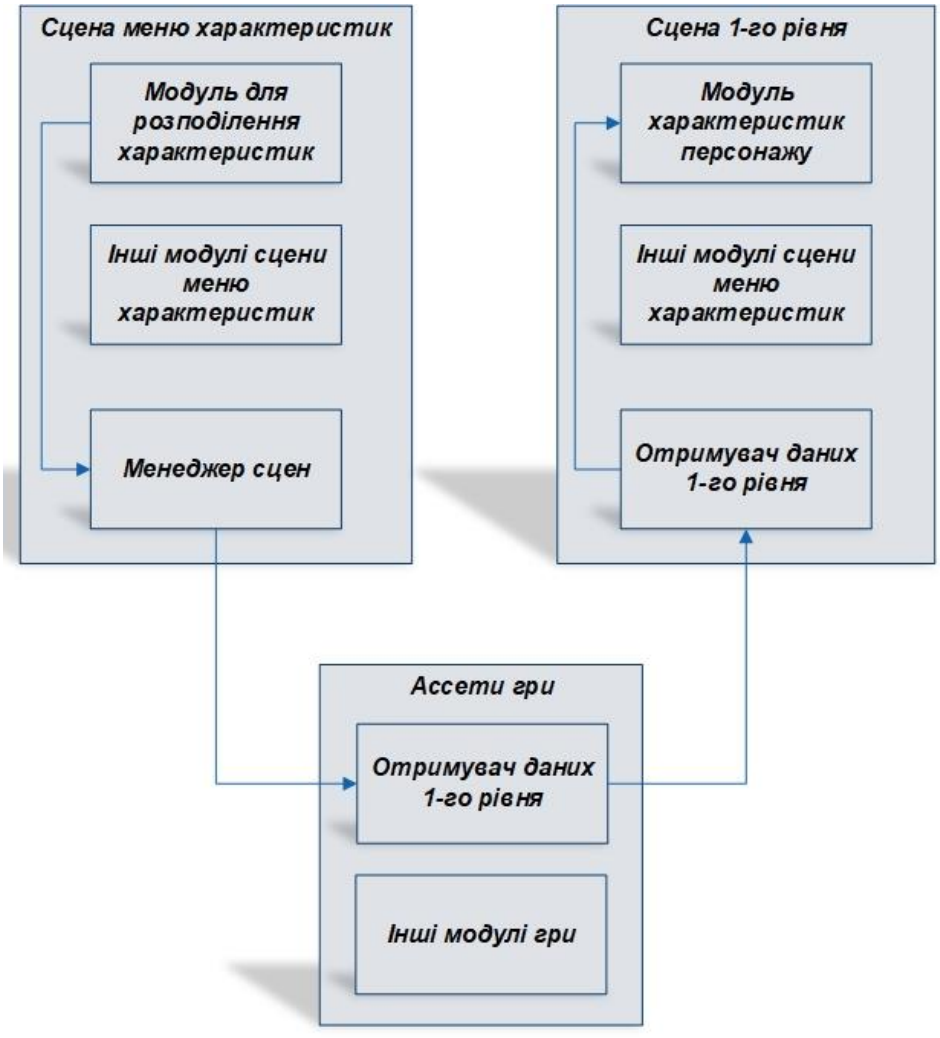


Рисунок 1.15. Схема роботи системи передачі даних між рівнями

На схемі можна побачити, що Менеджер сцен знаходиться у сцені розподілення характеристик. Інформація про розподілення очок характеристик персонажа гравця передаються до цього менеджеру всередині сцени під час ігрового процесу. В свою чергу Менеджер сцен має в собі перелік рівнів гри для того, щоби мати можливість викликати їх за потребою. Тепер до проблеми передачі даних між рівнями.

Проблема передачі даних вирішається за допомогою мови програмування C# та ООП. Як відомо об’єкти в ООП можуть мати різні рівні доступу та модифікатори. Коли створюється екземпляр класу, то він є точною копією оригіналу, але його дані належать лише йому. Проте деякі модифікатори створюють змінні – властивості – які є сталими для класу та його нащадків під час всього циклу життя програми. Річ у тому, що ігровий програмний рушій Unity має особливий підхід до роботи із скриптами, які створюються в проекті. Кожний скрипт буде працювати лише тоді, коли він доданий до ігрового об’єкту. І в момент приєднання, приєднується не скрипт, який написав розробник, а його нащадок. Тому у скрипті можна створити змінні із модифікатором доступу, який створить змінні, які будуть існувати на протязі всього циклу життя програми.

На схемі 1.15 можна побачити, що Отримувач даних 1-го рівня існує у двох екземплярах. Оригіналом у цьому випадку є той, що лежить у Ассетах гри, а той, що знаходиться на Сцені 1-го рівня є його екземпляром. Саме Отримувачі даних будуть прошарком для передачі інформації між рівнями, тому в них створюються змінні із модифікатором доступу static, який дозволяє передавати дані зі сцени до оригіналу, або батьківського класу, Отримувача даних рівнів, а потім звертатись до цих даних із екземпляру цього скрипту у сценах гри.

Тепер, коли я можу передавати дані між сценами, то можна імплементувати в існуючу схему Контролера передачі даних. Як вже було зазначено в частині розробки концепту, цей код повинен контролювати якість переданих даних, та дублювати передачу, у випадку, якщо дані були пошкоджені, або передались до сцени не в тому вигляді або об’ємі, як це було потрібно. На рисунку 1.16 зображено схему роботи контролера системи

контролю та передачі даних між рівнями гри:

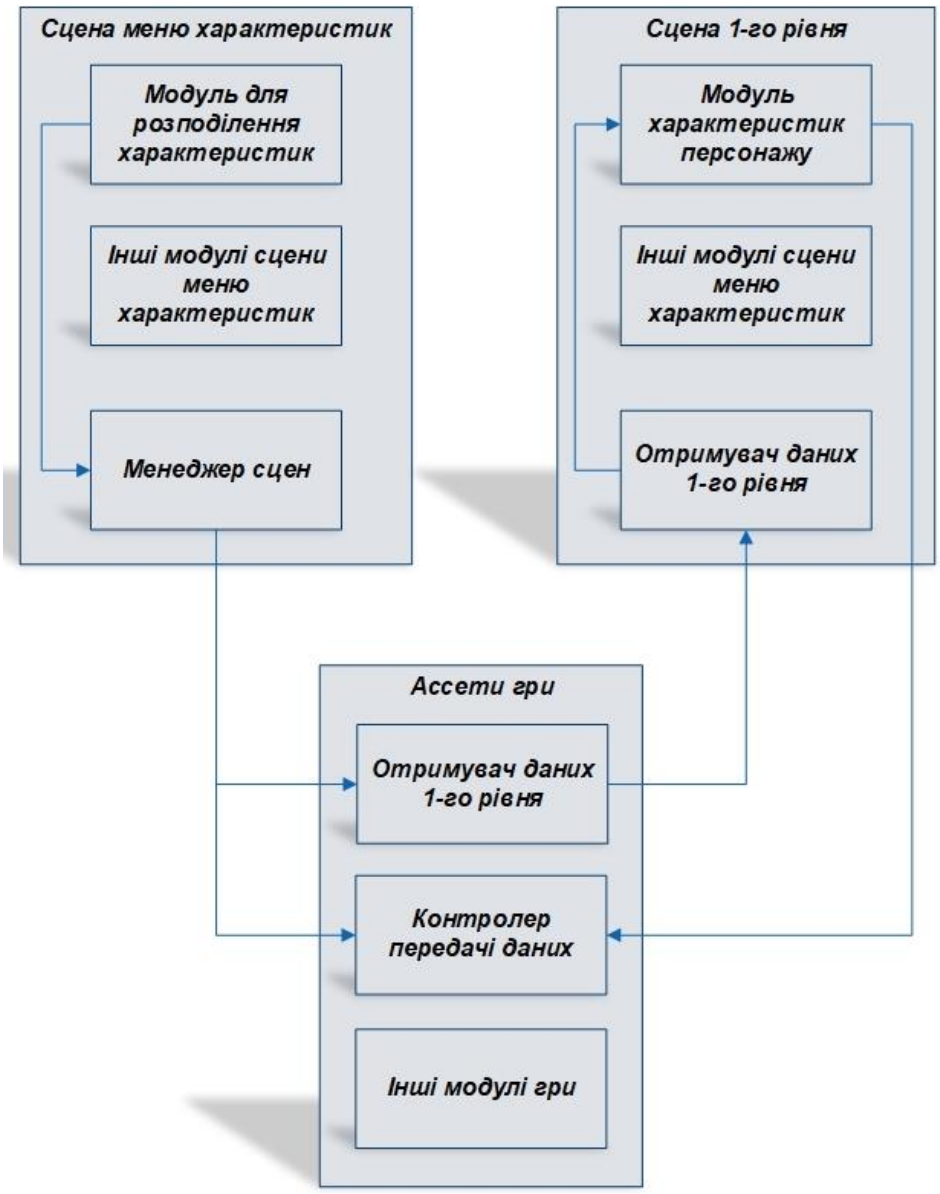


Рисунок 1.16. Схема роботи контролера системи контролю та передачі даних між рівнями

Зі схеми видно, що Контролер передачі даних отримає дані із Менеджеру сцени в той же час, коли ці дані отримує Отримувач даних 1-го рівня. Втім, сам контролер не виконує ніяких дій. З однієї сторони до нього звертається Менеджер сцен, а з іншої сторони Модуль характеристик персонажу для перевірки коректності отриманих даних. У разі, якщо отримані дані були помилковими, або неповними, Модуль характеристик персонажу спробує отримати дані від Контролера передачі даних. Таким чином буде передбачено дублювання, що позитивно відобразиться на якості передачі даних.

Отже, виходячи з результатів проектування, можна зазначити, що частиною системи контроля та передачі даних є так званий Менеджер рівней, а також Отримувач даних. Частиною з контроля виступає модуль Контролер передачі даних. Для ілюстрації передачі даних та тестування реалізації намічених систем також потрібно створити код характеристик гравця.

1.6 Реалізація елементів системи контролю та передачі даних між рівнями

Для успішного виконання теми мого дипломного проекту, потрібно створити гру в якій будуть реалізовуватись всі намічені темою системи. Це буде відбуватись на обраному ігровому програмному рушії. Для створення проекту використовується відповідний редактор та програма UnityHub, яка поєднує в собі не тільки інструменти створення ігрових проектів, але й концентрує у зручному одному місці їх перелік. На рисунку 1.17 можна побачити вікно створення проекту у UnityHub. Як можна побачити тут є деякі шаблони для створення проектів, які завантажують початкові ігрові об’єкти та налаштування сцени редактору.

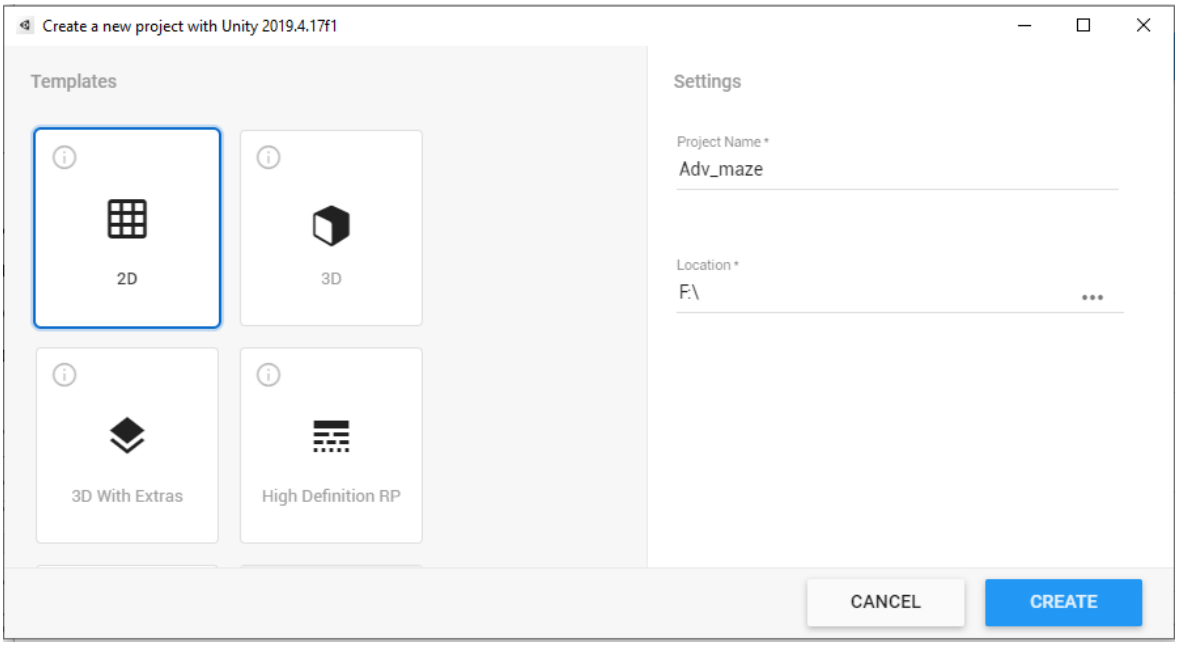


Рисунок 1.17. Вікно створення проекту у UnityHub

Після натискання на кнопку New проект буде створено автоматично, разом з усіма необхідними файлами, директоріями та папками. 2D-проект використано

не тільки через тему дипломного проекту, але й тому, що такі проекти потребують менше ресурсів для розробки, а також більш прості у розумінні для гравця.

Отже виконавши створення нового проекту буде автоматично згенерована пуста сцена проекту, із стандартним набором об’єктів – камери гравця та джерела світла, що можна побачити на рисунку 1.18. Ці об’єкти вже дають можливість запустити проект, але гра покаже лише пустий простій, адже «дивитись» на гру ми будемо через камеру, а джерело світла не має «тіла».

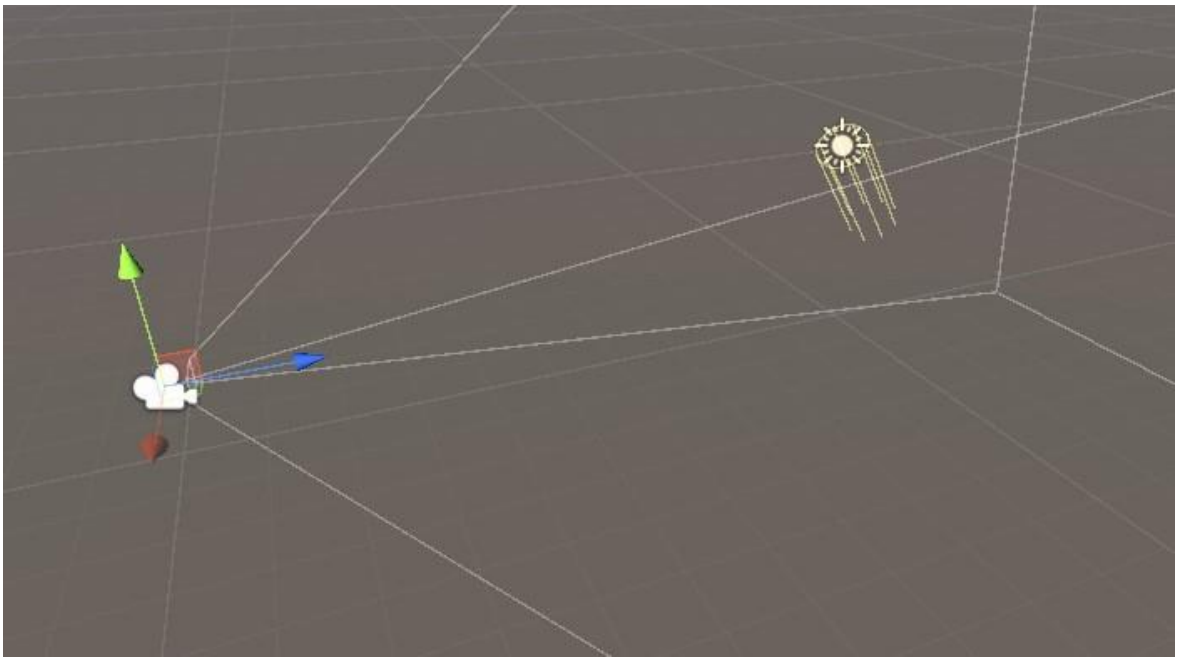


Рисунок 1.18. Стартові об’єкти в новому проекті Unity

Наступним кроком, є розміщення ігрових об’єктів. Для початку потрібно реалізувати задній фон гри. Оскільки для виконання даного проекту реалізація повноцінної гри не передбачено, на деякий час я зміню параметр фону камери, та виставлю колір, який буде слугувати заміною для неба. У майбутньому можна буде програмно змінювати параметри камери, а також, із появою повноцінного спрайту заднього фону, використати його, як скайбокс. Скайбокс – це зображення яке використовують для симуляції ігрового неба. На рисунку 1.19 можна побачити ігрову сцену із зміненими характеристиками фону камери гри:

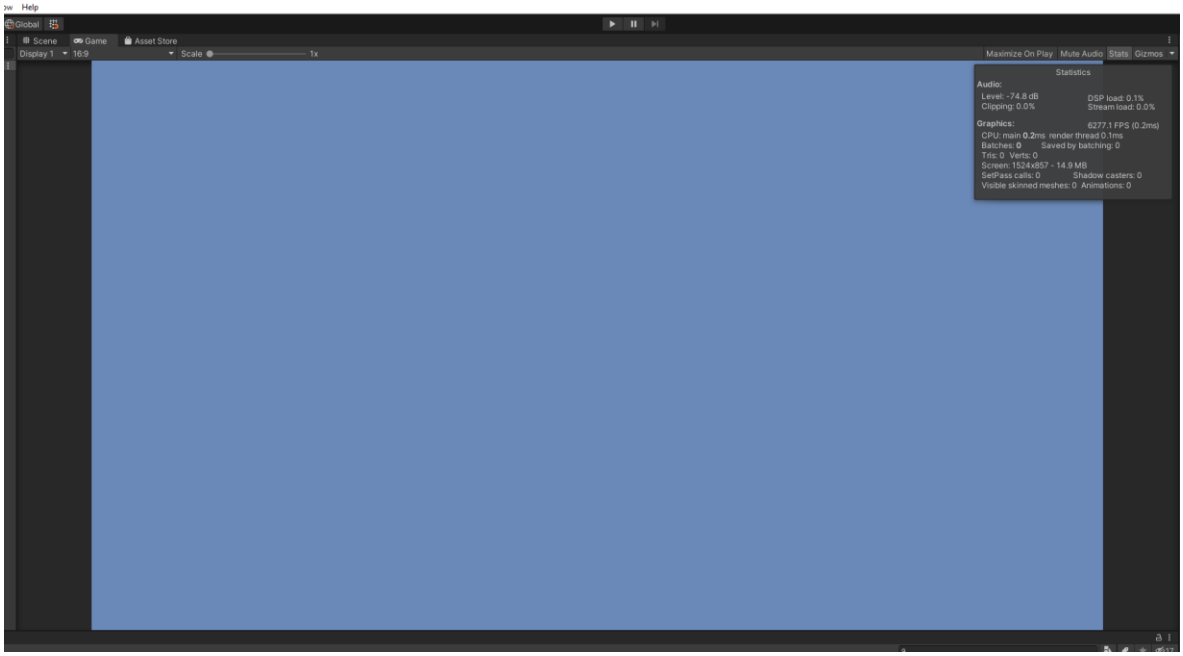


Рисунок 1.19. Ігрова сцена із зміненими характеристиками фону камери гри

Окремо слід зазначити про реалізацію елементів графіки, які пов’язані із побудовою рівнів. Оскільки гра реалізується виключно у 2D-просторі, а також гра створюється в жанрі адвенчура, потрібно створити комплект графічних елементів, з яких можна було б скласти рівень, ніби з конструктору. Для цього потрібно реалізувати комплекс спрайтів для землі, оточення та іншого. Маючи такий набор можна створювати рівні різної комплекції, від прямих до діагональних. Такий простий набор дає змогу швидко створювати локації, групуючи отримані об’єкти в один для того, щоби вони не заважали.

Окрім спрайтів рівня, як частин стін та підлоги, потрібно створити спрайти, які б відповідали за оточення гравця, а також події та інші елементи ігрового проекту. Реалізація графічної частини проекту завжди є важливої частини розробки, оскільки створює необхідну атмосферу у грі, навіть якщо створюється тестовий рівень. На рисунку 1.20 можна побачити графічні елементи для створення рівнів:

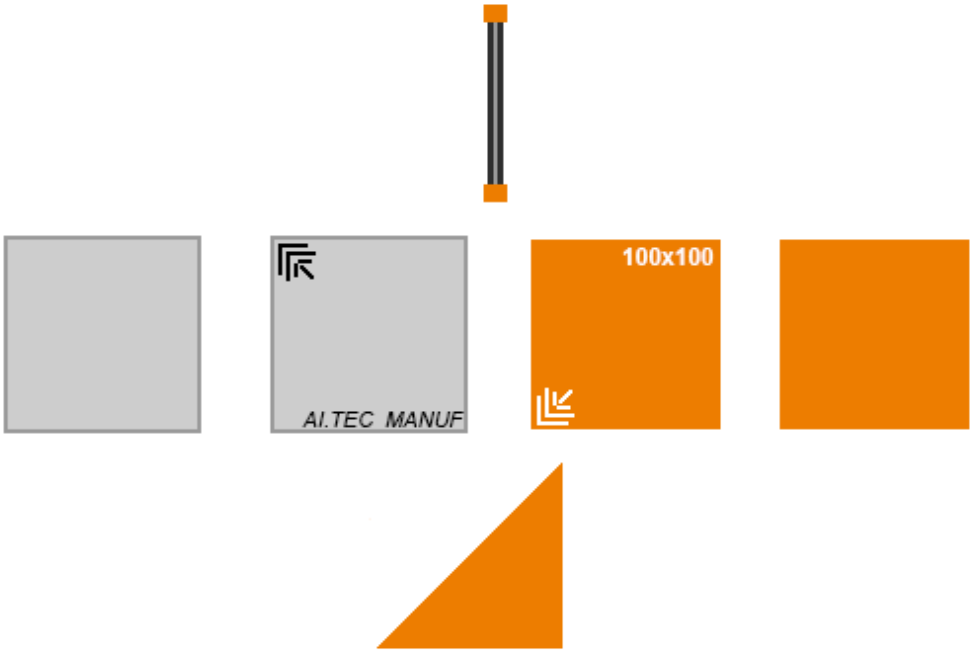


Рисунок 1.20. Графічні елементи для створення рівнів гри

Таким же чином у Adobe Photoshop створювались спрайти для гравця, частин рівню та спрайтів подій.

Деякі елементи ігрової графіки не було створено за допомогою програмного забезпечення, оскільки ігровий програмний рушій Unity дає змогу створювати примітивні графічні елементи самостійно із використанням інструментів за замовченням. Слід зазначити, що деякі спрайти фону створені таким чином, щоби їх можна було розміщувати поверх інших. Це зроблено для того, щоби декорувати ігрові рівні. Такі декорації допомагають створити атмосферу та антураж для ігрового процесу. Дуже часто таким чином створюють основу для освітлення, або інших об’єктів, які мають бути використані, але візуально не повинні змінюватись. Прикладом таких спрайтів можуть стати картини, дзеркала, вікна, шафи та інше. Всі реалізовані спрайти можна переглянути на рисунку 1.21:

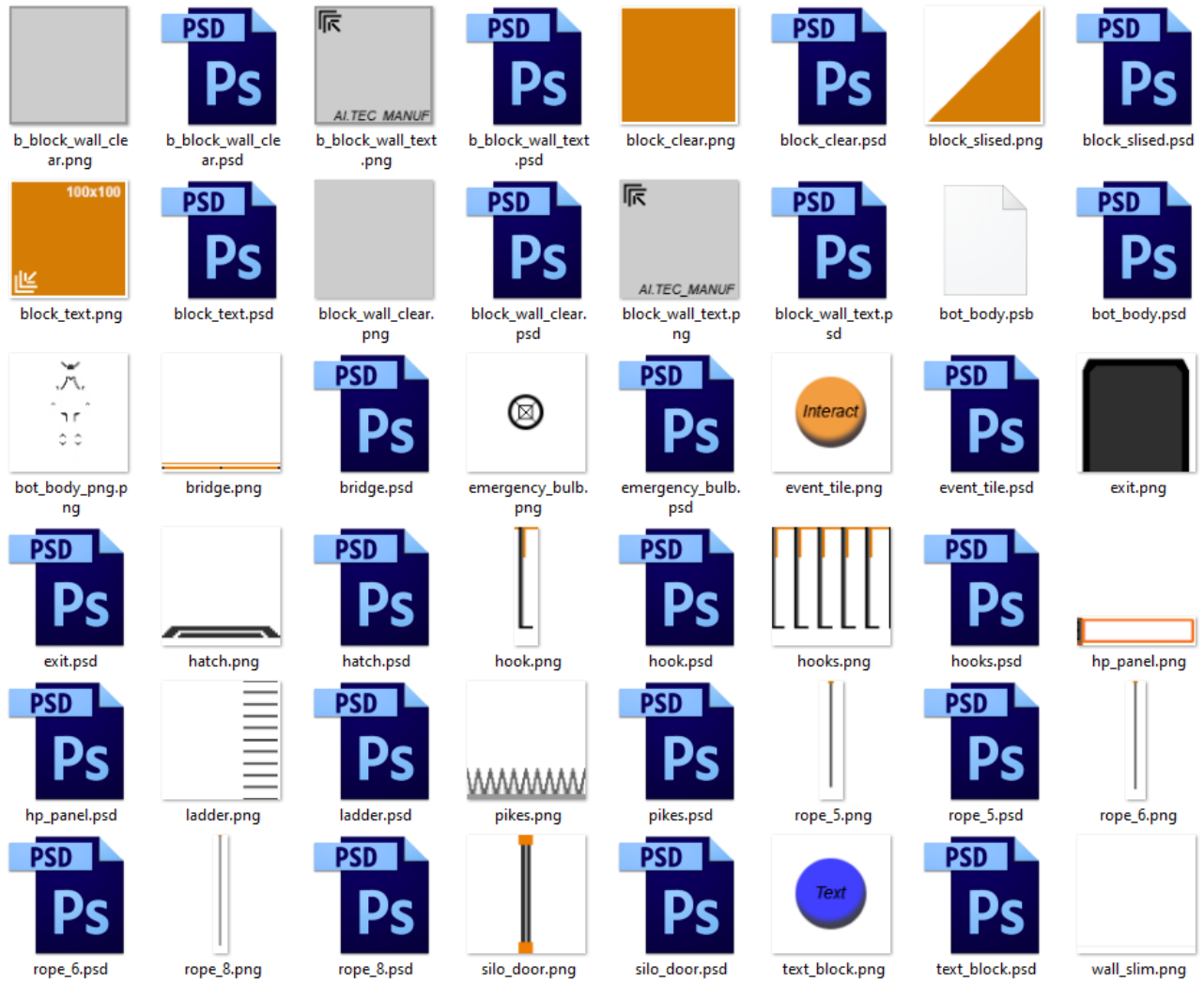


Рисунок 1.21. Всі спрайти реалізовані для проекту гри

Наступним кроком, для того щоби приступити до реалізації теми дипломного проекту, потрібно зібрати із створених елементів рівень гри, розмістивши на сцені землю, оточення та елементи для взаємодії.

Це необхідно в першу чергу через те, що ігрові скрипти, які створюються на програмному рушії Unity, працюють лише у зв'язку із ігровими об'єктами на сцені. Якщо уважно придивитись до імені початкового файлу класу скриптів, то можна побачити там MonoBehaviour, що можна перевести як самоповедінка. Тобто мається на увазі, що кожний такий скріп буде відповідати за поведінку об'єкта на сцені всеціло, або лише частково. Через це потрібно створити рівень для гравця.

Окрім того, для тестування працездатності реалізовуємої системи, потрібно буде розмістити події, оточення, предмети та інші речі, які б звертались

до тих параметрів, які буде передавати система контролю та передачі даних між рівнями гри. На рисунку 1.22 зображено частину тестового 1-го рівня гри:

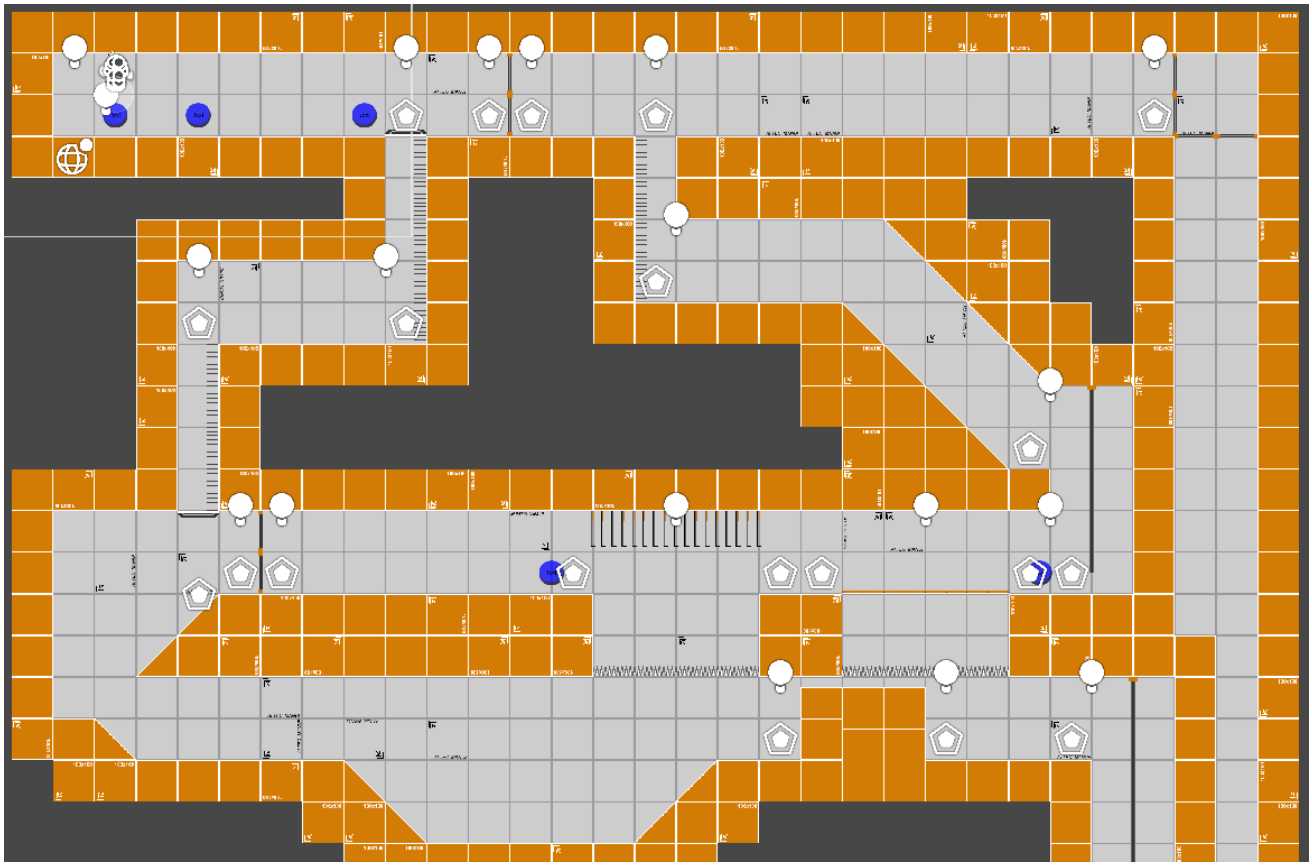


Рисунок 1.22. Частина тестового 1-го рівня гри

Коли розміщені всі ігрові об’єкти, елементи рівня, можна переходити безпосередньо до реалізації систем теми дипломного проектування, а саме системи контролю та передачі між рівнями гри. Отже комплекс модулів визначено у частині проектування. Необхідно створити скрипт Характеристик гравця player_parameters.cs, Менеджер рівнів scenes_loader.cs, Отримувач даних - 1го рівня level_1_receiver.cs та data_checker.cs як частину контролю розробляємої системи..

Розпочну реалізацію із коду характеристик гравця player_parameters.cs. Під час реалізації потрібно урахувати, що цей код повинен мати доступ до предметів гравця, його візуальних ефектів. Також, потрібно створити зміні для характеристик гравця. Ще одним функціоналом передбачений концепцією є самознищення. Нижче приведено частину коду скрипту із об’явою змінних:

```
scenes_loader _scenes_loader;
[SerializeField] health_bar_logic _health_bar_logic;
```

```
[SerializeField] Light2D _player_light_sphere;
[SerializeField] Light2D _player_flashlight;
[SerializeField] int strength = 0;
[SerializeField] int dexterity = 0;
[SerializeField] int perception = 0;
[SerializeField] int luck = 0;
[SerializeField] int health = 100;
```

З коду видно, що в першу чергу створюється код який буде звертатись до Отримувача даних 1-го рівня. В даному випадку він має загальне ім'я, але в кожній сцені цей скрипт буде своєю копією. Далі створюються змінні для предметів. Окрему увагу потрібно приділити самим характеристикам гравця, які не є публічними із розуміння безпеки. Після ініціалізації цих змінних має сенс відразу розглянути код самознищення. Нижче приведено фрагмент цього коду:

```
private void Update()
{
    if (Input.GetKeyUp(KeyCode.C))
    {
        take_damage(health);
    }
}

public void take_damage(int damage_value)
{
    health -= damage_value;
    if (health < 0)
        health = 0;
    if (health == 0)
    {
        _scenes_loader.load_main_menu();
    }
    _health_bar_logic.set_hp_value(health);
}
```

В цьому коді можна побачити, що у методі Update(), який виконується кожний кадр, що відрисовує комп'ютер, перевіряється чи була натиснута кнопка C, яка відповідає за самознищення. У цьому випадку викликається метод take_damage() та як аргумент передається поточне значення очок здоров'я гравця, що призводить до смерті персонажу.

Останнім кодом, який буде реалізований на даному етапі, є код повернення значення характеристики. Це потрібно у разі проходження події. Код приведено

нижче:

```
public int take_strength_value()
{
   return strength;
}
```

Тепер потрібно реалізувати код, який буде отримувати характеристики із Отримувача та зберігати їх у файлі гравця, нижче приведено фрагмент цього коду:

```
public void take_player_parameters(int strength_value, int dexterity_value, int perception_value, int luck_value)
{
   strength = strength_value;
   dexterity = dexterity_value;
   perception = perception_value;
   luck = luck_value;
   setup_player_modifier();
}

void setup_player_modifier()
{
   _health_bar_logic.set_hp_max_value(60+(strength * 10));
   _player_light_sphere.pointLightOuterRadius = perception * 0.5f;
   _player_flashlight.pointLightOuterRadius = perception * 1.25f;
}
```

Метод take_player_parameters() викликається при завантажені 1-го тестового рівня. Виклик буде виконуватись із коду Отримувача даних 1-го рівня та інформація буде передаватись як аргументи методу. В тілі цього методу, аргументи передаються до значень характеристик гравця безпосередньо у частині ігрового процесу. Метод setup_player_modifier() виконує налаштування характеристик гравця, його предметів у залежності від отриманих характеристик. Тепер я реалізую код Отримувача даних 1-го рівня, level_1_receiver.cs. Нижче буде приведено його код:

```
[SerializeField] player_parameters _player_parameters;
public static int strength_parameter;
public static int dexterity_parameter;
public static int perception_parameter;
public static int luck_parameter;
```

```
private void Start()
{
   send_player_parameters();
}

public void send_player_parameters()
{
   _player_parameters.take_player_parameters(strength_parameter,
dexterity_parameter, perception_parameter, luck_parameter);
}
```

З приведеного фрагменту видно, що спочатку цей скрипт отримує посилання на код характеристик гравця player_parameters.cs для того, щоби передати в нього актуальні дані отримані від Менеджеру рівнів. Наступними змінними є статичні цілочисельні значення характеристик гравця. Далі можна побачити, що відразу з моменту створення цього скрипта, буде звернення до методу send_player_parameters(). Цей метод звертається до скрипту характеристик гравця через посилання, яке було отримано раніше, та викликає його метод take_player_parameters() в параметри якого передаються послідовно значення статичних змінних.

Тепер переходжу до реалізації коду Менеджеру рівнів, який буде мати назву scenes_loader.cs. Цей скрипт буде звертатись до рівнів та виконувати їх завантаження за вимогою ігрового процесу. У поточній реалізації даний код буде звертатись лише до 1-го тестового рівня. Розташовуватись цей код буде у сцені розподілу характеристик. Нижче приведено перший фрагмент коду менеджеру:

```
[SerializeField] parameters_logic strength_parametr;
[SerializeField] parameters_logic dexterity_parametr;
[SerializeField] parameters_logic perception_parametr;
[SerializeField] parameters_logic luck_parametr;
```

Як можна побачити, спочатку я створим поля для отримання характеристик гравця із меню розподілення характеристик. Як тип вказано не звичайний тип даних, а код, що відповідає за роботу із характеристиками. Нижче приведено фрагмент коду, який завантажує сцену меню розподілу характеристик гравця:

```
public void load_main_menu()
{
```

```
        EditorSceneManager.LoadScene("Main_menu");
    }
```

З коду можна побачити спосіб виклику нових сцен у ігровому програмному рушії Unity. Цей процес виконується за допомогою команди EditorSceneManager та його методу LoadScene(). Цей метод отримує як аргумент строку із назвою сцени, яку необхідно завантажити. Нижче приведено ще один фрагмент коду:

```
    public void load_level_1()
    {
        level_1_receiver.strength_parameter = strength_parametr.value;
        level_1_receiver.dexterity_parameter = dexterity_parametr.value;
        level_1_receiver.perception_parameter = perception_parametr.value;
        level_1_receiver.luck_parameter = luck_parametr.value;
        data_checker.strength_parameter = strength_parametr.value;
        data_checker.dexterity_parameter = dexterity_parametr.value;
        data_checker.perception_parameter = perception_parametr.value;
        data_checker.luck_parameter = luck_parametr.value;
        EditorSceneManager.LoadScene("Level_1");
    }

    public void exit_game()
    {
        Application.Quit();
    }
}
```

З даного фрагменту можна побачити метод load_level_1(), який повинен, при звернені, викликати перший тестовий рівень. Хід виконання цього метода починається із передачі характеристик до Отримувача даних 1-го рівня. Поступово, характеристика за характеристикою будуть завантажені у відповідні змінні. Оскільки ці змінні статичні, то до них можна звернутись не створюючи посилання на код Отримувача даних 1-го рівня. Тут же виконується відправка даних до скрипта контролера data_checker.cs. Після того, як всі характеристики були передані, виконується завантаження сцени першого тестового рівня. Останній метод exit_game() при зверненні відповідає за завершення роботи гри.

Останнім елементом для реалізації є data_checker.cs – частина системи з контролю передачі даних. Він буде розташований так само у папці з ассетами

гри, як і код скрипта level_1_receiver.cs. Нижче приведено його код:

```
public static int strength_parameter;
public static int dexterity_parameter;
public static int perception_parameter;
public static int luck_parameter;

public bool check_player_parameters(int str, int dex, int per, int luck)
{
   if ((str == strength_parameter) && (dex == dexterity_parameter) && (per == perception_parameter) && (luck == luck_parameter))
   {
      return true;
   }
   else
      return false;
}

public void take_player_parameters(out int str, out int dex, out int per, out int luck)
{
   str = strength_parameter;
   dex = dexterity_parameter;
   per = perception_parameter;
   luck = luck_parameter;
}
```

Як можна побачити з коду вище, в першу чергу контролер створює публічні статичні змінні для характеристик гравця, для того, щоби можна було їх отримати із менеджеру рівнів.

Також, цей скрипт має два методи check_player_parameters() та take_player_parameters(). Перший викликається із скрипту характеристик гравця та має отримати як аргументи значення характеристик, які отримав код гравця. Після цього виконується перевірка, якщо всі характеристики коду характеристик збігаються із тими, що є у контролера, то метод повертає істину, в противному випадку не правду. Другий метод за вимогою повертає значення характеристик які має в своєму розпорядженні.

Тепер гравець має змогу використовувати систему контролю та передачі даних між рівнями гри. Важливо розуміти, що поточна реалізація гри не є кінцевою, але реалізація системи контролю та передачі даних реалізована таким

чином, що її архітектура дозволяє у майбутньому реалізовувати нові і нові рівні, додавати нові предмети для збереження у системі передачі даних, та створювати більше умов для ігрового процесу, що додатково відкриває нові можливості у ігровому дизайні поточної гри.

Всі файли проекту було розподіллено по підкаталогам розділу проекту Assets, розміщення котрих можна побачити на рисунку 1.23.

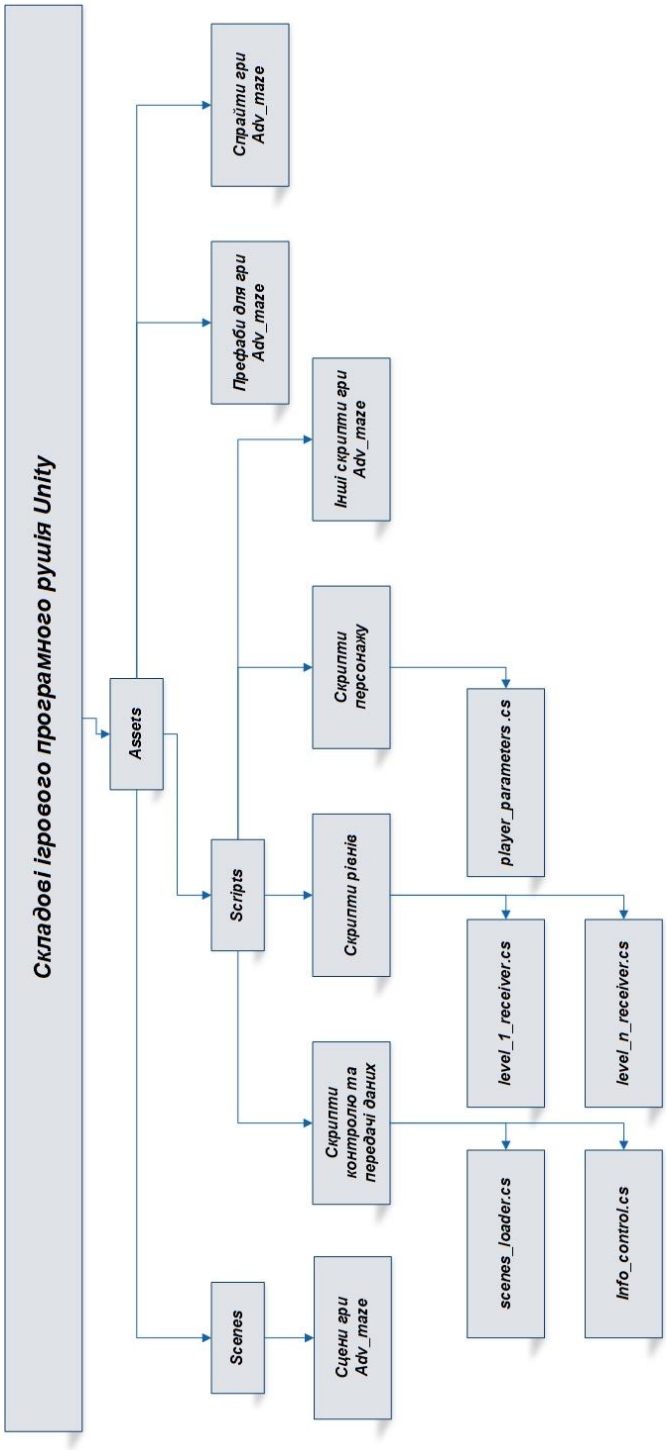


Рисунок 1.23. Розміщення файлів проекту по підкаталогам розділу Assets

1.7 Тестування працездатності реалізованих елементів

Перевірка та налагодження мають важливе значення у процесі розробки будь-якого програмного продукту. У випадку ігрових проектів перевірка стає ще критичнішою, оскільки вона допомагає виявити помилки, які можуть бути неочевидними. Слід мати на увазі, що у ігрових проектах помилки можуть призводити до візуальних або внутрішніх проблем. Помилки, пов’язані з візуальними аспектами або розташуванням об’єктів, можна відразу помітити під час гри. Але є такі помилки програмного забезпечення, які не можна побачити візуально в процесі гри. Іноді ці помилки можуть призвести до часткових порушень в геймплеї, а у крайніх випадках - навіть до неможливості завершити гру, оскільки вони можуть вплинути на послідовність механіки, пов’язаної з переходом на новий рівень або запуском ключових сценаріїв у сюжеті.

У випадку з реалізацією ігор на ігровому програмному рушії Unity, цей процес виконується безпосередньо в самому редакторі. Для тестування та відладки гри під час розробки, потрібно встановлювати ігрові об’єкти та сцени у такий стан, який потрібно протестувати та натиснути кнопку «Play». Таким же чином виконується тестування і відладка після закінчення розробки, достатньо виставити гру у її початковий стан и натиснути ту ж саму кнопку. На рисунку 1.24 та 1.25 можна побачити процес тестування гри. В ході тестування було виявлено ряд помилок у роботі скриптів по передачі інформації від меню розподілу характеристик гравця до першого рівня. Ці помилки були викликані відсутністю посилання у одному із полів скрипту завантажувальника рівнів.

Як і було сказано, основною метою тестування було проведення перевірки готовності поточного «білду» проекту. Він дає змогу отримати ігровий досвід гри в жанрі адвенчура, а головне перевірити роботу системи контролю та передачі даних між рівнями гри. Було перевірено якість передачі інформації, перезавантажуючи рівень по декілька разів з різними вхідними даними. Успішно перевіряти зміну у параметрах гравця дали змогу події та текстові повідомлення, що мали з’являтись лише 1 раз.

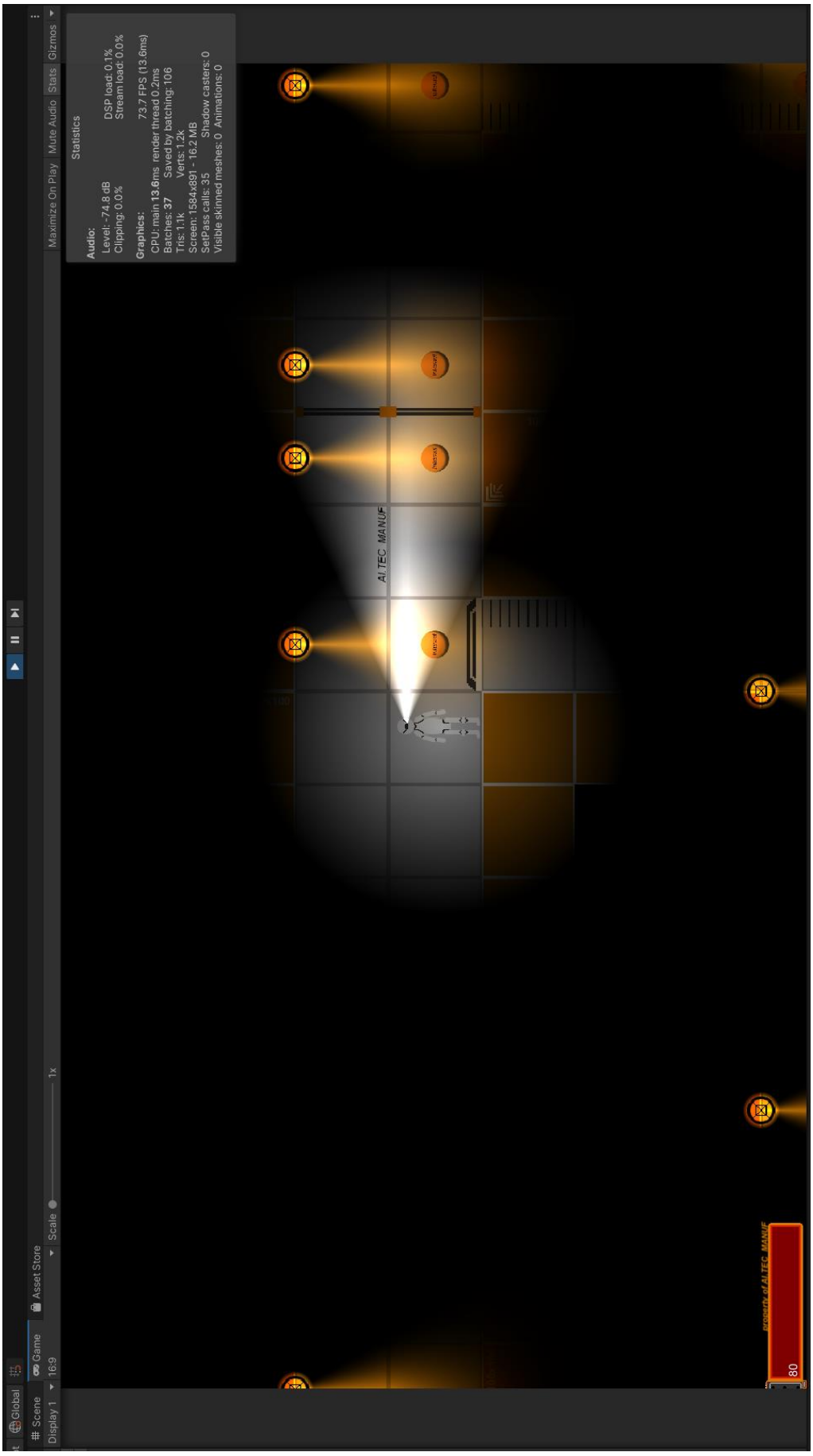


Рисунок 1.24. Процес тестування гри

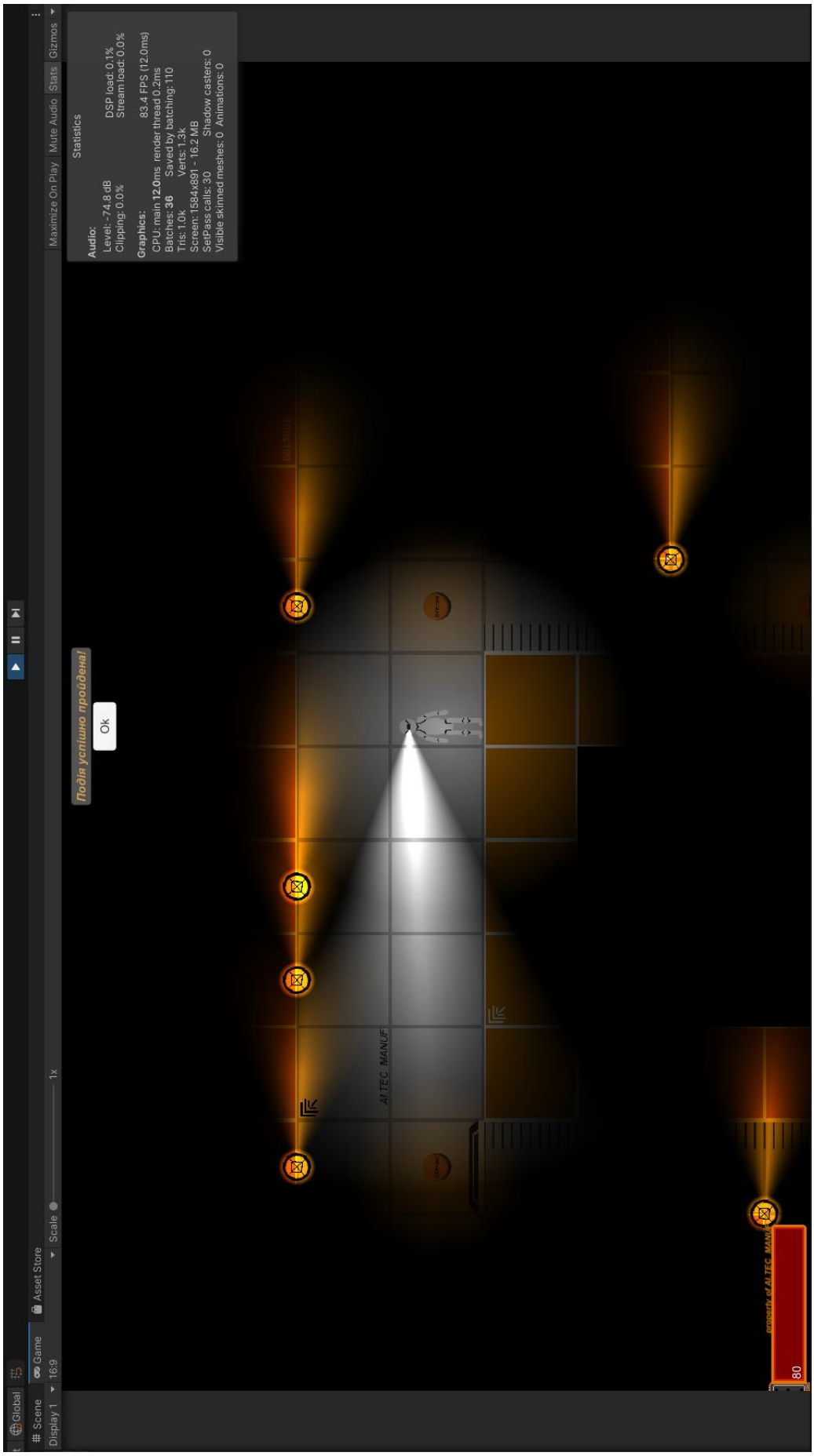


Рисунок 1.25. Процес тестування гри

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проекті розроблено та реалізовано 2D-гру у жанрі горизонтального скролл-шутеру на програмному рушії Unity. Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

Проведемо розрахунки визначення трудомісткості розробки даного програмного продукту.

2.2 Розрахунок ціни програмного продукту нормативним методом

2.2.1 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку.

Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.

Таблиця 2.1 Аналоги програмного забезпечення

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	950 – 7430
3. ПП введення інформації	1060 – 5750
4. ПП оптимізації розрахунків	1300 – 4200
5. ПП автоматизації засобів по каталогу	680 – 7000
6. ПП автоматизованих розрахунків	1300 – 8600
7. ПП загальної математики і ПП імітаційного моделювання	7800 – 8800
8. ПП організації обчислювального процесу	13000 – 10200

Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця 2.2 Визначення трудомысткості

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262
4.00	283
5.00	306
6.00	330
7.00	357
8.00	385
9.00	414
10.00	445
12.00	510
14.00	580
16.00	654
18.00	731
20.00	812

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,7 = 160,3 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4).

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нове ПЗ	1,75 – 1,2
Б	ПЗ – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПЗ, що має аналог	0,7

Для нашого варіанта виділено сірим кольором.

Тому що розробка системи є ПЗ, що має аналоги програмних продуктів, то код ступеня новизни для мого ПЗ – В, а значення коефіцієнта $К_н$=0,7. По таблиці 2.3, знаючи код ступеня новизни, тепер можна визначити значення питомих коефіцієнтів трудомісткості:

L_1=0,12;

L_2=0,11;

L_3=0,61;

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПЗ типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

У розробленому програмному продукті використовується від 40 до 60 відсотків існуючих функцій, це значить, що $К_т$=0,7.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$T_{тз}=Ta*L_1*K_н=160,3\ *0,12*0,8=15,38\ (люд/годин) \qquad (2.4)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=Ta*L_2*K_н=160,3\ *0,11*0,8=14,11\ (люд/годин) \qquad (2.5)$$

Трудомісткість розробки робочого проекту

$$T_{рп}=Ta*L_3*K_н*K_т=160,3\ *0,61*0,8*0,8=62,58\ (люд/годин) \qquad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап:

- технічне завдання $N_{тз}=3$ (стр),

- розробка ТП $N_{тп}=15$(стр),

- розробка робочого проекту $N_{рп}=20$ (стр),

- пояснювальна записка відповідно $N_{пз}=30$ (стр)

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Ртз}=15,38$	$T_{кк}=0,7*N_{тз}=0,7*3=2,1$	$T_{нк}=0,15*N_{тз}=0,15*3=0,45$
2.Розробка ТП	$T_{Ртп}=14,11$	$T_{кк}=0,7*N_{тп}=0,7*15=10,5$	$T_{нк}=0,15*N_{тп}=0,15*15=2,25$
3.Розробка РП	$T_{Ррп}=62,58$	$T_{кк}=0,7*N_{рп}=0,7*20=14,0$	$T_{нк}=0,15*N_{рп}=0,15*20=3,0$
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}=1,5*30=45$	$T_{кк}=0,7*N_{тз}=0,7*30=21,0$	$T_{нк}=0,15*N_{пз}=0,15*30=4,5$
Усього, в т.ч.:	194,87		
- на розробку	$\Sigma T_р=137,07$		
- контроль керівника		$\Sigma T_{кк}=47,6$	
- нормоконтроль			$\Sigma T_{нк}=10,2$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Роботодавець або уповноважені ним органи зобов'язані дбати про умови праці працівників, полегшувати їх, оздоровляти навколишнє середовище, дбати про виконання правил безпеки і інструкцій по техніці безпеки.

Координує всю цю діяльність служба охорони праці, яка в залежності від чисельності працюючих може функціонувати як самостійний структурний підрозділ (число працюючих 50 і більше), або у вигляді групи спеціалістів чи одного спеціаліста, у тому числі за сумісництвом (число працюючих 20 і менше). Задачі службі охорони праці та її функції викладені в Типовому положенні про службу охорони праці», яке затверджено наказом Комітетом Держнаглядохоронпраці (ДНАОП 0.00-4.21-93) .

Працівники також повинні відповідально ставитись до охорони праці, знати та виконувати вимоги, визначені нормативною документацією. В сучасних умовах кожному працівнику необхідно постійно підтримувати високий фізичний, психологічний та фаховий рівень, запобігати виникненню випадків травматизму та профзахворювань.

Безпечні умови праці на підприємстві досягаються за рахунок забезпечення безпеки виробничих процесів, які обґрунтовані і прийняті в технологічній частині дипломного проекту.

3.1 Аналіз небезпечних та шкідливих чинників, що впливають на працівника

Для установлення можливого впливу на здоров'я користувачів ВДТ виробничих чинників має значення ряд якісних характеристик робочого середовища. Це середовище у приміщеннях (офісах) в основному характеризується такими фізичними параметрами, як температура, вологість та електричний опір підлоги. Фізико-хімічні показники включають інформацію про вміст у повітрі іонів та різноманітних забруднювачів, а також деякі інші якісні характеристики середовища

3.2 Розробка заходів з охорони праці

3.2.1 Виробничі приміщення

Будівлі та приміщення, де розміщені робочі місця програмістів повинні відповідати вимогам СНіП 2.09.02-85 «Производственные здания» та ДСанПіН 3.3.2.007 «Державні санітарні правила і норми роботи з ВДТ ЕОМ» Вони мають бути не нижче другого ступеня вогнестійкості. Для всіх приміщень повинно бути визначено клас зони згідно з НПАОП 40.1-1.01-97. Відповідне позначення повинно бути нанесено на вхідних дверях кожного приміщення.

Не дозволяється розташування приміщень з робочими місцями операторів ПК у підвалах і цокольних поверхах. Площа приміщення із розрахунку на одне робоче місце має бути не менше 6,0 кв.м, а об'єм – не менше 20,0 куб.м.

Для внутрішнього оздоблення приміщень з ПК слід використовувати дифузно-відбивні матеріали з коефіцієнтом відбитті для стелі 0,7 – 0,8, для стін 0,5 – 0,6. Покриття підлоги повинне бути матовим, поверхня рівною, не слизькою, з антистатичними властивостями.

Віконні прорізи приміщень для роботи з ПК мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки).

Забороняється для оздоблення інтер'єру приміщень з ПК застосовувати полімерні матеріали, що виділяють у повітря шкідливі хімічні речовини. Приміщення можуть обладнуватись шафами для зберігання документів, полицями, стелажами.

У приміщеннях слід щоденно робити вологе прибирання. Вони мають бути оснащені аптечками першої медичної допомоги.

При приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження, де слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04 – 87).

3.2.2 Мікроклімат робочої зони працівників, вентиляція

Висока температура повітря негативно позначається на функціональному стані людини. Хоч генерація теплоти дисплеєм досягає критичного рівня тільки у саму теплу пору року, необхідно створювати комфортні теплові умови постійно.

Оптимальні та допустимі мікрокліматичні параметри у приміщеннях повинні враховувати специфіку технологічного процесу при використанні комп'ютерів. Згідно з діючими у нашій країні нормативними документами (ДСанПіН 3.3.2-007-98 у холодні періоди року температура повітря, швидкість його руху та відносна вологість повітря повинні відповідно складати: 22-24^{0}С; 0,1 м/с; 40-60%. Температура повітря може коливатись у межах від 21 до 25^{0}С при збереженні інших параметрів мікроклімату.

В теплі періоди року температура повітря, його рухливість та відносна вологість повинні відповідно становити: 23-25^{0}С; 0,1-0,2 м/с; 40-60 %.

Оптимальним рівнем аероіонізації у зоні дихання користувача вважається вміст легких аерофонів обох знаків від 150 до 5000 у 1 см^3 повітря.

Нормалізуючий вплив на склад повітря робочої зони справляють примусова вентиляція, захисні екрани (оснащені заземленням) та застосування іонізаторів.

3.2.3 Освітлення робочого місця, шум, вібрація

Освітлення у приміщеннях з ВДТ має бути змішаним – природним та штучним. Природне освітлення повинно здійснюватись у вигляді бічного освітлення та відповідати нормам ДБН В.2.5-28-2006 «Природне і штучне освітлення».

При природному освітленні слід передбачити наявність сонцезахисних засобів, що знижують перепади яскравостей між природним світлом та свіченням екрана ВДТ. З цією метою можна використовувати плівки з металізованим покриттям або жалюзі з вертикальними лам елями, що регулюються.

Штучне освітлення у приміщеннях з ВДТ треба здійснювати у вигляді комбінованої системи освітлення з використовуванням люмінесцентних джерел світла у світильниках загального освітлення. На робочих місцях має бути забезпечена рівномірна освітленість за допомогою переважно відбитого або розсіяного світлорозподілу. Світлових відблисків з клавіатури, екрана та від інших частин ВДТ у напрямку очей користувача не повинно бути.

Норма освітленості на робочих місцях складає 300-500лк.

Деякі ВДТ є потенційними джерелами цілого ряду звуків, що містять як коливання, які можна почути, так і коливання ультразвукового діапазону. Цей шум справляє негативний вплив на стан користувача, особливо при тривалому впливі.. У користувача, діяльність якого пов'язана з переробкою інформації це виражається у знижені розумової працездатності, зростає кількість помилок, розвиток зорового втомлення, зміні відчуття кольорів, появі головного болю, послаблення уваги. Нормованим параметром шуму на робочих місцях є рівень 50 дБ. Основними заходами боротьби з шумом є усунення або ослаблення причин шуму в самому його джерелі у процесі проектування, використання засобів звукопоглинання, раціональне планування виробничих приміщень.

3.2.4 Електробезпека

Причинами ураження працівника електрострумом можуть бути:

- Випадковий дотик до струмоведучих частин, у результаті ведення робіт поблизу або на цих частинах;
- Випадковий дотик до струмоведучих частин, у результаті ведення робіт поблизу або на цих частинах;
- Несправність захисних засобів, якими потерпілий доторкався до струмоведучих частин;

Помилкове прийняття устаткування, що перебуває під Електробезпека.

Значення сили струму, що проходить через організм людини, залежить від напруги, під якою перебуває людина й від опору ділянки тіла, до якого прикладена ця напруга. Джерелом живлячої напруги є мережа змінного струму з напругою 229В, на яку поширюється ГОСТ 25861-83.

Основними причинами електротравматизму є:

- напругою, як відключеного;
- Несподіване виникнення напруги через ушкодження ізоляції там, де в нормальних умовах його бути не повинно;
- Контакт струмопровідного устаткування із проводом, що перебуває під напругою.

Для попередження поразок електричним струмом необхідно чітко й у повному обсязі виконувати правила провадження робіт і правил технічної експлуатації. Необхідно виключити можливість доступу оператора до частин устаткування, що працює під небезпечною напругою, до неізольованим частинам, призначеним для роботи при малій напрузі й не підключеним до захисного заземлення, а також підводити електроживлення до ПЕОМ від розетки за допомогою спеціальної вилки із заземлюючим контактом.

3.2.5 Організація робочого місця користувача ПК

Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування, ергономічним вимогам, з урахуванням характеру і особливостей трудової діяльності (ДСанПіН 3.3.2.-007-98).

Конструкція робочого місця й взаємне розташування всіх його елементів відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів дає можливість забезпечувати можливість індивідуального регулювання їх відповідно до потреб працівника для підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей, на відстані близько 70 см, що укладається в припустимі рамки від 60 до 90 см. Частота мерехтіння екрана дорівнює 100 Гц, що відповідає умові більше 70 Гц.

Для зниження нервово-емоційного напруження, стомлювання, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно впроваджувати виконання комплексу вправ.

ВИСНОВКИ

На початку виконання дипломного проекту було виконано загальний аналіз ігрового продукту у наявності, а також того, що необхідно переміщувати між рівнями. Оскільки гра була виконана в жанрі адвенчура, кількість інформації для передачі між рівнями була не великою, тому проектування такої системи не було трудомісткою задачею.

Сама система контролю та передачі даних між рівнями гри була поділена на модулі, та спроектована таким чином, щоби урахувати всі особливості переходи з однієї сцени ігрового програмного рушія до іншої. Реалізація таких модулів була виконана у виді скриптів передачі інформації, які взаємодіяли із елементами на сценах. Для кожного рівня необхідно було створювати свій скрипт збереження інформації, оскільки елементи не були однотипними, а систему контролю можна було виконати у одному екземплярі.

Тестування виконаної роботи проводилось шляхом завантаження сцен та проходження рівнів гри, з виконанням різних дій, які потребували перенесення у інші рівні. Оскільки ігровий продукт був розроблений таким чином, що не було наскрізних подій, це полегшило роту та зменшило кількість ітерацій тестування.

Результатом виконання дипломного проектування було створення системи контролю та передачі даних між рівнями, яка дала змогу урізноманітнити ігровий досвід під час проходження гри. Модульність реалізації дає змогу використати шаблон виконаної системи, але через особливості ігрового програмного рушія Unity, кожний новий рівень потребує втручання з боку розробника для використання розробленого рішення. Для вирішення цієї проблеми необхідно перепроектовувати весь ігровий продукт, що не є доцільним рішенням.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Гусєв Б.С. Комп’ютерна схемотехніка [навчальний посібник] // – К.: НУБіП України, 2022. – 264с.
2. Матвієнко М.П., Розен В. П. Комп’ютерна схемотехніка: навчальний посібник. – К.: Видавництво Ліра-К, 2020. – 192 с.
3. Дейбук В.Г. Віртуальний електронний практикум: Навчальний посібник – Чернівці: Чернівецький нац. ун-т, 2021. – 188 с.
4. Трофименко О.Г. С++. Алгоритмізація та програмування: підручник / О.Г. Трофименко, Ю.В. Прокоп, Н.І. Логінова, О.В. Задерейко. 2-ге вид. перероб. і доповн. – Одеса : Фенікс, 2019. – 477 с.
5. Азаров О. Д., Гарнага В. А., Клятченко Я. М., Тарасенко В. П. Комп’ютерна схемотехніка: підручник. – Вінниця: ВНТУ, 2018. – 230 с.
6. Архангельский А.Я. Программирование в C++Builder, 7-е изд. – М.: ООО Бином-Пресс, 2010 г. – 1230с., ил.
7. Нікулін В.С. Перетворювальні пристрої, ведені мережею: Конспект лекцій. –Харків: УкрДАЗТ, 2008. – Ч.4. – 85 с.
8. Мосіюк О.О., Федорчук А.Л. Операційні системи та системне програмування: навчально-методичний посібник. Житомир: Вид-во ЖДУ ім. Івана Франка, 2022. – 76 с.
9. Stroustrup B. A Tour of C++ (Second Edition). – Addison-Wesley, 2018. – 240 p. – ISBN 978-0-13-499783-4.
10. Каганюк О.К. Комп'ютерна схемотехніка: Навчальний посібник. – Луцьк: РРВ Луцького НТУ, 2016. – 236 с.
11. Бібліотека MSDN [Електронний ресурс]. – Режим доступу: URL http://msdn.microsoft.com/ru-ru/library/default.aspx.

ДОДАТОК А. Лістинг коду системи контролю та передачі даних між рівнями на мові C#

Скрипт player_parameters.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Experimental.Rendering.Universal;

public class player_parameters : MonoBehaviour
{
    scenes_loader _scenes_loader;

    [SerializeField] health_bar_logic _health_bar_logic;
    [SerializeField] Light2D _player_light_sphere;
    [SerializeField] Light2D _player_flashlight;

    [SerializeField] int strength = 0;
    [SerializeField] int dexterity = 0;
    [SerializeField] int perception = 0;
    [SerializeField] int luck = 0;
    [SerializeField] int health = 100;

    Collider2D triggered_collider;
    event_script _event_script;

    private void Start()
    {
        _scenes_loader = this.GetComponent<scenes_loader>();
    }

    private void Update()
    {
        if (Input.GetKeyUp(KeyCode.C))
        {
            take_damage(health);
        }
    }

    public void take_player_parameters(int strength_value, int dexterity_value, int perception_value, int luck_value)
    {
        strength = strength_value;
```

```
    dexterity = dexterity_value;
    perception = perception_value;
    luck = luck_value;

    setup_player_modifier();
}

void setup_player_modifier()
{
    _health_bar_logic.set_hp_max_value(60+(strength * 10));

    _player_light_sphere.pointLightOuterRadius = perception * 0.5f;

    _player_flashlight.pointLightOuterRadius = perception * 1.25f;
}

public int take_strength_value()
{
    return strength;
}

public int take_dexterity_value()
{
    return dexterity;
}

public int take_perception_value()
{
    return perception;
}

public int take_luck_value()
{
    return luck;
}

public int take_health_value()
{
    return health;
}

public void take_damage(int damage_value)
{
```

```
        health -= damage_value;

        if (health < 0)
            health = 0;

        if (health == 0)
        {
            Debug.Log("Player is dead");//kill

            _scenes_loader.load_main_menu();
        }


        _health_bar_logic.set_hp_value(health);
    }
}
```

Скрипт scenes_loader.cs

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

public class scenes_loader : MonoBehaviour
{
    [SerializeField] parameters_logic strength_parametr;
    [SerializeField] parameters_logic dexterity_parametr;
    [SerializeField] parameters_logic perception_parametr;
    [SerializeField] parameters_logic luck_parametr;

    static bool[] training_messages_lvl1 = new bool[10];

    void Update()
    {

    }

    public void load_main_menu()
    {
        main_menu_receiver.increase_death_counts();

        EditorSceneManager.LoadScene("Main_menu");
    }
```

```
    public void load_level_1()
    {
        level_1_receiver.strength_parameter = strength_parametr.value;
        level_1_receiver.dexterity_parameter = dexterity_parametr.value;
        level_1_receiver.perception_parameter = perception_parametr.value;
        level_1_receiver.luck_parameter = luck_parametr.value;

        EditorSceneManager.LoadScene("Level_1");
    }

    public void change_training_messages_interact_state(int
number_of_message)
    {
        training_messages_lvl1[number_of_message] = true;
    }

    public bool take_training_message_interact_state(int number_of_message)
    {
        return training_messages_lvl1[number_of_message];
    }

    public void exit_game()
    {
        Application.Quit();
    }
}
```

Скрипт level_1_receiver.cs

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

public class level_1_receiver : MonoBehaviour
{
    [SerializeField] player_parameters _player_parameters;

    public static int strength_parameter;
    public static int dexterity_parameter;
    public static int perception_parameter;
    public static int luck_parameter;

    private void Start()
    {
        send_player_parameters();
```

```
        }

        public void send_player_parameters()
        {
            _player_parameters.take_player_parameters(strength_parameter,
dexterity_parameter, perception_parameter, luck_parameter);
        }
    }
```

Скрипт data_checker.cs

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

    public class data_checker : MonoBehaviour
    {
        public static int strength_parameter;
        public static int dexterity_parameter;
        public static int perception_parameter;
        public static int luck_parameter;

        public bool check_player_parameters(int str, int dex, int per, int luck)
        {
            if ((str == strength_parameter) && (dex == dexterity_parameter) && (per
== perception_parameter) && (luck == luck_parameter))
            {
                return true;
            }
            else
                return false;
        }

        public void take_player_parameters(out int str, out int dex, out int per, out int
luck)
        {
            str = strength_parameter;
            dex = dexterity_parameter;
            per = perception_parameter;
            luck = luck_parameter;
        }
    }
```

ДОДАТОК Б. Слайди мультимедійної презентації

Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity

ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА ГРУПИ 4КС-57: ПАЛАДІ СЕРГІЯ ВАЛЕРІЙОВИЧА

КЕРІВНИК: ШУВАЛОВА І.О.

Особливості ігрового жанру адвенчура

Адвенчура – це відеогра, яка акцентується на сюжеті та дослідженні світу гри. Дуже часто такий процес супроводжується розв'язуванням головоломок.

Основними характеристиками жанру є:

- Неквапливий ігровий процес;
- Концентрацію ігрового процесу на сюжеті;
- Вирішення головоломок;
- Виконання завдань від персонажів.

Особливості програмного рушія Unity та причини його вибору для розробки проекту

Ігровий двигун Unity є одним із популярніших ігрових двигунів у світі. За його допомогою створюють як мобільні, так і комп'ютерні ігри. Цей двигун досі активно розвивається!

- Має зручну візуальну середу розробки;
- Має можливість міжплатформенної розробки ігор;
- Підтримує модульну систему компонентів;
- Використання мови програмування C# та його можливостей;
- Велике ком'юніті розробників;
- Зрозуміла та вичерпна документація, безліч курсів у інтернеті та літературі;
- Гнучка система ліцензування ігрового двигуна;
- Безплатні та платні ассети для проектів будь-якого рівня.

Особливості механіки системи контролю та передачі даних між рівнями

Механіка контролю та передачі даних між рівнями у основана на вимогах збереження даних у іграх жанру адвенчури.

- Створює умови для гарантованого збереження всіх даних;
- Дає змогу проходити гру більш ніж за одну сесію;
- Система може передавати будь-які дані за потребою подальшої розробки;
- Можливість швидкого додавати нових даних до передачі та їх перевірку.

Огляд елементів системи контролю та передачі даних між рівнями

Для повноцінної реалізації системи контролю та передачі даних між рівнями потрібно створити умови для тестування.

Для цього буде використано поняття характеристик гравця, які є частиною модулю гравця.

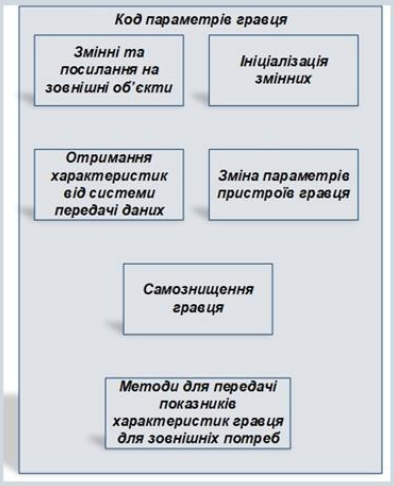
Таким чином цей модуль можна назвати частиною розробляэмої системи.

Отримує значення характеристик із системи передачі даних, виконує звернення до контролера.

Принцип роботи елементів системи контролю та передачі даних між рівнями

Код параметрів гравця складається з:

- Змінних та посилань на зовнішні об'єкти (в тому числи елементи системи контролю передачі даних);
- Ініціалізатор змінних;
- Отримання характеристик від системи передачі даних;
- Зміна параметрів пристроїв гравця;
- Самознищення;
- Методи для передачі показників характеристик гравця для зовнішніх потреб.



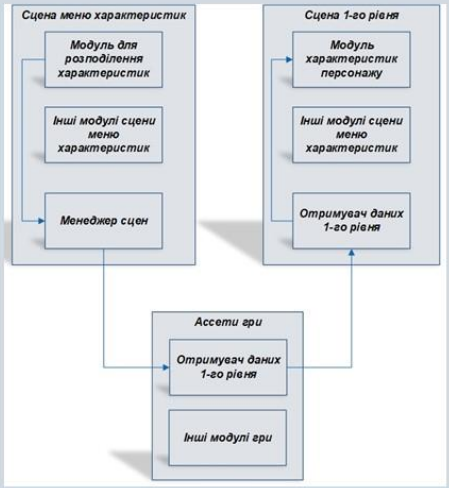
Принцип роботи елементів системи контролю та передачі даних між рівнями

Система передачі даних працює у наступному порядку:

Меню розподілу характеристик передає значення у Менеджер сцен.

Менеджер сцени звертається до цільового отримувача даних.

Отримувач даних передає ці дані до модуля характеристик гравця у потрібній сцені рівня.

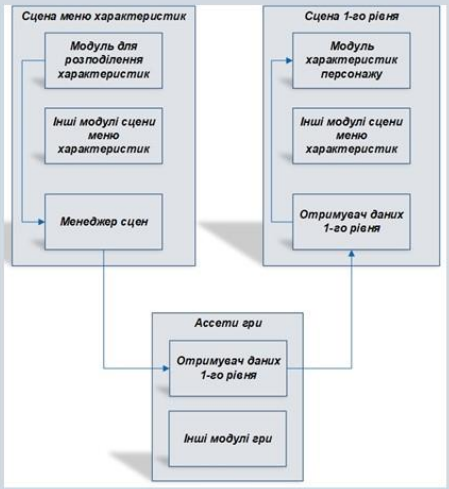


Принцип роботи елементів системи контролю та передачі даних між рівнями

Можливість передавати дані між сценами поза дією виконуючої середи досягається за допомогою статичності.

Кожний отримувач даних для кожного рівня має свої статичні параметри, на які записуються необхідні дані.

Принцип ООП забезпечує виконання збереження даних на протязі всього активного циклу виконання ігрового додатку



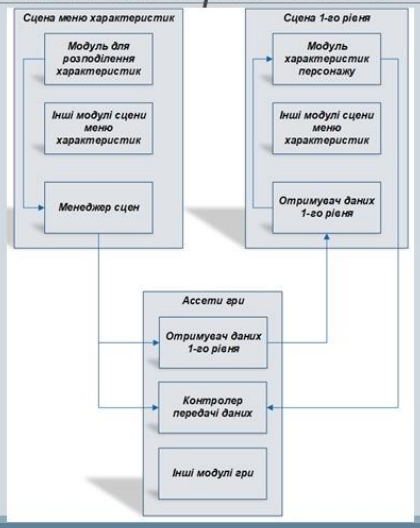
Принцип роботи елементів системи контролю та передачі даних між рівнями

Контролер передачі даних працює по тому ж принципу, що і отримувачі даних.

Контролер є дублюючим елементом, який також знаходиться зовні сцен.

Менеджер сцен при передачі даних до отримувачів даних, також передає їх до контролера.

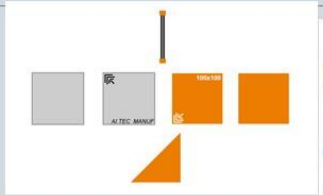
Контролер, за вимогою, може перевірити необхідні дані та у випадку неспівпадіння передати свої копії.



Етапи реалізації елементів системи контролю та передачі даних між рівнями

Реалізація елементів системи контролю та передачі даних між рівнями складалась з:

- Реалізації спрайтів для рівня гри;
- Створення рівня із спратів;
- Створення ігрового об’єкту гравця;
- Написання коду інших модулів гри;
- Написання коду системи контролю та передачі даних між рівнями;
- Тестування.

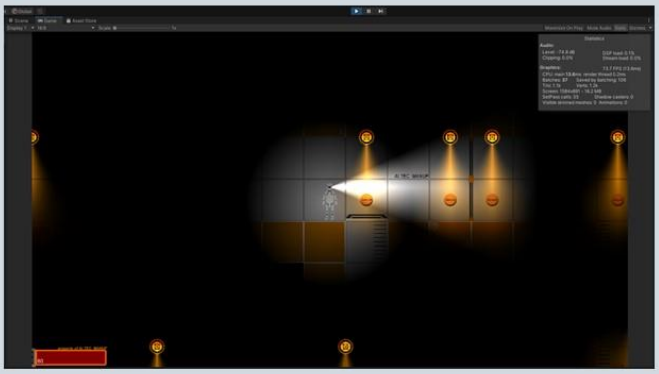


Хід тестування гри

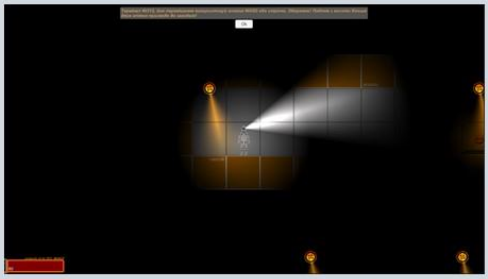
Тестування гри проводилось з допомогою вбудованого у ігровий програмний рушій Unity інструмент.

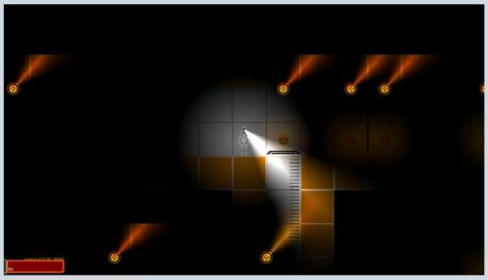
Метою тестування було:

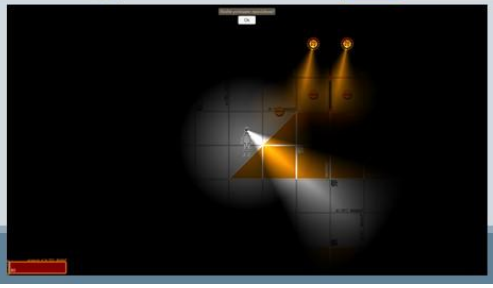
- Провести перевірку поточного «білду» ігрового проекту;
- Перевірити роботу системи контролю;
- Перевірити роботу системи передачі даних;
- Протестувати варіації передачі даних.



Скріншоти реалізованої гри







ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Паладі Сергія Валерійовича

(прізвище, ім`я та по батькові)

Спеціальність: 123 "Комп'ютерна інженерія"

Освітня програма: «Обслуговування комп'ютерних систем і мереж»

Тема дипломного проекту: Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 71 сторінок. У пояснювальній записці виконано опис предметної області, способи реалізації системи передачі та контролю даних між рівнями. Проведено проектування та реалізація систем передачі даних. Графічна частина складається з 12 слайдів мультимедійної презентації, які передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Паладі С.В. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.

в) теоретична підготовка випускника (випускниці): Здобувач освіти Паладі С.В. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Паладі С.В. мав змогу самостійно приймати рішення з реалізації системи контролю та передачі даних між рівнями гри, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Шувалова Ірина Олегівна

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач комісії комп'ютерних технологій та програмної інженерії

Підпис

«10» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Паладі Сергія Валерійовича

(прізвище, ім`я та по батькові)

Спеціальність 123 " Комп'ютерна інженерія "

Освітня програма «Обслуговування комп'ютерних систем і мереж»

Керівник дипломного проекту (роботи) Шувалова Ірина Олегівна

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity*

Обсяг розрахунково-пояснювальної записки 71 сторінок

Обсяг графічної (презентаційної) частини 12 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням створення ігрових продуктів в цілому та розробці ігор на ігровому програмному рушії Unity, в цілому.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі розглянуті питання проблематики розробки гри на ігровому програмному рушії Unity, сформовано вимоги до гри згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів розробляємої гри. За допомогою відповідного програмного забезпечення реалізовані всі намічені зміни до ігрового процесу

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально описано процес виконання розробки системи контролю та передачі даних між рівнями гри;
2. Виконано проектування намічених елементів гри із поясненнями на схемах;
3. Розроблено функціонуючу систему контролю та передачі даних між рівнями гри.

д) основні недоліки дипломного проекту (роботи)

1. Кількість даних для передачі невелика;
2. Реалізована передача даних лише між головним меню та першим рівнем гри;
3. Етапи роботи слід було описати докладніше

Оцінка розрахункової частини Добре

Оцінка графічної частини Відмінно

Загальна оцінка Добре

Прізвище, ім'я, по батькові рецензента Стайкуца Сергій Володимирович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, к.ф.н., доцент кафедри КБ та ТЗІ

Підпис

червня 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григорівна Краснокутська

ID перевірки:
1016351119

Дата перевірки:
12.06.2024 09:50:42 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2024 10:01:00 EEST

ID користувача:
100011688

Назва документа: 4КС-57_Паладі

Кількість сторінок: 43 Кількість слів: 8469 Кількість символів: 59353 Розмір файлу: 2.11 MB ID файлу: 1016154789

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.65%
Схожість

Найбільша схожість: 3.31% з Інтернет-джерелом (https://ela.kpi.ua/handle/123456789/49671)

4.65% Джерела з Інтернету | 56 ..... Сторінка 45

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 | 7 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Паладі Сергій Валерійович,
здобувач освіти гр. 4КС-57, та

Шувалова Ірина Олегівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного дипломного проекту фахового молодшого бакалавра на тему:

«Реалізація системи контролю та передачі даних між рівнями гри у жанрі адвенчура на програмному рушії Unity» (автор роботи – Паладі С.В., керівник роботи – Шувалова І.О.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 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 _____ / Паладі С.В./

Керівник _____ / Шувалова І.О./

«10» 06 2024 р.