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.57.12.000.ДП

ЛЄСНІКОВА
ВІТАЛІЯ
ОЛЕКСАНДРОВИЧА

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 характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity

Проектний матеріал складається з пояснювальної записки на 78 сторінках та графічного (презентаційного) матеріалу на 12 аркушах (слайдах)

Дипломник _____ (Лєсніков В.О.)

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 «18» 06 2024 р. Протокол ЕК № 2

Оцінка ЕК 4 (добре) 75 б.

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*" *01* 2024 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Лєснікову Віталію Олександровичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Реалізація системи характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity*

затверджена наказом по коледжу від "*02*" *11* 202*3* 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 характеристик та вмінь гравця комп'ютерної гри в жанрі адвенчура; Гра повинна мати основні признаки жанру адвенчура; Виконана частина повинна давати змогу гравцю отримувати варіативний ігровий досвід в залежності від характеристик та вмінь персонажу гравця.*

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 системи характеристик та вмінь гравця; Проектування системи характеристик та вмінь гравця; Реалізація елементів системи характеристик та вмінь гравця; 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 характеристик та вмінь у проекті; Огляд елементів системи характеристик та вмінь гравця; Принцип роботи елементів системи характеристик та вмінь гравця; Етапи реалізації елементів системи характеристик та вмінь гравця; 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 *Лєсніков В.О.* _____
(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 характеристик та вмінь*	*28.05.2024*	Виконав
5	*Проектування системи характеристик та вмінь*	*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 характеристик та вмінь*	*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. 12 000. 00 ДП	Дипломний проект		
А4			КС 57. 12 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 57. 12 000. 00 ДП		
Розробив		Лесніков В.О.		10.06	Реалізація системи характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity	Літ.	Аркуш 4 / Аркушів 78
Перевірив		Шувалова І.О.		10.06		Н Д П	
Н. контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4КС-57	
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

Ігрова індустрія має велику кількість основних жанрів, та ще більшу кількість варіацій змішаних жанрів. Кожний жанр відрізняється своїми особливостями у ігровому процесі, як жанри в кінематографі. Деякі ігри складаються тільки з одного жанру, що робить їх монолітними продуктами, які рівно дають гравцю саме той ігровий процес, який відповідає жанру. У випадку із змішаними жанрами, бувають комбінації, які створюють унікальний ігровий досвід, якій буде змінюватись від одного етапу гри до іншого, або захоплювати від початку до кінця.

Одною із основних рис деяких жанрів є системи характеристик персонажів гравців. Вони дозволяють створювати різноманітний досвід під час повторного проходження гри із зміненими характеристиками. Частіше за все така система створюється в іграх із рольовою системою, але в наш час доволі часто можна зустріти ігри, в яких також присутні характеристики персонажів гравців.

Темою мого дипломного проекту є «Реалізація системи характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity». Як мету я ставлю створення такого комплексу програмних рішень, які б дозволили гравцю різним чином проходити гру, змінюючи параметри персонажу гравця, або його вмінь.

Такі рішення будуть реалізовуватись для гри в жанрі адвенчура, що досить нестандартно для продуктів цього жанру. Із введенням системи характеристик, є можливість зробити ігровий процес адвенчури більш різноманітним, що позитивним чином позначиться на кінцевому продукті.

Окрім того, така система дасть змогу у подальшому додавати нові умови проходження гри, або додаткові рівні. Створювати продовження або додатковий контент пов’язаний із характеристиками та уміннями гравця.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз ігрового жанру адвенчура

1.1.1 Загальний огляд питання

Для початку роботи над дипломним проектом є потреба визначитись із початковими питаннями, які будуть основою для кінцевою розробки. Оскільки темою мого дипломного проекту є «Реалізація системи характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity», одним з таких питань є визначення жанру адвенчура. Це, можливо, дасть можливість 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

Для більш детального визначення основних рис жанру та його особливостей з точки зору теми мого дипломного проекту, я розглянув декілька прикладів ігор

жанру «Адвенчура».

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 1.1 зображено скріншот цієї гри:

Рисунок 1.1. Скріншот гри The Secret of Monkey Island

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 1.2 зображено скріншот із гри Myst:

Рисунок 1.2. 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.

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 Strang

1.1.3 Результати аналізу ігрового жанру адвенчура

Під час аналізу було розглянуто три гри одного ігрового жанру. Не дивлячись на це ігрові продукти давали різний підхід до ігрового процесу, але суть підходу завжди зосереджувалась на вирішені завдань, або головоломок всередині гри. Втім кожна гра підходила до цього по своєму. The Secret of Monkey Island робила це через контекстні меню, Life is Strange створювала ситуації в котрих потрібно було швидко вирішувати, а Myst зусереджувалась на головоломках. Втім, є загальна частина в цих проектах – гравець завжди має свободу обирати спосіб вирішення проблеми з тих варіантів, які доступні гравцю на даний момент.

Іншою важливою складовою цих ігор у жанрі адвенчура – є дослідження рівня. У будь-якій грі цього жанру, дослідження рівня є основою ігрового процесу, оскільки в ході переміщення по ігровому простору гравець знаходить предмети, або інформацію, яка допомагає йому у подальшому в діалогах, або ігрових ситуаціях, які з ним можуть статися. Знаходження того, чи іншого предмету може допомогти гравцю пройти гру, або навпаки затримати його на дуже довгий час.

Отже, з точки зору теми мого диплома, потрібно створити таку систему характеристик та вмінь, щоби вона відповідала жанру, та антуражу гри в цілому. Слід зазначити, що адвенчури за рідким виключенням, мають досить спокійний ігровий процес, який дає змогу гравцю розмірковувати та приймати рішення без

тиску зі сторони ігрових ситуацій. Тобто мені потрібно зробити систему характеристик такою, щоби поглибити ігровий процес та розширити його з точки зору різноманітності. Те ж саме стосується й системи вмінь. Необхідно створити таку систему, щоби вона створювала унікальні ситуації, або давала ситуативні бонуси до ігрового процесу. Все це створить умови для покращення якості ігрового процесу, та глибини жанру в цілому. На рисунку 1.4 можна побачити основні складові ігрового жанру Адвенчура.

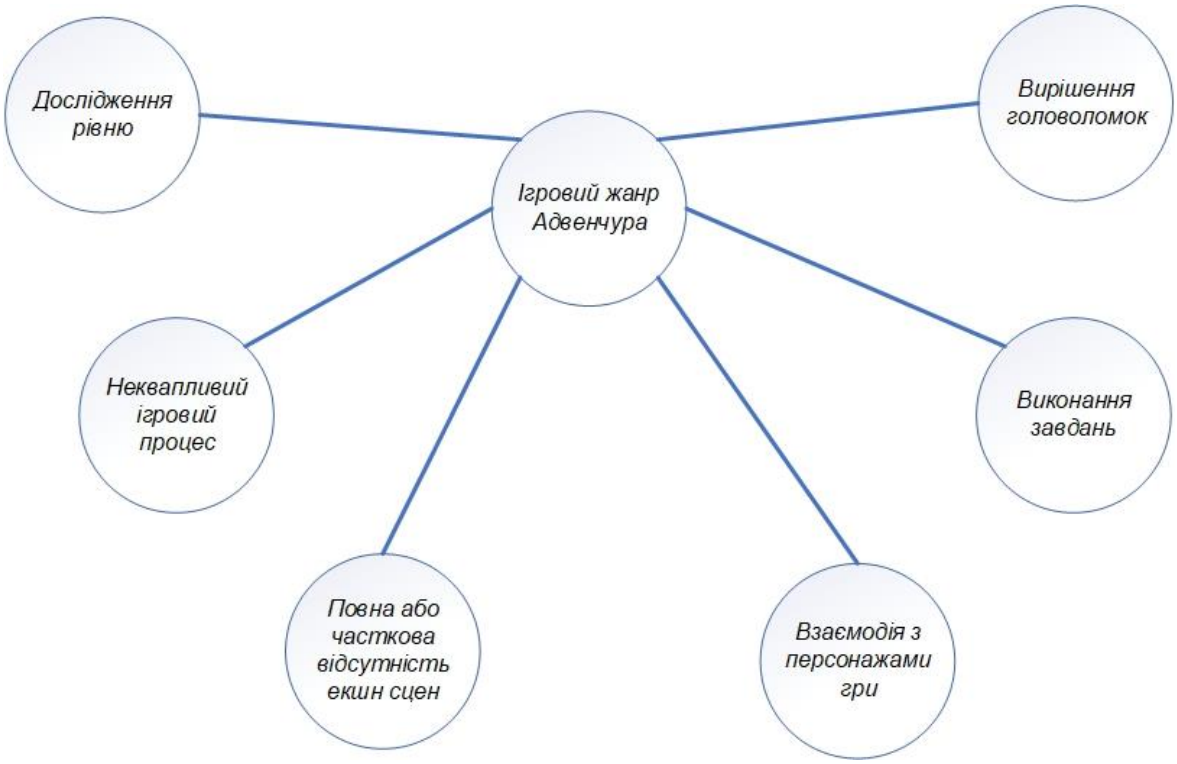


Рисунок 1.4. Складові ігрового жанру Адвенчура

1.2 Аналітичний огляд програмних рушіїв з умовно безкоштовним доступом

1.2.1 Огляд ігрового програмного рушія Unity

Буд-яка розробка ігрового проекту основана на виборі програмного забезпечення для вирішення питання технічної реалізації та супроводу. Вибір ігрового програмного рушія є дуже важливою частиною початку розробки, оскільки визначає важливі питання використання технологій розробки, принципів архітектури гри, способів реалізації механік та багато інших питань. Дуже часто ігрові програмні рушії диктують розробникам те, як буде працювати їх проект і

чи буде він працювати зовсім.

Наприклад, одною із головних причин провалу гри Anthem – це її технічна сторона, яка була дуже поганою через створення рольової гри із елементами кастомізації та менеджменту, відкритого світу та глибокої мультиплеєрної взаємодії, використовувався внутрішній ігровий програмний рушій компанії Electronic Arts, що початково створювався як ігровий програмний рушій для ігор жанру шутер, в яких не було наміру реалізовувати ресурси, глибоку кастомізацію та багаторівневу мультиплеєрну взаємодію. Через це розробникам доводилось витрачати багато часу та зусиль для того, щоби створювати окремий інструментарій для розробки рольових ігор, залучаючи для цього розробників з інших підрозділів компанії Electronic Arts.

Ігрові програмні рушії можна розділити на безкоштовні, умовно безкоштовні та платні. До останніх найчастіше відносяться ігрові програмні рушії створені всередині крупних ігрових компаній для внутрішніх проектів, найчастіше, ці компанії не діляться своїми рішеннями з конкурентами, тільки у випадках партнерства, або фінансових відносин. Безкоштовні ігрові програмні рушії найчастіше розповсюджуються у вигляді бібліотек із мінімалістичним інструментарієм для розробки, що викликає досить серйозні проблеми для розробки. Ідеальними у відношенні ціна-якість є ігрові програмні рушії, що розповсюджуються на умовно безплатній основі. Ці програмні рушії мають величезний інструментарій для розробки, а також не потребують плати за використання, при виконанні окремих умов.

Тема мого дипломного проекту «Реалізація системи характеристик та вмінь гравця 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:

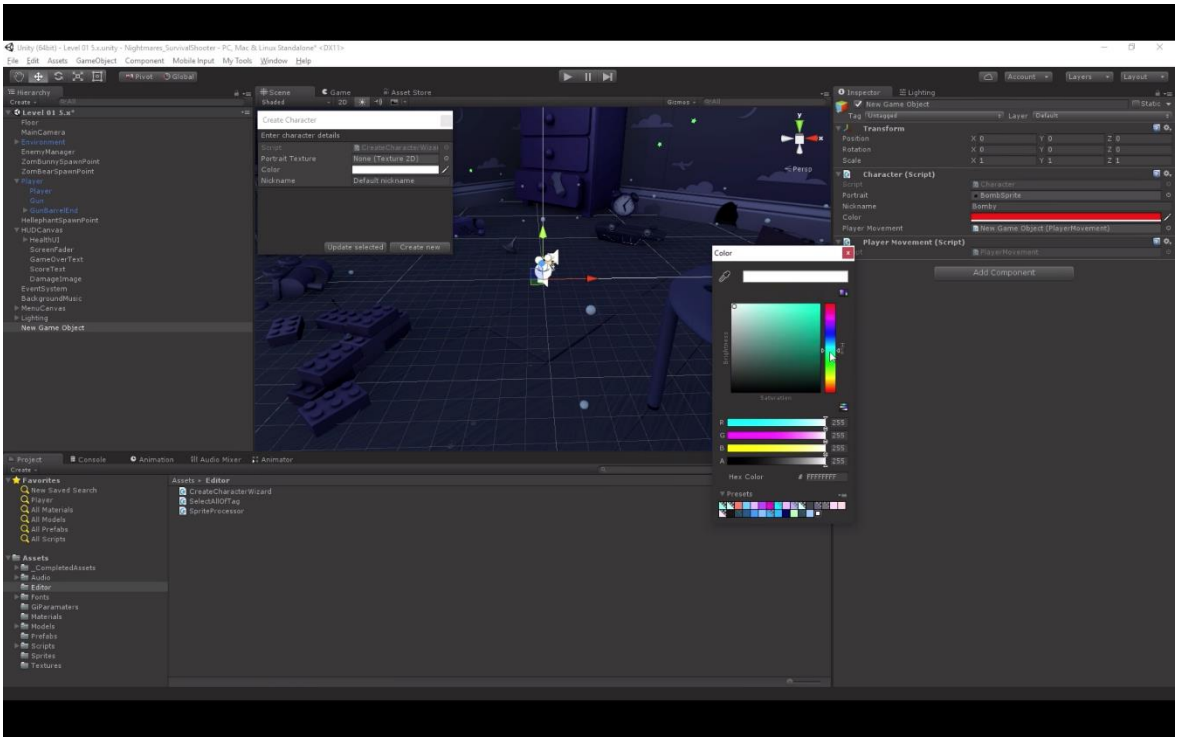


Рисунок 1.5. Приклад роботи у ігровому програмному рушії Unity

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 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 1.6 можна побачити приклад роботи із анімацією у ігровому рушії Unreal Engine.

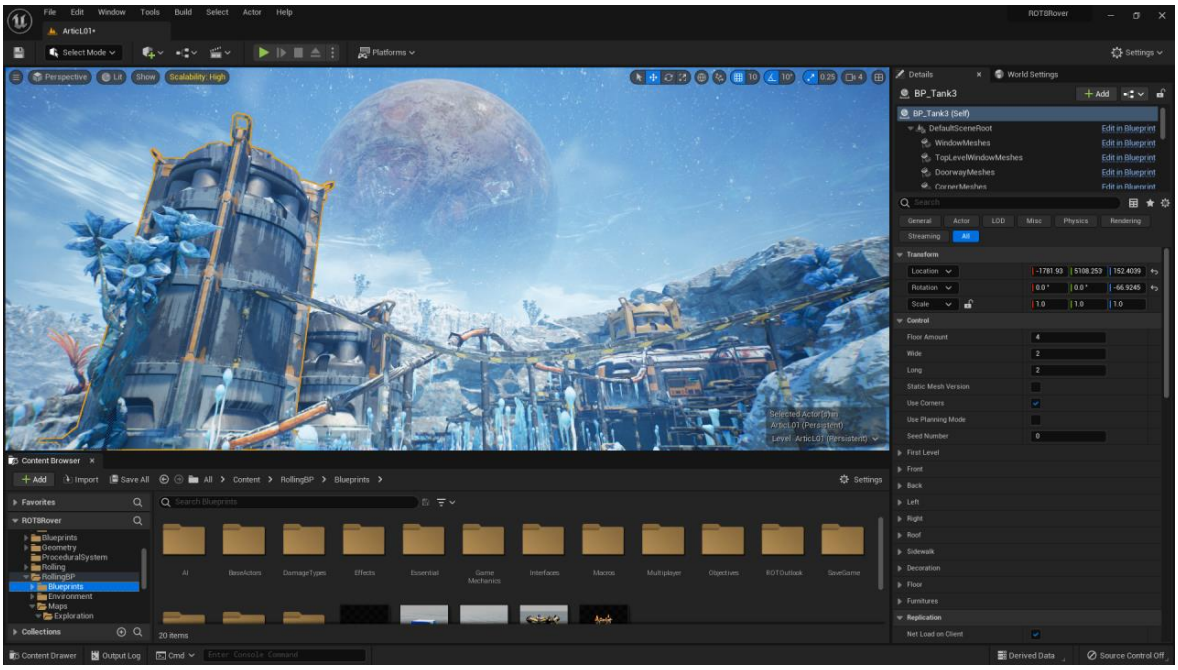


Рисунок 1.6. Приклад роботи із анімацією у ігровому рушії Unreal Engine

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 AAA-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 1.7 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 1.7. Приклад графіки яку відтворює Unreal Engine

1.2.3 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 1.8 можна побачити приклад графіки, що відтворює програмний рушій CryEngine.

Рисунок 1.8. 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 ААА-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 1.9 можна побачити приклад роботи з вбудованим аніматором рушія CryEngine.

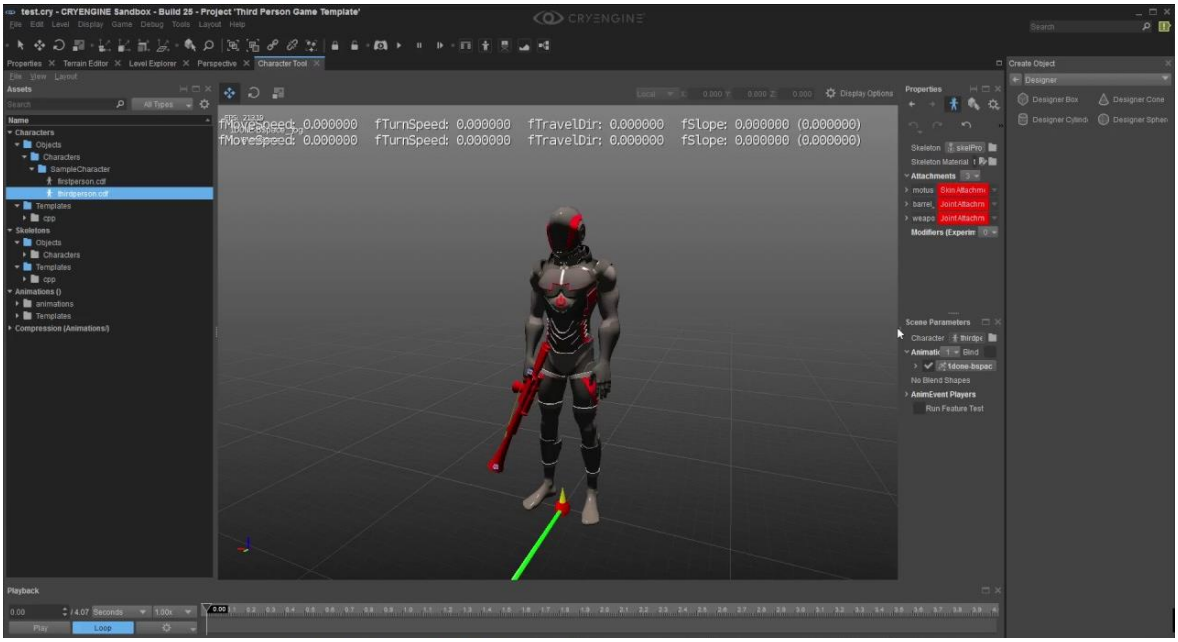


Рисунок 1.9. Приклад роботи з вбудованим аніматором рушія CryEngine

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.

1.2.4 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 обрання інструментарію. Програмне забезпечення для розробки має дуже велику роль у процесі роботи, адже кожне рішення має свої особливості та що не менш важливо, користувач може мати свої улюблені редактори, в яких має навички роботи та знає всі нюанси. Більшість крупних компаній розробників досить вільно відносяться до прикладного інструментарію розробки, наприклад, як редактори коду. У ході виконання моєї дипломної роботи, мені потрібно було реалізувати деяку графічну частину, тому до вибору графічного редактору потрібно було поставитись дуже відповідно.

Редактор коду – це спеціалізоване програмне забезпечення, яке використовується для створення, модифікації та організації вихідного коду програм. Такі редактори зазвичай мають інтуїтивно зрозумілий інтерфейс, підтримують підсвічування синтаксису, автоматичне доповнення коду, пошук та заміну тексту, а також інтеграцію з іншими інструментами розробки, включаючи системи контролю версій, дебагери та компілятори.

У процесі вибору з численних редакторів коду, я віддав перевагу Microsoft Visual Studio. Цей вибір був зумовлений тим, що ігровий движок Unity працює на мові програмування C#, що вимагає від середовища розробки не лише підтримки форматування коду, але й здатності взаємодіяти з іншими класами, методами, ігровими об'єктами та бібліотеками. Крім того, Microsoft Visual Studio є рекомендованим вибором серед розробників та підтримки Unity. На рисунку 1.9 зображено вікно розробки Visual Studio проекту у Unity:

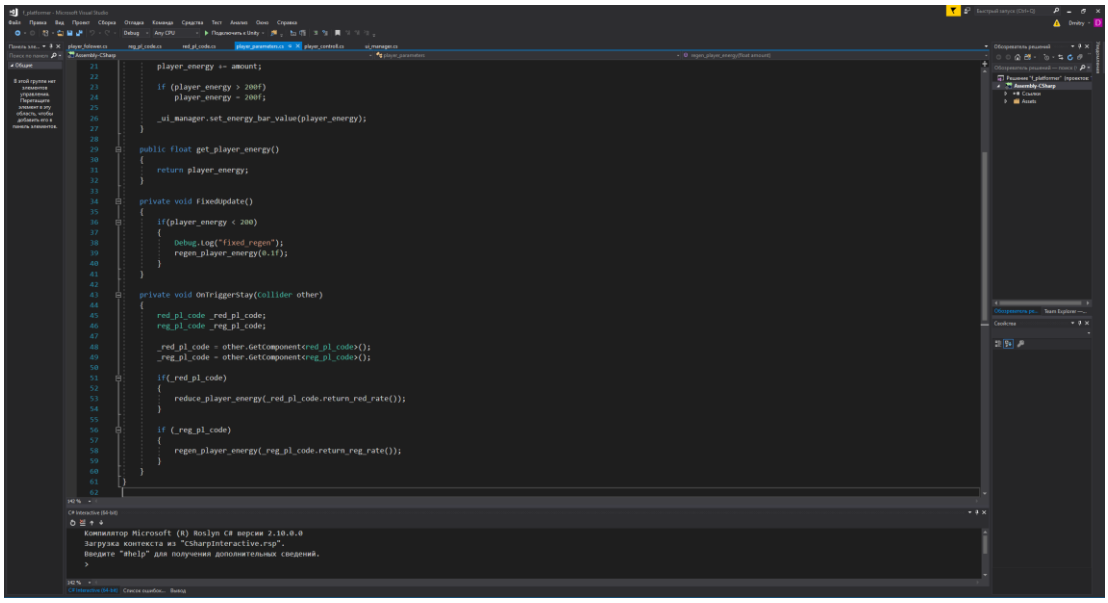


Рисунок 1.10. Вікно розробки Visual Studio проекту у Unity

Що стосується роботи з графікою, то потрібен графічний редактор. Графічний редактор – це програмне забезпечення, призначене для створення, модифікації та оптимізації графічних зображень. Він надає користувачам інструменти для створення ілюстрацій, редагування фотографій та інших видів графіки, включаючи кисті, олівці, спреї, штампи, інструменти виділення та багато іншого, а також можливості для коригування кольору, розміру, прозорості та інших параметрів зображення. Серед популярних графічних редакторів можна назвати Adobe Photoshop, GIMP, CorelDRAW та 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 боту для гри:

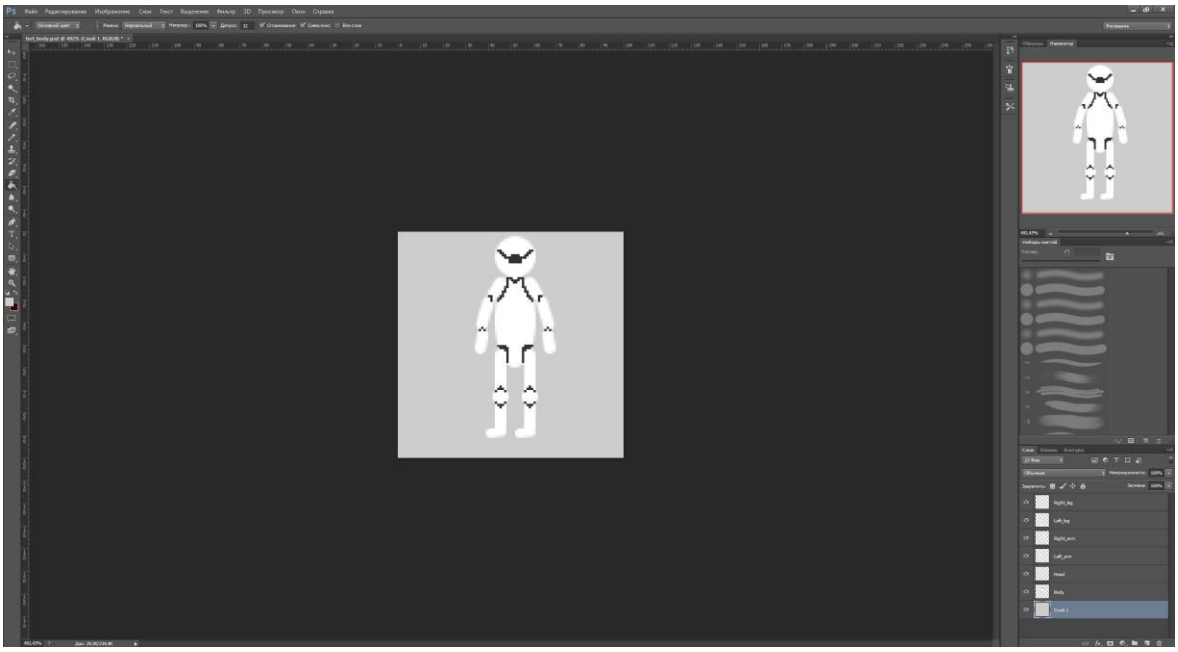


Рисунок 1.11. Використання Adobe Photoshop для створення боту для гри

Важливою частиною процесу розробки є також система контролю версій. Система контролю версій – це інструмент, який дозволяє розробникам відслідковувати зміни у вихідному коді та інших файлах проекту, порівнювати версії, відкочувати до попередніх станів та об'єднувати роботу різних учасників. Вона також забезпечує історію змін, що сприяє кращому розумінню процесу розробки та співпраці над проектом. До відомих систем контролю версій належать Git, Subversion та Mercurial.

У моєму випадку я використовую BitBucket та SourceTree. BitBucket – це веб-базована система контролю версій, яка є безкоштовною та інтегрована з багатьма іншими сервісами в рамках екосистеми Atlassian. SourceTree – це клієнтська програма, яка забезпечує графічний інтерфейс для роботи з системами контролю версій, що робить процес більш зручним і інтуїтивно зрозумілим. Ці інструменти є незамінними для збереження результатів розробки в онлайн-середовищі.

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. 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 характеристик та вмінь гравця

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 характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity». Виходячи з теми проекту, а також особливостей ігрового жанру, потрібно реалізувати характеристики для гравця для його більш варіативного проходження гри адвенчурі, а також створити систему вмінь для створення додаткових подій, або шляхів вирішення завдань. Реалізовані елементи мають бути створені в дусі жанру та і з урахуванням концепту гри.

Отже потрібно визначити потреби від жанру. Як вже було визначено раніше, адвенчури мають на увазі спокійний ігровий процес. Окрім того, важливою частиною проходження адвенчур є дослідження рівнів та виконання завдань – розв’язання головоломок. У нормальному випадку, в адвенчурі гравцю будуть даватись завдання, або створюватись припони на шляху, основані на виконанні якоїсь дії у ігровому світі, або присутності того, чи іншого предмету в інвентарі гравця. З моєї точки зору, можна трохи змінити цю концепцію і ввести у гру систему характеристик та вмінь, можливо у додаток до інвентарю. Розглянемо концепцію характеристик та вмінь окремо.

Характеристиками можна назвати значення персонажу гравця, які він буде мати на початку проходження адвенчури. Ці характеристики можуть задаватись гравцем спочатку гри, або покращуватись безпосередньо під час ігрового процесу під час подій, отримання предметів, тощо. Характеристики мають бути зрозумілими для гравця, тому повинні передавати свою сутність, яку відразу можна зрозуміти з її назви. У рольових іграх прийнято характеристики гравця іменувати параметрами здібностей людей, тому я використаю цю концепцію. В

поточній реалізації мають бути ураховані такі характеристики як:

- Сила;
- Спритність;
- Сприйняття;
- Вдача.

З переліку назв можна відразу визначити, яка характеристика за що відповідає. Сила відповідає за здібність персонажу виконувати силові навантаження – силове відкриття дверей, кидки, зсуви з місця. Спритність за можливість проходження подій основою яких я спритність, використання канатів, настельних крюків, тощо. Сприйняття відповідає за здібність персонажу орієнтуватись у просторі та далеко бачити, багато помічати. Вдача є універсальною характеристикою, яка сприяє на всі ігрові події – чим вище, тим краще.

Система вмінь має схожу ідею, як характеристики, але дає якісь ситуативні бонуси, що виконують при деяких умовах. Таким чином, можна створити сильного персонажа із середньою спритністю, а як вміння дати бонус до вирішення завдань – як бонус до вдачі. З іншого боку, потрібно дуже обережно створювати вміння, оскільки вони можуть створити дизбаланс у ігровому процесі та викликати у гравця ідею використовувати вміння не з точки зору відіграння ролі, а з точки зору практичності. Вміння можна реалізовувати таким чином, щоби гравець бачив назву, та описання. Наприклад вміння можуть мати наступний вигляд:

- Особливі відношення з Фортуною. Бонус до вдачі, але інколи вона істерить;
- Міцний корпус. Зменшує пошкодження при невдалому силовому виконанні, але висота, з якої можна безпечно впасти зменшується вдвічі;
- Шосте почуття. Події поряд з вами перестають бути прихованими.

Можна створювати велику кількість подібних вмінь, що будуть створювати унікальні ігрові ситуації та будови ігрового персонажу.

Маючи смислову концепцію, потрібно зрозуміти, як її донести до гравця.

Очевидним виходом є створити вікно з параметрами персонажу на початку гри, або при створенні гравця. У випадку із поточним проектом, при кожній поразці гравець зможе виставляти характеристики по новому перш ніж почати рівень знову. На рисунку 1.12 зображено приклад панелі розподілення характеристик та вмінь.

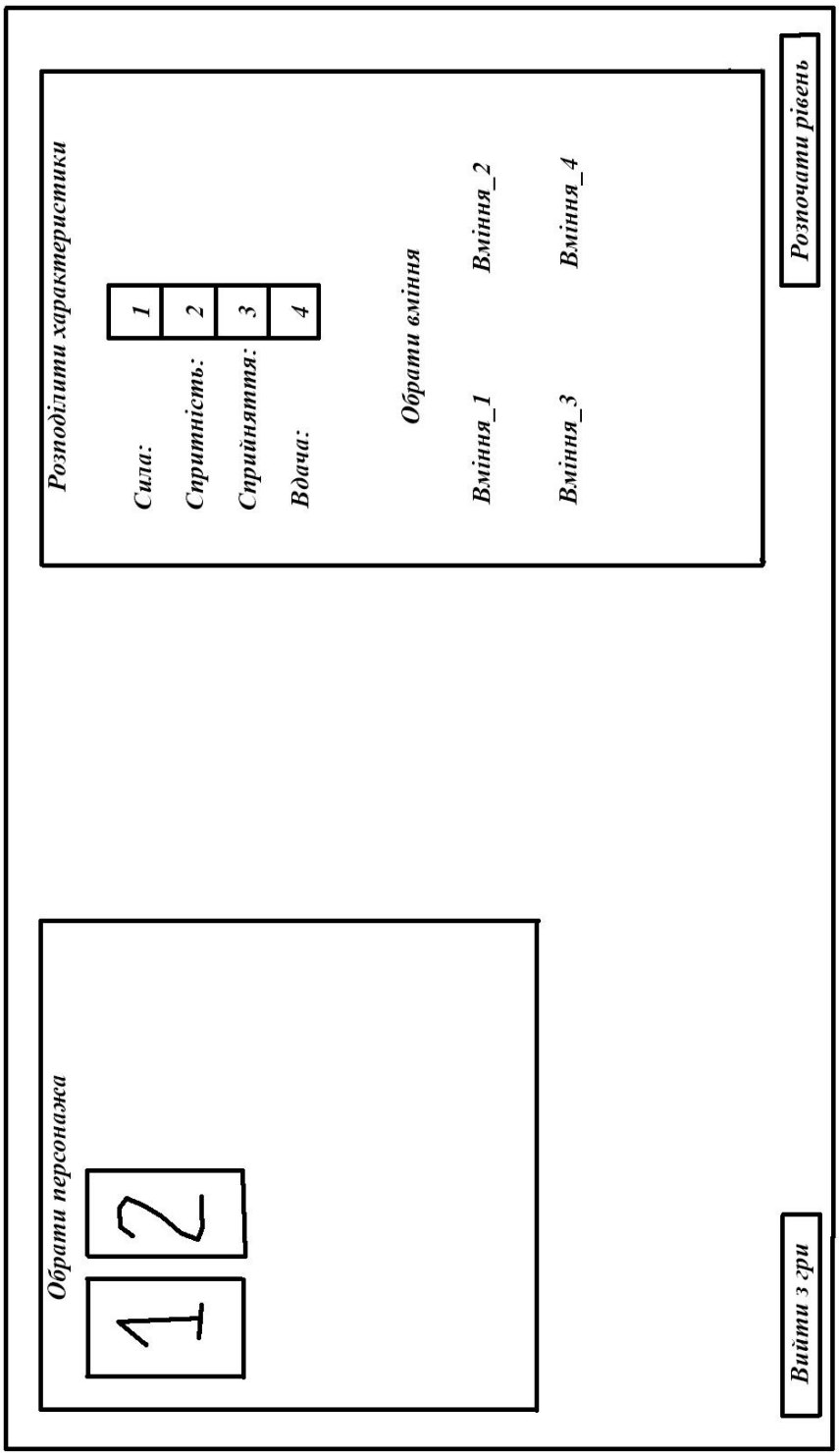


Рисунок 1.12. Приклад панелі розподілення характеристик та вмінь

Потрібно буде реалізувати подібну панель. На ній створити всі необхідні кнопки для збільшення, або зменшення характеристик, обрання чи відключення вмінь. Окрім того, на панелі можна обирати персонажів, але спочатку гри буде доступний лише один. Також мають бути реалізовані кнопки для виходу з гри та початку проходження рівня.

Вважаю, що смислова та візуальна проробка системи характеристик та вмінь персонажу гри проведена на достатньому рівні. Маючи поточний концепт, можна перейти до проектування безпосередньо систем, з програмної точки зору та опором на обраний ігровий програмний рушій Unity.

1.5 Проектування системи характеристик та вмінь гравця

1.5.1 Проектування першого рівня системи характеристик та вмінь

Перед початком реалізації розробляємих систем, згідно теми дипломного проекту, потрібно виконати їх попереднє проектування. Це дуже важлива частина розробки, оскільки вона закладає основу в процес реалізації та допомагає уникнути багатьох проблем на етапі реалізації. Професійні команди розробки проводять проектування будь-яких функційних елементів програмного забезпечення перед реалізацією не тільки для того, щоби мати уявлення про процес створення модулів програми, але й для орієнтації співробітників команли та підтримки коду у разі виникнення проблем із ним.

Отже спочатку потрібно зрозуміти, як саме буде виконуватись механіка систем характеристик та вмінь у грі. Для розуміння цього процесу потрібно зрозуміти сам принцип роботи ігрового програмного рушія Unity та важливих для реалізації теми диплому елементів.

Система характеристик та вмінь можна поділити на два рівні. В першому рівні, гравець розподіляє параметри характеристик та вміння, які хоче обрати для ігрового процесу. Також, як було вказано в концепції, гравець має змогу обрати варіанти персонажу, втім на початку він лише один, тому його проробкою можна знехтувати. На рисунку 1.13 зображено схему роботи системи характеристик та вмінь на початку гри:

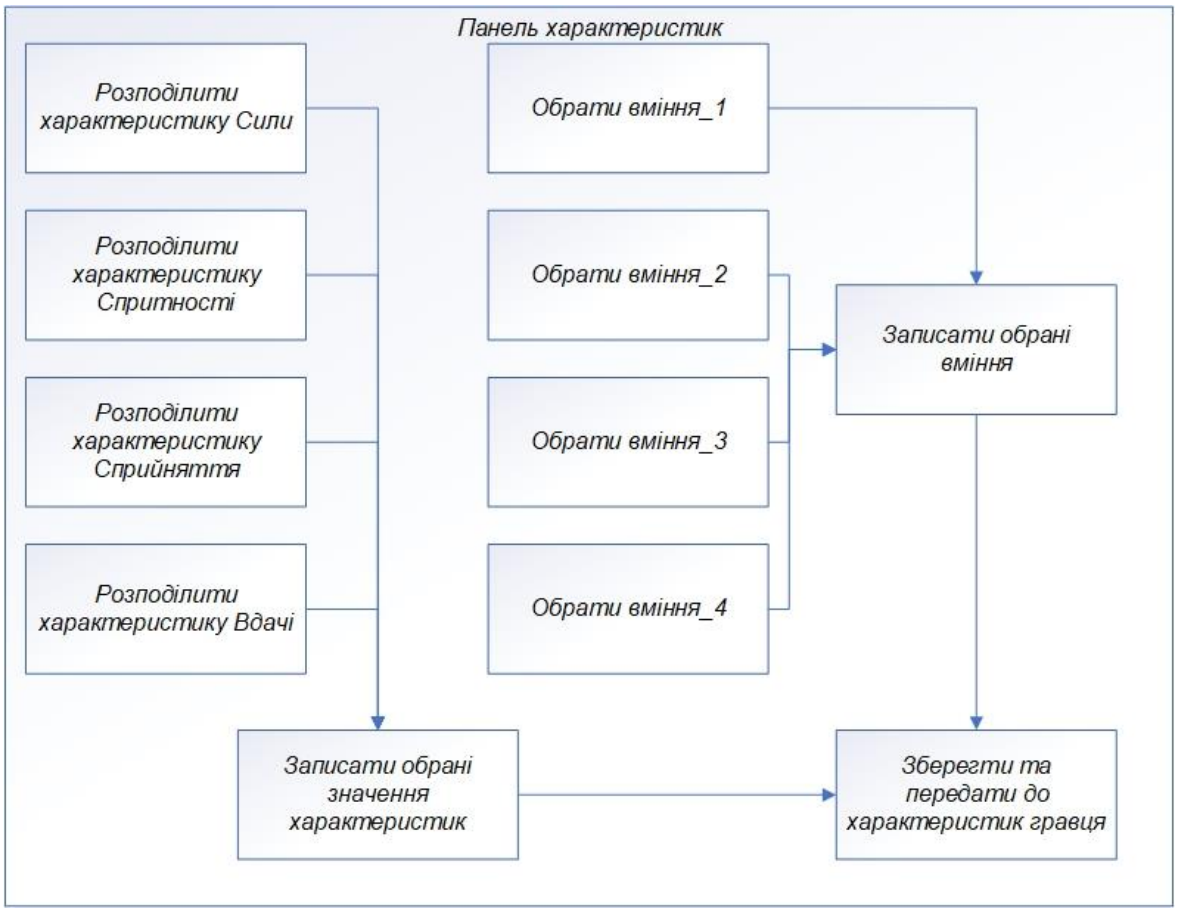


Рисунок 1.13. Схема роботи системи характеристик та вмінь на початку гри

Як можна побачити, гравець спочатку виконує розподілення характеристик для персонажу, обирає кількість вмінь, наприклад одне. Після встановлення значень характеристик, ці значення автоматично записуються для подальшої передачі. Такий самий принцип і для вмінь – коли гравець обирає вміння, він автоматично передає перелік обраних у секцію до передачі. Кінцевим етапом буде натискання на кнопку початку гри, яка збереже та передасть обрані характеристики та вміння далі до ігрового процесу.

Також, зі схеми можна побачити, що на даному етапі основна робота із характеристиками та вміннями з боку гравця виконується за допомогою інтерфейсу, тому потрібно буде розробити код інтерфейсу, який би зміг отримувати та передавати значення характеристик та вмінь.

1.5.2 Проектування другого рівня системи характеристик та вмінь

Другий рівень системи характеристик та вмінь розташований безпосередньо під час ігрового процесу. На рисунку 1.14 зображено схему роботи характеристик та вмінь під час ігрового процесу:

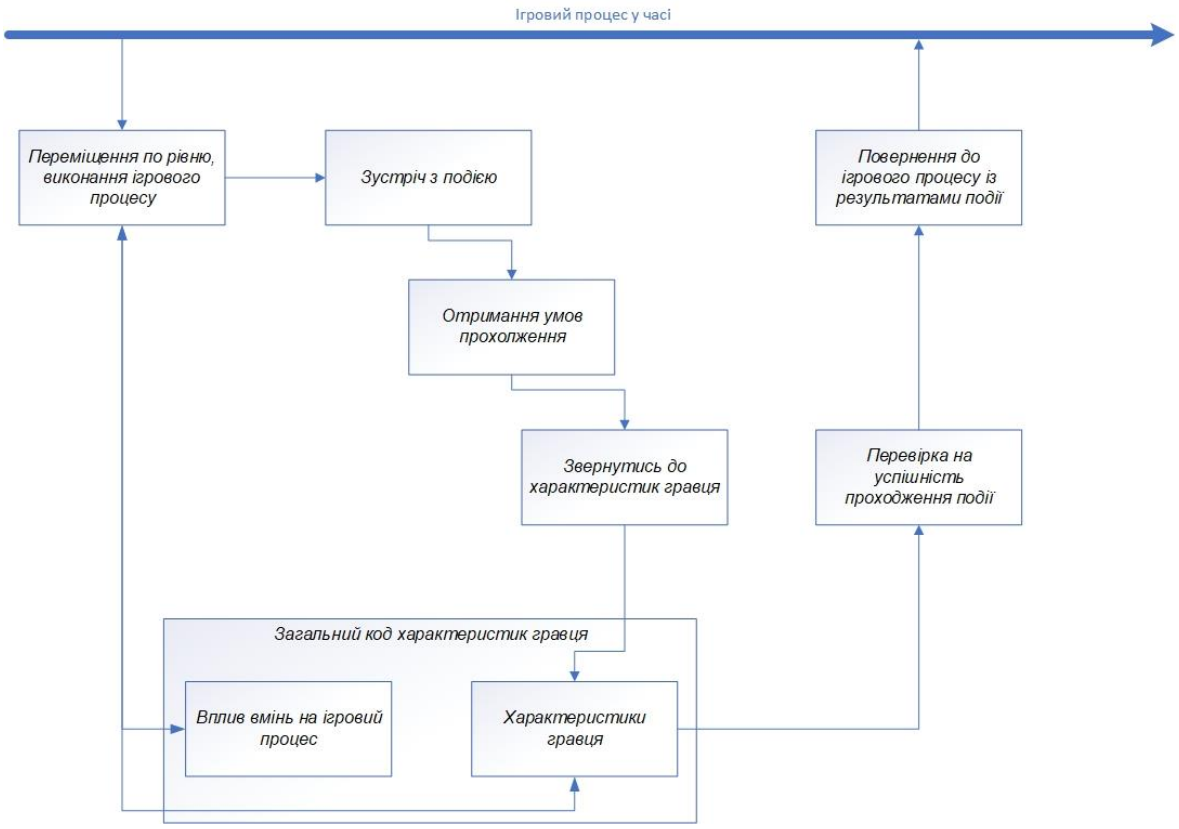


Рисунок 1.14. Схема роботи характеристик та вмінь під час ігрового процесу

Потрібно розуміти, що Ігровий процес – це цикл. Будь-яка гра виконується у циклі. Це нерушиме правило ігрового проектування. Якщо гра припиняє виконуватись по циклу, вона «зависає», що руйнує ігровий досвід повністю. Циклічна сутність ігор має коріння від того, що ігровий процес повторюється раз за разом. Кожну мить існування процесу гри, вона виконує розрахунки и до кінця цієї миті-кадру добігає свого кінця процесу, чим більше таких пробігів кадрів за секунду, тим більша частота кадрів.

Отже на рисунку 1.14 можна побачити, як ігровий процес виконується у часі. Паралельно із ним існує деякий Модуль Характеристик гравця. Цей модуль зберігає в собі характеристики гравця, його вміння та може їх передавати до ігрового процесу. Зі схеми видно, що під час переміщення можуть бути створені умови для звернення до характеристик гравця, наприклад, розпочнеться якась подія, що базована на випробувані характеристик. В цьому разі отримуються умови проходження, виконується звернення до Модуля Характеристик гравця, безпосередньо до характеристик вказаних у вимогах. Після цього виконується перевірка на успішність виконання умов події, реакція на завершення події, після

чого виконується повернення до ігрового процесу. Тут потрібно відмітити, що до перевірки на успішність виконання події також долучаються бонуси від вмінь, якщо такі є.

1.5.3 Проектування Модулю Характеристик гравця

Модуль Характеристик гравця, окрім реакції на події, своїми вміннями та характеристиками може безпосередньо впливати на ігровий процес. Наприклад те ж сприйняття збільшує дальність відрисовки ліхтарика гравця. Розглянемо будову Модуля Характеристик гравця, вона зображена на рисунку 1.15:

Рисунок 1.15. Будова Модулю Характеристик гравця

Модуль Характеристик гравця складається з змінних та посилань на зовнішні об’єкти та компоненти. У своєму складі він має метод ініціалізації стартових змінних. Також, оскільки може бути така ситуація, коли гравець не може пройти рівень із поточними характеристиками, то можна виконати самознищення та повернутись у меню розподілу характеристик. Також тут реалізоване отримання характеристик та вмінь з панелі розподілу характеристик та вмінь. Тут є метод, що змінює параметри «пристроїв гравця», наприклад ліхтарика в залежності від характеристик. Нарешті комплекс методів для передачі

характеристик до зовнішнього об’єкту, а також методи встановлення параметрів характеристик.

1.5.4 Проектування Модулю подій

Вважаю також доцільним проілюструвати будову Модулю подій. Це важливо з точки зору можливості не тільки тестування, але й самої суті ігрового процесу із системою характеристик та вмінь гравця. Схема будови модулю подій зображена на рисунку 1.16:

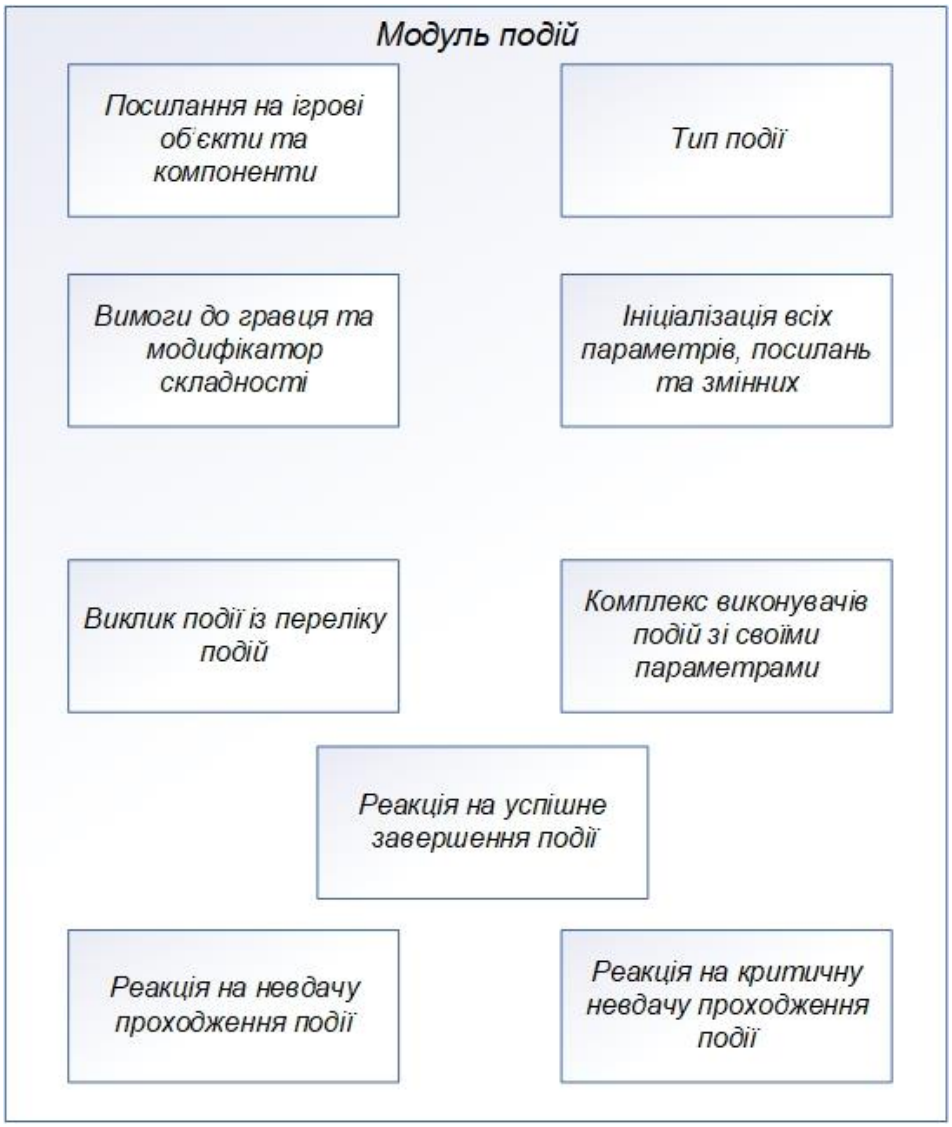


Рисунок 1.16. Схема будови Модулю подій

Модуль подій є дуже великим модулем, оскільки є універсальним блоком. Річь в тому, що події можуть бути різного типу. Виходячи з цього, я зіткнувся із дилемою. Можна створити велику кількість кодів для всієї множини подій, або створити один код для всіх варіантів подій. Оскільки події у здебільшого мають

однакову будову та функціонал, я вирішим створити один великий код для всіх подій. Розглянемо його будову, яка зображена на рисунку 1.16. Як можна бачити, цей модуль, як і Модуль Характеристик гравця починається із множин дій по оголошенню змінних, а також встановлення типу події, вимог до гравця, модифікатора складності і, нарешті початкової ініціалізації всієї кількості цих значень, змінних та посилань.

Наступним умовним блоком коду, який можна побачити на рисунку 1.16, є частина по роботі із подією. Тут можна побачити такі методи, як виклик події з переліку подій, великий комплекс методів по виконанню подій, кожна з яких отримує свої параметри необхідні для виконання. В кінці варіації реакцій на завершення розрахунку вирішення подій. Події можуть бути вирішені успішно, невдало та із критичною невдачею. Кожна із цих варіацій призводить до своїх наслідків. Наприклад критична невдача віднімає здоров’я у гравця.

Побудова даних проектів основних ігрових елементів дозволить перейти до реалізації запланованих елементів ігрового додатку. У подальшому такі схеми будуть використовуватись для розробки інших елементів гри адвенчури.

1.6 Реалізація елементів системи характеристик та вмінь гравця

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.

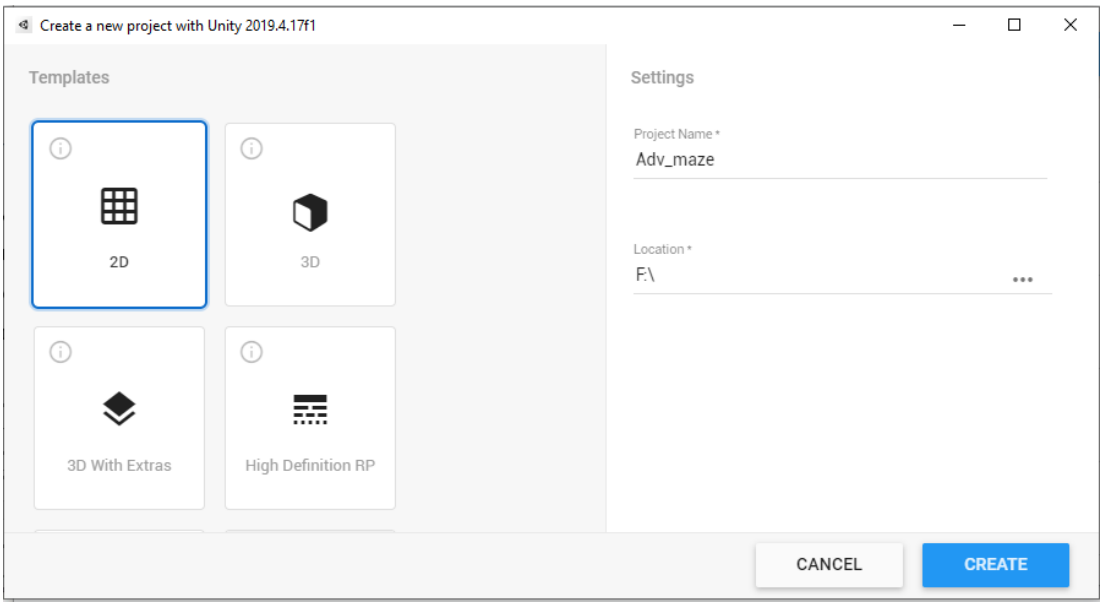


Рисунок 1.17. Вікно створення проекту у UnityHub

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».

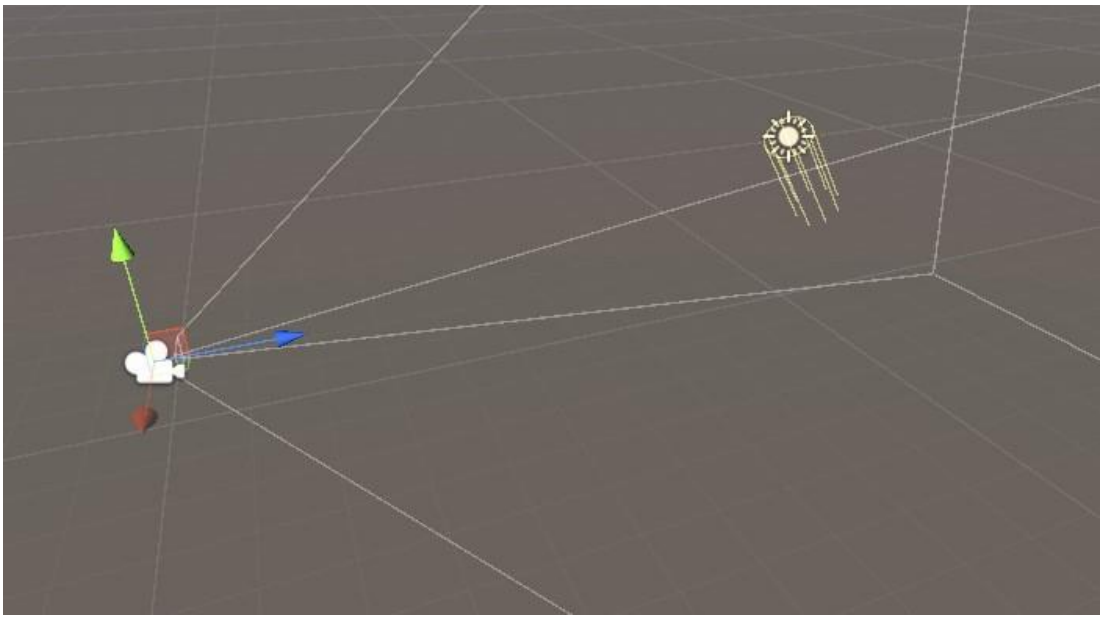


Рисунок 1.18. Стартові об’єкти в новому проекті Unity

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 1.18 можна побачити ігрову сцену із зміненими характеристиками фону камери гри:

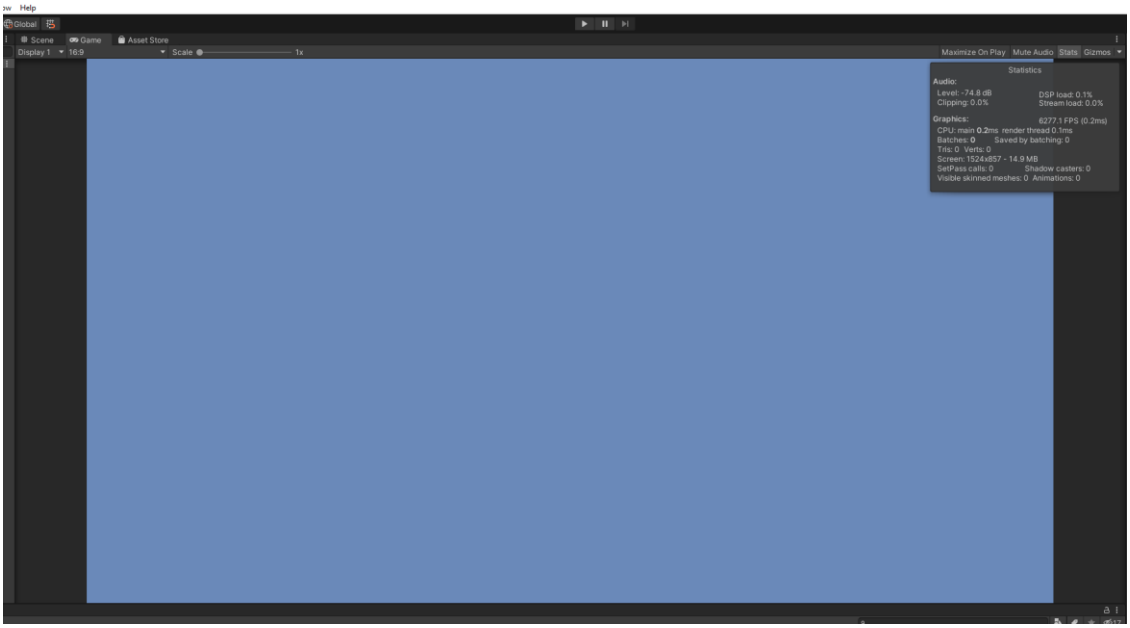


Рисунок 1.18. Ігрова сцена із зміненими характеристиками фону камери гри

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. На рисунку 1.19 можна побачити графічні елементи для створення рівнів.

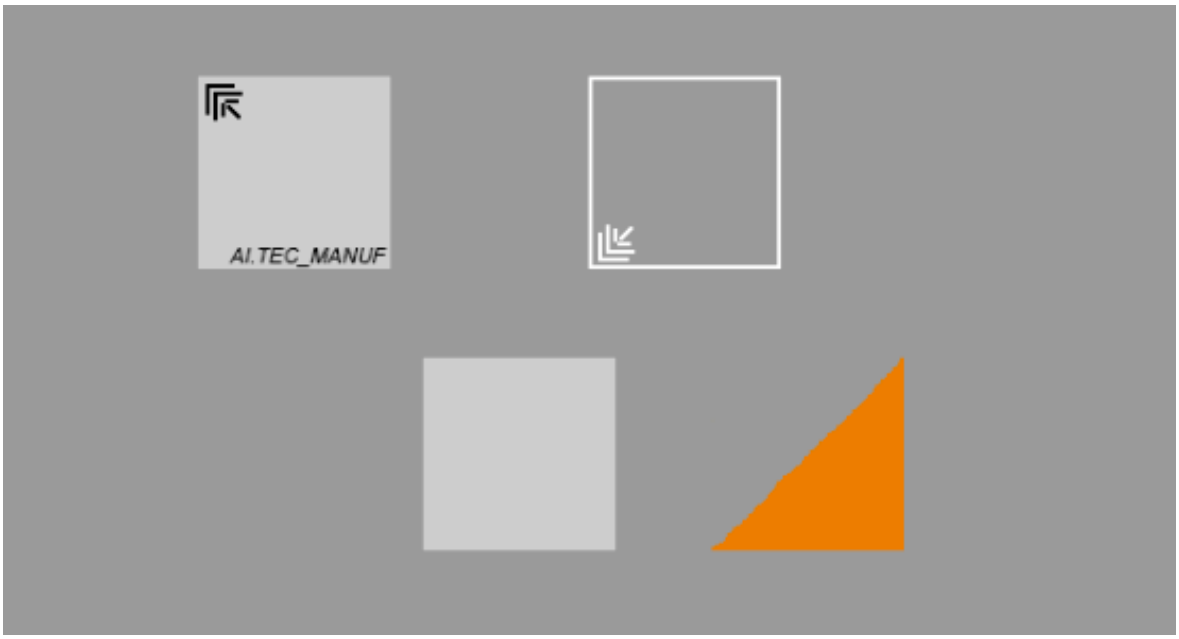


Рисунок 1.19. Графічні елементи для створення рівнів гри

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 1.20.

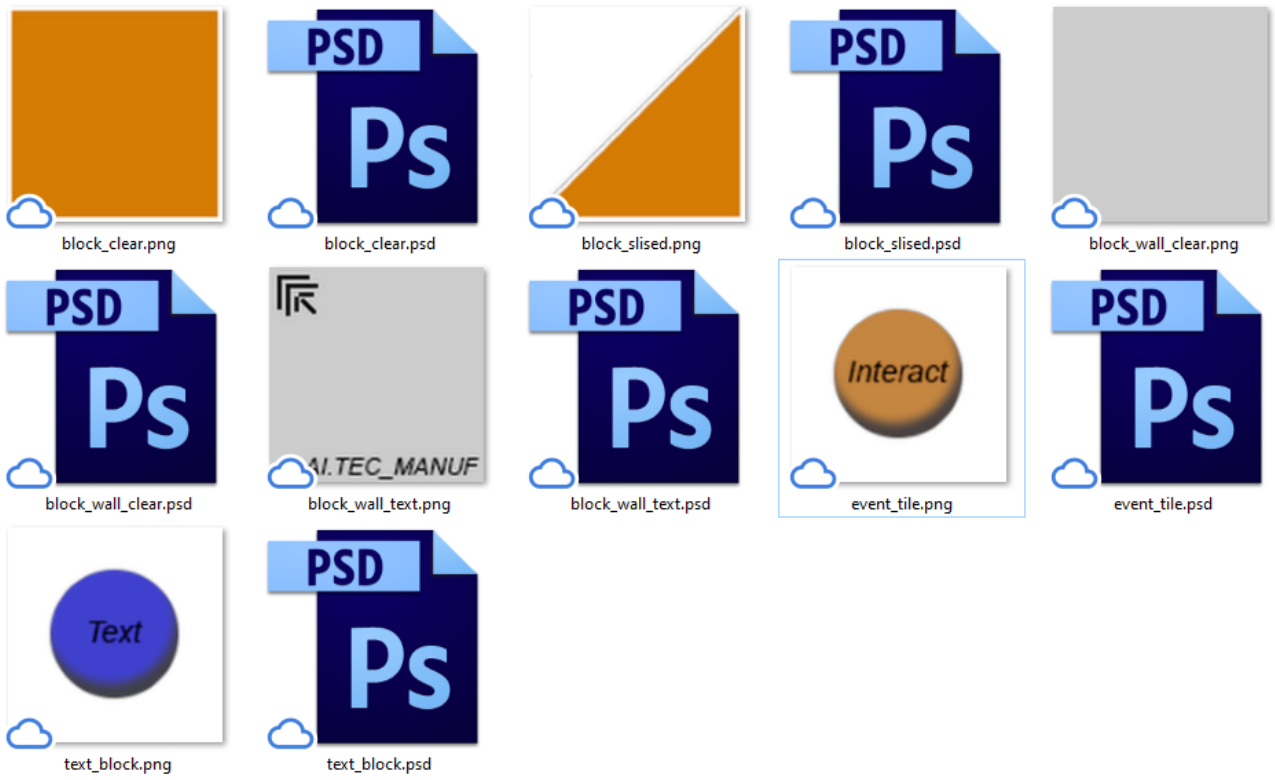


Рисунок 1.20. Всі спрайти реалізовані для проекту гри

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.

Також саме зборку рівня приходиться виконувати в першу чергу через те, що маючи готовий рівень, можна тестувати на працездатність окремих частин коду гри безпосередньо під час їх розробки. Для тестування роботи системи характеристик на вмінь буде достатньо невеличкої частини рівню. На рисунку 1.21 зображено зібраний тестовий рівень гри адвенчури:

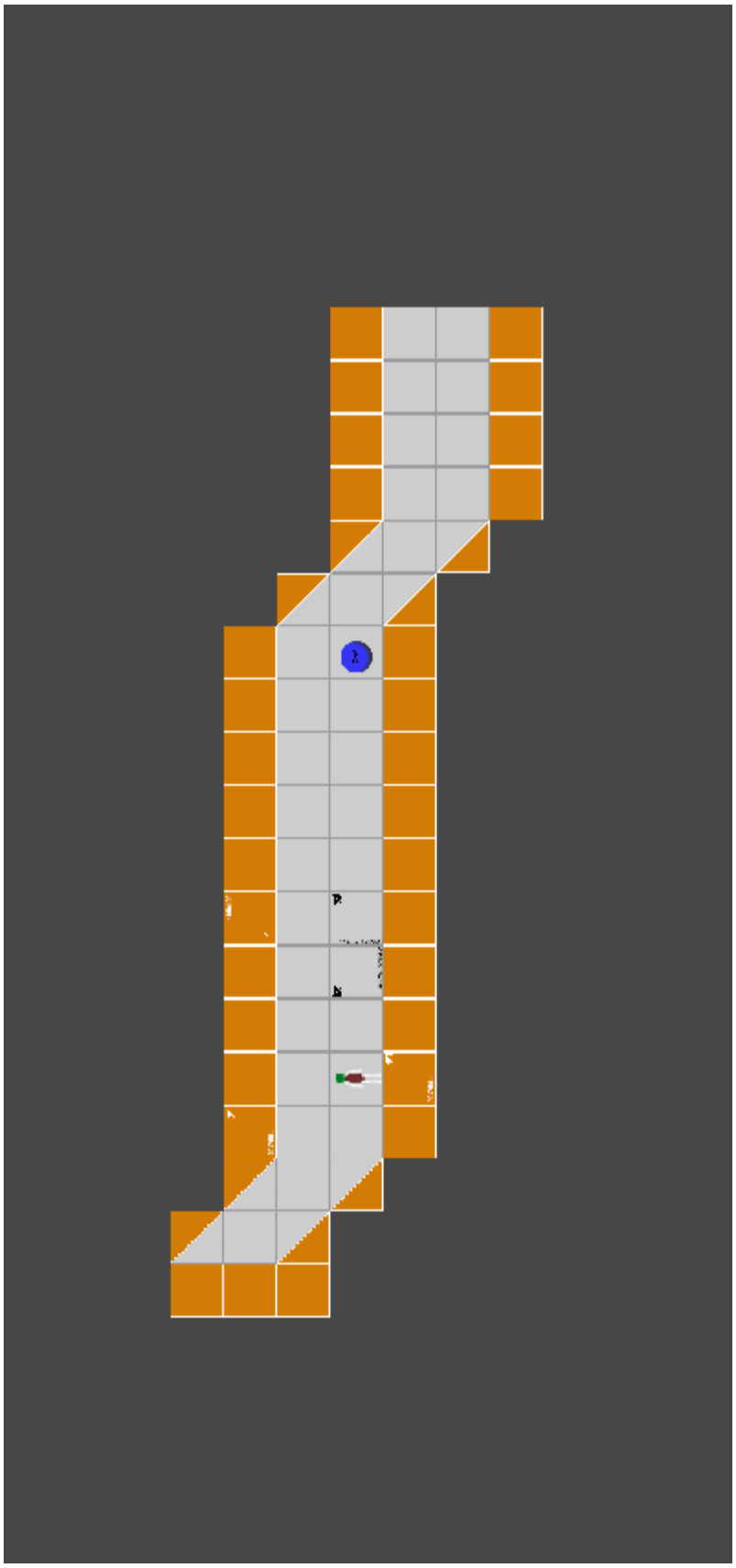


Рисунок 1.21. Тестовий рівень гри адвенчури

Також потрібно реалізувати ігровий інтерфейс, який відповідає за головне меню обрання характеристик. Він реалізовується за допомогою полотна Unity –

Canvas. Це система для створення інтерфейсів із заздалегіть заготовлених елементів, як кнопки, панелі, зображення, текстові поля. Створене меню розподілу характеристик та вмінь можна побачити на рисунку 1.22:

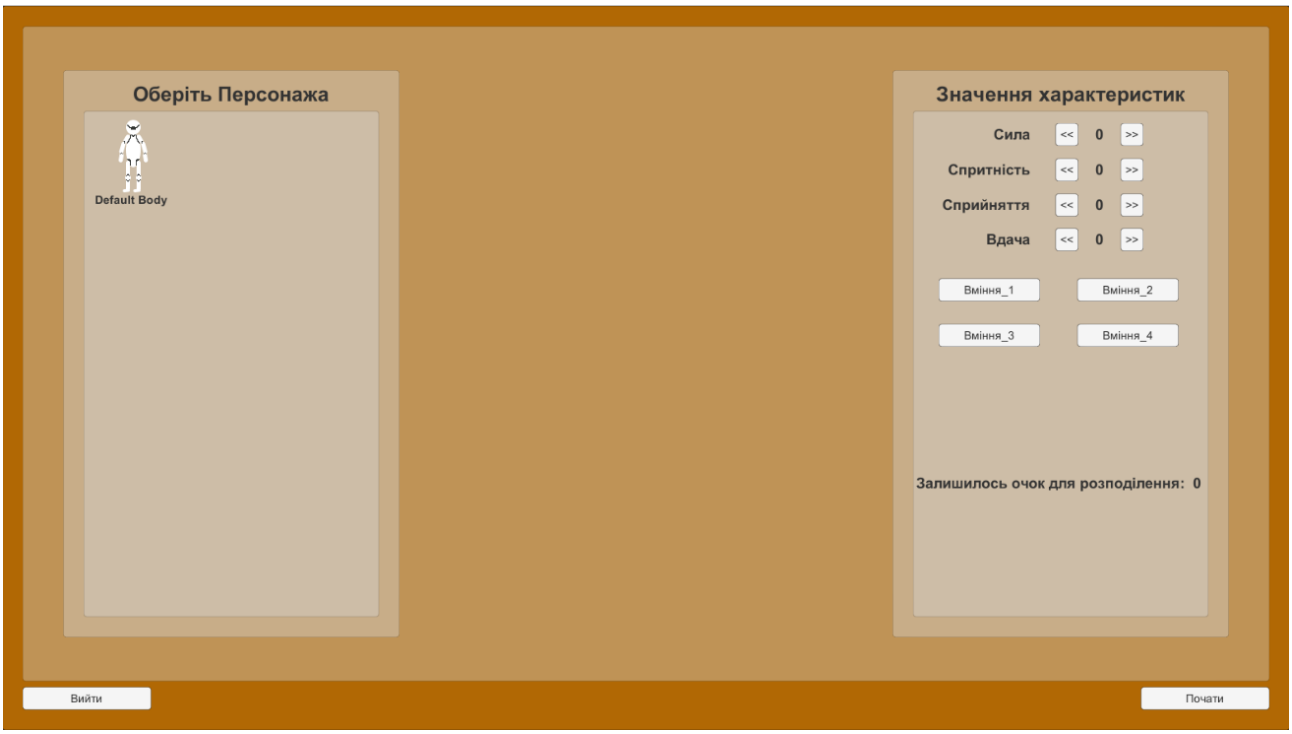


Рисунок 1.22. Меню розподілу характеристик та вмінь

Коли розміщені всі ігрові об’єкти, елементи рівня, можна переходити безпосередньо до реалізації систем теми дипломного проектування, а саме системи характеристик та вмінь гравця. Отже комплекс модулів визначено у частині проектування. Необхідно створити скрипт характеристик гравця player_parameters.cs та event_script.cs а також комплекс скриптів для меню розподілу характеристик та вмінь.

Почну реалізацію з меню. В першу чергу потрібно створити два скрипта, які будуть контролювати всю кількість елементів цього меню. Перший скрипт parameters_points_logic.cs виконує роль контролера можливості збільшувати характеристики та обирати нові вміння. Цей код повинен не давати гравцю збільшувати характеристики більше ліміту кількості очок для розподілення. Нижче приведено код цього скрипта:

```
[SerializeField] body_parameters _body_parameters;
[SerializeField] Text _points_to_spare_text;

public int value = 0;
```

```
public bool can_modify = true;
public bool can_increase_points = true;

private void Start()
{
   value = _body_parameters.statespoints_value;
   _points_to_spare_text.text = value.ToString();
}

public void take_new_body_statespoints(int states_point_value)
{
   value = states_point_value;
   _points_to_spare_text.text = value.ToString();
}

public void show_new_value()
{
   _points_to_spare_text.text = value.ToString();
}
```

Представлений код спочатку отримує посилання на параметри обраного персонажу. Саме в цих параметрах вказано, скільки очок для розподілу доступно гравцю. Далі отримую доступ до текстового поля кількості очок для розподілення і, послідовно, кількість очок, та можливість збільшувати характеристики, або обирати вміння. Наступним кроком йде метод ініціалізації в методі Start(), який викликається в момент створення ігрового об’єкту та поточного скрипта. Наступні методи отримують за командою нові параметри для розподілення очок, а також встановлюють нові значення для кількості очок на інтерфейсі.

Наступним скриптом для реалізації є parameters_logic.cs. Цей скрипт дозволяє розподіляти очки серед характеристик та змінювати їх значення. Саме цей код зберігає значення характеристик та вмінь для подальшої передачі до характеристик гравця. Нижче приведено фрагмент коду цього скрипта:

```
public void increase_value()
{
   if (_parameters_points_logic.value > 0)
   {
      if (value < 10)
      {
         ++value;
```

```
            --_parameters_points_logic.value;
            _parameter_text.text = value.ToString();
            _parameters_points_logic.show_new_value();
        }
        else
            return;
    }
    else
        return;
}

public void decrease_value()
{
    if (value > 0)
    {
        --value;
        ++_parameters_points_logic.value;
        _parameter_text.text = value.ToString();
        _parameters_points_logic.show_new_value();
    }
    else
        return;
}
```

Вище представлено два методи які збільшують, або зменшують значення характеристики. По своєму функціоналу, перший метод перевіряє, чи значення кількості очок для розподілення більше 0, після чого перевіряє, чи кількість очок у характеристики, яку намагаються збільшити, менше за 10. 10 – це максимальне значення характеристики. Якщо значення менше, то воно збільшується на 1, а кількість очок зменшується на 1. У противному випадку, метод закінчується нічого не змінюючи. Принцип роботи другого модулю схожий на перший, але перевіряє чи більше поточне значення характеристики, яку змінюють за 0. Якщо так, то зменшує її, та збільшує кількість очок для розподілення. У противному випадку нічого не змінюється.

Наступним кроком буде реалізація 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;
```

Як можна побачити з приведеного коду, спочатку створюється посилання на код, який буде викликати меню розподілу характеристик гравця після самознищення. 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.

Щодо реалізації безпосередньо збереження очок характеристик, та зміни параметрів предметів, то це виконується у двох окремих методах, код яких приведено нижче:

```
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 рівня та отримує як аргументи характеристики для гравця отримані у меню розподілу характеристик гравця. 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 та вмінь. Останнім кодом, що потрібно переглянути в цьому скрипті, є код повернення значення характеристики. Це потрібно у разі проходження події. Код приведено нижче:

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

Наступним кроком у реалізації мети дипломного проектування є реалізація подій у скрипту event_script.cs. Цей скрипт, як було видно із частини проектування дуже об’ємний, тому я розгляну лише його частини. Нижче приведено код виклику подій:

```
public void event_caller(int player_strength, int player_dexterity, int player_perception, int player_luck)
    {
       if (strength_requirements > 0)
          player_strength_to_send = player_strength;
       if (dexterity_requirements > 0)
          player_dexterity_to_send = player_dexterity;
       if (perception_requirements > 0)
          player_perception_to_send = player_perception;
       if (luck_requirements > 0)
          player_luck_to_send = player_luck;
       if (lader_event)
          lader_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
    }
```

На перший погляд код може бути складним, але це не так. Даний метод – event_caller – викликається із зовні і до нього як аргументи передаються значення характеристик гравця. Потім виконується передача значення характеристик до змін, які використовуються для подальшої передачі характеристик гравця у рамках події. Разом із тим виконується перевірка, чи більше значення характеристики за 0. Далі виконується перевірка того, яку подію зараз викликають.

Тут потрібно загострити увагу на те, що цей код прикріплюється до ігрових об’єктів подій. Тобто у кожного ігрового об’єкту події буде своя версія цього коду, а значить, у кожній події можуть бути обрані різні властивості події. Тому і виконується перевірка. Якщо є збіг по обраній події у ігровому об’єкті події, то викликається метод для обробки події та в нього передаються ті самі характеристики для подальшої передачі у рамках події. У випадку даного фрагменту, він був зменшений із виліком всіх подій гри. Нижче приводжу код обробника події:

```
void lader_event_processor(int strength_req_val, int dexterity_req_val, int perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val){
    int pass_value = 0;
    int player_cube_throw = 0;
    int player_result = 0;
```

```
        pass_value = strength_req_val + dexterity_req_val + perception_req_val +
luck_req_val + dificulty_modificator;
        player_cube_throw = Random.Range(1, 6);
        player_result = player_cube_throw + pl_strength_val + pl_dexterity_val +
pl_perception_val + pl_luck_val;
        if (player_result >= pass_value)
        {
            player_pass_event();
        }
        else if (player_result < (pass_value / 2))
        {
            player_critical_fail(pass_value, player_result);
        }
        else
        {
            player_fail_event(player_result, pass_value);
        }
    }
```

Цей метод отримує у свої параметри інформацію про вимоги для вдалого проходження події, після цього отримує параметри про характеристики гравця. У своєму тілі метод розраховує рівень складності, виходячи із тих аргументів, які він отримав. Потім виконує кидок куба. Успішність завершення залежить від «результату гравця» – суми кидка куба, та характеристик гравця, які були передані. Якщо ця сума більша за умову проходження, тоді викликається метод успішного завершення. У іншому випадку, якщо значення результату гравця у двічі менше умови проходження, то викликається критичний провал. В іншому випадку просто невдача. Методи успішності викликають повідомлення та виконують ті, чи інші дії, в залежності від налаштування події.

Тепер гравець має змогу використовувати систему характеристик та вмінь. Важливо розуміти, що поточна реалізація гри не є кінцевою, але реалізація системи характеристик та вмінь реалізована таким чином, що її архітектура дозволяє у майбутньому реалізовувати нові і нові механіки роботи із характеристиками та вміннями гравця.

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

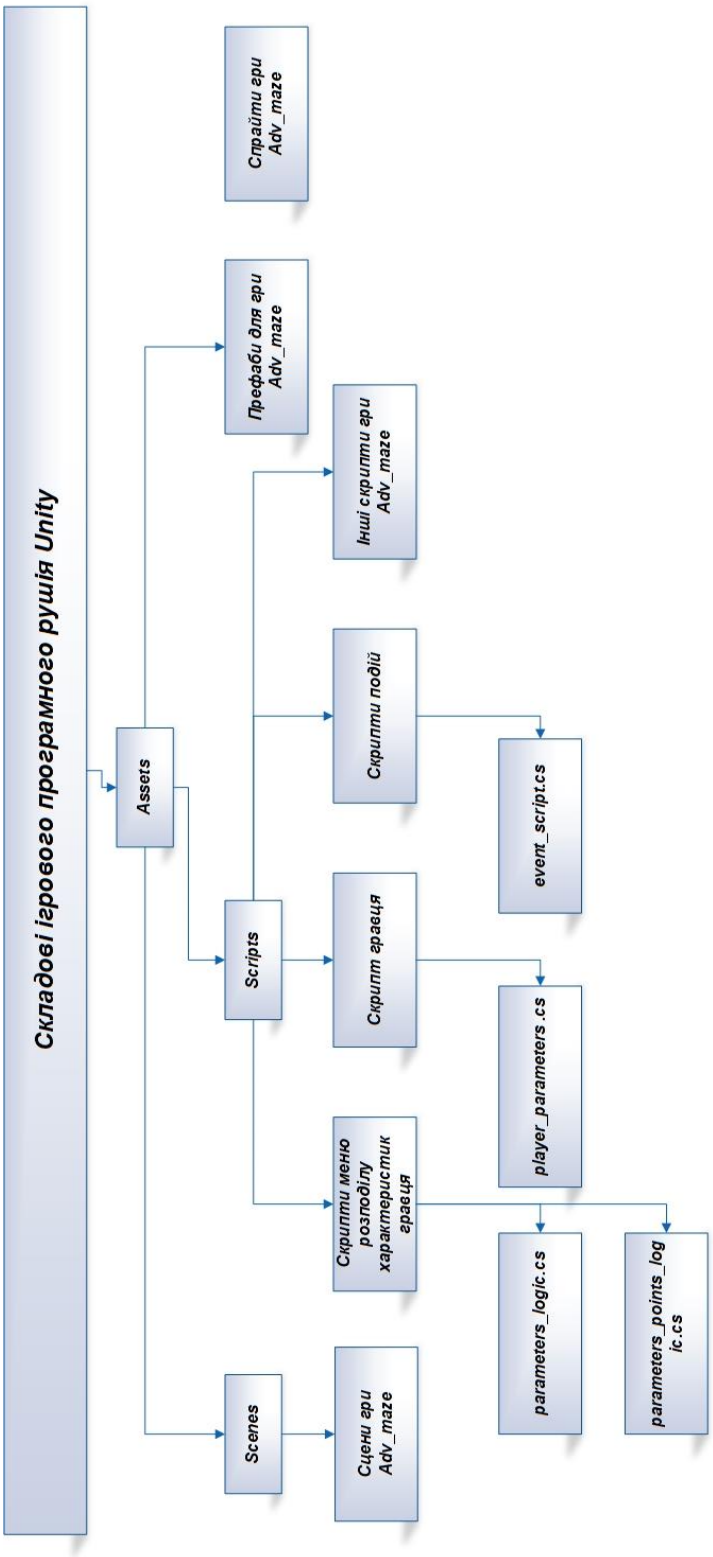


Рисунок 1.23. Розміщення файлів проекту по підкаталогам розділу Assets

1.7 Тестування працездатності реалізованих елементів

Тестування та налагодження є важливою частиною розробки будь-якого програмного продукту. Для ігрових проектів тестування є ще більш важливим процесом, оскільки воно дозволяє знаходити помилки, які не є очевидними. Слід

розуміти, що в ігрових проектах баги можуть викликати візуальні або внутрішні проблеми. Перше легко знайти, оскільки візуальні артефакти або помилки розміщення об’єктів швидко виявляються під час гри. Інша справа — баги програмного забезпечення, які візуально не ідентифікуються під час гри. Іноді такі помилки призводять до часткових порушень ігрового процесу, а в окремих випадках – до повної неможливості пройти гру, оскільки можуть вплинути на послідовність механіки, пов’язаної з переходом на інший рівень або запуском ключових сценаріїв в історії.

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 можна побачити процес тестування гри. В ході тестування було виявлено ряд помилок у роботі скриптів по передачі інформації від меню розподілу характеристик гравця до модуля характеристик гравця. Ці помилки були викликані відсутністю посилання у одному із полів скрипту меню розподілу характеристик гравця.

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 характеристик та вмінь. Було перевірено редагування та передачу характеристик з меню розподілу очок характеристик, безпосередньо до ігрового процесу. Для перевірки коректності роботи, було створено код подій, який виконується з різним результатом, виходячи із характеристик гравця.

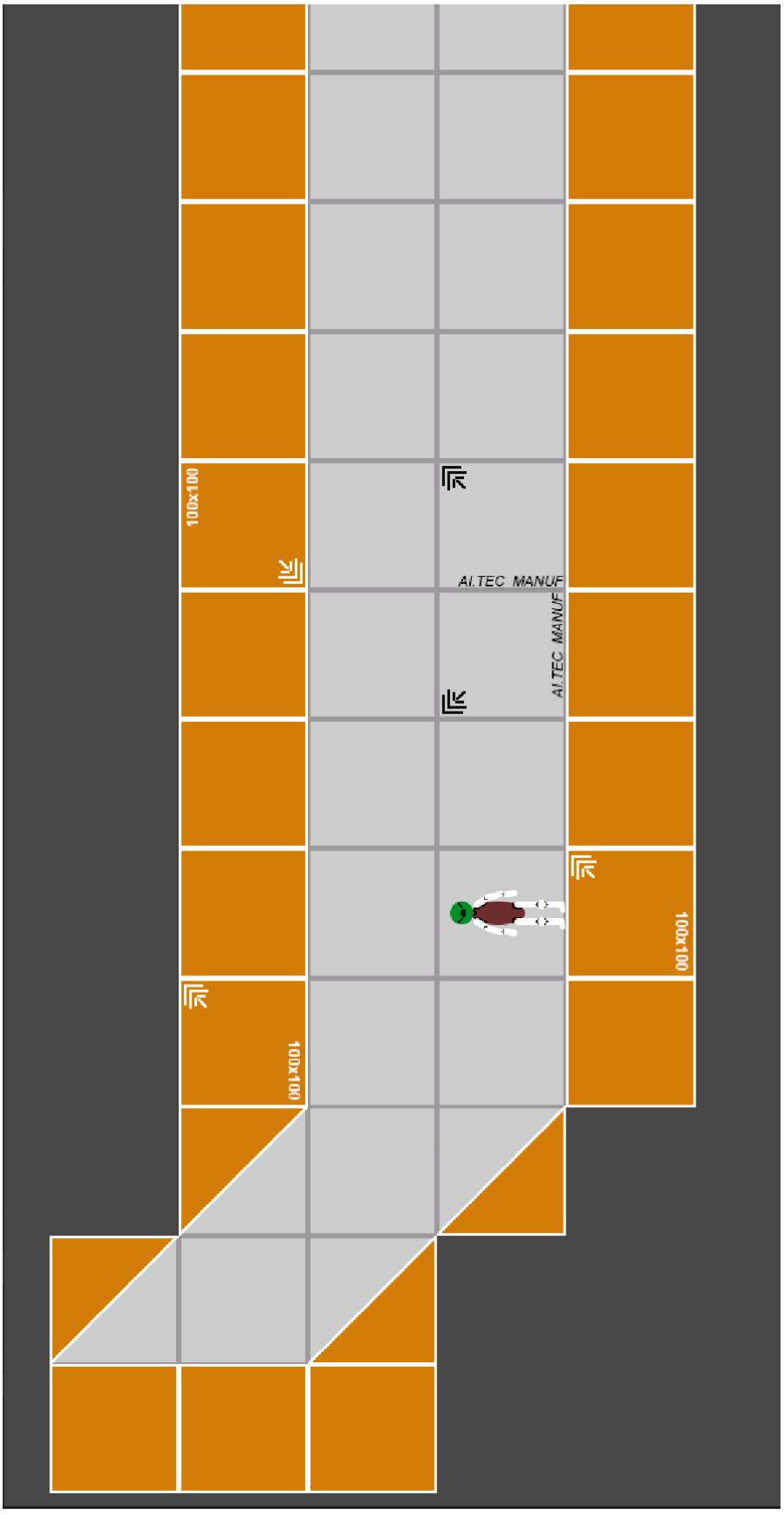


Рисунок 1.24. Процес тестування гри

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) (2.4)

Трудомісткість розробки технічного проекту

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

Під час виконання дипломного проектування, мною було виконано розробку системи характеристик та вмінь гравця в грі жанру адвенчура. Виконання проекту розпочалося з розробки концепції того, як можна ввести систему характеристик та вмінь у гру для того, щоби зробити ігровий процес більш різноманітним.

Проектування та розробка системи складалась з виділення модулів характеристик гравця, модуля самого гравця, його вмінь та інших частин системи. Ці модулі були спроектовані таким чином, щоби можна було у подальшому вносити в них зміни, які потрібно було б лише реалізувати в грі на рівні інтерфейсу.

Реалізація системи була зведена до реалізації проектів модулів характеристик гравця та його вмінь. У подальшому створений код був зв’язаний із ігровими об’єктами, які відповідали за відповідний ігровий функціонал, а також з елементами інтерфейсу. Робота з інтерфейсом була потрібна, оскільки характеристики та вміння гравця відображаються лише у панелі персонажу, яким в поточний час грає користувач. Використання мови програмування C# дало змогу реалізовувати елементи ООП у розробці, а також забезпечити кросплатформену основу для проекту.

Відладка виконаної роботи виконувалась методом тестування ігрового процесу із використанням створених модулів. Оскільки було з нуля створено систему взаємодії з оточенням, необхідно було виконати роботу с перевірки роботи кожного ігрового елементу оточення.

В результаті було розроблено систему характеристик та вмінь гравця для гри у жанрі адвенчура, що дало змогу розширити ігровий процес, та збільшити різноманіття варіантів проходження рівнів гри.

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 характеристик та вмінь на мові C#

Скрипт parameters_logic.cs

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

public class parameters_logic : MonoBehaviour
{
    [SerializeField] body_parameters _body_parameters;
    [SerializeField] Text _parameter_text;
    [SerializeField] parameters_points_logic _parameters_points_logic;

    public int value = 0;

    [SerializeField] bool take_strength = false;
    [SerializeField] bool take_dexterity = false;
    [SerializeField] bool take_perception = false;
    [SerializeField] bool take_luck = false;

    void Start()
    {
        if (take_strength)
            value = _body_parameters.strength;
        else if (take_dexterity)
            value = _body_parameters.dexterity;
        else if (take_perception)
            value = _body_parameters.perception;
        else if (take_luck)
            value = _body_parameters.luck;
        _parameter_text.text = value.ToString();
    }

    public void take_new_body_parameter(int strength_value, int dexterity_value,
int perception_value, int luck_value)
    {
        if (take_strength)
            value = strength_value;
        else if (take_dexterity)
            value = dexterity_value;
        else if (take_perception)
```

```
            value = perception_value;
        else if (take_luck)
            value = luck_value;
        _parameter_text.text = value.ToString();
    }

    public void increase_value()
    {
        if (_parameters_points_logic.value > 0)
        {
            if (value < 10)
            {
                ++value;
                --_parameters_points_logic.value;
                _parameter_text.text = value.ToString();
                _parameters_points_logic.show_new_value();
            }
            else
                return;
        }
        else
            return;
    }

    public void decrease_value()
    {
        if (value > 0)
        {
            --value;
            ++_parameters_points_logic.value;
            _parameter_text.text = value.ToString();
            _parameters_points_logic.show_new_value();
        }
        else
            return;
    }
}
```

Скрипт parameters_points_logic.cs

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

```
public class parameters_points_logic : MonoBehaviour
{
    [SerializeField] body_parameters _body_parameters;
    [SerializeField] Text _points_to_spare_text;
    public int value = 0;
    public bool can_modify = true;
    public bool can_increase_points = true;

    private void Start()
    {
        value = _body_parameters.statespoints_value;
        _points_to_spare_text.text = value.ToString();
    }

    public void take_new_body_statespoints(int states_point_value)
    {
        value = states_point_value;
        _points_to_spare_text.text = value.ToString();
    }

    public void show_new_value()
    {
        _points_to_spare_text.text = value.ToString();
    }
}
```

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)
        {
            _scenes_loader.load_main_menu();
        }
        _health_bar_logic.set_hp_value(health);
    }
}
```

Скрипт event_script.cs

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

public class event_script : MonoBehaviour
{
    [SerializeField] GameObject event_related_gameobject;
    [SerializeField] GameObject event_related_event;
    [SerializeField] GameObject player_gameobject;
    [SerializeField] Text text_panel;
    player_parameters _player_parameters;
    [SerializeField] bool lader_event = false;
    [SerializeField] bool rope_event = false;
    [SerializeField] bool hooks_event = false;
```

```
        [SerializeField] bool door_event = false;
        [SerializeField] bool elevator_event = false;
        [SerializeField] bool bridge_event = false;
        [SerializeField] bool hatches_event = false;
        [SerializeField] int strength_requirements = 0;
        [SerializeField] int dexterity_requirements = 0;
        [SerializeField] int perception_requirements = 0;
        [SerializeField] int luck_requirements = 0;
        [SerializeField] int dificulty_modificator = 0;
        int player_strength_to_send = 0;
        int player_dexterity_to_send = 0;
        int player_perception_to_send = 0;
        int player_luck_to_send = 0;

        private void Start()
        {
            _player_parameters = player_gameobject.GetComponent<player_parameters>();
        }

        public void event_caller(int player_strength, int player_dexterity, int player_perception, int player_luck)
        {
            if (strength_requirements > 0)
                player_strength_to_send = player_strength;
            if (dexterity_requirements > 0)
                player_dexterity_to_send = player_dexterity;
            if (perception_requirements > 0)
                player_perception_to_send = player_perception;
            if (luck_requirements > 0)
                player_luck_to_send = player_luck;
            if (lader_event)
                lader_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
            else if (rope_event)
                rope_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
            else if (hooks_event)
                hooks_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
```

```
        else if (door_event)
            door_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
        else if (elevator_event)
            elevator_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
        else if (bridge_event)
            bridge_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
        else if (hatches_event)
            hatches_event_processor(strength_requirements, dexterity_requirements, perception_requirements, luck_requirements, player_strength_to_send, player_dexterity_to_send, player_perception_to_send, player_luck);
    }

    void lader_event_processor(int strength_req_val, int dexterity_req_val, int perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val)
    {
        int pass_value = 0;
        int player_cube_throw = 0;
        int player_result = 0;
        pass_value = strength_req_val + dexterity_req_val + perception_req_val + luck_req_val + dificulty_modificator;
        player_cube_throw = Random.Range(1, 6);
        player_result = player_cube_throw + pl_strength_val + pl_dexterity_val + pl_perception_val + pl_luck_val;
        if (player_result >= pass_value)
        {
            player_pass_event();
        }
        else if (player_result < (pass_value / 2))
        {
            player_critical_fail(pass_value, player_result);
        }
        else
        {
            player_fail_event(player_result, pass_value);
        }
```

```
    }

    void rope_event_processor(int strength_req_val, int dexterity_req_val, int perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val)
    {
        int pass_value = 0;
        int player_cube_throw = 0;
        int player_result = 0;
        pass_value = strength_req_val + dexterity_req_val + perception_req_val + luck_req_val + dificulty_modificator;
        player_cube_throw = Random.Range(1, 6);
        player_result = player_cube_throw + pl_strength_val + pl_dexterity_val + pl_perception_val + pl_luck_val;
        if (player_result >= pass_value)
        {
            player_pass_event();
        }
        else if (player_result < (pass_value / 2))
        {
            player_critical_fail(pass_value, player_result);
        }
        else
        {
            player_fail_event(player_result, pass_value);
        }
    }

    void hooks_event_processor(int strength_req_val, int dexterity_req_val, int perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val)
    {
        int pass_value = 0;
        int player_cube_throw = 0;
        int player_result = 0;
        pass_value = strength_req_val + dexterity_req_val + perception_req_val + luck_req_val + dificulty_modificator;
        player_cube_throw = Random.Range(1, 6);
        player_result = player_cube_throw + pl_strength_val + pl_dexterity_val + pl_perception_val + pl_luck_val;
        if (player_result >= pass_value)
        {
            player_pass_event();
```

```
        }
        else if (player_result < (pass_value / 2))
        {
            player_critical_fail(pass_value, player_result);
        }
        else
        {
            player_fail_event(player_result, pass_value);
        }
    }

    void door_event_processor(int strength_req_val, int dexterity_req_val, int perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val)
    {
        int pass_value = 0;
        int player_cube_throw = 0;
        int player_result = 0;
        pass_value = strength_req_val + dexterity_req_val + perception_req_val + luck_req_val + dificulty_modificator;
        player_cube_throw = Random.Range(1, 6);
        player_result = player_cube_throw + pl_strength_val + pl_dexterity_val + pl_perception_val + pl_luck_val;
        if (player_result >= pass_value)
        {
            player_pass_event();
            if(event_related_gameobject != null)
                event_related_gameobject.SetActive(false);
            if(event_related_event != null)
                event_related_event.SetActive(false);
            this.gameObject.SetActive(false);
        }
        else if (player_result < (pass_value / 2))
        {
            player_critical_fail(pass_value, player_result);
        }
        else
        {
            player_fail_event(player_result, pass_value);
        }
    }

    void elevator_event_processor(int strength_req_val, int dexterity_req_val, int
```

```
perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val)
        {
            int pass_value = 0;
            int player_cube_throw = 0;
            int player_result = 0;
            pass_value = strength_req_val + dexterity_req_val + perception_req_val + luck_req_val + dificulty_modificator;
            player_cube_throw = Random.Range(1, 6);
            player_result = player_cube_throw + pl_strength_val + pl_dexterity_val + pl_perception_val + pl_luck_val;
            if (player_result >= pass_value)
            {
                player_pass_event();
            }
            else if (player_result < (pass_value / 2))
            {
                player_critical_fail(pass_value, player_result);
            }
            else
            {
                player_fail_event(player_result, pass_value);
            }
        }

        void bridge_event_processor(int strength_req_val, int dexterity_req_val, int perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val)
        {
            int pass_value = 0;
            int player_cube_throw = 0;
            int player_result = 0;
            pass_value = strength_req_val + dexterity_req_val + perception_req_val + luck_req_val + dificulty_modificator;
            player_cube_throw = Random.Range(1, 6);
            player_result = player_cube_throw + pl_strength_val + pl_dexterity_val + pl_perception_val + pl_luck_val;
            if (player_result >= pass_value)
            {
                player_pass_event();
            }
            else if (player_result < (pass_value / 2))
            {
```

```
            player_critical_fail(pass_value, player_result);
        }
        else
        {
            player_fail_event(player_result, pass_value);
        }
    }

    void hatches_event_processor(int strength_req_val, int dexterity_req_val, int perception_req_val, int luck_req_val, int pl_strength_val, int pl_dexterity_val, int pl_perception_val, int pl_luck_val)
    {
        int pass_value = 0;
        int player_cube_throw = 0;
        int player_result = 0;
        pass_value = strength_req_val + dexterity_req_val + perception_req_val + luck_req_val + dificulty_modificator;
        player_cube_throw = Random.Range(1, 6);
        player_result = player_cube_throw + pl_strength_val + pl_dexterity_val + pl_perception_val + pl_luck_val;
        if(player_result >= pass_value)
        {
            player_pass_event();
        }
        else if (player_result < (pass_value / 2))
        {
            player_critical_fail(pass_value, player_result);
        }
        else
        {
            player_fail_event(player_result, pass_value);
        }
    }

    void player_pass_event()
    {
        player_gameobject.transform.position = event_related_event.transform.position;
        text_panel.gameObject.SetActive(true);
        text_panel.text = "Подія успішно пройдена!";
    }

    void player_fail_event(int player_result, int pass_value)
```

```
        {
            text_panel.gameObject.SetActive(true);
            text_panel.text = "Подія не пройдено. Твоє ЗП:" + player_result + ". Значення ЗС:" + pass_value + ".";
        }

        void player_critical_fail(int damage_value, int player_result)
        {
            text_panel.gameObject.SetActive(true);
            text_panel.text = "Подія критично провалена! Твоє ЗП:" + player_result + ". Значення ЗС:" + damage_value + ".";
            _player_parameters.take_damage(damage_value);
        }
    }
```

ДОДАТОК Б. Слайди мультимедійної презентації

Реалізація системи характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity

ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА ГРУПИ 4КС-57: ЛЄСНІКОВА ВІТАЛІЯ ОЛЕКСАНДРОВИЧА

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

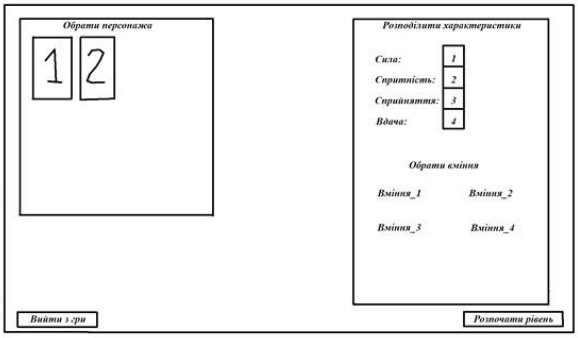
Ігровий двигун Unity є одним із популярніших ігрових двигунів у світі. За його допомогою створюють як мобільні, так і комп'ютерні ігри. Цей двигун досі активно розвивається!

- 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 характеристик у проекті

Механіка управління та вмінь у проекті основана на ідеї внесення до жанру адвенчури елементів рольової гри.

- Створює умови для варіативного проходження;
- Відкриває можливості створення подій, як елементів ігрового процесу;
- Вміння та характеристики можуть допомогти гравцю швидко пройти рівень, якщо правильно розподілені очки характеристик та вмінь;
- Можливість швидкого додавання нових характеристик та подій, за потребою.



Огляд елементів системи характеристик та вмінь гравця

Структурно система характеристик та вмінь гравця існує у двох ігрових станах – в меню розподілу характеристик (панель характеристик) та ігровому процесі.

На Панелі характеристик, система представлена механізмами розподілу очок характеристик між характеристиками.

Вміння також можуть бути обрані, за окремі очки.

Очки можна отримати обираючи персонажа для гри.

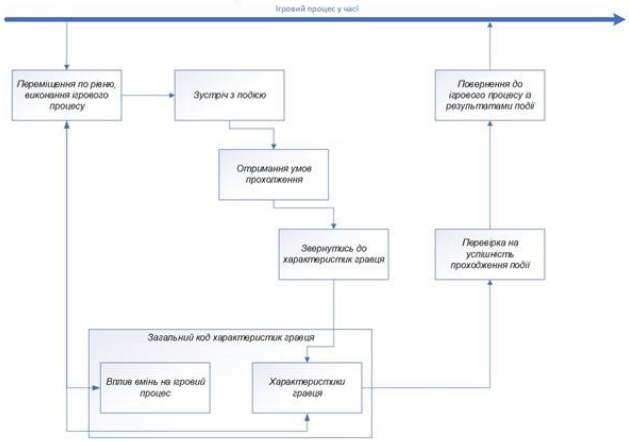
Всі дані розподілу зберігаються та передаються до ігрового процесу.

Принцип роботи елементів системи характеристик та вмінь гравця

В ігровому процесі система характеристик та вмінь представлена пасивним Модулем Характеристик гравця.

У разі потреби ігрові об’єкти під час ігрового процесу можуть звертатись до Модулю Характеристик гравця.

В свою чергу модуль може впливати на ігровий процес додаючи бонуси за рахунок характеристик та вмінь.



Принцип роботи елементів системи характеристик та вмінь гравця

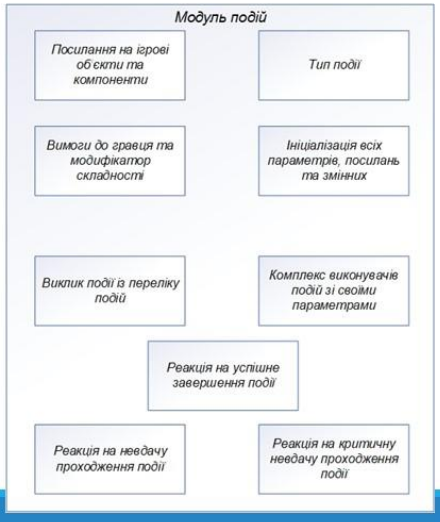
Модуль Характеристик гравця отримує дані характеристик та вмінь із відповідного меню.

- Змінює параметри пристроїв гравця;
- Має комплекс передачі значень характеристик у інші ігрові об’єкти за потреби;
- Звернення обробляються через відповідні методи, які у вильному доступі.

Принцип роботи елементів системи характеристик та вмінь гравця

Модуль подій створюється автоматично для будь-якої події на рівні як окремий екземпляр.

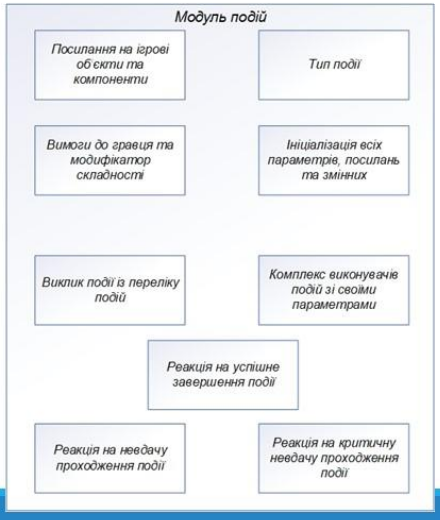
- Отримує посилання на ігрові об’єкти та компоненти для подій;
- Може бути вибраний тип події у ігрового об’єкту;
- Отримує та ініціалізує усі характеристики гравця;
- У раз взаємодії із гравцем, він може викликати окрему подію, пройти через перевірку успішного завершення.



Принцип роботи елементів системи характеристик та вмінь гравця

Модуль виконує роль елементу, який розширює ігровий досвід гравця, оскільки дещо доповнює ігровий жанр адвенчури елементами рольової гри.

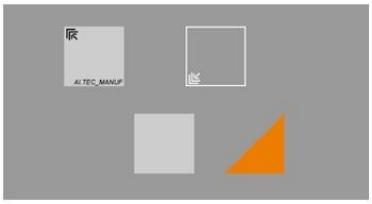
Викликається за допомогою коллайдера зі сторони гравця.

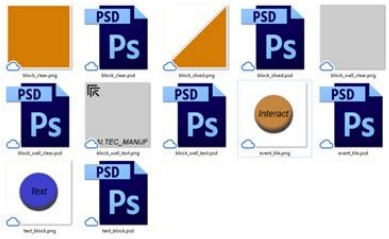


Етапи реалізації елементів системи характеристик та вмінь гравця

Реалізація елементів характеристик та вмінь складалась з:

- Реалізації спрайтів для рівня гри;
- Створення рівня із спратів;
- Створення ігрового об'єкту гравця;
- Написання коду інших модулів гри;
- Написання коду системи характеристик та вмінь гравця;
- Тестування.



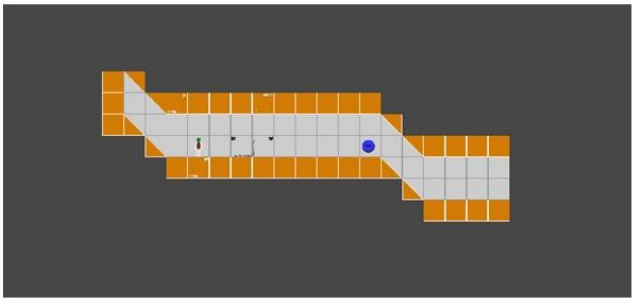


Хід тестування гри

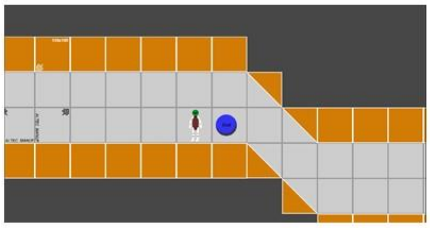
Тестування гри проводилось з допомогою вбудованого у ігровий програмний рушій Unity інструмент.

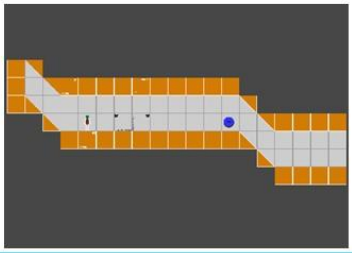
Метою тестування було:

- Визначити коректність роботи системи характеристик та вмінь;
- Взаємодію з ігровим оточенням;
- Реалізація можливості відтворювати ігрові події та систему їх програмування.



Скріншоти реалізованої гри





ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Лєснікова Віталія Олександровича

(прізвище, ім'я та по батькові)

Спеціальність: 123 "Комп'ютерна інженерія"

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

Тема дипломного проекту: Реалізація системи характеристик та вмінь гравця 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 78 сторінок. У пояснювальній записці виконано опис предметної області, способи реалізації системи характеристик та вмінь для гри в жанрі адвенчура. Проведено проектування та реалізацію намічених систем. 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 Лєсніков В.О. 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 Лєсніков В.О. 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 Лєсніков В.О. мав змогу самостійно приймати рішення з реалізації системи характеристик та вмінь, 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

Лєснікова Віталія Олександровича

(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 характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity

Обсяг розрахунково-пояснювальної записки _____ сторінок

Обсяг графічної (презентаційної) частини _____ 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 характеристик та вмінь гравця;
2. Виконано проектування намічених елементів гри із поясненнями на схемах;
3. Розроблено функціонуючу систему характеристик та вмінь гравця для гри.

д) основні недоліки дипломного проекту (роботи)

1. Кількість розроблених характеристик та глибина проробки мала;
2. Низький графічний супровід системи.

Оцінка розрахункової частини Добре

Оцінка графічної частини Добре

Загальна оцінка Добре

Прізвище, ім'я, по батькові рецензента к.т.н. Селіванова Алла Віталіївна

Місце роботи і посада рецензента Одеський національний технологічний університет, декан факультету комп'ютерної інженерії, програмування та кіберзахисту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ ФАКУЛЬТЕТ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ, ПРОГРАМУВАННЯ ТА КІБЕРЗАХИСТУ

Підпис:

«17» червня 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григорівна Краснокутська

ID перевірки:
1016348158

Дата перевірки:
11.06.2024 16:47:14 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2024 16:52:58 EEST

ID користувача:
100011688

Назва документа: 4КС-57_Лєсніков

Кількість сторінок: 43 Кількість слів: 8561 Кількість символів: 60561 Розмір файлу: 2.01 MB ID файлу: 1016151094

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.42%
Схожість

Найбільша схожість: 3.57% з Інтернет-джерелом (https://card.file.ontu.edu.ua/server/api/core/bitstreams/6c9508b-bff...

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,

Лєсніков Віталій Олександрович,
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 характеристик та вмінь гравця гри у жанрі адвенчура на програмному рушії Unity» (автор роботи – Лєсніков В.О., 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 _____ / Лєсніков В.О./

Керівник _____ / Шувалова І.О./

«10» 06 2024 р.