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.57.07.000.ДП

ЗАЙЦЕВА
ДМИТРА
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 управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity

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

Дипломник _____ (Зайцев Д.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 «20» 06 2024 р. Протокол ЕК № 4

Оцінка ЕК 4 (добре) 75 %

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 *Зайцеву Дмитру Олександровичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Реалізація системи управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity*

затверджена наказом по коледжу від "02" 11 2023 р. № 244-АД-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 управління та взаємодії гравця із оточенням комп'ютерної гри в жанрі адвенчура; Гра повинна мати основні признаки жанру адвенчура; Гра повинна мати інтерфейс; Виконана частина повинна давати змогу гравцю управляти персонажем, а також взаємодіяти с оточенням.*

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 управління та взаємодії гравця з оточенням; Проектування елементів системи управління, взаємодії гравця з оточенням та принцип їх роботи; Реалізація елементів системи управління та взаємодії з оточенням; 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 управління проекту; Особливості механіки взаємодії з оточенням; Огляд елементів системи управління та взаємодії з оточенням; Принцип роботи елементів системи управління та взаємодії з оточенням; Етапи реалізації елементів системи управління та взаємодії з оточенням; 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 *Зайцев Д.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 управління та взаємодії*	*28.05.2024*	Виконав
5	*Проектування системи управління та взаємодії*	*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 управління та взаємодії*	*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. 07 000. 00 ДП	Дипломний проект		
А4			КС 57. 07 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 57. 07 000. 00 ДП		
Розробив		Зайцев Д.О.		10.06	Реалізація системи управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity	Літ.	Аркуш 4 / Аркушів 71
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

В наші дні створенням ігрових додатків займаються як великі команди, нараховуючи сотні робітників, так і невеликі групи людей, об’єднаних інтересом та однією ідеєю. Найчастіше такі проекти відрізняються один від одного графікою, кількістю ігрового та наративного контенту, різноманіттям ігрових механік. Але бувають випадки, коли невеликі команди – навіть одна людина – можуть створювати справжні перлини ігрової індустрії. Прикладів таких вдосталь.

Ігрові проекти складаються з багатьох складових, кожна з них створює важливі частини ігрового процесу, враження від гри, або ігрових механік. Окремо одна від одної, ці складові не можуть гарантувати вдалого ігрового проекту, а їх вірне комбінування дозволяє створити збалансовану гру у всіх відношеннях.

Темою мого дипломного проекту є «Реалізація системи управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity». Маючи за мету створити цікаву адвенчуру необхідно реалізувати в ній просту та в той же час розвинуту систему управління, яка б давала змогу гравцю широко використовувати оточення, ігрові обставини та параметри гравця. Вдало реалізоване управління ігровим персонажем створює позитивний ігровий досвід на протязі всієї гри.

Використання сучасного ігрового програмного рушія Unity дає змогу гнучкого підходу до створення управління в будь-яких іграх, не залежно від жанрів. Окрім того уміння розробляти схеми управління для цього програмного рушія дозволяє отримати корисний досвід у розробці ігрових механік, що досить затребувано у команд, що займаються розробкою ігрових проектів.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз ігрового жанру адвенчура

1.1.1 Загальний огляд питання

Перед тим як розпочати роботу над дипломним проектом необхідно визначитись із декількома питаннями. Потрібно визначити, що таке ігровий жанр «Адвенчура», які його особливості та що з цього можна використати для виконання теми мого дипломного проектування.

Ігрові жанри в ігровій індустрії представлені дуже широким переліком, яких не менше, а в деякому сенсі навіть більший ніж перелік кінематографічних жанрів. 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. Ігровий жанр визначається ігровими механіками, прийомами зовнішнього відображення, елементами ігрового процесу, розташування ігрової камери.

Темою мого дипломного проекту є «Реалізація системи управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity», отже потрібно розкласти ігровий жанр адвенчури. 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.

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.

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

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. На рисунку 1.2 зображено скріншот з гри Life is Strange:

Рисунок 1.2. Скріншот з гри Life is Strang

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 1.3 зображено скріншот із гри Myst:

Рисунок 1.3. 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.

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 управління та взаємодії гравця із оточенням, 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 управління таким, щоби гравець міг досить вдумливо та неквапливо переміщуватись по ігровому світу виключаючи можливість пропустити зось важливе. Те ж саме стосується и системи взаємодії. Необхідно придумати таку систему, яка б давала змогу гравцю чітко розуміти, коли він може взаємодіяти з оточенням, а коли навпаки, не може.

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 управління та взаємодії гравця із оточенням 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 1.4 показано приклад роботи у редакторі ігрового програмного рушія Unity:

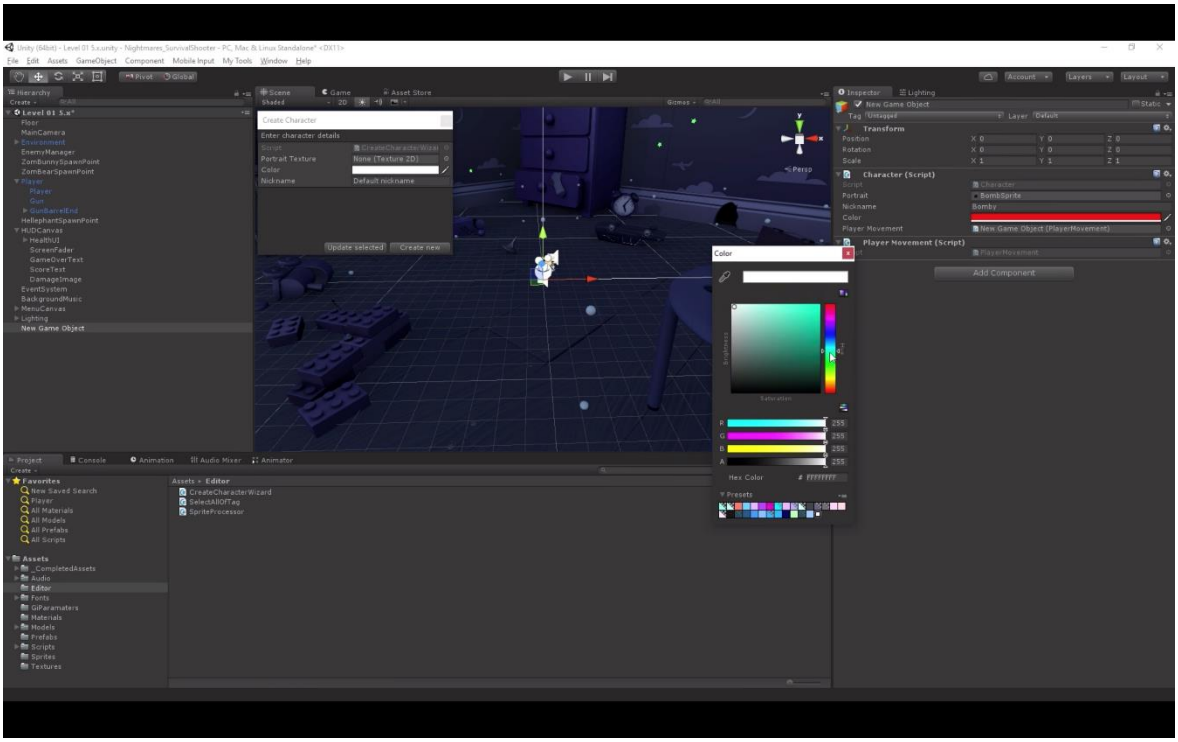


Рисунок 1.4. Приклад роботи у ігровому програмному рушії Unity

Однією з ключових особливостей Unity є його кросплатформенність. Він підтримує кілька платформ, включаючи ПК, консолі, мобільні пристрої, віртуальну реальність і доповнену реальність, що дозволяє розробникам охоплювати більшу аудиторію та запускати свої продукти на кількох пристроях.

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

1.2.2 Огляд ігрового програмного рушія CryEngine

Для того, щоби дати розуміння причини обрання ігрового рушія Unity, розглянемо його двох найближчих конкурентів. Почнемо з ігрового програмного рушія CryEngine який є потужним ігровим програмним рушієм, розробленим німецькою компанією Crytek. Сам ігровий програмний рушій відомий своїми вражаючими фізикою та графікою, а також широкими можливостями для створення відкритих багаторівневих та реалістичних ігрових світів. CryEngine був використаний у різних популярних іграх, таких як серія Crysis, Ryse: Son of Rome та Hunt: Showdown. Початково отримав відомість як ігровий програмний рушій, що відтворює настільки неймовірно реалістичну графіку, що навантажує найсильніші системи. Це стало в ті часи справжньою проблемою для розробників, оскільки використання цього програмного ігрового рушія потребувало сильну систему, так само, як і використання кінцевого продукту. Його правдоподібна симуляція фізики та руйнування довкілля також дуже довго була основною рисою цього ігрового програмного рушія. Через свої неймовірні візуальні та фізичні показники отримав відомість, як бєнчмарк для комп'ютерів того часу, а в деяких проектах сьогодення він таким продовжує залишатись. Одною з останніх крупних ігрових релізів на цьому ігровому програмному рушії став Ryse: Son of Rome, який був запускаючим ексклюзивом для приставки XBOX. На рисунку 1.5 можна побачити приклад графіки, що відтворює програмний рушій CryEngine.

Рисунок 1.5. Приклад графіки, що відтворює програмний рушій CryEngine

З точки зору можливостей, то однією з ключових особливостей CryEngine є його підтримка технології трасування променів (ray tracing), яка дозволяє досягти фотореалістичних ефектів освітлення та відображення поверхонь на різних матеріалах, як метал або водна поверхня. Крім того, CryEngine має високий рівень масштабованості, що дозволяє створювати як невеликі інді-проекти, так і великі AAA-ігри. Програмний рушій також забезпечує розробників інструментами для створення інтерактивного та живого ігрового середовища, включаючи систему штучного інтелекту, керування анімацією персонажів, а також інструменти для створення звукової атмосфери, що дуже часто використовують для створення досить якісних ігор в жанрі хоррор. CryEngine також пропонує гнучку систему багатоплатформної розробки, що дозволяє випускати ігри на різних платформах, включаючи ПК, консолі та мобільні пристрої. На рисунку 1.6 можна побачити приклад роботи з вбудованим аніматором рушія CryEngine.

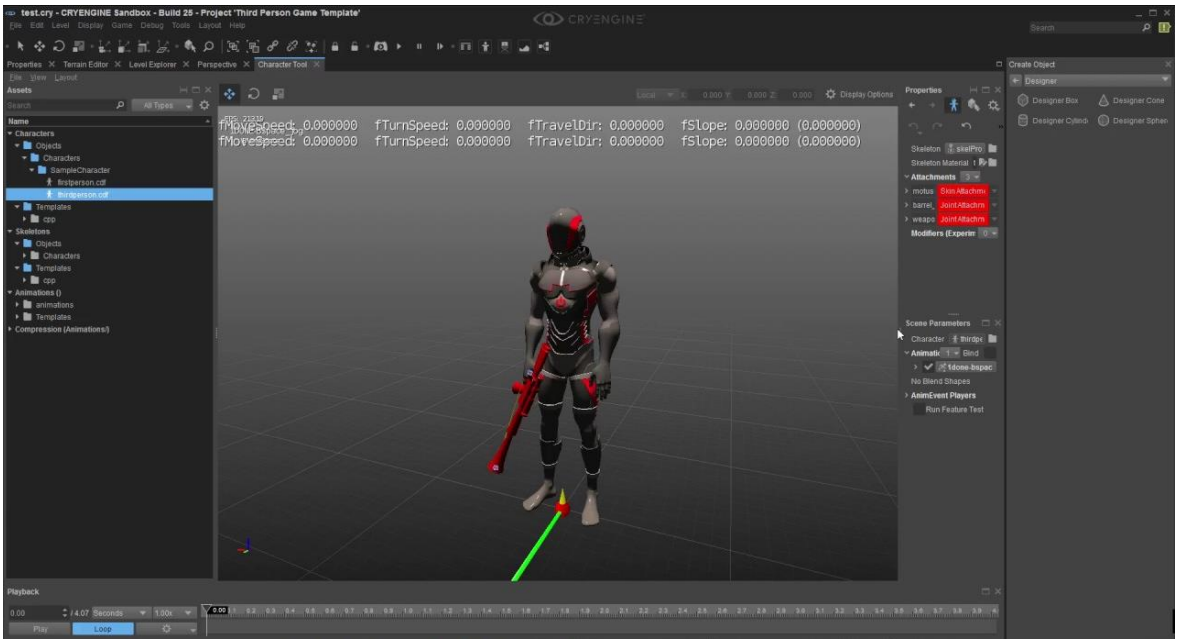


Рисунок 1.6. Приклад роботи з вбудованим аніматором рушія CryEngine

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

1.2.3 Огляд ігрового програмного рушія Unreal Engine

Другим відомим конкурентом Unity є ігровий програмний рушій Unreal Engine. Це потужний і широко використовуваний ігровий програмний рушій який був розроблений компанією Epic Games та вперше був використаний для створення гри Unreal Turnament звідки й взяв своє ім’я. Він надає розробникам інструменти для створення високоякісних ігор та візуалізацій, а також розробки віртуальної реальності (VR), доповненої реальності (AR) та багатьох інших інтерактивних проектів, різність та глибина яких залежить від вмінь розробника. Ігровий програмний рушій Unreal Engine пропонує широкий набір функцій, включаючи потужний рушій графіки, підтримку власного рушія фізіки, інструменти для роботи зі штучним інтелектом персонажів та середовища, глибоку анімацією персонажів, звуком та багатьом іншим. Він також включає графічний інтерфейс користувача, який спрощує процес розробки та створення контенту. На рисунку 1.7 можна побачити приклад роботи із анімацією у ігровому

рушії Unreal Engine.

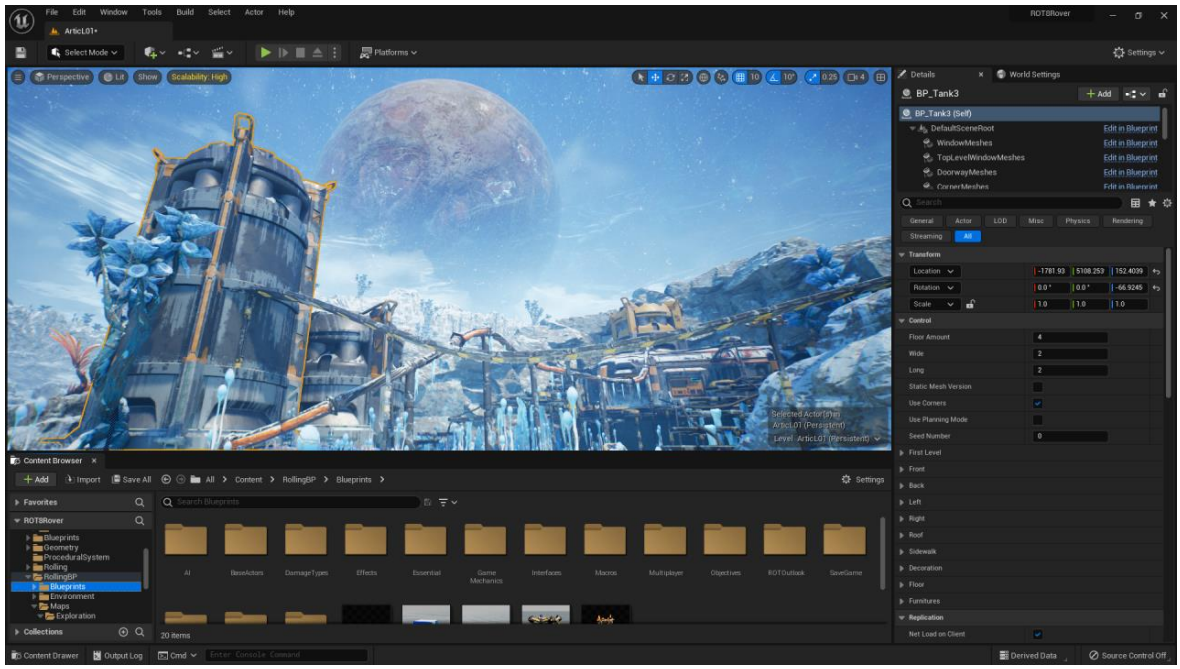


Рисунок 1.7. Приклад роботи із анімацією у ігровому рушії Unreal Engine

Однією з ключових особливостей Unreal Engine є його графічний рушій, який підтримує передові технології, такі як трасування променів (ray tracing), що дозволяє створювати неймовірно реалістичні графічні ефекти відображення та бліків на водній, або металічних поверхнях. Це робить Unreal Engine популярним вибором для розробників ААА-ігор, а також для створення вражаючих візуалізацій та симуляцій в інших галузях. Також, цей програмний рушій використовують для створення фільмів за допомогою спеціального інструментарію, який за допомогою штучного інтелекту будує кадр навколо людини. Ігровий програмний рушій Unreak Engine лежить в основі багатомільярдної гри Fortnite, яка є лідером у жанрі королівської битви з елементами платформеру, крафтингу та пісочниці. На рисунку 1.8 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 1.8. Приклад графіки яку відтворює Unreal Engine

1.2.4 Обґрунтування обрання ігрового програмного рушія Unity

Проаналізувавши ігрові програмні рушії було вирішено використовувати рушій Unity. Рішення було зумовлено в першу чергу доцільністю використання можливостей ігрових рушіїв. Його конкуренти надаються передову графіку дуже високого рівня, яка не потрібна у проекті в 2D жанрі. Темою мого дипломного проекту є «Реалізація системи управління та взаємодії гравця із оточенням 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:

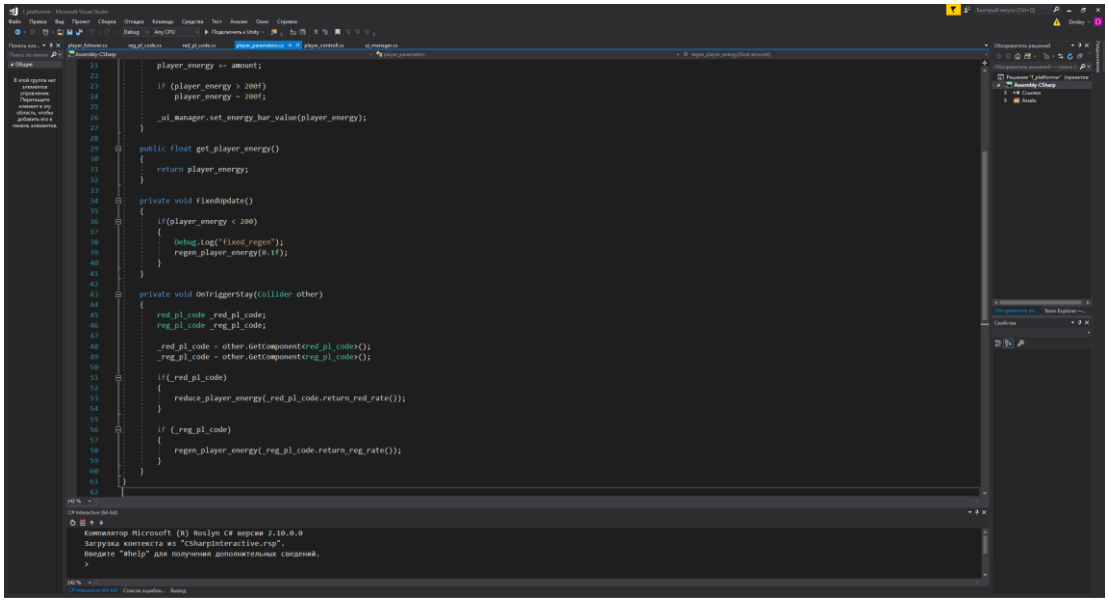


Рисунок 1.9. Вікно розробки Visual Studio проекту у Unity

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. На рисунку 1.10 зображено використання Adobe Photoshop для створення концепту гри:

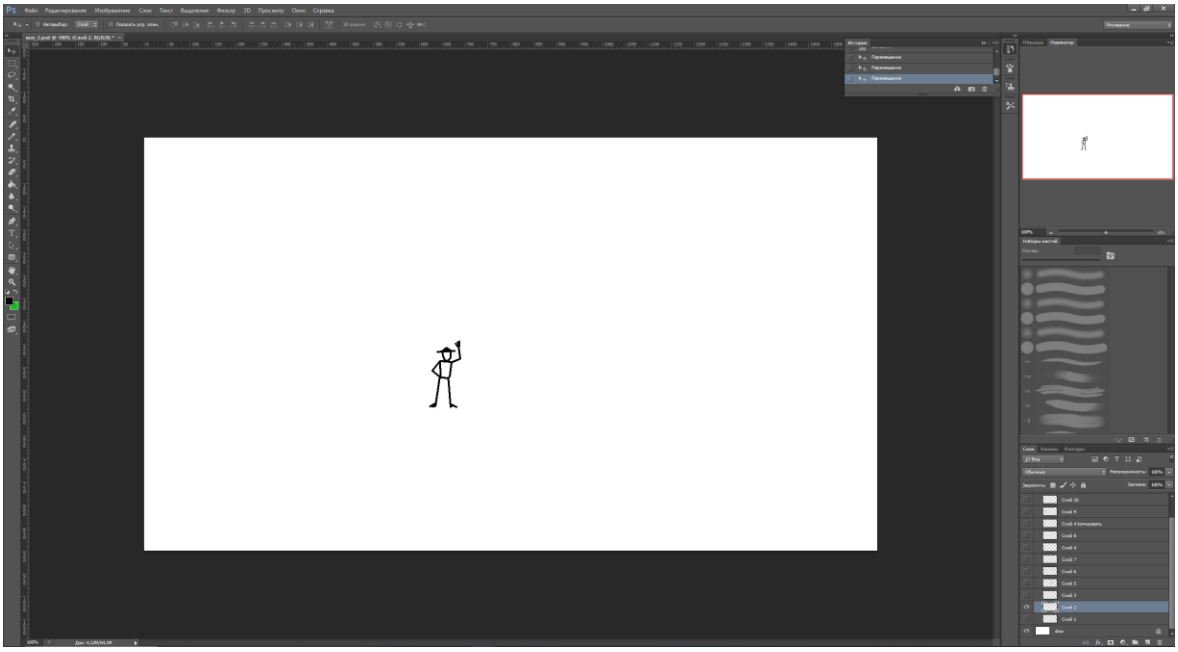


Рисунок 1.10. Використання Adobe Photoshop для створення концепту гри

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. Системи контролю версій я буду використовувати для зберігання робіт у випадку раптового вимкнення світла.

1.4 Формування концепту системи управління та взаємодії гравця з оточенням

1.4.1 Загальні відомості про концепт-документ

Для реалізації задачі 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;
- Механіка геймплею: Опис ігрових механік, механіки управління, фізики

ігрового світу, системи прогресії, системи бою та інших ігрових аспектів;

- 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.

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 управління та взаємодії гравцем із оточенням

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 управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity». Виходячи з теми проекту, а також особливостей ігрового жанру, потрібно реалізувати управління гравцю для його навігації у грі адвенчурі, а також створити систему для взаємодії з оточенням. Реалізовані елементи мають бути створені в дусі жанру та і з урахуванням концепту гри.

Отже потрібно визначити потреби від жанру. Спочатку розглянемо питання

переміщення по рівню. Жанрова складова гри диктує здібність для гравця вільно переміщуватись по рівню та взаємодіяти із оточенням. Опираючись на переглянуті раніше проекти, можна прийти до висновку, що управління не потребує клавіш для активного переміщення, наприклад стрибків, або присідів. Частіше за все, гравець має потребу рухатись від одного екрану, до іншого. Тому, буде достатньо руху вліво та вправо, або верх та вниз у разі руху по драбині.

Щодо взаємодії з оточенням, то, враховуючи жанрові особливості, для гравця потрібно урахувати момент взаємодії гравця із активними елементами оточення. Наприклад кнопками, перемикачами, або іншими елементами рівня.

Також, важливою складовою є не тільки взаємодія із активними елементами за допомогою клавіш клавіатури, але й використання для цих цілей миші. З іншого боку потрібно розуміти, що основою жанру адвенчура є головоломки, тому потрібно ураховувати просторовий принцип. Через це, гравець, звісно, не буде змоги натискати на елементи оточення мишкою на відстані від гравця.

Проробку концепції управління можна почати з руху. Як було написано вище, рух є важливою складовою ігрового процесу жанру адвенчура, тому потрібно приділити їй увагу, але пам’ятати, що ігровий процес не має на увазі стрибки та інші різки рухи.

Тому я вирішив використовувати класичну систему управління, яка базується на використанні клавіш клавіатури W A S D. Це універсальне рішення, яке інтуїтивно зрозуміло будь-якому гравцю в комп’ютерні ігри. Кнопки A та D будуть відповідати за рух аватару гравця вліво та вправо, відповідно. Клавіші клавіатури W і S будуть забезпечувати гравцю переміщення по драбині. Зчитування натискання цих кнопок буде виконуватись за допомогою ігрового програмного рушія Unity, а точніше його бібліотек. Можна використовувати і альтернативні способи управління, наприклад геймпад. Тому потрібно буде урахувати можливість використання інших систем управління проектом. На рисунку 1.11 можна побачити концепцію цієї ідеї:

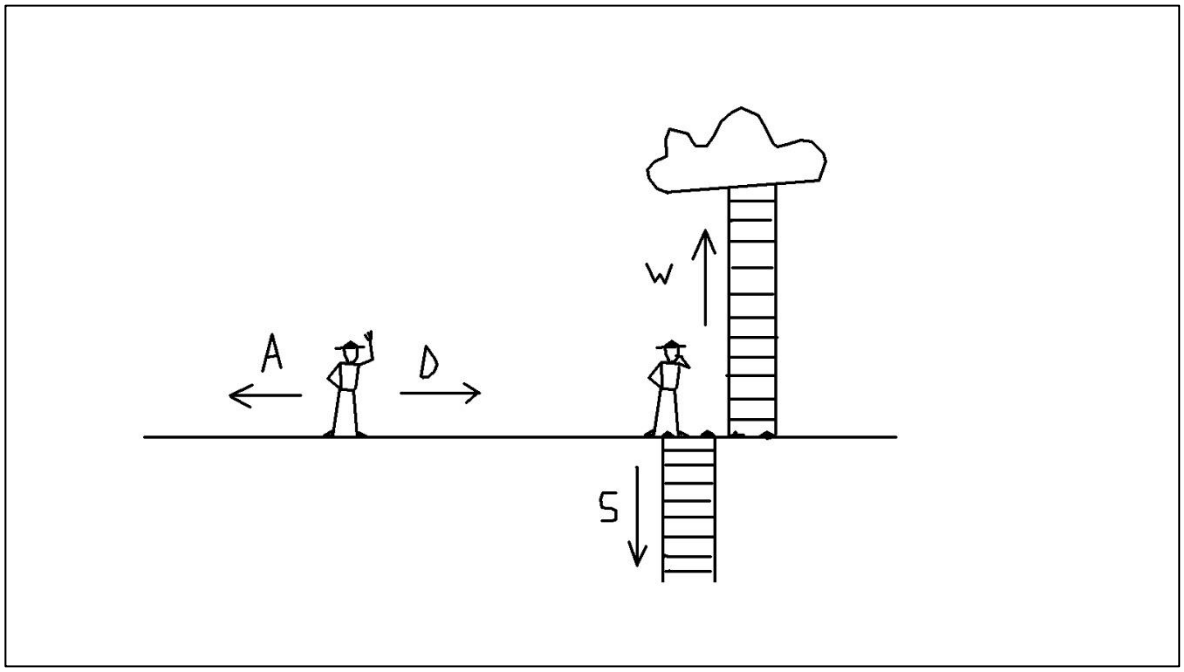


Рисунок 1.12. Концептуальне зображення системи руху гри адвенчури FarmerTrouble

Для виконання можливості гравця взаємодіяти з оточенням потрібно реалізувати механіку натискання на ігрові предмети на рівні. Це також можна реалізувати за допомогою клавіатури. Таким же чином, як і система руху, я використаю зарекомендовану у багатьох іграх клавішу E, яка відповідає за дію в багатьох іграх. Слід зазначити, що інколи цю роль виконує кнопка F, але з моєї точки зору кнопка E набагато легша з точки зору натискання у ігровому процесі. Також потрібно пам’ятати, що гравець повинен мати змогу натискати на ігрові об’єкти за допомогою миші. Тому взаємодію цього елементу управління також потрібно взяти до уваги. Втім, із використанням миші можуть з’явитись проблеми пов’язані із тим, що координати миші на екрані гравця не співпадають із координатами об’єкту у світі. Через це потрібно буде ураховувати таку проблему та використовувати вбудовану до ігрового програмного рушія Unity функцію, як переведення координат курсору миші на екрані до координат у ігровому світі. Тому, взаємодіючи із курсором я зможу отримувати його координати на рівні гри. На рисунку 1.13 зображено концепцію системи взаємодії гравця із оточенням:

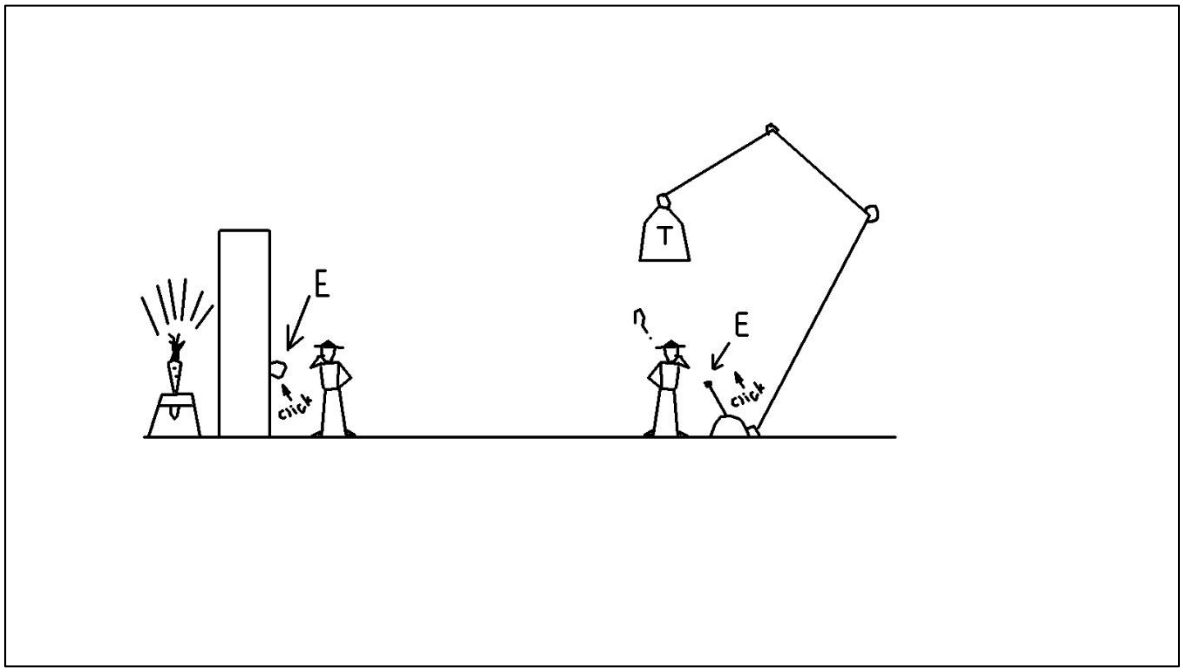


Рисунок 1.13. Концептуальне зображення системи взаємодії гравця із оточенням гри адвенчури FarmerTrouble

Виконавши формування ігрового концепту, або концепції використання систем управління гравцем, можна переходити до етапу проектування розробляємих елементів. Завдяки виділеним особливостям проекту, висунувши потреби до елементів та механік до реалізації, можна буде більш конкретно виконувати роботи з проектування не допускаючи помилок та не витрачаючи часу на опрацювання моментів у роботі та взаємодії всіх елементів проекту.

1.5 Проектування елементів системи управління, взаємодії гравця з оточенням та принцип їх роботи

1.5.1 Загальне проектування

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 управління та взаємодії з оточенням у грі. Для розуміння цього процесу потрібно зрозуміти сам принцип роботи ігрового програмного рушія Unity та важливих для реалізації теми диплому елементів. На рисунку 1.14 зображено схему взаємодії механізму вводу та ігрового процесу у іграх:

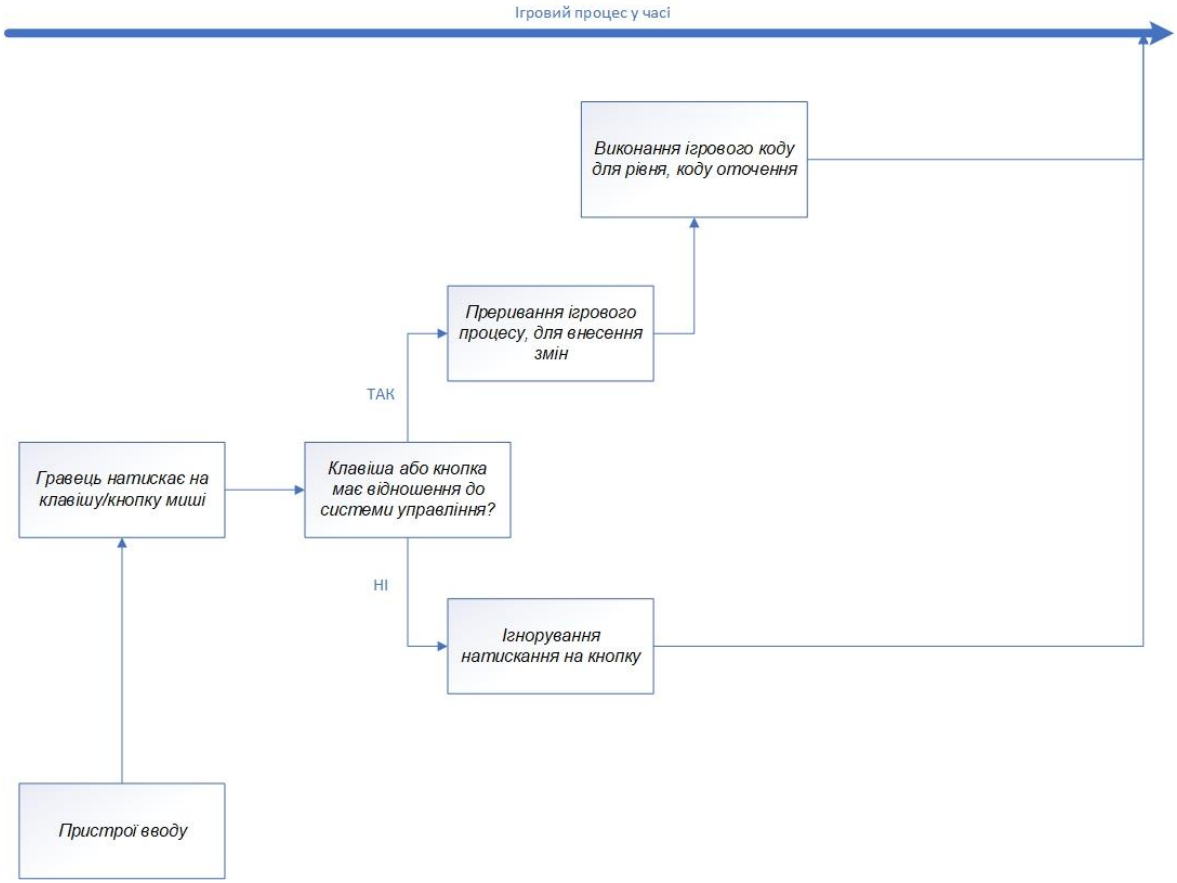


Рисунок 1.14. Схема взаємодії механізму вводу та ігрового процесу у іграх

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. На рисунку 1.14 можна побачити стрілку, яка імітує Ігровий процес у часі. Він виконується циклічно, але завжди у часі.

Паралельно із виконанням ігрового процесу, ігровий програмний рушій Unity має набор слухачів, які постійно зчитують дані з різних джерел. Одним із

таки слухачів є об’єкт вводу. Він має в собі дані щодо будь-якого пристрою вводу, що може бути підключений до ігрового пристрою. Цей об’єкт завжди зберігає значення натискання на ті, чи інші кнопки.

Для реалізації системи управління потрібно слухати натискання конкретних клавіш та кнопки миші. На рисунку 1.14 можна побачити, що після натискання будь-якої клавіши, ігровий програмний рушій має перевіряти, чи має натиснута кнопка відношення до системи управління гравця.

У разі, якщо натискання не має відношення до системи управління, то ігровий програмний рушій повертається до виконання залишкових команд, які залишилось виконати перед переходом до наступного кадру.

Якщо натискання на клавіші, або кнопки, мало відношення до системи управління, тоді ігровий програмний рушій виконує звернення до коду системи управління, та визначає, хто виконував перевірку, а також до якого коду потрібно передати управління. Коли такий код буде знайдено – він буде виконаний, після чого ігровий програмний рушій передає управління назад до виконання залишкових лій у Ігровому процесу у часі.

1.5.2 Проектування системи управління рухом гравця

Отже, ще з моменту розробки концепції, дуже чітко прослідковується чітке розмежування між системою управління гравцем для переміщенням, та системою взаємодії з оточенням. Спочатку, може здатись, що вони однакові, але це не так, оскільки принцип виконання змісту цих систем зовсім інший, як і компоненти, що використовуються для реалізації. Також відрізняється і призначення коду цих систем, тому, згідно раціональності та розуміння коду стороннім розробником, ці системи потрібно розділити одна від одної. Існуючи поряд з іншими системами та модулями гри Система управління гравцем має в собі деякі функційні блоки, які мають працювати у зв’язку один з одним. На рисунку 1.15 зображено схему модулів гри із додаванням управління гравцем.

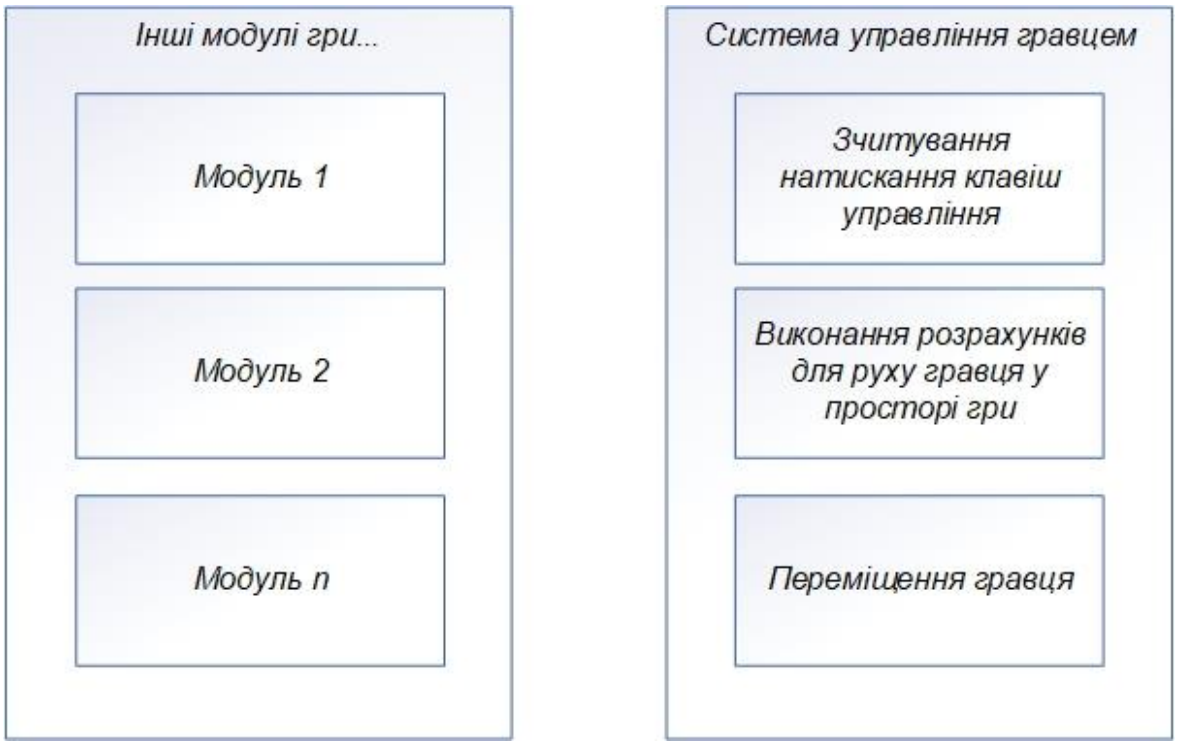


Рисунок 1.15. Схема модулів гри із додаванням Системи управлінням гравцем

Як можна побачити з рисунку 1.15, система управління гравцем складається із декількох складових модулів. По-перше виконується зчитування натискання на клавіші управління. Це зчитування має проводитись кожний кадр виконання гри. У разі зміни значень зчитуємих значень, система має виконати розрахунки для руху гравця у просторі гри. Після виконання розрахунків будуть отримані дані, щодо того, куди гравець хотів переміститись та на яку відстань. Ці дані будуть використані у частини переміщення гравця, яка змінить координати ігрового об’єкту гравця згідно із кодом.

1.5.3 Проектування системи взаємодії гравця із оточенням

Більш складу задачу складає реалізація системи взаємодії гравця із оточенням. Дану систему потрібно реалізувати таким чином, щоби вона могла бути покращена у майбутньому. Наприклад, у разі необхідності додавання до переліку ігрових предметів для взаємодії, скоротити кількість взаємодій із кодом, або ігровим об’єктом гравця для внесення цих змін.

Така система має існувати паралельно системі управління рухом гравця. Окрім того, ця система має реалізовувати перевірку, як на натискання кнопки для взаємодії, так і натискання кнопки миші. Все це прописано у концепті роботи такої

системи. На рисунку 1.16 схематично зображено схему модулів гри, систему управління гравцем та систему взаємодії з оточенням:

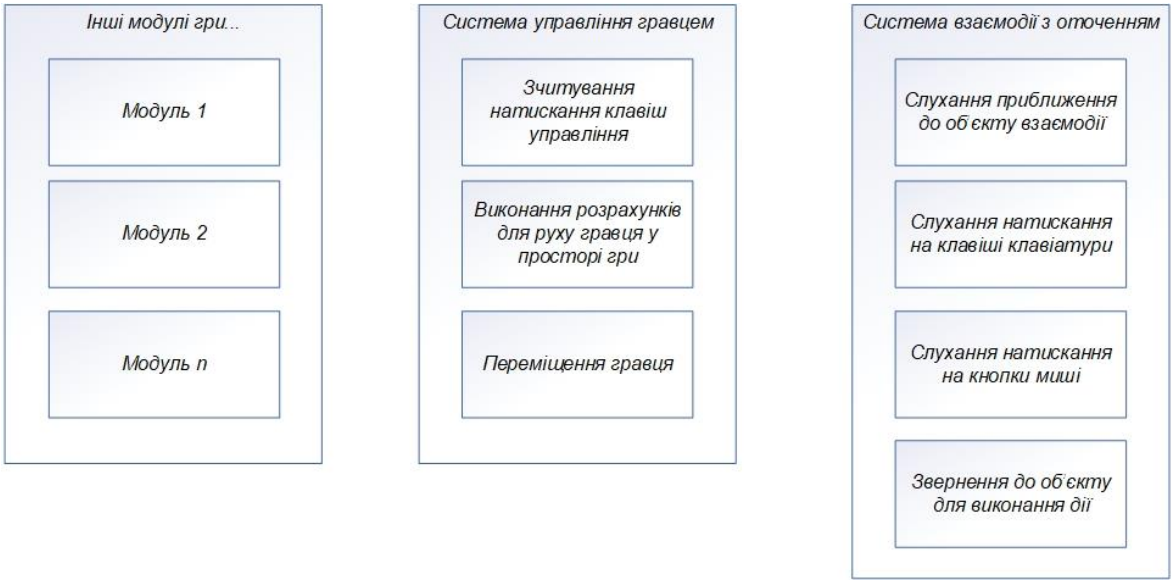


Рисунок 1.16. Схема модулів гри, системи управління гравцем та системи взаємодії з оточенням

Як можна побачити зі схеми, система взаємодії з оточенням повинна виконувати цілий ряд «слухань». В першу чергу потрібно слухати приближення гравця до об’єкту, з яким можна взаємодіяти. Тільки після цього виконується слухання натискання на клавіші клавіатури, або кнопку миші. У разі, якщо було натиснуті кнопки відповідаючі за взаємодію із оточенням, повинно викликатись звернення до об’єкту взаємодії для виконання запрограмованої дії.

Окрім того, така система має реалізовувати взаємодію гравця із предметом таким чином, щоби виправдовувати систему переміщення та ігровий процес гри в жанрі адвенчура. Якщо звернутись до ігор минулого, найчастіше за все потрібно було підійти до ігрового об’єкту оточення для взаємодії. Тому потрібно реалізувати таку систему, яка взаємодіяла б із об’єктами на відстані. Для цього можна використати технологію фізичних тіл ігрового програмного рушія Unity. У цієї технології є парний компонент який називається Collider, або колайдер. Він виконує роль так би мовити границі твердого тіла. Втім, у колайдеру є два режими роботи. Перший, як границя твердого тіла, а другий, як не фізичне поле, яке реагує на взаємодію із іншими колайдерами-границями твердих тіл – такий режим називається Тригером. На рисунку 1.17 можна побачити схематичну ілюстрацію

відсутньої інтеракції ігрового об’єкту гравця та ігрового об’єкту для взаємодії.

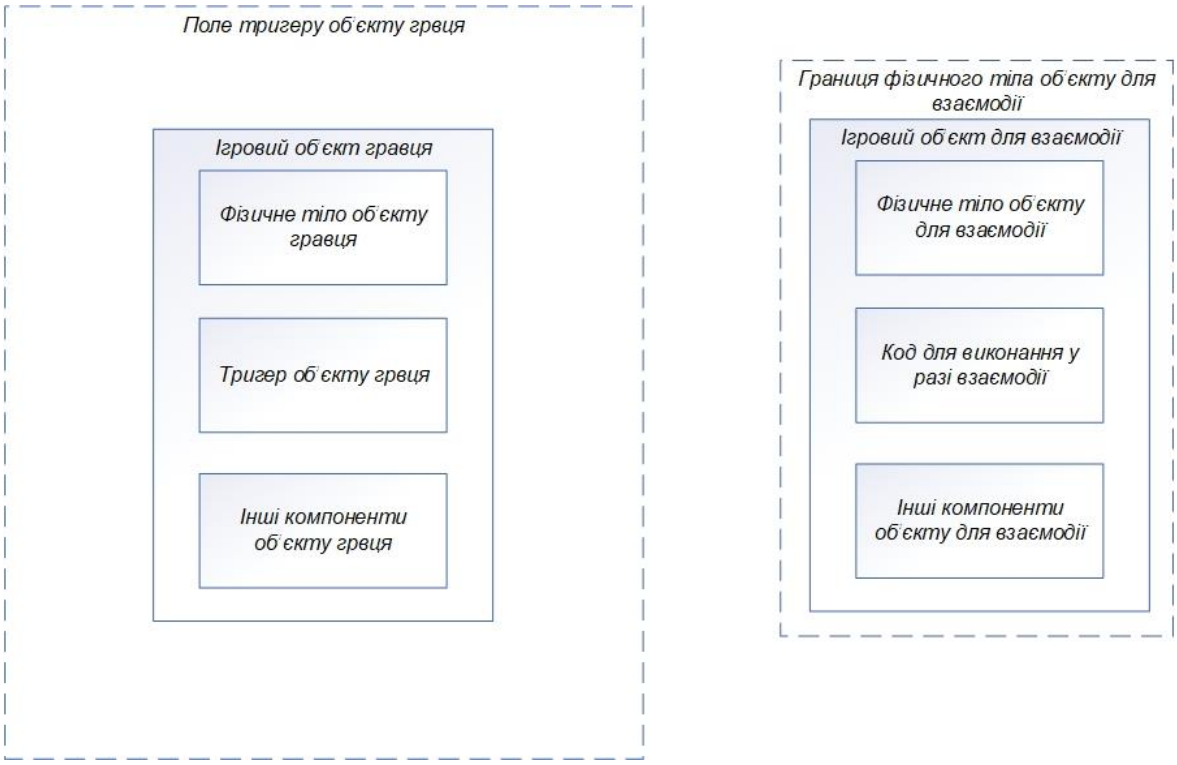


Рисунок 1.17. Схематична ілюстрація відсутньої інтеракції ігрового об’єкту гравця та ігрового об’єкту для взаємодії

Як можна побачити на рисунку 1.17, навколо ігрового об’єкту гравця існує так зване «Поле тригеру об’єкту гравця». Таке «поле» можна створити за допомогою колайдеру-тригеру. Втім, важливо відмітити, що у гравця також має бути своє тверде тіло та його границі для того, щоби пересуватись по поверхні. Тому у ігровому об’єкті гравця є два колайдери. Один працює як тригер і він більший, а другий, як границя твердого тіла.

Також, на рисунку 1.17 можна побачити блок ігрового об’єкту для взаємодії. В нього також є фізичне тіло, але тільки один колайдер, який і є границею цього ігрового об’єкту для взаємодії. У зображеній конфігурації, жодна дія не може бути виконана, оскільки ігровий об’єкт гравця сильно далеко від об’єкту для взаємодії. На рисунку 1.18 можна побачити схематичну ілюстрацію інтеракції ігрового об’єкту гравця та ігрового об’єкту для взаємодії:

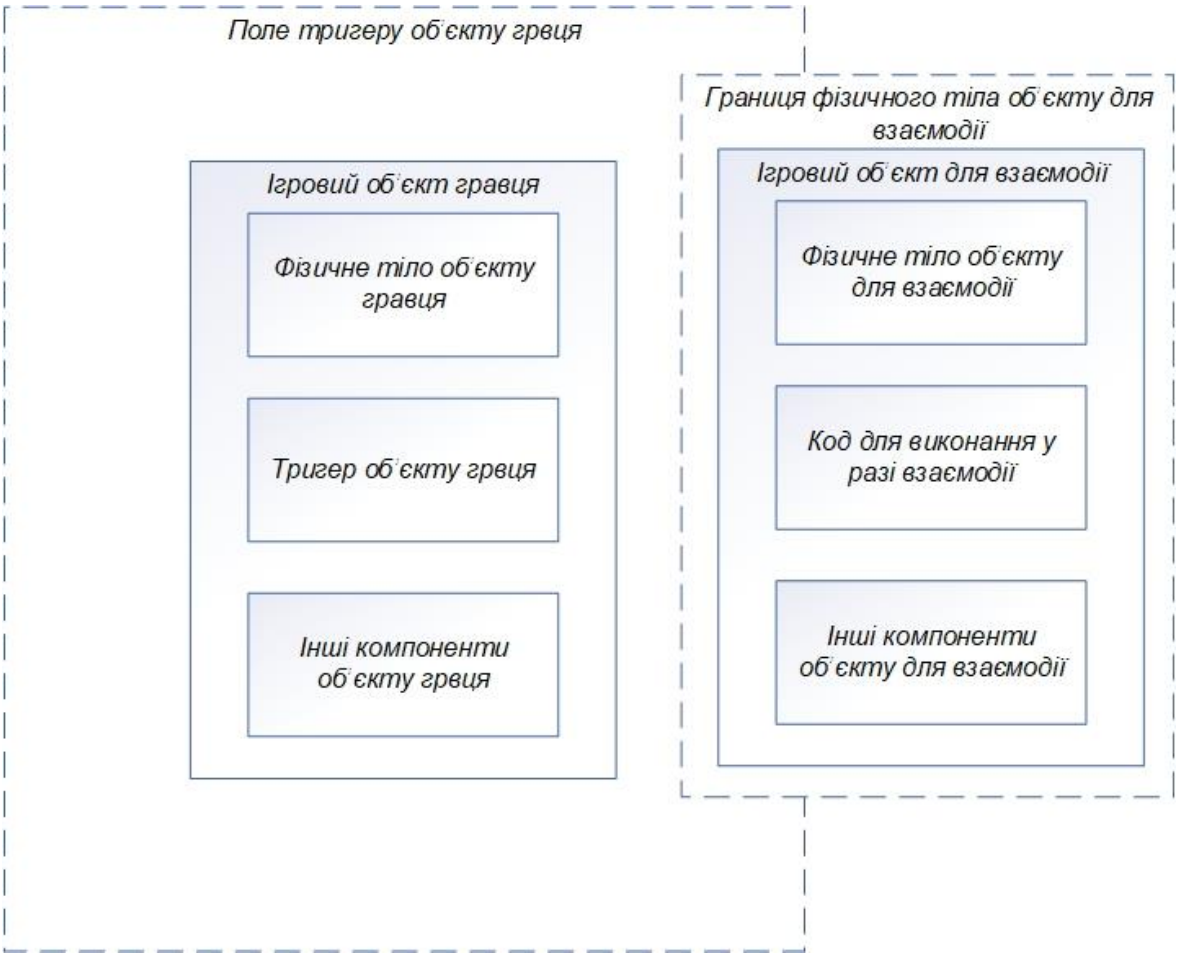


Рисунок 1.18. Схематична ілюстрація інтеракції ігрового об’єкту гравця та ігрового об’єкту для взаємодії

З рисунку 1.18 видно, як «Поле тригеру об’єкту гравця» перетинає границю фізичного тіла об’єкту для взаємодії. Це можливо тому, що тригери не мають фізичних границь, тому можуть проходити крізь інші об’єкти на сцені. Саме в той момент, коли тригер перетинає колайдер об’єкту для взаємодії, він передає керування до події, яка в свою чергу і буде викликати ті, чи інші дії у разі натискання відповідної клавіші клавіатури, або кнопки миші. У разі невірного натискання, дії та перетин буде ігноруватись. Таким чином реалізовуються принципи ігрового жанру адвенчура, свобода руху по рівню та можливість взаємодії гравця із оточенням із дотримання всіх вимог наміченої концепції.

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 управління та взаємодії з оточенням

Для успішного виконання теми мого дипломного проекту, потрібно створити гру 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 1.19 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.

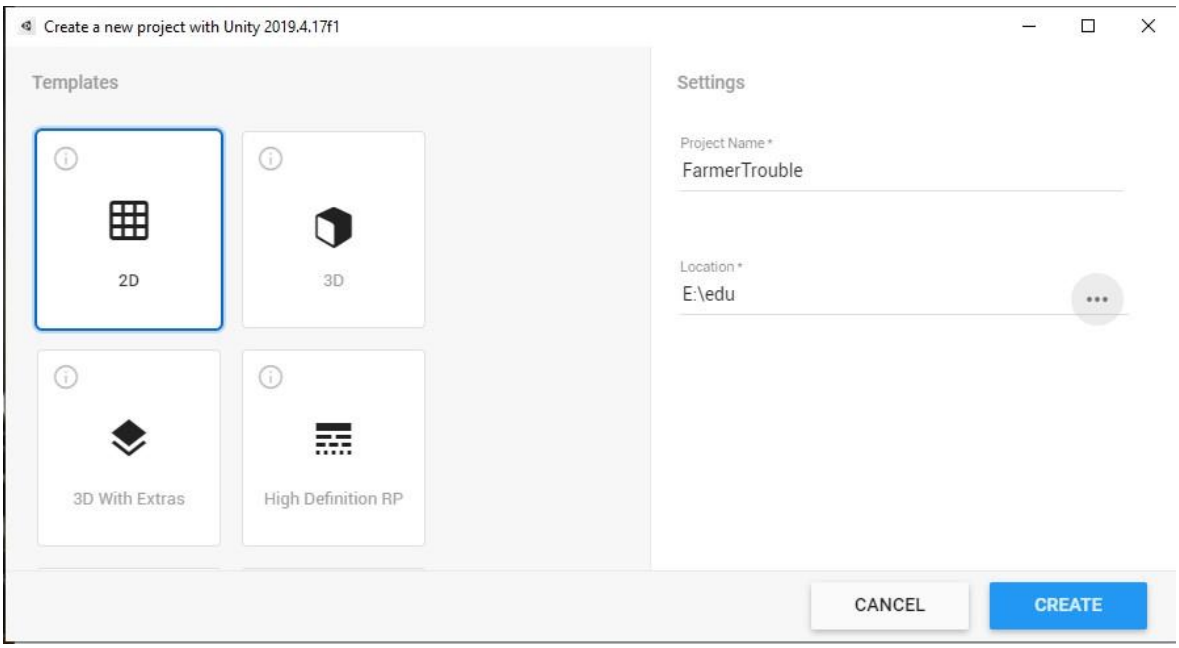


Рисунок 1.19. Вікно створення проекту у UnityHub

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 1.20. 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».

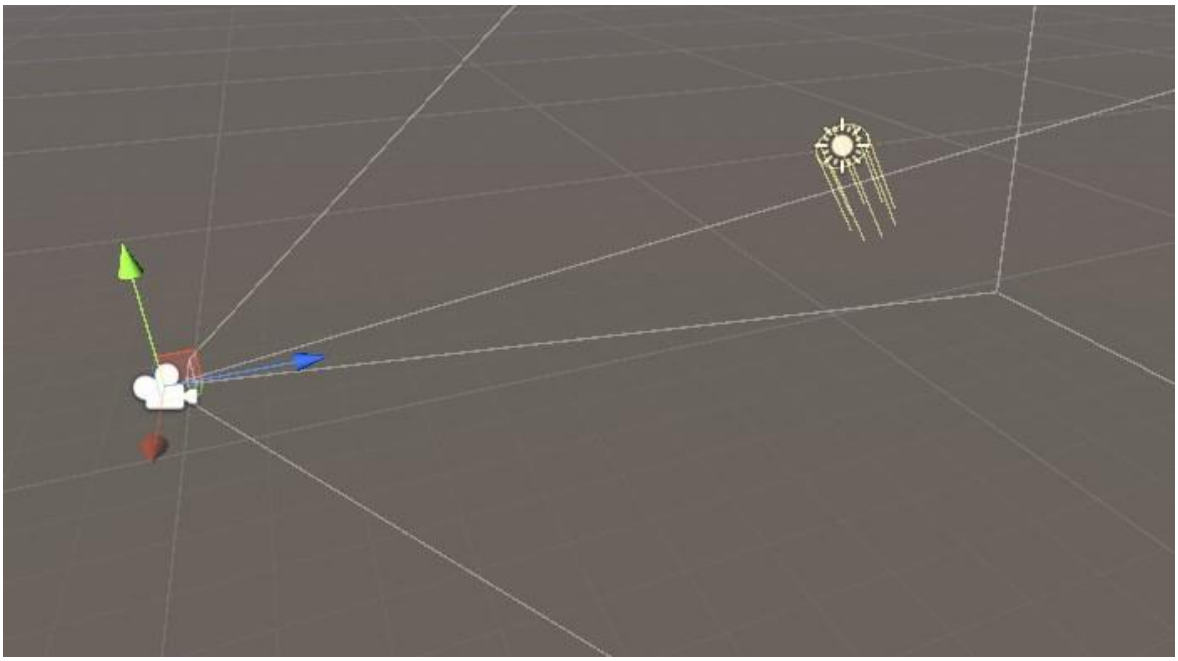


Рисунок 1.20. Стартові об’єкти в новому проекті Unity

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

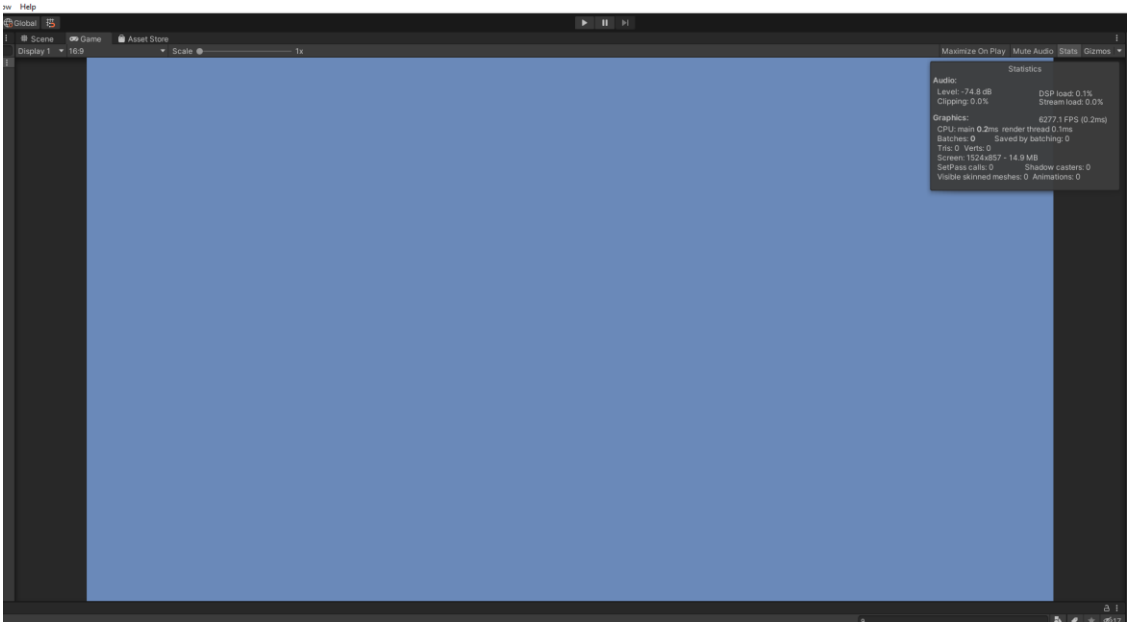


Рисунок 1.21. Ігрова сцена із зміненими характеристиками фону камери гри

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, драбин 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 1.22 можна побачити графічні елементи для створення рівнів.

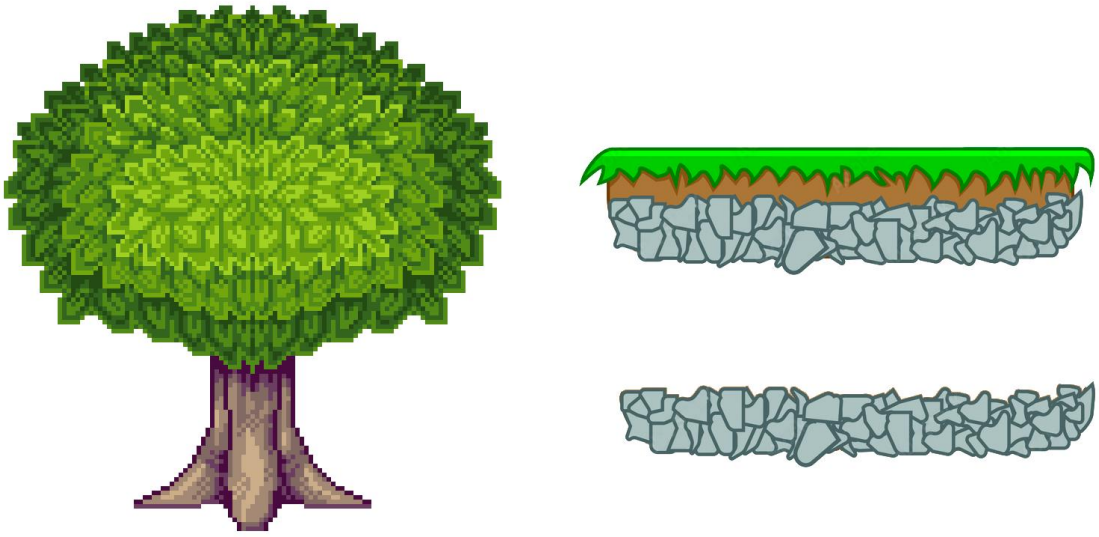


Рисунок 1.22. Графічні елементи для створення рівнів гри

Таким же чином у Adobe Photoshop створювались спрайти для гравця, частин рівню та об’єктів для взаємодії, як драбина або перемикач. Всі реалізовані спрайти можна переглянути на рисунку 1.23.

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.

Рисунок 1.23. 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. Такий підхід потрібно використовувати через те, що написаний у Unity код працює лише тоді, коли він доданий до якогось ігрового об’єкту. 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. На рисунку 1.24 зображено зібраний тестовий рівень гри адвенчури FarmerTrouble:

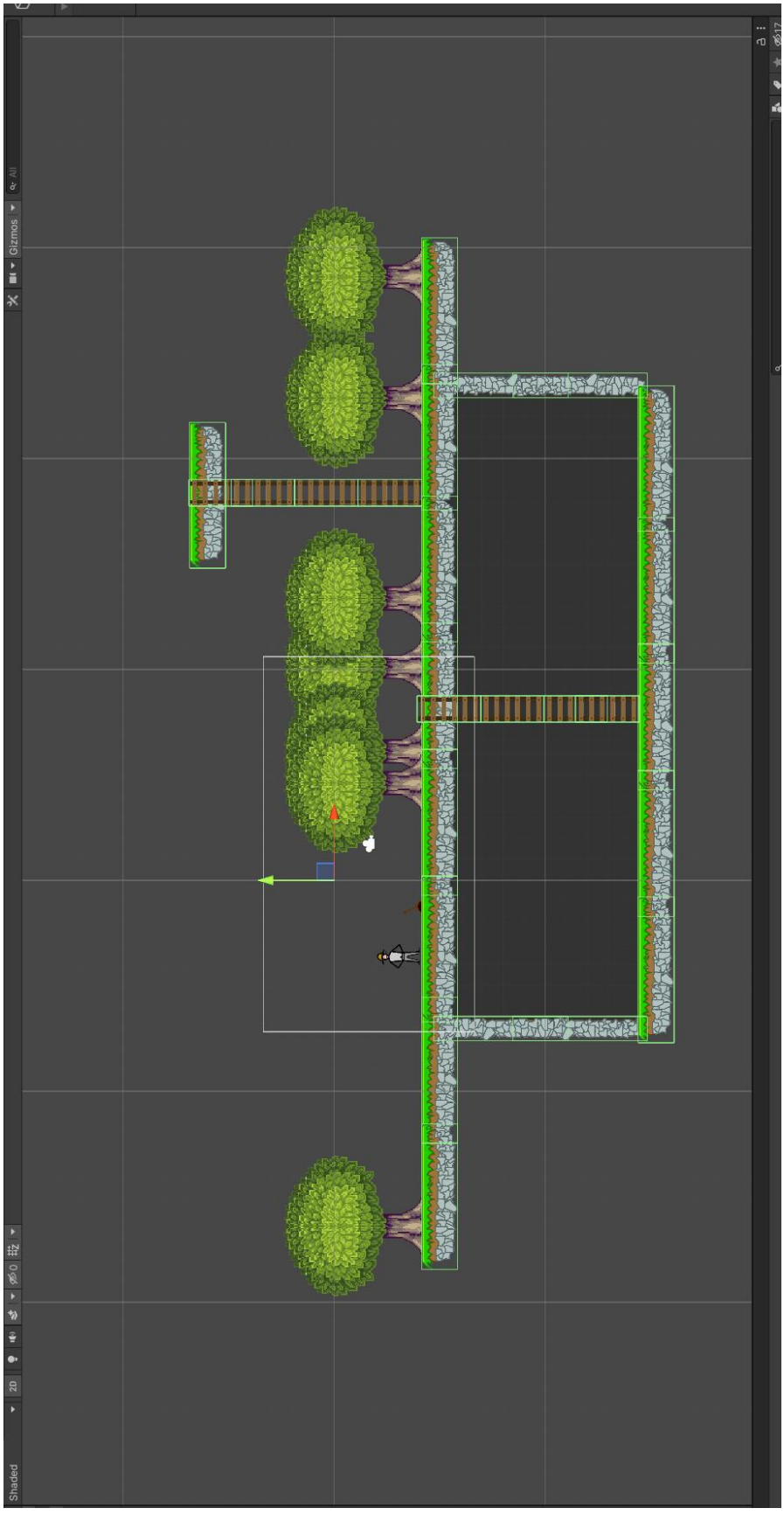


Рисунок 1.24. Тестовий рівень гри адвенчури FarmerTrouble

Коли розміщені всі ігрові об’єкти, елементи рівня, можна переходити безпосередньо до реалізації систем теми дипломного проектування, а саме

системи управління та взаємодії гравця із оточенням. Як вже було зазначено в частини проектування, то на перший погляд потрібно реалізувати два коди. Але це не зовсім так.

Система управління гравцем складається всього з одного скрипта player_movement.cs, оскільки задіює лише ігровий об’єкт гравця та всю необхідну інформацію отримує із об’єктів ігрового програмного рушія Unity. В першу чергу йде про об’єкт Input. Він постійно «слухає» статус всіх кнопок управління, які тільки може зчитувати ігровий програмний рушій Unity. І який би не був їх статус, завжди можна звернутись до цього об’єкту та отримати значення натискання на кнопку, тригер, клавішу або стік. Для управління переміщення гравця я буду використовувати метод GetAxis. Цей метод зчитує так звані осі управління, які задаються у файлі параметрів проекту. Кожна з таких осей має своє ім’я, тому щоби вказати яку саме ось ми хочемо зчитати, потрібно вказати її ім’я у форматі змінної строкового типу string. Цей метод завжди буде повертати значення, не зважаючи на те, чи була натиснута кнопка, чи ні, відрізняється лише значення, яке цей метод поверне. Якщо гравець не використовує ось до якої буде виконано звернення, то метод поверне значення «0». Також, для реалізації руху я буду використовувати компонент твердого тіла RigidBody, який додаю до ігрового об’єкту гравця. У цього компоненту є властивість velocity, яка виконує рух твердого тіла у просторі у визначеному векторі руху. Нижче приведено код, який реалізує рух гравця:

```
void Update()
{
   i_hor = Input.GetAxis("Horizontal");
   i_ver = Input.GetAxis("Vertical");
}

private void FixedUpdate()
{
      _player_rb.velocity = new Vector3(i_hor * 2f, 0f, 0f);
}
```

Як можна побачити з приведеного коду, спочатку у методі Update() отримується значення осей по горизонталі та вертикалі. Після чого рух

виконується у методі FixedUpdate(). Таке розділення зумовлене особливостями цих двох методів. Перший метод відтворює свій зміст кожний кадр, який може відтворити робоча машина. Якщо буде відтворено 50000 кадрів, то зміст цього методу буде виконано 50000 разів. FixedUpdate() є версією методу Update(), який намагається виконатись фіксовану кількість разів – 60. Ці два методи потрібно використовувати уважно, оскільки вони впливають на ігровий процес корінним чином. Наприклад рух об’єктів у Update() буде виконуватись значно швидше, ніж у FixedUpdate() та об’єкт буде проходити далі за секунду часу. І навпаки, зчитування кнопок найкраще проводити у методі Update(), оскільки він частіше звертається до слухання натиснутих кнопок за рахунок кількості відтворень свого коду.

На даний момент часу цього коду вистачить для переміщення гравця по рівню з однієї сторони у іншу. Але цього не достатньо для переміщення по драбинам верх, чи вниз. Для реалізації цього функціоналу, потрібно створити систему взаємодії персонажу з оточенням. Для цього створюється новий скрипт player_interaction.cs.

Принцип роботи взаємодії гравця із оточенням описано у проектній частині дипломного проекту. Там я розглядав приклад взаємодії ігрового об’єкту гравця та ігрового об’єкту для взаємодії. Для цих цілей я використовую у схемі компоненти коллайдери. Перший коллайдер, як фізичну границю ігрового об’єкту гравця із твердим тілом. Другий коллайдер виступає у ролі тригеру. Обидва компоненти додаються до ігрового об’єкту гравця та налаштовуються через інспектор.

Ці коллайдери будуть виконувати дуже важливу роль у системі взаємодії гравця із оточенням, оскільки саме за допомогою них програма гри буде розуміти, коли гравець знаходиться поряд із об’єктом для взаємодії. Чи може гравець виконувати взаємодію із оточенням та інше. Це можливе саме через поєднання властивостей компонентів коллайдерів фізичних границь та тригерів. На рисунку 1.25 зображено зміст ігрового об’єкту гравця із двома компонентами коллайдерами у різних режимах роботи:

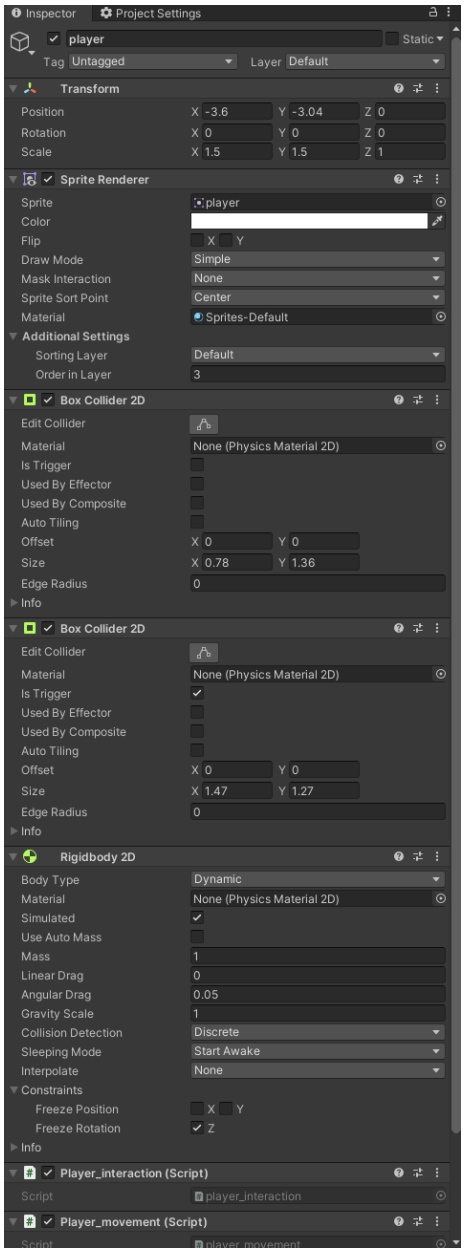


Рисунок 1.25 зміст ігрового об’єкту гравця із двома компонентами коллайдерами у різних режимах роботи

Тепер, коли компоненти додані, я напишу код, який буде використовувати їх. У скрипті player_interaction.cs головними складовими є внутрішні тригери – змінні типу boolean, які отримують свої значення у залежності від того, чи може виконуватись взаємодія із ігровим об’єктом. Наприклад є два види драбин – ті що ведуть наверх, та ті що ведуть вниз. Для їх використання драбинам потрібно придати властивість – тег – із відповідним іменем. Для драбин, що дають змогу лізти наверх – ladder_up, а для других – ladder_down.

У кожної драбини є свій коллайдер-тригер, так само, як і у гравця. Коли

ігровий об’єкт гравця підходить до драбини спрацьовує тригер, що реагує у разі, якщо гравець залишається поряд із драбиною. Далі скрипт player_interaction.cs, який знаходиться на ігровому об’єкті гравця, перевіряє із яким тегом ігровий об’єкт коллайдера, з котрим він взаємодіє. Якщо тег співпадає із тегом драбин, тоді відповідній змінній стану находження порад з драбиною передається команда перемкнутися у протилежний собі стан. Якщо гравець виходить із взаємодії коллайдерами із драбиною, то перевіряється із якого коллайдеру виходить тригер гравця, зчитується тег, змінюється стан булевої змінної на протилежний. Цей код приведено нижче:

```
bool ladder_up = false;
bool ladder_down = false;
bool switch_zone = false;

private void OnTriggerStay2D(Collider2D collision)
{
   if (collision.tag == "ladder_up")
      ladder_up = true;
   else if (collision.tag == "Ladder_down")
      ladder_down = true;
}

private void OnTriggerExit2D(Collider2D collision)
{
   if (collision.tag == "ladder_up")
      ladder_up = false;
   else if (collision.tag == "Ladder_down")
      ladder_down = false;
}
```

Тепер, коли я знаю, з яким об’єктом взаємодіє гравець, можна викликати дії, або давати змогу виконувати заблоковані функції. Наприклад, переміщення гравця виконується лише вліво та вправо. Знаходячись поряд із драбиною, гравець отримує можливість лізти по драбині верх, або вниз. Для цього код взаємодії має повертати значення статусу знаходження поряд із драбинами. Код приведено нижче:

```
public bool return_ladder_up()
{
   return ladder_up;
```

```
}

public bool return_ladder_down()
{
   return ladder_down;
}
```

Тепер, коли можна отримати ці значення зі скрипту взаємоді, потрібно внести відповідні зміни до коду переміщення гравця. Новий код буде виглядати наступним чином:

```
void Update()
{
   i_hor = Input.GetAxis("Horizontal");
   i_ver = Input.GetAxis("Vertical");
}

private void FixedUpdate()
{
   if((_player_interaction.return_ladder_up()) || (_player_interaction.
return_ladder_down()))
   {
      _player_rb.velocity = new Vector3(i_hor * 2f, i_ver * 2f, 0f);
   }
   else
      _player_rb.velocity = new Vector3(i_hor * 2f, 0f, 0f);
}
```

Як можна побачити, змін набули лише у коді руху, оскільки тепер скрипт перевіряє, як йому рухатись. Якщо є позитивна відповідь від будь-якої драбини, то гравець може рухатись не тільки вліво та вправо але й вниз верх. У іншому випадку, використовується стара схема руху ігрового об’єкту гравця.

Наступним є реалізація функціоналу взаємодії при натисканні клавіші клавіатури E та кнопки миші – лівої. Для цього я створив перемикач на рівні, при взаємодії з котрим перемикається день та ніч. Також цей перемикач повинен мати змогу переключати свій зовнішній вид з одного стану на інший. Тому потрібно написати код для об’єкту взаємодії: switch_logic.cs. Цей об’єкт стане першим у ряді таких об’єктів. Нижче приведено код:

```
[SerializeField] Camera _main_camera;
[SerializeField] Sprite _switch_1;
[SerializeField] Sprite _switch_2;
```

```
SpriteRenderer _switch_sprite_renderer;
public bool switch_state = false;

void Start()
{
   _switch_sprite_renderer = GetComponent<SpriteRenderer>();
}

public void _interact_switch()
{
   switch_state = !switch_state;
   if (switch_state)
      _switch_sprite_renderer.sprite = _switch_2;
   else
      _switch_sprite_renderer.sprite = _switch_1;
   if (switch_state)
      _main_camera.backgroundColor = new Color(0.1f, 0.1f, 0.1f);
   else
      _main_camera.backgroundColor = new Color(0.41f, 0.53f, 0.72f);
}
```

Даний код отримує дві версії спрайту перемикача, а також посилання на камеру. Доступ до компоненту відрисовки спрайту об’єкту. Безпосередньо сам скрипт додається до перемикача, тому він зможе вільно отримувати доступ до компонентів перемикача. Також скрипт перемикача має змінну логічного типу-перемикач, яка відповідає за статус перемикача. При викликанні методу public void _interact_switch() буде виновуватись перемикання змінної перемикача. У залежності від статусу перемикача, будуть у компонент відрисовки перемикача буде завантажуватись новий спрайт, а потім змінюватись колір фону камери, що буде викликати перемикання для та ночі.

Тепер потрібно прописати взаємодію у скрипту інтеракції - player_interaction.cs. Принцип той самий. Зчитуємо перетин із коллайдером перемикача, отримуємо його тег. Змінюємо логічну зміну. При виході повторюємо. Реалізовуємо передачу стану перемикача та ігровий об’єкт перемикача, оскільки потрібно знати, в якому саме перемикачі викликати код. Код представлено нижче:

```
private void OnTriggerStay2D(Collider2D collision)
{
```

```
        if (collision.tag == "ladder_up")
            ladder_up = true;
        else if (collision.tag == "Ladder_down")
            ladder_down = true;
        else if (collision.tag == "switch")
        {
            switch_zone = true;
            collision_game_object = collision.gameObject;
        }
    }
    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.tag == "ladder_up")
            ladder_up = false;
        else if (collision.tag == "Ladder_down")
            ladder_down = false;
        else if (collision.tag == "switch")
        {
            switch_zone = false;
        }
    }
    public bool return_switch_zone()
    {
        return switch_zone;
    }
    public GameObject return_switch_go()
    {
        return collision_game_object;
    }
```

У коді управління гравцем вносимо зміну у код зчитування натискання на клавіші клавіатури. Якщо отримується позитивний статус взаємодії із перемикачем, якщо натиснута кнопка E на клавіатурі, то отримується ігровий об’єкт перемикача, та викликається метод перемикання. Код приведено нижче:

```
    void Update(){
        i_hor = Input.GetAxis("Horizontal");
        i_ver = Input.GetAxis("Vertical");
        if (_player_interaction.return_switch_zone() && Input.GetKeyDown
(KeyCode.E)) {
            GameObject switch_go = _player_interaction.return_switch_go();
            switch_logic _switch_logic = switch_go.GetComponent<switch_logic>();
            _switch_logic._interact_switch();
        }
    }
```

Тепер гравець має змогу вільно переміщуватись по рівню, використовувати елементи оточення, змінювати режими своєї роботи в залежності від ігрових об’єктів поряд, а також використовувати об’єкти для отримання якихось ефектів. Важливо розуміти, що поточна реалізація гри не є кінцевою, але реалізація системи переміщення та взаємодії гравця із оточенням реалізована таким чином, що її архітектура дозволяє у майбутньому реалізовувати нові і нові механіки взаємодії із оточенням, та створювати додаткові об’єкти для використання. Наприклад двері можна відчиняти за тим же принципом, що і перемикати перемикач.

Після реалізації всіх основних кодів системи управління та взаємодії гравця із оточенням, а також спрайтів гри, проект отримав свою кінцеву структуру, яку можна побачити на рисунку 1.26:

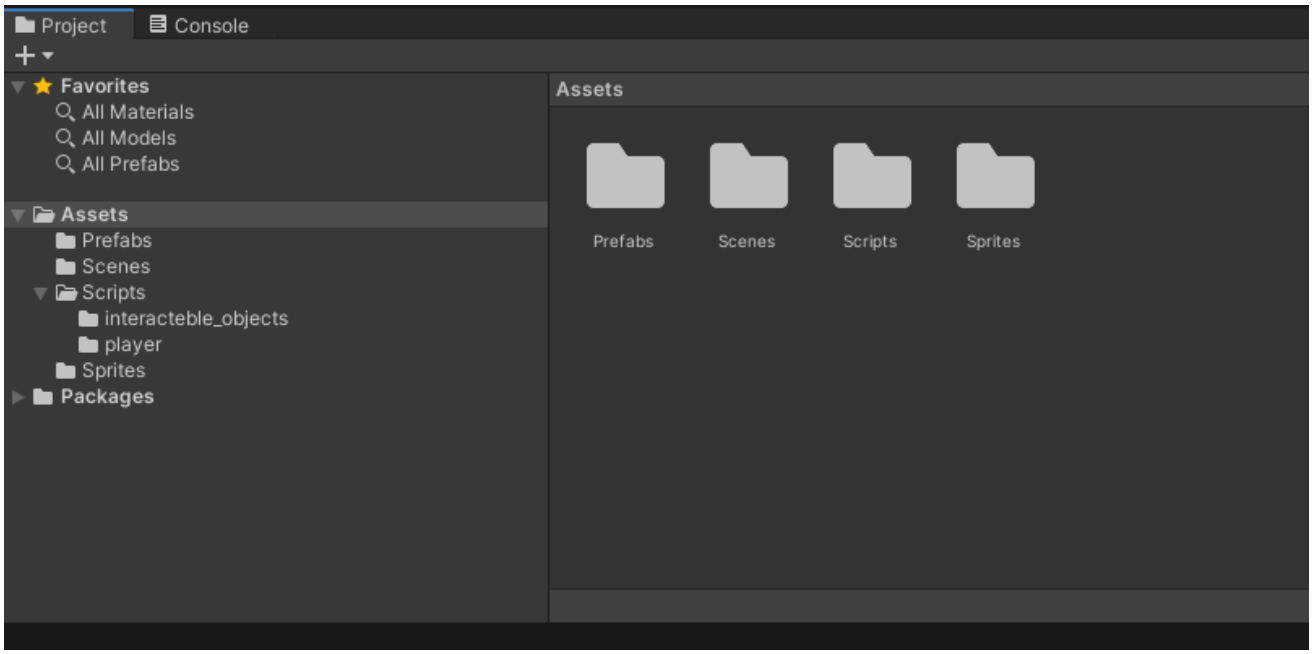


Рисунок 1.26 Кінцева структура проекту гри

Як можна бачити всі основні файли, з якими проводилась робота розміщені у підкаталогах. Також до складу проекту входять файли ігрового програмного рушія Unity. Всі файли проекту було розподіллено по підкаталогам розділу проекту Assets, розміщення котрих можна побачити на рисунку 1.27.

Дана реалізація проекту дає змогу у подальшому використовувати систему управління та взаємодії гравця із оточенням, переміщуватись по ігровому простору, взаємодіяти з об’єктами на рівні

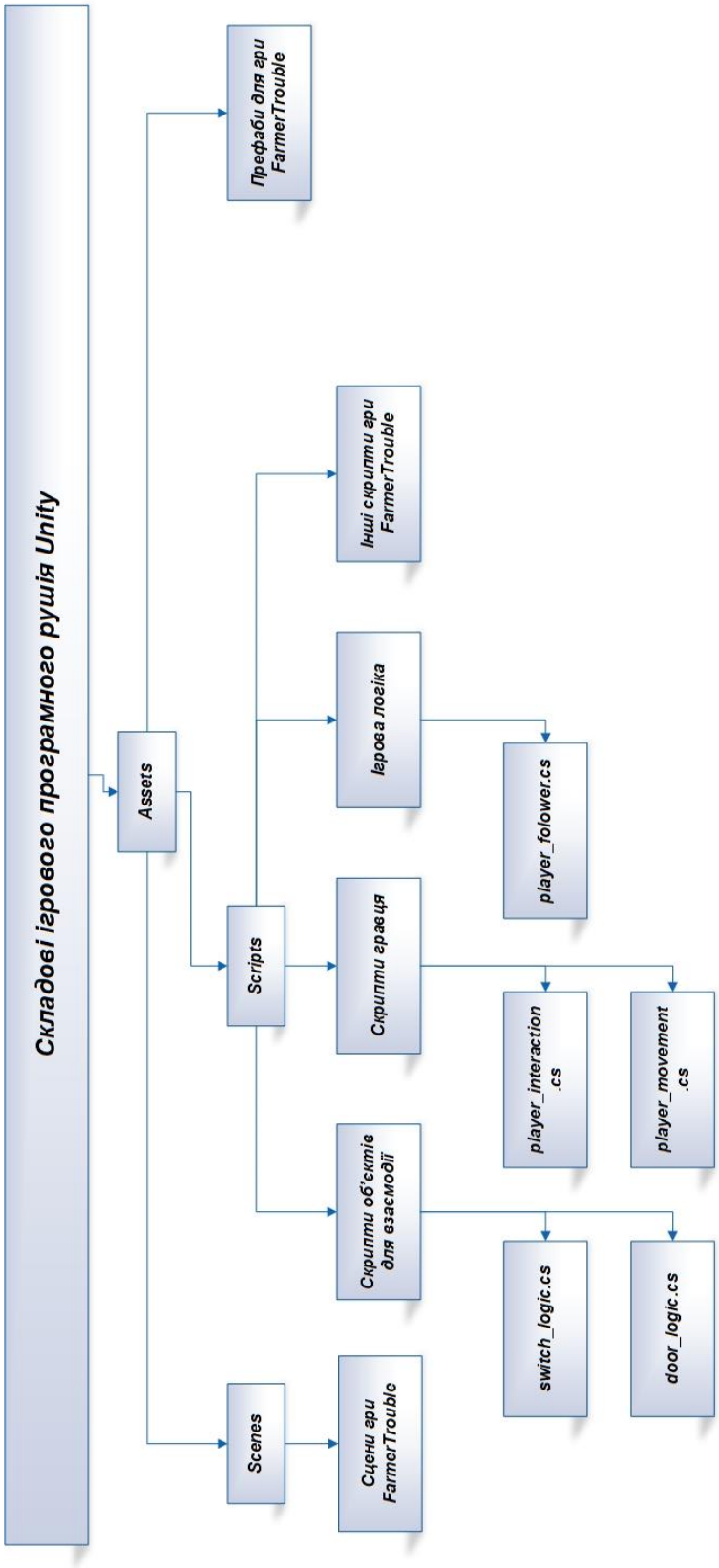


Рисунок1.27. Розміщення файлів проекту по підкаталогам розділу Assets

1.7 Тестування працездатності реалізованих елементів

Тестування та відладка є важливою частиною розробки будь-якого програмного продукту. Для ігрових проектів виконання тестування є ще більш

важливим процесом, оскільки він дозволяє знаходити помилки наявність котрих не є очевидною. Треба розуміти, що у ігрових проектах помилки можуть призводити до візуальних або внутрішніх проблем. Перші знаходяться дуже легко, оскільки візуальні артефакти, або помилки у розмішені об’єктів, дуже швидко знаходяться під час виконання ігрового процесу. Інша річ програмні помилки, які неможливо візуально оприділити під час виконання гри. Іноді такі помилки призводять до часткового порушення ігрового процесу, а в більш рідких випадках до повної неможливості пройти гру, адже можуть зачепляти механіки, які пов'язані з переходом до іншого рівня, або запуском ключового скрипту у сюжетній послідовності.

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 1.28 можна побачити процес тестування гри. Виявлено помилку із переміщенням по драбині вниз, яка була виправлена зміною параметрів компонентів землі та драбини.

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 управління та взаємодії гравця із ігровим оточенням. Було перевірено переміщення гравця по рівню, можливість вертикального переміщення по рівню. Використання перемикачів та дверей. Помилок у роботі основних ігрових механік виявлено не було. Було проведено балансування параметрів швидкості переміщення гравця та параметрів колайдерій гравця та предметів для взаємодії.

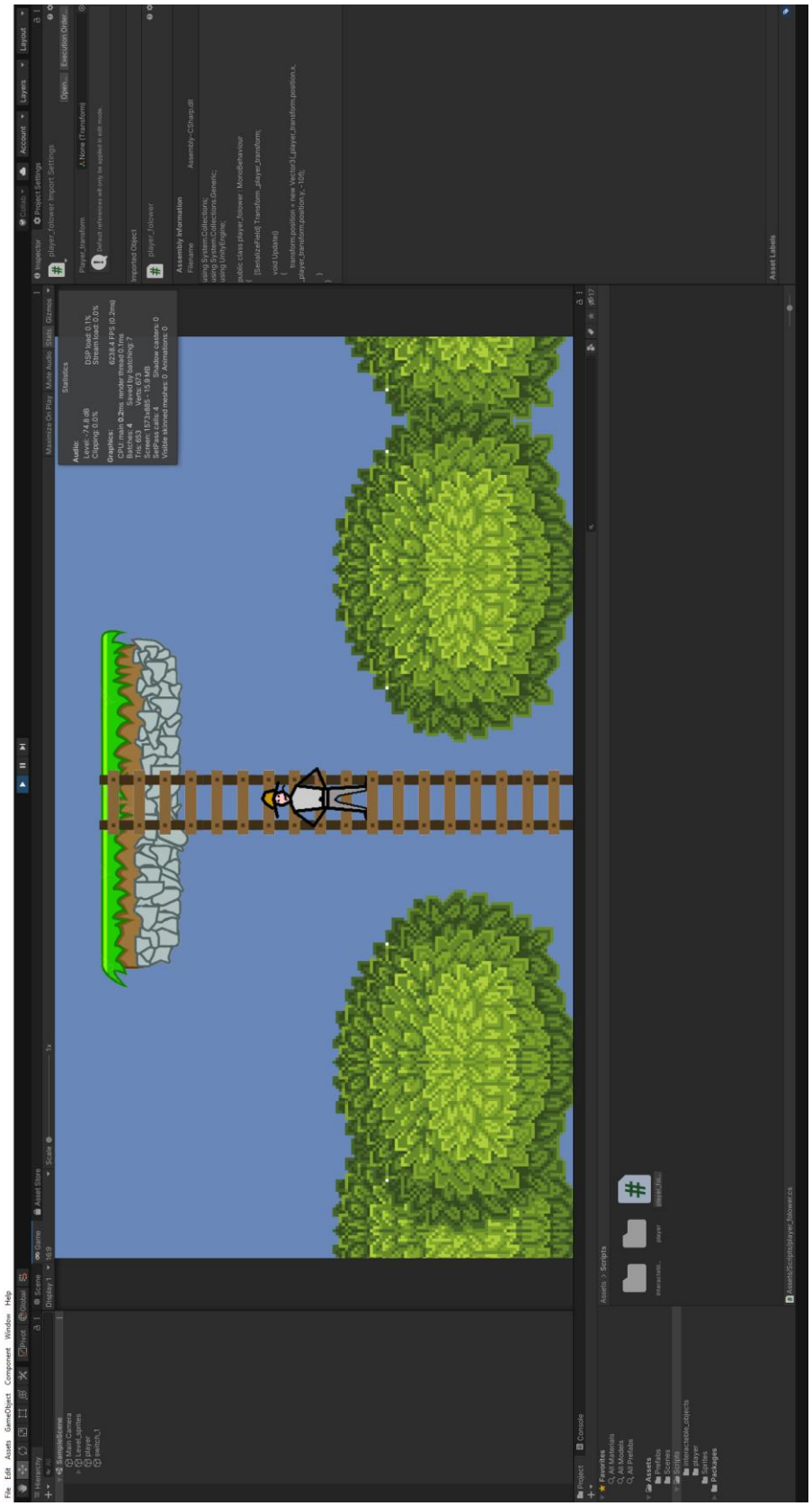


Рисунок 1.28. Процес тестування гри

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 було виконано ряд заходів для пошуку найрозповсюджених та робочих систем управління та взаємодії гравця із оточенням у іграх в жанрі адвенчура. Отримавши потрібні дані, було розроблено перелік необхідного функціоналу для гравця у майбутній грі.

Оскільки система управління є невід'ємною від інших ігрових елементів, було спроектовано модулі для управління та взаємодії з оточенням, які складались із різних скриптів, що виконували свої функційні завдання. Спроектовані модулі були створені таким чином, щоби їх можна було легко модифікувати, або вносити виправлення у випадку виявлення помилок.

Реалізація системи управління виконувалась на мові програмування C#, оскільки ігровий програмний рушій Unity повністю перейшов тільки на цю мову програмування. Для редагування коду використовувалась середа розробки Microsoft Visual Studio. Код для системи управління виконувався в окремих скриптах, відповідно до тих модулів, що було спроектовано на первинних етапах роботи.

Реалізована система була імплементована у ігровий процес шляхом додавання до ігрових об'єктів, які відповідали за роботу цих елементів гри, починаючи від ігрового об'єкта гравця, закінчуючи кінцевими слухачами подій у ігровому світі.

Відладка проекту дала змогу знайти та виправити сховані помилки, які не були знайдені компілятором ігрового програмного рушія. Такі помилки були виправлені.

Як результат роботи було реалізовано систему управління гравцям та взаємодії з оточенням у грі в жанрі адвенчура, відповідно до того, які вимоги було висунуто від дизайн документу 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 управління та взаємодії гравця із оточенням на мові C#

Скрипт switch_logic.cs

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

public class switch_logic : MonoBehaviour
{
    SpriteRenderer _switch_sprite_renderer;
    [SerializeField] Camera _main_camera;
    [SerializeField] Sprite _switch_1;
    [SerializeField] Sprite _switch_2;
    public bool switch_state = false;

    void Start()
    {
        _switch_sprite_renderer = GetComponent<SpriteRenderer>();
    }

    public void _interact_switch()
    {
        switch_state = !switch_state;
        if (switch_state)
            _switch_sprite_renderer.sprite = _switch_2;
        else
            _switch_sprite_renderer.sprite = _switch_1;
        if (switch_state)
            _main_camera.backgroundColor = new Color(0.1f, 0.1f, 0.1f);
        else
            _main_camera.backgroundColor = new Color(0.41f, 0.53f, 0.72f);
    }
}
```

Скрипт player_interaction.cs

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

public class player_interaction : MonoBehaviour
{
    bool ladder_up = false;
    bool ladder_down = false;
```

```
bool switch_zone = false;
float i_ver = 0f;
Rigidbody2D _player_rb;
GameObject collision_game_object;

void Start()
{
    _player_rb = GetComponent<Rigidbody2D>();
}

private void Update()
{
    i_ver = Input.GetAxis("Vertical");
}

private void FixedUpdate()
{
    if (ladder_up)
    {
        _player_rb.velocity = new Vector3(0f, i_ver * 2f, 0f);
    }
    if (ladder_down)
    {
        _player_rb.velocity = new Vector3(0f, i_ver * 2f, 0f);
    }
}

private void OnTriggerStay2D(Collider2D collision)
{
    if (collision.tag == "ladder_up")
        ladder_up = true;
    else if (collision.tag == "Ladder_down")
        ladder_down = true;
    else if (collision.tag == "switch")
    {
        switch_zone = true;
        collision_game_object = collision.gameObject;
    }
}

private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.tag == "ladder_up")
```

```
            ladder_up = false;
        else if (collision.tag == "Ladder_down")
            ladder_down = false;
        else if (collision.tag == "switch")
        {
            switch_zone = false;
        }
    }

    public bool return_ladder_up()
    {
        return ladder_up;
    }

    public bool return_ladder_down()
    {
        return ladder_down;
    }

    public bool return_switch_zone()
    {
        return switch_zone;
    }

    public GameObject return_switch_go()
    {
        return collision_game_object;
    }
}
```

Скрипт player_movement.cs

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

public class player_movement : MonoBehaviour
{
    player_interaction _player_interaction;
    Rigidbody2D _player_rb;
    float i_hor = 0f;
    float i_ver = 0f;

    void Start()
    {
```

```
        _player_rb = GetComponent<Rigidbody2D>();
        _player_interaction = GetComponent<player_interaction>();
    }

    void Update()
    {
        i_hor = Input.GetAxis("Horizontal");
        i_ver = Input.GetAxis("Vertical");
        if (_player_interaction.return_switch_zone() && Input.GetKeyDown
(KeyCode.E))
        {
            GameObject switch_go = _player_interaction.return_switch_go();
            switch_logic _switch_logic = switch_go.GetComponent
<switch_logic>();
            _switch_logic._interact_switch();
        }
    }

    private void FixedUpdate()
    {
        if((_player_interaction.return_ladder_up()) ||
(_player_interaction.return_ladder_down()))
        {
            _player_rb.velocity = new Vector3(i_hor * 2f, i_ver * 2f, 0f);
        }
        else
            _player_rb.velocity = new Vector3(i_hor * 2f, 0f, 0f);
    }
}
```

Скрипт player_folower.cs

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

public class player_folower : MonoBehaviour
{
    [SerializeField] Transform _player_transform;

    void Update()  {
        transform.position = new Vector3(_player_transform.position.x,
_player_transform.position.y, -10f);
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Реалізація системи управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity

ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА ГРУПИ 4КС-57: ЗАЙЦЕВА ДМИТРА 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

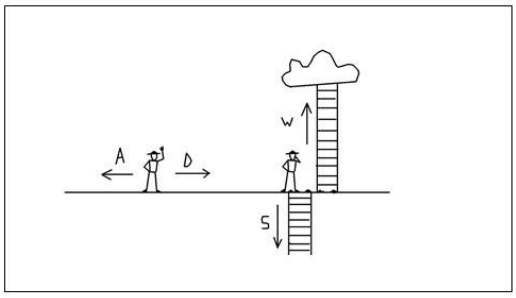
Ігровий двигун Unity є одним із популярніших ігрових двигунів у світі. За його допомогою створюють як мобільні, так і комп'ютерні ігри. Цей двигун досі активно розвивається!

- 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 управління проекту

Механіка управління у проекті основана на жанрових особливостях.

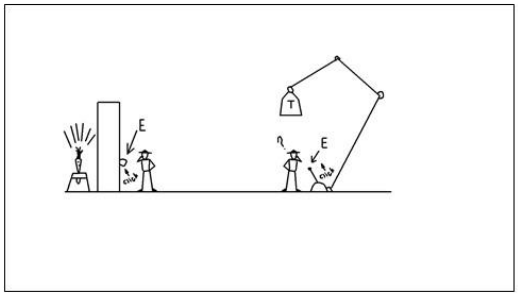
- Створення умов для неквапливого переміщення;
- Відсутність управління «різких» рухів;
- Горизонтальне та вертикальне переміщення;
- Ситуативність у системі управління;
- Прийняття участі у системі взаємодії з оточенням.



Особливості механіки взаємодії з оточенням

Механіка взаємодії з оточенням у проекті основана на жанрових особливостях та вимогах концепту.

- Для взаємодії з оточенням потрібно знаходитись поряд з об’єктом оточення;
- Можливість використовувати для взаємодії кнопку миші;
- Модульність у додаванні нових об’єктів до системи;
- Прийняття участі у системі управління.



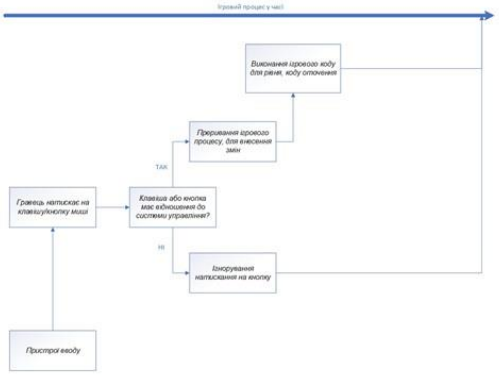
Огляд елементів системи управління та взаємодії з оточенням

Основними елементами системи управління та взаємодії з оточенням є скрипти, які реалізують ці механіки.

В свою чергу вони пов’язані із пристроями вводу.

Для отримання даних щодо натиснутих кнопок використовується об’єкт Input.

Код системи управління та код системи взаємодії з оточенням постійно у роботі.



Принцип роботи елементів системи управління та взаємодії з оточенням

Система управління, або система управління гравцем існує поряд із іншими модулями гри.

Принцип роботи оснований на наступних кроках:

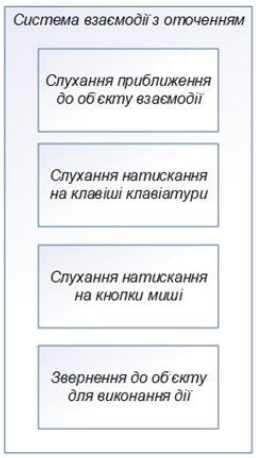
- Зчитувати натискання кнопок управління гравцем;
- Виконання розрахунків для переміщення гравця;
- Виконання переміщення гравця у просторі гри.

Принцип роботи елементів системи управління та взаємодії з оточенням

Система взаємодії з оточенням, так як система управління, існує поряд із іншими модулями гри.

Принцип роботи оснований на наступних кроках:

- Система слухає приближення гравця до об’єкту взаємодії;
- Система слухає клавіші клавіатури, які натискає гравець;
- Система слухає кнопки миші, які натискає гравец;
- Ідентифікує та звертається до об’єкту для дії.



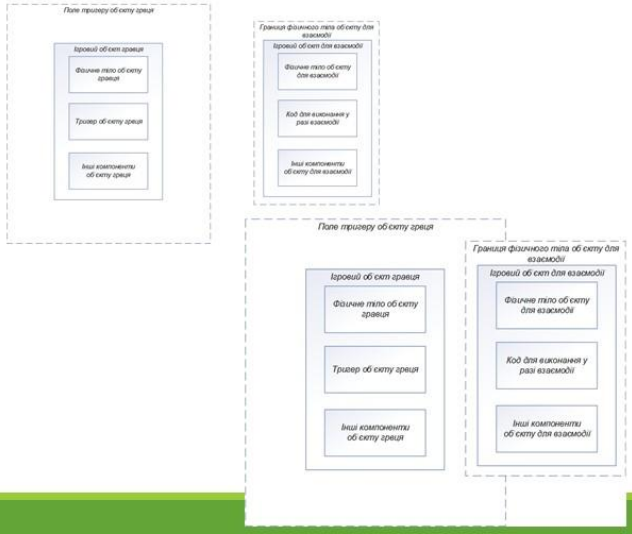
Принцип роботи елементів системи управління та взаємодії з оточенням

Система взаємодії з оточенням визначає наближення до об’єктів взаємодії за допомогою коллайдерів-тригерів.

Коллайдери-тригери взаємодіять із іншими коллайдерами.

Коллайдери тригери викликають подію, та отримають посилання на компонент ігрового об’єкта, з котрим може бути взаємодія.

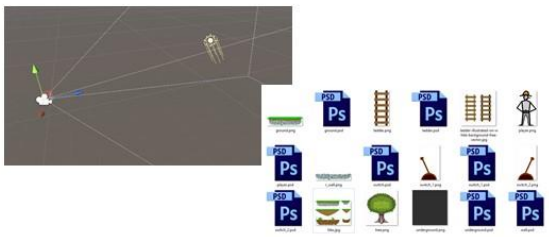
При натисканні на кнопку взаємодії – об’єкт викликається через посилання.



Етапи реалізації елементів системи управління та взаємодії з оточенням

Реалізація елементів управління та взаємодії з оточенням складалась з:

- Реалізації спрайтів для рівня гри;
- Створення рівня із спратів;
- Створення ігрового об’єкту гравця;
- Написання коду інших модулів гри;
- Написання коду систем управління та взаємодії;
- Тестування.

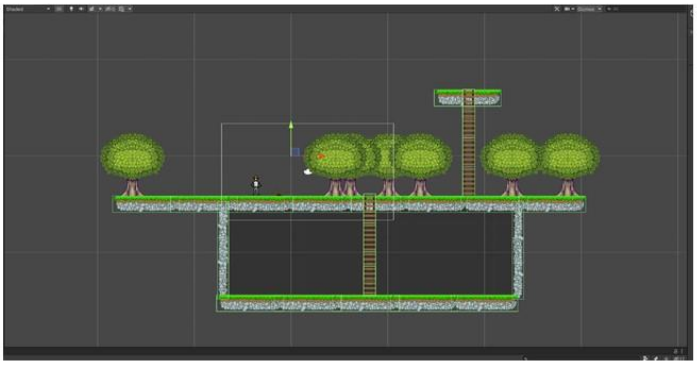


Хід тестування гри

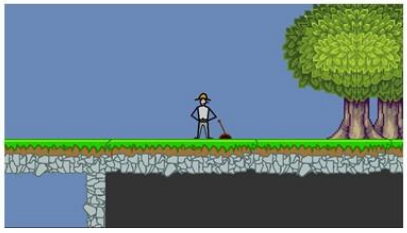
Тестування гри проводилось з допомогою вбудованого у ігровий програмний рушій Unity інструмент.

Метою тестування було:

- Визначити коректність управління ігровим об’єктом гравця;
- Визначити коректність роботи системи управління гравцем;
- Визначити коректність роботи системи взаємодії з оточенням.



Скріншоти реалізованої гри



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Зайцева Дмитра 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 управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 71 сторінок. У пояснювальній записці виконано опис предметної області, способи реалізації систем управління та взаємодії гравця з оточенням. Проведено проектування та реалізація намічених елементів гри в жанрі адвентчура. 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 Зайцев Д.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 Зайцев Д.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 Зайцев Д.О. мав змогу самостійно приймати рішення з реалізації систем управління та взаємодії гравця з оточенням гри в жанрі адвенчура, 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

Зайцева Дмитра 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 управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity*

Обсяг розрахунково-пояснювальної записки 71 сторінок

Обсяг графічної (презентаційної) частини 12 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

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

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

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі розглянуті питання проблематики розробки гри на ігровому програмному рушії Unity, сформовано вимоги до гри згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів розробляемої 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 управління та взаємодії гравця із оточенням;
2. Виконано проектування намічених елементів гри із поясненнями на схемах;
3. Розроблено функціонуючу систему управління та взаємодії із оточенням для гри.

д) основні недоліки дипломного проекту (роботи)

1. Кількість пророблених інтеракцій невелика;
2. Відсутня графічний супровід процесу взаємодії гравця із оточенням.
3. Присутні деякі помилки оформлення. На деяких скріншотах неможливо розібрати вміст.

Оцінка розрахункової частини Добре

Оцінка графічної частини Добре

Загальна оцінка Добре

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

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

Підпис

14 червня 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016341009

Дата перевірки:
10.06.2024 09:01:01 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.06.2024 14:14:26 EEST

ID користувача:
100011688

Назва документа: КС-57_Зайцев

Кількість сторінок: 44 **Кількість слів:** 8896 **Кількість символів:** 62384 **Розмір файлу:** 2.40 MB **ID файлу:** 1016142177

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.6%
Схожість

Найбільша схожість: 3.7% з Інтернет-джерелом (https://card-file.ontu.edu.ua/server/api/core/bitstreams/6c95086b-bffe...

5.6% Джерела з Інтернету | **78** ..... Сторінка 46

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,

Зайцев Дмитро 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 управління та взаємодії гравця із оточенням гри у жанрі адвенчура на програмному рушії Unity» (автор роботи – Зайцев Д.С., 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 _____ / Зайцев Д.С./

Керівник _____ / Шувалова І.О./

«10» 06 2024 р.