Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)

Кафедра обчислювальної техніки
(повна назва кафедри)

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр
(освітньо-кваліфікаційний рівень)

на тему:

Методи ігрового дизайну при розробці Масової Багатокористувацької Онлайнової Рольової Гри

Виконав: студент 2 курсу, групи КІ —19м
спеціальності:
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки)

Жаврук Ярослав Сергійович
(прізвище та ініціали)

Керівник: к.т.н., доц. Тарновський М.Г.
(прізвище та ініціали)

Вінниця 2020

Вінницький національний технічний університет

(повне найменування вищого навчального закладу)

Факультет Інформаційних технологій та комп'ютерної інженерії

Кафедра Обчислювальної техніки

Освітньо-кваліфікаційний рівень магістр

Спеціальність 123 «Комп'ютерна інженерія»

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____

д.т.н., професор Азаров О. Д.

"____" _____20___року

З А В Д А Н Н Я

на магістерську кваліфікаційну роботу

Жавруку Ярославу Сергійовичу

(прізвище, ім'я, по батькові)

1 Тема магістерської кваліфікаційної роботи: Методи ігрового дизайну при розробці Масової Багатокористувацької Рольової Онлайн гри

керівник магістерської кваліфікаційної роботи: Тарновський М.Г., к. т. н., доцент кафедри ОТ.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "___"_____20__року №___

2 Строк подання студентом роботи_____

3 Вихідні дані до роботи Жанр гри – Масова Багатокористувацька Онлайнова Рольова Гра, транзитивний метод балансування, мова програмування Python, середовище розробки PyCharm;

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Обгрунтування доцільності роботи. Розробка масової багатокристувацької рольової онлайн гри для месенджера телеграм. Розробка програмних елементів. Тестування.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Алгоритми генерації унікальниї артефактів та пересування між локаціями, алгоритми бойової системи, формула балансування транзитивним методом, формула балансування характеристик, графічні схеми інтерфейсів програми

6 Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Тарновський Микола Генадійович, к. т. н., доцент	_____ дата	_____ дата
		_____ підпис	_____ підпис
5	Руда Лілія Петрівна к. е. н., доцент	_____ дата	_____ дата
		_____ підпис	_____ підпис

7 Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Пошук та огляд інформаційних джерел	07.09.20р.	виконано
2	Обгрунтування доцільності розробки	20.09.20р.	виконано
3	Аналіз сновних принципів ігрового дизайну	01.10.20р.	виконано
4	Розробка ігрового балансу та програмних елементів гри	09.10.20р.	виконано
5	Економічна частина	18.11.20р.	виконано
6	Оформлення пояснювальної записки і презентації	20.11.20р.	виконано
7	Попередній захист	23.11.20р.	виконано

Студент _____ Жаврук Я.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тарновський М.Г.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дана магістерська кваліфікаційна робота присвсячена аналізу методів геймдизайну, формалізації процесу розробки масової багатокористувацької рольової онлайн гри.

У магісерській роботі було проаналізовано ринок ігрової індустрії, сучасні підходи до розробки ігор,і обгрунтована актуальність теми. Було розроблено ігровий баланс для недетермінованої гри транзитивним методом. Було розроблено основні алгоритми роботи програми, алгоритм тестового бою та структура інтерфейсу для ігрового бота Telegram. Проведено тестування.

ANNOTATION

This master's qualification work is devoted to the analysis of game design methods, formalization of the process of developing a mass multiplayer role-playing online game.

The thesis analyzed the market of the gaming industry, modern approaches to game development and substantiated the relevance of the topic. Game balance was developed for non-deterministic game by transitive method. The main algorithms of the program, the test battle algorithm and the interface structure for the game bot Telegram were developed. Testing was performed.

ЗМІСТ

ВСТУП

У сучасному світі створення відеоігор є одним з найбільш великих сегментів індустрії розваг. Масштаби ігрової індустрії можна порівняти, наприклад, з кіноіндустрією. А по швидкості росту за останні п'ять років індустрія відеоігор істотно її випереджала. Структура ігрової індустрії по ступені впливу на споживачів і залучення їх в інтерактивне оточення, пропоноване відеоіграми, цей сегмент уже давно виділяється серед інших видів розваг. При цьому безпосередньо в сфері розробки ігор, в цей час, більше всього виділяється сегмент мобільних ігор, який в останні роки отримав величезну популярність і наразі має приблизно половину доходу всієї ігрової індустрії. Попит на мобільні ігри зростає і при цьому правильне використання методів ігрового дизайну, при їх розробці, має дедалі більше значення для успіху гри.

Масова багатокористувацька онлайнова рольова гра (англ. Massively multiplayer online role-playing game, MMORPG) — жанр онлайнових рольових відеоігор, в якій велика кількість гравців взаємодіють у вигаданому автором віртуальному світі. Як і в більшості рольових ігор, гравцеві пропонується роль вигаданого героя і надається можливість управляти його діями. MMORPG відрізняються від однокористувацьких і невеликих мережевих рольових ігор великою кількістю гравців, а також віртуальним світом, який продовжує своє життя за відсутності гравця. Віртуальний світ продумується і підтримується видавцем гри [1].

Головною ідеєю MMORPG є забезпечення можливості граців зручно взаємодіяти між собою. Саме для цього було обрано вже існуючий месенджер Telegram. Головною його перевагою є те, що його використання просте, месенджер займає мало місця, частіше за все використовується на смартфоні і він за умовчуванням використовується задля спілкування і взаємодії між людьми. Це одразу робить вибір платформи Telegram чудовим варіантом для розробки гри. Крім цього також важливою перевагою є те, що у мессенджері Telegram є особливі акаунти, операторами яких можуть бути не люди, а спеціальним чином

написані програми, розташовані на сторонніх ресурсах (не на серверах Telegram). Ці програми-оператори називаються ботами. Боти можуть отримувати адресовані їм повідомлення, а також генерувати і відправляти відповідні повідомлення. Все це вони роблять через свій аккаунт, використовуючи спеціальний API. Саме базуючись на цьому буде розроблятись гра. Додатково, Telegram вже має свою велику аудиторію, що забезпечить легку рекламу і зручність для гравців: у них не буде потреби завантажувати окремий додаток, коли набагато простіше грати там, де й відбувається спілкування. І як ще одна перевага, Telegram дозволяє створювати спільні чати між людьми та канали для викладання деякої інформації, що забезпечить зручну взаємодію гравців.

Наразі існують деякі реалізації ігрових ботів для мессенджеру Telegram, проте у всіх них є свої недоліки. Основними недоліками є те, що інші ігрові боти розроблені в основному ігноруючи більшість принципів ігрового дизайну. По цій причині не дивлячись на попит більшість ігрових проектів на базі Telegram бота моють доволі низьку якість. Тому розробка ігрового Telegram-бота крім можливості дослідження способів вдосконалення методів ігрового дизайну, може представляти конкуренту здатність в цьому сегменті ринку і може в цілому підвищити планку якості і в цілому є досить актуальною задачею.

Отже, **актуальним** є розробка масової багатокористувацької рольової онлайн гри та формалізація методів ігрового дизайну.

Метою магістерської роботи є формалізація процесу розробки масової багатокористувацької рольової онлайн гри

Задачі дослідження магістерської роботи:

— аналіз методів ігрового дизайну;

— вибір методу ігрового дизайну та його формалізація;

— проектування ігрового дизайну для розробки масової багатокористувацької рольової онлайн гри;

— проектування "core"-механіки та математичний опис процесу балансування транзитивним методом та ігрового моделювання";

— тестування отриманого результату.

Об’єкт дослідження — процес розробки ігрового дизайну масової багатокористувацької рольової онлайн гри на базі телеграм-бота

Предмет дослідження — методи та засоби ігрового дизайну при розробці багатокористувацької рольової онлайн гри на базі телеграм-бота та балансування ігрових об’єктів і систем.

Методи дослідження: теорія чисел, методи математичної статистики, теорія ймовірності для генерації унікальних персоналізованих артефактів, комп’ютерне моделювання для перевірки та аналізу теоретичних положень.

Наукова новизна отриманих результатів полягає в тому, що вперше запропоновано використовувати транзитивний метод балансування для недетермінованих ігор, що дозволяє спростити процес їх розробки.

Практичне значення одержаних результатів полягає в тому, що запропонований підхід дозволяє формалізувати процес розробки масових багатокористувацьких онлайн ігор, що спрощує їх розробку та зменшує її тривалість та вартість.

Особистий внесок здобувача полягає в отриманні основних результатів роботи автором самостійно.

Магістерська робота містить вступ, 5 розділів, висновки, перелік джерел посилань і додатки. Загальний обсяг роботи — 122 сторінок, з яких основний зміст викладено на 88 сторінках друкованого тексту, містить 27 рисунків, 8 таблиць. Перелік джерел посилань містить 17 найменувань.

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ

1.1Аналіз ринку світової ігрової індустрії

Індустрія відеоігор наближається до статусу самої великої медійної також розважальної сфери. У наш час ігрова індустрія випереджає згідно прибуткам кіноіндустрію також наздоганяє спортивний ринок. Фахівці Newzoo, зайнятих консалтингом в області ігрової індустрії, вирахували, то що аж до закінчення 2020 року єдиний її розмір складає $ 159 300 000 000. Це поміщає її на топ-3 основних медіа також розважальних промисловостей [2].

Згідно прогнозами Newzoo, ігрова промисловість стане збільшуватися щороку в ступеня 9,3%. Аж до 2023 роки вона перетне оцінку 200 мільярдів доларів. Важливо, те що дані відомості ніяк не містять прибутку з інтегрованою реклами, які зросли в 59% в період карантину за допомогою COVID-19. Тільки лише в сполучених штатах америки даний ринок інтегрованої реклами зібрав 3 мільярда доларів, згідно з відомостями eMarketer. Крім того фахівці Newzoo ніяк не приймають до уваги ринок торгівлі серед самих користувачів.

Розміри інших масових індустрій медіа також розваг представляються таким чином:

— платне ТБ — $ 226 млрд в 2019 році (не включає стрімінгові служби);

— видавництво — $ 261 млрд в 2017 році. На видавництво книг припадає близько $ 121 млрд;

— фільми — $ 101 млрд в 2019 році;

— музика — $ 62 млрд в 2017 році ($ 30 млрд — продаж записів, $ 6 млрд — музичне видавництво, $ 26 млрд — «живі» концерти).

У світі сьогодні проживає близько 7,8 млрд осіб, майже 4,2 млрд (53,6%) мають доступ в інтернет. З них 2,69 млрд цього року гратимуть в ігри. У Newzoo передбачають, що до 2023 року кількість гравців досягне 3 млрд. Географічно аудиторія розподіляється так:

— 1,447 млрд (54%) — Азіатсько-Тихоокеанський регіон;

— 386 млн (14%) — Європа;

— 377 млн (14%) — Близький Схід і Африка;

— 266 млн (10%) — Латинська Америка;

— 210 млн — Північна Америка [3].

Організатори церемонії нагородження The Game Awards оголосили, що в 2019 році загальна кількість глядачів прямих трансляцій заходу досягла 45,2 млн. У той же час, в 2019 році церемонію вручення «Оскара» подивилися всього 29,6 мільйона телеглядачів.

Є кілька основних чинників, що пояснюють успіх. Один з них пов'язаний зі значною зміною за останні роки бізнес-моделі. Наразі сучасні споживачі почали купувати менше ігор, на відміну від минулих років,але почали проводити в них більше свого вільного часу. Для адаптування під нинішню ситуацію, компанії відійшли від звичної для всіх разової покупки до регулярного потоку доходів. Є кілька поширених можливостей монетизації в грі наприклад: продаж персонажів, зброї, пакетів розширення, ящиків з видобутком і безлічі інших віртуальних предметів.

Інший ключовий момент пов'язаний з розвитком грального обладнання та інтернет-підключень, за рахунок чого ігри стають більш доступними на більшості платформ.

Особливо стрімкий ріст продемонстрував ринок мобільних ігор. Він зробив величезний ривок за останні 10 років. В 2012 році бувши найменшим сегмент ринку, спромігся в 2018 році мати більше половини всіх доходів індустрії — 70 млрд дол. Причому ця частка швидко зростає — ще в минулому році вона дорівнювала 45%. Прогнозується, що в 2020 році сегмент мобільних ігор досягне 77,2 млрд доларів. У сегменті ПК, оціненого в 36,9 млрд доларів, зростання складе всього 4,8%, а в сегменті консольних ігор, оціненого в 45,2 млрд доларів, — 6,8%.

Прибуток від мобільних ігор за 2019 рік

Цифрові ігри, млрд дол.	
Мобільні	59,2
ПК	33
Приставки	8,3
Интерактивне медіа, млрд дол.	
Кіберспорт	0,7
GVC (gaming video content)	3,2
Розширена реальність	4

Рисунок 1.1— Прибуток від комп'ютерних ігор за 2019 р.

1.2 Аналіз ринку комп'ютерної ігрової індустрії в Україні

Український геймдев стрімко розвивається: всього за останні кілька років в сфері розробки ігор з'явилося більше 30-ти нових компаній. Такі студії є в половині регіонів країни. У штаті десятків студій працюють від декількох чоловік до декількох десятків людей. Але є компанії зі штатом 100, 500 і більше 500 чоловік. Велика частина компаній-розробників ігор зосереджені в п'яти великих містах України, де є сильні технічні вузи: Київ, Одеса, Харків, Дніпро, Львів.

Рисунок 1.2 — Геймдев на мапі України

Більшість компаній виробляють від одного до п'яти ігор. Понад 76% українських студій в основному використовують власні інструменти для створення ігор. Необхідно відзначити, що вітчизняні розробники збирають власні кошти для задоволення потреб великих іноземних ігрових студій. Дотримуючись світової тенденції, вітчизняні студії все частіше розробляють ігри для мобільних пристроїв для Android та iOS. Здебільшого в ігрових жанрах вони вважають за краще екшени, головоломки і стратегії.

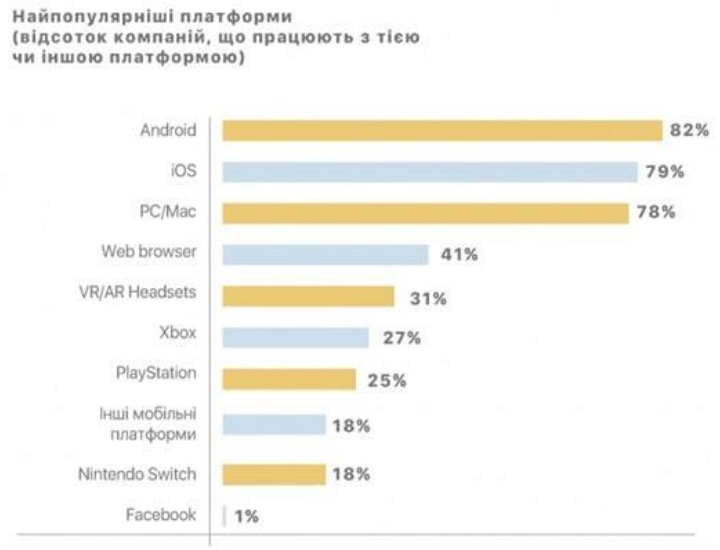


Рисунок 1.3 — Найпопулярніші платформи

Завдяки гарному рівні технічної освіти іноземні компанії можуть відкривати офіси в Україні. Доволі багато програмістів можна знайти тут. Якісна освітня структура дозволяє швидко навчати їх специфіці залузі. Серед художників також багато талановитих представників з хорошою академічною школою.

В цей же час, фахівці коштують нижче ніж в Північній Америці і в кілька разів нижче ніж в Європі, маючи, при цьому, якісні технічні і художні навички. Ці фактори роблять Україну дуже привабливим кандидатом для відкриття офісів. Той рівень якості, що надається українськими розробниками, перевищує рівень якості звичний аутсорсинг у Китаї та в Індії, маючи, водночас, ціни нарівні або навіть нижче. Але при цьому враховуючи бурхливий розвиток індустрії,

швидкість зростання потреби у фахівцях не повільніше зростає і більше за все — на ігрових дизайнерів.

1.3 Аналіз сучасних технологій комп'ютерних ігор

Гра — це вид осмисленої діяльності, основна ціль якого не в результаті, а в процесі. Крім того цей термін використовують для позначення набору елементів або програм призначених для цієї діяльності. Іншими словами це така діяльність, яка має правила і при цьому включає конфлікт. Але питання “що таке гра?” насправді значно складніше.

Складно визначити конкретніше значення цього слова через цілий ряд причин, але із основних факторів заключається в існуючій дуже великій різниці між ігровими жанрами, при чому навіть лише в одному сегменті, що в настільних, що у відеоіграх. Прикладами крайніх випадків, які можуть ускладнювати визначення чіткого значення.

Прикладом головоломок можуть бути кросворд, судоку, кубик Рубіка або головоломки на логіку. Чи можна їх вважати іграми? Кеті Салем і Ерік Зіммерман вважають, що це лише підрозділи ігр. Грег Костікян вважає, що вони не являються іграми, але при цьому можуть бути частиною іншої гри.

Рольові настільні ігри, як наприклад відома Dungeons & Dragons, у яких у самій назві є слово “гра”, але доволі часто їх не навіть вважають за гру, здебільшого по тій причині, що частіше за все, такі ігри не закінчуються конкретним результатом і не завжди там є можливість виграти або програти.

Інтерактивні книги, як наприклад серія ігор Choose-your-own-adventure. Як правило подібні проекти не вважають іграми, адже ви не можете “грати” в книгу, її можна тільки читати. Тим не менш по багатьом визначенням вони підходять під назву ігор. Ще більше путає, що якщо додати до інтерактивної книги карту персонажа з числовими значеннями, та перевірки навичок від цих значень, змінити пару назв і це вже можна назвати грою.

З одного боку більшість історій лінійні, тоді як ігри — динамічні. Але в цей же час велика кількість ігр обов'язково включає в себе якусь історію. Серед

розробників є окремі професійні письменники які займаються над сценаріями сюжета для багатомільйонних проектів відеоігор. Або в цей же час гра сама може породжувати унікальні для окремого гравця історії базованих на їх ігровому досвіді [4].

Деякі ігрові жанри можуть мати в собі одразу декілька ознак інших жанрів. Іноді ці поєднання можуть дати доволі ефективний результат, що може об'єднати в собі переваги декількох жанрів. Одним з таких можна вважати жанр MMORPG.

Масова багатокористувацька рольова онлайн гра (MMORPG — англ Massively Multiplayer Online Role Playing Game) це жанр комп'ютерних ігор, який з недавніх пір набуває все більших обертів у світі, в цьому жанрі рольова гра поєднується з багатокористувацькою мережевою рольовою грою — розрахована на велику кількість користувачів. Велика кількість гравців у віртуальному світі, а також можливість їх активної співпраці один з одним являється основною особливістю гри MMORPG. Соціальна складова MMORPG є однією з основних складових таких ігор, адже різні можливості активної співпраці між гравцями є результатом достатньо щільного спілкування між ними, тому одним з високих вимог до ігор такого жанру є забезпечення можливостями спілкуватися з певним комфортом. Персонаж гравця починає з низького рівня і впродовж гри розвивається, досягаючи максимального рівня.

Одним з шляхів до створення великої онлайн-гри є додаток Telegram який містить безліч платформ і дозволяє листуватися і ділитися мультимедіа в безлічі форматах. Telegram доступний для багатьох операційних систем таких як: Android, iOS, Windows Phone, Windows, macOS і GNU / Linux. Юзери мають можливість ділитись фотографіями, стікерами, голосовими і відео повідомленнями, файлами будь-якого формату, а також використовувати аудіо- і відеодзвінки.

На квітень 2020 року понад 400 млн чоловік активно користуються додатком. У серпні 2017 року кількість користувачів месенджера що денно збільшувалась більш ніж на 600 тисяч. За даними експериментального холдинга Romir за лютий 2018 року, середньостатистичний юзер Telegram витрачає на

нього 10-11 хвилин на день. Максимальна частка користувачів молодь 18-24 років.

Онлайн-гра за допомогою Telegram має можливість бути організована разом з підтримкою так званих ботів — спеціальних програм, які виробляють всілякі функції тим самим спрощують існування юзерів. Прописані для платформи телеграм, вони передбачені для того щоб виконувати найрізноманітніші функцій: з вилучення новинок до відбору даних в тому числі і торгівлі промо-акціями. Основне завдання бота вважається автоматизоване рішення вже після впровадженої йому користувачем вказівки. При цьому, діючи безпосередньо шляхом інтерфейсу телеграм, проект моделює вплив активного користувача, в підсумок чого користуватися подібним ботом стає у багато разів комфортніше і зрозуміліше.

Telegram застосовує один популярний вид ботів, якою від простих користувачів виділяє лише присутність в найменуванні приставки «*bot*». Самі боти розподіляються на кілька типів:

— чат-боти — це найбільш звичайний чат, що наслідує взаємодію на обумовлену користувачем тему;

— боти-інформатори — це самостійний вид ботів, головне завдання яких, - оповіщення користувача про ці або інші факти (анонси, дії, публікації, погода тощо);

— боти-асистенти — це боти, створені всілякими онлайн-сервісами у якості додавання до головної веб-версії;

— ігрові боти — це боти, в яких, можливо грати в різноманітні ігри. В основному це текстові версії ігор [5].

В дійсності, відсутнє точне розділення, внаслідок того, що певні боти включають ряд механік одночасно і також здійснюють дуже різні операції для користувачів. З їх допомогою можна перекладати з різних мов, досліджувати, перевіряти, шукати різні інформацію, грати в ігри також в тому числі і скористатися іншими сервісами і взаємодіяти з речами, які мають доступ до інтернету. Всі без винятків боти в Telegram безкоштовні, проте в 2017 р

адміністрація Telegtam заявив про можливість настройки, а також застосування подібних проектів через оплату. Внаслідок цього боти ставали кишеньковими асистентами, які дозволено застосовувати навіть не припиняючи роботу з месенджером. Вони дають можливість регулювати прості проблеми з підтримкою моментальних установок, при цьому всі без винятку дані програми ніяк не вимагають установки і також ніяк не займають окремого місця всередині пам'яті вашого приладу.

Відштовхуючись від вищесказаного, можна виділити, те що бот, по суті, просто "рупор і важіль", за допомогою якого функціонує одинична програма, прописана в різних мовах програмування в окремому сервері. За Цієї Причини краще було б відзначити, то що бот розпоряджається такою програмою. Зв'язок серед користувачем і ботом полягає в наступному:

User бота надає йому вказівку -> Bot посилає вказівку в ваш сервер -> Програма у вашому сервері обробляє завдання, вироблений ботом -> Сервер надає результат боту -> Bot відображає результат в віконце управління юзера.

Також даний цикл повторюється час від часу, якщо ви натискаєте клавіші і взаємодієте з різними телеграм-ботами.

Ігри в телеграм на відміну від звичайних іграшків для смартфонів не вимагають установки додаткових утиліт в пам'ять пристрою. Вони використовують кеш самого месенджера, завантажуючи дані частинами в міру необхідності. У результатів економиться пам'ять пристрою та забезпечується вища швидкодія.

Поряд із цим, Telegram гра надає можливість приймати в ній участь не виходячи з чату, на відміну від ігор у інших месенджерах, як наприклад в Вайбері, які передбачають не вбудовані утиліти, а банальні посилання на установку їх з маркету. Крім того, згідно з правилами месенджера, розробникам забороняється вбудовувати будь-яку рекламу в своїх розробках, що дозволяє уникнути виникнення під час гри всіляких неприємних рекламних повідомлень. Для отримання прибутку розробляється власна система монетизації.

Вбудовані додатки в месенджера Telegram використовують технологію HTML5 і можуть включати звуки і анімацію. Існуючі програми на HTML5 розробникам буде досить просто інтегрувати в Telegram. Фактично, цей HTML5 код подібний до HTML-сторінок. Додаток може бути не грою в звичному розумінні, а будь-яким інтерактивним сценарієм. Це дає необмежені можливості для розвитку game-платформи в майбутньому. Поточне API вже зараз дозволяє перенести в всім відомі і улюблені 8-ми і 16-ти бітні аркади. Тому можна очікувати їх появи вже протягом найближчих кількох місяців. Також навколо API платформи зараз сформувалося велике співтовариство розробників. Тому поява нових унікальних розробок можна чекати вже в найближчі 2-3 місяці.

На яких пристроях можна спробувати? На смартфоні запуск іграшки здійснюється у власному вбудованому браузері Telegram. Для запуску знадобиться iOS 4 і вище або Android 4.4 і вище, плюс версія месенджер 3.13 і вище. У десктопній версії програми іграшки відкриваються в вашому браузері за замовчуванням.

Плюси:

— можна весело провести час в очікуванні відповіді;

— не потрібно нічого завантажувати;

— вони не займають місця у пам'яті;

— вони мало важать і практично не «поїдають» інтернет.

Мінуси:

— не у всіх є гарна картинка;

— поки що розробники не навчилися реалізовувати нові можливості в повній мірі;

— ігри здебільшого погано реалізовані з точки зору геймдизайну.

2 РОЗРОБКА МАСОВОЇ БАГАТОКОРИСТУВАЦЬКОЇ РОЛЬОВОЇ ОНЛАЙН ГРИ ДЛЯ МЕСЕНДЖЕРА ТЕЛЕГРАМ

2.1 Основні підходи до створення комп'ютерних ігор

Історично, першим методом був так званий метод “водоспаду”: спочатку розробляється вся гра на папері, потім розробляється безпосередньо використовуючи програмування для відеоігор або створення дошки і фішок для нецифрових ігор, потім проводиться тестування гри, щоб переконатися в тому, що зпроектовані правила дійсно працюють, перевіряється на помилки і випускається.

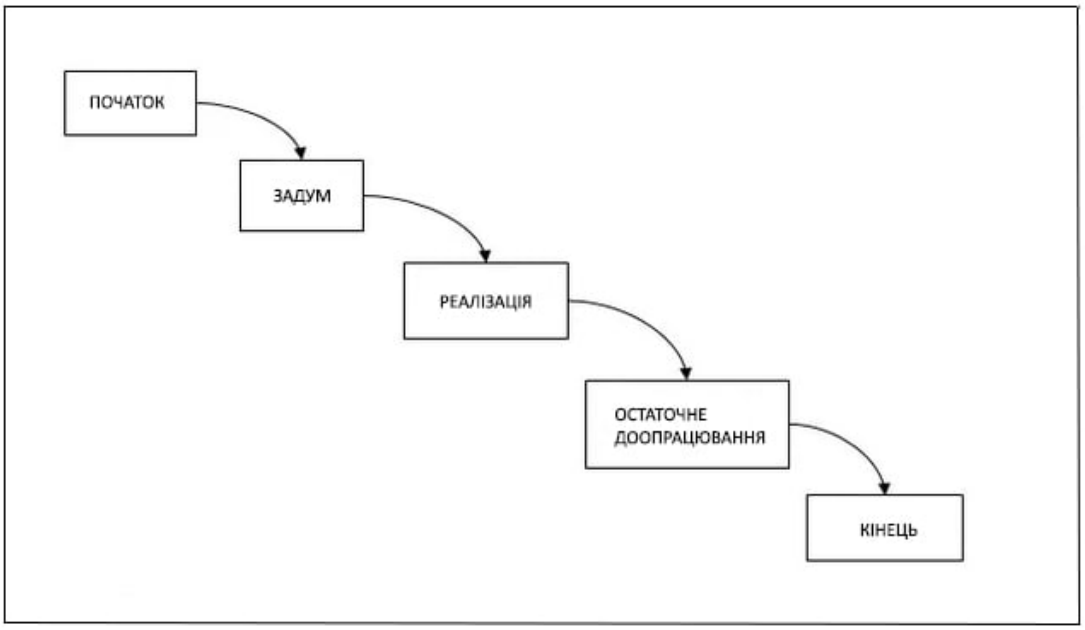


Рисунок 2.1 — Метод “водоспаду”

Але для більш ефективної розробки було б непогано хоча б мати можливість повернутися назад до більш ранніх етапах, тому доволі швидко цей підхід змінився тим, що іноді називають циклічним методом. Як і з “водоспадом”, спочатку проектується концепт гри, потім розробляється, а після цього проводиться перевірка справності. Але потім додається ще один крок для аналізу. Перед наступним кроком проводиться більш детальна оцінка, вирішується що вдалося, а що необхідно допрацювати. А потім, приймається рішення: все готово

чи потрібно повернутися назад до початку і внести певні зміни. Якщо рівень якості гри достатній то розробка далі переходить до випуску. Якщо потрібні зміни, то розробка повертається назад на етап проектування, шукаються шляхи вирішення виявлених проблем, втілюються рішення, і потім знову аналізується результат. Цей цикл потрібно продовжувати до тих пір, поки гра не буде готова.

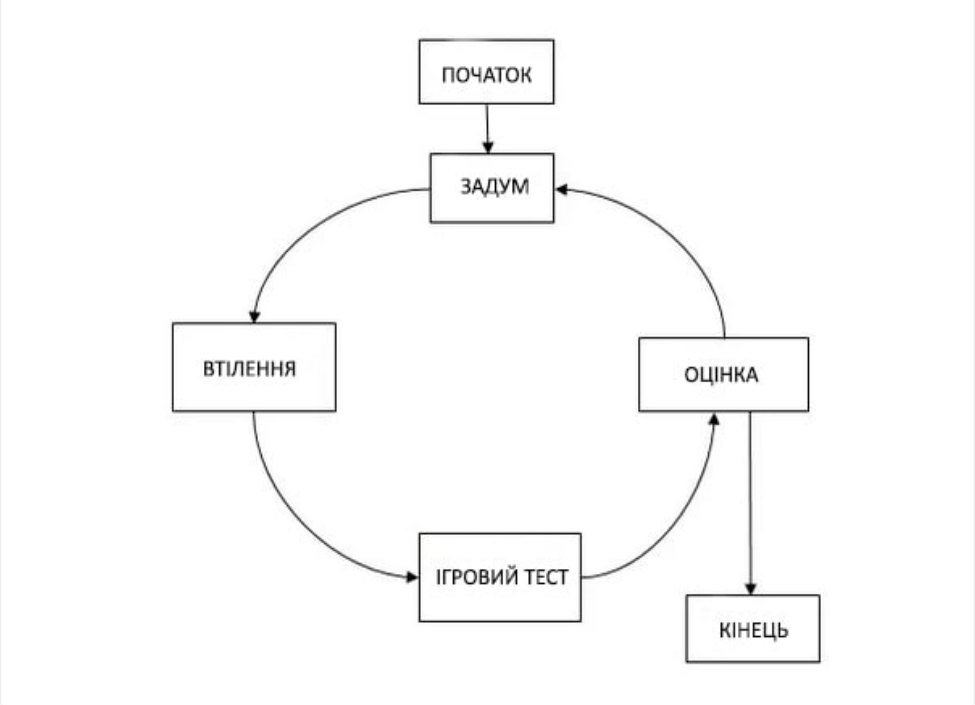


Рисунок 2.2 — Простий ітеративний метод

З нецифровимі іграми, як наприклад картковими і настільними, цей метод працює чудово, так як все це можливо робити достатньо швидко. Але в цей час з відеоіграми складніше. Проблемою є втілення, іншими словами програмування та пошук несправностей, доволі дорого буде обходитись і забирати багато часу. Для того щоб просто створити гру може бути необхідно півтора року, а якщо при цьому на весь проект є всього два роки, то залишається дуже мало часу на тестування і внесення змін.

Тим не менш в цілому, чим більше повторних проходжень буде по цьому колу, тим кращою у результаті вийде гра.

Отже, процес розробки будь-якої гри має включати в собі багаторазовий прогін — повний цикл задумки, проектування, втілення та аналізу — при цьому,

чим більше повторів проходження по цьому колу буде, тим кращою буде якість проекту, і всі дії направленні, щоб проходити цей цикл швидше, скоріше за все буде тільки покращувати кінцевий результат. Саме тому розробники відеоігор частіше всього проектують і моделюють ідеї на папері, а лиш після цього до розробки долучається написання коду. Це називають швидким створенням дослідних зразків.

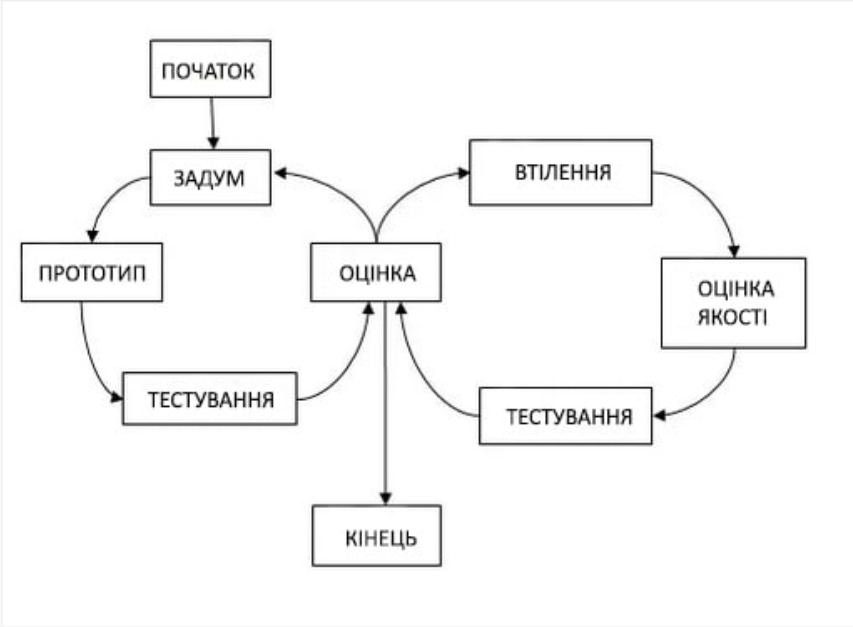


Рисунок 2.3 — Ітеративний метод

2.2 Аналіз основних принципів ігрового дизайну

Розробити просту гру, маючи знання програмування — нескладно. Нескладно зробити її естетично гарною, маючи художні навички і навички моделювання (в тому числі і в 3D). Але це не означає, що така гра буде цікавою і що вона взагалі може когось зацікавити більше ніж на п’ять хвилин, не говорячи вже про те, щоб її хтось купував. Як зробити так щоб в гру було цікаво грати? Це одне з питань які вирішує ігровий дизайн.

Ігровий дизайн (частіше геймдизайн, англ. Game design) — це процес проектування змісту, форми, а також правил ігрового процесу (зазвичай

називаємий геймплеєм, англ Gameplay) гри, що розробляється. Роль ігрового дизайну при створенні гри можливо можна порівнювати в програмуванні з роллю постановлення проектних завдань і в кінематографі з режисерською працею [6].

Ігровий дизайн — це процес розробки ігрового контенту та правил. Але хороший геймдизайн, в цей час, це процес насамперед створення цілей, що гравець, в процесі гри, буде мати бажання досягти, і, крім цього, правил, яким також гравець буде так само слідувати приймаючи значущі рішення у процесі досягнення цих цілей.

Хороший геймдизайн має бути орієнтованим і акцентованим виключно на гравця. Мається на увазі, що під час проектування та розробки проекту більш за все необхідно брати в розрахунок саме гравеця і його бажання. Хороший ігровий дизайн, замість направлення дій гравця за допомогою жорстких правил, буде мотивувати гравця щоб той рухався в певному напрямку, задуманим дизайнером. Недостатньо ставити завдання перед гравцями діставатися через ігрове поле до протилежного його краю або підвищити свій рівень. Якщо при цьому у гравця не буде зрозумілих йому причин або бажання виконати це завдання, то гра для нього стане тортурами. При розробці гри дизайнери по цим причинам намагаються дивитись на гру очима гравця.

Заглиблюючись більше до суті геймдизайну прийдемо до висновку, що це розробка можливостей для гравця в процесі гри робити істотні вибори, які безпосередньо будуть впливати на результат цієї гри.

Ігри по своєму схожі, якщо розглядати їх з боку мислення. Такі ігри як наприклад тетрис або шахи безперечно займають розум гравця, змушуючи його прораховувати наперед наступні можливі кроки. Йому відомо, що здійснюючи їх він може як і продовжити гру, так і водночас програти з розгромом. Такі ігри як Sims або серії Sid Meyer's Civilization вже змушують гравця іноді приймати і десятки рішень на хвилину. При цьому лише зовсім мала частина від цих прийнятих рішень банальна і прямолінійна такі як «Піти на схід або на захід?». В цей час будь яке навіть незначне рішення обов'язково буде впливати на ігровий процес.

Якщо розглядати ігри зі сторони ігрового дизайну, то вони просто набір значущих рішень. Кожен з перелічених пунктів — приклад значимого ігрового рішення:

— розташування військ в покрокових (TBS) стратегіях або стратегіях реального часу (RTS);

— розподіл балів при підвищенні рівня в рольових іграх (RPG);

— вибір фігури для ходу в шахах;

— прицілювання і стрільба в шутерах (FPS);

— натискання потрібної кнопки в потрібний час в Guitar Hero [7].

Дизайнер створює значимість рішення там, де гравець може відчути здатність зробити свій вибір, а вибір, в свою чергу, може впливати безпосередньо на результат гри. Щоб створити вибір необхідно дати гравцю також альтернативний варіант, що також може мати значення.

Тим не менш є випадки коли вибору у гравця немає. Наприклад гра «Монополія» є прикладом подібного випадку. У гравців немає іншого вибору окрім як просто кидати кубики і ходити, коли наступить та частина гри коли вся власність уже куплена. Це рішення не може вважатись вибором гравця, через те що у цієї дії немає альтернативи. Адже коли настає цей етап то в грі більше немає значущих рішень. Результат кожної окремої гри, в таких ситуаціях, стає повністю передбачуваним. Саме тому «Монополія» вже через кілька ігор може в гравця з легкістю викликати нудьгу. Тим не менш, зазвичай, такі проблеми не з'являються при створенні відеоігор.

Одним з головних принципів ігрового дизайну є модель МДЕ (Механіка-Динаміка-Естетика). Механіки породжують динаміку (комплекс механік), а динаміка породжує естетику (відчуття гравця від гри). Ця модель ігрової розробки використовується для створення дизайнерами естетичних моделей у різних видах ігрового процесу. Під естетичністю в цьому випадку мається на увазі не зовнішність гри, а здебільше саме емоційний резонанс. Саме у відтворенні цього через ігрові динаміки і полягає ціль розробника.

Якщо механіка — це правила, а динаміка — це безпосередньо гра, то

естетика — задоволення (або недолік такого) в процесі гри, що випробовується граючим. Дизайнери задаються питанням, який естетики вони хотіли б досягти, визначають динаміку, яка повинна привести до цього відчуття, і потім створюють механіку, яка б справила потрібну динаміку.

Співтовариші Лебланка окреслюють поняття гри в рамках її механіки, динаміки і естетики.

Механіка є синонімом «правил» гри. Це обмеження, в рамках яких функціонує гра. Як готується до гри? Які дії можуть здійснювати гравці? Як ці дії відображаються на стані гри? Коли закінчується гра, як визначається її результат? Все це визначається механікою.

Динаміка описує хід гри, коли правила почали діяти. Які стратегії породжуються правилами? Як взаємодіють один з одним гравці?

Естетика (в контексті цієї системи) не відноситься до візуальних елементів гри, а скоріше, до враженню від гри, ефекту, який гра надає на гравців. Цікава вона? Гнітюча, нудна або захоплююча? Захоплює гра емоційно чи інтелектуально [8].

До того, як була написана МДЕ-структура, терміни «механіка» і «динаміка» вже мали ходіння серед дизайнерів. Термін «естетика» в такому сенсі не використовувався, але в останні роки вживається все частіше.

2.3 Види геймдизайну

Класифікація видів геймдизайну не є стандартизованою і тому, частіше за все, різниться від компанії до компанії, проте, є цілий ряд ключових кваліфікацій, що, в більшості своїй, поширені в сьогоднішньому ігровому дизайні:

— системний дизайн — створення систем, правил і супутніх розрахунків для гри;

— дизайн рівнів — розробка ігрової карти і елементів, з яких буде складатися кожен ігровий рівень;

— проектування ігрової механіки — розробка законів, за якими влаштований ігровий світ, характеристик його об'єктів, формул руху;

— настройка балансу — конфігурація складності гри, щоб гравцю не було занадто важко або легко, рівні були не надто короткі або довгі;

— наративний дизайн — написання історії та сюжета в процесі розробки гри [9].

Крім цього в завдання цього дизайну відносяться і відображення історії в ігрових механіках, а також ще одну класифікацію — залежно від акценту на технічну частину або творчу.

Art гейм-дизайнер може відповідати за графічний і звуковий дизайн, написання текстів і сценаріїв, створення рівнів і персонажів, 3D-моделювання та анімацію. Як правило, при розвитку кар'єри такий фахівець поглиблюється в якийсь один аспект своєї роботи, але все одно важливою особливістю професії залишається універсальність і широкий спектр навичок.

Технічний гейм-дизайнер працює з математикою, ігровою статистикою і алгоритмами. Цей фахівець налаштовує всі числа, які можна побачити в іграх: зароблені очки, рівень персонажа, ймовірність критичного урону. Під налаштуванням мається на увазі не тільки суха математика, а й створення ігрового досвіду. Наприклад, тривалість сесії в шутері може бути налаштована так, щоб вирішальні події відбувалися саме в останні секунди матчу [10].

2.4 Розробка ігрового балансу

Ігровий баланс — це співвідношення серед ігрових предметів також концепцій, такими як герої, команди, стратегія гри і т.д. Ігровий баланс — одна з умов до правил "чесності". Рівновага це основа для багатокористувацьких ігор.

В комп'ютерний іграх, це рівновага серед чисел, що описує різноманітні властивості в грі — подібні на порушення сили, швидкість бігу, темп вилучення ресурсів і багато іншого. Ця рівновага в значній мірі встановлює складність, зацікавленість також м'якість геймплея [11].

Ігровий баланс — одна з найскладніших сторін при розробці гри. Розробник призводить гру до стану балансу, змінюючи ігрові характеристики в ході бета-тестування. Але остаточно баланс мережевої (онлайн) гри відточується протягом

деякого часу після закінчення розробки і виходу самої гри в відкритий доступ, так як це складний процес підгонки самих незначних відхилень від збалансованого значення. Після випуску нової версії балансу розробник пропонує встановити патч з оновленнями.

При аналізі балансу ігри можна поділити на детерміновані та недетерміновані.

Детермінована гра, або гра з повною (абсолютною) інформацією, — теоретико-ігровий термін, що позначає гру, в якій гравцям відомі функція корисності, правила гри, а також ходи інших гравців. Приклади ігор с повною інформацією - шахи і нарди.

Якщо простіше то детермінованою грою називається така гра, де при певному її стані одна і та ж дію буде завжди приводити до одного і того ж нового стану гри. Ігровий дизайн таких ігор позбавлений випадковостей, а всі дії - можливо передбачити. Гравець, в такому випадку, має в будь який момент повну інформацію про стан гри.

Відповідно недетермінована гра — гра з неповною інформацією. У кожній з них є механізм випадковості. Наприклад покер — недетермінірована гра. Гравець може зіграти поспіль кілька роздач, де стан гри буде здаватися однаковим (відкриті карти на столі і карти на руках у гравця ті ж), але результати роздачі, скоріше за все, будуть іншими, тому що він ніколи не може точно знати, які карти на руках у ваших суперників [13].

Згідно цього існують такі основні методи балансування ігрових механік в цілому, а саме транзитивний та ін транзитивний, а також не компаративний.

Транзитивний спосіб вважається способом безпосереднього зіставлення даних. Транзитний спосіб застосовується у тих випадках, якщо порівнюючі предмети мають ідентичний або схожий згідно змістом комплект даних. Для Того Щоб врівноважити даний спосіб, створена табличка, в яку вступили відомості про предмети.

Інтранзитивний спосіб. Урівноваження зсередини позитиву містить в собі введення даних про предметах, взаємодіючих всередині матриці, а також

результати їхньої взаємодії. З метою цього формується метод, що зобов'язаний симолювати вплив гравців.

Компаративний спосіб вважається способом балансу, у якому властивості ніяк не мають всі шанси бути співставлені. У такому випадку замість балансування так як такого, є зміна даних, які вимагають балансування, ті, які ніяк не зможуть бути співставлені, також, відповідно до цього, збалансовані. Даний спосіб досить неякісний також його застосування допустимо тільки лише в екстремальних моментах [14].

Тільки транзитивний спосіб годиться з метою балансування геймплея в розробленій грі бота, таким чином просто нереально стимулювати вплив гравців у багатокористувацьких онлайн іграх, з величезною мінливістю ймовірних подій.

Ігрові взаємодії — це комплект характеристик, які змінюють власні значення під впливом різних явищ. Коли у одного предмета є функція Дефекти, а у іншого предмета є функція «Здоров'я», можливо сформувати геймплей «Об'єкт атакує інший об'єкт», також в слідстві «Здоров'я» змінює власне значення в значення параметра«Шкода» . Подібним способом, як виявилося, те що вплив (в цьому зразку впливу на стан здоров'я шкоди) вважається фактором зміни характеристик предмета.

Отже, проаналізувавши способи балансування даних також ознайомившись з методологією транзитного балансування, було встановлено рішення створити формулу середньої шкоди для персонажа, з метою комфортного також результативного балансу шляхом зіставлення підсумків.

Основні цілі балансування — гра повинна відчуватися гравцем чесної і цікавою. У мультіплеєрних іграх баланс також важливий для врівноваження умов для гравців, а також для рівних шансів кожного окремого гравця на перемогу.

Максимально якісно розроблений баланс для жанру ММОРПГ, можливо, навіть більш важливий ніж інших жанрів відеоігор. Пов'язано це з тим, що, крім того, що ігри цього жанру найчастіше мають в собі самі різні активності, як PvE так і PvP, але також ці ігри розраховані на тривалий час життя. Наприклад найвідоміша і популярна гра в цьому жанрі World of Warcraft була випущена в

2004 році, але до цих пір підтримується, активно оновлюється і все ще багатьма вважається кращою в своєму жанрі. Чим довше існує гра тим більше буде помічено гравцями дрібні "шорсткості" в балансі. Це буде ставати проблемою якщо гравці будуть зловживати недоліками в балансі щоб, наприклад, стати сильніше інших гравців. Очевидно що це не буде позитивно складатися на успішності проекту. Безсумнівно для розробника важливо своєчасно виявляти недоліки в балансі і оперативно їх виправляти за допомогою нових патчів. Але щоб мінімізувати кількість помилок, а майбутні виправлення не породжували собою нові вкрай важливо створити хороший баланс ще на етапі проектування гри.

Навіть всередині свого жанру гри можуть помітно відрізнятися від інших представників і не мати певних механік які прийнято було використовувати в інших проектах. Проте певні аспекти найчастіше залишаються однаковими у всіх іграх жанру ММОРПГ і, як правило, відрізняють їх від ігор інших жанрів.

Просування гравця по грі завжди пов'язане з розвитком його персонажа, а воно, у свою чергу, так чи інакше, прямо буде пов'язано з битвами персонажа з ворогами (будь то дикі тварини, монстри або персонажі інших гравців). Розробка бойової системи — одне з найважливіших завдань в області системного геймдизайну. Бойова система повинна бути досить цікавою і багатогранною щоб вона не набридла гравцеві протягом тривалого періоду, а також вона повинна підштовхувати гравців до взаємодії між собою (Role-play складова).

Основними параметрами персонажа є: Здоров'я, Базова шкода, Опір, Захист, Критична шкода, Проникнення

Здоров'я (HP англ Hit Point) — найголовніший числовий показник персонажа, якщо опускається до нуля персонаж вважається переможеним. Має свій максимум для кожного персонажа в залежності від рівня персонажа і одягненою екіпіровки.

Базова шкода (DMG англ Damage) — числовий показник персонажа, який характеризує силу його звичайної атаки, може бути двох типів: фізичного

пошкодження і магічного пошкодження. Персонаж може одночасно наносити тільки один тип шкоди.

Захист (Eva — англ Evade) — процентний показник персонажа, характеризує шанс персонажа на те що він абсолютно не отримає пошкодження. Перевірка на уворот проводиться найпершою, при успіху хід закінчено, при невдачі, розрахунок проходить далі. Захист є двох типів: фізичний захист і магічний захист. Шанс Захисту певного типу впливає тільки на відповідний тип шкоди.

Опір (Resistance) — числовий показник персонажа скільки з одержуваного пошкодження по персонажу буде проігноровано. Решта пошкоджень віднімається від Здоров'я. Опір може бути двох типів: фізичний опір і магічний опір. Опір певного типу ігнорує шкоди тільки відповідного типу. Наприклад Фізичний Опір ігноруватиме частину фізичного пошкодження, але абсолютно не ігноруватиме Магічні Втрати.

Критична шкода (Critical Damage) — процентний показник персонажа, вказує шанс того, що атака завдасть в два рази більше шкоди. Критична шкода є двох типів: фізично критичне пошкодження і магічно критичне пошкодження, відповідно для кожного типу шкоди. Критична шкода розраховується перед Опором і Проникненням.

Проникнення (Penetration) — це параметр який вказує скільки з шкоди Гравця проігнорує ворожий Опір. Проникнення є двох типів: Фізичне проникнення і Магічне Проникнення — відповідно для кожного типу Опору.

Відносна ефективність персонажа = Втрати який він може нанести * Втрати який він здатний витримати.

Якщо описати простіше:

Ефективність = Атака * Виживання

Оскільки в грі більше параметрів потрібно також врахувати і їх. Розділимо всі параметри на ті які відносяться до Атаки (Базова шкода, Критична шкода, Проникнення) і ті які відносяться до Виживання (Здоров'я, Захист, Опір)
Для початку розрахуємо Атаку для Базовою Атаки і Шанс нанести критичну (подвійну) шкоду:

Атака = Базова шкода * (1 + Шанс критичної шкоди)

Проникнення можна вважати звичайною надбавкою до втрат, але він не завжди буває максимальним. Наприклад якщо у противника опір менше ніж Проникнення у персонажа гравця. Проникнення не може давати більше шкоди ніж опір ворога. А якщо опора немає зовсім, то відповідно і Проникнення не дасть ніякої надбавки до втрат. Але ми будемо розраховувати максимально можливу Ефективність персонажа, тому і Проникнення у формулі Атаки буде давати стабільне і максимальне посилення.

Атака = Базова шкода * (1 + Шанс критичної шкоди) + проникнення

Далі розрахуємо Виживання для Здоров'я і Шансу Ухилення:

Виживання = Здоров'я / (1 Шанс Ухилення)

Тепер потрібно врахувати ще параметр Опір, його можна розраховувати як звичайну надбавку до Здоров'я. Але, також як і Проникнення, воно не завжди може бути максимальним. Наприклад в ситуаціях коли шкода супротивника нижче ніж параметр Опору. Але так як ми будемо розраховувати максимально можливу Ефективність для персонажа, то і Опір порахуємо максимальною надбавкою.

Виживання = (Здоров'я / (1 Шанс Ухилення)) + Опір

У підсумку отримуємо Ефективність для персонажа (без урахування фізичного і магічного пошкодження)

Ефективність = (Базова шкода * (1 + Шанс критичної шкоди) + Проникнення) * ((здоров'я / (1 Шанс Ухилення)) + Опір)

Для супротивники не-гравців необхідно враховувати іншу формулу, так як вони мають шанс нанести кожен хід або фізичну або магічну атаку.

Атака фізична = ((Базова фізична шкода * (1 + шанс фізично критичної шкоди)) + фізичне проникнення) * шанс фізичної атаки

Атака магічна = ((Базова магічна шкода * (1 + шанс магічно критичної шкоди)) + магічне проникнення) * шанс магічної атаки

Атака = ((((Базова фізична шкода * (1 + шанс фізично критичної шкоди)) + фізичне проникнення)) * шанс фізичної атаки) + ((((Базова магічна шкода * (1 + шанс магічно критичної шкоди)) + магічне проникнення)) * шанс магічної атаки)

Виживання фізичне = ((Здоров'я / (1 - шанс Ухилення фізичний) + опір фізичний) * 0,5

Виживання магічне = ((Здоров'я / (1 - шанс Ухилення магічний) + опір магічний) * 0,5

Виживання = ((Здоров'я / (1 - шанс Ухилення магічний) + опір магічний) * 0,5 + ((Здоров'я / (1 - шанс Ухилення фізичний) + опір фізичний) * 0,5

У підсумку отримуємо остаточну формулу Ефективності з урахуванням шансу нанесення певного типу шкоди:

Ефективність = ((((Базова фізична шкода * (1 + шанс фізично критичної шкоди)) + фізичне проникнення)) * шанс фізичної атаки) + ((((Базова магічна шкода * (1 + шанс магічно критичної шкоди)) + магічне проникнення)) * шанс магічної атаки) * ((Здоров'я / (1 - шанс Ухилення магічний) + опір магічний) * 0,5 + ((Здоров'я / (1 - шанс Ухилення фізичний) + опір фізичний) * 0,5

3 РОЗРОБКА ПРОГРАМНИХ ЕЛЕМЕНТІВ

3.1 Розробка алгоритмів роботи програми

Одна з ключових задач розробки ігрового бота — потрібно створити алгоритм генерації оригінальних артефактів, переміщення героїв між локаціями, а також такі алгоритми як PVP і PVE і крім цього, вивести формулу з метою балансування характеристик цих персонажів.

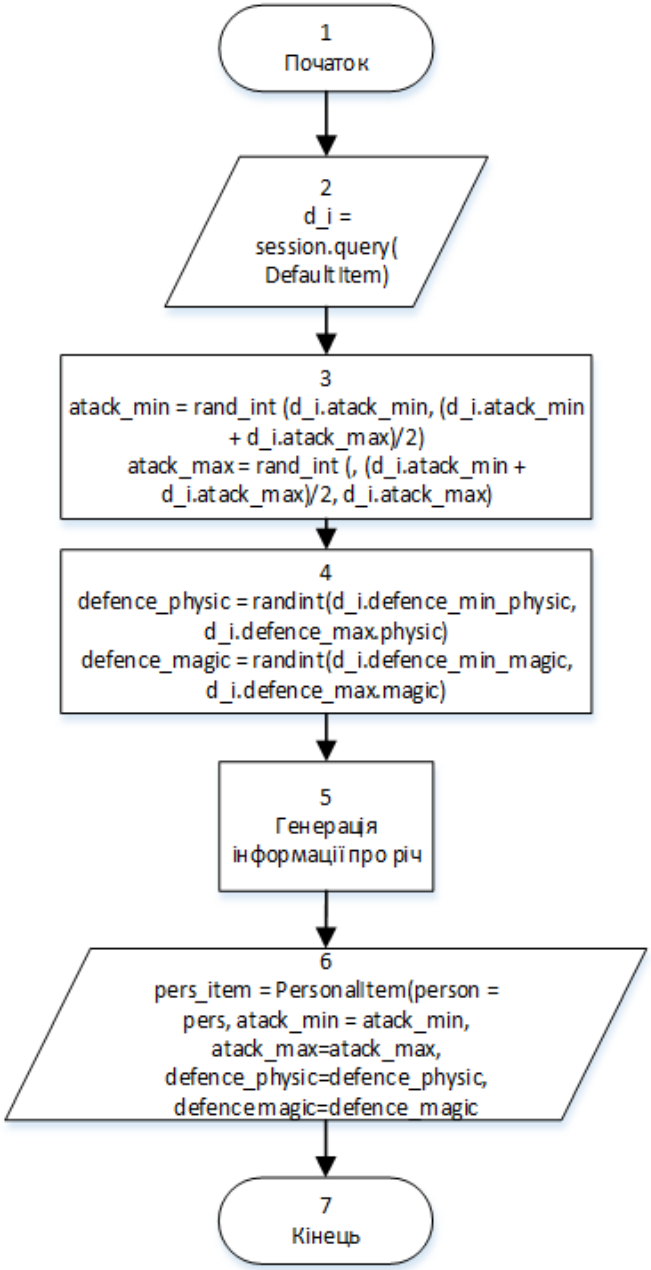


Рисунок 3.1 — Схема алгоритму генерації унікальних артефактів

Алгоритм генерації унікальних речей складається з 7 пунктів:

— початок;

— отримаємо з БД шаблон знайденої речі;

— вираховуємо мінімальне та максимальне пошкодження в залежності від типу пошкодження шляхом генерації випадкового числа в межах мінімального і середнього, а також середнього і максимального пошкодження, зазначеного в шаблоні, відповідно;

— вираховуємо параметри фізичного та магічного захисту шляхом генерації випадкового числа в межах вказаних мінімальних і максимальних значень в шаблоні для кожного з двох типів;

— генеруємо опис речі в залежності від зенерованих параметрів у пунктах 3 та 4;

— створюємо об’єкт персональної речі, записуючі знайдені значення, та присвоюючи унікальний ідентифікатор персонажа;

— кінець.

Алгоритм пересування між локаціями складається з 10 пунктів:

— початок;

— отримуємо поточну локацію персонажа, або в разі відсутності задаємо початкову;

— витягуємо з БД інформацію про час переміщення між локаціями;

— перевіряємо наявність потрібної інформації;

— якщо інформації немає, виводимо повідомлення про те, що такої локації не існує, або потрапити в неї на даний час не можливо;

— інакше присвоюємо персонажу стан виконання певної дії;

— запускаємо таймер на час переміщення персонажу;

— після завершення таймеру запускається функція, що знімає з персонажу стан виконання дії, присвоює йому локацію, та визначає тип потрібної клавіатури;

— відправляємо сповіщення про прибуття до локації з прикріпленною клавіатурою кнопок на цій локації;

— кінець.

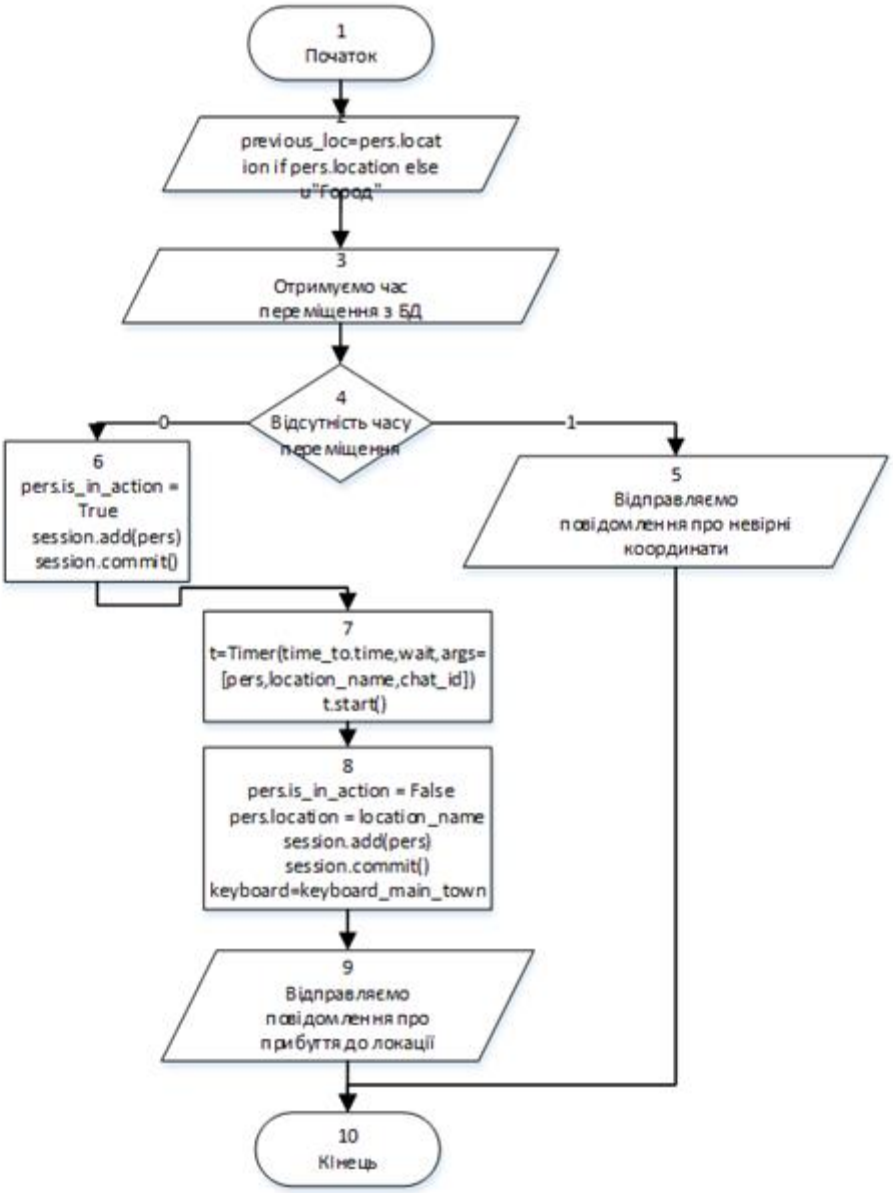


Рисунок 3.2 — Схема алгоритму пересування між локаціями

Алгоритми PVP та PVE складаються з 19 блоків, блоки 6, 9, 12 та 15 відповідають розробленому алгоритму функції нападу.

— початок;

— отримуємо об’єкти супротивників;

— обнуляємо лічильники ланцюгів успішних влучень, створюємо змінні для визначення смерті супротивників та визначаємо першочерговість ходу;

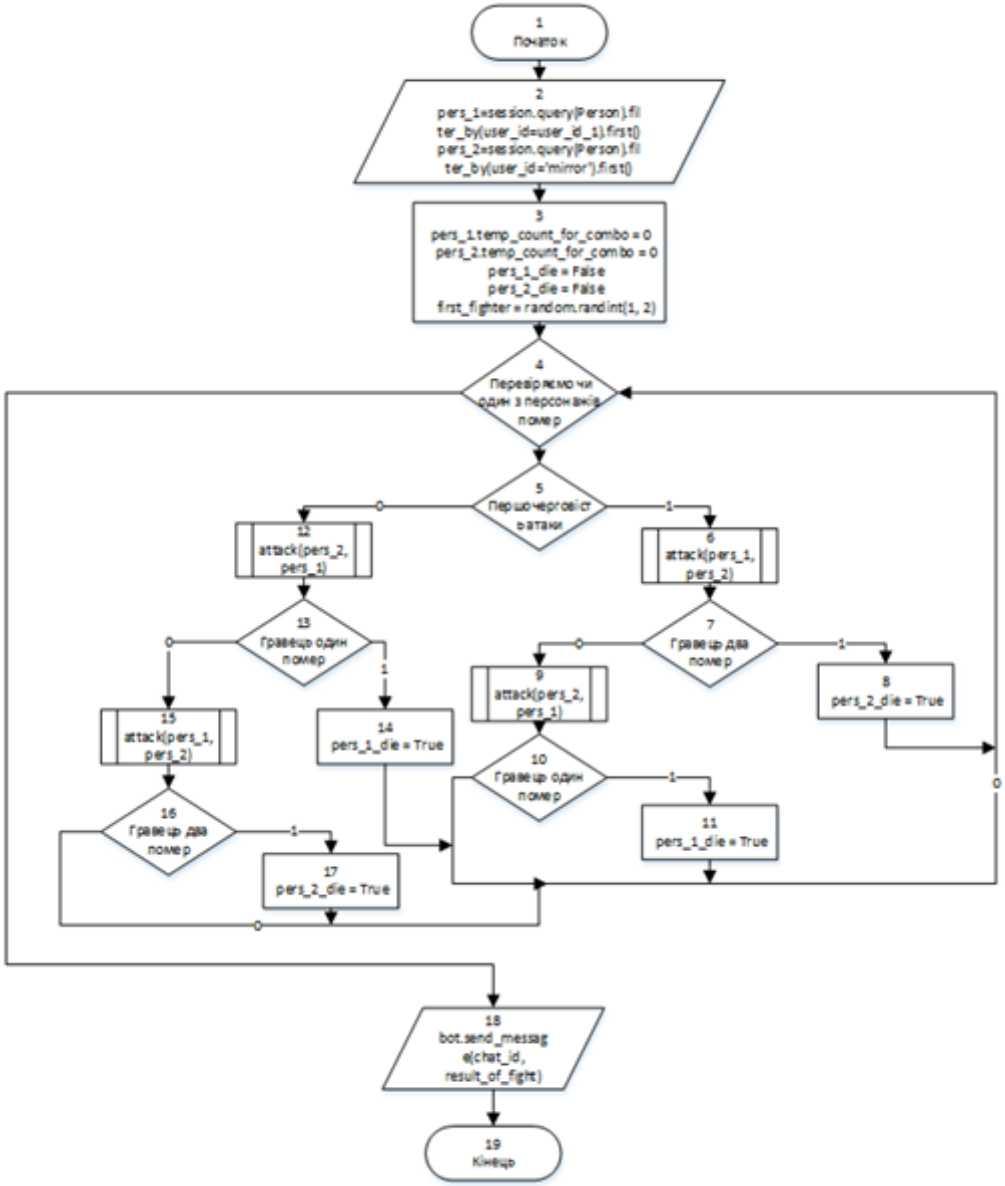


Рисунок 3.3 — Алгоритм PVP та PVE ч.1.

— запускаємо цикл, умовою виходу з якого є смерть одного із супротивників;

— перевіряємо значення змінної, що відповідає за першочерговість ходу;

— якщо черга першого супротивника, виконуємо функцію нападу, передаючи як відповідні аргументи нападаючого та захисника;

— перевіряємо чи було вбито захисника;

— якщо було вбито захисника, то встановлюємо значення змінної, що відповідає за його смерть, як істинне;

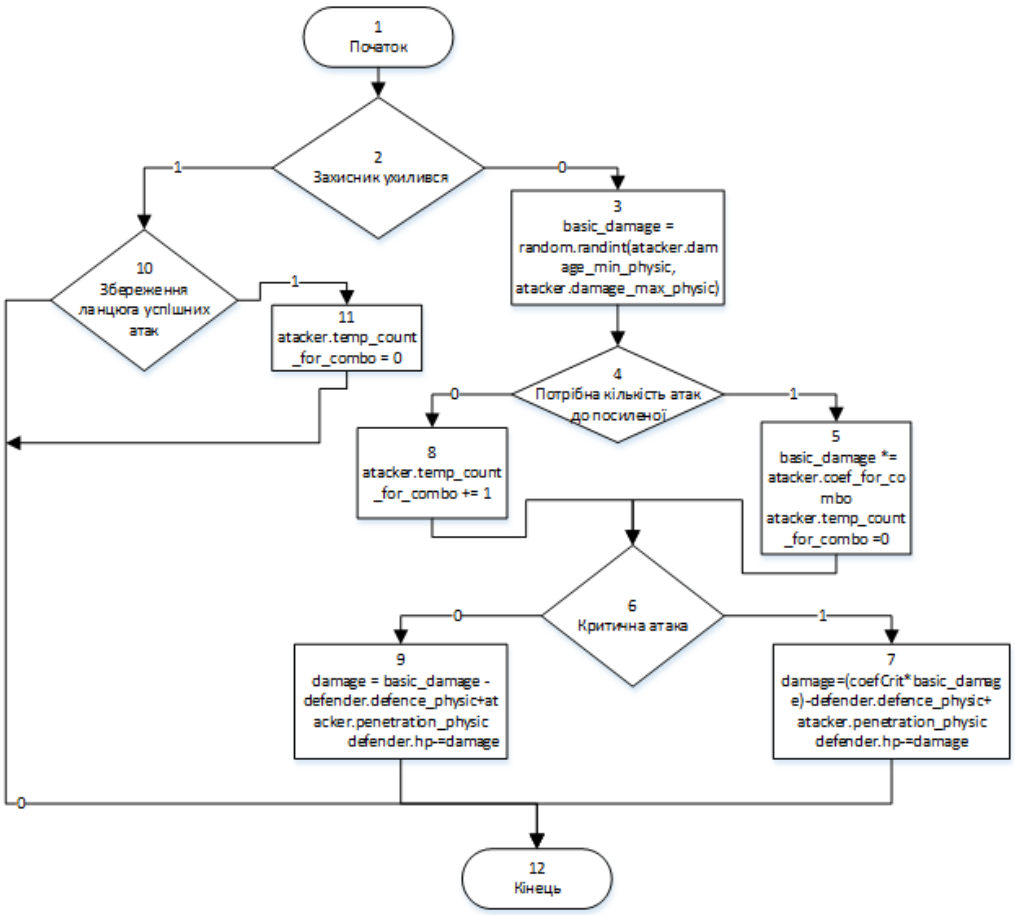


Рисунок 3.4 Алгоритм PVP та PVE ч.2.

— якщо захисник залишився живий, виконуємо функцію нападу, змінивши місцями супротивників;

— перевіряємо чи було вбито першого супротивника;

— якщо перший супротивник загинув, встановлюємо значення змінної, що відповідає за його смерть, як істинне;

— аналогічні дії з 7 пункта до 11 зі змінною черги нападу;

— відправляємо повідомлення про результати бою персонажам;

— кінець.

Алгоритм функції нападу складається з 12 пунктів.

— початок;

— перевіряємо чи захисник не ухилився;

— якщо захисник не ухилився, вираховуємо базове пошкодження шляхом отримання випадкового значення між можливим мінімальним та максимальним;

— перевіряємо чи набралася потрібна кількість влучень по супротивнику;

— якщо потрібна кількість влучень набралася, то множимо базове пошкодження на коефіцієнт посиленого пошкодження та обнуляємо лічильник;

— перевіряємо чи відбулась критичне влучення;

— якщо критичне влучення відбулось, то множимо результуюче пошкодження на коефіцієнт критичного пошкодження;

— якщо потрібна кількість влучень по супротивнику не набралась, то збільшуємо лічильник успішних влучень на 1 та виконуємо пункті 6 та 7;

— від результуючого пошкодження віднімаємо значення захисту захисника та додаємо значення проникнення нападаючого;

— якщо захисник ухилився, перевіряємо чи зберігся ланцюг успішних атак;

— якщо не зберігся, то обнуляємо лічильник;

— кінець.

Враховуючи розроблену бойову систему, формула для балансування характеристик отримала такий вигляд.

$$DamM = DamA - AD + CritA - EvaD + \frac{ComboDamA}{n}; \quad (3.1)$$

де DamA — середнє фізичне або магічне базове пошкодження персонажа в залежності від типу використаної зброї.

DamA розраховується за формулою.

$$DamA = \frac{DamAMin + DamAMax}{2}; \quad (3.2)$$

AD — різниця між захистом та проникненням захисника та атакуючого.

$$AD = DefD - PenA; \tag{3.3}$$

CritA — середня надбавка від критичного пошкодження, що розподілена на всю довжину бою.

$$CritA = \big((DamA * cofC) - DamA\big) * chanceCritA; \tag{3.4}$$

де cofC — в скільки разів критична атака більша за звичайну;

chanceCritA — шанс, з яким персонаж може нанести критичну атаку;

EvaD — кількість пошкоднення, від якого може ухилитись захисник.

$$EvaD = (DamA - AD + CritA) * (chanceEvaD - AccuracyA); \tag{3.5}$$

де AccuracyA — точність персонажа, яка залежить від навика володіння зброєю;

chanceEvaD — шанс захисника ухилитись;

ComboDamA — середня надбавка від посиленого пошкодження, яка збільшує базову атаку та може відбутись, якщо атакуючий персонаж проведе успішно *n* атак підряд.

$$ComboDamA = (DamA8ComboCoef - DamA) * \Big((100\% - chanceEvaD + AccuracyA) + \big((chanceEvaD - AccruracyA) * MempAttackA\big)\Big) * (1 + cofC * chanceCrirA); \tag{3.6}$$

де ComboCoef — коефіцієнт збільшення базової атаки,

TempAttackA — шанс зберегти ланцюжок успішних атак при неуспішно проведеній.

За допомогою цих формул можливо з легкістю контролювати баланс характеристик персонажів на різних рівнях, та надати гравцям можливість з доволі непоганим шансом перемогти когось, хто на 1-4 рівня більший за твій .

3.2 Розробка діаграм

На рисунку 3.5 представлено діаграму класа «Персонаж», який описує об’єкти персонажів за яких грають гравці. Об’єкт класа «Персонаж» має наступні поля:

— user_id — унікальний ідентифікатор гравця;

— chat_id — унікальний ідентивікатор аккаунту гравця;

— name — ім’я персонажа;

— lvl — рівень персонажа;

— current_energy — поточна кількість енергії;

— max_energy — максимально можлива кількість енергії;

— experience — поточна кількість очків досвіду;

— experience_to_next_lvl — кількість очків досвіду які потрібно набрати для досягнення наступного рівня;

— hp — поточна кількість здоров’я;

— max_hp — максимально можлива кількість здоров’я;

— damage_min_physic — мінімально можливе фізичне пошкодження;

— damage_max_physic — максимально можливе фізичне пошкодження;

— damage_min_magic — мінімально можливе магічне пошкодження;

Рисунок 3.5 — UML діаграма класу «Персонаж»

— damage_max_magic – максимально можливе магічне пошкодження;

— defence_physic – опір фізичному пошкодження;

— defence_magic – опір магічному пошкодженню;

— penetration_physic – фізичне проникнення;

— penetration_magic – магічне проникнення;

— accuracy_physic – точність нанесення фізичного пошкодження;

— accuracy_magic – точність нанесення магічного пошкодження;

— dodge_magic – шанс захиститися від магічного пошкодження;

— dodge_physic – шанс захиститися від фізичного пошкодження;

— crit_magic – шанс нанести критичне магічне пошкодження;

— crit_physic – шанс нанести критичне фізичне пошкодження;

— temp_attack – шанс зберегти ланцюг успішних попадань;

— count_for_combo – потрібна кількість успішних попадань для нанесення посиленої атаки;

— coef_for_combo – коефіцієнт визначаючий в скільки раз посилена атака більша за звичайну;

— temp_count_for_combo – поточна кількість успішних попадань;

— gold – кількість золотих монет у персонажа;

— donat_value – кількість платної валюти;

— weapon – зброя;

— head – броня на голову;

— body – броня на торс;

— legs – броня на ноги;

— finger_1 – кільце на одному пальці;

— finger_2 – кільце на іншому пальці;

— neck – прикраса на шию;

— hands – броня на руки;

— feet – броня на ступні;

— backpack_amount – поточна загруженість ранця;

— backpack_amount_max – вмістимість ранця;

— group – ідентифікатор групи;

— location – поточна локація в якій знаходиться персонаж;

— monster_for_fight – поточний супротивник персонажа(з не контролюємих персонажів) ;

— is_in_action – булева зміна яка визначає чи виконує персонаж якусь дію в даний момент часу;

— is_die – булева зміна яка визначає чи живий персонаж.

Окрім визначених полів, клас має також метод «__repr__(self)» який приймає об’єкт даного класу та виводить інформацію про нього.

На рисунку 3.6 представлено діаграму класа «НП», який описує об’єкти неконтролюємих гравцем персонажів.

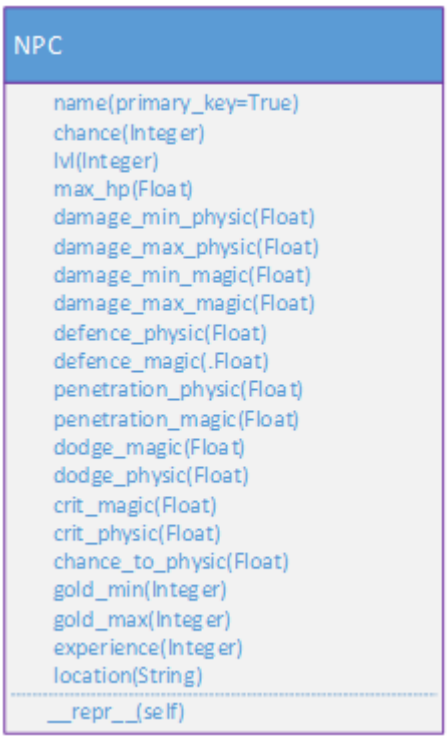


Рисунок 3.6 — UML діаграма класу «НП»

Окрім типових для звичайних персонажів атрибутів, даний клас містить наступні:

— chance – шанс на зустріч цього персонажу у певній локації;

— chance_to_physic – шанс на нанесеня фізичного пошкодження;

— gold_min – мінімальна кількість золота яку можна отримати за перемогу над цим персонажем;

— gold_max – максимальна кількість золота яку можна отримати за перемогу над цим персонажем;

— experience- кількість очків досвіду яку можна отримати за перемогу над цим персонажем;

— location – локація в якій можна зустріти цього персонажа.

На рисунку 3.7 представлено діаграму класа «Персональний предмет», який описує об’єкти предметів які належать конкретному персонажу.

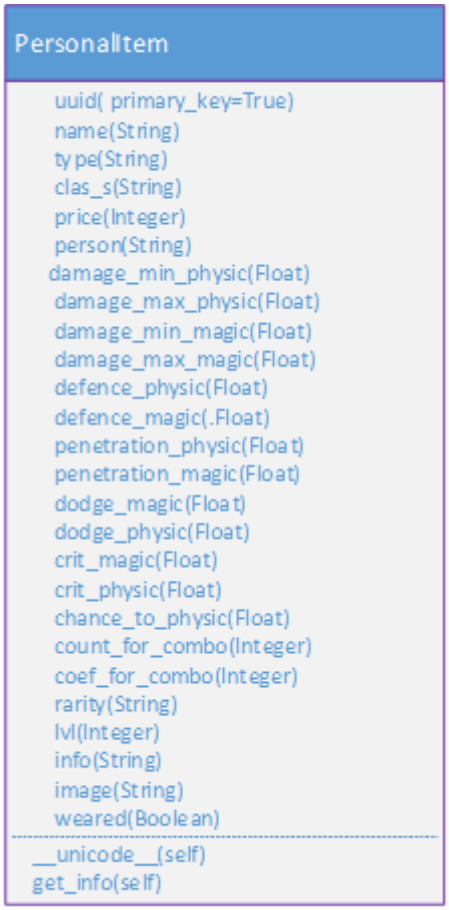


Рисунок 3.7 — UML діаграма класу «Персональний предмет»

Унікальні атрибути даного класу:

— type – тип предету;

— clas_s – клас предмету;

— price – ціна за яку можна продати цей предмет;

— rarity – рідкість з якою можна зустріти цей предмет;

— info – подробна інформація про предмет;

— image – картинка того, як виглядає даний предмет;

— weared – булева зміна, яка визначає вдягнений предмет на персонажа чи ні.

Окрім зазначених атрибутів, клас має наступні методи:

— __unicode__(self) – приймає об’єкт класу та повертає коротку інформацію про нього;

— get_info(self) – приймає об’єкт класу та відправляє гравцеві повідомлення з повним описом персонального предмета.

На рисунку 3.8 представлено діаграму класа «Локація», який описує об’єкт локації.

Рисунок 3.8 — UML діаграма класу «Локація»

Об’єкти локацій поки що мають лише один атрибут – назву локації, решта атрибутів, таких як час переміщення з одної локації до іншої, зазначені у інших класах.

На рисунку 3.9 представлено діаграму класа «Час», який описує час потрібний на переміщення між локаціями.

Рисунок 3.9 — UML діаграма класу «Локація»

Об’єкти даного класу мають наступні атрибути:

— id_ – унікальний ідентифікатор об’єкта;

— from_ – назва локації з якої проходить переміщення;

— to_ – назва локації в яку проходить переміщення;

— time – час потрібний на переміщення між відповідними локаціями.

3.3Розробка алгоритму та програмних елементів для тестового бою

Алгоритм тестового бою буде складатися з 3-х функцій, код яких зображено далі:

Функція «test_fight_comand(message)»

```
@bot.message_handler(func=lambda message: re.match(r'(^/test_fight_\w+$)\Z', message.text))
def test_fight_command(message):
    try:
        # pers = get_person(message.from_user.id)
        # if not pers.is_die and not pers.is_in_action:
        #     PvP(pers, get_person(extract_arg(message.text)))
        # else:
        lvl = message.text.split('_')[2]
        lvl_2 = message.text.split('_')[3]
        counts = message.text.split('_')[4]
        progress = session.query(Progress).filter_by(lvl=lvl).first()
        progress_2 = session.query(Progress).filter_by(lvl=lvl_2).first()
        temp_pers_1 = session.query(Person).filter_by(user_id="tester_1").first()
        temp_pers_2 = session.query(Person).filter_by(user_id="tester_2").first()
        try:
            session.delete(temp_pers_1)
            session.delete(temp_pers_2)
            session.commit()
        except Exception: pass
        temp_pers_1 = Person(user_id="tester_1", name="tester_1",
max_hp=progress.max_hp, damage_min_physic=progress.damage_min_physic,
                             damage_max_physic=progress.damage_max_physic,
dodge_physic=progress.dodge_physic,
```

```
                        dodge_magic=progress.dodge_magic,
defence_physic=progress.defence_physic,
                        defence_magic=progress.defence_magic,
crit_physic=progress.crit_physic,
                        crit_magic=progress.crit_magic,
penetration_magic=progress.penetration_magic,
                        penetration_physic=progress.penetration_physic,
chat_id=message.chat.id, accuracy_physic=0,
                        accuracy_magic=0)
            temp_pers_2 = Person(user_id="tester_2", name="tester_2",
max_hp=progress_2.max_hp, damage_min_physic=progress_2.damage_min_physic,
                        damage_max_physic=progress_2.damage_max_physic,
dodge_physic=progress_2.dodge_physic,
                        dodge_magic=progress_2.dodge_magic,
defence_physic=progress_2.defence_physic,
                        defence_magic=progress_2.defence_magic,
crit_physic=progress_2.crit_physic,
                        crit_magic=progress_2.crit_magic,
penetration_magic=progress_2.penetration_magic,
                        penetration_physic=progress_2.penetration_physic,
chat_id=message.chat.id, accuracy_physic=0,
                        accuracy_magic=0)
            session.add(temp_pers_1, temp_pers_2)
            session.commit()
            if int(counts) == 1:
                PvP(temp_pers_1, temp_pers_2)
            else:
                big_PvP(temp_pers_1, temp_pers_2, int(counts))
        except:
```

```
bot.send_message(message.chat.id, "Проверь команду еще раз")
```

Дана функція обернена в декоратор «bot.message_handler» через що викликається при відправлені повідомлення яке відповідає регулярному виразу «/test_fight_\w+$)\Z». Далі функція працює за наступним алгоритмом.

— отримання даних з регулярного виразу;

— отримання даних з бази даних;

— видалення попередніх тестових екземплярів персонажей;

— створення нових тестових екземплярів персонажей з вказаними параметрами;

— запис тестових персонажів у базу даних;

— якщо кількість поєдинків дорівнює 1-му, запускаємо звичайний алгоритм «PvP» (гравець проти гравця);

— інакше запускаємо функцію для тестових поєдинків «big_PvP(temp_pers_1, temp_pers_2, int(counts))» і передаємо потрібні параметри;

— у випадку помилки виводимо повідомлення про помилку.

Функція «big_PvP(temp_pers_1, temp_pers_2, int(counts))»:

```
def big_PvP(pers_1, pers_2, number):
    avarage_count = 0
    pers_1_wins = 0
    pers_2_wins = 0
    for i in range(0, number):
        count = 0
        pers_1.hp = pers_1.max_hp
        pers_2.hp = pers_2.max_hp
        pers_1.temp_count_for_combo = 0
        pers_2.temp_count_for_combo = 0
        pers_1_die = False
        pers_2_die = False
```

```
    first_fighter = random.randint(1, 2)
    chance_to_physic_1 = get_chance_to_physic(pers_1)
    chance_to_physic_2 = get_chance_to_physic(pers_2)
    while not pers_1_die and not pers_2_die:
        if first_fighter == 1:
            attack(pers_1, pers_2, chance_to_physic_1)
            if pers_2.hp <= 0:
                pers_2_die = True
            else:
                attack(pers_2, pers_1, chance_to_physic_2)
                if pers_1.hp <= 0:
                    pers_1_die = True
        else:
            attack(pers_2, pers_1, chance_to_physic_2)
            if pers_1.hp <= 0:
                pers_1_die = True
            else:
                attack(pers_1, pers_2, chance_to_physic_1)
                if pers_2.hp <= 0:
                    pers_2_die = True
        count += 1
    avarage_count += count
    if pers_1_die:
        pers_2_wins += 1
    else:
        pers_1_wins += 1
avarage_count = avarage_count/number
pers_1_wins = pers_1_wins * 100 / number
pers_2_wins = pers_2_wins * 100 / number
if pers_1.chat_id:
```

```
            bot.send_message(pers_1.chat_id,
                             u'Середня кількість тіків в битві:{0} \n Кількість перемог у
мершого гравця:{1}% \n'
                             u'Кількість перемог у другого
гравця:{2}%'.format(avarage_count, pers_1_wins,
                                                 pers_2_wins))
        pers_1.temp_count_for_combo = 0
        pers_2.temp_count_for_combo = 0
        session.add(pers_1)
        session.add(pers_2)
        session.commit()
        return
```

Дана функція імітує поєдинок між двумя персонажами визначену кількість раз, та працює за наступним алгоритмом.

— ініціалізуємо змінні;

— проходимось по циклу від 0 до визначеної кількості поєдинків;

— ініціалізуємо змінні, та зануляємо деякі з атрибутів персонажів;

— запускаємо цикл умовою виходу з якого є смерть одного з персонажів;

— в залежності від випадкової змінної визначаємо першочерговість нападу, а потім почергово визиваємо функцію нападу «def attack(attacker, defender, chance_to_physic)», яка приймає на вхід посилання на атакуючого персонажа та захисника, а також шанс на нанесення фізичної атаки;

— після кожного нападу дивимось чи життя одного з персонажів більше чи рівне 0;

— якщо так, то збільшуємо змінну «count» на одиницю та виходимо з внутрішнього циклу;

— після виходу з циклу додаємо отриману кількість раундів до загальної та збільшуємо на одиницю кількість перемог відповідного персонажа;

— після виходу із зовнішнього циклу рахуємо середню кількість раундів на поєдинок, та відсоток перемог кожного персонажа;

— виводимо результат;

— зануляємо зміні та записуємо в базу даних.

Функція «attack(attacker, defender, chance_to_physic)»:

```
def attack(atacker, defender, chance_to_physic):
    with_combo = False
    with_crit = False
    if atacker.weapon and random.uniform(0, 100) > chance_to_physic:
        if random.uniform(0, 100) > defender.dodge_magic - atacker.accuracy_magic:
            basic_damage = random.randint(atacker.damage_min_magic, atacker.damage_max_magic)
            if atacker.temp_count_for_combo == atacker.count_for_combo:
                with_combo = True
                basic_damage *= atacker.coef_for_combo
                atacker.temp_count_for_combo = 0
            else:
                atacker.temp_count_for_combo += 1
            if random.uniform(0, 100) <= atacker.crit_magic:
                with_crit = True
                damage = (coefCrit * basic_damage) - defender.defence_magic + atacker.penetration_magic
            else:
                damage = basic_damage - defender.defence_magic + atacker.penetration_magic
            defender.hp-=damage
            result = u"{0} нанес ".format(atacker.name)
            if with_crit:
```

```
                result += u"💥"
            if with_combo:
                result += u"□"
            result += u" 💁{0}урона, у {1} осталось {2}❤\n".format(damage,
defender.name, defender.hp)
        else:
            if random.uniform(0, 100) > atacker.temp_attack:
                atacker.temp_count_for_combo = 0
            result = u"{0} увернулся 💨\n".format(defender.name)
    else:
        if random.uniform(0, 100) > defender.dodge_physic -
atacker.accuracy_physic:
            basic_damage = random.randint(atacker.damage_min_physic,
atacker.damage_max_physic)
            if atacker.temp_count_for_combo == atacker.count_for_combo:
                with_combo = True
                basic_damage *= atacker.coef_for_combo
                atacker.temp_count_for_combo = 0
            else:
                atacker.temp_count_for_combo += 1
            if random.uniform(0, 100) <= atacker.crit_physic:
                with_crit = True
                damage = (coefCrit * basic_damage) - defender.defence_physic +
atacker.penetration_physic
            else:
                damage = basic_damage - defender.defence_physic +
atacker.penetration_physic
            defender.hp -= damage
            result = u"{0} нанес ".format(atacker.name)
            if with_crit:
```

```
                result += u"💥"
            if with_combo:
                result += u"□"
            result += u" ⚔{0}урона, у {1} осталось {2}❤\n".format(damage,
defender.name, defender.hp)
        else:
            if random.uniform(0, 100) > atacker.temp_attack:
                atacker.temp_count_for_combo = 0
            result = u"{0} увернулся 💨\n".format(defender.name)
        return result
```

Дана функція розраховує яке саме пошкодження наніс нападаючий захиснику. Вона працює наступним чином.

— ініціалізуємо зміні;

— якщо випадкова величина більша за шанс нанести фізичну атаку то переходимо до першого блоку в якому вираховується магічне пошкодженння;

— якщо випадкова зміна більша за різницю шансу ухилитися захисника та точності нападаючого то переходимо до підрахунку нанесеного пошкодження, в інакшому випадку переходимо до пункту 10;

— вираховуємо базове пошкодження, яке залежить від мінімально та максимально можливого пошкодження персонажа;

— якщо персонаж здійснив потрібну кількість успішних атак, то множимо базове пошкодження на коефіцієнт посиленої атаки, та зануляємо лічильник успішних атак;

— інакше збільшуємо лічильник на одиницю;

— якщо випадкова змінна менша або дорівнює шансу нанести критичне пошкодження, то вираховуємо результуюче пошкодження наступним чином: множимо базове пошкодження на коефіцієнт критичного пошкодження, слідом віднімаємо опір захисника та додаємо проникнення нападаючого;

— інакше розраховуємо результуюче пошкодження так само, проте без множення на коефіцієнт критичного пошкодження;

— віднімаємо від життя захисника результуюче пошкодження;

— якщо захисник ухилився, перевіряємо чи шанс на збереження ланцюга успішних атак більше ніж випадкова змінна, якщо так, то зануляємо лічильник успішних атак;

— якщо перевірка в пункті 2 не пройшла, то переходимо до другого блоку, де вираховується фізичне пошкодження, пункти аналогічні пунктам з 3 по 10;

— повертаємо результат.

3.4 Розробка структури інтерфейсу для ігрового Telegram боту

Загальна структура інтерфейсу головного вікна чат-ботів відповідає структурі будь-яких чатів у месенджері. Вона складається із панелі статусу, робочої області та панелі інструментів, яка складається з кнопок для відправлення повідомлень з різними вкладеннями.

Панель статус містить піктограму, кнопку виходу, кнопку меню налаштувань та коротку інформацію про бота.

При натисненні на панель статусу відкривається сторінка з повною доступною інформацією про бота. Окрім піктограми та назви в ній є кнопка відключення сповіщень, ім’я користувача та повний опис боту, наданий розробником.

Меню налаштувань дає можливість мінімального керування ботом, а саме пошуку по повідомленнях, відправки телефону, очищення історії, відключення сповіщень та видалення чату.

При розробці інтерфейсу програмного продукту було використано такі елементи керування: текстові повідомлення, іноді з прикріпленими картинками, декілька варіантів клавіатури з постійними кнопками, які змінюються в залежності від ситуації, та постійне меню, складене з основних потрібних користувачу команд. Інтерфейс програмного засобу містить в собі лише необхідну, в тій чи іншій ситуації для користувача, інформацію та кнопки керування.

На рисунках нище зображені розроблені структурні схеми інтерфейсів вікон програмного додатку.

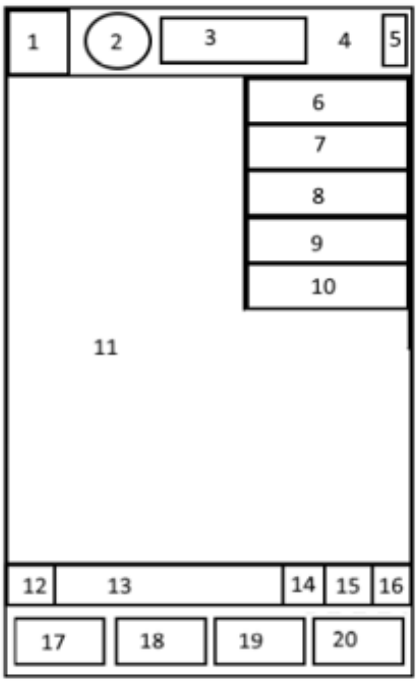


Рисунок 3.9 —Графічна схема головного вікна боту

Основні елементи інтерфейсу головного вікна боту «FireBox»:

1 Кнопка виходу.

2 Піктограма боту.

3 Назва боту.

4 Панель статусу.

5 Кнопка «Меню».

6 Пункт «Пошук».

7 Пункт «Відправити телефон».

8 Пункт «Очистити історію».

9 Пункт «Вимкнути сповіщення».

10 Пункт «Видалити чат».

11 Робоча область.

12 Кнопка для відправки емоцій та “.gif” анімацій.

13 Поле набору текстових повідомлень.

14 Кнопка переключення між клавіатурами.

15 Кнопка прикріплення файлів.

16 Кнопка запису аудіо та відео повідомлень.

17 Клавіша «Профіль»

18 Клавіша «Місто».

19 Клавіша «Пошук монстрів».

20 Клавіша «Атакувати».

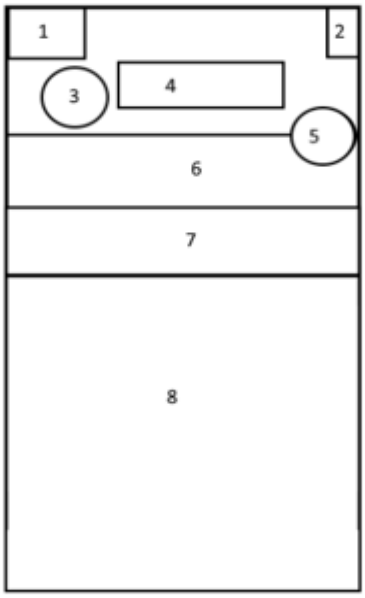


Рисунок 3.10 — Графічна схема вікна інформації про бота

Основні елементи інтерфейсу головного вікна боту «FireBox»:

1 Кнопка виходу.

2 Кнопка «Меню».

3 Піктограма боту.

4 Назва боту.

5 Кнопка «Головне вікно».

6 Ім’я користувача.

7 Кнопка «Виключити сповіщення».

8 Панель інформації про бота.

Розробка інтерфейсу є важливим етапом у процесі розробки чат-ботів, оскільки ефективність та зручність роботи користувача з ботом є вирішальним чинником, що визначає успішність розробленої системи.

4 ТЕСТУВАННЯ

4.1Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення (англ. Software Testing) — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись [15]. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:
відповідність вимогам, якими керувалися проектувальники та розробники;

— правильна відповідь для усіх можливих вхідних даних;

— виконання функцій за прийнятний час;

— практичність;

— сумісність з програмним забезпеченням та операційними системами;

— відповідність задачам замовника.

На сьогодні тестування програмного забезпечення – один з найбільш дорогих етапів життєвого циклу програмного забезпечення , на нього відводиться від 50% до 65% загальних витрат [16].
Є кілька основних методів тестування, які формують частину режиму тестування програмного забезпечення. Ці тести зазвичай вважаються самодостатніми в пошуку помилок і багів в усій системі.

Тестування методом «чорної скриньки» здійснюється без будь-яких знань внутрішньої роботи системи. Тестер симулюватиме дії користувача програмного середовища, надаючи різні входи і тестуючи згенеровані виходи. Цей тест також відомий як Black-box, closed-box тестування або функціональне тестування.

Тестування методом «білої скриньки», на відміну від «чорної скриньки», враховує внутрішнє функціонування і логіку роботи коду. Для виконання цього тесту, тестувальник повинен мати досвід програміста, щоб дізнатися точну частину коду, в якій виникають ті чи інші помилки. Цей тест також відомий як White-box, Open-Box або Glass box тестування.

Тестування методом «сірого ящика» — це щось середнє між тестуванням «білого ящика» та «чорного ящика», де тестувальник має лише керуватись загальними знаннями даного продукту, необхідними для виконання тесту. Ця перевірка здійснюється за допомогою документації та схеми інформаційних потоків. Тестування проводиться кінцевим користувачем, або користувачами, які представляються як кінцеві.

Враховуючи особливості кожного з розглянутих методів, для тестування програмного додатку було вирішено використати метод «чорної скриньки», оскільки цей метод дозволяє виявити помилки інтерфейсу, ініціалізації та завершення, некоректностні певних програмних модулів, що неможливо при тестуванні білої скриньки.

4.2 Тестування розробленого програмного продукту

Тестування розробленого програмного продукту проводиться за методикою «чорної скриньки». Вона базується на використанні шаблонів тестування, які називаються тест-кейсами. Це означає, що буде створено декілька тест-кейсів для перевірки правильності роботи основних функцій програмного додатку. Як приклад, деякі із них:

Тест-кейс №1 – Реєстрація в боті.

— відкрити бота;

— натиснути кнопку «Розпочати»;

— ввести команду реєстрації «/name (name)»;

— подивитись на відповідь бота.

Очікуваний результат — на екрані відображається повідомлення «Ваш профіль створено».

Враховуючи різноманітність «нік-неймів» користувачів, залежності від нього немає ніякої, тому обмеження на символи в ньому не накладаються. Система реєструє користувача, та сповіщає, що його персонаж знаходиться в локації «Місто». Результат виконання тест-кейсу №1 зображено на рисунку 4.1.

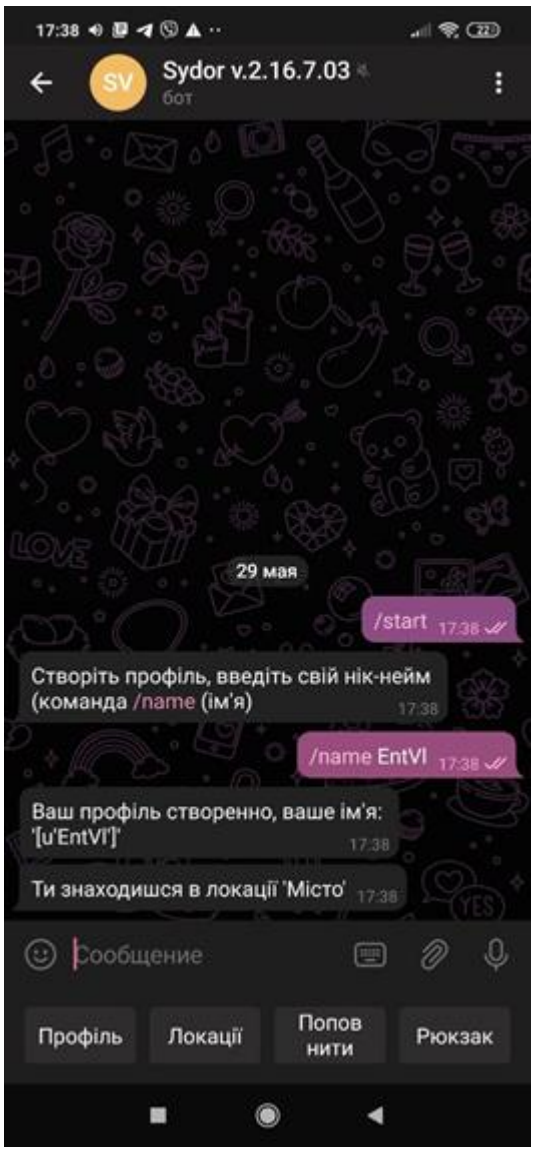


Рисунок 4.1 – Результат виконання тест-кейсу №1

Отриманий результат відповідає очікуваному, отже тест-кейс №1 успішно пройдено.

Тест-кейс №2 — Переміщення між локаціями.

— натиснути кнопку «Локації»;

— натиснути кнопку «Ліс»;

— дочекатись відповідь бота.

Очікуваний результат — через 15 секунд прийде повідомлення про прибуття в локацію.

Результат виконання тест-кейсу №2 показано на рисунку 4.2.

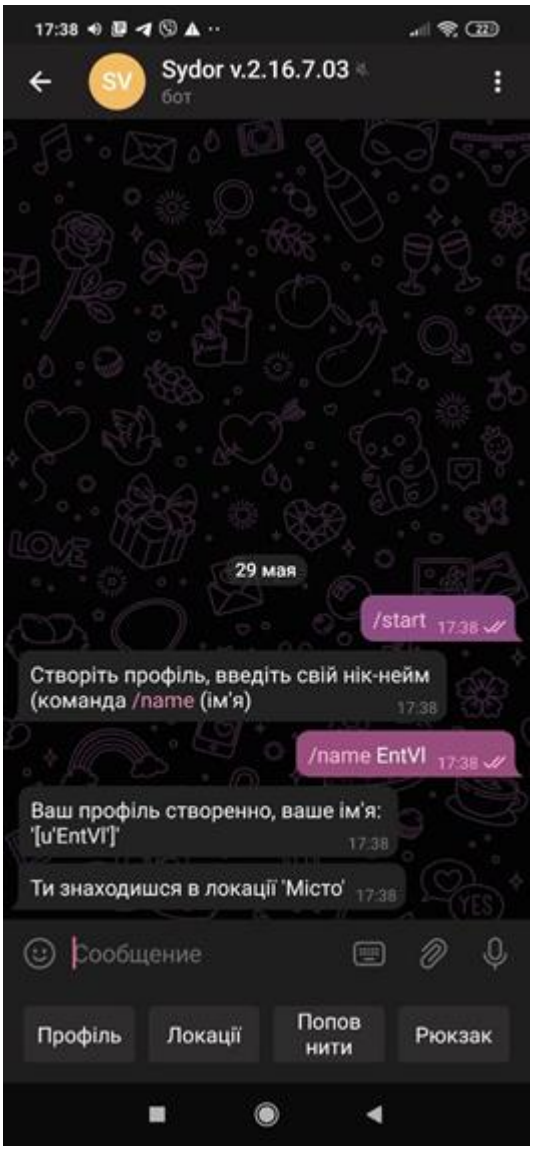


Рисунок 4.2 – Результат виконання тест-кейсу №2

Отриманий результат відповідає очікуваному, отже тест-кейс №2 успішно пройдено.

Тест-кейс №3 – Пошук супротивника та бій з ним.

— Натиснути кнопку «Пошук монстра».

— Дочекатись відповіді про знайденого супротивника.

— Натиснути кнопку «Напасти».

— Подивитись на відповідь бота.

Очікуваний результат — Відбудеться бій з монстром.

У кожній локації є різний шанс зустріти різних супротивників, проти кожного супротивника потрібна своя тактика, також з кожного з них можуть випасти свої речі. Результат тест-кейсу №3 зображено на рисунку 4.3.

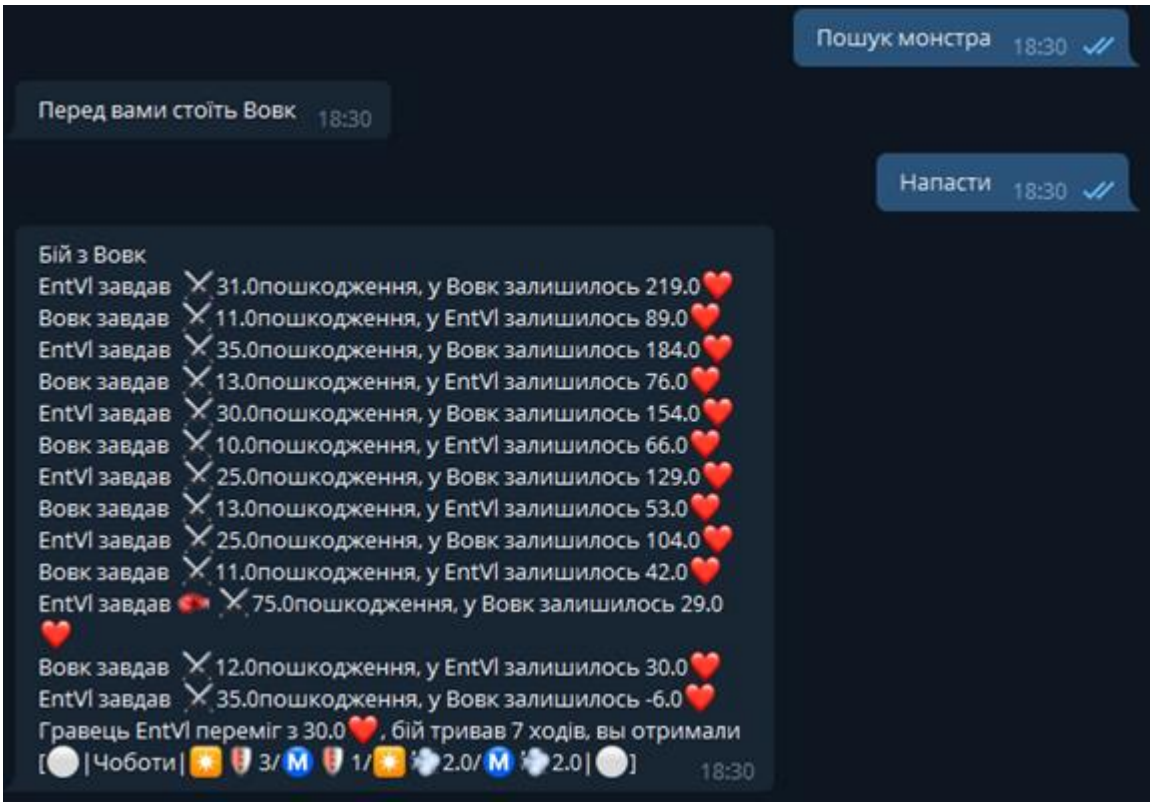


Рисунок 4.3 – Результат виконання тест-кейсу №3

Отриманий результат відповідає очікуваному, отже тест-кейс №3 успішно пройдено.

Тест-кейс №4 – Вдягання речі та випробування її в поєдинку.

— ввести команду «/backpack»;

— натиснуту команду «/wear» біля потрібної речі;

— перевірити стан інвентаря.

Очікуваний результат — інвентар працює справно, предмет вдягнувся на потрібне місце, та був вилучений з рюкзака. Результат виконання тест-кейсу №4 зображено на рисунку 4.4. Отриманий результат відповідає очікуваному, отже тест-кейс №4 успішно пройдено.

Рисунок 4.4 – Результат виконання тест-кейсу №4

Тест-кейс №5 – Перевірка механіки смерті.

— напасти на монстра маючи низький рівень життя;

— померти в поєдинку;

— дочекатись воскресіння;

— поповнити здоров’я.

Вдягнувши річ, видно, що той самий супротивник, почав наносити меньше пошкодження, що відбулось через підвищення характеристики «фізичний опір» через одягання на персонажа предмету «Чоботи». Після поразки, персонаж вмирає та через хвилину воскресає в локації «Місто». Результат виконання тест-кейсу №5 зображено на рисунку 4.5. Отриманий результат відповідає очікуваному, отже тест-кейс №5 успішно пройдено.

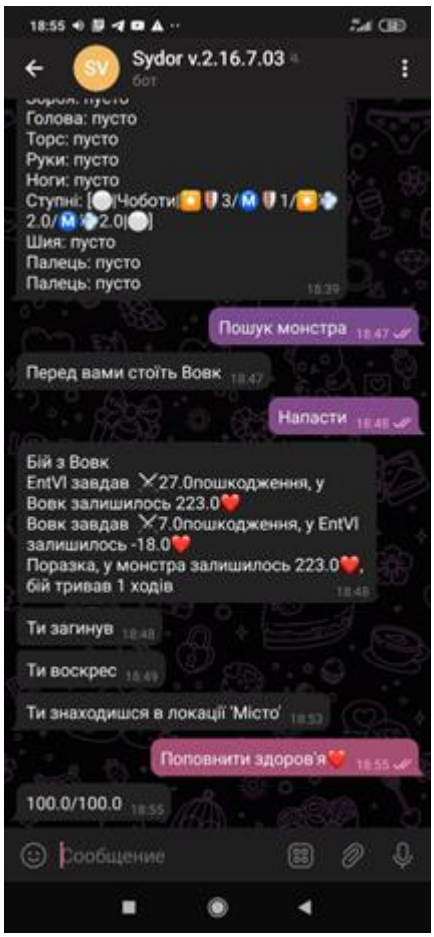


Рисунок 4.5 – Результат виконання тест-кейсу №5

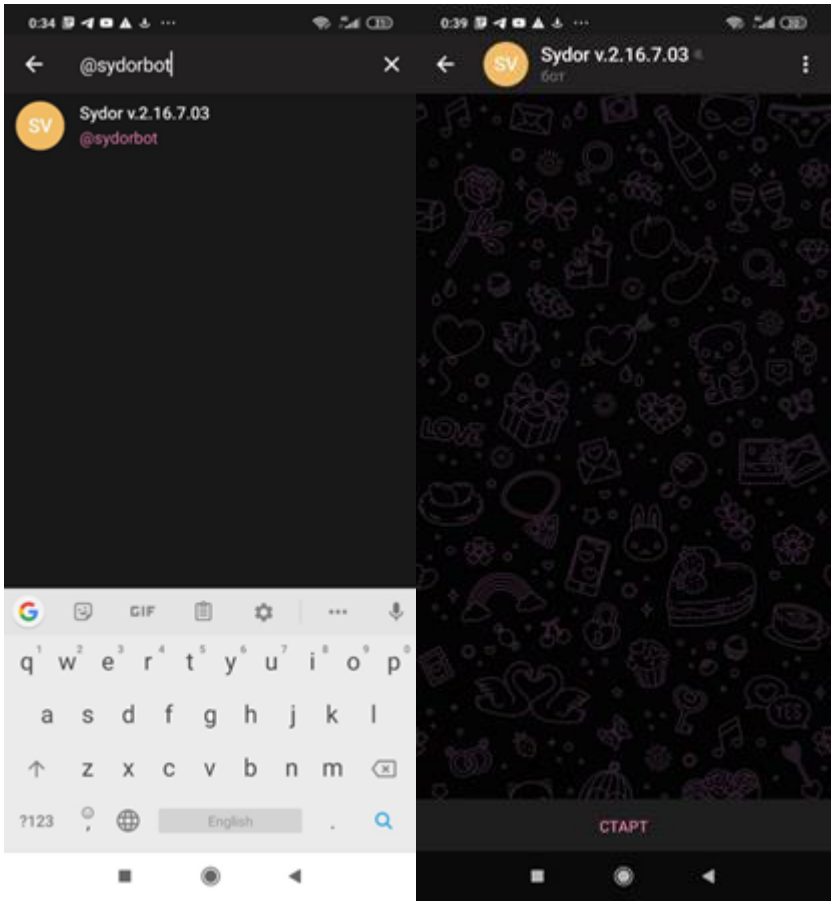
Було розроблено та виконано ще ряд різноманітних тест-кейсів, які перевіряють роботу усіх модулів Telegram-бота. Усі вони були успішно пройдені, у зв'язку з чим можна зробити висновок, що бот функціонує нормально і є придатним для використання.

4.3 Розробка інструкції користувача

«FireBox» — ігровий бот створений на базі мессенджеру «Telegram».

Для запуску боту потрібно встановити будь-який клієнт месенджеру та зареєструватись, або авторизуватись. Після входу на свій аккаунт потрібно в пошуку ввести «@sydorbot», що одночасно може виступати посиланням в смс-повідомленні (рисунок 4.6.а).

Далі потрібно зайти на аккаунт бота та натиснути команду старт, після якої бот привітається з користувачем та дасть інструкцію по реєстрації в грі (рис. 4.6.б).



а) б)

Рисунок 4.6 – Робота з ботом

Для реєстрації потрібно ввести команду «/name», після якої через пробіл вказати будь-який нік-нейм. Після реєстрації у користувача є свобода дії, обмежена лише наданими ботом командами. Ігровий процес у кожного індивідуальний, а навчати та допомогати новачкам будуть «ветерани» гри.

Окрім доступних команд на клавіатурі бот може розпізнавати команди швидкого доступу, такі як: «/backpack» та «/hero». Перша команда дозволяє відкрити та подивитися інвентар в будь-який момент, а друга дозволяє побачити свій профіль (рисунок 4.7).

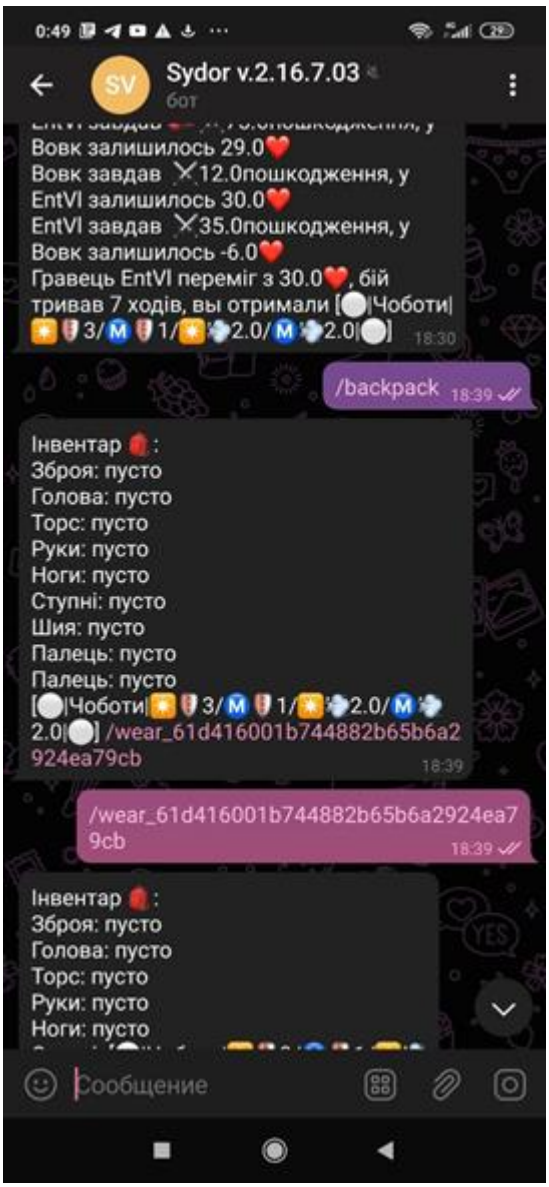


Рисунок 4.7 – Приклад використання швидких команд

У верхній правій частині бота при натисненні кнопки у вигляді 3-х крапок викликається випадне меню для керування ботом, в ньому є наступні функції:

— пошук по повідомленням;

— відправка телефону;

— очищення історії листування;

— включення/відключення сповіщень;

— видалення чату.

Приклад виклику меню зображено на рисунку 4.8.

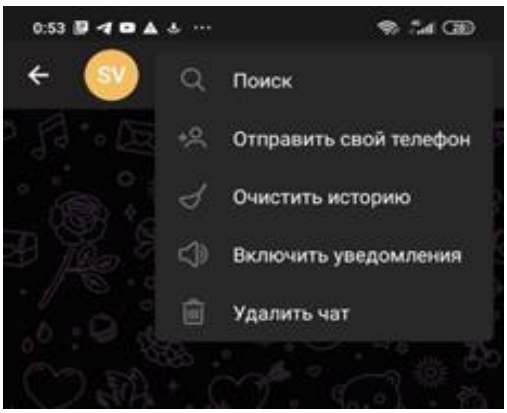


Рисунок 4.8 – Випадне меню для керування ботом

Натисканням на назву боту можливо відкрити вікно статусу, яке містить коротку інформацію про бота, кнопку для копіювання посилання на нього та перемикач для включення та відключення сповіщень (рисунок 4.9).

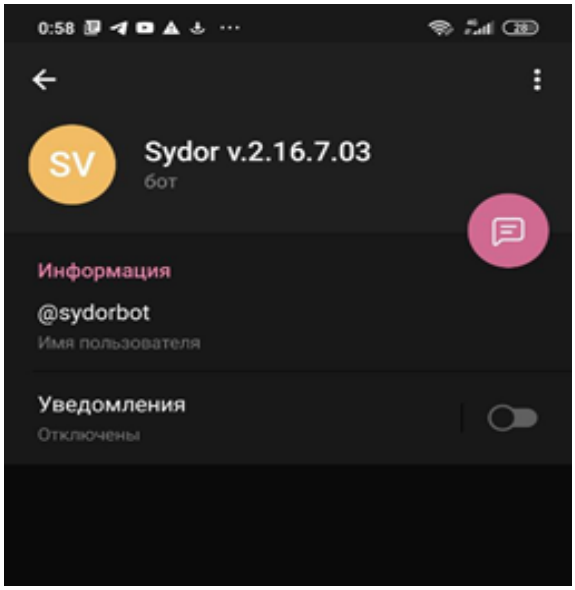


Рисунок 4.9 – Вікно статусу бота

У вікна статусу також є своє меню, яке містить наступні команди:

— додати в спільний чат;

— поділитись посиланням на бота;

— зупинити роботу бота;

— створити ярлик на робочому столі.

Виклик меню продемонстрований на рисунку 4.10.

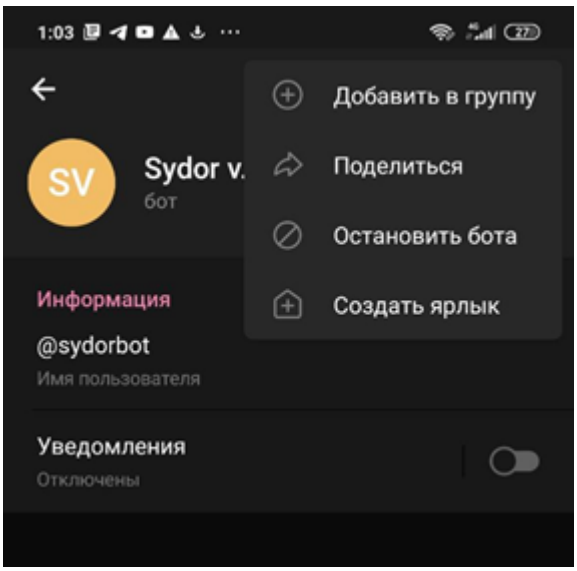


Рисунок 4.10 – Меню вікна статусу бота

Отже, було розроблено інструкцію користувача. В ній описано дії для початку роботи з ботом, та можливі дії користувача для досягнення того чи іншого результату.

4.4 Вимоги до персонального комп’ютера

Вимоги до складу і параметрів технічних засобів. Для запуску бота користувачу лише потрібно встановити мессенджер Telegram. Існують версії Telegram як і на телефони з ОС Android та IOS, так і версії на персональний комп’ютер і Web-версія [17]. Мінімальні та рекомендовані конфігурації телефону на базі ОС Android наведені в таблицях 4.1. та 4.2.

Таблиця 4.1 — Мінімальна конфігурація

Характеристика	Значення
Android	4.0 Ice Cream Sandwich
Об’єм оперативної пам’яті	512 МБ
Об’єм вбудованної пам’яті	2 ГБ

Таблиця 4.2 – Рекомендована конфігурація

Характеристика	Значення
Android	6.0 Ice Cream Sandwich
Об’єм оперативної пам’яті	1 ГБ
Об’єм вбудованної пам’яті	4 ГБ

Для того, щоб запустити бота після інсталяції мессенджеру Telegram, потрібно перейти по його посиланю «@sydorbot» та натиснути на запропоновану команду «/start».

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцюнювання комерційного потенціалу розробки «методи ігрового дизайну при розробці массової багатокористувацької рольової онлайн гри» або технологічний аудит розробки)

Проведення оцінювання комерційного потенціалу розробки є метою технологічного аудиту. Для проведення технологічного аудиту залучені три незалежних експерти: к.т.н., доцент Захарченко С.М., к.т.н., доцент Войцеховська О.В., к.т.н., доцент Богомолов С.В. Оцінювання комерційного потенціалу розробки здійснюється за 12-тю категоріями, наведені в таблиці 4.1

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам	Ціна продукту дещо нижче за ціни	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кр.	0	1	2	3	4
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Кінець таблиці 5.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки потрібно звести в таблицю за зразком таблиці 5.2.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	1.Захарченко С.М..	2. Войцеховська О.В.	3. Богомолов С.В.
	Бали, виставлені експертами:		
1	2	3	4
2	3	3	2
3	2	2	3
4	3	3	2
5	3	3	3
6	3	4	4
7	3	3	3
8	2	4	2
9	2	3	4
10	2	2	3
11	4	4	2
12	2	2	3
Сума балів	$СБ_1$=31	$СБ_2$=36	$СБ_3$=35
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ}=\frac{\sum_{1}^{3} СБ_i}{3}=34$		

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

В результаті було визначено, що рівень комерційного потенціалу розробки є вище середнього.

5.2 Прогнозування витрат на виконання науково-дослідної, (дослідно-конструкторської) та конструкторсько технологічної роботи

Прогнозування витрат на виконання даних робіт може складатись з таких етапів:

— розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

— розрахунок загальних витрат на виконання даної роботи;

— прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Першим етапом є розрахунок витрат, які безпосередньо стосуються виконавців даного розділу рботи, можна здійснити за такими статтями та формулами:

Основна заробітна плата кожного із розробників (дослідників) $З_о$, якщо вони працюють в наукових установах бюджетної сфери:

$$З_о = \frac{М}{Т_р} \cdot t \quad \text{грн.,} \qquad (5.1)$$

де М – місячний посадовий оклад конкретного розробника, грн.

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…22) дні;

t – число робочих днів роботи розробника (дослідника).

Таблиця 4.4 – Результат прогнозування витрат:

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
1. Науковий керівник	8000	380,1	25	9502,5
2. Інженер-програміст	7300	347,7	15	5215,5
3.Технічний інженер	5800	276,2	12	3314,4
Всього				18032,4

Додаткова заробітна плата $З_д$ всіх розробників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = (0{,}1...0{,}12) \cdot З_о + Зр \,. \qquad (5.2)$$

Для нашого випадку:

$$З_д = 0.1 \times (18032{,}4) = 1803{,}24 \text{ грн.}$$

Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою

$$H_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (5.3)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного соціального внеску ЄСВ, 22% .

Тоді

$$Н_{зп} = (18032,4+2637,7+1803,24) \times 0.22 = 4944,14 \text{ грн.}$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час (чи для) виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн.}, \quad (5.4)$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Для нашого випадку можна прийняти, що $Н_а$ = (10...25)%;

Т – термін, використання обладнання, приміщень тощо, місяці.

Таблиця 5.5 – Амортизація обладнання:

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.	Примітка

Кінець таблиці 5.5

Комп’ютери	28 000	10	2,5	583,3	
Приміщення лабораторії	70 000	10	3	1749,9	
Програмне забезпечення PyCharm	3500	10	3	87,5	
Всього				2442,35	

Витрати на матеріали М, що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.,} \quad (5.5)$$

де $Н_i$ – витрати матеріалу *і*-го найменування;

$Ц_i$ – вартість матеріалу *і*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1);

$В_i$ – маса відходів матеріалу *і*-го найменування;

$Ц_в$ – ціна відходів матеріалу *і*-го найменування, грн;

Таблиця 5.6 – Витрати на матеріал:

Найменування матеріалу	Ціна, грн	Витрачено, кг	Величина відходів,	Ціна відходів, грн	Вартість витраченого матеріалу, грн.	Примітка
Папір А4	95	-	-	-	95	
Диск	15,5	-	-	-	15,5	
Заправка для картриджа	109	-	-	-	109	
Всього					219,5	

Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ грн}, \quad (5.6)$$

де В – вартість 1 кВт-год. електроенергії, у 2020 р. В ≈ 3,85 грн./кВт;

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи обладнання, годин,

$К_п$ – коефіцієнт використання потужності; $К_п$ < 1.

$$В_е = 3,85 \times 1,413 \times 65 \times 0,85 = 300 \text{ грн.}$$

Витрати на послуги, що були використані під час виконання наукової розробки.

Таблиця 5.7 - Послуги, що використовуються при виготовленні дослідного зразка

Найменування послуг	Термін використання, місяців	Ціна за місяць, грн.	Сума, грн.
1. Послуга «AWS хостинг», шт.	2	400	800
Всього	4100 грн.		

Інші витрати $В_{ін}$ охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу.

$$В_{ін} = (1..3)* (З_о). \quad (4.7)$$

$$В_{ін} = 1 \times 18032,4 = 18032,4 \text{ грн.}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи – В.

В = 18032,4+1803,24+4944,14+2442,35+219,5+300+18032,4+800=
= 45774,19 грн.

Другим етапом є розрахунок загальних витрат на виконання даної роботи. Розрахунок загальних витрат здійснюється у тому випадку, коли дипломник виконує тільки певну частину даної роботи. У подальшому ця наукова робота буде продовжена.

Тоді загальна вартість всієї наукової роботи визначається за $В_{заг}$ формулою.

$$B_{заг} = \frac{B}{\alpha}, \qquad (5.8)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях, α =0,75

$$В_{заг} = \frac{45774,19}{0,75} = 62204,28$$

Третім етапом є прогнозування загальних витрат на виконання та впровадження результатів виконаної наукової роботи. Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, \qquad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Так, якщо розробка знаходиться:

— на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;

— на стадії технічного проектування, то $\beta \approx 0{,}2$;

— на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;

— на стадії розробки технологій, то $\beta \approx 0{,}4$;

— на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;

— на стадії розробки промислового зразка, $\beta \approx 0{,}7$;

— на стадії впровадження, то $\beta \approx 0{,}9$.

Оскільки дана розробка знаходиться на стадії розробки дослідного зразка , то $\beta = 0{,}5$

$$ЗВ = \frac{B_{заг}}{\beta} = \frac{62204{,}28}{0{,}5} = 124408{,}56$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки «Методи ігрового дизайну при розробці массової багатокористувацької рольової онлайн гри»

Прогнозування комерційних ефектів здійснюється у двох основних випадках:

— коли можна прямо оцінити зростання чистого прибутку підприємства від впровадження результатів наукової розробки;

— коли не можливо прямо оцінити зростання чистого прибутку підприємства від впровадження результатів наукової розробки.

Дана розробка підпадає під другий випадок, припустимо, що в результаті впровадження результатів наукової розробки покращується якість розробки, що дозволяє підвищити ціну його реалізації на 300 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 500 шт., протягом другого року – ще на 300 шт., протягом третього року – ще на 100 шт.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 10000 шт., а її ціна – 1500 грн.

Потрібно спрогнозувати збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Розрахунки здійснюватимемо за формулою 5.13:

$$\Delta П_i = \sum_1^n (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \quad (5.10)$$

де $\Delta Ц_o$ – покращення основного оціночного показника від впровадження результатів розробки у даному році, $\Delta Ц_o$=300 грн ;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки, N=10000 шт.;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки, ΔN=500 шт. ;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_0$=1500 грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки, n=3;

λ – коефіцієнт, який враховує сплату податку на додану вартість $\lambda = 0{,}8547$

ρ – коефіцієнт, який враховує рентабельність продукту, ρ = 0,25;

υ – ставка податку на прибуток, υ = 18%.

$$\Delta П_1 = [300 \cdot 10000 + (1500 + 300) \cdot 500] \cdot 0.8547 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right) = 663000$$

$$\Delta П_2 = [300 \cdot 10000 + (1500 + 300) \cdot (500 + 300)] \cdot 0.8547 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right) = 777000$$

$$\Delta П_3 = [300 \cdot 10000 + (1500 + 300) \cdot (500 + 300 + 100)] \cdot 0.8547 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right) = 80850$$

5.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$.

Для цього використаємо формулою:

$$E_{абс} = (ПП – PV), \quad (5.11)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

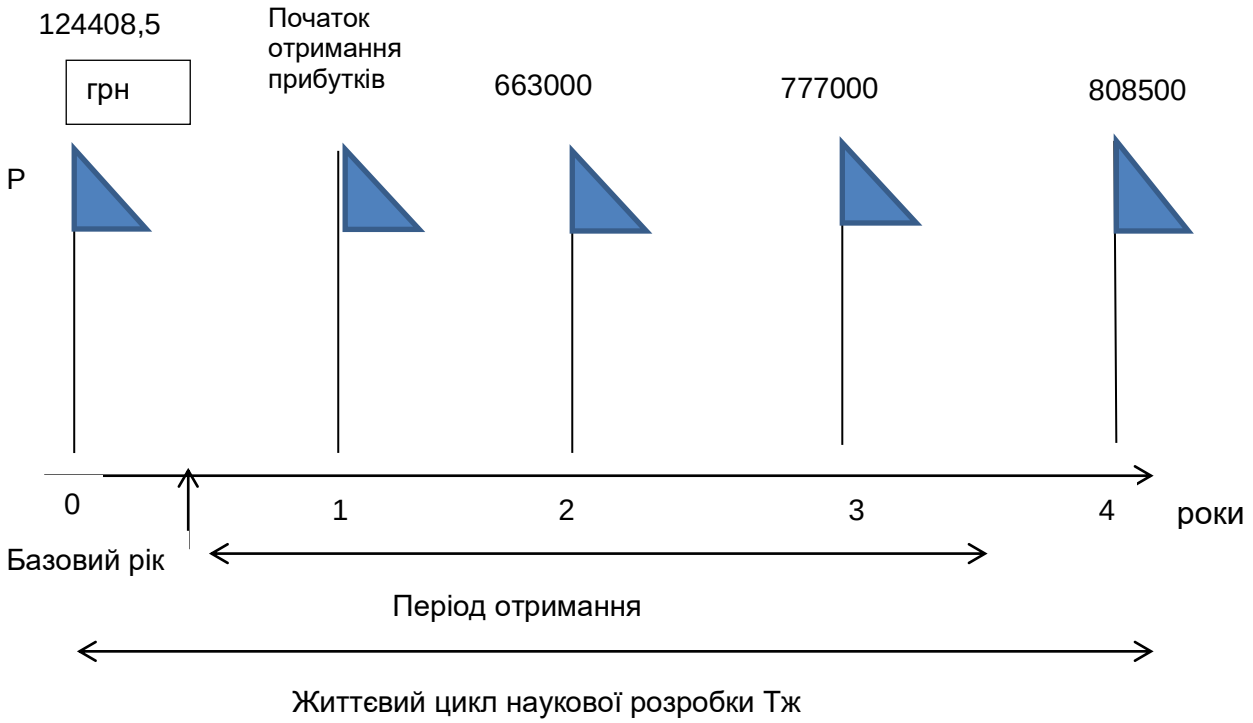


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{m} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, т=3;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; τ=0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{663000}{(1+0,1)^1} + \frac{777000}{(1+0,1)^2} + \frac{808500}{(1+0,1)^3} = 1852393$$

$$Е_{абс} = ПП - ЗВ = 1852393 - 124400 = 1727993$$

Пятим етапом є Розраховують відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього користуються формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.13)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV –теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[4]{1 + \frac{1727993}{124400}} - 1 = 0,964 \qquad (5.14)$$

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \qquad (5.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,17;

f – показник, що характеризує ризикованість вкладень; зазвичай, f = 0,08.

$$\tau = 0{,}17 + 0{,}08 = 0{,}25$$

Оскільки $E_в > \tau$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Термін окупності вкладених у реалізацію наукового пректу інвестицій $T_{ок}$ можна розрахувати за формулою:

$$T_{ок} = \frac{1}{E_в}. \quad (5.16)$$

$$T_{ок} = \frac{1}{0{,}964} = 1{,}04 \quad (5.17)$$

Оскільки $T_{ок} < 3$ років, то це свідчить про доцільність фінансування такої розробки.

Отже, в економічному розділі даної роботи було проведено розрахунок витрат: основна заробітна плата розробників(дослідників) склала 18032,4 грн. Додаткова заробітна плата всіх розробників складає 1803,24 грн. Нарахування на заробітну плату розробників — 4944,14 грн. Амортизація обладнання склала 2442,35 грн. На другому етапі розраховано суму загальних витрат — 62204,28 грн. Третій етап – прогнозування загальних витрат на виконання та впровадження результатів наукової роботи, він склав 124408,5 грн.

Прогнозування витрат комерційних ефектів. Було визначено, що дана розробка підпадає під другий випадок, оскільки вона знаходиться на стадії науково-дослідних робіт. В результаті впровадження результатів наукової розробки покращується якість, що дозволяє підвищити ціну його реалізації на 300 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом

першого року – на 500 шт., протягом другого року – ще на 300 шт., протягом третього року – ще на 100 шт.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 10000 шт., а її ціна – 1500 грн.

ВИСНОВКИ

Аналіз ринку ігрової індустрії підтвердив актуальність теми магістерської роботи та показав доцільність формалізації прроцесу ігрового дизайну при створенні масових багатокористувацьких рольових онлайн ігор. Аналіз методів ігрового дизайну та сучасних підходів до розробки масових багатокористувацьких рольових онлайн ігор показав, що найбільшим підходом до їх реалізапції є використання можливостей месенджеру Телеграм та ігр-ботів.

Аналіз методів розробки ігор показав, що для розробки масових багатокористувацьких рольових онлайн ігор найбльш доцільно використовувати інтеративний метод.

За результатами аналізу способів балансування даних, а також відповідно до методології транзитного балансування, були запропоновані способи оцінювання середньої шкоди для персонажа та результативного балансу шляхом зіставлення підсумків.

На основі аналізу способів балансування даних, а також методології транзитивного балансування було запропонований аналітичний вираз для спрощення та формалізації процесу розробки балансу. Було спроектовано "core"-механіку для масової багатокористувацької рольової онлайн гри.

В роботі були розроблені основні алгоритми роботи програмного продукту, а саме алгоритм генерації унікальних речей, алгоритми переміщення між локаціями, алгоритми PVP та PVE, а також запропоновано аналітичний вираз для балансування характеристик персонажів на різних рівнях. Кірм того запропоновано зрозумілий інтерфейс користувача.

Проведене тестування ігрового Telegram-боту проводилось за методикою «чорної скриньки». Результат тестування показав працездатність програмного продукту та відповідність поставленому технічному завданню. Розроблено інструкцію користувача, що допоможе користувачу полегшити користування програмним продуктом. Визначено мінімальну та рекомендовану конфігурації

персональних комп’ютерів та телефонів, необхідних для коректної роботи програмного засобу.

Результатом виконання магістерської кваліфікаційної роботи було вперше запропоновано використовувати транзитивний метод балансування для недетермінованих ігор, що дозволяє спростити процес їх розробки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Massively multiplayer online role-playing game [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/mmo_rpg

2. Game traffic analysis: An MMORPG perspective [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S1389128605003956

3. Identifying MMORPG Bots [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/article/10.1155/2009/797159

4. Newzoo Adjusts Global Games Forecasthttps://newzoo.com/insights/articles/newzoo-adjusts-global-games-forecast-to-148-8-billion-slower-growth-in-console-spending-starts-sooner-than-expected/

5. Инструкция по созданию Telegram ботов [Електронний ресурс] – Режим доступу до ресурсу: https://radiohlam.ru/telegram_bot_1/

6. Обзор существующих подходов к динамической балансировке сложности игрового процесса [Електронний ресурс] – Режим доступу до ресурсу: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Npdntu_inf_2016_2_14.pdf

7. Game design [Електронний ресурс] https://en.wikipedia.org/wiki/Game_design

8. Айан Шрайбер, Бренда Ромеро - Challenges for Game Designers

9. Tracy Fullerton. Game Design Workshop

10. Jesse Schell. The Art of Game Design: A Deck of Lenses

11. Katie Salen, Eric Zimmerman. Rules of Play: Game Design Fundamentals

12. Алгоритмы: построение и анализ / Т.Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн; пер. с англ. И. Красникова – Москва: ООО "И.Д. Вильямс", 2013. – 1328 с.

13. James Paul Gee. What Video Games Have to Teach Us About Learning and Literac

14. Robin Hunicke, Marc LeBlanc, Robert ZubekMDA - : A Formal Approach to Game Design and Game Research. [Електронний ресурс] - Режим доступу: https://users.cs.northwestern.edu/~hunicke/MDA.pdf

15. Калбертсон Р. Быстрое тестирование. / Р. Калбертсон, К. Браун, Г. Кобб. – М.: «Вильямс»,2002. – 374 с.–ISBN 5-8459-0336-X.

16. Тестирование програ́ммного обеспечения. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Тестирование_ программного _обеспечения

17. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. / Б. Бейзер — СПб.: Питер, 2004. — 320