

## РЕФЕРАТ

Темою дипломного проєкту є «Піксельна 2D гра в жанрі Vampire Survivors за допомогою двигуна Godot 4.3».

Дипломний проєкт: 38 рисунків, 3 таблиці, 5 лістингів, 47 джерел та 1 додаток.

Об'єкт дослідження: програмний модуль десктопного ігрового додатка у жанрі action roguelike.

Мета роботи: розробити десктопний ігровий додаток для платформ Android з використанням ігрового рушія Godot 4.3, що забезпечує захоплюючий ігровий досвід у жанрі action roguelike.

Постановка завдання: 16 сторінок. Інформація про вимоги до використовуваних технічних засобів, опис інструментальних засобів, що використовувались при розробці десктопного додатка.

Опис етапів реалізації: 15 сторінок. Докладна інформація про проєктування та розроблення десктопного додатка, включаючи розробку ігрової механіки, інтеграцію графіки та анімації, а також тестування функціоналу.

Опис програмного продукту: 10 сторінок. Докладна інструкція користувача для роботи з мобільним додатком, опис ігрового інтерфейсу та функцій.

Аналіз дослідної експлуатації: 3 сторінки. Інформація про тестування програмного додатку, тестування продуктивності та стабільності.

Охорона праці: 9 сторінок. Аналіз небезпечних та шкідливих чинників, які впливають на здоров'я людини на робочому місці за комп'ютером; розрахунок захисту від шкідливих виробничих факторів.


Економічна частина: 4 сторінки. Економічний розрахунок вартості розробки програмного продукту, аналіз витрат на розробку, маркетинг та підтримку мобільного додатка.

В даному дипломному проєкті засобами середовища Godot, розроблено десктопний ігровий додаток, який призначений для занурення користувача у світ підземних пригод у жанрі action roguelike.

Ключові слова: десктопний ігровий додаток, Godot, Windows, графіка, анімація, тестування, продуктивність.

## ЗМІСТ

ВСТУП.....	4
1 ПОСТАНОВКА ЗАВДАННЯ .....	6
1.1 Технічне завдання на розробку програмного продукту .....	6
1.1.1 Найменування розробки.....	6
1.1.2 Підстава для розробки .....	6
1.1.3 Призначення розробки .....	6
1.1.4 Вимоги до програмного продукту, що розробляється .....	6
1.1.5 Вимоги до програмного та апаратного забезпечення .....	9
1.1.6 Вимоги до програмної документації.....	9
1.1.7 Календарний план робіт .....	10
1.2 Огляд існуючих рішень .....	10
1.3 Обґрунтування вибору засобів розробки та мови програмування .....	19
2 ОПИС ЕТАПІВ РЕАЛІЗАЦІЇ.....	22
2.1 . Опрацювання можливостей ігрового рушія Godot 4.3 для створення ігрового проекту на платформі Windows. ....	22
2.2 Створення окремих ігрових об'єктів за завданням.....	24
2.3 Опис тестування та відлагодження ігрового додатку .....	35
3 ОПИС ПРОГРАМНОГО ПРОДУКТУ .....	37
4 АНАЛІЗ ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ.....	47
5 ОХОРОНА ПРАЦІ .....	49
5.1 Аналіз небезпечних і шкідливих виробничих чинників.....	49
5.2 Інженерно-технічні заходи з охорони праці .....	51
5.3 Пожежна профілактика .....	53
5.4 Заходи з ергономіки.....	55
6 ЕКОНОМІЧНА ЧАСТИНА.....	58

					ДП.ПЗ.211.06.ПЗ		
Змн.	Лист	№ докум.	Підпис	Дата			
Розроб.		Гуненко Я. М.			Піксельна 2D гра в жанрі Vampire Survivors за допомогою двигуна Godot 4.3		
Перевір.		Ланська С.С.					
Реценз.							
Н. контр.							
Затверд.							
					Літ.	Арк.	Аркушів
						2	70
					ВСП «ФКРКМ ДНУ»		

ВИСНОВКИ.....	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	65
ДОДАТОК А.....	70

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

## ВСТУП

Стрімке зростання ролі цифрових технологій у сучасному суспільстві призвело до переосмислення підходів до розробки комп'ютерних ігор. Ігрова індустрія сьогодні виступає не лише засобом розваги, а й одним із напрямів технічного та творчого розвитку. З появою нових інструментів створення ігор стало доступнішим для широкого кола розробників, що стимулює появу інноваційних проєктів у жанровому та технологічному різноманітті. Одним із таких інструментів є рушій Godot Engine, який у версії 4.3 демонструє значний прорив у можливостях розробки як двовимірних, так і тривимірних ігрових середовищ.

Тематика даної роботи присвячена розробці оригінальної комп'ютерної гри у жанрі action roguelike, натхненної концепцією популярної гри Brotato. Проєкт реалізовано для десктопної платформи Windows із використанням можливостей Godot 4.3. Основною ідеєю гри є виживання персонажа у хвилях ворогів, при цьому після успішного завершення кожної хвилі гравець отримує доступ до внутрішньоігрового магазину, де може придбати нову зброю або покращення за накопичені ресурси. Однією з ключових механік є система розвитку персонажа через накопичення досвіду за знищення ворогів та вибір карток посилень, що дозволяє адаптувати стратегію гри під власні потреби користувача. Також передбачено можливість збереження прогресу та налаштування локалізації для забезпечення ширшої доступності проєкту.

Процес реалізації даної гри вимагав вирішення низки складних технічних завдань, серед яких інтеграція системи магазинних транзакцій, розробка механізму накопичення і обробки досвіду, реалізація стійкої роботи хвиль ворогів із поступовим зростанням складності, а також забезпечення коректного збереження і відновлення ігрового прогресу. Додаткову увагу приділено адаптації користувацького інтерфейсу та налаштуванню багатомовної підтримки, що є одним із важливих аспектів сучасних ігрових продуктів.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

Вибір Godot Engine як основної технологічної бази обумовлений його відкритою природою, активною спільнотою розробників, підтримкою кросплатформених стандартів та потужними засобами інтеграції графічних, фізичних і логічних компонентів. Особливості версії 4.3, зокрема нові можливості в роботі з ресурсами, оптимізація системи сигналів і розширення інструментів розробки користувацьких сцен, стали важливими факторами успішної реалізації даного проєкту.

Таким чином, створення гри у зазначеному жанрі дозволяє не лише дослідити практичні аспекти використання сучасних технологій розробки ігор, а й продемонструвати ефективність застосування рушія Godot 4.3 для вирішення комплексних завдань у сфері інді-розробки на платформі Windows.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

# 1 ПОСТАНОВКА ЗАВДАННЯ

## 1.1 Технічне завдання на розробку програмного продукту

### 1.1.1 Найменування розробки

Темою дипломного проєкту є «Піксельна 2D гра в жанрі Vampire Survivors за допомогою двигуна Godot 4.3».

### 1.1.2 Підстава для розробки

Підставою для даної розробки слугує завдання для дипломного проєктування, яке видане відокремленим структурним підрозділом "Фаховий коледж ракетно-космічного машинобудування Дніпровського національного університету імені Олеся Гончара"

### 1.1.3 Призначення розробки

Ігровий застосунок призначений для роботи в операційному середовищі Windows. Програмний продукт реалізовано у жанрі roguelike shoot 'em up, що забезпечує тривалий і насичений ігровий процес. Механіка ґрунтується на послідовній елімінації хвиль супротивників, після кожної з яких користувач отримує можливість підвищення характеристик персонажа шляхом використання карт удосконалень та придбання нового озброєння у внутрішньоігровому магазині.

### 1.1.4 Вимоги до програмного продукту, що розробляється

Застосунок розроблено на рушії Godot 4.3 (конфігураційний файл project.godot) та призначено для запуску у середовищі Windows. Головна сцена MainScene.tscn розташована за шляхом res://src/Scenes/MainMenu/MainScene.tscn і

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

підвантажуються під час старту програми. Додатково у проєкті використано автозавантажувальні модулі. Нижче приведено призначення автозавантажувальних модулів:

а) WeaponDB — база даних озброєння;

б) GameManager — керування хвилями, сейвами й масштабуванням ігрових параметрів.

1) Структура ігрового процесу:

– раунд орієнтована модель: гравець послідовно знищує хвилі супротивників, що прогресивно ускладнюються;

– після кожної хвилі активується фаза підготовки для придбання зброї;

– GameManager зберігає номер хвилі, XP і HP множники та інші службові параметри; збереження ведеться у файли save\_slot\_X.json за шаблоном шляху user://save\_slot\_%d.json.

2) Система покращень:

– Карти покращень класифікуються за типом покращення поля статистики гравця

– Після підвищення рівня гравець обирає одну карту зі згенерованого набору.

– Активні карти безпосередньо змінюють базові характеристики героя (шкода, швидкість атаки, швидкість пересування тощо).

3) Крамниця озброєння:

– Асортимент формується WeaponDatabase з урахуванням хвилі й розподілу рідкісностей (COMMON, RARE, EPIC, LEGENDARY).

– Для кожного виду зброї визначені параметри: базова шкода, рівень, ціна.

– У разі покупки зброї вміст слота для покупки автоматично оновлюється на відповідну до хвилі зброю.

4) Характеристики героя:

Здоров'я персонажа має максимальне значення, яке зростає завдяки активним покращенням; отримувана шкода зменшується абсолютним показником

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

«Захист», тому чим вищий захист, тим менше втрат здоров'я за той самий удар. Досвід накопичується щоразу, коли гравець убиває супротивників під час зачистки хвилі; досягнення нового порога досвіду підвищує рівень і відкриває вибір карток покращення. Швидкість руху героя та швидкість атаки змінюються картами: достатньо підібрати відповідні покращення, щоб пересуватися або бити швидше. Аналогічно карти впливають на броню, підвищуючи її та дозволяючи впевненіше витримувати атаки, а також на шанс критичного удару, збільшуючи ймовірність завдати супротивнику посилену шкоду. Монети гравець отримує за кожного монстра, переможеного під час хвилі; їх можна витратити на подальші покращення чи спорядження. Нарешті, рівень персонажа підвищується автоматично, щойно сума досвіду досягає наступного необхідного значення, розрахованого за логікою гри, — і цей прогрес знову повертає нас до вибору нових карток, що підтримують безперервний цикл розвитку героя.

#### 5) Вороги:

- Типи супротивників відрізняються здоров'ям, шкодою та швидкістю руху.
- Покращення ворогів виконується відповідно процесу покращення гравця для балансування гри.
- Після смерті ворог створює об'єкт класу ExperienceItem, при контакті з гравцем якого гравцю нараховується значення досвіду.

#### 6) Озброєння:

- Категорії: ближня та дальня.
- Кожен екземпляр має рівень(tier), підвищення якого збільшує базові характеристики.
- Гравець може продавати, замінювати або об'єднувати (merge) зброю одразу у крамниці.

#### 7) Графічні та аудіоресурси:

- Статичні об'єкти: локація гри, стіни-обмежувачі.
- Динамічні об'єкти: вороги, снаряди.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		8



– Підтримуються три локалізації (ua, ru, en) через .po файли і систему TranslationServer.

– Аудіо-підсистема: саундтрек та SFX.

#### 8) Інтерфейс користувача:

– Екрани: головне меню, налаштування, ігровий рівень, сцена магазину, сцена збереження та завантаження ігрового процесу, сцени магазину та екран завершення гри у разі смерті гравця.

– Управління реалізовано за допомогою клавіатури.

#### 9) Система збереження:

– Ручне збереження доступне з меню паузи та на сцені магазину.

– Формат збереження – JSON-файл зі станом хвили, статистикою героя та інвентарем.

#### 10) Технічні вимоги:

– Мінімальна відеокарта – Radeo RX6600 або еквівалент (Vulkan, fallback OpenGL3).

– Підтримка вертикальної синхронізації вимкнена за замовчуванням.

– Застосунок використовує пакування ресурсів .tres та багаторівневу систему шарів фізики/рендеру.

### 1.1.5 Вимоги до програмного та апаратного забезпечення

Вимоги до програмного та апаратного забезпечення на етапі експлуатації є:

а) Наявність пристрою з операційною системою Windows 10 і вище.

б) Оперативний запам'ятовувальний пристрій об'ємом 1 Гб або більше.

в) 200 Мб вільного місця на накопичувачі.

### 1.1.6 Вимоги до програмної документації

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Основними документами, що регламентують розробку програм, повинні бути документи Єдиної системи програмної документації (ЄСПД): постановка задачі, опис етапів проєктування та реалізації додатку, керівництво користувача.

#### 1.1.7 Календарний план робіт

Календарний план робіт над проєктом наведений в таблиці 1.1.

Таблиця 1.1 – Календарний план робіт

№	Найменування етапів роботи	Дата виконання
1	Отримання та узгодження теми дипломного проєкту	01.04.2025
2	Формулювання та деталізація технічного завдання	15.04.2025
3	Розробка ключових модулів додатка	30.04.2025
4	Розділ «Охорона праці»	05.05.2025
5	Економічний розділ	15.05.2025
6	Захист дипломного проєкту	23.06.2025

#### 1.2 Огляд існуючих рішень

Упродовж останніх кількох років на ринку відеоігор сформувався своєрідний « пісочний годинник» уваги гравців: дедалі більше людей шукають короткі, але насичені сесії, які можна пройти « на одному подиху» дорогою на роботу чи в обідній перерві. У такому контексті стрімко виріс жанр Survivors like — мінімалістичні auto shooters, де герой рухається, а зброя стріляє сама. Першою ластівкою стала Vampire Survivors [1]: бюджетний інді проєкт несподівано перетворився на хіт з понад 30 млн USD виторгу, про що докладно писала GamesRadar+[9].

Аналітики GameDiscoverCo підраховали, що у 2024 році середня ціна таких тайтлів у магазині Steam становила всього 3–10 USD, а над більшістю з них працювали маленькі команди або навіть один два ентузіасти [8]. Високий попит і низький поріг входу для розробників запустили «ланцюгову реакцію»: з 2023 до

Вик.	Гуненко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					10
Змн.	Арк.	№ докум.	Підпис	Дата		

весни 2025 го на ПК та мобільних пристроях з'явилося вже понад дві сотні релізів із процедурною генерацією хвиль ворогів і короткими раундами тривалістю 10–20 хвилин.

Формула успіху проста, але ефектна: синергетичні збірки предметів дозволяють нарощувати шкоду експоненційно, даруючи гравцям відчуття «непереможної сили» вже на четвертій хвилині. Автоматичні атаки знижують поріг входу для новачка, тоді як наростаюча складність утримує хардкор. Саме тому далі розглядаються сім найпомітніших представників сегменту — Vampire Survivors [1], Brotato [2], 20 Minutes Till Dawn [3], Magic Survival [4], Soulstone Survivors [5], HoloCure – Save the Fans! [6] і класична roguelike гра The Binding of Isaac [7] — їхні ключові механіки, новаторські рішення й внесок у формування феномену «bullet heaven».

Vampire Survivors — перший об'єкт поглибленого аналізу. Інді-проект Лука Галанте (poncle) стартував у ранньому доступі 17 грудня 2021 р., а повний реліз відбувся 20 жовтня 2022 р. [1]. Гра відкрила хвилю Survivors-like: герой рухається вручну, а зброя стріляє автоматично, тому гравець зосереджується на мікропозиціонуванні й тактичному плануванні маршрутів. Основне завдання полягає у витримуванні дедалі щільніших хвиль супротивників і накопиченні золота для подальших апгрейдів. Протягом забігу користувач рівень за рівнем обирає випадкові поліпшення, формуючи синергії між пасивними бонусами та видами зброї; експоненційне зростання потужності дарує відчуття «power fantasy», а початкові бонуси на броню чи удачу спрощують ранні хвилини й роблять гру доступною новачкам [1]. Розмаїття забезпечують понад п'ятдесят видів зброї й пасивів, десятки унікальних персонажів і кілька процедурно згенерованих арен, завдяки чому кожен запуск відрізняється від попереднього. Скриншот ігрового процесу продемонстровано на рисунку 1.1.

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				11
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 1.1 – Ігровий процес Vampire Survivors

У серпні 2023 р. безплатне оновлення «Version 1.6» перевело проєкт з Phaser на Unity, підвищило продуктивність, додало локальний кооператив до чотирьох учасників і забезпечило повну підтримку миші, клавіатури, геймпадів і сенсорних екранів [8]. Міграція на новий рушій полегшила подальший порт на Nintendo Switch 2023 р. і PlayStation 4/5 2024 р. [9]. Станом на 3 травня 2025 р. у Steam зафіксовано 236 783 рецензії, з яких 98 % позначені як позитивні, що дозволило грі зберегти статус «Overwhelmingly Positive» [1]. Уже в 2023 р. виторг перевищив 30 млн USD, про що повідомляло видання GamesRadar+ [10]. Аналітичний звіт GameDiscoverCo підтверджує, що феномен Vampire Survivors породив сотні наслідувань та експериментальних варіацій, утверджуючи окрему нішу інді-розробки з середнім цінником 3–10 USD [11]. Гра отримала нагороди BAFTA 2023 р. у категоріях Best Game і Game Design, а також відзнаки Golden Joystick та D.I.C.E. Awards [9]. Короткі п'ятнадцяти-тридцятихвилинні сесії, стрімке експоненційне зростання сили персонажа й мета-прокачування зробили Vampire Survivors однаково привабливою для новачків і хардкорної аудиторії, встановивши стандарт, на який тепер орієнтується весь сегмент тайм-сурвайверів.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Brotato посідає другу позицію в аналізі. Автор під псевдонімом Blobfish випустив гру в ранньому доступі 27 вересня 2022 р., а повний реліз відбувся в січні 2024 р. [12]. Це динамічний аренний шутер із рогалайт-елементами, де самотня, але войовнича картоплина одночасно тримає до шести видів зброї й відбиває концентровані хвилі інопланетян. Гравець може перемкнутися між автоматичною стріляниною та ручним прицілюванням, тому стиль гри легко адаптувати до особистих уподобань. Сесії тривають від двадцяти до дев'яноста секунд, тож ігровий процес зберігає високу динаміку та чудово підходить для «швидких забігів» у перервах між справами.

Базова мета полягає у виживанні до прибуття допомоги; при цьому кожна хвиля завершується фазою закупівлі, де за зібране золото можна змінити зброю або докупити перки. Завдяки цьому гравець безперервно конструює унікальний «білд», комбінуючи пасивні бонуси з характеристиками персонажа. Скриншот ігрового процесу продемонстровано на рисунку 1.2.



Рисунок 1.2 – Ігровий процес Brotato

Наприклад, вибір героя з високою швидкістю руху й арсеналу контактної зброї перетворює забіг на ризиковану, але вибухову стратегію «хіт-енд-ран».

Вик.	Гуненко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					13
Змн.	Арк.	№ докум.	Підпис	Дата		

Рівень складності можна налаштувати через параметри здоров'я, шкоди та швидкості ворогів, тому Brotato лишається доступною як новачкам, так і хардкорним фанатам жанру. Повна версія додала локальний кооператив на чотирьох осіб і розширила пул зброї та персонажів, про що розробник повідомив у релізних нотатках «1.0 Launch & Co-op Update» [13].

Популярність гри підтверджує статистика Steam: станом на 3 травня 2025 р. зареєстровано понад 91 тис. рецензій, з яких 96 % позначені як позитивні, що забезпечує позначку «Overwhelmingly Positive» [2]. Завдяки гумористичному сетингу, швидкому темпу та глибині системи розвитку Brotato посідає помітне місце серед сучасних Survivors-like.

20 Minutes Till Dawn завершує тріаду флагманських Survivors-like. Індідев під ніком Flanne випустив гру в ранньому доступі 8 червня 2022 р., а повністю завершив релізний цикл торік; за цей час проєкт зібрав понад двадцять шість тисяч відгуків у Steam, із яких дев'яносто один відсоток позначені як позитивні [3]. На відміну від автоматизованих систем Vampire Survivors чи Brotato, шутер робить наголос на ручному керуванні: гравець одночасно рухається, цілиться і стріляє, вибудовуючи тактику ухилів під безупинним натиском лавкрафтівських істот. Базовий раунд триває рівно двадцять хвилин; для тих, хто поспішає, автор додав десятихвилинний режим, а невгамовним пропонується нескінченна арена. Поступове відкриття рівнів «Темряви» підвищує виклик до п'ятнадцяти разів поспіль і перетворює кожну наступну спробу на серйозне випробування реакції та винахідливості.

Вибір зброї, рун і активних умінь формує ядро метагри: пістолети, дробовики, вогнемети й навіть променеві гармати комбінуються з модифікаторами відмов ствола чи підпалювання, а руни додають пасивні бонуси на кшталт швидкої перезарядки або вибухових куль. Досвід, що падає з переможених ворогів, дозволяє піднімати рівень і миттєво вдосконалювати поточний білд, тоді як уламки кристалів, накопичені за забіг, витрачаються поза боєм на розблокування нових мисливців, більш потужної зброї та додаткових слотів рун. Такий ритм — напружений “багатокутник” ухилів і прицілювання,

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		14



відчутне нарощування потужності всередині матчу й постійна довгострокова винагорода між раундами — зробив 20 Minutes Till Dawn одним із найбільш цитованих прикладів еволюції жанру, коли ручний шутерний геймплей органічно поєднується з roguelite-метою [14]. Скриншот ігрового процесу продемонстровано на рисунку 1.3.

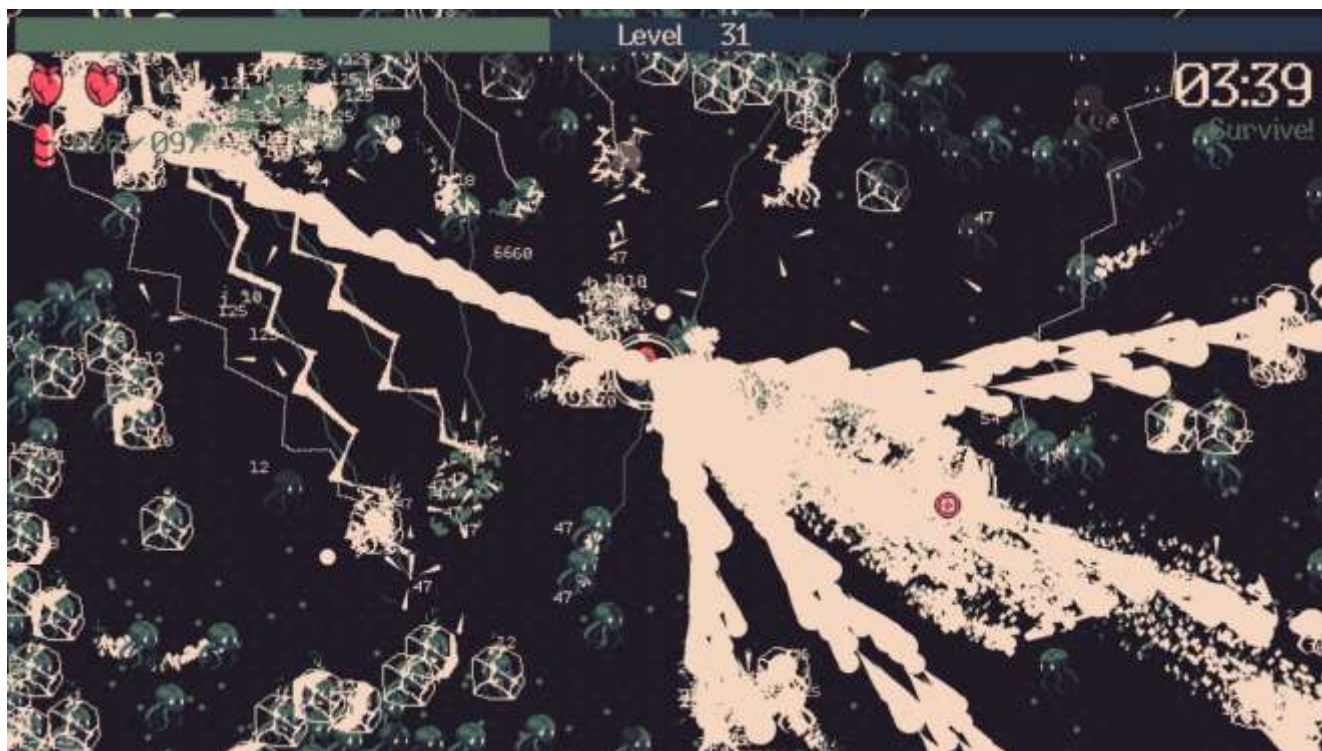


Рисунок 1.3 – Ігровий процес 20 Minutes Till Dawn

Magic Survival, мобільний ексклюзив студії LEME, з’явився восени 2021 р. як експериментальний «магічний сурвайвер» і швидко здобув культовий статус серед власників Android-пристроїв [4]. Гравець керує гомункулозом: пересування відбувається через віртуальний джойстик, тоді як заклинання спрацьовують автоматично, тому увага концентрується на ухилянні та стратегічному відборі умінь. Здобутий під час бою досвід миттєво перетворюється на рівні, які відкривають усе потужніші чарівні комбінації; після кожного апгрейду пропонується вибір із трьох випадкових умінь, що стимулює експерименти. Два формати проходження — таймерні арени з лімітом у сто ворогів і нескінченні забіги, де духи стають дедалі міцнішими, — забезпечують як короткі тренувальні сесії, так і затяжні випробування на витривалість.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Артефакти та випадковий дроп екіпіровки підсилюють базові закляття, а постійні патчі щомісяця додають нові елементи: нещодавнє оновлення «Emerald Grimoire» принесло ще п'ять книжок заклинань, три унікальні артефакти та групу елітних ворогів, які кардинально змінюють темп пізніх хвиль [15]. Регулярний контент-потік та проста, але глибока рольова система роблять Magic Survival привабливою для гравців, що люблять майструвати нетривіальні білди та відчувати швидке зростання сили без складного менеджменту інвентаря.

Soulstone Survivors, реліз якого відбувся 7 листопада 2022 р., продовжує еволюцію жанру «виживи-якомога-довше». Проєкт студії Game Smithing Ltd. позиціонується як динамічний рогаляк-арена, де мисливець за титанами крізь безперервні хвилі супротивників збирає «душекамені» — універсальну валюту для розблокування умінь і створення реліквій [5]. Уже у безплатній демо-версії гра здобула дев'яносто п'ять відсотків позитивних оцінок у Steam, а повна версія зберегла цей рівень схвалення, що свідчить про високу лояльність спільноти [16]. Скриншот ігрового процесу продемонстровано на рисунку 1.4.



Рисунок 1.4 – Ігровий процес Soulstone Survivors

Вик.	Гуменко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					16
Змн.	Арк.	№ докум.	Підпис	Дата		



Кожен із вісімнадцяти героїв вирізняється персональною зброєю, стартовою активною навичкою та набором прокачуваних параметрів, а багаторівневе дерево талантів охоплює сотні пасивних модифікаторів, що перетворюють навіть базові уміння на руйнівні комбінації. Перед стартом матчу гравець обирає до трьох рун, які кардинально змінюють початкові умови; наприклад, збільшують шанс випадіння епічних матеріалів або розширюють радіус підбирання душекаменів. На карті підсвічуються портали проклять: активація підсилює ворогів, але збільшує шанс випадіння реліквій вищої рідкості, тому ризик винагороджується. Кожна з кількох арен має унікальне оформлення, набір босів і власний перелік матеріалів, необхідних для крафту легендарної зброї; останнє велике оновлення «Titan's Reach» додало пустельну локацію з піщаними червами-еліксирами та шість нових активних умінь [17]. Таким чином, акцент на синергетичних білдах, глибокому метапрогресі та різноманітті рівнів робить Soulstone Survivors вагомим конкурентом серед сучасних rogue-arena.

HoloCure – Save the Fans! продемонструвала, що фанатський ентузіазм здатен породити повноцінний хіт. Після релізу в Steam 16 вересня 2023 р. безплатний рогалайт Kay Yu миттю став вірусною сенсацією: понад тридцять сім тисяч рецензій, з яких дев'яносто дев'ять відсотків позитивні, підтвердили, що комбінація механік Vampire Survivors [1] і Magic Survival [4] укупі з упізнаваними VTuber Hololive зачіпає емоції аудиторії [6]. Гравець керує обраною стримеркою, ухиляється від «фандомних» хвиль і поступово нарощує силу через гачу-систему: валюту, зароблену в бою, можна витратити на випадковий «пак», який відкриває нового персонажа або поліпшує наявного. Кожна з сорока семи героїнь має власну пасивну здатність і зрелищну «ульту», а зброя та допоміжні предмети, що випадають під час матчу, створюють лавиноподібні комбінації. У поєднанні з реміксами офіційних треків Hololive це перетворює кожен забіг на міні-концерт, де ритм битви синхронізується з музикою. Рейтингові таблиці рангу заохочують перфекціоністів удосконалювати маршрути, а регулярні патчі — зокрема оновлення «Blue Journey», що принесло п'ять нових учасниць і кооперативний «Live Stage» із гостьовими босами [18] — підтримують постійний інтерес

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

спільноти. Автоматична атака спрощує вхід для нових гравців, тоді як глибина білдів і безплатна модель монетизації забезпечують довгострокову реіграбельність. Скриншот ігрового процесу продемонстровано на рисунку 1.5.



Рисунок 1.5 – Ігровий процес HoloCure – Save the Fans

The Binding of Isaac: Rebirth підсумовує добірку як «класика жанру». Ремейк 2014 р. на новому рушії не тільки додав значно більше контенту, а й розширив технічні можливості оригіналу 2011 р. [7]. Процедурно генеровані підземелля, дев'ять стартових персонажів та понад шістсот активних і пасивних предметів забезпечують майже безкінечну варіативність, а складні боси, альтернативні маршрути й кілька кінцівок стимулюють десятки повторних проходжень. Акцент на ручному прицілюванні, ухилянні та ситуативному використанні предметів робить Isaac суттєво складнішим за авто-шутерні Survivors-like; локальний кооператив у режимі Rebirth додає соціальної динаміки, дозволяючи другові виступати «духом-помічником». Завдяки постійним безплатним оновленням і великим DLC (Afterbirth, Repentance) гра зберігає дев'яносто сім відсотків позитивних відгуків серед понад трьохсот тисяч користувачів Steam [7]. Скриншот ігрового процесу продемонстровано на рисунку 1.6.

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				18
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 1.6 – Ігровий процес Binding of Isaac: Rebirth

Аналіз семи тайтлів підтверджує потенціал формули «вижити якомога довше». Vampire Survivors і Brotato демонструють, як автоматизований бій та короткі сесії приваблюють масову аудиторію [1; 2], тоді як The Binding of Isaac показує, що глибока ручна механіка й величезний лут-пул утримують ветеранів [7]. 20 Minutes Till Dawn додає активне прицілювання кулею-за-кулею [3]; Magic Survival робить ставку на магічні комбо та мобільну доступність [4]; Soulstone Survivors розширює метапрогрес за допомогою дерев талантів і рун [5]; а HoloCure привносить фанатський сетинг Hololive і систему гачи, що постійно підживлює цікавість спільноти [6]. Разом ці проєкти формують різноманітний спектр піджанру, поєднуючи простоту входу з високою реіграбельністю та доводячи, що «bullet-heaven» залишається перспективним напрямом інді-розробки.

### 1.3 Обґрунтування вибору засобів розробки та мови програмування

Розроблення програмного продукту розпочинається з обґрунтованого вибору технологічної платформи, адже саме вона задає темп ітерацій, спектр інструментальних засобів і довгострокові витрати супроводу. У контексті

Вик.	Гуменко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					19
Змн.	Арк.	№ докум.	Підпис	Дата		

дипломного проєкту, присвяченого грі формату bullet-heaven, критично важливими виявилися безроялтіна ліцензія, сучасний графічний стек, мінімальний бар'єр входу для швидкого прототипування та активна технічна спільнота. З огляду на ці вимоги було обрано Godot Engine 4.3 і інтегровану мову GDScript 2.0. Рушій, що розвивається під ліцензією MIT і працює на Vulkan, поєднує відкритість вихідного коду з продуктивністю, достатньою для відтворення сотень об'єктів одночасно, характерних для ігор із високою щільністю ворогів. Його вузлова архітектура побудована на дереві об'єктів-вузлів і системі сигналів, тому керований герой, таймер хвиль, спрайтові вороги й елементи інтерфейсу існують у спільній ієрархії без громіздких менеджерів.

Версія 4.3 остаточно перейшла на Vulkan-рендер: рушій формує ациклічний граф команд, перемикає пакети шейдерів асинхронно й дозволяє процесору розраховувати штучний інтелект, поки відеокарта завершує постобробку. У внутрішньому профайлері середня тривалість кадру падає майже удвічі порівняно з гілкою 3.x, оскільки батчинг спрайтів і спільне кешування текстур мінімізують draw-call і споживання VRAM. Паралельно оновлено фізичне ядро GodotPhysics: інкрементальне BVH-дерево та «island sleeping» помітно скорочують навантаження ЦП, а безперервне зіткнення на основі swept AABB запобігає «тунелюванню» швидких снарядів. Завдяки цьому тестова сцена із шістсот ворогами стабільно утримує понад сто двадцять кадрів на секунду на відеоадаптері середнього класу, що відповідає вимогам жанру.

GDScript 2.0, інтегрований у ядро рушія, поєднує синтаксичну лаконічність Python зі статичною типізацією та компілюється у власний байт-код, який напряду працює з C++-класами Godot. Інкрементальний збирач сміття скорочує паузи навіть під час масового спавну пікапів, а гаряче перезавантаження скриптів скорочує цикл «редагування — перевірка» до кількох секунд. Анотація @tool дозволяє виконувати код безпосередньо у редакторі, тож дизайнер змінює кулдаун зброї й одразу бачить результат у вже запущеній сцені. Дані зброї та ворогів зберігаються у файлах-ресурсах .tres, які рушій кешує при старті: це розділяє логіку і балансні параметри та зменшує зчеплення модулів.

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Система імпорту ресурсів у 4.3 автоматично перетворює вихідні спрайти у компресований формат BasisU UASTC або ETC2, зберігаючи оптимізовані копії у «.import» файлах. Компресор працює в окремому потоці, тому навіть атласи 4096 × 4096 обробляються без блокування редактора. Оновлена анімаційна підсистема підтримує ретаргетування через SkeletonProfile: рушій у реальному часі перераховує матриці поз, тому бібліотеки рухів із Mixamo імпортуються без сторонніх скриптів. Поліпшена система вводу абстрагує клавіатуру, геймпад, мишу та сенсорні жести в єдиний об'єкт Input Event, що спрощує реалізацію twin-stick керування і дає змогу тестувати проєкт одночасно на ПК та Android без зміни коду.

Експортний пайплайн підтримує Windows, Linux, macOS, Android, iOS і WebAssembly. Під час створення Web-збірки рушій додає JavaScript-шар з підтримкою WebAssembly Threads і зберігає прогрес у IndexedDB; на Android використовується формат AAB, який скорочує початкове завантаження майже утричі відносно необробленого APK. Автоматизація збірок у GitHub Actions включає lint-перевірку, юніт-тести на GUT і автоматичне завантаження артефактів на itch.io, що забезпечує контроль якості та повторюваність релізів.

Godot розвивається за прозорим графіком: версії 4.0, 4.1, 4.2 та 4.3 вийшли з інтервалом близько семи місяців, а поточна 4.3 має статус supported і гарантовано отримуватиме патчі до появи 4.5. Активна спільнота з понад ста тисяч учасників, тисячі плагінів в Asset Library і регулярні джеми забезпечують швидкий обмін рішеннями й формують культуру відкритого коду, що співзвучно з освітньою метою дипломного дослідження.

Таким чином, Godot 4.3 у поєднанні з GDScript 2.0 пропонує безроятійну модель ліцензування, вузлову архітектуру, оптимізований Vulkan-рендер, інструментальний імпорт ресурсів, удосконалене фізичне ядро та мультиплатформний експорт. Сукупність цих характеристик гарантує продуктивність і гнучкість, необхідні для реалізації інтенсивної гри-«виживалки», і водночас надає студенту-розробнику швидкі ітерації та доступ до дослідницького потенціалу відкритої спільноти.

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

## 2 ОПИС ЕТАПІВ РЕАЛІЗАЦІЇ

### 2.1 . Опрацювання можливостей ігрового рушія Godot 4.3 для створення ігрового проєкту на платформі Windows.

Мета цього дипломного проєкту полягає у вивченні можливостей сучасних інструментів для створення ігрових застосунків під Windows, з акцентом на практичне використання рушія Godot. В основі ігрової концепції — поєднання жанрів аренного шутера та роґалика. Гравцеві пропонується захищати свого персонажа від хвиль ворожих істот, що з'являються одна за одною. Після кожної хвилі він має змогу отримати нову зброю, а з підвищенням рівня — покращити характеристики героя. Такий підхід формує систему постійного прогресу, яка заохочує гравця до подальшої участі в грі. Для більш детального відображення влаштування мого додатку було створено діаграму переходів між сценами, що продемонстрована на рисунку 2.1.

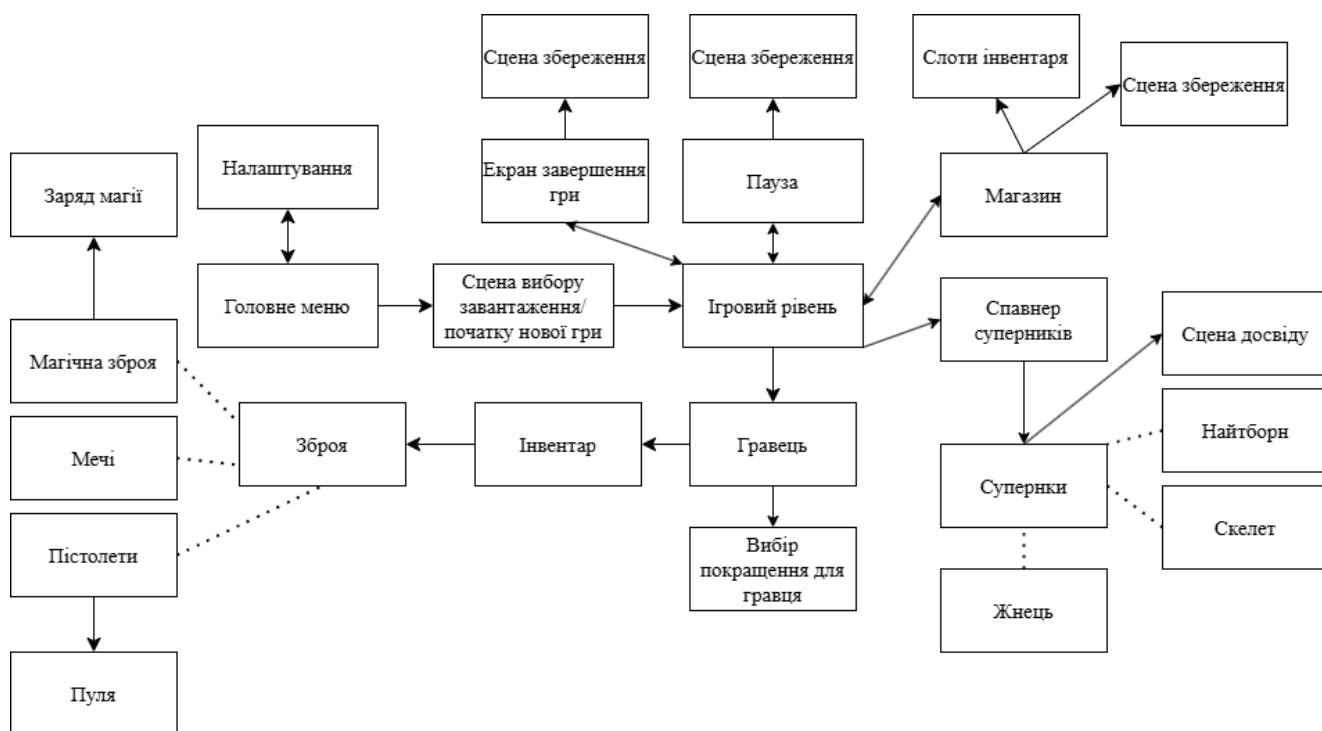


Рисунок 2.1 – Діаграма переходів між сценами

У процесі розробки було використано низку програмних засобів, що доповнюють один одного. Візуальні елементи гри — моделі персонажів, ворогів



та фону — створювались за допомогою онлайн-платформи PixilArt та графічного редактора Krita. Обидва ці інструменти дозволили досягти стилістики піксель-арту, яка гармонійно поєднується з ретро-настроєм гри. Логіку ігрового процесу реалізовано за допомогою мови GDScript безпосередньо в середовищі Godot. Середовище забезпечило зручну роботу з анімаціями, спрайтами та фізикою зіткнень. Завдяки компонентам AnimatedSprite2D та SpriteSheet вдалось реалізувати плавні рухи персонажів і підвищити загальну візуальну якість гри.

Результатом роботи стала повністю функціональна гра під назвою «DarkLifer», у якій втілено не лише теоретичні знання, а й особисті напрацювання автора. Всі етапи проєктування й реалізації узгоджені з чинними державними стандартами (ДСТУ), що регламентують структуру та оформлення дипломних робіт.

У процесі роботи над додатком було створено ігрові сцени за допомогою вбудованих у рушій елементів, а саме TileMapLayer для сцени ігрової локації та ParallaxBackground для головного меню. Роботу над фоном з ефектом паралакс продемонстровано на рисунках 2.2 – 2.3.

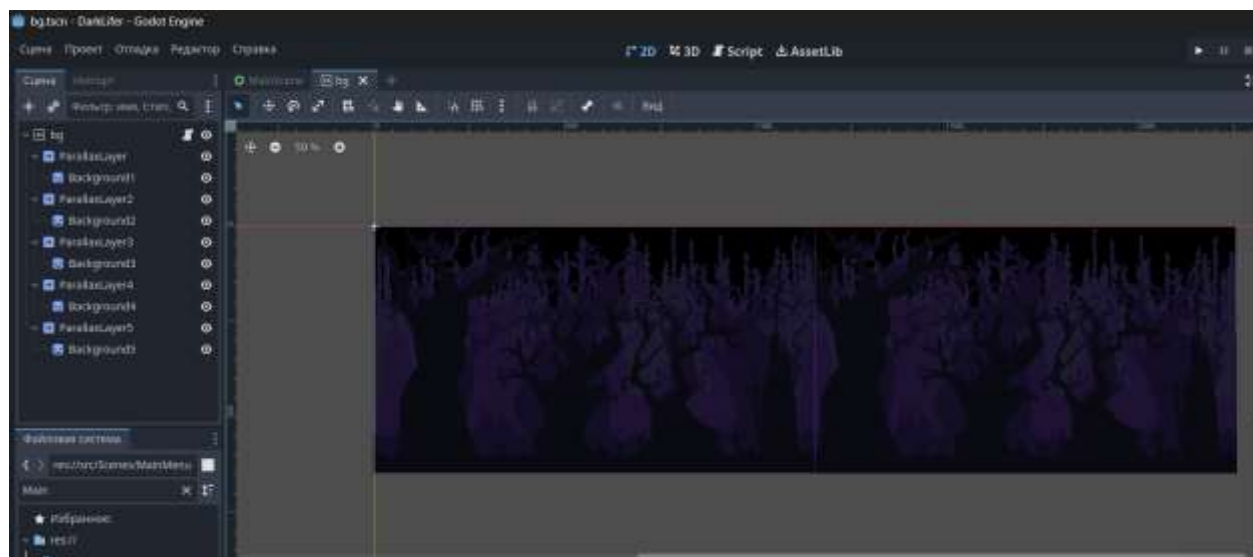


Рисунок 2.2 – Сцена bg.tscn створена за допомогою інструменту ParallaxLayer

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

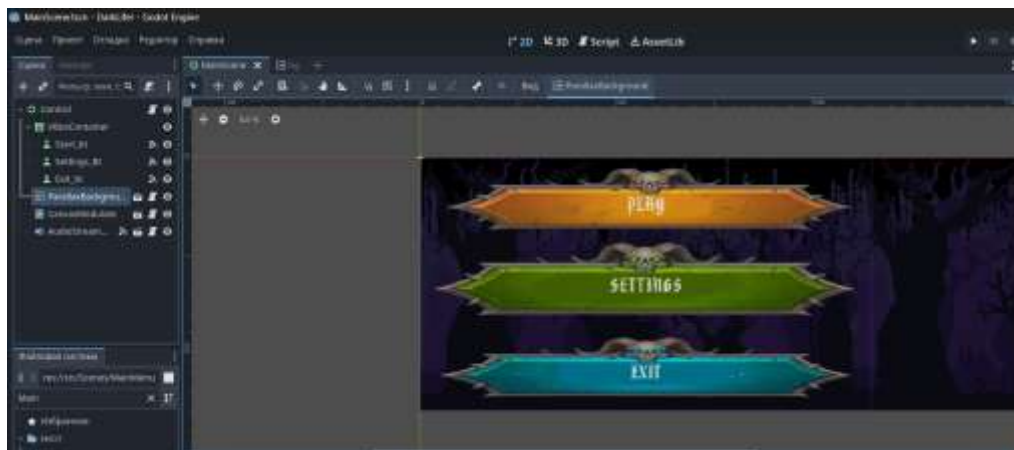


Рисунок 2.3 – Приклад використання у сцені фону з ефектом паралакс

Роботу над інструментом TileMapLayer продемонстровано на рисунку 2.4.

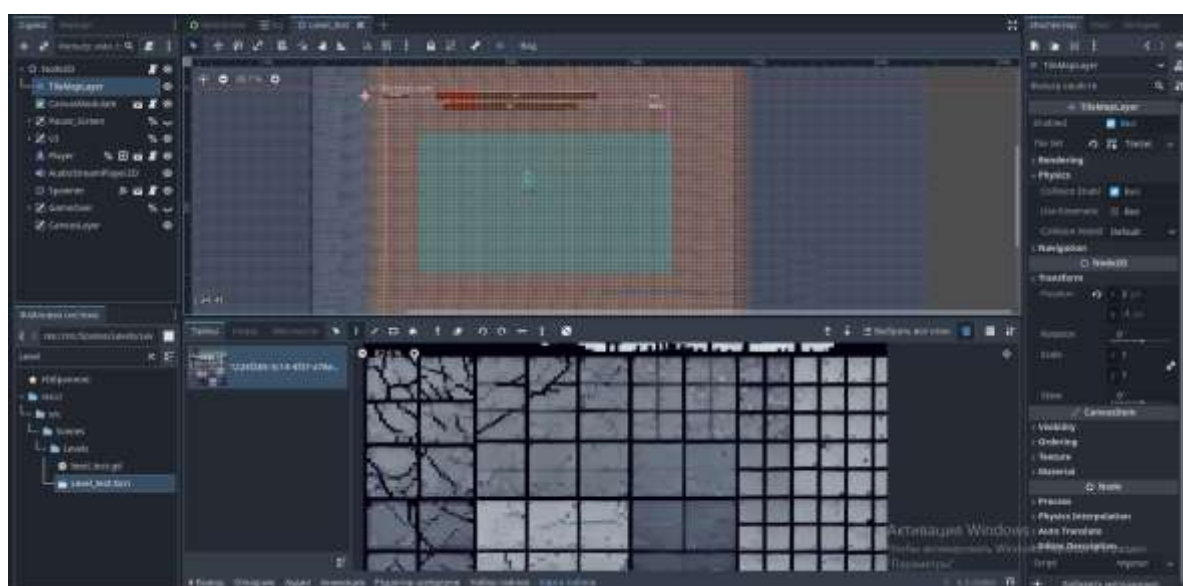


Рисунок 2.4 – Робота з інструментом TileMapLayer

## 2.2 Створення окремих ігрових об'єктів за завданням

Виконавши аналіз встановленої на дипломний проєкт задачі було визначено основні механіки для розробки ігрового додатку. Як правило, будь-яка гра має містити у собі гравця або його аналог, який вимагає жанрова особливості. Роль гравця у проєкті виконує сцена Player.tscn. Детально влаштування сцени гравця можна побачити на рисунку 2.5.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				24
Змн.	Арк.	№ докум.	Підпис	Дата		



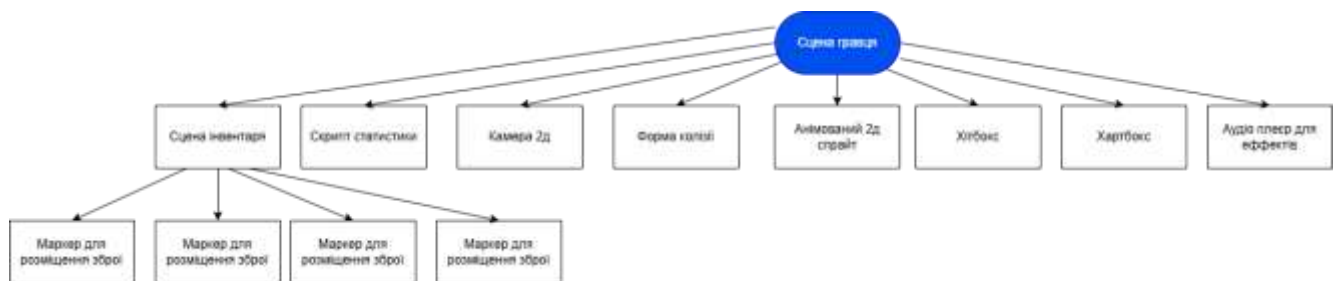


Рисунок 2.5 – UML-діаграма структури сцени гравця

З метою коректного відображення персонажа під час ігрового процесу було використано інструмент AnimatedSprite2D для створення спрайту. Налаштування AnimatedSprite2D продемонстровано на рисунку 2.5.

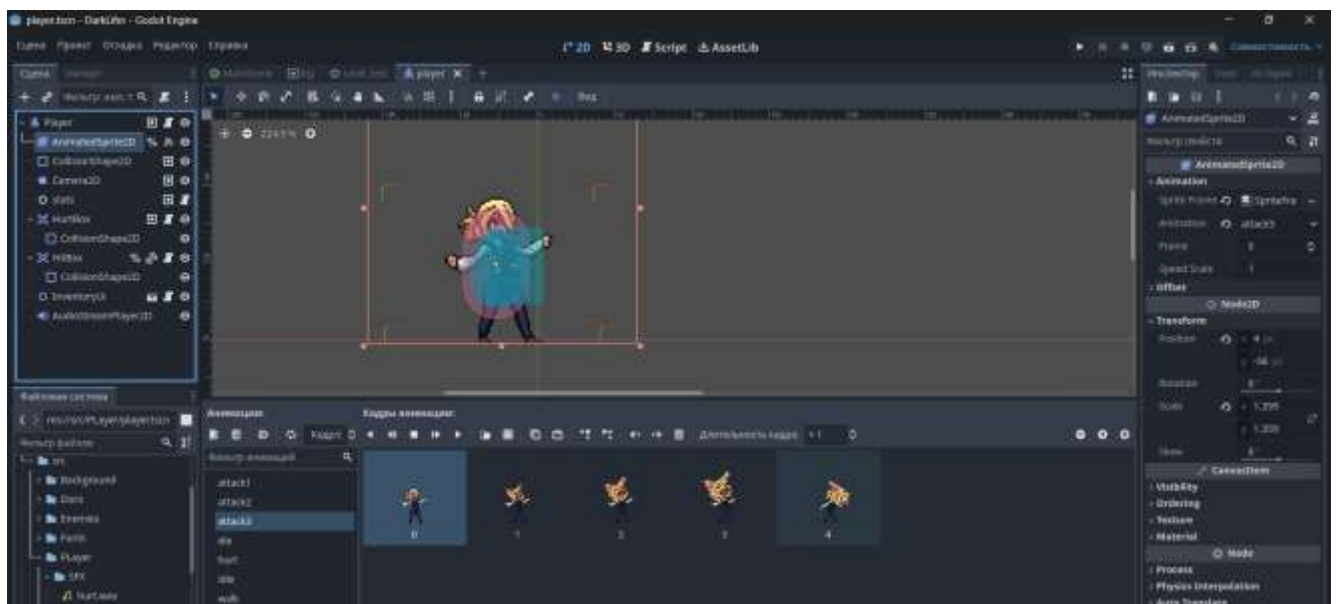


Рисунок 2.5 – Налаштування AnimatedSprite2D

Наступним кроком було розроблено систему колізії гравця за допомогою інструментів Area2D з використанням сигналів `area_entered(area:Area2D)` та `area_exited(area:Area2D)`. Код для роботи із колізією гравця наведено у лістингу 2.1.

### Лістинг 2.1 – Робота із колізією гравця

```

func __enemy_entered_attack_range(area: Area2D) -> void:
    var enemy = area.get_parent()
    if enemy is EnemyBase:
        __is_in_zone = true
        __enemy_in_attack_range = enemy

func __enemy_exited_attack_range(area: Area2D) -> void:
    __enemy_in_attack_range = null
  
```

Вик.	Гуненко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					25
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    __is_in_zone = false
    return

func _physics_process(delta: float) -> void:
    if stats.is_dead:
        return
    __enemy_in_attack_range = null
    for area in hitbox.get_overlapping_areas():
        var parent = area.get_parent()
        if parent is EnemyBase:
            __enemy_in_attack_range = parent
            __is_in_zone = true
            break
    if not __enemy_in_attack_range:
        __is_in_zone = false

    if not __attacking:
        if __enemy_in_attack_range:
            play_effect("res://src/Player/SFX/player_attack.wav")
            perform_attack()
        else:
            movement(delta)

```

Щоб гравець чітко відчував межі свого прогресу — від перших кроків до фінальної хвилі ворогів — у грі необхідна система, яка б уважно стежила за здоров'ям, шкодою та іншими ключовими характеристиками персонажа. Цю функцію виконує скрипт PlayerStats.gd: він інкапсулює всі показники героя, дозволяє динамічно змінювати їх під час бою, а через сигнали миттєво сповіщає інші вузли про втрату життів, здобуття рівня чи застосування бафів. Таким чином скрипт стає єдиною точкою керування прогресією персонажа й ігровою економікою, що забезпечує логічну завершеність та цілісність ігрового процесу. Крім того, він містить механізми збереження / відновлення статистики, що допомагає коректно завантажувати сейви, а функція revive() дає змогу «повернути до життя» героя після поразки, підтримуючи безперервність геймплею. Завдяки взаємодії з GameManager скрипт автоматично підлаштовує базові параметри під рівень складності, тож кожна нова спроба проходження відчувається свіжою й збалансованою.

У «DarkLifer» за розвиток героя відповідає тандем із двох підсистем: PlayerStats та SimplifiedInventory. Скрипт PlayerStats.gd зберігає всі основні показники персонажа — здоров'я, базову й критичну шкоду, броню, швидкість атаки, а також золото й досвід. Під час гри він автоматично підлаштовує ці

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

значення під поточну складність хвилі, а через сигнали негайно повідомляє інші вузли про зменшення здоров'я, отримання рівня або загибель. Ба більше, цей скрипт уміє читати й записувати сейви, накладати тимчасові бафи, нараховувати монети й відроджувати героя без перезавантаження сцени.

Інвентар, реалізований у вузлі InventoryUi (клас SimplifiedInventory), містить чотири активні комірки та один буферний слот. Під час покупки зброї скрипт магазину спершу шукає першу порожню активну клітинку з індексами від 0 до 3; якщо така знаходиться, предмет одразу займає її, оминаючи буфер. Коли всі активні слоти зайняті, система перевіряє, чи є серед них дубль тієї ж назви й тиру. Якщо так, куплена зброя тимчасово вирушає до буфера, після чого функція merge\_weapons об'єднує пару, підвищує tier, перераховує шкоду й ціну та повертає покращений екземпляр у звільнений слот. Якщо ж вільного місця немає і злиття неможливе, гравець бачить повідомлення «FULL INVENTORY». Продаж предмета запускає діалог підтвердження, і після згоди 80 відсотків його вартості автоматично нараховуються через PlayerStats.gain\_coins(). Кольорове маркування кнопок постійно оновлюється згідно з новим tier'ом, а методи merge\_weapons і populate\_inventory\_ui, що викликаються щокадрово, гарантують актуальний вигляд інтерфейсу.

Таким чином, PlayerStats задає «силу» героя, а SimplifiedInventory надає простір для тактичних рішень: гравець або вибирає швидке посилення за рахунок скупчення дублікатів, або ризикує, сподіваючись на вигідне злиття. Разом ці підсистеми формують прозорий, але глибокий цикл прогресії, завдяки якому кожне повторне проходження сприймається свіжим, збалансованим і справді винагороджувальним.

Після завершення кожної хвилі керування передається сцені магазину; логіку цього переходу наочно відображає перша діаграма «Робота магазину», що продемонстрована на рисунку 2.6.

Вона починається з появи самої сцени, одразу ж ініціюється звернення до синглтона WeaponDB, який наперед, під час старту гри, просканував папку зі .tres-ресурсами, зчитав усі властивості кожного файлу-зброї й сформував

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

внутрішню таблицю з полями назва, tier, шкода, ціна, раритет і хвиля доступності. На запит магазину WeaponDB повертає два екземпляри, що статистично відповідають номеру хвилі та випадковому коефіцієнту рідкості; саме ці об'єкти заповнюють слоти купівлі в інтерфейсі.

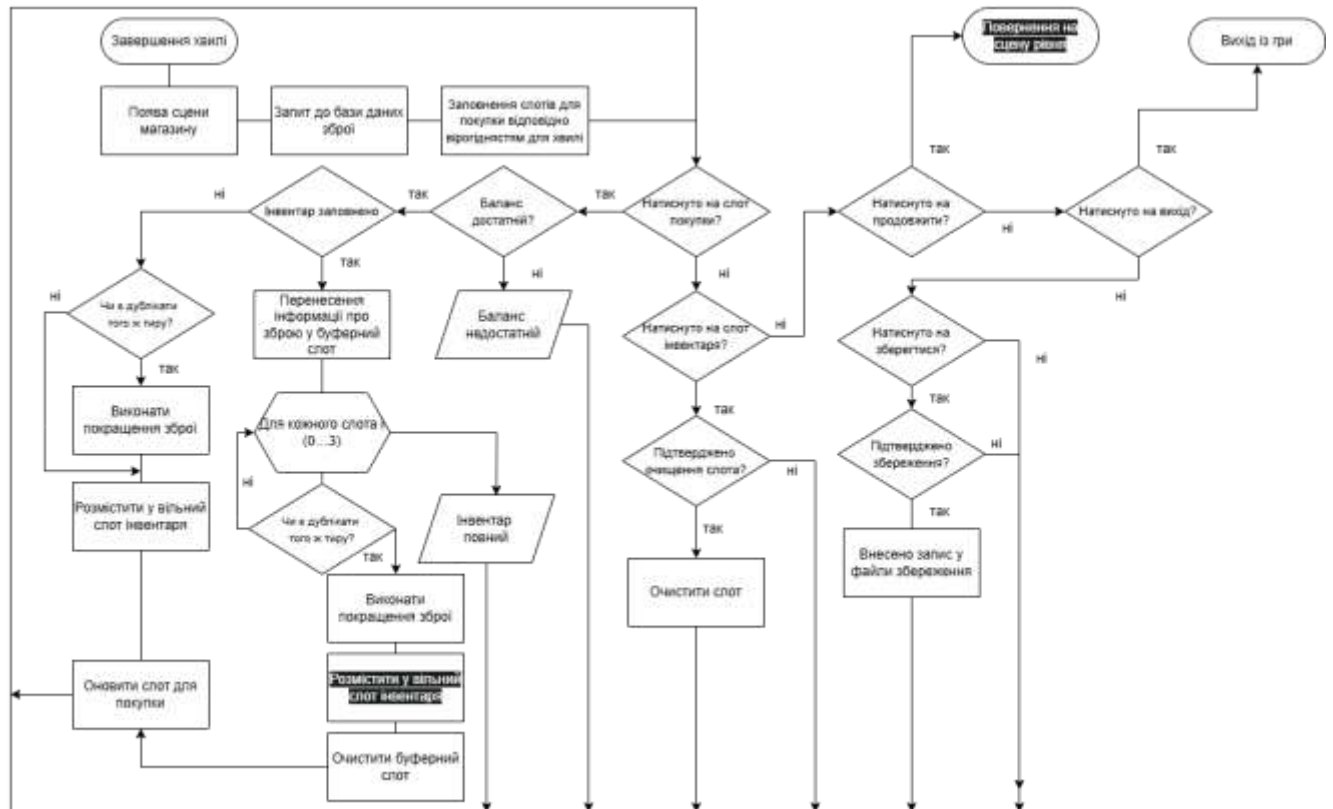


Рисунок 2.6 – Схема роботи магазину зброї

Далі діаграма показує гілкування перевірок: спершу оцінюється, чи вистачає гравцеві монет; якщо так, і користувач натискає на слот купівлі, система без зайвих пауз намагається розмістити придбану зброю-ресурс у першій доступній активній клітинці інвентаря. Буферний слот свідомо пропускається. Коли всі чотири осередки заповнені, алгоритм заглядає до буфера: якщо там або в основних комірках уже є зброя з ідентичною назвою та tier, то спрацьовує блок «Виконати покращення», який підвищує tier на одиницю, перераховує характеристики через формули `calculate_upgraded_damage` і `calculate_upgraded_price`, а буферне місце очищає. Якщо жодна умова не виконана, потік переходить на вузол «FULL INVENTORY» та виводить повідомлення про брак місця. З правого краю тієї самої схеми видно ще дві незалежні траєкторії:

перша реагує на натискання «Повернутися на сцену рівня», друга завершує гру через кнопку виходу або виконує операцію збереження, записуючи у файл JSON поточний склад інвентаря разом із полями PlayerStats.

Друга діаграма «Компонування інтерфейсу» розкладає сцену магазину на вузли Godot. Діаграма продемонстрована на рисунку 2.7.

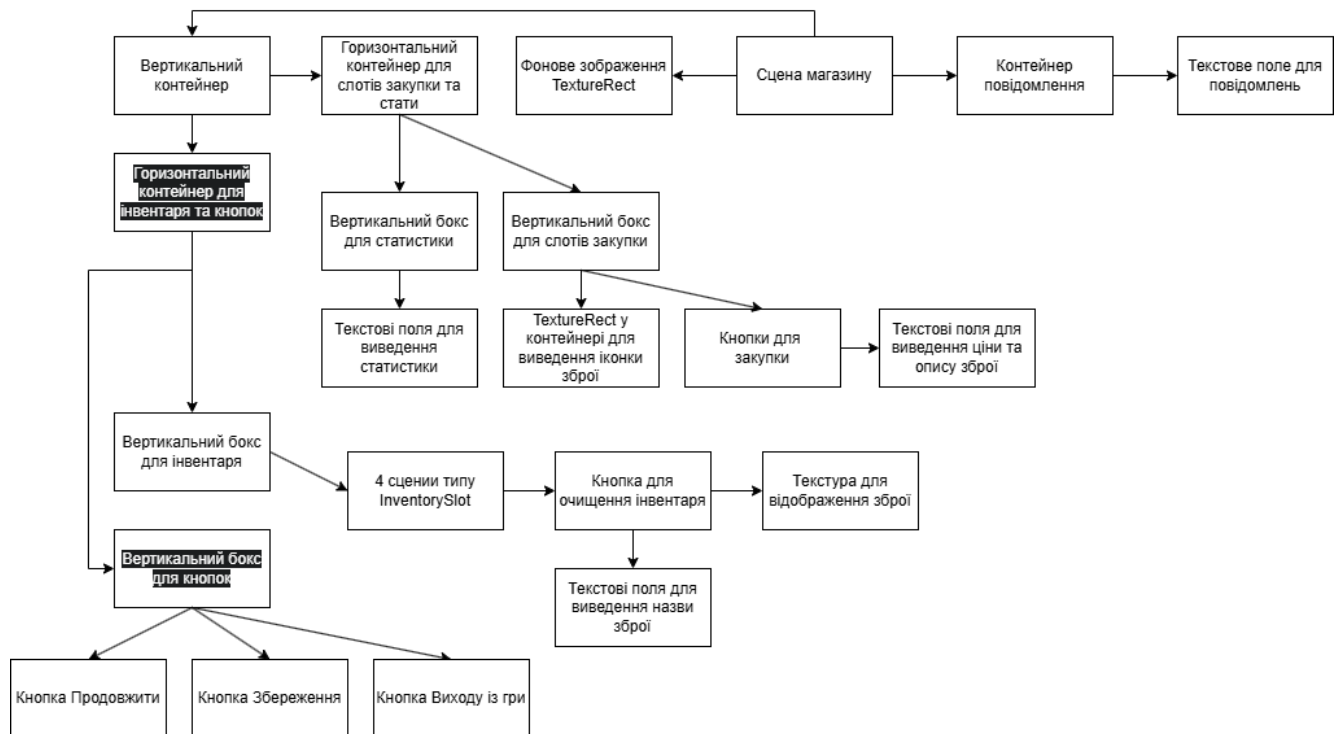


Рисунок 2.7 – Компонування магазину

У її верхній частині позначено головний вертикальний контейнер, що тримає все дерево. Усередині першої горизонтальної секції розташовано блок із двома зонами: ліворуч статистика гравця, праворуч спарені слоти купівлі, кожен із власним TextureRect для іконки, підписами назви, раритету, ціни та коротким описом. Під цією секцією знаходиться вертикальний бокс інвентаря, що містить чотири екземпляри сцени InventorySlot; для кожного передбачено кнопку очищення зі спливним ConfirmationDialog, аби уникнути випадкового продажу. Знизу інтерфейсу винесено горизонтальний контейнер керування, де розміщені кнопки «Продовжити», «Зберегти» та «Вихід із гри». Праворуч від основного каркаса під'єднано окремий контейнер повідомлень, який спливає у разі помилок на кшталт недостатнього балансу або повного інвентаря; його текстове поле отримує рядок через виклик `_show_message`.

Уся інформація про зброю зберігається у вигляді .tres-ресурсів, які лежать у каталозі `res://src/Weapons/Data/`. Кожен із цих файлів є екземпляром класу `WeaponData` та містить читабельний набір полів: назву, опис, базову шкоду, tier, стартову ціну, колір, іконку, посилання на `PackedScene` конкретної зброї, її раритет, тип (`RANGE` або `MELEE`) і набір анімаційних кадрів. Код шаблон для `tres` файлів зброї продемонстровано у лістингу 2.2.

### Лістинг 2.2 – Код шаблон для `tres` файлів зброї

```
extends Resource
class_name WeaponData

@export var weapon_name: String
@export var description: String
@export var damage: float
@export var tier: int
@export var price: float
@export var color: Color
@export var icon: Texture2D
@export var weapon_scene: PackedScene
@export var rarity: String
@export var weapon_type: String
@export var sprite_frames: SpriteFrames
```

Скриншот продемонстрований на рисунку 2.8. із редактора Godot показує, як виглядає запис `Arcane Beacon`: у правій панелі `Inspector` видно, що шкода дорівнює 80, tier — 1, ціна — 200, а раритет позначений як `LEGENDARY`.

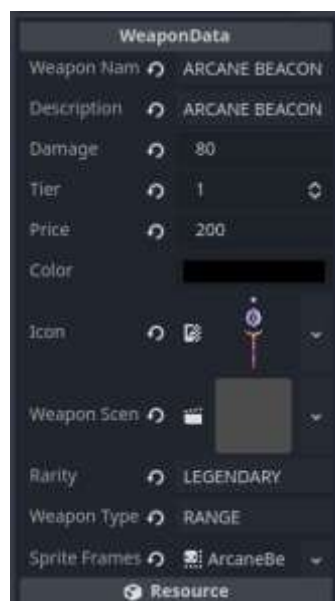


Рисунок 2.8 – Приклад файлу \*.tres для зброї

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

Після старту гри синглтон WeaponsDB рекурсивно сканує папку Data, через ResourceLoader.load зчитує кожен .tres і формує словник, де ключем виступає назва файлу, а значенням — посилання на WeaponData. Коли магазин запитує «зброю для хвили N», база обирає два ресурси відповідної доступності, відразу клонує їх, щоб не змінювати оригінал, та повертає у вигляді масиву. Фрагмент коду синглтона WeaponDB продемонстровано у лістингу 2.3.

### Лістинг 2.3 – Фрагмент коду синглтона WeaponDB

```
func load_all_weapon_data(path: String) -> Array:
    var result = []
    var dir = DirAccess.open(path)
    if dir == null:
        push_error("Не удалось открыть папку: " + path)
        return result
    dir.list_dir_begin()
    var file_name = dir.get_next()
    while file_name != "":
        if file_name.begins_with("."):
            file_name = dir.get_next()
            continue
        var full_path = "%s/%s" % [path, file_name]
        if dir.current_is_dir():
            result += load_all_weapon_data(full_path)
        else:
            var res: Resource = null
            if file_name.ends_with(".remap"):
                var tres_path = full_path.substr(0, full_path.length() -
".remap".length())
                res = ResourceLoader.load(tres_path)
                if not res:
                    push_error("Не удалось загрузить remapped ресурс: " +
tres_path)
            else:
                res = ResourceLoader.load(full_path)
                if res and res is WeaponData:
                    result.append(res)
                elif res:
                    push_warning("Загружен ресурс, но не WeaponData: " +
full_path)
            file_name = dir.get_next()
    dir.list_dir_end()
    return result
```

У самих сценах зброї скрипти MagicStick, Pistol і Sword успадковують базовий клас Weapon. У момент вставляння в інвентар магазин створює

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

PackedScene, інстанціює Node та викликає set\_data(data), передаючи клоновану структуру WeaponData. Метод записує шкоду, анімаційні кадри й одразу запускає тестову анімацію «attack», щоб гравець бачив коректний спрайт. Далі логіка атаки вже різниться: MagicStick після завершення анімації створює MagikHit-проектилю, Pistol породжує bullet-сцену й програв звук пострілу, а Sword просто вмикає хитбокс і, якщо ворог потрапив у зону, наносить контактну шкоду. Усі скрипти узгоджують швидкість відтворення зі значенням \_\_attack\_speed, яке читається зі Stats, тому зброя реагує на бафи гравця автоматично.

Такий підхід дає змогу додати новий вид озброєння без зміни коду магазину: достатньо створити сцену-префаб, зробити для неї WeaponData.tres і покласти файл у ту саму папку. При наступному запуску WeaponsDB підхопить ресурс, магазин зможе його продати, а інвентар — коректно відобразити та, за потреби, змерджити з дублікатом.

Інвентар та арсенал гравця стають гнучкими й легко розширюваними, але цього мало: аби підтримати ескалацію напруги, вороги також повинні зростати разом із силою героя. Тут у гру вступає сцена Spawner. Щойно Level.gd подає сигнал про початок чергової хвилі, Spawner інстанціює задану кількість супротивників і розставляє їх по периметру арени, використовуючи випадкові координати. Для цього він звертається до тієї самої таблиці складності, з якої магазин брав коефіцієнти рідкості зброї, але тепер витягає параметри хвилі — початковий рівень здоров'я, базову шкоду та інтервали появи. Далі на кожного новоствореного ворога накладаються множники hp\_scale і dmg\_scale, що лінійно чи експоненційно зростають разом із номером хвилі; у результаті Skeleton на першому рівні падає від одного влучного пострілу, а на десятому вже витримує серію ударів і сам завдає суттєвих втрат.

Кожен супротивник наслідує EnemyBase, у якому зосереджено пересування до гравця, перевірку дистанції атаки, отримання шкоди та базову анімацію смерті. Конкретні класи, такі як Nightborne або Zhes, успадковують цю поведінку і лише підміняють спрайти, ефекти та тип атаки. Завдяки цьому Spawner може

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		32



створювати будь-яку суміш ворогів однією стрічкою коду, передаючи їм універсальні множники. Основну логіку спавну суперників та вибору типу суперника відповідно хвилі продемонстровано у лістингу 2.4.

#### Лістинг 2.4 – Основна логіка спавну суперників та вибору типу суперника

```
func start_wave():
    GameManager.__current_wave = current_wave
    for enemy in spawned_enemies:
        if is_instance_valid(enemy):
            enemy.queue_free()
    spawned_enemies.clear()
    current_wave += 1
    enemies_to_spawn = int(base_enemy_count * pow(wave_multiplier,
current_wave))
    spawn_timer.start()
    emit_signal("wave_started")

func spawn_enemy():
    if spawned_enemies.size() >= max_enemies:
        return
    var enemy_type = choose_enemy_type()
    var enemy_scene = get_enemy_scene(enemy_type)
    var enemy = enemy_scene.instantiate()
    enemy.position = get_random_point_in_area()
    spawned_enemies.append(enemy)
    get_tree().current_scene.add_child(enemy)
    if enemy.has_signal("died"):
        enemy.connect("died",                                     Callable(self,
"_on_enemy_died").bind(enemy))
    if enemy.has_method("apply_wave_modifiers"):
        enemy.apply_wave_modifiers(current_wave)

func choose_enemy_type() -> String:
    var probabilities = ENEMY_PROBABILITIES[0]["probabilities"]
    for entry in ENEMY_PROBABILITIES:
        if current_wave >= entry["wave"]:
            probabilities = entry["probabilities"]
        else:
            break
    var rand = randf()
    var cumulative = 0.0
    for enemy_type in probabilities.keys():
        cumulative += probabilities[enemy_type]
        if rand <= cumulative:
            return enemy_type
    return "skeleton"

func get_enemy_scene(enemy_type: String) -> PackedScene:
    return ENEMY_SCENES.get(enemy_type, ENEMY_SCENES["skeleton"])
```

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

Уся екосистема виходить збалансованою: магазин поступово пропонує потужнішу зброю, інвентар дозволяє комбінувати її у вищій tier, а вороги щоразу отримують нову «дозу» здоров'я й ударної сили. Так утворюється замкнене коло прогресії, у якому кожен компонент — від WeaponData.tres до EnemyBase — підвищує динаміку й гарантує, що наступна хвиля відчуватиметься помітно складнішою за попередню.

Щоби прогрес — усі здобуті рівні, монети й завершена хвиля — не зникав після виходу, у грі працює вузол-синглтон, що відповідає за збереження та відновлення стану. Під час виклику `save_game()` він формує словник із номером поточної хвилі (за потреби збільшує його, якщо збереження здійснюється «наступною хвилею»), додає серіалізовані дані `PlayerStats` через `to_dict()` та масив інвентаря, отриманий із `get_inventory_data()`, а потім записує JSON у файл `user://save_slot_N.json`. Відповідні множники здоров'я, досвіду й золота беруться з конфіг-файла `settings.cfg`, тому після перезапуску гра відтворює не лише цифри, а й обрану складність. Метод `load_game()` усього одним рядком відкриває файл, парсить JSON і повертає структуру, яку `MainScene` передає назад у `PlayerStats`, `SimplifiedInventory` та `Spawner`; так усі системи синхронно скачують свої значення, і сесія продовжується рівно з того місця, де гравець натиснув «Зберегти». На стартовому екрані сцена `SaveSlotsScene` показує стислий підсумок кожного слота: хвиля, рівень і кількість монет, які метод `get_save_summary()` вибирає без повного завантаження гри. Якщо файл більше не потрібен, `delete_save()` видаляє його через `DirAccess`, залишаючи слот порожнім і готовим до нової пригоди. Код, що відповідає за логіку збереження та завантаження продемонстровано у лістингу 2.5.

#### Лістинг 2.5 – Код, що відповідає за логіку збереження та завантаження

```
func save_game(slot: int, stats: PlayerStats, inventory:
SimplifiedInventory, spawner: Spawner_logic) -> void:
    var wave = spawner.get_current_wave()
    if save_next_wave:
        wave += 1
    var save_data = {
        "wave": wave,
        "player_stats": stats.to_dict(),
        "inventory": inventory.get_inventory_data()
```

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    var path = SAVE_PATH_TEMPLATE % slot
    var file = FileAccess.open(path, FileAccess.WRITE)
    file.store_string(JSON.stringify(save_data, "\t"))
    file.close()
    print("Ігра сохрानена в слот", slot, "на волне",
save_data.wave)

func load_game(slot: int) -> Dictionary:
    var path = SAVE_PATH_TEMPLATE % slot
    if not FileAccess.file_exists(path):
        push_warning("Файл сейва не найден!")
        return {}
    var file = FileAccess.open(path, FileAccess.READ)
    var data = JSON.parse_string(file.get_as_text())
    file.close()
    print("Ігра загружена из слота", slot, "волна:",
data.get("wave", "?"))
    return data

func delete_save(slot: int) -> void:
    var path = SAVE_PATH_TEMPLATE % slot
    if FileAccess.file_exists(path):
        DirAccess.remove_absolute(path)
        print("Сейв слот %d удалён" % slot)

func get_save_summary(slot: int) -> Dictionary:
    var path = SAVE_PATH_TEMPLATE % slot
    if not FileAccess.file_exists(path):
        return {"exists": false}
    var file = FileAccess.open(path, FileAccess.READ)
    var data = JSON.parse_string(file.get_as_text())
    file.close()
    return {
        "exists": true,
        "wave": data.get("wave", 0),
        "level": data.get("player_stats", {}).get("level", 1),
        "coins": data.get("player_stats", {}).get("coins", 0)
    }

```

## 2.3 Опис тестування та відлагодження ігрового додатку

Етап повномасштабного тестування розпочався вже після того, як у проєкті були інтегровані всі ключові підсистеми — інвентар, магазин, генератор ворогів та механізм збереження прогресу. Насамперед було відтворено типовий ігровий процес з обов’язковою перевіркою рухів персонажа у всіх напрямках, коректності розвороту спрайтів і своєчасного оновлення індикатора здоров’я після атак

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

супротивників. Під час проходження рівнів також оцінювалася точність хітбоксів статичних перешкод, реакція рушія на появу і знищення ворогів. Паралельно виконувалося відлагодження інвентаря: підтверджено, що нова зброя автоматично розміщується у першій доступній активній комірці, буфер пропускається, а алгоритм об'єднання дублікатів працює без збоїв. Особливу увагу приділено ситуаціям, коли всі комірки заповнено, а гравець намагається придбати новий предмет — у результаті система або коректно виконує злиття, або виводить повідомлення про брак місця без втрати ресурсу. Після перевірки цих сценаріїв проведено профілювання продуктивності; моніторинг частоти кадрів засвідчив стабільні показники навіть за штучно підвищеного навантаження, коли збільшувалася кількість ворогів і частота їхніх атак.

Далі було верифіковано відповідність рівня гучності звукових ефектів обраним налаштуванням, відсутність затримок при відтворенні аудіо та правильне завантаження спрайтових атласів, що зберігаються у файлах .tres. Виявлені графічні артефакти, спричинені прозорими пікселями, усунено після перезапису відповідних кадрів.

Фінальна перевірка стосувалася системи збереження. Прогрес послідовно фіксувався на різних хвилях, після чого виконувалося повернення до головного меню, зміна рівня складності, а потім завантаження відповідного слота. Було підтверджено, що номер хвилі, статистика персонажа, вміст інвентаря та поточний стан магазину відтворюються без відхилень.

Проведені випробування довели стабільність роботи «DarkLifer» за будь-якої тривалості сеансу.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

### 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ

Після першого входу у ігровий додаток користувачу буде відображено сцену головного меню з трьома кнопками для вибору дії. Для зручності користування при наведенні або взаємодії з кнопку вона виділяється графічно. Інтерфейс головного меню продемонстровано на рисунку 3.1.



Рисунок 3.1 – Інтерфейс головного меню

За бажанням користувача гра надає можливість перейти до налаштувань ігрового додатку. Після переходу на сцену налаштувань користувачу доступні налаштування локалізації, яскравості, гучності ефектів та музики і налаштування спеціальних ускладнень ігрового процесу. У разі бажання зміни локалізації користувачу необхідно натиснути на одну із кнопок блоку налаштування локалізації. Інтерфейс налаштувань для зміни локалізації гри продемонстровано на рисунку 3.2.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

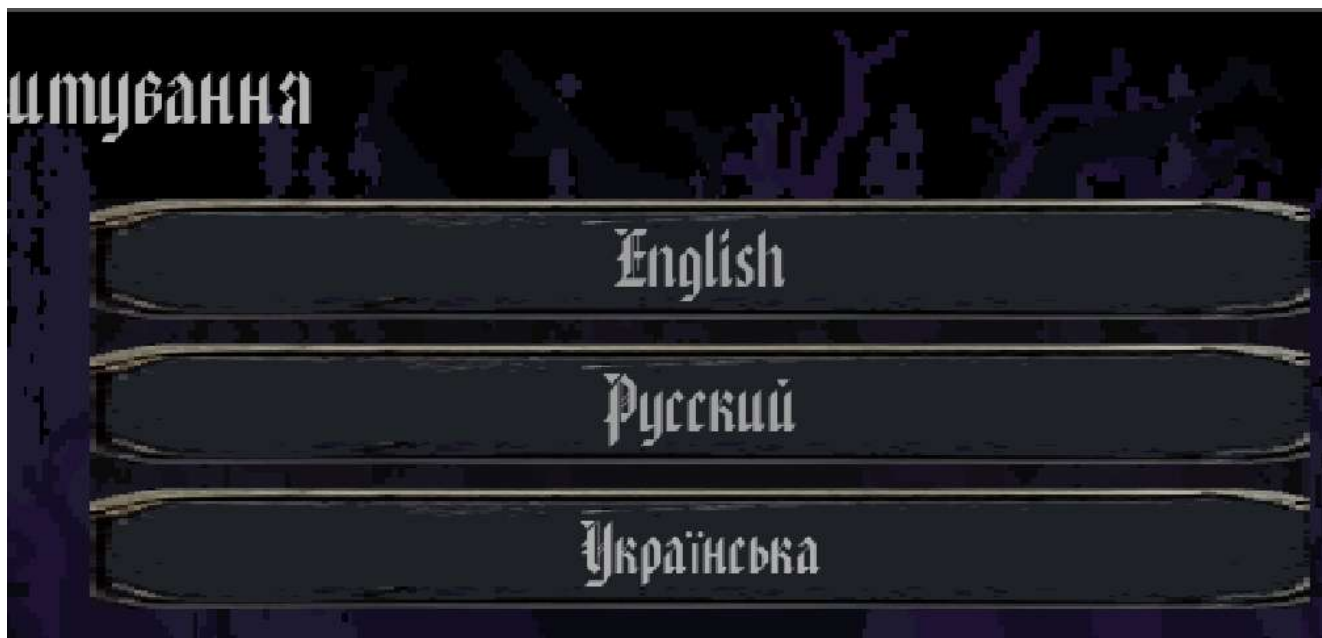


Рисунок 3.2 – Налаштування локалізації гри

За бажання змінити гучність ефектів або музики у інтерфейсі передбачено блок налаштувань роботи з гучністю, що для зручності користувача зроблено у вигляді слайдерів. Інтерфейс налаштувань для роботи з гучністю продемонстровано на рисунку 3.3.



Рисунок 3.3 – Налаштування гучності

Якщо ж необхідно збільшити або зменшити яскравість то можна використати відповідний слайдер. Інтерфейс налаштувань для роботи з гучністю продемонстровано на рисунку 3.4.



Рисунок 3.4 – Налаштування яскравості

Вик.	Гуменко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					38
Змн.	Арк.	№ докум.	Підпис	Дата		

Враховуючи репетитивність основного геймплею ігрового додатку для досвідчених гравців передбачено блок налаштувань з додатковими ускладненнями. Зокрема серед них можна виділити Важкий режим, що зменшує кількість здоров'я гравця надає челендж вже з першої хвили. Також за бажання гравця можна додатково увімкнути зменшення кількості нагороди та досвіду за вбивство суперника. Інтерфейс налаштувань для роботи з додатковими ускладненнями продемонстровано на рисунку 3.5.

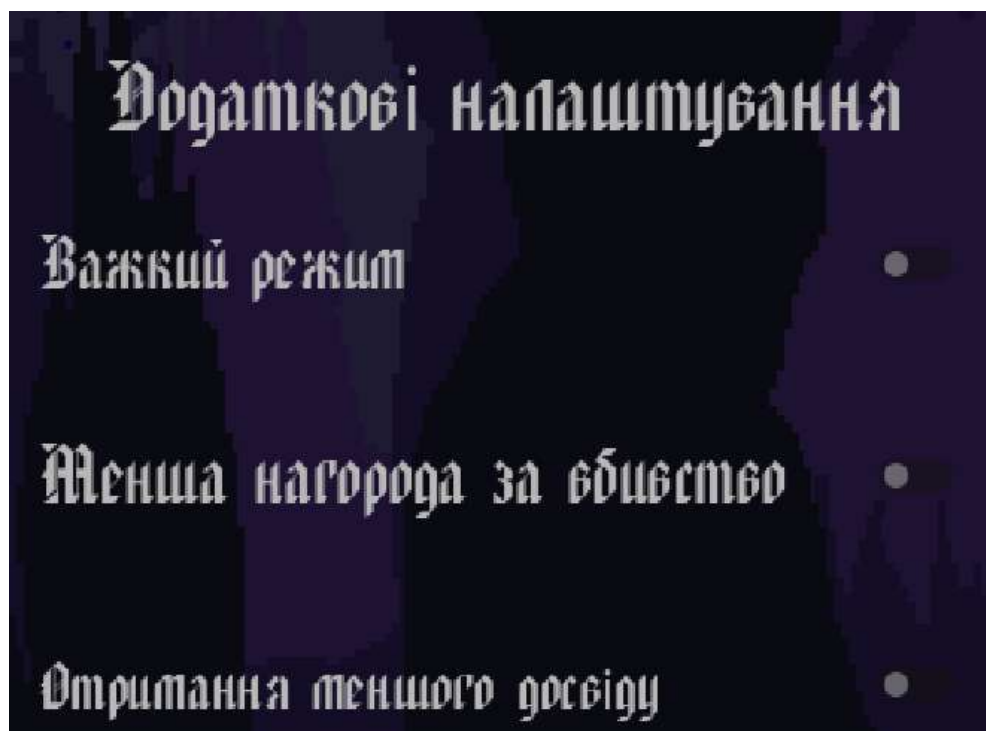


Рисунок 3.5 – Блок налаштувань додаткових ускладнень

Завершивши налаштування користувач може повернутися до головного меню за допомогою кнопки Повернутися у інтерфейсі налаштувань. Натиснувши кнопку грати користувач у разі відсутності збережень одразу буде переміщений безпосередньо до ігрового процесу. У іншому випадку користувач побачить сцену Завантаження ігрового процесу та буде зобов'язаний або розпочати нову гру або обрати слот для завантаження ігрового процесу, при тому порожні слоти будуть заблоковані для взаємодії. Інтерфейс сцени завантаження ігрового процесу продемонстровано на рисунку 3.6.

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

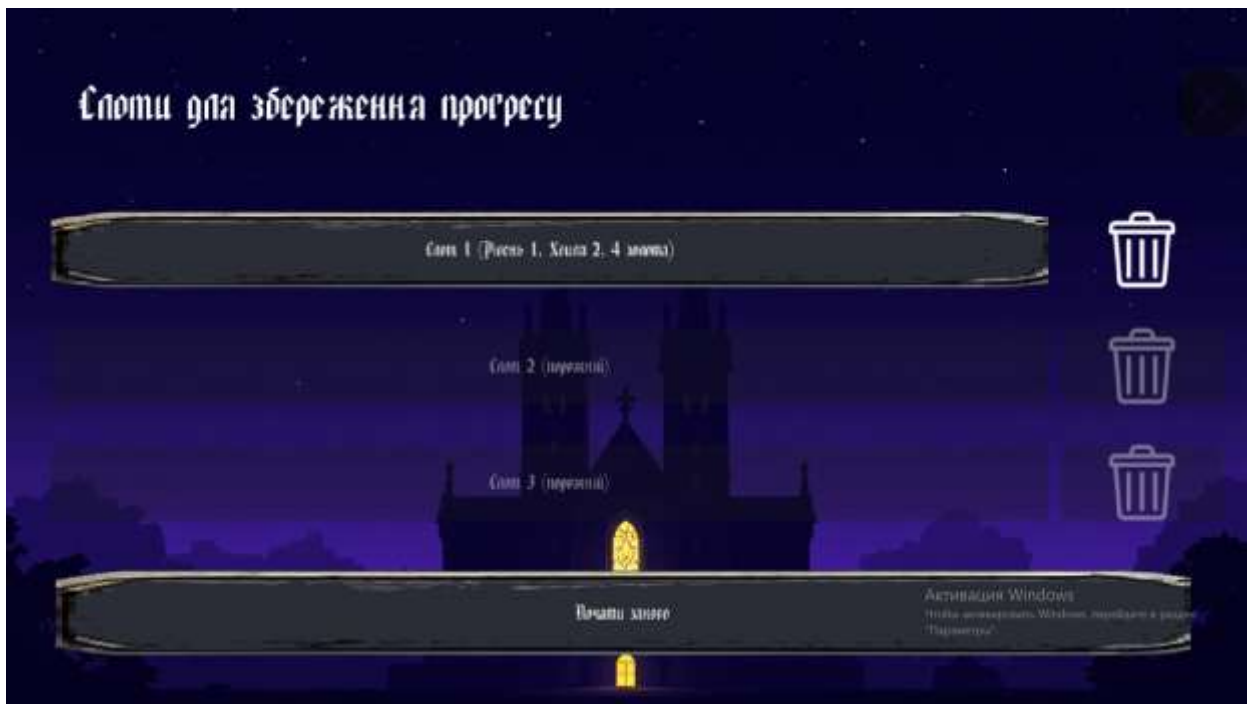


Рисунок 3.6 – Сцена завантаження ігрового процесу

Додатково якщо користувач хоче видалити збереження реалізована піктограма смітника, що викликає очищення слота збереження ігрового процесу. Для того щоб підтвердити видалення викликається діалог підтвердження дії, де користувач може або підтвердити або відмовитись від видалення. Інтерфейс ігрового додатку з підтвердження видалення збереження продемонстровано на рисунку 3.7.

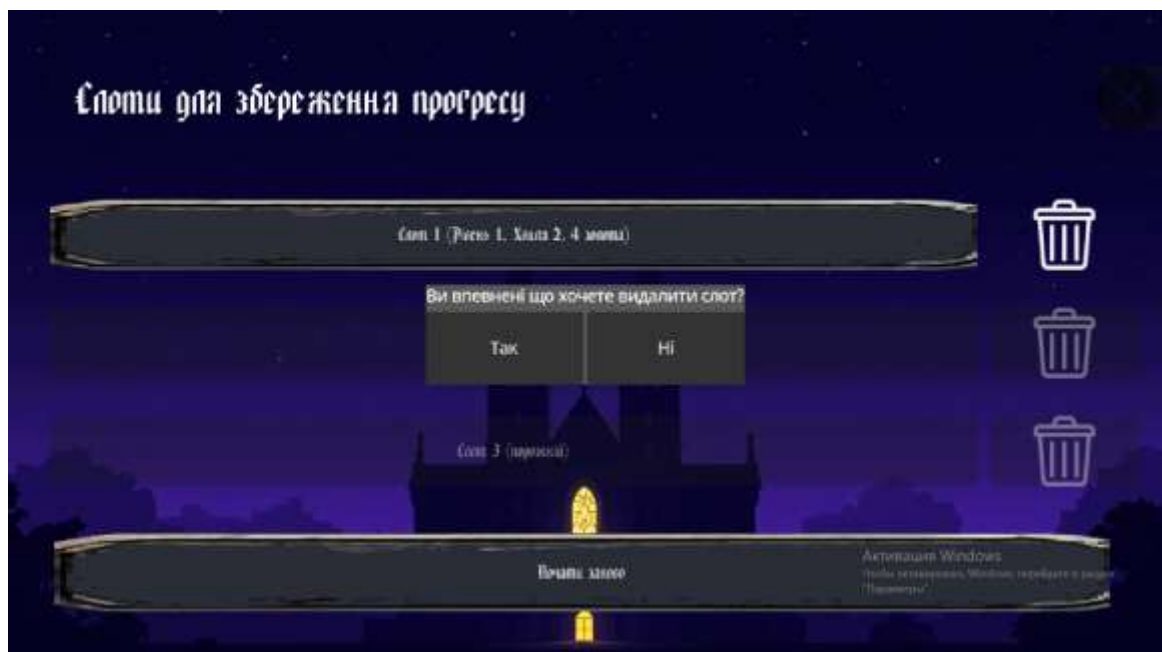


Рисунок 3.7 – Очищення слота збереження ігрового процесу

Вик.	Гуненко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					40
Змн.	Арк.	№ докум.	Підпис	Дата		



Переходимо безпосередньо до ігрового процесу. Потрапивши на ігровий рівень у HUDі гравець бачить кнопку Паузи, що може викликати екран паузи шкалу здоров'я та шкалу досвіду. Додатково для зручності гравця розміщено інформацію про номер данної хвили та поточний баланс. Приклад ігрового процесу розміщено на рисунку 3.7.



Рисунок 3.7 – Вигляд ігрового процесу

У разі накопичення достатньої кількості досвіду підбиранням очок досвіду або після автоматичного підбирання очок після вдалої зачистки хвили користувачу запропоновано обрати одне із трьох покращень на вибір кліком по картці покращення. Вигляд одиниці покращення на рівні продемонстровано на рисунку 3.8. Додатково для зручності користувача при наведенні на кратку вона виділяється графічно.



Рисунок 3.8 – Зовнішній вигляд очок досвіду

Вик.	Гуненко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					41
Змн.	Арк.	№ докум.	Підпис	Дата		

Приклад карток покращення продемонстровано на рисунку 3.9.



Рисунок 3.9 – Вибір картки покращень

Якщо користувач бажає завершити ігрову сесію або зберегтись у грі передбачено екран паузи, що викликається кнопкою пауза у інтерфейсі або ж натисканням клавіші Escape. Інтерфейс екрану паузи продемонстровано на рисунку 3.10.

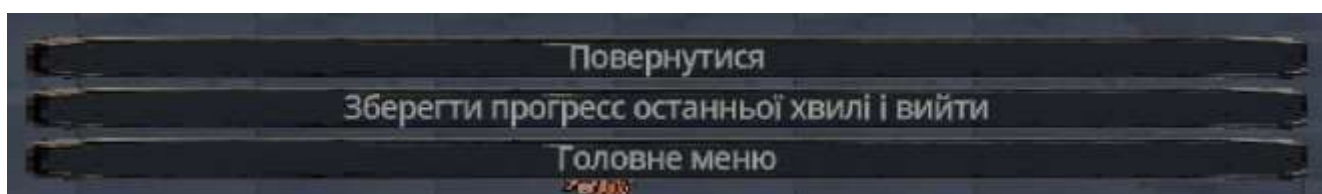


Рисунок 3.10 – Екран паузи

У разі бажання зберегтись користувачу необхідно обрати пункт меню Зберегти прогрес останньої хвили, де після кліку користувач буде перенаправлений на сцену збереження ігрового процесу. У разі кліку на пустий слот збереження буде виконано збереження у слот збереження. У випадку ж кліку по заповненому слоту користувач побачить пропозицію перезаписати вміст слоту, де може як погодитись так і відмовитись від перезапису. Додатково користувач може видалити вміст слотів відео терміналами збереження клікнувши по піктограмі смітника і підтвердивши свою дію у діалозі підтвердження. Якщо ж користувач бажає повернути до екрану паузи то у правому верхньому куті

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

передбачено кнопку виходу з меню збереження. Інтерфейс кнопки повернення продемонстровано на рисунку 3.11.



Рисунок 3.11 – Кнопка повернення в головне меню

Приклад підтвердження перезапису збереження продемонстровано на рисунку 3.12.

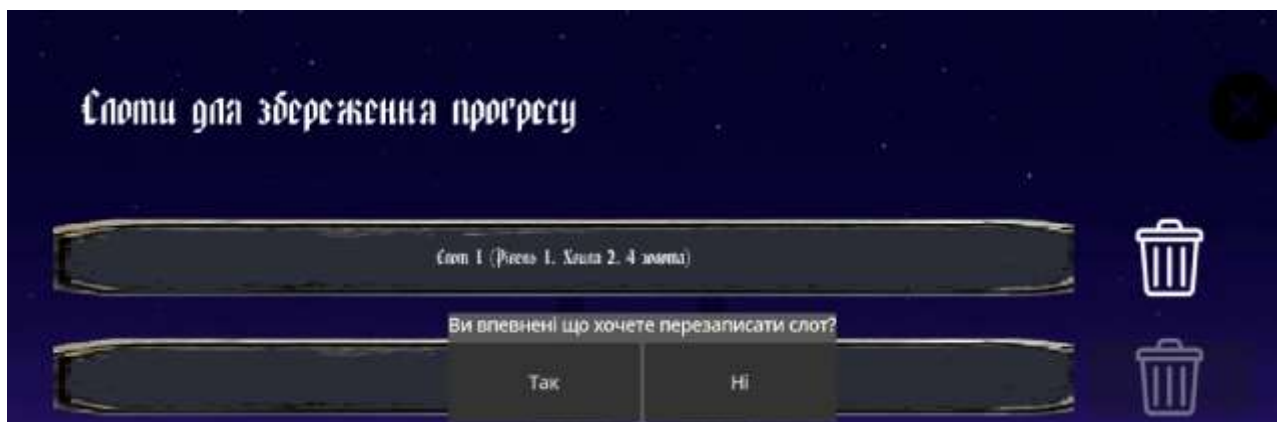


Рисунок 3.12 – Приклад підтвердження перезапису збереження

У разі успішного завершення хвилі користувачу буде показано сцену магазину і запропоновано придбати зброю, зберегтися, продовжити гру або вийти з гри повністю. У разі покупки зброї слот автоматично оновлюється новою зброєю. Якщо ж користувач купує дві однакові зброї то виконується підвищення її рівня. У подальшому кожне покращення зброї виконується автоматичним зливанням двох елементів зброї одного рівня та типу. Графічно покращена зброя виділяється кольором. Якщо ж користувач бажає продати зброю то гра повертає йому 80% її вартості. Приклад вигляду покращеної зброї продемонстровано на рисунку 3.13.

Вик.	Гуменко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					43
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 3.13 – Приклади покращення зброї

Інтерфейс магазину продемонстровано на рисунку 3.14.



Рисунок 3.14 – Інтерфейс магазину

Додатково передбачені перевірки на випадок якщо користувачу забракне коштів або місця у інвентарі. Приклади роботи таких перевірок продемонстровано на рисунках 3.15-3.16.



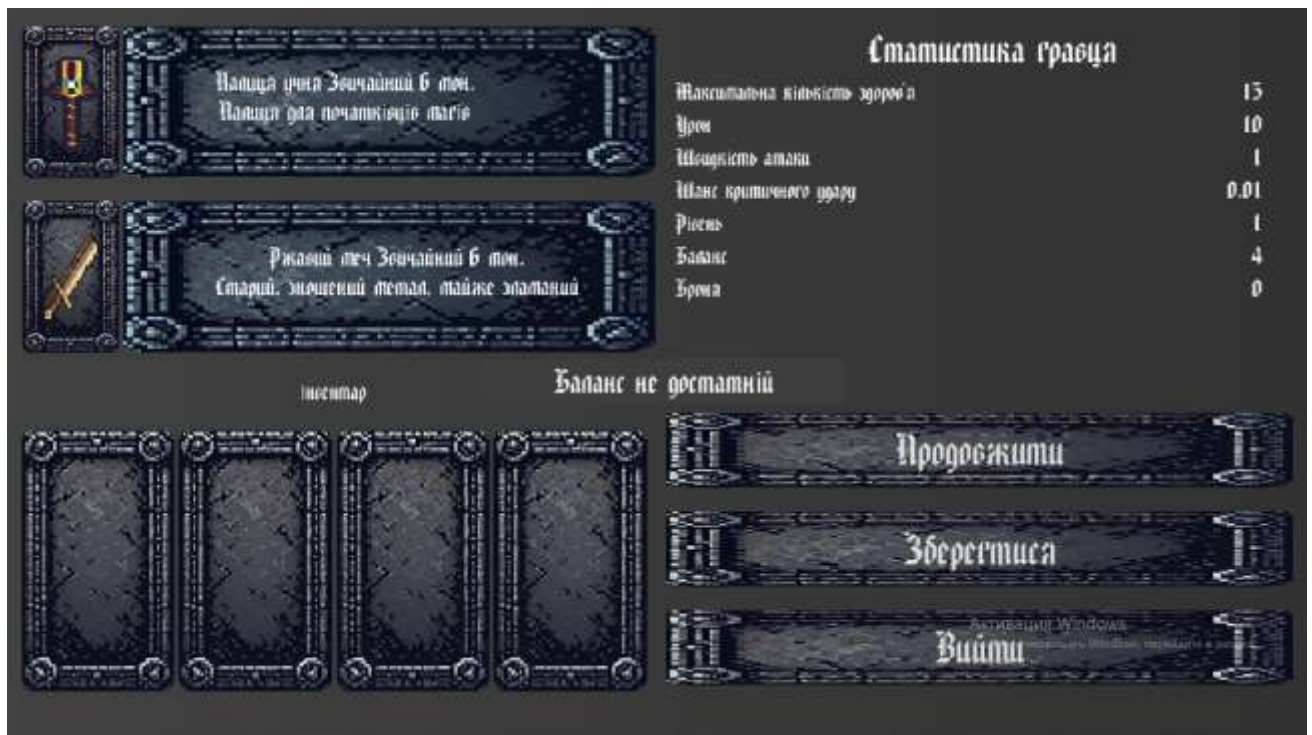


Рисунок 3.15 – Недостатній баланс



Рисунок 3.16 – Заповнений інвентар

У разі завершення хвили невдало гравцю буде запропоновано завантажити збереження або почати спочатку. Інтерфейс екрану поразки продемонстровано на рисунку 3.17.

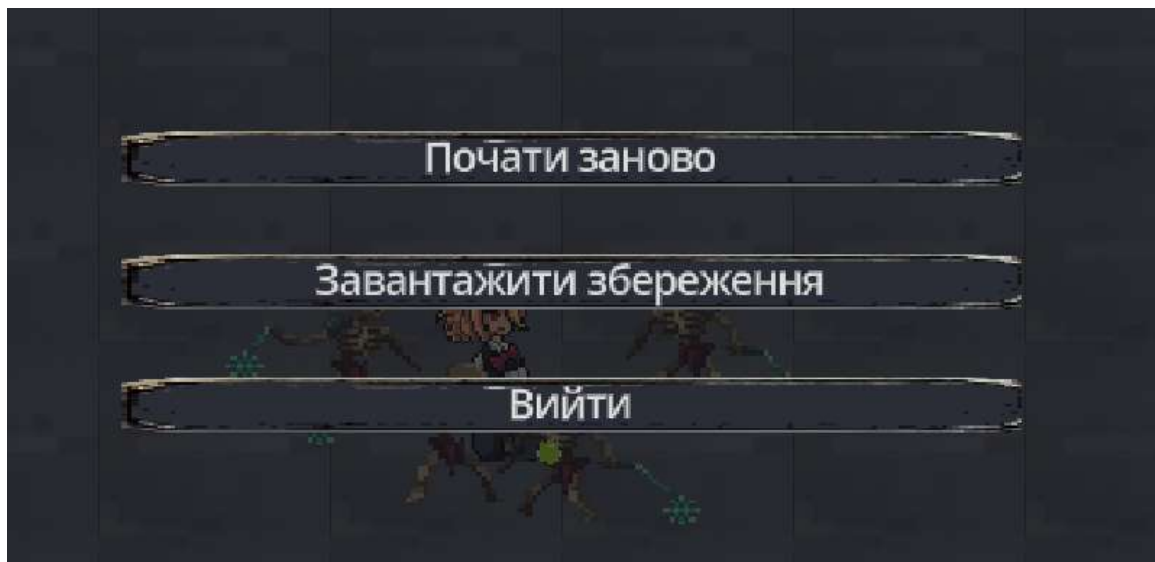


Рисунок 3.17 – Невдале завершення хвили

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

## 4 АНАЛІЗ ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ

Розроблений програмний додаток містить в собі декілька перевірок на різні ситуації використання кінцевим користувачем. Це зроблено для зручності користування та уникнення помилок під час геймплею майбутнього гравця.

Основні перевірки було розміщено у роботі з магазином. Користувач не має мати можливості купити щось при недостатній кількості накопичень. Приклад такого попередження продемонстровано на рисунку 4.1

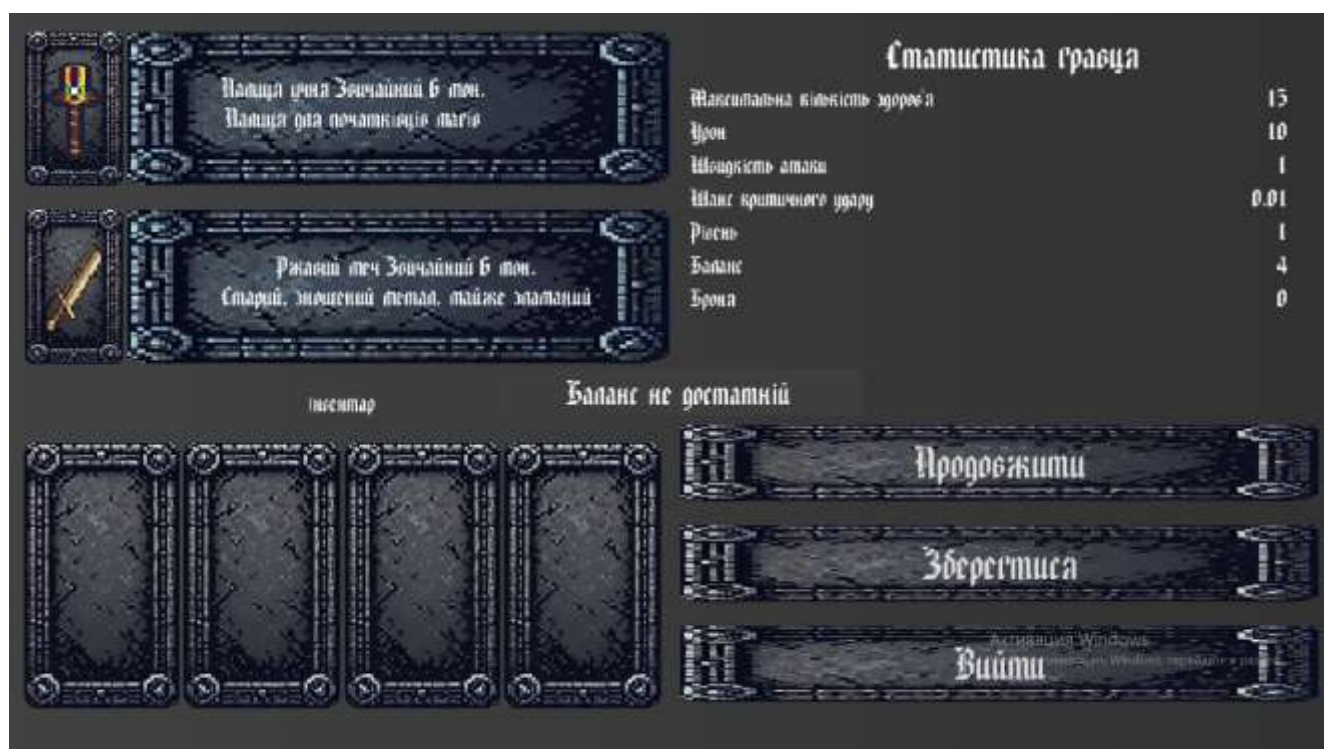


Рисунок 4.1 – Баланс недостатній

Наступним кроком завжди перевіряється чи може користувач розмістити куплену зброю у інвентарі. У разі недостатньої кількості місця користувачу буде видано повідомлення, що інвентар заповнений. Приклад такого повідомлення продемонстровано на рисунку 4.2.





Рисунок 4.2 – Спроба покупки при заповненому інвентарі

Додатково для зручності користувача на кожну дію пов'язану зі збереженням ігрового процесу гра потребує підтвердження своєї дії. Приклади таких підтверджень приведено на рисунках 4.3 – 4.4.

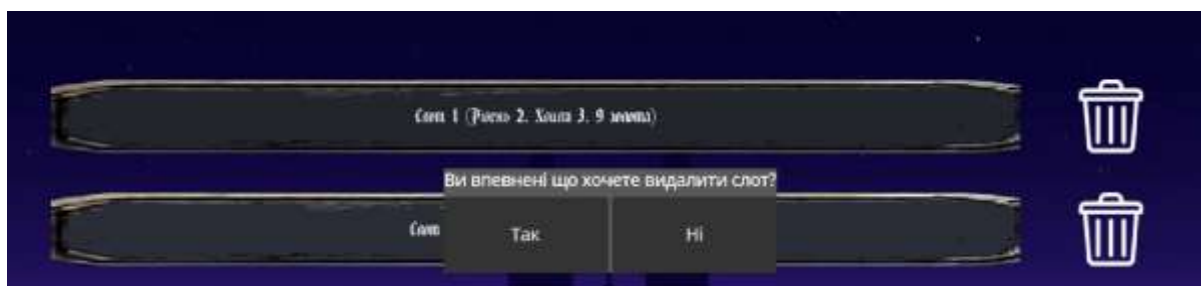


Рисунок 4.3 – Підтвердження видалення слоту збереження ігрового процесу



Рисунок 4.4 – Підтвердження перезапису вмісту слоту для збереження ігрового процесу

Вик.	Гуненко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					48
Змн.	Арк.	№ докум.	Підпис	Дата		

## 5 ОХОРОНА ПРАЦІ

### 5.1 Аналіз небезпечних і шкідливих виробничих чинників

Проектований технологічний процес розробки програмного забезпечення натеper розглядається як окремий різновид виробничого середовища, для якого законодавчо встановлено комплекс норм і вимог безпеки праці. Базовий перелік регламентів включає ДСанПіН 3.3.2.007-98, ДСТУ ISO 9241-5:2004, ДСТУ-Н Б А.3.2-1:2007, ДСТУ 7237:2011, ДБН В.2.5-67:2013, ДСН 3.3.6.037-99 та ДСН 173-96, які встановлюють гранично допустимі рівні небезпечних та шкідливих виробничих чинників, методики їх вимірювання і періодичність контролю. Нижче подано технічне обґрунтування найважливіших параметрів.

Стабільна статична поза оператора ЕОМ кваліфікується як ергономічний ризик II категорії за ДСТУ EN ISO 9241-5:2002.[47] При безперервній роботі, що перевищує 50 хв, фіксується приріст компресійного тиску в міжхребцевих дисках до 1,5–1,8 МПа, що в довгостроковій перспективі корелює зі зростанням частоти остеохондрозу на 18 % [29]. Норматив передбачає регульований робочий стіл із діапазоном висот 650–1250 мм, крісло з механізмом синхропідймання спинки та поперековим упором радіусом 60–90 мм, а також циклічні перерви не менше 10 хв після кожної години інтенсивної роботи з клавіатурою [29; 30].

Зорове навантаження оцінюється за Вимогами щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями, затвердженими наказом Мінсоцполітики України від 14.02.2018 № 207, через показник критичної частоти злиття мерехтіння (CFF). При освітленості підстілля < 300 лк CFF знижується на 7–9 Гц, що свідчить про ранню зорову втому. Стандарт установлює мінімальну освітленість 300 лк для робіт із групи зорової важкості IV-б та 500 лк для IV-а; коефіцієнт пульсації світлового потоку не повинен перевищувати 5 % [35]. Додатково застосовуються РК-панелі із сертифікацією TCO 6.0 та коефіцієнтом контрасту не нижче 1:1000, а кут між нормаллю екрана і лінією погляду фіксується в межах 15–20° для мінімізації відблисків [35].

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Електромагнітні поля наднизької частоти (0,03–3 кГц) регламентуються ДСТУ EN IEC 61000-4-3:2021. У постійному робочому місці допустимий рівень напруженості електричного поля становить 5 В/м, індукції магнітного поля — 250 нТл. Вимірювання виконуються згідно з ДСТУ IEC 61000-4-3 з періодичністю раз на 12 місяців; результати заносять до журналу санітарно-технічного контролю підприємства. Сучасні монітори класу EnergyStar 8.0 мають екрановані джерела живлення та багатошарові фільтри, що знижують НЧ-емісію на 30–40 % від ліміту [34].

Параметри мікроклімату нормуються ДСТУ 7237:2011 і ДБН В.2.5-67:2013. Для категорії робіт «Іа» (витрати енергії до 139 Вт) температура повітря в холодний період року встановлюється 22–25 °С за відносної вологості 40–60 %; швидкість руху повітря не перевищує 0,1 м/с. Розрахунок припливно-витяжної вентиляції виконується за питомим повітрообміном 60 м³/год на одного оператора, що забезпечує концентрацію CO<sub>2</sub> ≤ 1000 ppm у робочій зоні [42]. Температурний дрейф понад ±2 °С та відносна вологість менше 30 % підвищують імовірність астенічних станів на 12–15 % [42].

Акустичний фон у приміщеннях інтенсивної інтелектуальної праці нормується ДСН 3.3.6.037-99. Граничний еквівалентний коригований рівень шуму L<sub>Aeq,T</sub> становить 50 дБА за восьмигодинну зміну. Перевищення нормативу на 3 дБА вимагає використання звукопоглинальних панелей зі зниженням коефіцієнта відбиття до 0,15 у діапазоні 1 кГц [32].

Освітлення проектується згідно з ДСП 173-96 та ДСТУ Б А.3.2-15:2011. Коефіцієнт природної освітленості (КПО) для дисплейних залів має бути ≥ 1,5 %, що забезпечується орієнтацією світлових прорізів у сектор 0–45° північної широти. Штучне світло формують LED-світильники з нейтральною корельованою температурою 4000–4500 К, індексом кольоропередавання Ra ≥ 80 та рівномірністю освітленості E<sub>min</sub>/E<sub>avg</sub> ≥ 0,7. Робочі поверхні дисплеїв розміщують під кутом 90° до світлових потоків для усунення прямих та відбитих відблисків [36].

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

У контексті індустрії розробки ПЗ до класичних гігієнічних факторів додаються фотобіологічні та психосоціальні ризики, які потребують окремого нормативного обґрунтування. Енергетична складова синього спектра дисплеїв (415-455 нм) за методикою ДСТУ ISO 9241-11:2006 не повинна перевищувати  $0,01 \text{ Вт} \cdot \text{м}^{-2} \cdot \text{ср}^{-1}$  [48] при яскравості  $250 \text{ кд/м}^2$ ; польові вимірювання НДІ Медицини праці (2023 р.) засвідчили перевищення показника у 17 % моніторів, тому до виробничих інструкцій включили обов'язкове увімкнення режиму LowBlueLight або накладання фільтрів із зрізом 430 нм, що зменшує спектральне навантаження на 40 % [35]. Тривале сидіння — більш як шість годин за зміну — відповідно до класифікації належить до чинників групи «А» і майже удвічі підвищує ризик розвитку метаболічного синдрому; впроваджений цикл робочих пауз «Помодоро-45/5» та столи з електропідйманням, дозволяють скоротити час статичної пози до чотирьох годин та підвищити варіабельність серцевого ритму на 8 % [30]. Для когнітивної праці критичним лишається приховане інформаційне перевантаження: хоча ДСН 3.3.6.037-99 обмежує еквівалентний шум 50 дБА, регулярне опитування COPSQ-III показало, що при середньому стрес-індексі понад 60 % спостерігається падіння продуктивності на 11 %, тому у корпоративний КРІ керівників включили показник психологічного благополуччя команд [32]. Додаткову небезпеку становлять леткі органічні сполуки меблевих матеріалів: гранична концентрація формальдегіду дорівнює  $0,05 \text{ мг} \cdot \text{м}^{-3}$ , тоді як випробування зразків ЛДСП класу Е1 демонструють  $0,03 \text{ мг} \cdot \text{м}^{-3}$ ; підвищення класу фільтрації припливної вентиляції до ePM1 70 % дало змогу знизити сумарний TVOC на 55 % [30].

## 5.2 Інженерно-технічні заходи з охорони праці

Забезпечення охорони праці у сфері інформаційних технологій ґрунтується на системному поєднанні інженерних, санітарно-гігієнічних і організаційних рішень, які формуються на основі міждержавних стандартів серії ССБТ та чинних державних санітарних норм. Центральним елементом є автоматизовані комплекси

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

моніторингу робочого середовища, що складаються з цифрових датчиків температури, вологості, CO<sub>2</sub>, озону й формальдегіду, блоків збирання даних та програмної платформи для аналізу трендів. Відповідно до ДСТУ EN ISO 12100:2016, такі системи повинні мати три порогові рівні попередження та незалежні канали живлення, аби залишатися працездатними при аварійному відключенні мережі [33]. Параметри мікроклімату деталізуються у ДБН В.2.5-67:2013: температура 22–25 °С у холодний період і 23–28 °С у теплий, відносна вологість 40–60 %, швидкість руху повітря не більш як 0,1 м/с. При цьому об’єм припливного повітря розраховують за формулою  $L = n \cdot G_{CO_2} / (C_{зовн} - C_{гпд})$ . Для типового офісу на 20 робочих місць це становить  $\approx 600$  м<sup>3</sup>/год [42]. Вентиляційні установки комплектують ЕС-двигунами з КПД > 80 % та фільтрами класу ePM1 70 %. Шумове навантаження нормоване ДСН 3.3.6.037-99 із граничним рівнем 50 дБ(А). Практична реалізація включає застосування звукопоглинальних акустичних панелей із щільністю 60 кг/м<sup>3</sup>, демпфувальні підкладки під серверні шафи та безшумні блоки живлення з рівнем шуму < 25 дБ(А) у режимі idle. Комплексна акустична експертиза проводиться вимірювачем рівня звуку 2-го класу точності з октавним аналізом; карти шумових полів зберігаються в електронному журналі технічних оглядів [32]. Для зорової гігієни Вимоги Мінсоцполітики України № 207 встановлюють мінімальну освітленість 300 лк для робіт групи IV-б. Сучасні LED-світильники з коефіцієнтом пульсації  $\leq 1$  % монтують у системах «світлових доріжок», що забезпечують рівномірність освітленості 0,7. Світлотехнічний розрахунок виконується методом коефіцієнтів використання у програмі DIALux; результати підтверджуються натурними вимірами люксометром класу А [35]. Захист від електромагнітних полів (ЕМП) реалізується шляхом використання моніторів зі знаком TCO 8.0, екранізації кабелів у серверних і встановлення загального шина-заземлення. ДСТУ EN IEC 61000-4-3:2021 обмежує напруженість ЕП 5 В/м та індукцію МП 250 нТл у діапазоні 0,03–3 кГц. Контроль здійснюють індукційним зондуванням за методикою ДСТУ IEC 61000-4-3 не рідше ніж раз на 12 місяців, а протоколи вимірювань зберігають протягом п’яти років [31]. У серверних і лабораторіях

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

системи прецизійного кондиціонування підтримують температуру 20–23 °С з точністю  $\pm 1$  °С і вологість  $45 \% \pm 5 \%$ . Розрахунок тепловиділення виконується за питомою потужністю 600–800 Вт/м<sup>2</sup>; при перевищенні 100 кВт установлену водяну систему доповнюють охолоджувальними контурами in-row із високоефективними теплообмінниками [42].

Додатково до базових вимог на підприємствах ІТ-галузі впроваджують систему технічного обслуговування за станом (CBM), що послуговується алгоритмами машинного навчання для прогнозу відмов HVAC- та UPS-обладнання. У межах ДСТУ HD 60364-6:2022 передбачено щоквартальну самодіагностику вузлів із протоколюванням у SCADA [43]. Безперервність електроживлення забезпечують онлайн-UPS подвійного перетворення з автономією  $\geq 10$  хв при 100 % навантаженні, як того вимагає ДНАОП 0.00-1.21-98 щодо резервних джерел живлення [10]. Портативні електроприлади проходять щорічне РАТ-тестування, результати якого зберігають у цифровому архіві безпеки [41]. Для оцінки залишкового ризику використовується матриця SIL/PL (ДСТУ ІЕС 31010:2013) із ранжуванням подій за частотою та тяжкістю наслідків [44]. На робочих місцях встановлюють сенсори «розумного стола», що фіксують температуру поверхні, вологість і тривалість статичної пози та генерують рекомендації працівнику через корпоративний застосунок [39].

### 5.3 Пожежна профілактика

До основних ризиків загоряння належать струмові перевантаження, статичні розряди, перегрів елементів живлення та наявність горючих полімерів у будівельних матеріалах. Для їх нейтралізації НАПБ А.01.001-2014 вимагає встановлення автоматизованих установок пожежної сигналізації (АУПС) адресного типу з чутливістю 0,05 дБ/м та часовою затримкою спрацювання  $\leq 120$  с. АУПС інтегрують із системами оповіщення СОУЕ 4-го типу, що забезпечують мовні повідомлення та світлові показники напрямку евакуації [34]. У серверних з щільністю обладнання  $> 500$  Вт/м<sup>2</sup> застосовують модульні газові установки на

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

основі FK-5-1-12 з робочим тиском 25 бар. Система проектується у відповідності до ДСТУ EN 15004-1:2015 з розрахунковою концентрацією 5,6 % об. для часу витримки 10 хв [37]. Порошкові засоби (тип ВСЕ) залишаються резервними й розміщуються поза зонами з високою концентрацією електроніки. Евакуаційні виходи повинні мати двері з опором вогню не нижче EI-30, відчинятися у напрямку виходу й оснащуватися пристроями Anti-Panique. Ширина основного коридору — не менше 1,2 м, мінімальна освітленість аварійного режиму 10 лк при тривалості автономної роботи світильників 1 год. Плани евакуації графічно виконують за ДСТУ ISO 23601:2019 у масштабі 1:200 та дублюють на рівні 1,5 м від підлоги. Перевірка електромереж виконується згідно з ДНАОП 0.00-1.21-98: опір ізоляції групових ліній  $\geq 0,5 \text{ М}\Omega$ , петля «фаза-нуль» —  $\leq 0,8 \text{ }\Omega$ , час спрацьовування ПЗВ при диференційному струмі 30 мА —  $\leq 0,06 \text{ с}$ . Протоколи вимірювань зберігаються в журналі енергонагляду не менше 3 років [38]. Щоквартально проводять пожежно-технічні інструктажі з практичним відпрацюванням роботи вогнегасників. Типові вогнегасники, рекомендовані для ІТ-офісів: вуглекислотні ВВК-2(з)-02 та аерозольні ВР-И-3 з класом гасіння Е 5 кВ. Всі покривні та облицювальні матеріали в робочих зонах повинні належати не нижче групи Г1 та В1 за ДСН 173-96; для підвісних стель у серверних допускається застосування негорючих плит групи НГ з питомою теплоотою згоряння  $\leq 3 \text{ МДж/кг}$  [36].

У залах із пожежонавантаженням понад 600 МДж монтують систему димовидалення з аеродинамічними клапанами, які тестуються щомісяця за ДСТУ EN 12101-8:2014 [45]. Кабельні лінії силові й СКС прокладають у вогнестійких коробах категорії Е90 відповідно до ДСТУ EN 1366-3:2021, що гарантує працездатність протягом 90 хв під дією полум'я [45]. Для серверних площ  $> 30 \text{ м}^2$  формують подвійну зону інертного газу FK-5-1-12; резервуари обладнують датчиками тиску і поданням даних у BMS-систему, регламент перезавантаження — 1 раз на 10 років (ДСТУ EN 15004-1:2015) [45]. Раз на рік проводять навчальні евакуаційні тренування із залученням добровільних пожежних дружин; норматив повного виходу персоналу має становити  $\leq 2 \text{ хв}$  (НАПБ А.01.001-2014) [34]. Після

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				54
Змн.	Арк.	№ докум.	Підпис	Дата		



тренування складають протоколи, аналізують «вузькі» місця та коригують маршрути [34].

#### 5.4 Заходи з ергономіки

Ергономічні параметри робочого місця в ІТ — це комплекс людино-машинних параметрів, що зменшують хронічні професійні ризики та підвищують стійку продуктивність. ДСТУ EN ISO 9241-5:2004 регламентує геометрію меблів: висота стільниці 720 мм (з регулюванням  $\pm 200$  мм), глибина 800–900 мм, ширина не менше 1200 мм. Крісло повинно мати механізм синхропідймання спинки, регулювання сидіння по висоті 400–550 мм і поперековий упор, що висувається на 60–90 мм [30]. Відстань від очей до екрана 500–700 мм, верхній край екрана — на рівні лінії погляду або на 50 мм нижче. Приклад правильного розміщення оператора ПК продемонстровано на рисунку 1.1.

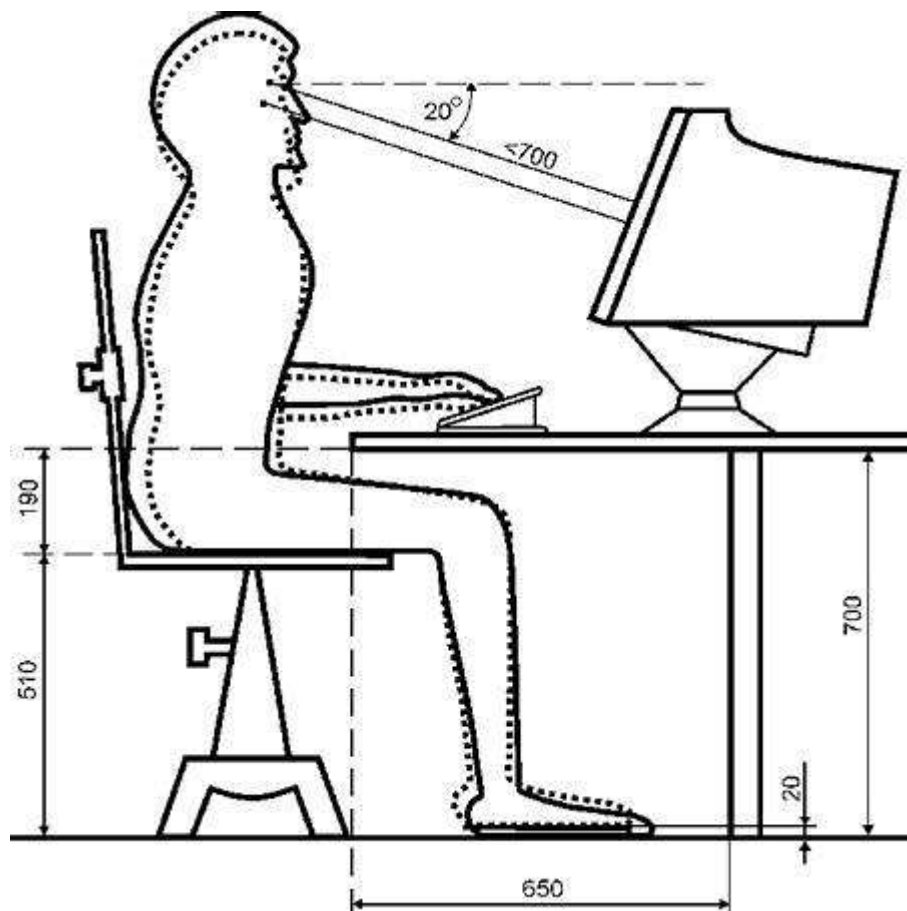


Рисунок 5.1 – Оптимальне розміщення оператора ПК

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Вимоги Мінсоцполітики України № 207 визначають освітленість 300–500 лк для груп зорових робіт IV-а/IV-б. Люмінесцентні лампи старих поколінь допускаються лише з електронними ПРА; перевага надається LED-модулям із CRI  $\geq 80$  та CCT 4000–4500 К [35]. Для запобігання кумулятивним травмам верхніх кінцівок застосовуються клавіатури з роздільною геометрією клавіш та вертикальні миші з кутом захвату 57–90°. Дослідження Інституту медицини праці НАМН України свідчать про зниження показника дискомфорту за шкалою RULA з 5/7 до 3/7 при переході на такі маніпулятори ( $n = 54$ ,  $p < 0,05$ ). Організація перерв: за методикою «60/10» кожні 60 хв роботи робляться 10-хв фізіологічні паузи; програмні нагадувачі інтегрують із системою управління персоналом (HRIS), що фіксує дотримання режиму й генерує рекомендації щодо гімнастики для очей. Площа робочої зони не менше 4,5 м<sup>2</sup> на особу та ширина міжпрохідної смуги не менш як 1,5 м наведені у ДСН 3.3.6.042-99; при проектуванні відкритих просторів застосовують модуль 1500 × 1500 мм, що забезпечує достатню приватність при відборі акустичних перегородок висотою 1200–1400 мм [39]. Психофізіологічний комфорт підтримується кольорокорекцією робочої зони, наявністю зелених насаджень, зонами мікровідпочинку з рівнем шуму до 40 дБ(А) і нейтральною температурою кольору. Комплексна оцінка ергономічної відповідності здійснюється методикою REBA двічі на рік; при сумарному балі  $> 7$  впроваджують коригувальні заходи.

Сучасні рекомендації з профілактики зорової втоми передбачають динамічну зміну спектрального складу освітлення відповідно до циркадних ритмів (Human-Centric Lighting). Світильники із підтримкою DALI-2 програмують за ДСН 173-96 — підвищення CCT до 5000 К о 10-й годині та зниження до 3500 К після 16-ї, що, за даними НУ «ЛП» (польовий експеримент, 2024), зменшує сонливість на 12 % [36]. Столи із електричним регулюванням (діапазон 650-1250 мм) відповідають вимогам і дають змогу реалізувати цикл «sit-stand-move», скоротивши статичне сидіння до 4 год/зміну [30]. Для акустичного комфорту використовують мобільні ширми класу NRC 0,8 (ДСН 3.3.6.037-99) [32], а під час відеоконференцій — навушники з активним шумозаглушенням.

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Монітори з датчиками освітленості автоматично знижують яскравість до 120 кд/м<sup>2</sup> у темних умовах, що узгоджується з межами, установленими ДСанПіН 3.3.2.007-98 [35]. У Scrum-командах після кожного спринту (90-120 хв) проводять 3-хв дихальні вправи; факт виконання фіксує трекер Well-being, інтегрований у HRIS, і враховується у KPI тім-лідів[1].

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

## 6 ЕКОНОМІЧНА ЧАСТИНА

Згідно із завданням дипломного проекту необхідно визначити вартість робіт і відпускну ціну програмного продукту для замовника. Для виконання розрахунку були використані початкові дані, представлені в таблиці 6.1.

Таблиця 6.1 – Початкові дані для розрахунку

Найменування початкових даних	Показник	Джерело отримання
Трудомісткість складання програмного продукту	нормо/год.	Фактичні витрати часу на розробку програми
Місячна ставка програміста	20550 грн.	Дані переддипломної практики
Кількість годин в місяці	160	Кількість робочих днів – 20
Додаткова зарплата (20-50 %)	20	Дані переддипломної практики
Нарахування на соціальне страхування (ЄСВ)	22%	Чинне законодавство
Загальновиробничі витрати (180-200%)	200%	Дані переддипломної практики
ПДВ (податок на додану вартість)	20%	Чинне законодавство

### 6.1 Розрахунок повної вартості робіт розробки програмного продукту

За видами витрати підприємства класифікуються за економічними елементами та за статтями калькуляції.

Під економічними елементами витрат слід розуміти сукупність економічно однорідних витрат в грошовому виразі за їх видами (це групування дає змогу відповісти на питання, що витрачено на даний об'єкт).

Статті калькуляції вказують як формуються ці витрати для визначення собівартості продукції (вартості робіт) - одні витрати показуються за їх видами (елементами), інші - за комплексними статтями (включають декілька елементів).

Вик.	Гуменко Я.М.				ДП.ПЗ.211.06.ПЗ	Арк.
Пер.	Ланська С.С.					
Змн.	Арк.	№ докум.	Підпис	Дата		58

При цьому один елемент витрат може бути присутній у кількох статтях калькуляції.

#### Стаття 1. Сировина та матеріали.

До основних матеріалів відносять металопрокат, півфабрикати, тобто ті матеріали, які входять до складу виробу. Студент повинен вказати основний матеріал, що використовується при виготовленні продукції відповідно отриманого завдання.

Матеріали – не використовуються.

#### Стаття 2. Комплектуючі вироби

$$B_k = \sum C_k \times n_k, \quad (6.1)$$

де  $B_k$  — витрати на комплектуючі вироби, грн.;

$C_k$  — ціна за 1 одиницю комплектуючих виробів, грн.;

$n_k$  — кількість комплектуючих виробів по кожному типорозмірі, шт.

Витрати на матеріали й комплектуючі вироби в планову виробничу собівартість не входять, а оплачуються замовником додатково при заміні окремих комплектуючих виробів (як виняток).

Комплектуючі вироби – відсутні.

#### Стаття 3. Основна заробітна плата

$$ЗП_{\text{осн}} = l_{\text{год}} \times T_{\text{год}}, \quad (6.2)$$

де  $l_{\text{год}}$  — годинна тарифна ставка програміста, грн.;

$T_{\text{год}}$  — кількість годин у місяці, приймається 160 – вихідні дані.

Визначаємо годинну тарифну ставку програміста:

$$l_{\text{год}} = \frac{L_{\text{міс}}}{T_{\text{год}}}, \quad (6.3)$$

де  $L_{\text{міс}}$  – місячна ставка програміста, грн.

$$l_{\text{год}} = \frac{20550}{160} = 128,44 \text{ грн}$$

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

#### Стаття 4. Додаткова заробітна плата

$$ЗП_{\text{дод}} = \frac{З_{\text{осн}} \times Д\%}{100}, \text{грн.} \quad (6.4)$$

де  $ЗП_{\text{дод}}$  – додаткова заробітна плата, грн.;

$Д\%$  – відсоток додаткової заробітної плати, приймається 20% – вихідні дані.

$$ЗП_{\text{дод}} = \frac{20550 \times 20}{100} = \frac{411000}{100} = 4110 \text{ грн.}$$

#### Стаття 5. Нарахування на соціальне страхування

$$В_{\text{есв}} = \frac{(ЗП_{\text{осн}} + ЗП_{\text{дод}}) \times \text{ЕСВ}\%}{100}, \quad (6.5)$$

де  $В_{\text{есв}}$  – нарахування на соціальне страхування, грн.

ЕСВ % – відсоток нарахувань на соціальне страхування, приймаємо 22% – вихідні дані.

$$В_{\text{есв}} = \frac{(20550 + 4110) \times 22}{100} = 5425,20 \text{ грн}$$

#### Стаття 6. Загальновиробничі витрати

$$В_{\text{заг}} = \frac{ЗП_{\text{осн}} \times Н_1\%}{100}, \quad (6.6)$$

де  $В_{\text{заг}}$  – загальновиробничі витрати, грн.;

$Н_1\%$  – відсоток загальновиробничих витрат, приймається 200% – вихідні дані.

$$В_{\text{заг}} = \frac{20550 \times 200}{100} = 41100 \text{ грн}$$

Виробнича собівартість розраховується як добуток всіх статей калькуляції (сума ст.1- 6).

Виробнича собівартість:

$$В_{\text{соб}} = В_{\text{к}} + ЗП_{\text{осн}} + ЗП_{\text{дод}} + В_{\text{есв}} + В_{\text{заг}}, \text{грн.} \quad (6.7)$$

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

$$B_{\text{соб}} = 0 + 20550 + 4110 + 5425,20 + 41100 = 71185,20 \text{ грн}$$

Прибуток підприємства (фірми)

$$П_n = B_{\text{соб}} \times \frac{П_n\%}{100}, \text{ грн.} \quad (6.8)$$

де  $П_n$  – прибуток підприємства, грн.;

$П_n\%$  – відсоток прибутку підприємства, приймається 25 % – вихідні дані.

$$П_n = 71185,20 \times \frac{25}{100} = 17796,30 \text{ грн.}$$

Оптова ціна підприємства (фірми)

$$Ц_n = B_{\text{соб}} + П_n, \quad (6.9)$$

де  $Ц_n$  – ціна підприємства, грн.

$$Ц_n = 71185,20 + 17796,30 = 88981,50 \text{ грн.}$$

Податок на додану вартість

$$ПДВ = Ц_n \times \frac{ПДВ\%}{100}, \text{ грн.} \quad (6.10)$$

де  $ПДВ\%$  – відсоток податку на додану вартість, приймається 20% (діюча ставка  $ПДВ$  на сучасний момент).

$$ПДВ = 88981,50 \times \frac{20}{100} = 17796,30 \text{ грн.}$$

Ціна для замовника (відпускна підприємства)

$$Ц_{\text{зам}} = Ц_n + ПДВ, \text{ грн.} \quad (6.11)$$

$$Ц_{\text{зам}} = 88981,50 + 17796,30 = 106777,80 \text{ грн.}$$

Планова калькуляція вартості і ціни робіт розробки наведено в таблиці 6.2.

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				61
Змн.	Арк.	№ докум.	Підпис	Дата		



Таблиця 6.2 – Планова калькуляція вартості і ціни робіт розробки програмного продукту

№	Стаття калькуляції	Позначення	Одиниця виміру	Значення
1	Матеріали	Вм	грн.	0
2	Комплектуючі вироби	Вк	грн.	0
3	Основна заробітна плата	ЗПосн.	грн.	20550
4	Додаткова заробітна плата	ЗПдод.	грн.	4110
5	Відрахування на соціальне страхування	Весв	грн.	5425,20
6	Загальновиробничі витрати	Взаг.	грн.	41100
Виробнича собівартість (сума ст. 1-6)		Всоб	грн.	71185,20
Прибуток підприємства		Пп	грн.	17796,30
Оптова ціна підприємства (фірми)		Цп	грн.	88981,50
Податок на додану вартість		ПДВ	%	17796,30
Ціна для замовника (відпускна ціна)		Цвід	грн.	106777,80

Висновок: таким чином розрахунок показав, що вартість робіт і ціна програмного продукту DarkLifer складає 71185,20 грн., якщо розробник продаватиме цю програму, то її ціна для споживача складе 106777,80 грн. При цьому з кожного екземпляра проданої програми розробник матиме прибуток 17796,30 грн.

## ВИСНОВКИ

У процесі роботи над дипломом було створено цілісну комп'ютерну гру у стилі action-roguelike, зібрану на Godot Engine 4.3. Завдання полягало в тому, щоб поєднати стрімкий геймплей із механікою виживання, процедурним генеруванням контенту та поступовим посиленням героя. Проєкт орієнтовано на Windows-платформу; він використовує 2D-графіку, має багатомовний інтерфейс і відповідає технічним стандартам сучасних інді-релізів.

Під час аналізу найуспішніших представників піджанру bullet heaven (Vampire Survivors, Brotato, 20 Minutes Till Dawn тощо) стали очевидними кілька трендів: автоматизований бій, нетривалі сесії, стрімке зростання потужності персонажа та висока реіграбельність. Ці принципи отримали власну інтерпретацію: запроваджено економіку, що розвивається поетапно, систему покращень на основі випадкових карт і внутрішньоігровий магазин, асортимент якого оновлюється залежно від прогресу.

Головним ядром геймплею став цикл «хвиля — пауза — покращення», де складність ворогів пропорційно збільшується, змушуючи гравця тактично комбінувати зброю й активні здібності. Система збереження дозволяє повернутися до останньої пройденої хвилі, що робить проходження гнучким. Із технічного боку проєкт спирається на ресурси .tres, сигналінгову модель Godot, Vulkan-рендер та оптимізовану фізику, завдяки чому сцена без проблем утримує велику кількість об'єктів.

Ергономіці та реіграбельності приділено окрему увагу: змінні шаблони ворогів, випадкові події та глибока система модифікаторів тримають інтерес як у казуальної, так і у вимогливої аудиторії. Архітектура коду спроектована з урахуванням стандартних патернів Godot, що полегшує внесення майбутніх доповнень і баланс-правок.

Практична частина диплома дала змогу зміцнити навички об'єктно-орієнтованого програмування, роботи з внутрішніми економічними моделями та системами збереження даних. Підсумковий результат підтверджує, що обрана

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

технологічна база дозволяє швидко прототипувати, тестувати та доводити ідеї до готового продукту.

Таким чином, розроблений застосунок відповідає вимогам сучасної індустрії, демонструє грамотне використання відкритих інструментів і має потенціал для подальшого розширення або комерційного релізу.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vampire Survivors : [Електронний ресурс] / poncle. – Steam, 20.10.2022. – Режим доступу: <https://store.steampowered.com/app/1794680> (дата звернення: 03.05.2025).
2. Brotato : [Електронний ресурс] / Blobfish. – Steam, 23.06.2023. – Режим доступу: <https://store.steampowered.com/app/1942280> (дата звернення: 03.05.2025).
3. 20 Minutes Till Dawn : [Електронний ресурс] / flanne. – Steam, 08.06.2023. – Режим доступу: <https://store.steampowered.com/app/1966900> (дата звернення: 03.05.2025).
4. Magic Survival : [Електронний ресурс]. – Fandom Wiki. – Режим доступу: <https://magic-survival-rpg.fandom.com> (дата звернення: 03.05.2025).
5. Soulstone Survivors : [Електронний ресурс] / Game Smithing Ltd. – Steam, 07.11.2022. – Режим доступу: <https://store.steampowered.com/app/2066020> (дата звернення: 03.05.2025).
6. HoloCure – Save the Fans! : [Електронний ресурс] / KayAnimate. – Steam, 16.09.2023. – Режим доступу: <https://store.steampowered.com/app/2420510> (дата звернення: 03.05.2025).
7. The Binding of Isaac : [Електронний ресурс]. – Fandom Wiki. – Режим доступу: <https://bindingofisaac.fandom.com> (дата звернення: 03.05.2025).
8. Steam. Version 1.6: New engine & Local co-op : [Електронний ресурс]. – 11.08.2023. – Режим доступу: <https://steamcommunity.com/games/1794680/announcements/detail/3678929205875258039> (дата звернення: 03.05.2025).
9. Vampire Survivors – Wikipedia : [Електронний ресурс]. – Режим доступу: [https://en.wikipedia.org/wiki/Vampire\\_Survivors](https://en.wikipedia.org/wiki/Vampire_Survivors) (дата звернення: 03.05.2025).
10. Wales M. Vampire Survivors kicked off a game development gold rush... : GamesRadar+, 29.04.2025 : [Електронний ресурс]. – Режим доступу: <https://www.gamesradar.com/games/action/vampire-survivors-kicked-off-a-game->

	Вик.	Гуменко Я.М.			<i>ДП.ПЗ.211.06.ПЗ</i>	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

development-gold-rush-but-has-a-legitimately-new-genre-emerged-between-the-cash-ins/ (дата звернення: 03.05.2025).

11. Carless S. Charting the rise of (Vampire) Survivors-likes : GameDiscoverCo newsletter, 14.12.2024 : [Електронний ресурс]. – Режим доступу: <https://newsletter.gamediscover.co/p/charting-the-rise-of-vampire-survivors> (дата звернення: 03.05.2025).

12. Brotato – Wikipedia : [Електронний ресурс]. – Режим доступу: <https://en.wikipedia.org/wiki/Brotato> (03.05.2025).

13. Brotato. 1.0 Launch & Co-op Update : [Електронний ресурс] / Blobfish, 18.01.2024. – Режим доступу: <https://steamcommunity.com/app/1942280> (03.05.2025).

14. 20 Minutes Till Dawn – Wikipedia : [Електронний ресурс]. – Режим доступу: [https://en.wikipedia.org/wiki/20\\_Minutes\\_Till\\_Dawn](https://en.wikipedia.org/wiki/20_Minutes_Till_Dawn) (03.05.2025).

15. Magic Survival. Emerald Grimoire Update : [Електронний ресурс] / LEME, 15.04.2025. – Режим доступу: [https://www.reddit.com/r/magic\\_survival/comments/1gk3hg1/update\\_v094\\_details/](https://www.reddit.com/r/magic_survival/comments/1gk3hg1/update_v094_details/) (03.05.2025).

16. Soulstone Survivors – Wikipedia : [Електронний ресурс]. – Режим доступу: [https://soulstone-survivors.fandom.com/wiki/Soulstone\\_Survivors\\_Wiki](https://soulstone-survivors.fandom.com/wiki/Soulstone_Survivors_Wiki) (03.05.2025).

17. Soulstone Survivors. Titan's Reach Update : [Електронний ресурс] / Game Smithing Ltd., 12.03.2025. – Режим доступу: <https://store.steampowered.com/news/app/2066020/view/530970142088102697?l=ukrainian> (03.05.2025).

18. HoloCure. Blue Journey Update : [Електронний ресурс] / KayAnimate, 07.02.2025. – Режим доступу: [https://www.reddit.com/r/holocure/comments/1ja6beh/holocure\\_update\\_when/](https://www.reddit.com/r/holocure/comments/1ja6beh/holocure_update_when/) (03.05.2025).

19. Godot 4.3, a shared effort. – Godot Engine, 14.08.2024. – Режим доступу: <https://godotengine.org/releases/4.3> (03.05.2025).

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

20. Godot release policy. – Godot Docs. – Режим доступу: [https://docs.godotengine.org/en/stable/about/release\\_policy.html](https://docs.godotengine.org/en/stable/about/release_policy.html) (03.05.2025).
21. GDScript reference. – Godot Docs. – Режим доступу: [https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript\\_basics.html](https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html) (03.05.2025).
22. Godot Engine – Version history. – Wikipedia. – Режим доступу: [https://en.wikipedia.org/wiki/Godot\\_\(game\\_engine\)#Release\\_history](https://en.wikipedia.org/wiki/Godot_(game_engine)#Release_history) (03.05.2025).
23. Godot Docs. Main scene system. – Режим доступу: [https://docs.godotengine.org/en/stable/getting\\_started/first\\_2d\\_game/05.the\\_main\\_game\\_scene.html](https://docs.godotengine.org/en/stable/getting_started/first_2d_game/05.the_main_game_scene.html) (03.05.2025).
24. Godot Docs. Optimizing for speed. – Режим доступу: [https://docs.godotengine.org/ru/4.x/tutorials/performance/cpu\\_optimization.html](https://docs.godotengine.org/ru/4.x/tutorials/performance/cpu_optimization.html) (03.05.2025).
25. Godot Docs. Importing images and textures. – Режим доступу: [https://docs.godotengine.org/en/stable/tutorials/assets\\_pipeline/importing\\_images.html](https://docs.godotengine.org/en/stable/tutorials/assets_pipeline/importing_images.html) (03.05.2025).
26. Godot Docs. Retargeting 3D skeletons. – Режим доступу: [https://godot-doc.readthedocs.io/en/3.0/tutorials/3d/working\\_with\\_3d\\_skeletons.html](https://godot-doc.readthedocs.io/en/3.0/tutorials/3d/working_with_3d_skeletons.html) (03.05.2025).
27. Godot Docs. Physics introduction. – Режим доступу: [https://docs.godotengine.org/en/stable/tutorials/physics/physics\\_introduction.html](https://docs.godotengine.org/en/stable/tutorials/physics/physics_introduction.html) (03.05.2025).
28. Godot Docs. Exporting projects. – Режим доступу: [https://docs.godotengine.org/en/stable/tutorials/export/exporting\\_projects.html](https://docs.godotengine.org/en/stable/tutorials/export/exporting_projects.html) (03.05.2025).
29. Наказ про затвердження Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» : [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/laws/show/z0472-14#Text>(29.04.2025).

	Вик.	Гуменко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

30. ДСТУ EN ISO 9241-5:2004. Ергономічні вимоги до офісної роботи з відеодисплейними терміналами (ВДТ). Частина 5. Планування робочого місця та вимоги до постави. — Київ : ДП «УкрНДНЦ», 2004. — 24 с.

31. ДСТУ EN IEC 61000-4-3:2021. Електромагнітна сумісність. Частина 4-3. Методики випробування та вимірювання. Випробування на несприйнятливість до радіочастотних електромагнітних полів. — Київ : ДП «УкрНДНЦ», 2022. — 73 с.

32. ДСН 3.3.6.037-99. Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку. — Київ : МОЗ України, 1999. — 32 с.

33. ДСТУ EN ISO 12100:2016. Безпечність машин. Загальні принципи проєктування. Оцінювання ризиків та зменшення ризиків (EN ISO 12100:2010, IDT; ISO 12100:2010, IDT). — Київ : ДП «УкрНДНЦ», 2017. — 74 с.

34. НАПБ А.01.001-2014. Правила пожежної безпеки в Україні. — Київ : МВС України, 2015. — 155 с.

35. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами ЕОМ. — Київ : МОЗ України, 1998. — 29 с.

36. ДСН 173-96. Державні санітарні норми планування та забудови населених пунктів. — Київ : МОЗ України, 1996. — 19 с.

37. ДСТУ EN 15004-1:2015 (EN 15004-1:2008, IDT). Стаціонарні системи пожежогасіння. Газові вогнегасні установки. Частина 1. Проєктування, монтаж та обслуговування. — Київ : ДП «УкрНДНЦ», 2016. — 65 с.

38. ДНАОП 0.00-1.21-98. Правила безпечної експлуатації електроустановок споживачів. — Київ : Держнаглядохоронпраці України, 1998. — 170 с.

39. ДСН 3.3.6.042-99. Державні санітарні норми параметрів мікроклімату виробничих приміщень. — Київ : МОЗ України, 1999. — 28 с.

40. ДСТУ EN 50678:2022 (EN 50678:2020/AC:2021-04, IDT). Загальний порядок перевірки ефективності захисних заходів електрообладнання після ремонту. — Київ : ДП «УкрНДНЦ», 2022. — 24 с.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				68
Змн.	Арк.	№ докум.	Підпис	Дата		



41. ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціонування. — Київ : Мінрегіон України, 2013. — 188 с.
42. ДСТУ HD 60364-6:2022 Електроустановки низьковольтні. Частина 6. Перевірка. — Київ : ДП «УкрНДНЦ», 2018. — 54 с.
43. ДСТУ EN IEC 31010:2022. Керування ризиками. Методи оцінки ризиків (EN IEC 31010:2019, IDT; IEC 31010:2019, IDT). — Київ : ДП «УкрНДНЦ», 2023. — 80 с.
44. ДСТУ EN 1366-3:2021 (EN 1366-3:2009, IDT). Випробування інженерних систем на вогнестійкість. Частина 3. Проходки інженерних комунікацій — Київ : ДП «УкрНДНЦ», 2015. — 54 с.
45. ДСТУ EN 12101-8:2014 Системи димовидалення. Клапани контрольованого димовидалення. — Київ : ДП «УкрНДНЦ», 2014. — 58 с.
46. ДСТУ ISO 9241-5:2004. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 5. Вимоги до компонування робочого місця та до робочої пози (ISO 9241-5:1998, IDT). — Київ : ДП «УкрНДНЦ», 2005. — 24 с.
47. ДСТУ EN ISO 9241-16:2022. Ергономічні вимоги до офісної роботи з терміналами візуального дисплея (VDT). Частина 16. Діалоги прямого управління (EN ISO 9241-16:1999, IDT; ISO 9241-16:1999, IDT). — Київ : ДП «УкрНДНЦ», 2023. — 24 с.

	Вик.	Гуненко Я.М.			ДП.ПЗ.211.06.ПЗ	Арк.
	Пер.	Ланська С.С.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

## ДОДАТОК А

### Текст програми

#### Лістинг А.1 – Текст файлу EnemyBase.gd

```
class_name EnemyBase
extends CharacterBody2D

signal died

@export var base_health: float = 1
@export var base_speed: float = 60.0
@export var base_damage: float = 1
@export var health_growth_factor: float = 1.25
@export var damage_growth_factor: float = 1.10
@export var speed_growth_factor: float = 1.05
@export var xp_growth_factor: float = 1.15
@export var coins_growth_factor: float = 1.03
@export var base_xp_reward: float = 1
@export var base_coins_reward: float = 1

var                                     XP_ITEM_SCENE =
load("res://src/Scenes/Experience_item/Exp_scene.tscn")

var movement_speed: float
var health: float
var damage: float
var armor: int = 0
var stun_duration: float = 1.0

var xp_reward: float
var coins_reward: float

@onready var player = get_tree().get_first_node_in_group("Player")
@onready var hitbox: Area2D = $HitBox
@onready var hurtbox: Area2D = $HurtBox
@onready var sprite: AnimatedSprite2D = null

var is_attacking: bool = false
var is_dead: bool = false
var is_stunned: bool = false
var last_flip: int = 1

var __player_in_attack_range: Node = null

func apply_wave_modifiers(current_wave: float) -> void:
    var wave_index = max(current_wave - 1, 0)
    health = base_health * pow(health_growth_factor, wave_index)
    damage = base_damage * pow(damage_growth_factor, wave_index)
    movement_speed = (base_speed * pow(speed_growth_factor, wave_index))
+ randf_range(-10, 10)
    xp_reward = base_xp_reward * pow(xp_growth_factor, wave_index)
    coins_reward = base_coins_reward * pow(coins_growth_factor,
wave_index)
```

```

func set_sprite(a: AnimatedSprite2D) -> void:
    sprite = a

func take_damage(amount: float) -> void:
    if is_dead:
        return
    health -= max(amount - armor, 1)
    if health <= 0:
        __die()
    else:
        if is_attacking:
            is_attacking = false
        sprite.play("take_damage")
        __apply_stun()

func _physics_process(delta: float) -> void:
    if is_dead or is_stunned:
        return
    if player and is_instance_valid(player):
        var direction = (player.global_position -
global_position).normalized()
        if __player_in_attack_range:
            velocity = Vector2.ZERO
            if not is_attacking:
                start_attack()
        else:
            velocity = direction * movement_speed
            face_player(direction)
            if not is_attacking:
                play_walk_animation()
            move_and_slide()

func start_attack() -> void:
    is_attacking = true
    sprite.play("attack")

func face_player(direction: Vector2) -> void:
    var is_facing_left = direction.x < 0
    var flip_value = 1
    if is_facing_left:
        flip_value = -1
    if last_flip != flip_value:
        sprite.flip_h = is_facing_left
        update_hitbox_position(is_facing_left)
        last_flip = flip_value

func update_hitbox_position(is_facing_left: bool) -> void:
    var flip_value = 1
    if is_facing_left:
        flip_value = -1
    hitbox.scale.x = abs(hitbox.scale.x) * flip_value
    hurtbox.scale.x = abs(hurtbox.scale.x) * flip_value
    hitbox.position.x = sprite.position.x
    hurtbox.position.x = sprite.position.x

```

```

func play_walk_animation() -> void:
    sprite.play("walk")

func __die() -> void:
    if not is_dead:
        is_dead = true
        sprite.play("die")
        give_coins_to_player()
        call_deferred("drop_experience")
        emit_signal("died")

func give_coins_to_player() -> void:
    var player_stats =
get_tree().get_first_node_in_group("PlayerStats")
    if player_stats:
        player_stats.gain_coins(round(coins_reward))

func drop_experience() -> void:
    var xp_item = XP_ITEM_SCENE.instantiate()
    xp_item.global_position = self.position
    xp_item.set_experience(xp_reward * GameManager.__xp_scale)
    get_parent().add_child(xp_item)

func __apply_stun() -> void:
    is_stunned = true
    await get_tree().create_timer(stun_duration).timeout
    is_stunned = false

func __animation_finished() -> void:
    match sprite.animation:
        "attack":
            if __player_in_attack_range and
__player_in_attack_range.has_method("take_damage"):
                __player_in_attack_range.take_damage(damage)
            is_attacking = false
        "die":
            call_deferred("queue_free")

func __player_entered(area: Area2D) -> void:
    __player_in_attack_range = area.get_parent()

func __player_exited(area: Area2D) -> void:
    __player_in_attack_range = null

func play_sound(effect_path: String):
    pass

```

## Лістинг А.2 – Текст файлу NightBorne.gd

```

extends EnemyBase
class_name Nightborne

@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D
var music_volume: int

func _ready() -> void:

```

```

movement_speed = 70
health = 10
damage = 4
var new_sprite: AnimatedSprite2D = $NightBorne
set_sprite(new_sprite)
var config = ConfigFile.new()
if config.load("user://settings.cfg") == OK:
    music_volume = config.get_value("Settings", "s_volume", 40)
else:
    music_volume = 0
sfx.volume_db = music_volume

func start_attack() -> void:
    is_attacking = true
    play_sound("res://src/Enemies/Nightborne/Sword_attack.wav")
    sprite.play("attack")

func take_damage(amount: float) -> void:
    if is_dead:
        return
    health -= max(amount - armor, 1)
    if health <= 0:
        __die()
    else:
        if is_attacking:
            is_attacking = false
        play_sound("res://src/Enemies/hurt.wav")
        sprite.play("take_damage")
        __apply_stun()

func play_sound(effect_path: String):
    var effect_stream = load(effect_path)
    if effect_stream:
        sfx.stop()
        sfx.stream = effect_stream
        sfx.play()
    else:
        push_error("Не вдалося завантажити ефект: " + effect_path)

func __die() -> void:
    if not is_dead:
        is_dead = true
        sprite.play(&"die")
        play_sound("res://src/Enemies/Nightborne/monster_die.wav")
        give_coins_to_player()
        call_deferred("drop_experience")
        emit_signal("died")

```

### ЛІСТИНГ А.3 – Текст файлу SkeletonEnemy.gd

```

extends EnemyBase
class_name Skelet

@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D
var music_volume: int

```

```

func _ready() -> void:
    movement_speed = 40
    health = 5
    damage = 1.5
    var new_sprite: AnimatedSprite2D = $Skeleton
    set_sprite(new_sprite)
    var config = ConfigFile.new()
    if config.load("user://settings.cfg") == OK:
        music_volume = config.get_value("Settings", "s_volume", 40)
    else:
        music_volume = 0
    sfx.volume_db = music_volume

func __die() -> void:
    if not is_dead:
        is_dead = true
        sprite.play(&"die")
        give_coins_to_player()
        call_deferred("drop_experience")
        emit_signal("died")

func start_attack() -> void:
    is_attacking = true
    play_sound("res://src/Enemies/Skeleton/Skelet_attack.wav")
    sprite.play("attack")

func take_damage(amount: float) -> void:
    if is_dead:
        return
    health -= max(amount - armor, 1)
    if health <= 0:
        __die()
    else:
        if is_attacking:
            is_attacking = false
        play_sound("res://src/Enemies/hurt.wav")
        sprite.play("take_damage")
        __apply_stun()

func play_sound(effect_path: String):
    var effect_stream = load(effect_path)
    if effect_stream:
        sfx.stop()
        sfx.stream = effect_stream
        sfx.play()
    else:
        push_error("Не вдалося завантажити ефект: " + effect_path)

```

#### ЛІСТИНГ А.4 – Текст файлу CharacterBody2D.gd

```

extends EnemyBase
class_name Reaper

@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D

```

```

var music_volume: int

func _ready() -> void:
    movement_speed = 70
    health = 25
    damage = 1.5
    var new_sprite: AnimatedSprite2D = $Reaper
    set_sprite(new_sprite)
    $WordCollision.scale.x = abs($WordCollision.scale.x) * -1
    $WordCollision.position = sprite.position
    $WordCollision.position.x -= 45
    $WordCollision.position.y += 20
    var config = ConfigFile.new()
    if config.load("user://settings.cfg") == OK:
        music_volume = config.get_value("Settings", "s_volume", 40)
    else:
        music_volume = 0
    sfx.volume_db = music_volume

func face_player(direction: Vector2) -> void:
    var is_facing_left = direction.x < 0
    var flip_value = 1
    if is_facing_left:
        flip_value = -1
    if last_flip != flip_value:
        sprite.flip_h = is_facing_left
        $WordCollision.scale.x = abs($WordCollision.scale.x) * flip_value
        $WordCollision.position = sprite.position
        if is_facing_left:
            $WordCollision.position.x += 40
            $WordCollision.position.y += 20
        else:
            $WordCollision.position.x -= 40
            $WordCollision.position.y += 15
        update_hitbox_position(is_facing_left)
        last_flip = flip_value

func __die() -> void:
    if not is_dead:
        is_dead = true
        sprite.play(&"die")
        play_sound("res://src/Enemies/Zhnec/monster_die.wav")
        give_coins_to_player()
        call_deferred("drop_experience")
        emit_signal("died")

func start_attack() -> void:
    is_attacking = true
    play_sound("res://src/Enemies/Zhnec/z nec_attack.wav")
    sprite.play("attack")

func take_damage(amount: float) -> void:
    if is_dead:

```

```

        return
    health -= max(amount - armor, 1)
    if health <= 0:
        __die()
    else:
        if is_attacking:
            is_attacking = false
        play_sound("res://src/Enemies/hurt.wav")
        sprite.play("take_damage")
        __apply_stun()

func play_sound(effect_path: String):
    var effect_stream = load(effect_path)
    if effect_stream:
        sfx.stop()
        sfx.stream = effect_stream
        sfx.play()
    else:
        push_error("Не вдалося завантажити ефект: " + effect_path)

```

### Лістинг А.5 – Текст файлу Player.gd

```

class_name PlayerCharacterBody extends CharacterBody2D

@export_range(10.0, 500.0) var movement_speed : float = 100.0

@onready var stats = $stats
@onready var hitbox: Area2D = $HitBox
@onready var hurtbox: Area2D = $HurtBox
@onready var sprite: AnimatedSprite2D = %AnimatedSprite2D
@onready var inventory_ui: SimplifiedInventory = $InventoryUi
@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D

var last_flip: int = 1
var __is_in_zone: bool = false
var __enemy_in_attack_range: EnemyBase = null
var __facing_left: bool = false
var __attacking: bool = false
var music_volume: int = 150

func _ready() -> void:
    stats.connect(&"health_changed", Callable(self, &"_on_health_changed"))
    stats.connect(&"died", Callable(self, &"_on_player_died"))
    stats.connect(&"level_up", Callable(self, &"_on_level_up"))
    __update_hitbox_position()
    var config = ConfigFile.new()
    if config.load("user://settings.cfg") == OK:
        music_volume = config.get_value("Settings", "s_volume", 40)
    else:
        music_volume = 0
    sfx.volume_db = music_volume

func _physics_process(delta: float) -> void:
    if stats.is_dead:
        return

```



```

__enemy_in_attack_range = null
for area in hitbox.get_overlapping_areas():
    var parent = area.get_parent()
    if parent is EnemyBase:
        __enemy_in_attack_range = parent
        __is_in_zone = true
        break
if not __enemy_in_attack_range:
    __is_in_zone = false

if not __attacking:
    if __enemy_in_attack_range:
        play_effect("res://src/Player/SFX/player_attack.wav")
        perform_attack()
    else:
        movement(delta)

func movement(delta: float) -> void:
    var move = Input.get_vector(
        &"player_move_left",
        &"player_move_right",
        &"player_move_up",
        &"player_move_down"
    ).normalized()
    if move.x != 0:
        __facing_left = move.x < 0
        sprite.flip_h = __facing_left
        __face_player()
    if move != Vector2.ZERO:
        sprite.play(&"walk")
        sprite.speed_scale = 1
    else:
        sprite.play(&"idle")
        sprite.speed_scale = 1
    inventory_ui.synchronize_weapon_orientation(__facing_left)
    velocity = move * self.movement_speed
    move_and_slide()

func __face_player() -> void:
    var new_flip = -1 if __facing_left else 1
    if last_flip != new_flip:
        __update_hitbox_position()
        last_flip = new_flip

func __update_hitbox_position() -> void:
    var flip_value = -1 if __facing_left else 1
    hitbox.scale.x = abs(hitbox.scale.x) * flip_value
    hurtbox.scale.x = abs(hurtbox.scale.x) * flip_value
    hitbox.position.x = sprite.position.x
    hurtbox.position.x = sprite.position.x

func take_damage(amount: float) -> void:
    play_effect("res://src/Player/SFX/hurt.wav")
    stats.take_damage(amount)

func _on_player_died():

```

```

        sprite.play(&"die")
        play_effect("res://src/Player/SFX/Player_die.wav")
        get_tree().paused = true

func perform_attack():
    if stats.is_dead:
        return
    if not __enemy_in_attack_range:
        sprite.play(&"idle")
        sprite.speed_scale = 1
        return
    if __enemy_in_attack_range or __is_in_zone:
        __attacking = true
        sprite.play(&"attack3")
        sprite.speed_scale = stats.__attack_speed
        await self.sprite.animation_finished
        __attacking = false

func _on_animation_finished():
    match sprite.animation:
        &"attack3":
            if __enemy_in_attack_range && __is_in_zone:
                var damage = stats.calculate_damage()
                __enemy_in_attack_range.take_damage(damage)

func __enemy_entered_attack_range(area: Area2D) -> void:
    var enemy = area.get_parent()
    if enemy is EnemyBase:
        __is_in_zone = true
        __enemy_in_attack_range = enemy

func __enemy_exited_attack_range(area: Area2D) -> void:
    __enemy_in_attack_range = null
    __is_in_zone = false
    return

func play_effect(effect_path: String) -> void:
    var effect_stream = load(effect_path)
    if effect_stream:
        sfx.stop()
        sfx.stream = effect_stream
        sfx.play()
    else:
        push_error("Не вдалося завантажити ефект: " + effect_path)

func _clear_inventory():
    for i in range(inventory_ui.slots.size()-1):
        inventory_ui.remove_weapon(i)

```

## ЛІСТИНГ А.6 – save\_slot\_scene.gd

```

extends Control

signal slot_selected(slot: int)
signal slot_deleted(slot: int)

```

```

var load_next_wave_after_load := false
signal new_game_requested
signal closed()

@export var is_saving: bool = false
@export var block_cancel_button: bool = false

@onready var slot_1: Button = $VBoxContainer2/VBoxContainer/slot1/Slot1
@onready var del_1: Button = $VBoxContainer2/VBoxContainer/slot1/Del1
@onready var slot_2: Button = $VBoxContainer2/VBoxContainer/slot2/Slot2
@onready var del_2: Button = $VBoxContainer2/VBoxContainer/slot2/Del2
@onready var slot_3: Button = $VBoxContainer2/VBoxContainer/slot3/Slot3
@onready var del_3: Button = $VBoxContainer2/VBoxContainer/slot3/Del3
@onready var new_game: Button = $VBoxContainer2/HBoxContainer/NewGame
@onready var cancel_button: Button = $VBoxContainer2/HBoxContainer2/Cancel

@onready var slot_buttons := [slot_1, slot_2, slot_3]
@onready var delete_buttons := [del_1, del_2, del_3]

@onready var popup: Popup = $Popup
@onready var popup_label: Label = $Popup/PanelContainer/VBoxContainer/Label
@onready var popup_yes: Button = $Popup/PanelContainer/VBoxContainer/Buttons/Yes
@onready var popup_no: Button = $Popup/PanelContainer/VBoxContainer/Buttons/No

var pending_slot: int = -1
var pending_delete_slot: int = -1

func _ready():
    _update_slots()

    for i in range(3):
        slot_buttons[i].connect("pressed", Callable(self,
            "_on_slot_pressed").bind(i + 1))
        delete_buttons[i].connect("pressed", Callable(self,
            "_on_delete_pressed").bind(i + 1))

        new_game.connect("pressed", Callable(self, "_on_new_game_pressed"))
        cancel_button.connect("pressed", Callable(self,
            "_on_cancel_pressed"))
        popup_yes.connect("pressed", Callable(self, "_on_popup_yes"))
        popup_no.connect("pressed", Callable(self, "_on_popup_no"))

    if block_cancel_button:
        cancel_button.disabled = true

func _update_slots():
    for i in range(3):
        var info = GameManager.get_save_summary(i + 1)
        if info.exists:
            slot_buttons[i].text = tr("SAVE_SLOT_INFO") % [i + 1, info.level,
            info.wave, info.coins]
            slot_buttons[i].disabled = false
            delete_buttons[i].disabled = false
        else:

```

```

        slot_buttons[i].text = tr("SAVE_SLOT_EMPTY") % (i + 1)
        slot_buttons[i].disabled = not is_saving
        delete_buttons[i].disabled = true
    new_game.disabled = is_saving

func _on_slot_pressed(slot: int):
    var info = GameManager.get_save_summary(slot)
    if is_saving and info.exists:
        pending_slot = slot
        pending_delete_slot = -1
        popup_label.text = tr("OVERWRITE_SLOT")
        popup_yes.text = tr("YES")
        popup_no.text = tr("NO") #% slot
        popup.popup_centered()
    else:
        emit_signal("slot_selected", slot)
        queue_free()

func _on_delete_pressed(slot: int):
    pending_delete_slot = slot
    pending_slot = -1
    popup_label.text = tr("DELETE_SLOT") #% slot
    popup_yes.text = tr("YES")
    popup_no.text = tr("NO")
    popup.popup_centered()

func _on_popup_yes():
    if pending_slot != -1:
        emit_signal("slot_selected", pending_slot)
        queue_free()
    elif pending_delete_slot != -1:
        GameManager.delete_save(pending_delete_slot)
        _update_slots()
        pending_delete_slot = -1
    popup.hide()

func _on_popup_no():
    pending_slot = -1
    pending_delete_slot = -1
    popup.hide()

func _on_new_game_pressed():
    emit_signal("new_game_requested")
    queue_free()

func _on_close_pressed():
    emit_signal("closed")
    queue_free()

func _on_cancel_pressed() -> void:
    emit_signal("closed")
    queue_free()

```

## Лістинг А.7 – bufftscn.gd

```
extends Control

@onready var color_rect: ColorRect = $ColorRect
@onready var buttons = [
    $Buffs_block/First_button,
    $Buffs_block/Second_button,
    $Buffs_block/Third_button
]

enum Stats { HEALTH, DAMAGE, ARMOR, CRIT_CHANCE, ATTACK_SPEED }

const STAT_NAMES = {
    Stats.HEALTH: "Здоров'я",
    Stats.DAMAGE: "Урон",
    Stats.ARMOR: "Броня",
    Stats.CRIT_CHANCE: "Шанс крит. удару",
    Stats.ATTACK_SPEED: "Швидкість атаки"
}

const STAT_KEYS = {
    Stats.HEALTH: "stat_health",
    Stats.DAMAGE: "stat_damage",
    Stats.ARMOR: "stat_armor",
    Stats.CRIT_CHANCE: "stat_crit_chance",
    Stats.ATTACK_SPEED: "stat_attack_speed"
}

const CARDS_STATS = [
    { "stat": Stats.HEALTH, "description": "HP_BUFF", "icon":
"res://src/Scenes/Buffs/Buff_icons/hp_buff.webp" },
    { "stat": Stats.DAMAGE, "description": "DAMAGE_BUFF", "icon":
"res://src/Scenes/Buffs/Buff_icons/Attack_buff.webp" },
    { "stat": Stats.ARMOR, "description": "ARMOR_BUFF", "icon":
"res://src/Scenes/Buffs/Buff_icons/armor.png" },
    { "stat": Stats.CRIT_CHANCE, "description": "CRIT_BUFF", "icon":
"res://src/Scenes/Buffs/Buff_icons/crit_chace.png" },
    { "stat": Stats.ATTACK_SPEED, "description": "AS_BUFF", "icon":
"res://src/Scenes/Buffs/Buff_icons/attack_speed.png" }
]

const BUFF_VALUES = {
    Stats.HEALTH: 20,
    Stats.DAMAGE: 2,
    Stats.ARMOR: 1,
    Stats.CRIT_CHANCE: 0.05,
    Stats.ATTACK_SPEED: 0.2
}

func _ready() -> void:
    get_tree().paused = true
    process_mode = Node.PROCESS_MODE_ALWAYS
    color_rect.color = Color(0.1, 0.1, 0.1, 0.8)
    var shuffled_cards = CARDS_STATS.duplicate()
    shuffled_cards.shuffle()
```

```

    for i in range(buttons.size()):
        var card = shuffled_cards[i]
        fill_button(buttons[i], card)
        buttons[i].set_meta("card", card)
        buttons[i].connect("pressed", Callable(self,
"_on_buff_button_pressed").bind(buttons[i]))

func fill_button(button: Button, card: Dictionary) -> void:
    var details = button.get_node_or_null("Button_details")
    if details:
        var stat_name_label = details.get_node_or_null("Stat_name")
        if stat_name_label:
            stat_name_label.text = tr(STAT_KEYS.get(card["stat"],
"stat_unknown"))
            stat_name_label.horizontal_alignment =
HORIZONTAL_ALIGNMENT_CENTER
            stat_name_label.vertical_alignment = VERTICAL_ALIGNMENT_CENTER
        var stat_icon = details.get_node_or_null("Stat_ico")
        if stat_icon:
            var texture = ResourceLoader.load(card["icon"])
            if texture and texture is Texture2D:
                stat_icon.texture = texture
            else:
                print_debug("Помилка завантаження іконки: ", card["icon"])
                stat_icon.stretch_mode =
TextureRect.STRETCH_KEEP_ASPECT_CENTERED
            var description_label = details.get_node_or_null("Description")
            if description_label:
                description_label.text = tr("desc_" +
card["description"].to_lower())
                description_label.horizontal_alignment =
HORIZONTAL_ALIGNMENT_CENTER
                description_label.vertical_alignment = VERTICAL_ALIGNMENT_CENTER

func _on_buff_button_pressed(button: Button) -> void:
    var card: Dictionary = button.get_meta("card")
    var stat = card["stat"]
    var buff_value = BUFF_VALUES[stat]
    var player = get_tree().current_scene.get_node("Player")
    if player:
        player.stats.apply_buff(stat, buff_value)
        var buff_name = STAT_NAMES.get(stat, "Невідомий бафф")
    else:
        print("Помилка: гравець не знайдений!")
        get_tree().paused = false
        player.stats.print_stats()
        queue_free()

```

## Лістинг А.8 – exp\_scene.gd

```

class_name Experience
extends Node2D

@onready var player = get_tree().get_first_node_in_group(&"Player")
var __player_in_attack_range : PlayerCharacterBody = null
var __experience: float = 1

```

```

func _process(delta: float) -> void:
    if __player_in_attack_range != null and __experience!=0:
        player.stats.gain_experience(__experience)
        queue_free()

func _on_area_2d_area_entered(area: Area2D) -> void:
    __player_in_attack_range = area.get_parent()

func _on_area_2d_area_exited(area: Area2D) -> void:
    __player_in_attack_range = null

func set_experience(a: float) -> void:
    if a >= 0:
        __experience = a

```

### Лістинг А.9 – level\_test.gd

```

extends Node2D

@onready var pause_screen: CanvasLayer = $Pause_Screen
@onready var player = %Player
@onready var exp_bar: ProgressBar = $UI/MarginContainer/VBoxContainer/HBoxContainer2/Exp_bar
@onready var label_2: Label = %Label2
@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D
@export var shop_scene: PackedScene
@onready var spawner: Spawner_logic = $Spawner
@onready var label: Label = $UI/MarginContainer/VBoxContainer/HBoxContainer/Label
@onready var canvas_layer: CanvasLayer = $CanvasLayer

const BUFF_SCENE = preload("res://src/Scenes/Buffs/Buffstscn.tscn")
var __load_next_wave_after_load := false

func _ready():
    $UI.visible = true
    pause_screen.visible = false
    player.stats.connect("show_buff_cards", Callable(self,
    "_on_show_buff_cards"))
    player.stats.connect("died", Callable(self, "_on_died"))
    %GameOver.visible = false
    var has_any_save = false
    for i in range(1, 4):
        if GameManager.get_save_summary(i).exists:
            has_any_save = true
            break
    if has_any_save:
        _show_save_menu(false, true)
    else:
        _start_new_game()

func _process(_delta: float) -> void:
    %Health_bar.value = player.stats.get_health()
    exp_bar.value = player.stats.get_exp()

```

```

        label_2.text = tr("BALANCE") + " : " +
str(round(player.stats.__coins))

func _on_resume_pressed() -> void:
    get_tree().paused = false
    pause_screen.visible = false

func _input(event):
    if event.is_action_pressed("Pause"):
        if not get_tree().paused:
            get_tree().paused = true
            pause_screen.visible = true

func _on_main_menu_pressed() -> void:
    get_tree().paused = false
    get_tree().change_scene_to_file("res://src/Scenes/Setting_w/setting
gs.tscn")

func _on_exit_pressed() -> void:
    get_tree().paused = false
    get_tree().change_scene_to_file("res://src/Scenes/MainMenu/MainSce
ne.tscn")

func _on_pause_ui_pressed() -> void:
    get_tree().paused = true
    pause_screen.visible = true

func _on_show_buff_cards() -> void:
    var buff_scene_instance = BUFF_SCENE.instantiate()
    get_tree().current_scene.add_child(buff_scene_instance)
    await get_tree().process_frame
    var player_position = player.global_position
    var buff_size = buff_scene_instance.get_rect().size
    buff_scene_instance.global_position = player_position -
Vector2(buff_size.x / 2, buff_size.y * 0.6)

func _on_spawner_wave_finished() -> void:
    var window_size = get_viewport().get_visible_rect().size
    var center_position = window_size / 2
    await get_tree().process_frame
    player.global_position = center_position
    player.stats.__current_health = player.stats.__max_health
    shop_scene = load("res://src/shop/shop2.tscn")
    var shop_scene_instance = shop_scene.instantiate()
    canvas_layer.add_child(shop_scene_instance)
    var base_node = shop_scene_instance.get_node("Base")
    var shop_rect = base_node.get_rect()
    var shop_size = shop_rect.size
    var shop_control = shop_scene_instance.get_node("Base")
    shop_control.connect("shop_closed", Callable(self,
"_on_control_shop_closed"))
    GameManager.save_next_wave = true
    shop_control.connect("save_progress", Callable(self,
"_on_save_game_menu"))
    player.stats.heal()

```



```

    get_tree().paused = true

func _on_control_shop_closed() -> void:
    for child in canvas_layer.get_children():
        child.queue_free()
    get_tree().paused = false
    _pick_up_experience()
    spawner.start_wave()
    shop_scene = null

func _on_spawner_wave_started() -> void:
    label.text = tr("WAVE") + ": " + str(spawner.get_current_wave())

func _on_exit_button_pressed() -> void:
    get_tree().quit()

func _on_new_game_pressed() -> void:
    respawn_player()
    player.stats._default_stats()
    player._clear_inventory()
    _clear_lost_xp()
    spawner.set_current_wave(0)
    %GameOver.visible = false
    get_tree().paused = false

func _on_load_game_pressed() -> void:
    _show_save_menu(false)

func _on_save_game_menu() -> void:
    _show_save_menu(true)

func _on_slot_selected(slot: int, is_saving: bool):
    if is_saving:
        GameManager.save_game(slot, player.stats, player.inventory_ui,
spawner)
        print("Сохраниено в слот", slot)
    else:
        var data = GameManager.load_game(slot)
        if data:
            player.stats.from_dict(data["player_stats"])
            player.inventory_ui.load_inventory_data(data["inventory"])
            var loaded_wave = data.get("wave", 1)
            if not __load_next_wave_after_load and loaded_wave > 0:
                loaded_wave -= 1
            spawner.set_current_wave(loaded_wave)
            respawn_player()
            if __load_next_wave_after_load:
                spawner.start_wave()
            get_tree().paused = false
            %GameOver.visible = false

func _show_save_menu(is_saving: bool, block_cancel := false, load_next_wave
:= false):
    var menu_scene = preload("res://src/Saves
slots/SaveSlotsScene.tscn")
    var menu = menu_scene.instantiate()

```

```

    menu.is_saving = is_saving
    menu.block_cancel_button = block_cancel
    menu.load_next_wave_after_load = load_next_wave
    var new_canvas_layer = CanvasLayer.new()
    new_canvas_layer.name = "SaveMenuCanvas"
    new_canvas_layer.add_child(menu)
    add_child(new_canvas_layer)
    menu.connect("slot_selected", Callable(self,
"_on_slot_selected").bind(is_saving))
    menu.connect("new_game_requested", Callable(self,
"_start_new_game"))
    __load_next_wave_after_load = load_next_wave

func respawn_player():
    var center = get_viewport().size / 2
    player.global_position = center
    _clear_lost_xp()
    player.stats.revive()
    player.sprite.play("idle")

func _on_died():
    get_tree().paused = true
    %GameOver.visible = true

func _clear_lost_xp():
    for child in get_tree().get_current_scene().get_children():
        if child is Experience:
            child.queue_free()

func _start_new_game():
    player.stats._default_stats()
    player._clear_inventory()
    spawner.set_current_wave(0)
    respawn_player()
    %GameOver.visible = false
    get_tree().paused = false

func _pick_up_experience():
    for child in get_tree().get_current_scene().get_children():
        if child is Experience:
            player.stats.gain_experience(child.__experience)
            child.queue_free()

func _on_save_last_wave_pressed() -> void:
    GameManager.save_next_wave = false
    player.stats.heal()
    _show_save_menu(true)

```

## ЛІСТИНГ A.10 – settings.gd

```

extends Control

const CONFIG_PATH = "user://settings.cfg"
var hp_scale = false

```

```

var gold_scale = false
var reduced_experience = false
var sound_volume = 0
var brightness = 1
var music_volume = 0
var localisation = "uk"

@onready var brightness_slider: HSlider = $"mainContainer/ProgressBars and
language Choice/ProgressBars and language
Choice/ProgressBars/BrightnessProgressBar/brightness_slider"
@onready var music_player: AudioStreamPlayer2D = $bg_music
@onready var ru_button: Button = $"mainContainer/ProgressBars and language
Choice/ProgressBars and language
Choice/Localisation_buttons/VBoxContainer/RuButton"
@onready var ua_button: Button = $"mainContainer/ProgressBars and language
Choice/ProgressBars and language
Choice/Localisation_buttons/VBoxContainer/UaButton"
@onready var music_mute_off: Button = $"mainContainer/Additional and mute
buttons/Additional and mute buttons container/MusicBox/music_mute_off"
@onready var sound_mute_off: Button = $"mainContainer/Additional and mute
buttons/Additional and mute buttons container/SoundBox/sound_mute_off"
@onready var low_exp_scale_button: CheckButton = $"mainContainer/Additional
and mute buttons/Additional and mute buttons
container/Addition_settings_border_box/Additional_Settings/Experience_sca
le/low_exp_scale_button"
@onready var caption_bright_2: Label = $"mainContainer/ProgressBars and
language Choice/ProgressBars and language
Choice/ProgressBars/BrightnessProgressBar/Caption_bright2"
@onready var music_sound_slider: HSlider = $"mainContainer/ProgressBars and
language Choice/ProgressBars and language
Choice/ProgressBars/MusicProgressBar/music_sound_slider"
@onready var volume_slider: HSlider = $"mainContainer/ProgressBars and
language Choice/ProgressBars and language
Choice/ProgressBars/SoundProgressBar/volume_slider"
@onready var en_button: Button = $"mainContainer/ProgressBars and language
Choice/ProgressBars and language
Choice/Localisation_buttons/VBoxContainer/EnButton"
@onready var less_gold_button: CheckButton = $"mainContainer/Additional and
mute buttons/Additional and mute buttons
container/Addition_settings_border_box/Additional_Settings/HPBarView/less
_gold_button"
@onready var hard_mode: CheckButton = $"mainContainer/Additional and mute
buttons/Additional and mute buttons
container/Addition_settings_border_box/Additional_Settings/Hard_mode/hard
_mode"

func _ready():
    load_settings()
    update_ui()

#Конфиг

func load_settings():
    var config = ConfigFile.new()
    if config.load(CONFIG_PATH) == OK:
        hp_scale = config.get_value("Settings", "hp_scale", false)

```

```

        gold_scale = config.get_value("Settings", "gold_scale", false)
        reduced_experience = config.get_value("Settings",
"reduced_experience", false)
        sound_volume = config.get_value("Settings", "s_volume", 0)
        music_volume = config.get_value("Settings", "m_volume", 0)
        brightness = config.get_value("Settings", "brightness", 1)
        localisation = config.get_value("Settings", "locale", "en")
        TranslationServer.set_locale(localisation)

func save_settings():
    var config = ConfigFile.new()
    config.set_value("Settings", "hp_scale", hp_scale)
    config.set_value("Settings", "gold_scale", gold_scale)
    config.set_value("Settings", "reduced_experience",
reduced_experience)
    config.set_value("Settings", "m_volume", music_volume)
    config.set_value("Settings", "s_volume", sound_volume)
    config.set_value("Settings", "brightness", brightness)
    config.set_value("Settings", "locale", localisation)
    config.save(CONFIG_PATH)

func update_ui():
    hard_mode.set_pressed(hp_scale)
    less_gold_button.set_pressed(gold_scale)
    low_exp_scale_button.set_pressed(reduced_experience)
    brightness_slider.value = ((brightness - 0.4)/0.6)
    caption_bright_2.text = str((brightness_slider.value)*100) + "%"
    music_sound_slider.value = 1 - (music_volume/-(10))
    %Caption_music2.text = str(music_sound_slider.value * 100) + "%"
    volumnme_slider.value = 1 - (sound_volume/-(10))
    %Caption_sound2.text = str(volumnme_slider.value * 100) + "%"
    var current_locale = TranslationServer.get_locale()
    match current_locale:
        "ru":
            ru_button.grab_focus()
        "uk":
            ua_button.grab_focus()
        _:
            en_button.grab_focus()

func _on_back_main_pressed():
    get_tree().change_scene_to_file("res://src/Scenes/MainMenu/MainScene.tscn")

#Кнопки мута

func _on_music_mute_off_pressed():
    if music_volume == -80:
        music_volume = 0
    else:
        music_volume = -80
    music_sound_slider.value = 1 - (music_volume/-(10))
    %Caption_music2.text = str(music_sound_slider.value * 100) + "%"
    save_settings()
    music_player.set_music_volume(music_volume)

```

```

func _on_sound_mute_off_pressed():
    if sound_volume == -80:
        sound_volume = 0
    else:
        sound_volume = -80
    %volumme_slider.value = 1 - (sound_volume/-(10))
    %Caption_sound2.text = str(%volumme_slider.value * 100) + "%"
    save_settings()
    GameManager.sfx_volume = sound_volume

#Локализация

func _on_ru_button_pressed():
    TranslationServer.set_locale("ru")
    localisation = "ru"
    save_settings()

func _on_ua_button_pressed():
    TranslationServer.set_locale("uk")
    localisation = "uk"
    save_settings()

func _on_en_button_pressed() -> void:
    TranslationServer.set_locale("en")
    localisation = "en"
    save_settings()

#Слайдеры

func _on_brightness_slider_drag_ended(value_changed: bool) -> void:
    var s = brightness_slider.value
    brightness = 0.4 + 0.6 * s
    caption_bright_2.text = str(((brightness - 0.4)/(0.6))*100) + "%"
    save_settings()
    var canvas_modulate: CanvasModulate = $CanvasModulate
    canvas_modulate.color = Color(brightness, brightness, brightness)

func _on_music_sound_slider_drag_ended(value_changed: bool) -> void:
    if music_sound_slider.value == 0:
        music_volume = -80
        %Caption_music2.text = str(0) + "%"
    else:
        var percentage = music_sound_slider.value
        music_volume = -10 + (percentage * 10)
        %Caption_music2.text = str(percentage * 100) + "%"
    save_settings()
    music_player.set_music_volume(music_volume)

func _on_volumme_slider_drag_ended(value_changed: bool) -> void:
    if volumme_slider.value == 0:
        sound_volume = -80
        %Caption_sound2.text = str(0)+"%"
    else:
        var percent = volumme_slider.value
        sound_volume = -10 + (percent * 10)
        %Caption_sound2.text = str(percent*100)+"%"

```

```

        save_settings()
        GameManager.sfx_volume = sound_volume

func _on_hard_mode_pressed() -> void:
    if (hp_scale):
        GameManager.set_hp_scale(1)
        hp_scale = false
    else:
        GameManager.set_hp_scale(0.5)
        hp_scale = true
    save_settings()

func _on_less_gold_button_pressed() -> void:
    if (gold_scale):
        GameManager.set_gold_scale(1)
        gold_scale = false
    else:
        GameManager.set_gold_scale(0.5)
        gold_scale = true
    save_settings()

func _on_low_exp_scale_button_pressed() -> void:
    if (reduced_experience):
        GameManager.set_xp_scale(1)
        reduced_experience = false
    else:
        GameManager.set_xp_scale(0.5)
        reduced_experience = true
    save_settings()

```

### Лістинг A.11 – inventory\_slot.gd

```

extends Button

var slot_index: int
var inventory: Node
var weapon_data: WeaponData
var player: CharacterBody2D
signal inventory_changed
signal inventory_cleared

@onready var texture_rect: TextureRect = $VBoxContainer/TextureRect
@onready var name_label: Label = $VBoxContainer/Label
@onready var confirm_dialog: ConfirmationDialog = $ConfirmationDialog

func set_player(a: CharacterBody2D):
    player = a

func update_display(data: WeaponData, index: int, inv: Node):
    weapon_data = data
    slot_index = index
    inventory = inv
    if data == null:
        texture_rect.texture = null
        name_label.text = ""
    return

```

```

        texture_rect.texture = data.icon if data.icon else null
        name_label.text = data.weapon_name

func _ready() -> void:
    update_display(null, 0,null)
    update_display(null, 1,null)
    update_display(null, 2,null)
    update_display(null, 3,null)

func _on_confirmed():
    if inventory and inventory.has_method("remove_weapon"):
        inventory.remove_weapon(slot_index)
        player.stats.__coins += weapon_data.price * 0.8
        emit_signal("inventory_changed")
        texture_rect.texture = null
        name_label.text = ""
        weapon_data = null

func _on_pressed() -> void:
    if weapon_data != null:
        confirm_dialog.popup_centered()
        emit_signal("inventory_cleared")

func _on_confirmation_dialog_canceled() -> void:
    pass

```

## Лістинг А.12 – shop.gd

```

extends Control

signal shop_closed
signal save_progress

@onready var slot_1_buy: Button = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot1/Slot_1_buy"
@onready var slot_2_buy: Button = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot2/Slot2_buy"

@onready var texture_rect_1: TextureRect = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot1/TextureRect/TextureRect"
@onready var weapon_name_1: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot1/Slot_1_buy/HBoxContainer/Slot
1/HBoxContainer/WeaponName"
@onready var rarity_1: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot1/Slot_1_buy/HBoxContainer/Slot
1/HBoxContainer/Rarity"
@onready var price_1: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot1/Slot_1_buy/HBoxContainer/Slot
1/HBoxContainer/Price"
@onready var description_1: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot1/Slot_1_buy/HBoxContainer/Slot
1/VBoxContainer/Description"

```

```

@onready var texture_rect_2: TextureRect = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot2/TextureRect/TextureRect"
@onready var weapon_name_2: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot2/Slot2_buy/HBoxContainer/Slot
1/HBoxContainer/WeaponName"
@onready var rarity_2: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot2/Slot2_buy/HBoxContainer/Slot
1/HBoxContainer/Rarity"
@onready var price_2: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot2/Slot2_buy/HBoxContainer/Slot
1/HBoxContainer/Price"
@onready var description_2: Label = $"VBoxContainer/BuySlots and
Stats/Buy_slots/BuySlot2/Slot2_buy/HBoxContainer/Slot
1/VBoxContainer/Description"

@onready var player := get_tree().current_scene.get_node("Player")
@onready var inventory := player.get_node("InventoryUi")
@onready var player_stats := player.get_node("stats")

@onready var slot_1: Button = $"VBoxContainer/Inventory and
Buttons/Inventory/Inventory/Slot1"
@onready var slot_2: Button = $"VBoxContainer/Inventory and
Buttons/Inventory/Inventory/Slot2"
@onready var slot_3: Button = $"VBoxContainer/Inventory and
Buttons/Inventory/Inventory/Slot 3"
@onready var slot_4: Button = $"VBoxContainer/Inventory and
Buttons/Inventory/Inventory/Slot4"
@onready var label_error: Label = $"../Message/PanelContainer/Label"

var buy_slot_weapons: Array = []

func _ready():
    slot_1.set_player(player)
    slot_2.set_player(player)
    slot_3.set_player(player)
    slot_4.set_player(player)
    populate_buy_slots(-1)
    connect_buy_signals()
    merge_weapons()
    populate_inventory_ui()
    fill_stats()
    %Message.layer = 2
    connect("visibility_changed", Callable(self,
"_on_visibility_changed"))
    set_process(visible)

func _on_visibility_changed():
    set_process(visible)

func _show_message(a: String):
    label_error.text = a
    %Message.visible = true
    %Timer.start()

func _on_timer_timeout():
    %Message.visible = false

```



```

    %Timer.stop()

func _on_resume_pressed():
    emit_signal("shop_closed")

func _on_exit_pressed():
    get_tree().quit()

func _on_save_pressed():
    emit_signal("save_progress")

func connect_buy_signals():
    slot_1_buy.connect("pressed",
                        Callable(self,
"_on_buy_slot_pressed").bind(1))
    slot_2_buy.connect("pressed",
                        Callable(self,
"_on_buy_slot_pressed").bind(2))

func populate_buy_slots(slot_index := -1):
    var wave = GameManager.__current_wave+1
    if slot_index == -1 or slot_index == 1:
        var weapon_1 = WeaponDB.get_weapon_for_wave(wave)
        if buy_slot_weapons.size() >= 1:
            buy_slot_weapons[0] = weapon_1
        else:
            buy_slot_weapons.append(weapon_1)
            texture_rect_1.texture = weapon_1.icon
            weapon_name_1.text = weapon_1.weapon_name
            rarity_1.text = weapon_1.rarity
            price_1.text = str(weapon_1.price) + " " + tr("COINS")
            description_1.text = weapon_1.description
    if slot_index == -1 or slot_index == 2:
        var weapon_2 = WeaponDB.get_weapon_for_wave(wave)
        if buy_slot_weapons.size() >= 2:
            buy_slot_weapons[1] = weapon_2
        else:
            while buy_slot_weapons.size() < 1:
                buy_slot_weapons.append(null)
            buy_slot_weapons.append(weapon_2)
            texture_rect_2.texture = weapon_2.icon
            weapon_name_2.text = weapon_2.weapon_name
            rarity_2.text = weapon_2.rarity
            price_2.text = str(weapon_2.price) + " " + tr("COINS")
            description_2.text = weapon_2.description
    debug_print_inventory()

func _on_buy_slot_pressed(slot_index):
    if buy_slot_weapons.size() == 0:
        _show_message(tr("EMPTY_SLOT"))
        return
    var weapon = buy_slot_weapons[slot_index - 1]
    if player_stats.__coins < weapon.price:
        _show_message(tr("NO_MONEY"))
        return
    var inserted = false
    for i in range(4):
        if inventory.slots[i].get("weapon", null) == null:

```

```

        inventory.add_weapon_from_data(weapon, i)
        inserted = true
        break
    if not inserted:
        for i in range(4):
            var w_data = inventory.slots[i].get("weapon_data", null)
            if w_data and w_data.weapon_name == weapon.weapon_name and
w_data.tier == weapon.tier:
                var new_weapon = weapon.duplicate()
                new_weapon.tier += 1
                new_weapon.damage
calculate_upgraded_damage(new_weapon.weapon_name, new_weapon.tier,
new_weapon.rarity)
                new_weapon.price = calculate_upgraded_price(weapon.price,
new_weapon.tier, new_weapon.rarity)
                inventory.remove_weapon(i)
                inventory.remove_weapon(4)
                inventory.add_weapon_from_data(new_weapon, i)
                player_stats.__coins -= weapon.price
                populate_inventory_ui()
                fill_stats()
                debug_print_inventory()
                return
            _show_message(tr("FULL INVENTORY"))
        return
    player_stats.__coins -= weapon.price
    clear_shop_section(slot_index)
    populate_inventory_ui()
    fill_stats()

```

```

func clear_shop_section(slot_index):
    if slot_index == 1:
        texture_rect_1.texture = null
        weapon_name_1.text = ""
        rarity_1.text = ""
        price_1.text = ""
        description_1.text = ""
        if buy_slot_weapons.size() >= 1:
            buy_slot_weapons[0] = null
    elif slot_index == 2:
        texture_rect_2.texture = null
        weapon_name_2.text = ""
        rarity_2.text = ""
        price_2.text = ""
        description_2.text = ""
        if buy_slot_weapons.size() >= 2:
            buy_slot_weapons[1] = null
    populate_buy_slots(slot_index)

```

```

func merge_weapons():
    var ui_slots := [slot_1, slot_2, slot_3, slot_4]
    var merged = true
    while merged:
        merged = false
        var grouped = {}

```

```

for i in range(5):
    var data = inventory.slots[i]
    var weapon_data = data.get("weapon_data", null)
    if weapon_data != null:
        var key = "%s_%d" % [weapon_data.weapon_name, weapon_data.tier]
        if not grouped.has(key):
            grouped[key] = []
        grouped[key].append(i)
for key in grouped.keys():
    var indexes = grouped[key]
    while indexes.size() >= 2:
        var index1 = -1
        var index2 = -1
        if indexes.has(3):
            index1 = 3
            indexes.erase(3)
            index2 = indexes.pop_back()
        else:
            index1 = indexes.pop_back()
            index2 = indexes.pop_back()
        var w1 = inventory.slots[index1].weapon_data
        var new_weapon = w1.duplicate()
        new_weapon.tier += 1
        new_weapon.damage =
calculate_upgraded_damage(new_weapon.weapon_name, new_weapon.tier,
new_weapon.rarity)
        new_weapon.price = calculate_upgraded_price(w1.price,
new_weapon.tier, new_weapon.rarity)
        inventory.remove_weapon(index1)
        inventory.remove_weapon(index2)
        var target_index = index1 if index1 == 3 or
find_first_free_inventory_slot() == -1 else
find_first_free_inventory_slot()
        inventory.add_weapon_from_data(new_weapon, target_index)
        if target_index < 4:
            ui_slots[target_index].update_display(new_weapon,
target_index, inventory)
            apply_tier_color_to_slot(ui_slots[target_index],
new_weapon.tier)
            merged = true
        populate_inventory_ui()
        fill_stats()

func find_first_free_inventory_slot() -> int:
    for i in range(5):
        if inventory.slots[i].get("weapon", null) == null:
            return i
    return -1

func populate_inventory_ui():
    var ui_slots := [slot_1, slot_2, slot_3, slot_4]
    for i in range(4):
        var ui_slot = ui_slots[i]
        var slot_data = inventory.slots[i]

```

```

        var weapon_data = slot_data.get("weapon_data", null)
        if slot_data.weapon != null:
            ui_slot.update_display(weapon_data, i, inventory)
            apply_tier_color_to_slot(ui_slot, weapon_data.tier)
        else:
            ui_slot.update_display(null, i, inventory)
            apply_tier_color_to_slot(ui_slot, 0)
    fill_stats()

func fill_stats():
    $"VBoxContainer/BuySlots    and    Stats/Stats/Health/Value".text =
str(player_stats.__max_health)
    $"VBoxContainer/BuySlots    and    Stats/Stats/Damage/Value".text =
str(player_stats.__damage)
    $"VBoxContainer/BuySlots    and    Stats/Stats/AS/Value".text =
str(player_stats.__attack_speed)
    $"VBoxContainer/BuySlots    and    Stats/Stats/Crit_chance/Value".text =
str(player_stats.__crit_chance)
    $"VBoxContainer/BuySlots    and    Stats/Stats/Level/Value".text =
str(player_stats.__level)
    $"VBoxContainer/BuySlots    and    Stats/Stats/Gold/Value".text =
str(round(player_stats.__coins))
    $"VBoxContainer/BuySlots    and    Stats/Stats/Armor/Value".text =
str(player_stats.__armor)

func apply_tier_color_to_slot(slot_button: Button, tier: int):
    var tier_colors := {
        1: Color(1, 1, 1),
        2: Color(0.6, 1, 0.6),
        3: Color(0.6, 0.6, 1),
        4: Color(0.9, 0.6, 1),
        5: Color(1, 0.8, 0.2),
    }
    var clamped_tier = tier if tier <= 4 else 5
    slot_button.modulate = tier_colors.get(clamped_tier, Color(1, 1, 1))

func debug_print_inventory():
    print("=== Инвентарь игрока ===")
    for i in range(inventory.slots.size()):
        var data = inventory.slots[i]
        var weapon_data = data.get("weapon_data", null)
        if weapon_data != null:
            print("Слот %d: %s | Тип: %d | Цена: %d | Урон: %.2f | Раритет:
%s" % [
                i,
                weapon_data.weapon_name,
                weapon_data.tier,
                weapon_data.price,
                weapon_data.damage,
                weapon_data.rarity,
            ])
        else:
            print("Слот %d: пустой" % i)
    print("=====")

```

```

func calculate_upgraded_damage(name: String, tier: int, rarity: String) ->
float:
    var base_damage = 10.0
    var rarity_bonus = {
        "common": 1.0,
        "uncommon": 1.1,
        "rare": 1.2,
        "epic": 1.3,
        "legendary": 1.5
    }.get(rarity, 1.0)
    var wave_bonus = GameManager.__current_wave * 0.2
    var tier_multiplier = pow(1.1, tier)
    return base_damage * tier_multiplier * rarity_bonus + wave_bonus

func calculate_upgraded_price(base_price: int, tier: int, rarity: String)
-> int:
    var rarity_multiplier = {
        "common": 1.0,
        "uncommon": 1.2,
        "rare": 1.5,
        "epic": 1.8,
        "legendary": 2.2
    }.get(rarity, 1.0)
    var tier_multiplier = pow(1.3, tier)
    return int(base_price * tier_multiplier * rarity_multiplier)

func _process(delta: float) -> void:
    merge_weapons()
    populate_inventory_ui()

```

### Лістинг A.13 – back\_ground\_music.gd

```

extends AudioStreamPlayer2D

var music_volume: int

func _ready():
    var config = ConfigFile.new()
    if config.load("user://settings.cfg") == OK:
        music_volume = config.get_value("Settings", "m_volume", 0)
    else:
        music_volume = 0
    self.volume_db = music_volume
    play(GameManager.music_position)

func _process(delta: float) -> void:
    GameManager.music_position = self.get_playback_position()

func set_music_volume(volume: int):
    music_volume = volume
    self.volume_db = music_volume

```

### Лістинг A.14 – brightness\_filter.gd

```

extends CanvasModulate

```

```

const CONFIG_PATH = "user://settings.cfg"
@onready var canvas_modulate: CanvasModulate = $"."

func _ready():
    var config = ConfigFile.new()
    if config.load(CONFIG_PATH) == OK:
        var brightness = config.get_value("Settings", "brightness", 0)
        var brightness_value = brightness
        canvas_modulate.color = Color(brightness_value, brightness_value,
brightness_value)

func manual_analyse():
    var config = ConfigFile.new()
    if config.load(CONFIG_PATH) == OK:
        var brightness = config.get_value("Settings", "brightness", 0)
        var brightness_value = brightness
        canvas_modulate.color = Color(brightness_value, brightness_value,
brightness_value)

```

### Лістинг A.15 – hit\_box.gd

```

extends Area2D

@export var damage: float = 1.0

func _ready() -> void:
    pass

func get_damage() -> float:
    return damage

```

### Лістинг A.16 – hurt\_box.gd

```

extends Area2D

signal hit(damage)

@onready var damage_timer: Timer = Timer.new()

var attacker: Node2D = null

func _on_area_entered(area):
    if not area.is_in_group("attack"):
        return
    if not area.has_method("get_damage"):
        return
    if area.owner == owner:
        return
    if area.get_parent().is_in_group("enemies") and
get_parent().is_in_group("enemies"):
        return
    attacker = area
    if not attacker.has_method("get_damage"):
        return
    _apply_damage()
    damage_timer.start()

```

```

func _on_area_exited(area):
    if area == attacker:
        damage_timer.stop()
        attacker = null

func _apply_damage():
    if attacker and is_instance_valid(attacker):
        var damage = attacker.get_damage()
        emit_signal("hit", damage)
        if get_parent().has_method("take_damage"):
            get_parent().take_damage(damage)
        else:
            damage_timer.stop()

```

### Лістинг A.17 – inventory\_ui.gd

```

extends Node2D
class_name SimplifiedInventory

@onready var markers = [
    $Marker1,
    $Marker2,
    $Marker3,
    $Marker4
]

@export var slots_count: int = 5
var slots: Array = []
var current_slot: int = 0
var player_node = null

func _ready():
    _initialize_slots()
    _find_inventory_markers()

func _find_inventory_markers():
    for i in range(slots_count-1):
        if i < markers.size():
            print("Найден маркер для слота", i)
        else:
            print("Ошибка: маркер для слота", i, "не найден!")

func _initialize_slots():
    for i in range(slots_count):
        var slot = {}
        slot.weapon = null
        slots.append(slot)

func synchronize_weapon_orientation(is_facing_left: bool):
    for slot in slots:
        if slot.weapon != null:
            slot.weapon.set_flip(is_facing_left)

func remove_weapon(slot_index: int):
    if slot_index < 0 or slot_index >= slots_count:

```

```

        return
    if slots[slot_index].weapon:
        slots[slot_index].weapon.queue_free()
        slots[slot_index].weapon = null
    slots[slot_index].weapon_data = null

func select_slot(slot_index: int):
    if slot_index < 0 or slot_index >= slots_count:
        return
    current_slot = slot_index

func add_weapon(weapon_scene: PackedScene, slot_index: int) -> bool:
    if slot_index < 0 or slot_index >= slots_count:
        return false
    var new_weapon = weapon_scene.instantiate()
    if player_node and "set_player" in new_weapon:
        new_weapon.set_player(player_node)
    markers[slot_index].add_child(new_weapon)
    slots[slot_index].weapon = new_weapon
    return true

func add_weapon_from_data(data: WeaponData, slot_index: int) -> bool:
    if slot_index < 0 or slot_index >= slots_count-1:
        return false
    var new_weapon = data.weapon_scene.instantiate()
    if player_node and "set_player" in new_weapon:
        new_weapon.set_player(player_node)
    if "set_data" in new_weapon:
        new_weapon.set_data(data)
    markers[slot_index].add_child(new_weapon)
    slots[slot_index].weapon = new_weapon
    slots[slot_index].weapon_data = data
    return true

func _process(delta: float):
    for i in range(slots_count):
        var current_weapon = slots[i].weapon
        if current_weapon != null:
            if current_weapon.has_method("attack"):
                current_weapon.attack()

func get_inventory_data() -> Array:
    var inventory_data = []
    for i in range(min(slots.size(), 4)):
        var slot = slots[i]
        if slot.has("weapon_data") and slot.weapon_data:
            inventory_data.append({
                "name": slot.weapon_data.weapon_name,
                "tier": slot.weapon_data.tier,
                "damage": slot.weapon_data.damage
            })
        else:
            inventory_data.append(null)
    return inventory_data

func load_inventory_data(data: Array) -> void:

```



```

for i in range(min(data.size(), 4)):
    remove_weapon(i)
    var entry = data[i]
    if entry != null:
        var base_data = WeaponDB.get_by_name(entry.name)
        if base_data:
            var weapon = base_data.duplicate()
            weapon.tier = entry.tier
            weapon.damage = entry.damage
            add_weapon_from_data(weapon, i)

```

### Лістинг A.18 – bg.gd

```

extends ParallaxBackground

var SPEED = 100

func _process(delta):
    scroll_offset.x -= SPEED * delta

```

### Лістинг A.19 – PlayerStats.gd

```

class_name PlayerStats
extends Node

enum Stats { HEALTH, DAMAGE, ARMOR, CRIT_CHANCE, ATTACK_SPEED }

@export var __max_health: float = 5
@export var __current_health: float = 1
@export var __damage: float = 10
@export var __armor: int = 0
@export var __crit_chance: float = 0.01
@export var __crit_multiplier: float = 2.0
@export var __attack_speed: float = 1
@export var __current_experience: float = 0
@export var __experience_to_level_up: float = 4
@export var __level: int = 1
@export var __coins: float = 0

signal health_changed(current_health)
signal died
signal level_up(new_level)
signal show_buff_cards

var is_dead: bool = false
var buff_effects = {}

var base_max_health: float = 15
var base_gold: float = 0

func _ready():
    base_max_health = __max_health
    base_gold = __coins

    __max_health = base_max_health * GameManager.__hp_scale
    __current_health = __max_health

```

```

__coins = base_gold * GameManager.__gold_scale

buff_effects = {
    Stats.HEALTH: func(value): __max_health += value; __current_health
= __max_health,
    Stats.DAMAGE: func(value): __damage += value,
    Stats.ARMOR: func(value): __armor += int(value),
    Stats.CRIT_CHANCE: func(value): __crit_chance += value,
    Stats.ATTACK_SPEED: func(value): __attack_speed += value
}

func get_health() -> int:
    return (__current_health / __max_health) * 100

func get_exp() -> float:
    return (__current_experience / __experience_to_level_up) * 100

func get_lvl() -> int:
    return __level

func take_damage(amount: int):
    if is_dead:
        return
    var reduced_damage = max(amount - __armor, 1)
    __current_health -= reduced_damage
    __current_health = max(__current_health, 0)
    emit_signal("health_changed", __current_health)
    if __current_health <= 0 and not is_dead:
        is_dead = true
        emit_signal("died")
        die()

func get_coins() -> float:
    return __coins

func heal():
    if not is_dead:
        __current_health = __max_health
        emit_signal("health_changed", __current_health)

func gain_experience(amount: float):
    __current_experience += amount
    while __current_experience >= __experience_to_level_up:
        __current_experience -= __experience_to_level_up
        level_uped()

func level_uped():
    __level += 1
    var extra_xp = 3 if __level <= 3 else 3 + (__level - 3) / 3
    __experience_to_level_up += extra_xp * 2
    emit_signal("level_up", __level)
    emit_signal("show_buff_cards")
    print("Вітаємо з новим рівнем:", __level)

func calculate_damage() -> float:
    var is_critical = randf() < __crit_chance

```

```

        var final_damage = __damage * __crit_multiplier if is_critical else
__damage
        return final_damage

func apply_buff(buff_type: int, value: float) -> void:
    if buff_effects.has(buff_type):
        buff_effects[buff_type].call(value)

func die():
    print("Герой помер.")

func gain_coins(amount: float):
    base_gold+=amount
    __coins += amount * GameManager.__gold_scale

func print_stats() -> void:
    print("# Статистика гравця:")
    print("  - Здоров'я:", __current_health, "/", __max_health)
    print("  - Урон:", __damage)
    print("  - Броня:", __armor)
    print("  - Шанс критичного удару:", __crit_chance * 100, "%")
    print("  - Множник критичного удару:", __crit_multiplier, "x")
    print("  - Швидкість атаки:", __attack_speed)
    print("  - Досвід:", __current_experience, "/",
__experience_to_level_up, "(", str(get_exp()), "% )")
    print("  - Рівень:", __level)
    print("  - Монети:", __coins)

func to_dict() -> Dictionary:
    return {
        "max_health": base_max_health,
        "current_health": base_max_health, # __current_health,
        "damage": __damage,
        "armor": __armor,
        "crit_chance": __crit_chance,
        "crit_multiplier": __crit_multiplier,
        "attack_speed": __attack_speed,
        "current_experience": __current_experience,
        "experience_to_level_up": __experience_to_level_up,
        "level": __level,
        "coins": base_gold
    }

func from_dict(data: Dictionary) -> void:
    base_max_health = data.get("max_health", 15)
    base_gold = data.get("coins", 0)
    __max_health = base_max_health * GameManager.__hp_scale
    __current_health = data.get("current_health", __max_health)
    __current_experience = data.get("current_experience",
__current_experience)
    __coins = base_gold * GameManager.__gold_scale
    __damage = data.get("damage", 10)
    __armor = data.get("armor", 0)
    __crit_chance = data.get("crit_chance", 0.01)
    __crit_multiplier = data.get("crit_multiplier", 2.0)
    __attack_speed = data.get("attack_speed", 1)

```

```

__experience_to_level_up = data.get("experience_to_level_up", 15)
__level = data.get("level", 1)

func revive():
    is_dead = false
    if __current_health <= 0:
        __current_health = __max_health
    emit_signal("health_changed", __current_health)

func _default_stats():
    base_max_health = 15
    base_gold = 0
    __max_health = base_max_health * GameManager.__hp_scale
    __current_health = __max_health
    __damage = 10
    __armor = 0
    __crit_chance = 0.01
    __crit_multiplier = 2.0
    __attack_speed = 1
    __current_experience = 0
    __experience_to_level_up = 4
    __level = 1
    __coins = base_gold * GameManager.__gold_scale
    emit_signal("health_changed", __current_health)

```

## Лістинг A.20 – Spawner.gd

```

class_name Spawner_logic
extends Node2D

signal wave_finished
signal wave_started

const ENEMY_SCENES = {
    "skeleton":
preload("res://src/Enemies/Skeleton/skeleton_enemy.tscn"),
    "nightborne":
preload("res://src/Enemies/Nightborne/nightborne.tscn"),
    "reaper": preload("res://src/Enemies/Zhnec/zhec.tscn")
}

const ENEMY_PROBABILITIES = [
    {"wave": 0, "probabilities": {"skeleton": 1.0}},
    {"wave": 5, "probabilities": {"skeleton": 0.7, "nightborne": 0.3}},
    {"wave": 10, "probabilities": {"skeleton": 0.5, "nightborne": 0.35,
"reaper": 0.15}},
    {"wave": 15, "probabilities": {"skeleton": 0.4, "nightborne": 0.35,
"reaper": 0.25}},
    {"wave": 20, "probabilities": {"skeleton": 0.4, "nightborne": 0.3,
"reaper": 0.3}}
]

@onready var spawn_timer: Timer = $Area2D/SpawnTimer
@onready var spawn_area: CollisionShape2D = $Area2D/CollisionShape2D

```

```

@onready          var          label:          Label          =
$"..../UI/MarginContainer/VBoxContainer/HBoxContainer/Label"

@export var wave_delay: float = 5.0
@export var spawn_interval: float = 1.5
@export var max_enemies: int = 30

var current_wave: int = 0
var base_enemy_count: int = 4
var wave_multiplier: float = 1.2
var enemies_to_spawn: int = 0
var spawned_enemies: Array = []
var shop_instance

func _ready():
    spawn_timer.wait_time = spawn_interval
    spawn_timer.timeout.connect(_on_SpawnTimer_timeout)

func start_wave():
    GameManager.__current_wave = current_wave
    for enemy in spawned_enemies:
        if is_instance_valid(enemy):
            enemy.queue_free()
    spawned_enemies.clear()
    current_wave += 1
    enemies_to_spawn = int(base_enemy_count * pow(wave_multiplier,
current_wave))
    spawn_timer.start()
    emit_signal("wave_started")

func _on_SpawnTimer_timeout():
    if enemies_to_spawn > 0:
        spawn_enemy()
        enemies_to_spawn -= 1
    elif spawned_enemies.is_empty():
        spawn_timer.stop()

func spawn_enemy():
    if spawned_enemies.size() >= max_enemies:
        return

    var enemy_type = choose_enemy_type()
    var enemy_scene = get_enemy_scene(enemy_type)
    var enemy = enemy_scene.instantiate()

    enemy.position = get_random_point_in_area()
    spawned_enemies.append(enemy)
    get_tree().current_scene.add_child(enemy)

    if enemy.has_signal("died"):
        enemy.connect("died",
Callable(self,
"_on_enemy_died").bind(enemy))

    if enemy.has_method("apply_wave_modifiers"):

```

```

        enemy.apply_wave_modifiers(current_wave)

func _on_enemy_died(enemy):
    spawned_enemies.erase(enemy)

    if enemies_to_spawn == 0 and spawned_enemies.is_empty():
        wave_finished.emit()

func choose_enemy_type() -> String:
    var probabilities = ENEMY_PROBABILITIES[0]["probabilities"]
    for entry in ENEMY_PROBABILITIES:
        if current_wave >= entry["wave"]:
            probabilities = entry["probabilities"]
        else:
            break

    var rand = randf()
    var cumulative = 0.0
    for enemy_type in probabilities.keys():
        cumulative += probabilities[enemy_type]
        if rand <= cumulative:
            return enemy_type
    return "skeleton"

func get_enemy_scene(enemy_type: String) -> PackedScene:
    return ENEMY_SCENES.get(enemy_type, ENEMY_SCENES["skeleton"])

func get_random_point_in_area() -> Vector2:
    var rect = spawn_area.shape.get_rect()
    var x = randf_range(rect.position.x, rect.position.x + rect.size.x)
    var y = randf_range(rect.position.y, rect.position.y + rect.size.y)
    return Vector2(x, y) + spawn_area.global_position

func get_current_wave() -> int:
    return current_wave

func set_current_wave(a: int) -> void:
    current_wave = a
    start_wave()

```

### Лістинг A.21 – bullet.gd

```

class_name Bullet
extends Node2D

@export var speed: float = 300.0
@export var direction: Vector2 = Vector2.RIGHT
@export var damage: float = 10.0
var __is_in_zone: bool = false
var __enemy_in_attack_range: EnemyBase = null

@onready var sprite: AnimatedSprite2D = $AnimatedSprite2D

func _ready():
    sprite.flip_h = direction.x < 0

```

```

func set_direction(new_direction: Vector2):
    direction = new_direction
    if sprite:
        sprite.flip_h = direction.x < 0

func _process(delta):
    position += direction * speed * delta

func __enemy_entered_attack_range(area: Area2D) -> void:
    var enemy = area.get_parent()
    if enemy is EnemyBase:
        __is_in_zone = true
        __enemy_in_attack_range = enemy
        __enemy_in_attack_range.take_damage(damage)
        queue_free()

func __enemy_exited_attack_range(area: Area2D) -> void:
    if __enemy_in_attack_range and __enemy_in_attack_range ==
area.get_parent():
        __enemy_in_attack_range = null
        __is_in_zone = false

```

## Лістинг A.22 – magik\_projectile.gd

```

class_name MagikHit
extends Node2D

@export var speed: float = 300.0
@export var direction: Vector2 = Vector2.RIGHT
@export var damage: float = 10.0
var __is_in_zone: bool = false
var __enemy_in_attack_range: EnemyBase = null

@onready var sprite: AnimatedSprite2D = $AnimatedSprite2D

func set_direction(new_direction: Vector2):
    direction = new_direction
    if sprite:
        sprite.flip_h = direction.x < 0

func _process(delta):
    position += direction * speed * delta

func __enemy_entered_attack_range(area: Area2D) -> void:
    var enemy = area.get_parent()
    if enemy is EnemyBase:
        __is_in_zone = true
        __enemy_in_attack_range = enemy
        __enemy_in_attack_range.take_damage(damage)
        queue_free()

func __enemy_exited_attack_range(area: Area2D) -> void:
    if __enemy_in_attack_range and __enemy_in_attack_range ==
area.get_parent():
        __enemy_in_attack_range = null
        __is_in_zone = false

```

## Лістинг А.23 – magik\_stick.gd

```
class_name MagicStick
extends Weapon

@onready var __stick_sprite: AnimatedSprite2D = $StickSprite
var bullet_scene =
load("res://src/Weapons/Magic_projectile/Magik_projectile.tscn")
@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D
var music_volume: int
var damage: float = 0

func _ready() -> void:
    if __stick_sprite == null:
        __stick_sprite = $StickSprite
        self.connect("attack_started", Callable(self,
"_on_attack_started"))
        __stick_sprite.connect("animation_finished", Callable(self,
"_on_animated_sprite_2d_animation_finished"))
    var config = ConfigFile.new()
    if config.load("user://settings.cfg") == OK:
        music_volume = config.get_value("Settings", "s_volume", 40)
    else:
        music_volume = 0
    sfx.volume_db = music_volume

func attack():
    __stick_sprite.play("attack")

func _process(delta: float) -> void:
    __stick_sprite.flip_h = super.get_flip()

func _on_attack_started():
    pass

func _on_animated_sprite_2d_animation_finished():
    var bullet = bullet_scene.instantiate()
    self.add_child(bullet)
    bullet.damage = self.damage
    bullet.global_position = self.global_position + Vector2(0, 0)
    var is_facing_left = __stick_sprite.flip_h
    bullet.set_direction(Vector2.LEFT if is_facing_left else
Vector2.RIGHT)
    play_effect("res://src/Weapons/Magic_stick/Magic_attack.wav")
    attack_completed()

func play_effect(effect_path: String) -> void:
    var effect_stream = load(effect_path)
    if effect_stream:
        sfx.stop()
        sfx.stream = effect_stream
        sfx.play()
    else:
        push_error("Не вдалося завантажити ефект: " + effect_path)
```



```

func set_data(data: WeaponData):
    if not __stick_sprite:
        __stick_sprite = $StickSprite
    if data.sprite_frames:
        __stick_sprite.sprite_frames = data.sprite_frames
    self.damage = data.damage
    __stick_sprite.play("attack")

```

### Лістинг А.23 – pistol.gd

```

class_name Pistol
extends Weapon

var __gun_sprite: AnimatedSprite2D = null
var bullet_scene = load("res://src/Weapons/bullet/bullet.tscn")
var damage: float = 0
@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D
var music_volume: int

func _ready() -> void:
    if __gun_sprite == null:
        __gun_sprite = $GunSprite
        self.connect("attack_started", Callable(self,
        "_on_attack_started"))
        __gun_sprite.connect("animation_finished", Callable(self,
        "_on_animated_sprite_2d_animation_finished"))
    var config = ConfigFile.new()
    if config.load("user://settings.cfg") == OK:
        music_volume = config.get_value("Settings", "s_volume", -50)
    else:
        music_volume = 0
    sfx.volume_db = music_volume

func attack():
    __gun_sprite.play("attack")

func _process(delta: float) -> void:
    __gun_sprite.flip_h = super.get_flip()

func _on_attack_started():
    pass

func _on_animated_sprite_2d_animation_finished():
    var bullet = bullet_scene.instantiate()
    bullet.damage = self.damage
    self.add_child(bullet)
    bullet.global_position = self.global_position + Vector2(0, +10)
    var is_facing_left = __gun_sprite.flip_h
    bullet.set_direction(Vector2.LEFT if is_facing_left else
    Vector2.RIGHT)
    play_effect("res://src/Weapons/Pistol/pistol_shot.wav")
    attack_completed()

func play_effect(effect_path: String) -> void:
    var effect_stream = load(effect_path)
    if effect_stream:

```

```

    sfx.stop()
    sfx.stream = effect_stream
    sfx.play()
else:
    push_error("Не вдалося завантажити ефект: " + effect_path)

```

```

func set_data(data: WeaponData):
    if not __gun_sprite:
        __gun_sprite = $GunSprite
    if data.sprite_frames:
        __gun_sprite.sprite_frames = data.sprite_frames
        __gun_sprite.play("attack")
    self.damage = data.damage

```

## Лістинг A.24 – sword.gd

```

class_name Sword
extends Weapon

@onready var hitbox: Area2D = $Sword_sprite/HitBox

var sword_sprite: AnimatedSprite2D = null
var __is_in_zone: bool = false
var __enemy_in_attack_range: EnemyBase = null
var __facing_left: bool = false
var last_flip: bool = false
@export var damage_sword = 150
@onready var sfx: AudioStreamPlayer2D = $AudioStreamPlayer2D
var music_volume: int

func _ready() -> void:
    if sword_sprite == null:
        sword_sprite = $Sword_sprite
    var config = ConfigFile.new()
    if config.load("user://settings.cfg") == OK:
        music_volume = config.get_value("Settings", "s_volume", 40)
    else:
        music_volume = 0
    sfx.volume_db = music_volume
    __update_hitbox_position()

func attack():
    __face_sword()

    sword_sprite.play("attack")

func _process(delta: float) -> void:
    __facing_left = sword_sprite.flip_h
    sword_sprite.flip_h = super.get_flip()
    if not __attacking and __enemy_in_attack_range:
        perform_attack()

func __face_sword() -> void:
    var new_flip: bool = __facing_left
    if last_flip != new_flip:

```

```

    __update_hitbox_position()
    last_flip = new_flip

func __update_hitbox_position() -> void:
    var flip_value = 1 if __facing_left else -1
    hitbox.scale.x = abs(hitbox.scale.x) * flip_value
    if not __facing_left:
        hitbox.position.x = (sword_sprite.position.x)
    else:
        hitbox.position.x = (sword_sprite.position.x)+10

func perform_attack():
    if __enemy_in_attack_range:
        __attacking = true
        play_effect("res://src/Weapons/Sword/Sword_attack.wav")
        sword_sprite.play("attack")
        await sword_sprite.animation_finished
        __attacking = false
        if __enemy_in_attack_range != null:
            __enemy_in_attack_range.take_damage(damage_sword)
        else:
            print("# Ворот відсутній.")

func __enemy_entered_attack_range(area: Area2D) -> void:
    var enemy = area.get_parent()
    if enemy is EnemyBase:
        __is_in_zone = true
        __enemy_in_attack_range = enemy

func __enemy_exited_attack_range(area: Area2D) -> void:
    if __enemy_in_attack_range and __enemy_in_attack_range ==
area.get_parent():
        __enemy_in_attack_range = null
        __is_in_zone = false

func play_effect(effect_path: String) -> void:
    var effect_stream = load(effect_path)
    if effect_stream:
        sfx.stop()
        sfx.stream = effect_stream
        sfx.play()
    else:
        push_error("Не вдалося завантажити ефект: " + effect_path)

func set_data(data: WeaponData):
    if not sword_sprite:
        sword_sprite = $Sword_sprite
    if data.sprite_frames:
        sword_sprite.sprite_frames = data.sprite_frames
        damage_sword = data.damage
        sword_sprite.play("attack")

```

### Лістинг A.25 – WeaponData.gd

```

extends Resource
class_name WeaponData

```

```

@export var weapon_name: String
@export var description: String
@export var damage: float
@export var tier: int
@export var price: float
@export var color: Color
@export var icon: Texture2D
@export var weapon_scene: PackedScene
@export var rarity: String
@export var weapon_type: String
@export var sprite_frames: SpriteFrames

```

### Лістинг A.26 – weapon\_base.gd

```

class_name Weapon
extends Node2D

@export var damage_multiplier: float = 1.0

var __attacking: bool = false
var __flip_hei: bool = false
var attack_speed: float = 1.0
var stats = null
var player_ref = null

signal attack_started
signal attack_finished

func set_flip(h: bool) -> void:
    __flip_hei = h

func get_flip() -> bool:
    return __flip_hei

func _ready():
    pass
    #var player = get_tree().get_first_node_in_group("Player")
    #if player == null:
    #    print("Ошибка: Игрок не найден!")
    #    return
    #stats = player.get("stats")
    #if stats == null:
    #    print("Ошибка: Статистика игрока не найдена!")
    #    return
    #var attack_speed_value = stats.get("__attack_speed")
    #if attack_speed_value == null:
    #    print("Ошибка: Скорость атаки не найдена!")
    #    return
    #attack_speed = 1.0 / attack_speed_value

func set_player(p: Node):
    player_ref = p
    if player_ref == null:
        print("Ошибка: Игрок не установлен!")
        return

```

```

stats = player_ref.get("stats")
if stats == null:
    print("Ошибка: Статистика игрока не найдена!")
    return
var attack_speed_value = stats.get("__attack_speed")
if attack_speed_value == null:
    print("Ошибка: Скорость атаки не найдена!")
    return
attack_speed = 1.0 / float(attack_speed_value)

func attack():
    if __attacking:
        return
    __attacking = true
    attack_started.emit()
    await get_tree().create_timer(attack_speed).timeout
    attack_completed()

func attack_completed():
    __attacking = false
    attack_finished.emit()

```

### Лістинг A.27 – GameManager.gd

```

extends Node

const SAVE_PATH_TEMPLATE := "user://save_slot_%d.json"

var __xp_scale: float
var __gold_scale: float
var __hp_scale: float
var music_position: float = 0.0
var sfx_volume: float = 0.0
const CONFIG_PATH = "user://settings.cfg"
var __current_wave:int = 0
var save_next_wave:bool = false

func _ready() -> void:
    __load_from_config()

func __load_from_config() -> void:
    var config = ConfigFile.new()
    if config.load(CONFIG_PATH) == OK:
        if config.get_value("Settings", "hp_scale", false):
            __hp_scale = 0.5
        else:
            __hp_scale = 1
        if config.get_value("Settings", "gold_scale", false):
            __gold_scale = 0.5
        else:
            __gold_scale = 1
        if config.get_value("Settings", "reduced_experience", false):
            __xp_scale = 0.5
        else:
            __xp_scale = 1

```

```

func set_xp_scale(a: float):
    __xp_scale = a

func set_gold_scale(a: float):
    __gold_scale = a

func set_hp_scale(a: float):
    __hp_scale = a

func save_game(slot: int, stats: PlayerStats, inventory:
SimplifiedInventory, spawner: Spawner_logic) -> void:
    var wave = spawner.get_current_wave()
    if save_next_wave:
        wave += 1
    var save_data = {
        "wave": wave,
        "player_stats": stats.to_dict(),
        "inventory": inventory.get_inventory_data()
    }
    var path = SAVE_PATH_TEMPLATE % slot
    var file = FileAccess.open(path, FileAccess.WRITE)
    file.store_string(JSON.stringify(save_data, "\t"))
    file.close()
    print("Игра сохранена в слот", slot, "на волне", save_data.wave)

func load_game(slot: int) -> Dictionary:
    var path = SAVE_PATH_TEMPLATE % slot
    if not FileAccess.file_exists(path):
        push_warning("Файл сейва не найден!")
        return {}
    var file = FileAccess.open(path, FileAccess.READ)
    var data = JSON.parse_string(file.get_as_text())
    file.close()
    print("Игра загружена из слота", slot, "волна:", data.get("wave",
"?"))
    return data

func delete_save(slot: int) -> void:
    var path = SAVE_PATH_TEMPLATE % slot
    if FileAccess.file_exists(path):
        DirAccess.remove_absolute(path)
        print("Сейв слот %d удалён" % slot)

func get_save_summary(slot: int) -> Dictionary:
    var path = SAVE_PATH_TEMPLATE % slot
    if not FileAccess.file_exists(path):
        return {"exists": false}
    var file = FileAccess.open(path, FileAccess.READ)
    var data = JSON.parse_string(file.get_as_text())
    file.close()
    return {
        "exists": true,
        "wave": data.get("wave", 0),
        "level": data.get("player_stats", {}).get("level", 1),

```

```

        "coins": data.get("player_stats", {}).get("coins", 0)
    }

```

## Лістинг A.28 – WeaponDatabase.gd

```

extends Node
class_name WeaponDatabase
var weapons: Array = []
var wave_config = [
    {
        "min_wave": 1,
        "max_wave": 14,
        "distribution": [
            ["COMMON", 1.0]
        ]
    },
    {
        "min_wave": 15,
        "max_wave": 24,
        "distribution": [
            ["COMMON", 0.7],
            ["RARE", 0.3]
        ]
    },
    {
        "min_wave": 25,
        "max_wave": 34,
        "distribution": [
            ["RARE", 0.7],
            ["EPIC", 0.3]
        ]
    },
    {
        "min_wave": 35,
        "max_wave": 9999,
        "distribution": [
            ["EPIC", 0.7],
            ["LEGENDARY", 0.3]
        ]
    }
]

func _ready():
    weapons = load_all_weapon_data("res://src/Weapons/Data")
    print("WeaponDatabase загрузжено орудий:", weapons.size())

func load_all_weapon_data(path: String) -> Array:
    var result = []
    var dir = DirAccess.open(path)
    if dir == null:
        push_error("Не удалось открыть папку: " + path)
        return result
    dir.list_dir_begin()
    var file_name = dir.get_next()
    while file_name != "":
        if file_name.begins_with("."):

```

```

        file_name = dir.get_next()
        continue
    var full_path = "%s/%s" % [path, file_name]
    if dir.current_is_dir():
        result += load_all_weapon_data(full_path)
    else:
        var res: Resource = null
        if file_name.ends_with(".remap"):
            var tres_path = full_path.substr(0, full_path.length() -
".remap".length())
            res = ResourceLoader.load(tres_path)
            if not res:
                push_error("Не удалось загрузить remapped ресурс: " +
tres_path)
            else:
                res = ResourceLoader.load(full_path)
            if res and res is WeaponData:
                result.append(res)
            elif res:
                push_warning("Загружен ресурс, но не WeaponData: " + full_path)
        file_name = dir.get_next()
    dir.list_dir_end()
    return result

func get_distribution_for_wave(wave: int) -> Array:
    for segment in wave_config:
        if wave >= segment["min_wave"] and wave <= segment["max_wave"]:
            return segment["distribution"]
    return []

func pick_rarity_for_wave(wave: int) -> String:
    var dist = get_distribution_for_wave(wave)
    if dist.size() == 0:
        return "COMMON"
    var roll = randf()
    var cumulative = 0.0
    for item in dist:
        var r = item[0]
        var chance = item[1]
        cumulative += chance
        if roll <= cumulative:
            return r
    return dist[dist.size() - 1][0]

func get_by_rarity(rarity: String) -> Array:
    return weapons.filter(func(w):
        return w.rarity == rarity
    )

func get_weapon_for_wave(wave: int) -> WeaponData:
    var chosen_rarity = pick_rarity_for_wave(wave)
    var arr = get_by_rarity(chosen_rarity)
    if arr.size() == 0:
        return null
    return arr[randi() % arr.size()]

```



```

static func get_weapon_by_name_and_tier(name: String, tier: int) ->
Resource:
    for w in WeaponDB.weapons:
        if w.weapon_name == name and w.tier == tier:
            return w
    print("НЕ НАЙДЕНО: %s tier %d" % [name, tier])
    return null

static func get_by_name(name: String) -> Resource:
    for w in WeaponDB.weapons:
        if w.weapon_name == name:
            return w
    print("НЕ НАЙДЕНО: %s" % [name])
    return null

```