СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування_____
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь ____бакалавр_____
(бакалавр, магістр)
спеціальність 126 «Інформаційні системи та технології»____
(шифр і назва спеціальності)

на тему "Розробка 2D-платформерної гри для мобільних пристроїв на базі Android

Виконав: студент групи ІСТ-21д _____ Є.Д Немченко
(підпис) (ініціали і прізвище)

Керівник _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О.І. Захожай
(підпис) (ініціали і прізвище)

Рецензент _____О.І. Захожай _____

Київ – 2025

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
(повна назва кафедри)

Освітній ступінь бакалавр
(бакалавр, магістр)

спеціальність 126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Захожай О.І.
"___" _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Немченко Євген Денисович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка 2D-платформерної гри для мобільних пристроїв на базі Android

керівник роботи Лифар Володимир Олексійович Доц., д.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від "_27_" _травня_2025 року №67/15.15-С

2. Строк подання студентом роботи 16.06.2025р.
3. Вихідні дані до роботи: Об'єктом даної роботи є процес. Розробка 2D-платформерної гри

4.Зміст розробка 2D-платформерної гри для мобільних для мобільних пристороїв на базі Android з використанням ігрового рушія Godot 4. Мета проекту — творити повноцінний ігровий додаток, який поєднує динамічний геймплей, просту візуальну стилістику та інтуїтивне сенсорне керування. Висновки. Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01.04.2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	01.04.25	виконано
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.25	виконано
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	16.04.25	виконано
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	26.04.25	виконано
5	Укладання та тестування програмного продукту	26.05.25	виконано
6	Укладання, оформлення та погодження пояснювальної записки з керівником	14.06.25	виконано
7	Надання готової пояснювальної записки на кафедру	16.06.25	виконано
8	Укладання доповіді і презентації	17.06.25	виконано

Студент _____ Є.Д. Немченко
підпис (ініціали і прізвище)

Керівник роботи _____ В.О. Лифар
підпис (ініціали і прізвище)

РЕФЕРАТ

Робота містить: 63 сторінки основного тексту, 12 сторінок додатків, 42 рисунки, 31 використане джерело.

Метою кваліфікаційної випускної роботи є розробка двовимірної платформер-гри для Android-пристроїв з використанням рушія Godot Engine 4.0. Основна ідея полягає у створенні повноцінного мобільного застосунку з сенсорним керуванням, експортом у форматі APK та можливістю подальшого бета-тестування.

У ході дослідження проаналізовано особливості жанру платформерів, обрано відповідні інструменти для розробки, а також розглянуто існуючі ігри-аналоги та визначено ключові механіки, які доцільно реалізувати. Розроблено архітектуру гри, структуру проєкту, сценарії взаємодії користувача з ігровими об’єктами, та інтерфейс мобільного керування.

Реалізовано функціональний ігровий прототип, що містить декілька рівнів, головне меню, систему паузи, екран перемоги й поразки, а також сенсорний джойстик для Android. Проведено тестування гри, виявлено потенційні недоліки та запропоновано способи їх усунення. Гра відповідає технічному завданню, має практичне застосування в освітніх або демонстраційних цілях.

Ключові слова: Godot, платформер, Android, 2D-гра, сенсорне керування, APK, GDScript, розробка ігор.

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій, зокрема ігрової індустрії, відкрив нові можливості як для професійних розробників, так і для ентузіастів, що прагнуть створювати власні ігрові проєкти. Ігри стали невіддільною частиною цифрової культури, вони поєднують у собі технічну інженерію, художній дизайн та емоційний вплив на користувача. Особливу популярність отримали 2D-платформери — ігри, у яких головна увага зосереджена на управлінні персонажем, пересуванні по рівнях із перешкодами та досягненні певної мети. Саме завдяки простоті реалізації та широким можливостям для творчості цей жанр став чудовою платформою для вивчення основ розробки ігор.

У рамках дипломного проєкту було поставлено завдання — створити функціональну 2D-платформер гру із сенсорним управлінням для Android-пристроїв з використанням сучасного рушія Godot Engine 4.0. Розробка гри дозволяє поєднати набуті під час навчання знання з програмування, системного аналізу, дизайну інтерфейсів та оптимізації програмного забезпечення. У процесі реалізації проєкту було враховано особливості мобільної платформи, адаптовано керування під сенсорні екрани та реалізовано ключові компоненти геймплею.

Результатом виконання проєкту є повноцінна Android-гра, яка може бути використана як демонстраційний приклад, відправлена на бета-тестування або подальше вдосконалення. Дипломна робота демонструє практичні навички в розробці інтерактивних застосунків, дотримання сучасних вимог до зручності та доступності програмного продукту, а також здатність до вирішення комплексних завдань у сфері інформаційних систем і технологій.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Жанрова приналежність гри

Комп'ютерні ігри розподіляються на велику кількість жанрів, серед яких окреме місце займають платформери. Цей жанр зосереджений на переміщенні персонажа по рівнях, що складаються з платформ, перешкод та ворогів. Гравець повинен точно контролювати стрибки, уникати пасток і долати логічні або фізичні виклики. Історично платформери отримали популярність з виходом таких ігор, як Super Mario Bros, Sonic the Hedgehog та інші. Сьогодні платформери існують як у класичній 2D-формі, так і в сучасних 3D-варіаціях, часто поєднуючи елементи пригод, головоломок і бойових механік.

1.2 Інструменти розробки

Для створення гри використано ігровий рушій Godot Engine версії 4.0, який є потужним інструментом для розробки 2D та 3D ігор з відкритим кодом. Godot підтримує візуальне середовище для створення сцен, а також власну мову скриптів — GDScript, що схожа на Python. Рушій дозволяє легко експортувати проєкт на різні платформи, включаючи Android, Windows, Linux та Web. Серед інших інструментів — графічні редактори для створення спрайтів, текстових файлів та налаштувань анімацій.

1.3 Аналіз існуючих аналогів

Серед найбільш популярних 2D-платформерів, які стали прикладом для наслідування, варто згадати:

– Celeste — складна, але надихаюча гра, яка поєднує платформінг з глибоким сюжетом;

– Hollow Knight — гра з відкритим світом, складними босами та глибокою атмосферою;

– Ori and the Blind Forest — емоційна історія з красивим візуалом і плавним управлінням.

Аналіз цих проєктів дозволив виявити ключові елементи успішного платформера: точне управління, зрозумілий інтерфейс, прогресію рівнів, унікальні візуальні стилі та атмосферу.

1.4 Постановка цілей

Основною метою дипломного проєкту є розробка повноцінної 2D-платформер гри для платформи Android із сенсорним керуванням, реалізованої за допомогою рушія Godot. Серед конкретних цілей можна виділити:

– створення ігрового процесу зі змінними рівнями;

– реалізація системи керування для мобільних пристроїв;

– оптимізація під Android;

– створення головного меню, екрану паузи, перемоги та поразки;

– проведення тестування гри та підготовка APK-файлу для бета-тесту.

Проєкт також має на меті продемонструвати практичні навички розробки ігор, планування, написання скриптів та дизайну рівнів.

РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ

2.1 Визначення функціональних вимог

Функціональні вимоги до програмного продукту визначають основні дії, які повинна виконувати система, та її очікувану поведінку в різних ситуаціях. У випадку розробки 2D-платформер гри для Android ключовими функціональними вимогами є:

– забезпечення базового геймплею: переміщення гравця, стрибки, колізії з платформами;

– підтримка сенсорного керування, адаптованого до мобільних пристроїв;

– реалізація інтерфейсу користувача (меню, кнопки, HUD);

– наявність декількох рівнів з поступовим ускладненням;

– реалізація логіки поразки, перемоги та переходу між сценами;

– оптимізація продуктивності гри на середньостатистичних Android-пристроях;

– можливість експорту проєкту у формат APK для тестування та публікації.

2.2 Перелік задач розробки

Для реалізації вказаних функціональних вимог необхідно виконати низку задач, що охоплюють як технічні, так і дизайнерські аспекти розробки гри. Основні задачі дипломного проєкту можна сформулювати так:

1. Провести аналіз ігрового жанру платформерів, обрати механіки та концепцію гри.
2. Створити прототип проєкту в Godot Engine з базовою сценою.
3. Реалізувати скрипти управління персонажем, враховуючи стрибки, падіння, колізії.
4. Розробити декілька рівнів гри з елементами декорацій, платформ, ворогів (опціонально).
5. Створити головне меню, меню паузи та переходу між рівнями.
6. Розробити сенсорне керування: віртуальний джойстик, кнопки стрибка та дій.
7. Здійснити оптимізацію гри для мобільних пристроїв.
8. Зібрати проєкт у формат APK, протестувати на Android.
9. Підготувати графічні ресурси, шрифти, фони та аудіоефекти (при необхідності).
10. Задокументувати процес розробки у вигляді пояснювальної записки.

РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Архітектура гри

Архітектура гри визначає логічну структуру її складових, взаємозв'язки між ними, а також спосіб обробки подій та станів. У проєкті реалізовано ієрархічну архітектуру з використанням шаблонів, таких як розділення відповідальностей та повторне використання сцен. Основними архітектурними компонентами гри є: сцени (рівні), гравець, інтерфейс, обробник подій, логіка перемоги/поразки, сенсорне керування, а також менеджер переходів.

Кожна сцена (Scene) в Godot виступає як самостійний логічний блок. Сцена рівня, наприклад, включає гравця, платформу (TileMap), камеру, елементи UI та логіку переходу між рівнями. Гравець реалізований як окрема сцена з фізичним тілом KinematicBody2D, обробником вводу, анімацією та колізіями. Логіка управління зосереджена в окремих скриптах, що дозволяє зберігати чистоту архітектури та забезпечує повторне використання коду.

Інтерфейс користувача реалізовано у вигляді окремих сцен, які підключаються до основної сцени через вузли-контейнеры (CanvasLayer, Control). Це дозволяє відображати елементи UI незалежно від ігрової камери, зберігаючи зручність керування. Меню гри, перемоги, паузи та поразки також реалізовані окремими сценами, що підвантажуються за допомогою функцій переходів або сигналів.

3.2 Структура проєкту в Godot

Структура файлів і папок у проєкті Godot організована таким чином, щоб забезпечити легкість у навігації, логічне групування об'єктів та зручність підтримки коду. Основні каталоги та їх призначення:

– scenes/ — містить ключові сцени: головне меню, рівні, меню паузи, сцени результатів;

– player/ — окремі ресурси гравця, включаючи скрипти (Player.gd), сцени (Player.tscn), анімації та колізії;

– ui/ — інтерфейсні елементи: кнопки, HUD (панель здоров'я, лічильники балів), меню налаштувань;

– scripts/ — загальні скрипти, у тому числі для завантаження сцен, логіки гри, обробки подій;

– levels/ — сцени кожного рівня (Level_01.tscn, Level_02.tscn тощо), які мають спільну структуру;

– assets/ — спрайти, плитки, шрифти, фонові зображення, аудіо.

Така структура спрощує командну роботу над проєктом, пришвидшує

навігацію в редакторі та дозволяє швидко оновлювати конкретні частини гри без впливу на інші модулі.

3.3 Сцени, вузли та їх взаємодія

У Godot кожен елемент гри представлений у вигляді вузла (Node). Вузли формують ієрархічну структуру, де батьківський вузол може мати необмежену кількість дочірніх. Наприклад, сцена Player має такі вузли: KinematicBody2D (фізична основа), Sprite/AnimatedSprite (графічне представлення), CollisionShape2D (колізія), Camera2D (слідує за гравцем), Timer (затримки) та інші. Об’єднання вузлів у сцени дозволяє створювати складні поведінки через комбінації простих компонентів.
Взаємодія між об’єктами реалізується через механізм сигналів. Наприклад, після торкання об’єкта 'фінішу' на рівні генерується сигнал, який обробляється менеджером гри для завершення рівня. Сигнали також використовуються для реалізації паузи, відображення меню, запуску анімацій тощо. Завдяки модульному підходу та використанню вузлів, додавання нових рівнів, ворогів або функціоналу вимагає мінімальних змін в основному коді.

РОЗДІЛ 4. РОЗРОБКА ГРИ

4.1 Створення головного меню

Головне меню реалізовано у вигляді окремої сцени MainMenu.tscn. Основним вузлом є Control, до якого додаються VBoxContainer з кнопками: 'Почати гру', 'Продовжити', 'Вийти'. Кожна кнопка має підключений сигнал 'pressed', який викликає відповідну функцію у скрипті.
Наприклад, при натисканні кнопки 'Почати гру' викликається наступна функція:

```
func _on_start_button_pressed():
    get_tree().change_scene_to_file("res://scenes/Level_01.tscn")
```

Інтерфейс адаптовано під сенсорне керування: кнопки мають достатній розмір і розташовані на відстані для зручності на мобільному екрані.

4.2 Реалізація персонажа

4.2.1 Рух гравця

Гравець реалізований у вигляді сцени Player.tscn з вузлом KinematicBody2D. До нього додаються дочірні вузли: Sprite або AnimatedSprite для візуалізації, CollisionShape2D для обробки зіткнень, Camera2D для слідкування за гравцем, та скрипт Player.gd для логіки керування.
Основні константи та змінні:

```
const SPEED = 200
const GRAVITY = 600
const JUMP_FORCE = -400

var velocity = Vector2.ZERO
```

Функція _physics_process(delta) обробляє логіку руху гравця, враховуючи гравітацію та перевірку на зіткнення із землею:


Модель пержонажа

Зображення 1.

```
func _physics_process(delta):
    velocity.y += GRAVITY * delta
    if Input.is_action_pressed("move_right"):
        velocity.x = SPEED
    elif Input.is_action_pressed("move_left"):
        velocity.x = -SPEED
    else:
        velocity.x = 0
    if is_on_floor() and Input.is_action_just_pressed("jump"):
        velocity.y = JUMP_FORCE
    velocity = move_and_slide(velocity, Vector2.UP)
```

4.2.2 Анімація та стани

Для плавності гри в Player.gd реалізовано зміну анімацій залежно від стану персонажа (рух, стрибок, очікування). AnimatedSprite перемикається між анімаціями через функцію:

```
func update_animation():
    if not is_on_floor():
        $AnimatedSprite.play("jump")
    elif velocity.x == 0:
        $AnimatedSprite.play("idle")
    else:
        $AnimatedSprite.play("run")
```

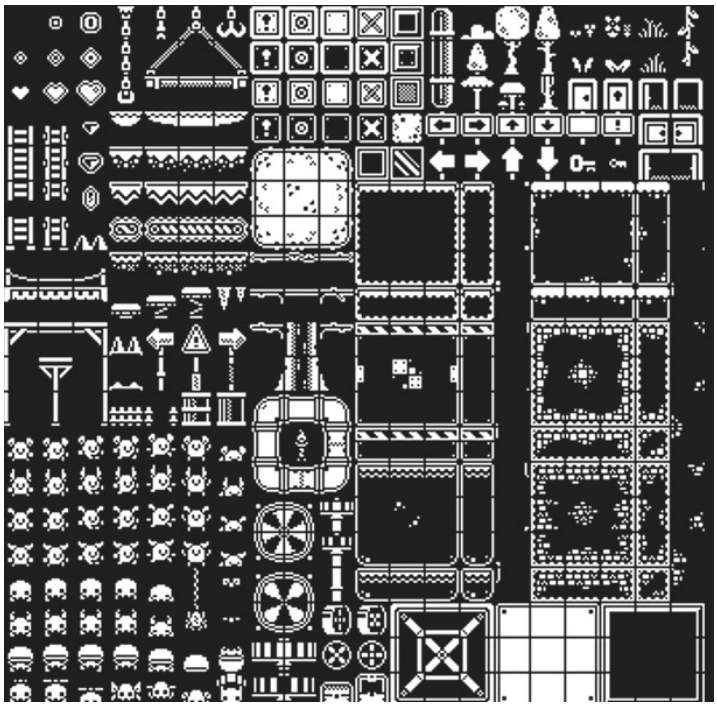
4.2.3. Колізії та фізика

Гравець має CollisionShape2D для взаємодії з середовищем. Функція move_and_slide() автоматично обробляє зіткнення. Також ми відстежуємо is_on_floor() для перевірки, чи гравець торкається землі.

4.3 Побудова рівнів

Рівні створюються як сцени з вузлом TileMap, який містить платформу, та об'єктами: гравцем, фінішною точкою, пастками. Кожен рівень має унікальну назву (Level_01.tscn, Level_02.tscn тощо). Для переходу між рівнями використовується сигнал із зони фінішу: TileMap містить шар для основи (земля, платформи) та додаткові шари для об’єктів і фону. Усі тайли мають колізії, визначені через TileSet Editor. . Тайли імпортуються зі спрайтів, згенерованих у форматі tileset, де кожен тайл має розміри 32x32 пікселі. . Тайли імпортуються зі спрайтів, згенерованих у форматі tileset, де кожен тайл має розміри 32x32 пікселі.

```
func _on_finish_body_entered(body):
    if body.name == "Player":
        get_tree().change_scene_to_file("res://scenes/Level_02.tscn")
```



Моделі персонажу та рівнів (Зображення 2.)

4.4. Взаємодія з об’єктами

Взаємодія реалізується через вузол Area2D. Наприклад, монети, ключі або пастки — це сцени з вузлом Area2D, CollisionShape2D та спрайтом. Обробка відбувається через сигнал body_entered:

```
func _on_coin_body_entered(body):
    # Перевіряємо, чи це гравець
    if body.name == "Player":
        queue_free() # Видаляємо об'єкт
        GameManager.score += 1
```

Кожен об'єкт має свої властивості, які задаються через інспектор або скрипт.

4.5. Система перемоги/поразки

На рівні встановлений фініш (наприклад, прапорець), який сигналізує про перемогу. Також розміщено тригери, які реагують на падіння в яму або зіткнення з ворогами — вони ведуть до поразки.

Приклад сигналу перемоги:

```
func _on_goal_body_entered(body):
  if body.name == "Player":
    get_tree().change_scene_to_file("res://scenes/Victory.tscn")
```

Аналогічно при поразці відкривається сцена GameOver. Це дозволяє чітко розмежувати завершення рівня: позитивне чи негативне.

4.6. Система паузи та повернення в меню

У грі реалізована функція паузи, яка викликається клавішею 'Esc' або натисканням сенсорної кнопки. Використовується механізм:
get_tree().paused = true

Коли гра на паузі, на екрані відображається меню (PauseMenu.tscn) з кнопками «Продовжити» та «Вийти в меню». Ці кнопки відновлюють гру або завантажують головне меню відповідно.

4.7. Сенсорне керування на Android

Сенсорне керування реалізоване через TouchScreenButton. Кожна кнопка має сигнал 'pressed' або 'released', які активують дії гравця. Кнопки мають свої зображення і розміщені у спеціальному Control-контейнері для масштабування на різних пристроях.

Обробка введення на мобільних пристроях виглядає так:

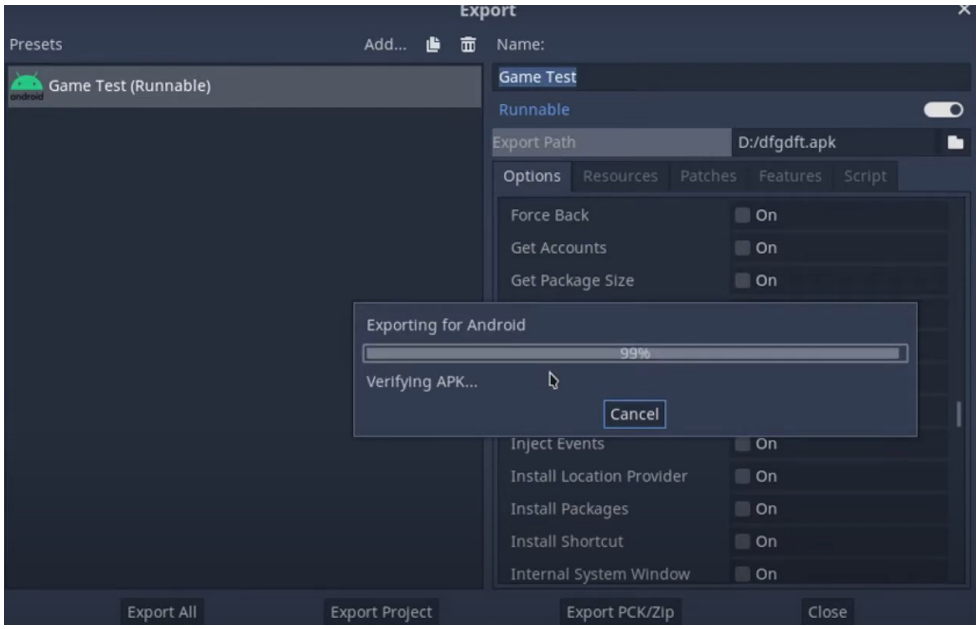
```
if Input.is_action_pressed("move_right"):
  velocity.x = SPEED
elif Input.is_action_pressed("move_left"):
  velocity.x = -SPEED
```

4.8. Побудова та експорт APK-файлу

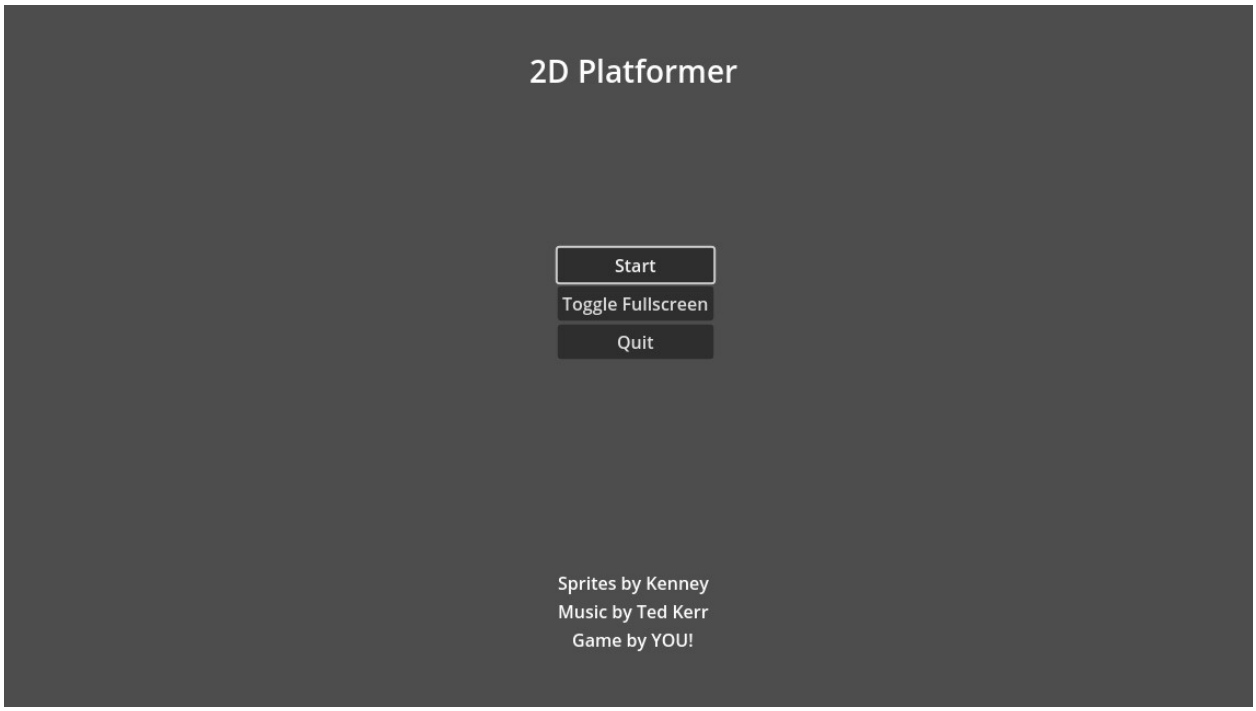
Після завершення розробки гру було експортовано у формат APK для Android. Процес складався з кількох етапів:
1. Завантаження Android SDK та налаштування шляхів у Godot (Editor → Editor Settings → Android);
2. Створення нового профілю експорту (Project → Export → Add → Android);
3. Вказано шлях до підписаного ключа keystore (або використано debug);
4. Налаштовано назву пакету, іконку, орієнтацію та версію;

5. Натиснуто Export Project → обрано шлях → збережено APK.

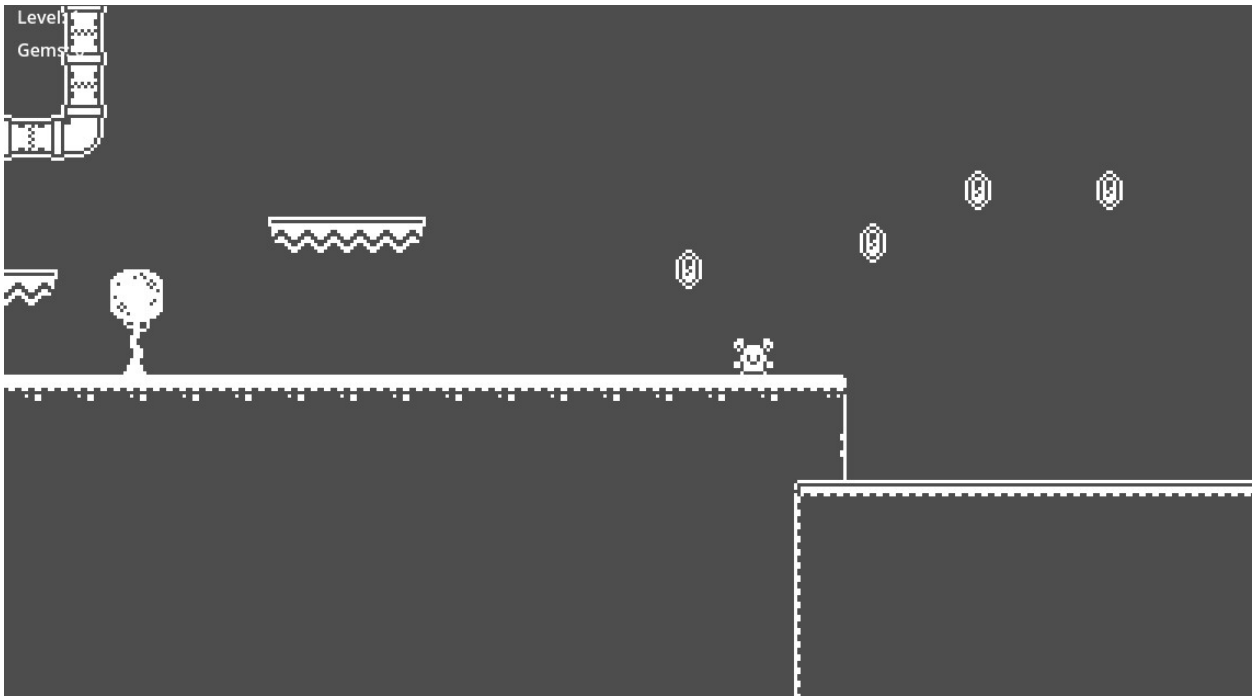
Файл APK може бути завантажений на мобільний пристрій, а також виставлений для бета-тесту, наприклад, через Google Play Console або сайти типу itch.io.



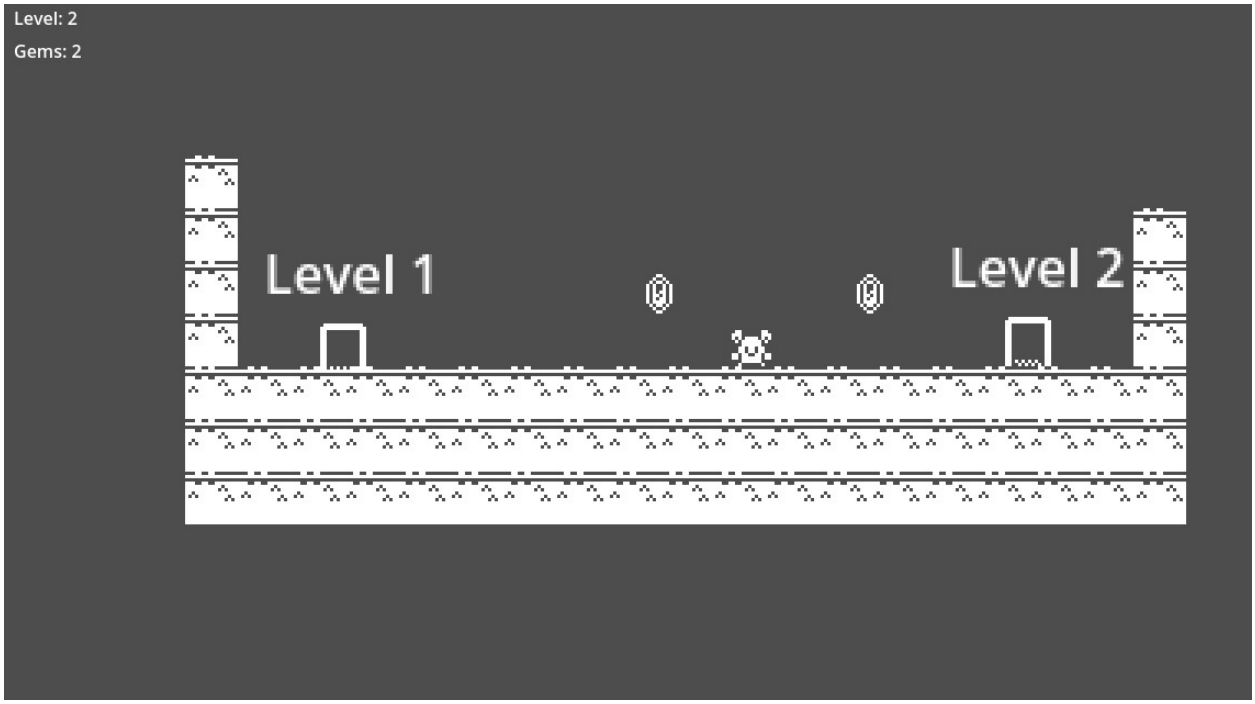
Зображення 3.



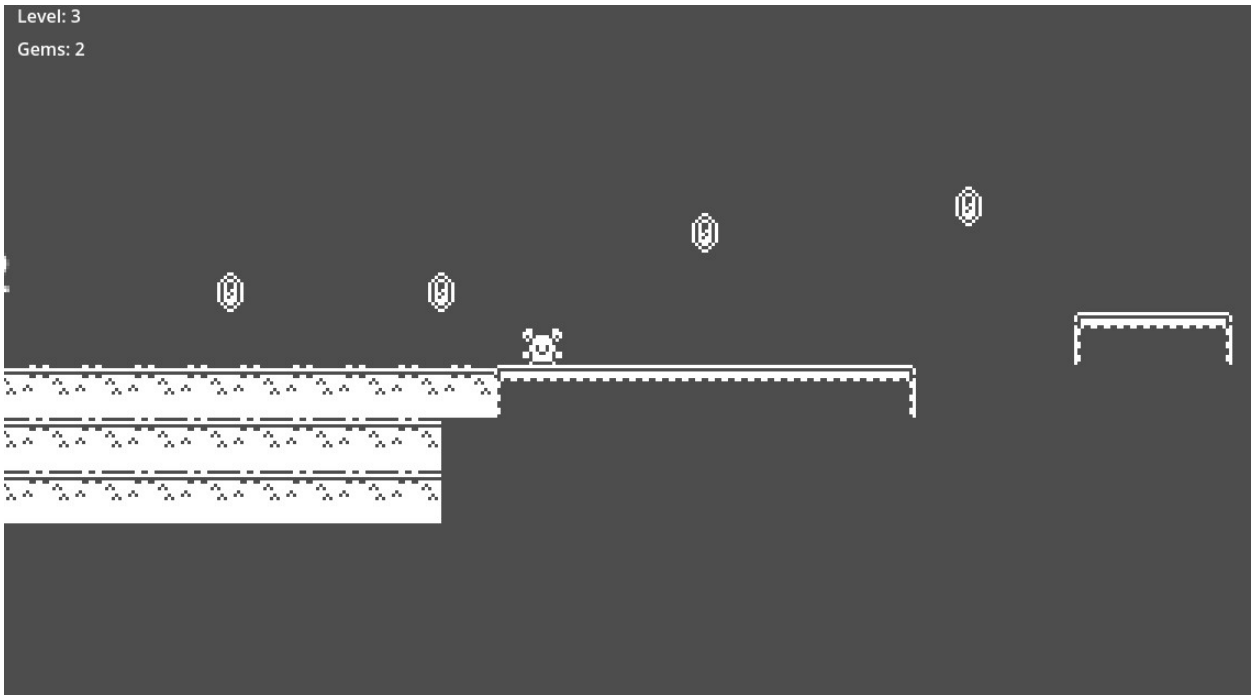
Зображення 4.



Зображення 5.



Зображення 6.



Зображення 7

```gdscript
extends Area2D

@export var level = ""
signal player_entered(level)

func _on_body_entered(body):
    if body.name == "Player":
        player_entered.emit(level)

```

Зображення 8.

```gdscript
extends Area2D

signal gem_collected

func _on_body_entered(body):
    if body.name == "Player":
        Global.gems_collected += 1
        gem_collected.emit()
        $CollectedSfx.play()
        hide()

func _on_collected_sfx_finished():
    queue_free()

```

Зображення 9.

```gdscript
extends Area2D

signal gem_collected

func _on_body_entered(body):
    if body.name == "Player":
        Global.gems_collected += 1
        gem_collected.emit()
        $CollectedSfx.play()
        hide()

func _on_collected_sfx_finished():
    queue_free()

```

Зображення 10.

```gdscript
extends CanvasLayer

func level(num):
    $CurrentLevel.text = "Level: " + str(num)

func gems(num):
    $GemsLabel.text = "Gems: " + str(num)

```

Зображення 11.

```gdscript
extends Node

var gems_collected = 0

func _ready():
    print_debug("global ready!")
    print_debug(gems_collected)

func _input(event):
    if event.is_action_pressed("return_to_main_menu"):
        get_tree().change_scene_to_file("res://main_menu.tscn")

```

Зображення 12.

```gdscript
extends Node2D

@export var level_num = 0

func _ready():
    $HUD.level(level_num)
    set_gems_label()
    for gem in $Gems.get_children():
        gem.gem_collected.connect(_on_gem_collected)

func _on_gem_collected():
    set_gems_label()

func set_gems_label():
    $HUD.gems(Global.gems_collected)

func _on_door_player_entered(level):
    get_tree().change_scene_to_file(level)

func _input(event):
    if event.is_action_pressed("reset_level"):
        get_tree().reload_current_scene.call_deferred()
        Global.gems_collected = 0
        set_gems_label()

```

Зображення 13.

```gdscript
extends CharacterBody2D

const SPEED = 250.0
const JUMP_VELOCITY = -400.0

var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")

func _physics_process(delta):
    if not is_on_floor():
        velocity.y += gravity * delta

    if Input.is_action_just_pressed("jump") and is_on_floor():
        velocity.y = JUMP_VELOCITY
        $JumpSfx.play()

    var direction = Input.get_axis("move_left", "move_right")
    if direction:
        velocity.x = direction * SPEED
        $AnimatedSprite2D.play("run")
        if direction == -1:
            $AnimatedSprite2D.flip_h = true
        else:
            $AnimatedSprite2D.flip_h = false
    else:
        $AnimatedSprite2D.play("idle")
        velocity.x = move_toward(velocity.x, 0, SPEED / 2)

    if not is_on_floor():
        $AnimatedSprite2D.play("jump")

    move_and_slide()
```

Зображення 14.

РОЗДІЛ 5. ТЕСТУВАННЯ ГРИ

5.1 Виявлені помилки та їх виправлення

У процесі розробки та тестування гри було виявлено низку помилок, пов’язаних із механікою руху, взаємодією з об’єктами та адаптацією керування до сенсорних пристроїв. Найбільш поширені з них включали:
- Некоректне перемикання анімацій при зміні станів гравця;
- Неможливість виконання стрибка на деяких пристроях Android;
- Помилки при переході між рівнями через неправильну обробку сигналів;
- Вихід гравця за межі карти при перевірці колізій.

Для усунення проблем було проведено відлагодження коду в середовищі Godot, використано функції друку станів (print/debug), а також профілювання логіки руху гравця. Особливу увагу було приділено Input Map та обробці подій сенсорного введення. Також було оновлено шари колізій та виправлено координати переходу камери.

5.2 Результати запуску на ПК та Android

Гру було протестовано на двох основних платформах: Windows (ПК) та Android (мобільні пристрої). У обох випадках гра продемонструвала стабільну роботу, швидке завантаження сцен і правильну обробку подій. На Android-тестуванні було підтверджено працездатність сенсорного керування: кнопки руху, стрибка та меню функціонували коректно.
Графіка коректно масштабувалась відповідно до розширення екрана, а перехід між рівнями відбувався без затримок. Були протестовані ситуації програшу, перемоги, повернення в меню та повторного запуску рівня — усі функції працювали згідно з очікуванням.

Результати тестування підтверджують, що проєкт готовий до демонстрації та подальшого використання на Android-платформах.

РОЗДІЛ 6. ПЕРСПЕКТИВИ РОЗВИТКУ ПРОЄКТУ

6.1 Розширення функціоналу

Поточна версія гри є базовою реалізацією 2D-платформера з класичними механіками стрибків, руху та переходів між рівнями. Водночас проєкт має великий потенціал для розвитку і масштабування до повноцінного комерційного продукту. Одним із ключових напрямів є вдосконалення ігрової механіки та розширення контенту. Наприклад:
- Розробка сюжетної лінії: введення історії головного героя, його мети, ворогів, з якими пов’язана певна мотивація;
- Створення системи діалогів з NPC, які можуть давати підказки, місії або нову інформацію;
- Додавання кількох типів рівнів: підземелля, ліс, місто, технологічний світ тощо, кожен зі своїм стилем і механіками;
- Динамічна зміна погоди і часу доби, що впливає на ігровий процес (наприклад, уночі видимість нижча);
- Режим змагань або тайм-атак, де гравець проходить рівні на швидкість і змагається в рейтингу;
- Підтримка контролерів для більшої гнучкості на Android та ПК;
- Багатокористувацький кооператив або PvP через локальну мережу чи інтернет.

У майбутньому можна реалізувати систему збережень і прогресу: чекпоінти, автосейви, або інтеграцію з Google Play Games для синхронізації даних. Також можлива реалізація редактора рівнів для гравців, які зможуть створювати та ділитися власними картами через внутрішній сервер або онлайн-сховище. Це значно підвищить залучення користувачів і дозволить побудувати спільноту навколо гри.

Прикладом успішної реалізації таких ідей є гра *Celeste* — спочатку невеликий інді-проєкт, що переріс у визнаний платформер зі складними механіками, глибоким сюжетом та підтримкою модів. Подібний підхід можна адаптувати до цього проєкту, перетворивши гру з навчального проєкту на повноцінну гру з багатим контентом.

6.2 Можливості публікації та монетизації

Завдяки можливості експорту APK-файлів, проєкт легко адаптується до популярних мобільних платформ. Першим етапом публікації може стати розміщення на Itch.io або Game Jolt для збору зворотного зв'язку від спільноти. Наступним кроком може бути реліз на Google Play з можливістю оновлення та поліпшень за результатами тестування.

Монетизація гри може здійснюватися кількома способами:

- Реклама (AdMob, Unity Ads) з можливістю її вимкнення за оплату;
- Покупка додаткових персонажів, рівнів або режимів гри;
- Продаж косметичних елементів (скіни, ефекти, саундпаки);
- Запуск підписки з ексклюзивним доступом до нових оновлень або тестових функцій.

Крім того, публікація на GitHub як відкритого проєкту дозволить залучити інших розробників до спільної роботи або отримати внески у вигляді pull-request'ів. Це сприятиме розвитку проєкту в умовах open-source моделі.

ВИСНОВКИ

У результаті виконання кваліфікаційної випускної роботи було реалізовано повноцінний 2D-платформер з підтримкою сенсорного керування на базі ігрового рушія Godot Engine. Проєкт охоплює всі ключові етапи розробки програмного забезпечення: від аналізу предметної області та постановки задачі до побудови архітектури, реалізації функціоналу, тестування та оцінки перспектив подальшого розвитку.

Під час роботи було досліджено основні можливості Godot Engine, зокрема організацію сцен, використання вузлів, розробку скриптів мовою GDScript, систему сигналів та обробку подій. Було реалізовано адаптивний інтерфейс, побудовано декілька повноцінних рівнів, створено керування персонажем із фізикою, анімацією, логікою програшу та перемоги. Особливу увагу приділено оптимізації під мобільні пристрої та створенню зручного сенсорного управління через TouchScreenButton. Проведено налаштування експорту під Android та побудовано APK-файл для розгортання гри на реальних пристроях.

Результати тестування свідчать про стабільну та передбачувану роботу застосунку на різних платформах. Інтерфейс виявився інтуїтивно зрозумілим, а геймплей — динамічним і зручним для користувача. Розроблена гра може слугувати не лише навчальним прикладом, а й основою для комерційного проєкту або демонстрації навичок розробника.

Варто також зазначити, що проєкт має вагомі перспективи розвитку. Розширення рівнів, поява нових механік, введення сюжету та багатокористувацьких можливостей відкриває нові горизонти для залучення гравців та потенційної монетизації. Окрему увагу можна звернути на реалізацію збережень, редактора рівнів або публікацію гри на відкритих платформах для тестування спільнотою.

У процесі виконання роботи були поглиблені знання з об’єктно-орієнтованого програмування, проектування ігрової архітектури, побудови UI/UX інтерфейсів та особливостей кросплатформенної розробки. Вивчення особливостей Android-експорту також дало змогу на практиці ознайомитися з публікацією застосунків для мобільних ОС.
Отже, поставлені цілі та задачі були повністю досягнуті. Робота є актуальною, практично значущою та має потенціал для подальшого розвитку як у межах навчальних дисциплін, так і в професійній діяльності у сфері розробки ігор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Godot Engine: https://docs.godotengine.org

2. Офіційний сайт Godot Engine: https://godotengine.org

3. Документація GDScript: https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/index.html

4. YouTube-канал GDQuest — навчальні відео з Godot.

5. GitHub-репозиторії прикладів проєктів на Godot.

6. Game Design Patterns — Robert Nystrom.

7. Книга 'Game Programming Patterns' — Robert Nystrom.

8. Вікіпедія: жанри комп’ютерних ігор.

9. Стаття: «Mobile Game UI Design Principles», GameAnalytics.

10. Блог Itch.io про публікацію ігор на платформі.

11. Android Export Settings — Godot Docs.

12. Стаття «How to Monetize a Mobile Game», GameDeveloper.com.

13. Документація AdMob: https://developers.google.com/admob

14. Celeste (гра): https://www.celestegame.com/ — приклад інді-гри з успішною реалізацією.

15. Форум розробників Godot: https://godotforums.org

16. GameDev StackExchange — відповіді на технічні питання.

17. Dev.to статті по GDScript.

18. Medium — статті з теми UI/UX для ігор.

19. OpenGameArt.org — джерело безкоштовних ресурсів для ігор.

20. Tiled Map Editor — створення тайл-сетів.

21. Android Studio Docs — документація з Android SDK.

22. Blender для створення 2D/3D спрайтів (опціонально).

23. itch.io FAQ — технічні вимоги до проєктів.