Звіт

Лабораторна робота №1

Тема: Практичне застосування системи контролю версій Git та юніт-тестування у процесі розробки ПЗ

Проект: «Blazing Darkness»

Виконала: Козирська Дар’я

Група: ІПС-21

Факультет комп’ютерних наук та кібернетики

Дисципліна: Інструментальні засоби розробки ПЗ

Київ-2025

1. **Тема**

Практичне застосування системи контролю версій Git та юніт-тестування у процесі розробки програмного забезпечення.

2. **Мета роботи**

Отримати практичні навички використання сучасних інструментів розробки ПЗ: роботи з Git та GitHub, побудови історії комітів і гілок проекту, написання юніт-тестів до коду та створення Pull Request.

Перевірити коректність ключових підсистем гри: сутності (Player, Enemy, Character), предмети (Item, Weapon, HealthPotion), світ (WorldMap, LocationNode) та ігрова логіка (Battle/Combat, Game цикл).

3. **Короткий опис проєкту**

Blazing Darkness — консольна RPG гра із покроковими боями та елементами процедурної генерації. *Проект має навчальну мету — продемонструвати ООП-принципи та роботу з граф-структурами (карта світу).*

Репозиторій: https://github.com/wwerniss/blazing-tests

4. **Структура**

- Character / Player / Enemy: створення, бойова взаємодія, досвід і підвищення рівня.
- Items / Weapon / HealthPotion: додавання/вилучення з інвентаря, ефекти, екві́п/анекві́п.
- WorldMap / LocationNode: побудова карти, переміщення, explore() (генерація/взаємодії).
- Battle / Combat: послідовність атака/втеча/використання предметів, коректні стани після бою.

5. **Архітектура (об’єкти та ключові методи)**

Основні класи

- Game — головний контролер гри: run(), handleCombat(), showStatus(), showHelp(), showGameOver(), showInventory(), clearScreen(), showTitle().
- Character (базовий): стани (name, level, health/maxHealth, attack/baseAttack, defense, інвентар); поведінка: attackTarget(), takeDamage(), heal(), isAlive(), addItem(), showStatus(), equipWeapon(), unequipWeapon(), hasWeaponEquipped(), getClassName().
- Player: прогрес та взаємодія — gainExperience(), levelUp(), useItem(), getAttack(), getExperience(), getExperienceToNextLevel(), getLevel().
- Enemy: getExperienceReward(), getClassName() = "Enemy".
- WorldMap / LocationNode: createWorld(), moveLeft(), moveRight(), getCurrentLocation(), explore(player), addEnemy(), addItem(), hasEnemies(), getEnemies(), removeEnemy(), а також доступ до зв'язків вузлів і метаданих локацій.
- Item / Weapon / Sword / Axe / Stick / HealthPotion — ієрархія предметів і зброї.

6. Організація тестів

Принципи

- `AAA (Arrange-Act-Assert): ініціалізація фікстур → виконання методів → перевірки EXPECT_*.`
- Мінімальні залежності: замість повного запуску гри тестуються вузькі одиниці логіки (класи/методи).
- Граничні випадки: 0 HP, переповнення досвіду для levelUp, порожній інвентар, використання предмету не за індексом, атака без зброї.
- Ідемпотентність і інваріанти: здоров'я не перевищує maxHealth; атака не стає від'ємною; перемикання зброї відкотить стани коректно; вузли карти не втрачають посилання при навігації.

7. Приклади тест-кейсів (узагальнено)

Player: підвищення рівня

Arrange: Player рівня 1, expToNext = 100, baseAttack = 5.

Act: gainExperience(120) викликає levelUp().

Assert: рівень = 2; досвід = 20; атака не менше базової; здоров'я відновлюється до maxHealth.

Character: завдання урону

Arrange: Атакуючий (attack = 10), захисник (health = 30, defense = 3).

Act: attackTarget(defender).

Assert: здоров'я захисника зменшується на max(1, 10 − 3).

Inventory/Items: використання зілля

Arrange: Player має HealthPotion (+20), поточне здоров'я 50/60.

Act: useItem(індекс зілля).

Assert: здоров'я дорівнює 60 (не перевищує maxHealth), інвентар скорочується на один елемент.

WorldMap/LocationNode: навігація

Arrange: Побудована мапа WorldMap.createWorld(), старт у корені.

Act: moveLeft(); moveRight().

Assert: getCurrentLocation() існує, посилання left/right узгоджені.

Battle/Combat: вплив екві́па на урон

Arrange: Player без зброї проти Enemy; далі equip(Sword +5).

Act: attackTarget(enemy) до та після екві́па.

Assert: урон зі зброєю більший, ніж без зброї.

8. Вимоги та їх тестове покриття

Підсистема	Перевірені вимоги	Тип перевірки	Нотатки

Player/Character	LevelUp відновлює HP, підвищує атаку	позитивний, граничний	Перевірка переповнення досвіду
Inventory/Items	Додавання/видалення; HealthPotion не перевищує максимум	позитивний, негативний	Перевірка індексів/винятків
Weapon	Equip/unequip змінює атаку	позитивний	Ефект не кумулюється
WorldMap	Навігація left/right; explore() без крашу	позитивний	Перевірка посилань вузлів
Battle	Порядок дій: attack/use/run	сценарний	Коректний фінальний стан

9. Історія комітів

Нижче — фрагмент журналу комітів із GitHub із основними віхами роботи над тестами:

- Entities unit tests and better tests structure — виділення груп тестів для сутностей, реорганізація структури.
- Common tests & Logger update — уніфікація спільних перевірок, апдейт логера.
- Item tests & removed unused showInventory function — тести предметів, чистка невикористаного коду.
- World tests & World memory leak fix — тести для світу, виправлення витоку пам’яті.
- More tests, comments for test functions — розширення покриття, коментарі в тестах.
- Now test correctly checking for memory leaks — коректна перевірка на витоки пам’яті.

Commits on Oct 23, 2025

Entities unit tests and better tests structure

wwerniss committed 2 weeks ago

Common tests & Logger update

wwerniss committed 2 weeks ago

Commits on Oct 25, 2025

Item tests & removed unused showInventory function

wwerniss committed 2 weeks ago

World tests & World memory leak fix

wwerniss committed 2 weeks ago

Commits on Oct 27, 2025

More tests, comments for test functions

wwerniss committed 2 weeks ago

Now test correctly checking for memory leaks

wwerniss committed 2 weeks ago

10. Оточення та інструменти

- Мова і стандарт: C++17.
- Збірка: CMake; скрипт tools/build.sh.
- Фреймворк тестування: GoogleTest (gtest/gmock).
- Запуск тестів: ctest --output-on-failure або окремий виконуваний таргет у каталозі tests (за наявності).

Для локального запуску гри:

1) chmod 777 ./tools/build.sh && ./tools/build.sh

2) chmod 777 ./tools/run.sh && ./tools/run.sh

11. Результати запуску

Усі 22 модульні тести проходять без помилок (PASSED).

12. Висновки

- Побудовано системний набір модульних тестів, що покриває основні гілки логіки гри та граничні випадки.
- Архітектура з чіткими інтерфейсами (класи сутностей/світу/предметів) полегшує ізольоване тестування.

13. Посилання

- README проєкту: інструкції зі збірки та запуску.
- MIT License, CMake, Doxygen — використані інструменти та ліцензія.