УДК 004.4
Крупа М. – ст. гр. СП-42
Тернопільський національний технічний університет імені Івана Пулюя

МОВА ПРОГРАМУВАННЯ PYTHON ЯК ЗАСІБ РОЗРОБКИ І ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Науковий керівник: к.т.н., доцент Цуприк Г. Б.

Krupa M.
Ternopil Ivan Puluj National Technical University

THE PYTHON PROGRAMMING LANGUAGE AS A TOOL FOR DEVELOPING AND TESTING SOFTWARE

Supervisor: Tsupryk H. B.

Ключові слова: об'єктно-орієнтоване програмування, розробка та тестування, життєвий цикл програмного забезпечення.
Keywords: object-oriented programming, development and testing, software development life cycle.

Мова програмування Python є однією з найзручніших для розробки інді-проєктів у жанрі Dungeon Crawler, який вимагає чіткої організації програмної архітектури для забезпечення масштабованості та зручності підтримки коду. Переваги Python полягають в об’єктно-орієнтованому підході, широкому виборі бібліотек та зрозумілості синтаксису.

Змога підтримки повного життєвого циклу розробки програмного забезпечення є надважливим аспектом у побудові будь-якого програмного продукту, адже ґрунтується на оцінці якості кожного етапу.

Грамотне та ефективне використання принципів об’єктно-орієнтованого програмування сприяє забезпеченню масштабованості, зрозумілості та ізоляції коду – важливих для продукту характеристик.

Архітектура, побудована на основі абстрактних класів, дозволяє розділити відповідальність між ключовими компонентами системи. Базова модель передбачає використання незалежних модулів, зокрема для управління станом сутностей, генерації ігрового середовища та контролю глобального ігрового сеансу. Такий підхід спрямований на ізоляцію логіки обробки внутрішніх ігрових подій від представлення даних, що спрощує подальше розширення функціоналу.

Використання менеджера станів як основний механізм взаємодії, передбачає відслідковування переходів між ігровими фазами, обробкою команди користувача та ініціацією відповідних змін у параметрах об’єктів. Застосування патернів проєктування для управління елементами середовища забезпечує їх динамічне створення, взаємодію та видалення. Такий підхід орієнтований на оптимізацію життєвого циклу ігрових компонентів та підтримку стабільної швидкодії.

Особливу увагу приділено організації взаємодії між компонентами системи. Для зменшення зв’язаності між модулями передбачається використання принципів слабкої зв’язності та інверсії залежностей. Це дозволяє підвищити гнучкість архітектури та спростити модифікацію окремих компонентів без впливу на інші частини системи.

Окремим аспектом є питання масштабованості застосунку. Запропонований об'єктно-орієнтований підхід до побудови архітектури з використанням основних принципів ООП, зокрема інкапсуляції, наслідування та поліморфізму, а також розділення відповідальностей, дозволяє розширювати функціональні можливості гри шляхом додавання нових типів сутностей, механік взаємодії та сценаріїв без необхідності суттєвої зміни існуючого коду. Це є важливим для проєктів у жанрі Dungeon Crawler, які часто передбачають поступове ускладнення ігрового процесу.

Питання організації збереження та відновлення стану гри вирішується реалізацією механізмів серіалізації ігрових об'єктів, що дозволить зберігати поточний стан ігрового сеансу та відновлювати його у подальшому. Це підвищує зручність користування застосунком та створює основу для реалізації додаткових функцій, таких як контроль прогресу гравця.

Важливим етапом циклу розробки – тестування програмного забезпечення. Для перевірки надійності продукту, а саме ключових алгоритмів гри, добре підійде модульне тестування. Воно дозволить валідувати логіку зміни станів, перевіряти коректність розрахунків під час взаємодій між об'єктами та оцінювати стабільність роботи менеджера сеансів при різних ігрових сценаріях.

Запропонована архітектурна модель з використанням принципів об'єктно-орієнтованого програмування та підтримкою усіх етапів повного життєвого циклу програмного забезпечення орієнтована на якісні розробку та тестування, успішне розгортання програми початкового продукту, доведення її до рівня повноцінної гри та продовження підтримки та оновлення для подальших покращень та інтеграцій.

Література:

1. Олянін, Д., Цуприк, Г. (2025) Transformer Neural Networks in Industry 4.0 / Д. Олянін, Г. Цуприк, Т. Говорущенко, О. Багрій-Заяць, І. Андрущак // Computer Information Technologies in Industry 4.0: proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. – Ternopil : Ternopil Ivan Puluj National Technical University, 2025 (Scopus) https://ceur-ws.org/Vol-4057/

2. Tsupryk, H., Olianin, D. (2025). Vydobuvannia danyh z tekstu vykorystovuiuchy transformerni neironni merezhi [Data extraction from text using Transformer Neural Networks]. Information Technology: Computer Science, Software Engineering and Cyber Security, 125–130, DOI: https://doi.org/10.32782/IT/2025-2-13

3. ОЛЯНІН D., & ЦУПРИК Н. (2025). Огляд ролі трансформерних нейроних мереж у видобувані інформації із неструктурованих даних. Measuring and computing devices in technological processes, 82(2), 360–364. https://doi.org/10.31891/2219-9365-2025-82-52

4. Зосим М. (2023). Життєвий цикл розробки програмного забезпечення (Software Development Life Cycle - SDLC). https://www.maxzosim.com/software-development-life-cycle-sdlc/

5. Michał Piórkowski (2024). Python for Game Dev: Is It a Good or Bad Idea? https://sunscrapers.com/blog/python-for-game-development-is-it-a-good-or-bad-idea/

6. Sharon Hart (2022). Testing Python module-level code and why it smells. https://medium.com/@sharon.dev/testing-python-module-level-code-and-why-it-smells-3bf1cc5a79d5