Венгловська Ю. В., студентка,
Марчук Д.К., старший викладач
Державний університет «Житомирська політехніка»

ВИКОРИСТАННЯ ПАТЕРНІВ У ІГРОВИХ ПРОЕКТАХ

Патерн являє собою не якийсь конкретний код, а загальний принцип вирішення певної проблеми, який майже завжди треба підлаштовувати для потреб тієї чи іншої програми. Патерни корисно використовувати в іграх, це спрощує орієнтування в коді, економить ресурси та допомагає управляти елементами.

Паттерн Singleton є одним з найпоширеніших патернів проектування, який використовується для створення класу, який може мати тільки один екземпляр із глобальним доступом до цього екземпляру.

В ігровій розробці, паттерн Singleton може бути корисним для зберігання глобальних даних, таких як інформація про рівень гри, кількість очок, налаштування гри та інше. Клас, який реалізує Singleton, може бути доступним з будь-якої частини програми, що робить його зручним для зберігання глобальних даних.

Наприклад, якщо в грі є рівні, кожен рівень може бути представлений окремим класом. Клас, який реалізує Singleton, може бути використаний для зберігання інформації про поточний рівень гри, а також для зберігання загальних даних, таких як кількість очок гравця. Патерн може забезпечувати доступ до цих даних з будь-якої частини гри, що спрощує роботу з даними.

Патерн Template Method може бути корисним в логічних іграх, якщо головоломки мають деяку спільну структуру гри.

У цьому патерні базовий клас містить абстрактні методи, які визначають загальну структуру гри, а конкретні підкласи можуть забезпечувати різні реалізації цих методів для кожної конкретної гри.

Наприклад, для предметів, з якими можна інтерактувати по кліку миші, зручно застосувати Template Method, для того, щоб описати основну поведінку, і перевизначати її в класах-нащадках. Це допоможе скоротити код та зменшити кількість можливих помилок, бо логіка буде зосереджена не в кожному класі, а лише в одному основному.

Використання патерну Template Method дозволяє розділити загальну логіку гри від її конкретної реалізації, що полегшує розробку та зберігання коду.

Принципи SOLID – це набір базових принципів, які використовуються в об'єктно-орієнтованому програмуванні для створення програм, які є ефективними, масштабованими та легко змінюваними. Ці принципи можуть бути застосовані до програмування ігор, щоб зробити їх більш ефективними та легко змінюваними.

Короткий огляд кожного принципу SOLID та його можливого застосування в програмуванні ігор:

- Принцип одиничної відповідальності (Single Responsibility Principle - SRP). Кожен клас або модуль повинен мати тільки одну причину для зміни. У грі це означає, що кожен клас повинен бути відповідальним за конкретну функціональність, і якщо потрібно внести зміни, то це повинно бути зручно та безпечно для інших класів.
- Принцип відкритості/закритості (Open/Closed Principle - OCP). Код повинен бути відкритим для розширення, але закритим для зміни. У грі це означає, що класи повинні бути здатні до динамічного розширення без зміни початкового коду, наприклад, можна додати нові класи персонажів, але не потрібно змінювати код головного героя.
- Принцип підстановки Барбари Лісков (Liskov Substitution Principle - LSP). Об'єкти класів-спадкоємців повинні бути здатні замінювати свої батьківські класи без зміни правильності програми. У грі це означає, що класи героїв повинні бути здатні замінювати один одного без зміни коду.
- Принцип розділення інтерфейсу (Interface Segregation Principle - ISP). Клієнти не повинні залежати від методів, які вони не використовують. У грі це означає, що класи повинні мати тільки ті методи, які їм потрібні для виконання своїх функцій, і не повинні мати надлишкових методів, які вони не використовують.
- Принцип інверсії залежності (Dependency Inversion Principle - DIP). Залежності повинні бути встановлені на абстракції, а не на конкретних класах. У грі це означає, що класи повинні використовувати абстрактні інтерфейси та залежності, щоб бути здатними до зміни та розширення без зміни початкового коду.

Загалом, застосування принципів SOLID в програмуванні ігор допоможе створити більш ефективний, масштабований та легко змінюваний код, що є важливим для розробки гри з багатим функціоналом і довгим терміном життя.

Отже, для написання якісного програмного додатку потрібно використовувати загальноприйняті норми, дотримуватися принципів SOLID та вміти використовувати патерни проектування.

Список використаних джерел

1. Патерни проектування [Електронний ресурс] – Режим доступу до ресурсу: https://refactoring.guru/uk/design-patterns.