Левицький А. А., здобувач,
Левківський В. Л., ст. викладач
Державний університет «Житомирська політехніка»

ПАТЕРН STATE ЯК ЗАСІБ ДЛЯ СТРУКТУРИЗАЦІЇ ПРОГРАМНОГО КОДУ

Під час розробки комп'ютерної гри розробник може зіштовхнутись з потребою в змінюваному функціоналі об'єктів, наприклад, рух персонажа. Однак чим більше функціоналу в об'єкта, тим складніше буде розробнику додавати нові, адже в такому випадку потрібно переглянути весь програмний код та змінити його задля запобігання неполадок.

Для таких випадків доречно використовувати паттерн стану (State pattern) [1]. Від дозволяє структурувати програмний код, розділивши весь функціонал на окремі логічні блоки, що виконуватимуть свою дію.

Для прикладу, візьмемо програмний код руху персонажа, що дозволяє: стояти (бути в спокої), бігати, стрибати та робити ривок. З патерном всі рухи персонажа будуть окремими класами, в яких виконуватимуться необхідні дії та перевірятимуться умови, при яких стан буде замінюватись на інший. Поточний стан зберігається в класі машини стану (State Machine), а основний клас руху гравця циклічно виконує функціонал стану через єдиний метод машини.

Схема на рисунку 1 візуалізує роботу паттерна.

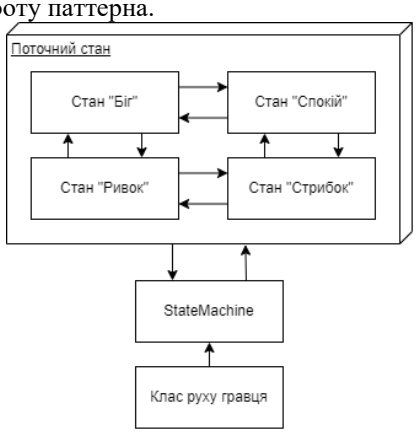

Рис. 1. Схема роботи патерна

Класи станів є наслідувані від базового класу, що має основні поля та віртуальні методи, які, за потреби функціоналу класу, будуть реалізуватись. Така структура відповідає принципам об'єктно-орієнтованого програмування та допоможе записувати в поле поточного стану машини будь-який створений стан (рис. 2).

Сам клас машини станів, окрім поля поточного стану, має метод його початкової ініціалізації та метод задання поточного стану, який виконує завершення минулого стану та підготовку для роботи наступного (рис. 3).

```
public class PlayerBasicState
{
    Required fields

    Ссылок: 6
    public PlayerBasicState(
        PlayerMovement playerMovement, PlayerMovementStateMachine stateMachine)...

    Ссылок: 28
    public virtual void Enter()...

    Ссылок: 23
    public virtual void Exit()...

    Ссылок: 33
    public virtual void LogicUpdate() { }

    Ссылок: 19
    public virtual void PhysicsUpdate()
    {
        DoChecks();
    }

    Ссылок: 20
    public virtual void DoChecks() { }
```

Рис. 2. Приклад базового класу стану

```
public class PlayerMovementStateMachine
{
    Ссылок: 11
    public PlayerBasicState CurrentState { get; private set; }
    Ссылок: 2
    public PlayerBasicState PreviousState {  get; private set; }

    ссылка: 1
    public void Initialize(PlayerBasicState state)
    {
        CurrentState = state;
        CurrentState.Enter();
    }

    Ссылок: 33
    public void ChangeState(PlayerBasicState state)
    {
        CurrentState.Exit();
        PreviousState = CurrentState;
        CurrentState = state;
        CurrentState.Enter();
    }
}
```

Рис. 3. Приклад класу машини станів

Перевагою паттерна є:

- спрощення розуміння програмного коду;
- можливість простого додавання або вдосконалення функціоналу.

Однак слід пам'ятати, що будь-який паттерн слід використовувати тоді, коли цього потребує ситуація. State не є виключенням: він не буде доречним, якщо функціонал об'єкта не є великим.

Отже, паттерн State є необхідним інструментом для реалізації комплексного функціоналу об'єктів. З його допомогою можна спростити розуміння програмного коду, легше вдосконалювати його в подальшому. Крім того, паттерн дозволяє запобігти виникненню певних можливих конфліктів за допомогою сталій структурі змінюваного функціоналу.

Список використаних джерел

1. State [Електронний ресурс] – Режим доступу: https://refactoring.guru/design-patterns/state