УДК 004.4’23
В.В. Яцишин, канд. техн. наук, доцент, В.В. Хацюр
(Тернопільський національний технічний університет імені Івана Пулюя)

ПЕРЕВАГИ КОМПОНЕНТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

UDC 004.4’23
V.V. Yatsyshyn PhD, Assoc. Prof., V.V. Khatsiur

BENEFITS OF COMPONENT-ORIENTED PROGRAMMING

Ігрова індустрія – один з найскладніших сегментів у галузі інформаційних технологій, оскільки, окрім високого технічного рівня підготовки розробника, вимагає також знань у галузі психології, ергономіки, фізіологічних особливостей різних вікових категорій.

Великі за розміром і/або нестандартні за логікою ігрові додатки характеризуються досить складною архітектурою, що обумовлено характеристиками предметної області. Оскільки, предметна область, в переважній більшості, є досить обширною, то відповідно описується складними сутностями, представленими у вигляді класів та взаємодії між ними. Якщо намагатися розробляти ігри, використовуючи стандартний ООП підхід, то виникає необхідність внесення постійних змін у код, що породжує збільшення тривалості розробки і відповідно її вартість.

Проблема криється в наслідуванні (проблема «тендітних» базових класів – ситуація, коли змінити реалізацію типу-предка неможливо, не порушивши коректність функціонування типів-нащадків). При пошуку розв’язання цієї проблеми запропоновано скористатися компонентно-орієнтованим підходом (КОП).

Компоне́нтно-орієнто́ване програмува́ння – одна з парадигм програмування, що виникла як свого роду дисципліна, тобто набір певних обмежень, що накладаються на механізм об’єктно-орієнтованого програмування (ООП). Компонентно-орієнтоване програмування усуває проблему безконтрольного застосування ООП, що раніше призводило до виникнення проблем з надійністю великих програмних комплексів.

Якщо коротко, суть КОП наступна: є певний клас-контейнер, а також клас-компонент, який можна додати в клас-контейнер. Об’єкт складається з контейнера і компонентів в цьому контейнері.

Компоненти трохи нагадують інтерфейси. Але інтерфейси дозволяють тільки виділити у класів загальну сигнатуру функцій і властивостей, а компоненти дозволяють винести загальну реалізацію класів окремо.

В ООП підході об’єкт визначається, як екземпляр класу. У КОП підході об’єкт визначається компонентами, з яких він складається. Не важливо, що це за об’єкт. Важливо, що у нього є і що він вміє робити.

КОП спрощує повторне використання написаного коду – використання одного компонента в різних об’єктах. Також з різних комбінацій вже існуючих компонентів, можна зібрати новий тип об’єкту.

Для прикладу, візьмемо об’єкт «персонаж». З точки зору ООП – він був би один великий класом, можливо, наслідуваним від якогось іншого класу. З точки зору КОП – це набір компонентів, що складають об’єкт «персонаж». Наприклад:

- характеристики персонажа – компонент «Stats»;
- управління персонажем «CharacterController»;
- анімація персонажа – «CharacterAnimationController»;
- обробник зіткнень – «CharacterCollisionHandler».

Не варто відмовлятися від наслідування в іграх. Наслідування компонентів цілком нормальна практика. Та й в деяких ситуаціях, це буде більш правильним. Але, якщо видно, що буде кілька рівнів успадкування класів для опису об’єктів, то краще використовувати компоненти. На рис. 1 наведено архітектуру при використанні ОПП.

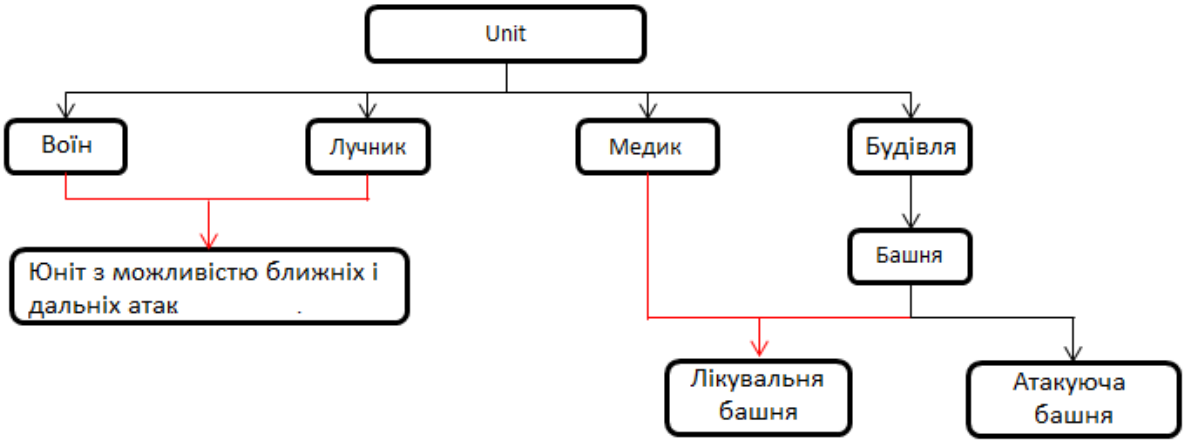

Рисунок 1. Представлення архітектури з використанням ООП підходу

При використанні КОП підходу (рис. 2) архітектура буде відрізняються від тієї, яка показана на рис. 1.

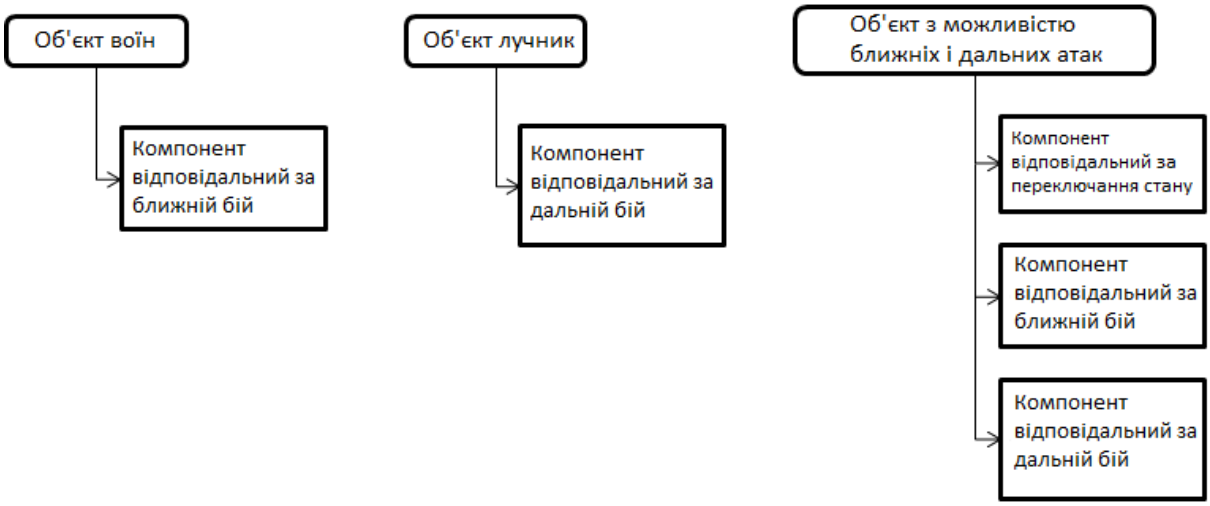

Рисунок 2. Архітектура при використанні компонентно-орієнтованого підходу

Окрім цього, перевагою компонентно-орієнтованого підходу є те, що в подальшому компоненти можуть використовуватись при побудові інших систем. Тому доцільним є використання КОП при проектуванні та імплементації ігрових додатків.