УДК 004.514

ПОРІВНЯННЯ АРХІТЕКТУРНИХ ПІДХОДІВ У СТВОРЕННІ ІНТЕРФЕЙСІВ ДЛЯ ВІДЕО-ІГОР

Тихонов А.О., Новіков Ю.С.
e-mail: anton.tykhonov@nure.ua, yuriy.novikov@nure.ua
Харківський національний університет радіоелектроніки, каф.ПІ
м.Харків, Україна

This paper provides a detailed comparison of several UI architectures commonly used in game development, with a particular focus on their efficiency, scalability, and overall impact on performance. The architectures examined in this study include monolithic, component-based, event-driven, and multi-layered approaches. By analyzing these architectures, this work aims to highlight the key differences between them, emphasizing their strengths and weaknesses in different development scenarios. At the end of the study, a comparative graph is presented, illustrating the relative productivity of each approach and providing a clear visual representation of their performance differences.

У сучасному світі відеоігор, інтерфейси користувача відіграють важливу роль у створенні зрозумілого і приємного середовища для гравця. В залежності від жанру гри, такі інтерфейси можуть бути простими одно–або двох–степеневими системами, що оновлюються викликом команди при виконанні певної дії у світі, або багатошаровими системами, що відслідковують стани світу у автоматичному режимі. Особливо у перенасичених ігровими механіками, жанрах RPG та стратегій, розробники прагнуть створити інтерфейси, які будуть швидко та без проблем виводити та оновлювати актуально для ігрової сесії інформацію, не створюючи навантаження на гру за рахунок перерендерювання. З огляду на ці актуальні вимоги, автор цієї роботи поставив на меті дослідити, проаналазувати та висвітлити ключові переваги і недоліки різних архітектур ігрових інтерфейсів. Першим завданням цієї роботи буде формулювання представлення про предмет дослідження. Другим завданням цієї роботи буде порівняння архітектур за продуктивністю у контрольованій середі рушія Unreal Engine 5.

Монолітний підхід використовується при створенні простих ігор, де інтерфейс не часто оновлюється, або ціна оновлення не є високою. Найкращими прикладами можуть бути гіпер–казуальні ігри, платформери або нескладні шутери старих поколінь. Інтерфейси в таких іграх не перевантажені і тому їх розробка є найпростішою, але масштабування неможливе без розділення функціоналу на окремі сегменти. Особливістю такого підходу є те, що всі частини інтерфейсу міцно пов’язані між собою, що спрощує контроль за станами та даними при оновлені. Всі сучасні

алгоритми розробки інтерфейсів використовують окремі монолітні частини для виконання базових функцій виведення даних.

Компонентно–орієнтована архітектура стає розвитком ідей монолітної, але намагається вирішити проблему відокремлення елементів у окремі компоненти - віджети. Цей підхід є наступною ступінню еволюції інтерфейсів, дозволяючи перевикористовувати створені частини і більш відокремлено керувати їх станами. Також цей підхід спрощує тестування і сприяє розвитку проекту через використання принципів модульності.

Подієво-орієнтована архітектура - це еволюційний крок в сторону асинхронності даних і відокремлення їх від основного процесу гри. При такому підході кожному компонентові присвоюється стан і дані, а реалізацію оновлення бере на себе алгоритм програмування «Наглядач». Цей метод гарно підходить для динамічних UI в MMO–іграх, де потребується часта зміна даних, але ресурсу на перемальовування і відслідковування не має. При такому підході важливо правильно синхронізувати оновлення і кешувати стани інтерфейсу [1].

Найбільш продвинутим методом подання і керування інтерфейсами стає комплексна багатошарова архітектура, де поєднуються відразу усі попередні методи, і за рахунок збільшення комплексності системи та дозволяє досягти балансу між якістю відображення та продуктивністю, забезпечуючи плавність інтерфейсу без значних витрат обчислювальних ресурсів. На рисунку 1 представлено архітектуру такої системи, реалізовану за допомогою програмних рішень двигуна Unreal Engine 5.

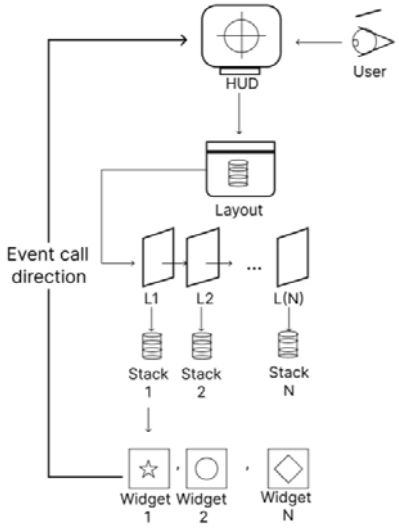

Рисунок 1 – Представлення архітектури багатошарового інтерфейсу

Однією з головних переваг цієї архітектури є ефективне кешування на кожному рівні, що значно зменшує кількість обчислень і підвищує продуктивність [2]:

- кешування шарів у Layout дозволяє зберігати вже відрендерені секції інтерфейсу;

- кешування віджетів у Layer дає змогу уникати повторного створення та знищення UI-елементів;

- кешування даних у віджетах допомагає зменшити кількість запитів до зовнішніх джерел.

Через об'єднання багатьох підходів з іншими архітектурами ця система стає більш комплексною, що, хоча й підвищує її гнучкість і масштабованість, водночас збільшує час і складність її розробки.

На рисунку 2 представлене порівняння продуктивності вищеперелічених архітектурних підходів у рушії Unreal Engine 5, де завантаження кожного з них було здійснено викликом акторів з віджетами, що відслідковують їх поточний стан.

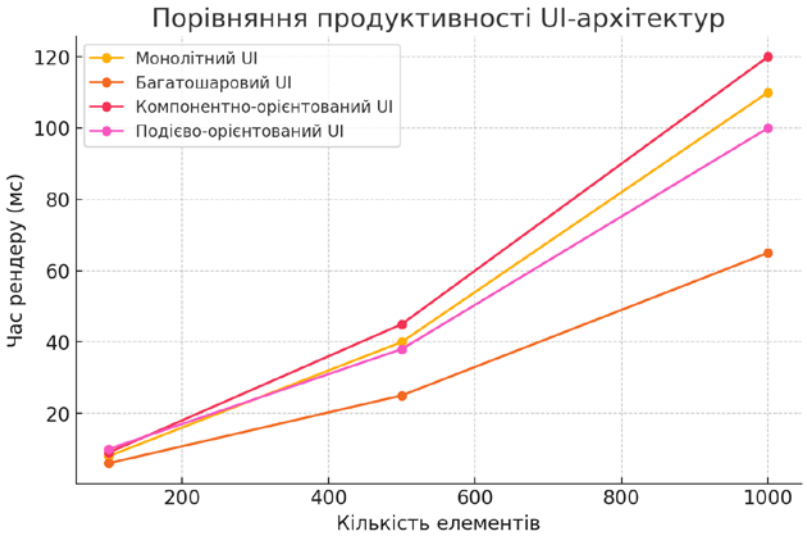

Рисунок 2 - Графік порівняння продуктивності різних UI-архітектур

З результатів дослідження продуктивності вищезазначених архітектур, можна побачити, що найпродуктивнішою є багатошарова архітектура, а зважаючи на раніше викладені переваги цієї архітектури – її можна вважати найкращим рішенням для розробки ігрових інтерфейсів.

Список використаних джерел:

1. Brent Fox. Game Interface Design. 1st edition. Cengage Learning PTR, 2004. 212 p.

2. Бєліков Д. Ю., Масштабування VFX як спосіб оптимізації в Unreal Engine. Т. 6 : Конференція «Інформаційні інтелектуальні системи» : матеріали 28-го Міжнар. молодіж. форуму «Радіоелектроніка та молодь у XXI столітті», 16–18 квіт. 2024 р. / М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2024. 958 с.