УДК 004.41

В.В. Колодій
О.В. Романюк

ПЕРЕВАГИ ЗАСТОСУВАННЯ АСИНХРОННОГО ПРОГРАМУВАННЯ У РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет

Анотація
Проаналізовано переваги застосування асинхронного програмування у розробці програмного забезпечення. Описано ключові особливості асинхронного програмування.
Ключові слова: програмне забезпечення, асинхронне програмування, парадигма програмування.

Abstract
The advantages of using asynchronous programming in software development were analyzed. Key features of asynchronous programming are described.
Keywords: software, asynchronous programming, programming paradigm.

Вступ

Традиційні підходи до програмування часто обмежуються виконанням задач одна за одною. Це може призвести до зниження продуктивності та негативно вплинути на користувацький досвід, особливо коли програма виконує довготривалі операції. Асинхронне програмування пропонує альтернативний підхід, який дозволяє програмі виконувати декілька задач одночасно, не блокуючи головний потік виконання. Цей підхід має ряд суттєвих переваг, які роблять його все більш популярним серед розробників.

Основна частина

Асинхронне програмування – це парадигма програмування, яка дозволяє виконувати декілька задач одночасно, не блокуючи головний потік виконання програми [1]. На відміну від синхронного програмування, де кожна задача виконується послідовно, асинхронні задачі можуть розпочинатися та виконуватися незалежно одна від одної. Це робить їх ідеальними для задач, які потребують значного часу на виконання, таких як обробка даних з файлів або введення/виведення з користувачем.

Способи реалізації асинхронного програмування на мові програмування C++:

1. Функції зворотного виклику або callbacks – це один із найпростіших і найпоширеніших способів обробки асинхронних операцій [2]. Вони передаються як параметри в асинхронні функції та викликаються після завершення виконання завдання. Callbacks використовуються для обробки результатів асинхронних операцій, але їх вкладення один в одного може призвести до проблем з читабельністю коду та керуванням помилками. Така практика може спричинити так зване "пекло колбеків", коли управління кодом стає надто складним і незручним.
2. Futures/Promises – це більш структурований і безпечний підхід до асинхронного програмування [3]. Future представляє собою результат асинхронної операції, а promise - об'єкт, який використовується для створення Future.
3. Корутини або coroutines в C++20 забезпечують ще більш сучасний підхід до асинхронного програмування, дозволяючи писати асинхронний код в стилі синхронного. Корутина — це функція, яка може призупинятися і продовжуватися з точки призупинення, що дозволяє писати асинхронний код, який виглядає як послідовний [4].

Асинхронне програмування є ключовим для сучасного програмного забезпечення, оскільки вони повинні обробляти великі обсяги даних та забезпечувати високу продуктивність і швидкий відгук користувацького інтерфейсу. У традиційному синхронному програмуванні тривала операція може заблокувати всю програму, роблячи її недоступною для користувачів. Натомість асинхронне

програмування дозволяє виконувати довготривалі операції у фоновому режимі без блокування головного потоку, що забезпечує швидкодію і чуйність програмного забезпечення.

Основні переваги асинхронного програмування [5]:

1. Підвищення продуктивності. Замість того, щоб чекати на завершення однієї задачі перед тим, щоб розпочати іншу, асинхронний підхід дозволяє програмі виконувати їх паралельно. Це призводить до більш ефективного використання ресурсів системи та покращення загальної швидкості виконання програми. Наприклад, користувальницький інтерфейс може залишатися чутливим під час виконання фонових операцій, таких як завантаження даних з мережі.

2. Масштабованість. Асинхронне програмування більш ефективно справляються з великим навантаженням. Традиційні програми, які використовують блокуючі операції, можуть "застрягати" при очікуванні на завершення довготривалих задач. Це призводить до зниження продуктивності та поганого користувацького досвіду, особливо для веб-сервісів, що обслуговують багато користувачів одночасно. Асинхронні застосунки, навпаки, можуть обробляти багато запитів одночасно, не блокуючись, що робить їх ідеальними для систем, які потребують високої масштабованості.

3. Краща комунікація з користувачем. Асинхронні операції дозволяють програмі реагувати на дії користувача без затримки. Наприклад, під час завантаження великого файлу, асинхронна програма може відображати індикатор прогресу та дозволяти користувачеві взаємодіяти з інтерфейсом, не чекаючи на завершення завантаження. Це створює більш приємний та чутливий користувацький досвід.

4. Зниження витрат ресурсів. Асинхронні програми можуть бути більш ефективними з точки зору використання ресурсів, особливо процесора. Традиційні програми, які використовують блокуючі операції, можуть витрачати значну кількість часу просто чекаючи на завершення цих операцій, натомість асинхронні програми, навпаки, можуть виконувати корисну роботу під час очікування, що призводить до зниження загального споживання ресурсів та покращення ефективності роботи програми.

Хоча асинхронне програмування пропонує ряд суттєвих переваг, воно також має певні недоліки, які слід враховувати при розробці програмного забезпечення [5]:

1. Асинхронний код може бути складнішим для написання та розуміння, порівняно з синхронним кодом. Це пов'язано з необхідністю керувати декількома потоками виконання, обробляти callbacks або promises, а також враховувати можливість виникнення гонок даних та інших проблем синхронізації.

2. Налагодження асинхронного коду може бути більш складним, порівняно з синхронним кодом. Це пов'язано з тим, що асинхронні операції можуть виконуватися в непередбачувані моменти часу, що ускладнює відстеження стану програми та виявлення помилок.

3. Зниження продуктивності. У деяких випадках асинхронне програмування може призвести до зниження продуктивності, особливо якщо використовується багато дрібних асинхронних задач. Це пов'язано з додатковими накладними витратами на перемикання контекстів та управління потоками виконання.

4. Вплив на тестування. Асинхронний код може ускладнити тестування програмного забезпечення. Це пов'язано з необхідністю емулювати асинхронну поведінку та писати тести, які стійкі до часових інтервалів.

Таким чином, перед прийняттям рішення про використання асинхронного програмування, рекомендується ретельно оцінити його переваги та недоліки, а також конкретні потреби програмного забезпечення. Зокрема, асинхронне програмування може значно підвищити продуктивність та ефективність обробки проєкту. Однак, варто пам’ятати, що асинхронне програмування може навпаки, ускладнити код, його налагодження, а також саму розробку без значної користі.

Проаналізувавши переваги застосування асинхронного програмування було прийнято рішення у застосуванні його для розробки програмного забезпечення системи дослідження алгоритму згладжування MSAA при різних параметрах графічної сцени [6]. Завдяки ньому розроблено асинхронне завантаження програмних модулів, що забезпечить швидкодію завантаження інтерфейсу користувача та графічної сцени.

Висновок

Дослідження показало, що при розробці програмного забезпечення асинхронне програмування має значні переваги. Ця парадигма програмування значно пришвидшує виконання роботи програми

завдяки можливості обробляти кілька операцій одночасно без блокування основного потоку виконання, ефективно керувати ресурсами системи та зменшувати час очікування на завершення повільних операцій, таких як введення-виведення, доступ до баз даних, читання і запис файлів, затримки при роботі з користувацьким інтерфейсом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Praseed Pai, Peter Abraham. «C++ Reactive Programming: Design Concurrent and Asynchronous Applications Using the RxCpp Library and Modern C++17» 2018 p. 87.

2. Callback (програмування). URL: https://uk.wikipedia.org/wiki/Callback_(програмування) (data of access 05.05.2024)

3. Futures and promises. URL: https://en.wikipedia.org/wiki/Futures_and_promises (data of access 05.05.2024)

4. Coroutine. URL: https://en.wikipedia.org/wiki/Coroutine (data of access 05.05.2024)

5. Synchronous And Asynchronous Programming - Core Differences URL : https://bitbytesoft.com/synchronous-and-asynchronous-programming/ (data of access 05.05.2024)

6. Колодій В.В. Аналіз методів згладжування / В.В Колодій, О.В. Романюк // Матеріали LIII Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця 2024. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20739

Колодій Владислав Віталійович – студент групи 4ПІ-20б, факультет інформаційних технологій та комп’ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: vladkolodiy2014@gmail.com

Романюк Оксана Володимирівна – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: romaniukoksanav@gmail.com

Vlad Kolodii – student of group 4PI-20b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: vladkolodiy2014@gmail.com

Oksana Romaniuk – Candidate of Technical Sciences, Associate Professor of the Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com