Ссылка на статью:
// Ученые записки УлГУ. Сер. Математика и информационные технологии. 2023, № 1, с. 158-164.

Поступила: 20.05.2023
Окончательный вариант: 01.06.2023

УДК 004.[05+032.24]

Повышение производительности игровых проектов при помощи асинхронного программирования

Филиппов К. А.

kirillfilippov991@yandex.ru
УлГУ, Ульяновск, Россия

В работе исследуется применение асинхронного программирования в контексте игровых проектов на платформе Unity. Выявляются его преимущества и недостатки, а также практическое применение в различных аспектах разработки игр. Особое внимание уделяется оптимизации производительности и улучшению отзывчивости игровых проектов с помощью асинхронных подходов. Анализируются существующие методы, примеры кода, а также рекомендации и руководства по использованию асинхронного программирования в игровых проектах на Unity.

Ключевые слова: *асинхронное программирование, Unity, улучшение производительности игровых проектов*

Введение

Несмотря на мощные возможности современных игровых движков, разработчики часто сталкиваются с проблемами производительности, которые могут существенно повлиять на удовлетворенность игроков. Долгие задержки, лаги и низкая отзывчивость игрового процесса могут разочаровать пользователей и снизить популярность игры.

Для преодоления этих вызовов и повышения производительности игровых проектов на платформе Unity разработчики активно применяют асинхронное программирование. Эта мощная техника программирования позволяет эффективно управлять параллельными операциями, не блокируя основной поток выполнения, и достигать оптимальной отзывчивости и производительности игры.

Одним из основных вызовов, с которыми сталкиваются разработчики игр, является обработка долгих операций, таких как загрузка ресурсов, обмен данными по сети или вы-

полнение сложных вычислений. Когда такие операции выполняются в основном потоке выполнения игры, это может привести к блокировке потока и задержкам в игровом процессе. Именно здесь асинхронное программирование становится незаменимым инструментом.

Асинхронное программирование позволяет разработчикам выполнять долгие операции параллельно с основным потоком игры, не блокируя его и не создавая задержек. Это достигается путем создания асинхронных методов и использования операторов, которые позволяют управлять асинхронными операциями. Параллельное выполнение операций повышает отзывчивость игры, устраняет блокировки основного потока и улучшает скорость загрузки ресурсов.

В данной работе рассмотрены различные подходы и инструменты, которые помогут использовать асинхронное программирование в игровых проектах на платформе Unity, а также выявлены лучшие практики и рекомендации по оптимизации и управлению асинхронными операциями.

1. Понятие асинхронного программирования

Асинхронное программирование — это подход к разработке программного кода, который позволяет выполнять задачи независимо друг от друга и параллельно с основным потоком выполнения программы. Основным преимуществом асинхронного программирования является возможность продолжать выполнение других задач во время ожидания длительных операций, таких как загрузка данных или обращение к внешним ресурсам [1].

В традиционном синхронном программировании задачи выполняются последовательно: одна задача должна быть завершена, прежде чем начнется следующая. Это может приводить к блокировке основного потока выполнения программы и замедлять ее работу. Особенно это заметно в случае длительных операций, которые требуют ожидания, например, при загрузке больших текстур или выполнении сложных вычислений.

Асинхронное программирование позволяет избежать блокировки основного потока выполнения путем делегирования задачи другому потоку или процессу. Таким образом, основной поток свободен для выполнения других задач, пока длительная операция выполняется в фоновом режиме. После завершения операции результат может быть получен и обработан, и программа может продолжить свою работу без простоев и задержек [1].

В синхронном программировании задачи выполняются последовательно, и каждая задача ждет завершения предыдущей, прежде чем начнется следующая. Это может приводить к задержкам и простоям.

В асинхронном программировании задачи могут выполняться параллельно и независимо друг от друга. Это позволяет улучшить производительность и отзывчивость программы, так как время ожидания длительных операций сокращается, а основной поток программы может продолжать выполнение других задач.

Для реализации асинхронного программирования в Unity используются различные подходы и инструменты, такие как асинхронные методы, операторы и паттерны. Они поз-

воляют эффективно работать с асинхронными операциями и управлять потоками выполнения.

Для наглядного сравнения были реализованы задачи с разным количеством входных данных для сравнительной диаграммы. В качестве поставленной задачи, было необходимо загрузить текстуры в игровой проект синхронно и асинхронно для сравнительной оценки:

1. Пример синхронного выполнения задачи: Задача заключалась в загрузке изображения с диска и присваивании его текстуре в основном потоке выполнения. В результате синхронной загрузки, основной поток блокируется до завершения операции загрузки, что может привести к замедлению работы приложения.
2. Пример асинхронного выполнения задачи: Задача была разделена на две части: начало загрузки текстуры в фоновом потоке и завершение загрузки и присваивание текстуры в основном потоке. Асинхронная загрузка позволяет основному потоку продолжать свою работу, не блокируясь, в то время как загрузка происходит в фоновом потоке. Это может повысить производительность и отзывчивость игрового проекта.

Таким образом, на примере синхронного и асинхронного выполнения задач иллюстрирована разница в подходах и влияние на производительность игровых проектов (рис. 1).

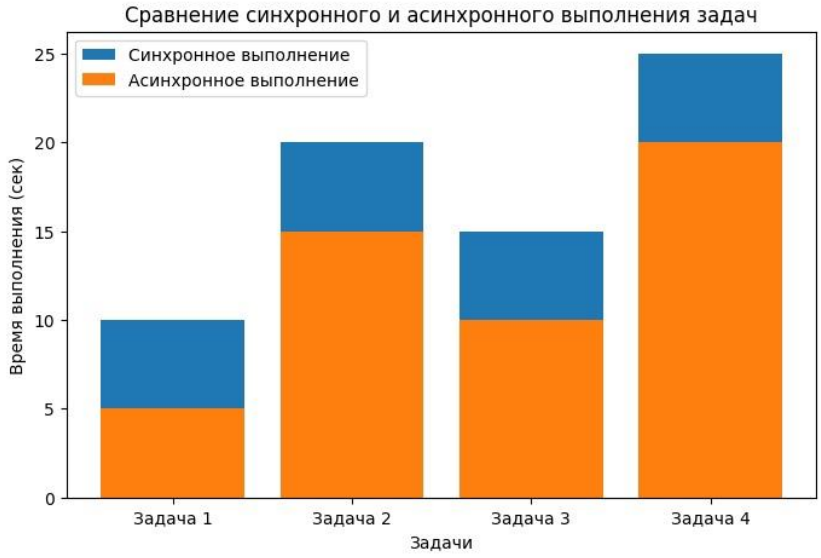

Рис. 1. Отчет о тестировании

2. Рекомендации по использованию асинхронного программирования

При применении асинхронного программирования в игровых проектах на платформе Unity следует учитывать несколько важных рекомендаций. Эти рекомендации помогают оптимизировать процесс разработки, улучшить производительность и обеспечить стабильную работу игры.

Необходимо идентифицировать задачи, которые могут быть выполнены асинхронно, чтобы избежать блокировки основного потока выполнения. Основной поток в Unity отвечает за отрисовку графики и обработку пользовательского ввода. Блокировка основного потока может привести к снижению отзывчивости игры и рывкам в графике. Поэтому

важно определить, какие операции могут быть выполнены асинхронно, чтобы основной поток оставался свободным для других задач [2].

Важно использовать асинхронные методы и операторы, предоставляемые Unity или сторонними библиотеками, для выполнения асинхронных операций. Unity предоставляет различные средства для асинхронного программирования, такие как корутины и асинхронные методы. Кроме того, существуют сторонние библиотеки, которые расширяют возможности асинхронного программирования в Unity. Использование этих инструментов позволяет эффективно организовать асинхронные операции и упростить разработку.

При разработке необходимо обратить внимание на правильную обработку ошибок и отмену асинхронных операций, чтобы избежать утечек ресурсов и непредвиденного поведения игры. При работе с кодом важно предусмотреть механизмы обработки ошибок и возможность отмены операций. Неправильная обработка ошибок может привести к непредсказуемым сбоям и нестабильности игры. Адекватная отмена асинхронных операций позволяет избежать утечек ресурсов и предотвратить выполнение ненужных операций [2].

Профилирование и измерение производительности необходимо, чтобы определить, где асинхронное программирование может принести наибольшую пользу. Используя инструменты профилирования, предоставляемые Unity, можно проанализировать производительность проекта. Они позволят идентифицировать узкие места, где асинхронное программирование может существенно улучшить производительность и оптимизировать работу игры (табл. 1).

Таблица 1. Примеры применения асинхронного программирования

Задача	Асинхронный подход	Преимущества
Загрузка ресурсов	Асинхронная загрузка	Повышение отзывчивости, улучшенное управление памятью, сокращение времени ожидания
Сетевые операции	Асинхронные запросы к серверу	Более плавная работа с сетью, снижение блокировки основного потока
Анимации и эффекты	Асинхронное проигрывание	Повышение плавности анимаций и эффектов, избежание задержек
Компоненты и системы игровых объектов	Асинхронные обновления	Более эффективное использование вычислительных ресурсов, улучшение производительности

Применение асинхронного программирования в Unity позволяет значительно улучшить производительность игровых проектов, повысить отзывчивость и достичь более плавной работы игры. Следование рекомендациям по использованию асинхронного программирования, идентификация задач, подлежащих асинхронному выполнению, и использование соответствующих методов и операторов поможет создать высококачественный и оптимизированный игровой проект в Unity.

Асинхронное программирование в контексте игровых проектов на Unity можно представить следующей моделью:

1. *Множество задач* P: в игровом проекте существует набор задач, которые нужно выполнить. Обозначим это множество как $P = \{p1, p2, \dots, pn\}$, где pi - отдельная задача.
2. *Время выполнения задач* $T(p)$: каждая задача pi имеет свое время выполнения, обозначаемое как $T(p)$. Это время может быть различным для каждой задачи и зависит от ее сложности и требуемых ресурсов.
3. *Ресурсы выполнения* C: для выполнения задач используются вычислительные ресурсы, представленные множеством $C = \{c1, c2, \dots, cm\}$. Каждый ресурс ci обладает определенными вычислительными возможностями.
4. *Пропускная способность ресурсов* $B(c)$: у каждого ресурса ci есть ограниченная пропускная способность $B(ci)$, которая определяет, сколько задач он может обработать одновременно.

Целью асинхронного программирования является эффективное распределение задач из множества P на доступные ресурсы из множества C таким образом, чтобы общее время выполнения всех задач было минимальным, учитывая ограничения пропускной способности ресурсов и зависимости между задачами.

3. Многопоточность и асинхронные задачи

Одним из вариантов использования многопоточности и асинхронных задач в Unity является распараллеливание вычислений. Например, при работе с физикой или сложными математическими операциями можно разделить вычисления на независимые фрагменты и выполнять их параллельно на разных потоках. Это позволит сократить общее время выполнения и улучшить производительность игры [3,5].

Другим примером применения многопоточности является загрузка ресурсов. Unity позволяет асинхронно загружать текстуры, модели, звуки и другие ресурсы, в фоновом режиме, используя специальные методы и асинхронные операции. Это позволяет избежать простоев и улучшить отзывчивость игры, особенно при работе с большими объемами данных.

Для реализации многопоточности и асинхронных задач в Unity также могут быть использованы пулы потоков. Пулы потоков представляют собой набор предварительно созданных потоков, которые могут быть повторно использованы для выполнения различных задач. Использование пулов потоков помогает избежать излишней нагрузки на систему созданием и уничтожением потоков и способствует более эффективному распределению вычислительной работы (табл. 2).

С использованием многопоточности и асинхронных задач в Unity можно эффективно распараллеливать выполнение задач и повысить производительность игрового проекта. Однако, необходимо тщательно планировать и управлять потоками, чтобы избежать проблем с синхронизацией и гонками данных. Также следует учитывать особенности платформы, на которой будет запускаться игра, чтобы достичь оптимальной производительности и совместимости.

Таблица 2. Сравнение между синхронным и асинхронным выполнением задач

Тип	Синхронное выполнение	Асинхронное выполнение
Ожидание задач	Блокирует основной поток	Не блокирует основной поток
Использование ресурсов	Неэффективное использование ресурсов	Эффективное использование ресурсов
Отзывчивость игры	Может быть низкой из-за блокировки потока	Высокая отзывчивость благодаря неблокирующему потоку
Производительность	Может быть низкой из-за последовательного выполнения задач	Повышается благодаря параллельному выполнению задач

4. Практическое применение асинхронного программирования

Задача: Загрузка больших объемов данных в игре.

Описание: В игровых проектах часто возникает необходимость загрузки больших объемов данных, таких как текстуры, модели, аудиофайлы и прочее. При синхронной загрузке таких данных основной поток выполнения блокируется до завершения операции, что может привести к заметным задержкам и падению производительности игры. Для решения этой проблемы можно использовать асинхронное программирование.

Решение: Применение асинхронного программирования позволяет загружать данные параллельно основному потоку выполнения игры. Вместо блокировки основного потока на время загрузки, асинхронные операции могут выполняться в фоновом режиме, позволяя игрокам продолжать взаимодействие с игрой без задержек.

Пример: для реализации асинхронной загрузки данных в Unity можно использовать функционал таких компонентов, как WWW, UnityWebRequest или AssetBundle. Они предоставляют возможность асинхронной загрузки данных с использованием колбэков или корутин. Также можно воспользоваться асинхронными операциями и операторами, предоставляемыми языком C# и Unity, такими как async/await, Task, AsyncOperation и другие [4,6].

В ходе работы создан класс DataLoader, который отвечает за загрузку данных. В методе LoadDataAsync() создается экземпляр UnityWebRequest и выполнятся асинхроннаю загрузку данных с использованием yield return request.SendWebRequest(). После завершения загрузки происходит проверка результата и, в случае успеха, обрабатываются загруженные данные в методе ProcessData().

Таким образом, применение асинхронного программирования в задаче загрузки данных позволяет повысить производительность игрового проекта на Unity и обеспечить более комфортное игровое впечатление для пользователей.

Заключение

В работе были произведены исследования применение асинхронного программирования в игровых проектах на Unity и обнаружено его значительное влияние на производительность и отзывчивость игр. Асинхронное программирование позволяет эффективно работать с асинхронными операциями, улучшать загрузку ресурсов и повышать качество игрового опыта. Одной из практических задач, решаемых с помощью асинхронного программирования, является оптимизация процесса загрузки игровых ресурсов. Однако успешное применение асинхронного программирования требует грамотного планирования, проектирования и тестирования.

Список литературы

1. Албахари Дж., Албахари Б. *Асинхронное программирование в C# 5.0.* Вильямс, 2013. 240 с.
2. Джонсон Дж. *Асинхронное программирование с использованием async/await в C# 5.0.* ДМК Пресс, 2013. 119 с.
3. Ралф Х., Купер Э. *Unity в действии: мультиплатформенная разработка игр.* ДМК Пресс, 2015. 350 с.
4. Unity Documentation. Режим доступа: https://docs.unity3d.com/ (дата обращения 20.04.2023).
5. Клири С. *Конкурентность в C#. Асинхронное, параллельное и многопоточное программирование.* O'Reilly, 2020. 304 с.
6. Async/await в ASP.NET Core. Асинхронный код. Режим доступа: https://alekseev74.ru/lessons/show/aspnet-core-mvc/async-await (дата обращения 20.04.2023).

Improving game project performance with asynchronous programming

Filippov, K. A.

kirillfilippov991@yandex.ru

Ulyanovsk State University, Russia

The paper studies the use of asynchronous programming in the context of game projects on the Unity platform. Its advantages and disadvantages are revealed, as well as its practical application in various aspects of game development. Particular attention is paid to optimizing performance and improving the responsiveness of game projects using asynchronous approaches. Existing methods, code examples, as well as recommendations and guidelines for using asynchronous programming in game projects on Unity are analyzed.

***Keywords**: asynchronous programming, Unity, improving the performance of game projects*