Инженерный вестник Дона, №9 (2024)
ivdon.ru/ru/magazine/archive/n9y2024/9516

Архитектурный паттерн ESC как способ реализации объектно-ориентированного подхода в программировании

О.В. Игнатьева, О.Г. Ведерникова, Н.А. Москат
Ростовский государственный университет путей сообщения

Аннотация: Представлен способ программирования на основе паттерна ESC. Актуальность работы связана с тем, что объектно-ориентированный подход является одним из самых популярных и востребованных способов разработки информационного продукта благодаря огромному регулярно обновляемому выбору различных методов, шаблонов и способов его реализации. Наиболее значимый из них - Entity System Component (ESC). Данный метод реализации ООП позволяет делать программный продукт гибким и расширяемым. Паттерн ESC базируется на методе реактивного программирования и разделяет всю архитектуру кода на три составляющие: сущность, система, компонент. Пакетом инструментов, реализующих паттерн ESC, является пакет ESC DOTS, предназначенный для среды Unity3D. Встроенный пакет Jobs System предоставляет возможность работы с многопоточным программированием в Unity. Этот пакет распределяет созданные во время исполнения потоки на группы определенного типа, которые имеют строго ограниченное время исполнения. Так, задача перебора массива из несколько сотен элементов переходит в поток типа Temp, который выполняется в течение одного кадра в Unity, а перебор миллиона полигонов ландшафта Unity размещается в потоке типа Persistent, имеющем неограниченный лимит на время.
Ключевые слова: объектно-ориентированное программирование, среда Unity, паттерн ESC, многопоточное программирование, реактивное программирование, расширяемая архитектура, менеджер пакета.

Современная методика проектирования программы – объектно-ориентированный подход. В объектно-ориентированном программировании программа разрабатывается с использованием набора взаимодействующих объектов. Объект – это компонент программы, имеющий набор методов и структуру данных. Методы предназначены и используются для доступа к данным объекта и /или для преобразования данных. После того, как объект для любой программы спроектирован, его можно повторно использовать в любой другой программе [1, 2]. В этом основное преимущество объектно-ориентированного подхода.

На сегодняшний день объектно-ориентированный подход (ООП) является одним из самых популярных и часто используемых в сфере разработки информационного продукта. Объектно-ориентированный подход

постоянно остается востребованным благодаря огромному регулярно обновляемому выбору различных методов, шаблонов и способов его реализации [3, 4].

Однако, в результате такого большого выбора, часто возникает проблема, когда какой-то подход не соответствует потребностям разработки и ухудшает гибкость и расширяемость программного продукта [5]. Так, например, событийно-ориентированный подход на основе ООП замедляет и усложняет процесс разработки, так как создает множество неявных уязвимостей, которые могут конфликтовать с введенным дополнительным функционалом продукта [6]. Тем не менее, существует ряд шаблонов или паттернов, способных разработать гибкую и масштабируемую архитектуру, которую можно регулярно обновлять на любом этапе разработки [7].

Предметом данной работы является сущностный подход на основе ООП. Один из наиболее значимых шаблонов на данный момент – это шаблон Entity System Component (ESC). Этот термин можно перевести как «Сущность-Система-Компонент», именно эти три сущности и представляют работу всей архитектуры. Данный метод реализации ООП позволяет делать программный продукт более гибким, расширяемым и простым в разработке. При этом данный шаблон ESC или паттерн базируется на методе реактивного программирования, что позволяет во многих аспектах сделать код максимально оптимизированным [8].

Архитектурный паттерн - Entity Component System (ECS), который ставит во главу угла данные, а не объекты, и тем самым переворачивает стандартное представление о программировании в методологии ООП [9]. Подобный подход развивает идею композиции над наследованием и позволяет легко адаптироваться под динамичные потребности гейм-дизайнера. ESC – это шаблон ООП, разделяющий всю архитектуру кода на

три составляющие: сущность, система, компонент. Общая схема данного подхода представлена на рисунке 1.

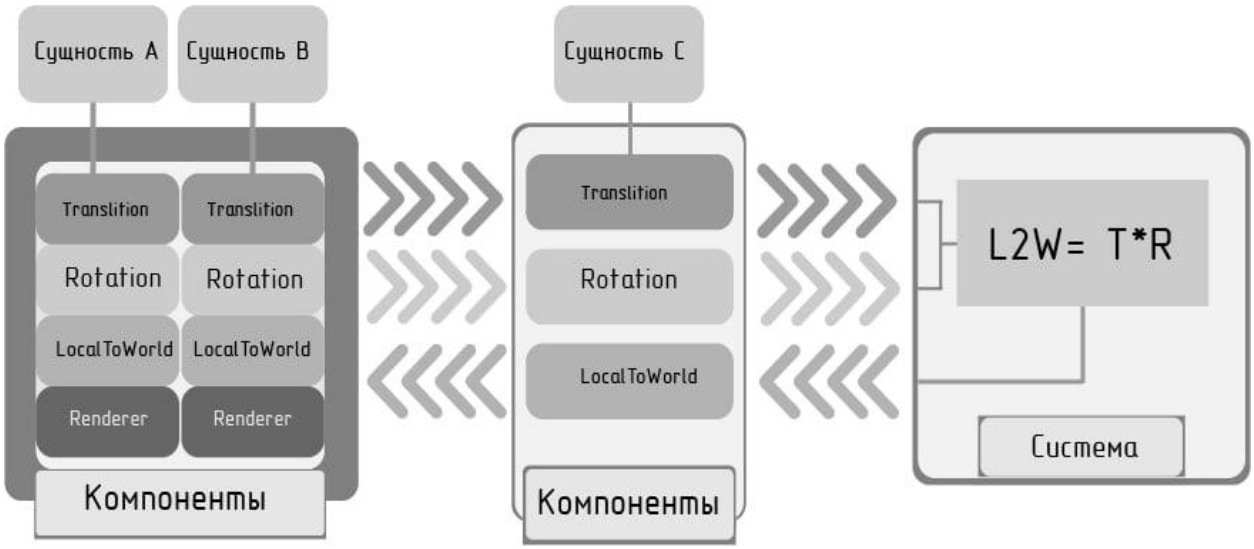

Рис. 1. – Общая схема подхода шаблона Entity System Component

Сущность в ECS – это, по сути, пустой указатель на набор компонентов (Components), которые привязываются к конкретной сущности (Entity). А компоненты – это набор данных: например, для расчета гравитации это ускорение, для определения позиции – это координаты игрового объекта (transition, rotation) [10]. Вся логика содержится в системах (System) – это классы, выполняющие какую-то конкретную задачу, но делающие это особым образом [11].

Таким образом, системы, в которых реализована основная логика продукта, обращаются к сущностям через настройку фильтрации компонентов и работают только с теми данными, которые им нужны. Такой шаблон дает возможность в будущем быстро и без затруднений изменить логику необходимых систем, не затронув при этом остальные части кода. Тем не менее, настройка фильтрации и строгое разделение кода формируют

высокий порог входа, что, естественно, замедляет процесс обучения данному шаблону.

При реализации программных продуктов посредством шаблона ESC выделяют следующие характеристики архитектуры:

1. Эффективность использования памяти – этот подход подразумевает собственный распределитель памяти, эффективно экономящий пространство для динамических объектов.
2. Журналирование. Данная характеристика представляет собой реализацию удобной системы записи происходящего в программном продукте.
3. Масштабируемость – означает, что шаблон ESC предоставляет возможность добавлять новые компоненты, сущности или системы, не затрагивая при этом остальные части кода.
4. Гибкость – этот подход подразумевает, что между сущностями, компонентами и системами нет никаких зависимостей.
5. Эффективный поиск объектов – это реализация простой и минимально затратной по времени системы поиска необходимых объектов программы.
6. Контроль за выполнением – данная характеристика связанна с тем, что системы ка правило имеют приоритеты, поэтому предоставляется возможность установки топологического порядка выполнения.
7. Простота использования – другими словами, нет ограничений на совместимость с другими подходами ООП.

Описанные выше характеристики архитектуры шаблона ESC визуально представлены на рисунке 2.

Рис. 2. – Характеристики архитектуры шаблона ESC

На сегодняшний день не возникает острой необходимости в разработке шаблона ESC с нуля, так как уже сейчас существует множество источников, которые предоставляют мощные инструменты, реализующих данный шаблон. Одним из таких является ESC DOTS.

ESC DOTS – это пакет инструментов, предназначенных для среды Unity3D, реализующий паттерн ESC. Этот пакет очень прост в освоении и не требует каких-либо ограничений для разрабатываемого продукта. Так, для того, чтобы создать сущность с необходимым набором компонентов достаточно обратиться к менеджеру пакета и вызвать метод CreateEntity (Создать Сущность). К тому же, ESC DOTS имеет особый класс EntityArchetype (Архетип Сущности), позволяющий группировать схожие по набору компонентов сущности. Наконец, пакет имеет свою собственную нативную систему QueryEntity (Сущность Запроса), способную напрямую через ядро Unity находить запрашиваемые сущности. Такая система работает

гораздо быстрее, чем обычная реализация поиска объектов в Unity API [12, 13].

Чтобы реализовать компонент, необходимо создать класс, который наследовался бы от IComponentData. А для систем необходим родительский класс SystemBase, который реализует функционал взаимодействия системы с движком шаблона ESC.

Еще одним из значимых инструментов является встроенный пакет Jobs System, который предоставляет возможность простой работы с многопоточным программированием в Unity. Структура работы показана на рисунке 3.

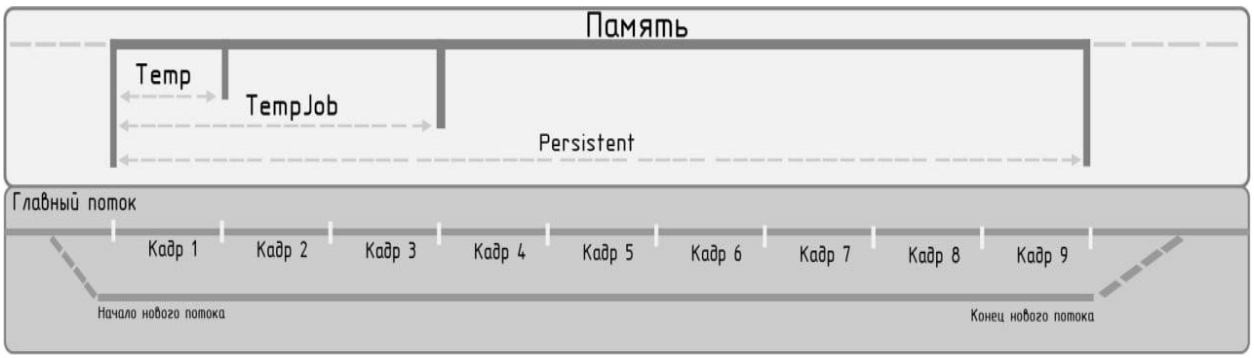

Рис. 3. – Структура работы с многопоточным программированием

На этом рисунке видно, что Jobs System распределяет созданные во время исполнения потоки на группы определенного типа. Каждый тип потока в свою очередь имеет строго ограниченное время исполнения. Такой подход дает возможность группировать различные задачи по значимости и не нагружать устройство созданием тяжелых потоков, как, к примеру, Task<T> в C#. Например, задача перебора массива из несколько сотен элементов может уйти в поток типа Temp, который выполняется всего в течение одного кадра в Unity. А перебор миллиона полигонов ландшафта можно разместить

в потоке типа Persistent (Постоянный), имеющем неограниченный лимит на время.

Таким образом, предоставляемый подход является актуальным в современной разработке программного обеспечения. Данный шаблон создает достаточно гибкую и расширяемую архитектуру. К таким результатам можно прийти благодаря использованию предоставляемой базы программных интерфейсов. При этом данная технология приносит и ряд дополнительных положительных результатов, таких, как – отказоустойчивость, ускоренная работа продукта и экономия памяти на носимом устройстве.

Литература

1. Бутакова, М.А. А.Н. Гуда, А.В. Чернов Теоретические аспекты визуальной разработки имитационных моделей проблемно-ориентированных информационных систем // Программные продукты, системы и алгоритмы, 2014. №4. URL: swsys-web.ru/theoretical-aspects-of-visual-development-of-simulation-models.html. EDN: VKRIQR

2. Игнатьева О.В., Москат Н.А., Рубцова С.С. Архитектурные приемы при разработке программного обеспечения, зависимого от интерфейса пользователя // Инженерный вестник Дона, 2022, № 2. URL: ivdon.ru/ru/magazine/archive/n2y2022/7478

3. Валеев С.С., Кондратьева Н.В. Паттерны проектирования архитектуры нулевого доверия // Инженерный вестник Дона, 2023, №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8674

4. Кучеров С.А. Шаблон архитектуры распределенной системы, основанной на технологии блокчейн // Инженерный вестник Дона, 2022 №11. URL: ivdon.ru/ru/magazine/archive/n11y2022/8035

5. Ведерникова О. Г., Москат Н. А. Расширение и использование редактора визуального программирования для разработки виртуальных тренажеров // Инженерный вестник Дона. 2021. № 1. URL: ivdon.ru/ru/magazine/archive/n1y2021/6783.

6. Ведерникова О. Г., Игнатьева О. В. Сравнение подхода классического программирования и визуального программирования при разработке шейдеров исчезновения // Транспорт: наука, образование, производство («Транспорт-2020»). Технические науки. Ростов-на-Дону, Изд-во РГУПС, 2020. С. 43–48.

7. Тепляков, С. Паттерны проектирования на платформе NET. СПб.: Питер, 2015. 320 с.

8. Абрахам, П. Реактивное программирование. М.: ДМК 2019. 324 с.

9. Мартин, Р.С. Принципы, паттерны и методики гибкой разработки на языке C#. СПб.: Питер, 2011. 768 с.

10. Hocking Joseph. Multiplatporm Game Development in C#. Manning. Питер. 2019. С. 21.

11. Хокинг, Д. Unity в действии. Мультиплатформенная разработка на C#. СПб.: Питер, 2016. 336 с.

12. Sung K., Gregory S. Basic Math for Game Development with Unity 3D. - New York: Springer Science + Business Media, 2019. 414 p.

13. Patrick Felicia. Unity From Zero to Proficiency (Foundations). 2019. 412 p.

References

1. Butakova M.A., Guda A.N., Chernov A.V. Programmnye produkty, sistemy i algoritmy, 2014, №4. URL: swsys-web.ru/theoretical-aspects-of-visual-development-of-simulation-models.html.

2. Ignat'eva O.V., Moskat N.A., Rubtsova S.S. Inzhenernyj vestnik Dona, 2022, № 2. URL: ivdon.ru/ru/magazine/archive/n2y2022/7478

3. Valeev S.S., Kondrat'eva N.V. Inzhenernyj vestnik Dona, 2023, №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8674

4. Kucherov S.A., Lipko Yu.Yu, Rogozov Yu.I., Sviridov A.S. Inzhenernyj vestnik Dona, 2022, №11. URL: ivdon.ru/ru/magazine/archive/n11y2022/8035

5. Vedernikova O.G, Moskat N.A. Inzhenernyj vestnik Dona, 2021. № 1 URL: ivdon.ru/ru/magazine/archive/n1y2021/6783

6. Vedernikova O. G., Ignatieva O. V. Transport: nauka, obrazovanie, proizvodstvo («Transport-2020»). Texnicheskie nauki. Rostov-na-Donu, Izd-vo RGUPS, 2020. Pp. 43–48.

7. Teplyakov, S. Patterny proektirovaniya na platforme NET [Design patterns on the platform NET]. SPb.: Piter, 2015. 320 p.

8. Abrakham, P. Reaktivnoe programmirovanie [Reactive programming]. M.: DMK 2019. 324 p.

9. Martin, R.S. Printsipy, patterny i metodiki gibkoy razrabotki na yazyke C# [Principles, patterns and techniques of agile development in C#]. SPb.: Piter, 2011. 768 p.

10. Hocking Joseph. Multiplatporm Game Development in C#. Manning. Питер. 2019. p. 21.

11. Hocking J. Unity v deystvii. Mul′tiplatformennaya razrabotka na C# [Unity in action. Multi-platform development in C#]. SPb.: Piter, 2016. 336 p.

12. Sung K., Gregory S. Basic Math for Game Development with Unity 3D. - New York: Springer Science + Business Media, 2019. 414 p.

13. Patrick Felicia. Unity From Zero to Proficiency (Foundations). 2019. 412 p.

Дата поступления: 11.08.2024

Дата публикации: 20.09.2024