ECS-подход в разработке компьютерных игр

А. А. Введенский, e-mail: artvved2000@mail.ru

Воронежский Государственный Университет

Аннотация. *Рассмотрен архитектурный паттерн Entity-Component-System, его структура и особенности, выявлены положительные и отрицательные свойства, определены полезные практики использования.*

Ключевые слова: *архитектурный паттерн, Entity-Component-System, разработка игр.*

Введение

В прошлом веке в повседневную жизнь человека и в сферы его деятельности начали проникать различные информационные системы. Разработка таких систем, развиваясь, приобретала некоторые черты: формулировались основные принципы, появлялись методологии, как например: “Водопадная модель”, “V-модель”. Но с ростом плохо структурированных приложений без разделения логики становилась понятна их запутанность и невозможность поддержки из-за большой траты ресурсов, поэтому нередко разросшиеся приложения приходилось переписывать с нуля. Чтобы решить проблему с разделением обязанностей частей кода и улучшить его повторную используемость был предложен архитектурный паттерн MVC. Стали появляться и другие паттерны, популярность которых увеличилась после публикации книги Design Patterns: Elements of Reusable Object-Oriented Software.

Архитектурный паттерн – это общее, многократно используемое решение часто встречающейся проблемы в архитектуре программного обеспечения в данном контексте [1]. В контексте разработки компьютерных игр также используются различные паттерны [2]. Нередко можно встретить реализацию с использованием MVC, в игровом движке Unity предлагается паттерн Entity-Component, в котором объекты на сцене выступают в роли сущностей, а компоненты хранят данные и реализуют логику работы с ними. Каждый компонент, будучи отдельным объектом со своим API и ожидаемым поведением, зачастую сам обрабатывает или изменяет свои данные по чьему-либо обращению. Под специфичную задачу разработки видеоигр создаются собственные паттерны. Так, в 2007 году был создан новый паттерн Entity-Component-System (ECS).

ECS – это архитектурный паттерн, созданный для описания динамического виртуального мира, в силу своей специфики он отлично подходит под задачу разработки игр [3]. Имеет место следующая актуальная задача – рассмотрение структуры, особенностей архитектурного паттерна ECS, выявление его положительных и отрицательных свойств, определение полезных практик его использования.

1. Паттерн ECS и его структура

Расшифруем название паттерна:

– Entity – сущность, представляющая из себя абстрактный объект. Хранилище для свойств, которые определяют, чем будет являться эта сущность. Зачастую представляется в виде идентификатора для доступа к данным.

– Component – компонент, свойство с данными объекта. В ECS делается акцент на то, что компонент содержит только чистые данные, без логики работы с ними.

– System – система, логика обработки данных. В ECS важно, что системы должны содержать только логику работы с данными, но не сами данные.

На рис. 1 представлена структура рассматриваемого паттерна.

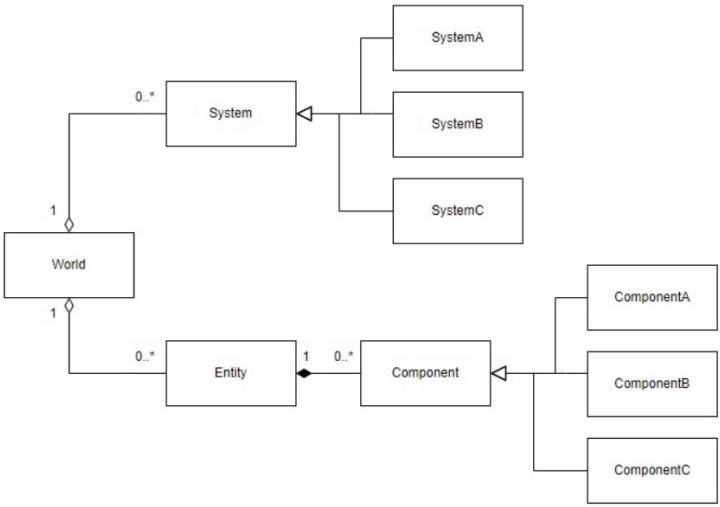

Рис. 1. Структура паттерна ECS

Согласно приведенной структуре, ECS-мир содержит в себе коллекции систем и сущностей. Сущности содержат изменяемый набор компонентов. Доступ из систем к набору сущностей, содержащих определенный набор компонентов, предоставляется публичными методами ECS-мира. Специфика вызова этих методов зависит от реализации в конкретном ECS-фреймворке, но в большинстве из них получение доступа к определенному набору сущностей схоже с формированием запроса к базе данных.

В Entity-Component-System данные строго отделены от логики. Поведение объекта определяется не интерфейсами, как в классическом объектно-ориентированном программировании, а присвоенными объекту свойствами с данными и существующей отдельно логикой обработки этих данных. Главное свойство ECS, которое выделяет его на фоне других подходов к разработке – всё есть данные [4]. Данными выражены и свойства объекта, и его характеристики, и события. Такой подход поможет соблюдать принцип Composition Over Inheritance [5].

Логика реализуется конвейерной обработкой данных. В каждой итерации игрового цикла системы последовательно получают доступ к определенным компонентам сущностей и изменяют данные этих компонентов. Таким образом, порядок выполнения систем играет большую роль. На рис. 2 представлена конвейерная схема работы систем.

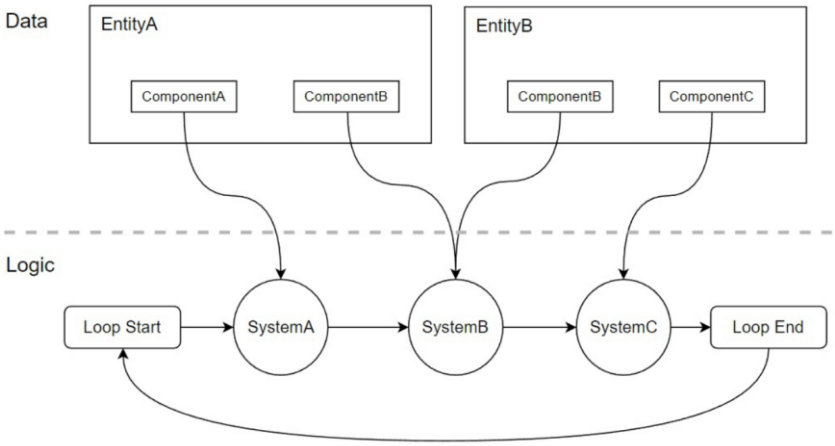

Рис. 2. Конвейерная схема работы систем

2. Достоинства и недостатки ECS

У каждого архитектурного паттерна есть свои достоинства и недостатки, и ECS не является исключением. Рассмотрим некоторые преимущества данного подхода:

– Слабое сопряжение кода – хорошее свойство кода, выражающееся в отсутствии необходимости изменения классов при изменении сопряженных классов [6]. Это свойство полезно при разработке игр, так как позволит легко расширить кодовую базу добавлением новой логики без изменения старых частей.

– Хорошая модульность и тестируемость логики. В ECS логическая часть работает только с чистыми данными, поэтому она отвязана от их источника. Это свойство позволит облегчить как повторное использование логики в другом проекте, так и ее тестирование.

– Комбинаторика свойств. Это достоинство паттерна позволит проще изменять и добавлять игровую логику под постоянно меняющиеся требования геймдизайнеров.

– Простота в соблюдении принципа единой ответственности. Так как логика не привязана к какому-либо объекту, то становится проще разбивать ее по назначению и не привязываться к ее месту в иерархии. Каждая система выполняет конкретную задачу, свойственную только ей.

Рассмотрим некоторые недостатки данного подхода:

– Публичный доступ к данным. В ECS данные компонентов каждой сущности публично доступны. Таким образом, их можно изменить в любом месте программы, что может привести к некорректности ее работы, а произойти из-за неопытности разработчиков или их некачественного взаимодействия внутри команды.

– Важна независимость систем. Системы работают только с данными, не знают о существовании других систем и не могут к ним обращаться.

– Важен порядок выполнения систем. Системы выполняются последовательно в заданном порядке, поэтому важно учитывать влияние конкретной системы на данные при выборе ее очередности.

– Большое количество классов. Количество файлов в проекте с использованием ECS растет быстрее, чем в проектах, использующих классические подходы, так как вместо одного класса с данными и логикой приходится создавать минимум один компонент и минимум одну систему.

3. Хорошие практики при использовании ECS

За время существования ECS сформировались некоторые полезные практики его использования, которые помогают бороться с

недостатками паттерна и грамотно использовать его достоинства. Рассмотрим некоторые из них:

– Маркировка сущностей. Чтобы помечать определенные сущности в ECS, можно добавить этой сущности компонент без полей – компонент-маркер. Благодаря этому компоненту можно будет легко найти нужные сущности.

– Распределение классов в проекте по функциональности, а не по типу. Распределение по типу приведет к усложнению поиска нужного класса, так как количество файлов в ECS больше по сравнению с классическим подходом.

– Отложенная реактивность. Вместо моментальной реакции на возникшее событие происходит создание компонента, который обозначает возникновение этого события. Благодаря данному компоненту определенные системы будут уведомлены о произошедшем событии и смогут на него отреагировать.

Заключение

Рассмотрен архитектурный паттерн Entity-Component-System, его структура и особенности, выявлены положительные и отрицательные свойства, определены полезные практики использования.

Паттерн является ориентированным на данные, имеет строгое распределение обязанностей частей своей структуры. Благодаря его особенностям и положительным характеристикам рассмотренный архитектурный паттерн является достойной альтернативой существующим и удобным инструментом для описания динамического виртуального мира, что делает его востребованным в области разработки компьютерных игр.

Литература

1. Архитектурный паттерн [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Architectural_pattern

2. Nystrom, R. Game Programming Patterns / R. Nystrom. – New York: Genever Benning, 2014. – 8 p.

3. Основные принципы использования ECS в разработке игр [Электронный ресурс]. – Режим доступа: https://vc.ru/education/118840-osnovnye-principy-ispolzovaniya-ecs-v-razrabotke-igr

4. Всё что нужно знать про ECS [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/665276/

5. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma [et al.]. – Boston: Addison-Wesley, 1995. – 31 p.

6. McConnell, S. Code Complete / S. McConnell. – 2nd ed. – Redmond: Microsoft Press, 2004. – 80 p.