

УДК 004.415.2+004.946

Опыт применения паттернов проектирования при разработке виртуального тренажера - симулятора металлорежущих станков на Unity

Харитонов Лев Сергеевич^(*)

leo.khariton@yandex.ru

SPIN-код: 5666-3965

Раскаткин Ярослав Алексеевич

raskatkin.yarosl@mail.ru

Эварт Татьяна Евгеньевна

t.zhilina@bk.ru

АПИ НГТУ им. Р.Е. Алексеева, Арзамас, Россия

Аннотация. Рассмотрен опыт использования шаблонов проектирования в процессе разработки виртуального тренажера-симулятора металлорежущих станков «Эсман» с помощью Unity 3D. Описывается применение таких паттернов, как «Наблюдатель», «Машина состояний» и «Объектный пул». Их внедрение позволило сделать код приложения более расширяемым, производительным и гибким. Статья может быть полезна разработчикам виртуальных тренажеров, симуляторов и игр на платформе Unity.

Ключевые слова: паттерны проектирования, тренажер, симулятор, виртуальная реальность, архитектура программного обеспечения, иммерсивные технологии, Unity

Experience in applying design patterns in the development of a metal-cutting machines virtual simulator using unity

Kharitonov Lev Sergeevich^(*)

leo.khariton@yandex.ru

SPIN-code: 5666-3965

Raskatkin Yaroslav Alekseevich

raskatkin.yarosl@mail.ru

Evart Tatiana Evgenievna

t.zhilina@bk.ru

Arzamas Polytechnic Institute of R.E. Alekseev NNSTU, Arzamas, Russia

Abstract. This article discusses the experience of using design patterns in the development of the “Esiman” virtual training simulator of metal-cutting machines using Unity 3D. The application of patterns such as “Observer”, “State Machine” and “Object Pool” is described. Their implementation allowed the application code to become more extensible, performant, and flexible. This article may be useful for developers of virtual imitators, simulators, and games based on the Unity platform.

Keywords: design patterns, simulator, imitator, virtual reality, software architecture, immersive technologies, Unity

Введение. Разработка виртуальных тренажеров и симуляторов является актуальным и перспективным направлением развития образовательных технологий. Они обеспечивают безопасную виртуальную среду для получения реали-

стичного опыта, позволяя многократно повторять обучающие упражнения и моделировать различные, в том числе опасные ситуации. Применение тренажеров востребовано не только в образовательных учреждениях, но и в корпоративном секторе, где многие предприятия используют их для обучения персонала.

Разработка виртуального симулятора требует грамотного проектирования его архитектуры, так как от организации кода и компонентов зависит производительность, гибкость, расширяемость, масштабируемость, удобство тестирования и сопровождения проекта.

При разработке программной архитектуры часто используются шаблоны проектирования (англ. design patterns — паттерны проектирования), повторяемые конструкции в сфере проектирования программного обеспечения, предлагающие решение задачи в рамках некоторого часто возникающего контекста. Классическим собранием паттернов является знаменитая книга [1] «Банды четырех» (англ. the Gang of Four).

Внедрение паттернов может значительно улучшить восприятие кода и уменьшить количество ошибок при его написании. Однако всегда важно оценивать необходимость использования шаблонов, так как для небольших проектов их применение может быть избыточным и лишь усложняющим. Использование паттернов становится эффективным, когда количество программных компонентов увеличивается, а управление ими усложняется.

Тема работы является актуальной, поскольку, несмотря на популярность разработки обучающих VR-симуляторов в среде Unity, опыт применения паттернов проектирования в этой области недостаточно освещен в литературе.

Цель данной работы — рассмотреть применение различных паттернов проектирования при разработке архитектуры виртуального тренажера-симулятора металлорежущих станков «Эсиман» на платформе Unity.

Материалы и методы. Основным инструментом для разработки симулятора является движок Unity. Поскольку используется готовый фреймворк, на «низком» уровне архитектура VR-проекта уже сформирована: есть связанные между собой системы рендеринга, физики, звука, виртуальной реальности, скриптов, многопоточности и т. д. Unity предоставляет реализацию ряда паттернов «из коробки»: это «Игровой цикл» (Game Loop), «Метод обновления» (Update Method), «Компонент» (Component) и др. [2]. Для программирования на Unity используется язык C#. В движке реализован компонентно-ориентированный подход (базовый класс MonoBehaviour служит ярким примером шаблона проектирования «Компонент» [2]). На сцене в Unity размещаются игровые объекты (GameObject), к которым прикрепляются компоненты: Transform для задания позиционирования, MeshFilter — 3D-сетки, MeshRenderer — визуального материала, Collider — границ взаимодействий и т. д. Также к игровым объектам можно прикрепить собственные C#-сценарии, унаследованные от класса MonoBehaviour. При разработке проекта файлы удобно организовывать по папкам, разделяющим ресурсы по типам: скрипты, модели, материалы, текстуры, шейдеры, аудио и пр.



Рис. 1. Скриншот виртуальной сцены симулятора

Однако даже при использовании готового игрового движка возникает необходимость применения различных архитектурных подходов и паттернов. Данная статья описывает опыт применения шаблонов проектирования при разработке виртуального тренажера-симулятора металлорежущих станков «Эсман» [3], скриншот сцены которого представлен на рис. 1.

Разработанный симулятор имеет реалистичную графику, звук и взаимодействия в режиме реального времени. Также ключевым моментом является поддержка нескольких функциональных моделей станков и наличие обучающих режимов для них. Выбор архитектурных паттернов определялся прежде всего необходимостью обеспечения взаимодействий в реальном времени, возможностью расширения коллекции виртуального оборудования, режимов его работы и обучающих сценариев, а также достижением максимальной производительности.

Результаты. В основе функциональности симулятора лежит возможность пользователя взаимодействовать с VR-объектами в режиме реального времени. Сигналы от виртуальных элементов управления (рукояток, кнопок) могут быть нужны реагирующему механизму, а также обучающим режимам. В таком случае удобно использовать событийную архитектуру и паттерн «Наблюдатель» (Observer), который устанавливает между объектами связь типа «один ко многим» (рис. 2). В этом отношении есть один наблюдаемый объект и множество наблюдателей. При изменении наблюдаемого объекта автоматически происходит оповещение всех подписанных наблюдателей. Шаблон можно реализовать различными способами: по классической схеме из книги «Банды четырех», или используя события языка C#, события Unity (UnityEvents) или «Скриптовые объекты» (Scriptable Objects) [4]. В данном проекте предпочтение было отдано использованию более удобных и производительных UnityEvents.



Рис. 2. Схема шаблона «Наблюдатель» для оповещения о взаимодействии с элементом управления

При разработке виртуальных симуляторов полезно «думать состояниями». Во многих компонентах тренажера возникает необходимость выбора текущего состояния из множества возможных и перехода между ними в соответствии с заданными правилами. Для этого используется конечный автомат или машина состояний (англ. *finite state machine* — FSM). Этот шаблон широко распространен также в кибернетике, искусственном интеллекте и робототехнике. Он подобен паттерну «Состояние» (State) из книги «Банды четырех». При этом определяется набор состояний, в которых может находиться объект, и условия переходов между ними. Каждое состояние имеет три основные точки: Enter (выполняется при входе в состояние), Update (выполняется в состоянии в цикле), Exit (выполняется при выходе из состояния). На рис. 3 представлена схема состояний и переходов обучающего режима, который построен как сочетание FSM и двусвязного списка. Проверочный режим также использует гибридную архитектуру, сочетающую конечный автомат и дерево решений для автоматической оценки действий пользователя.

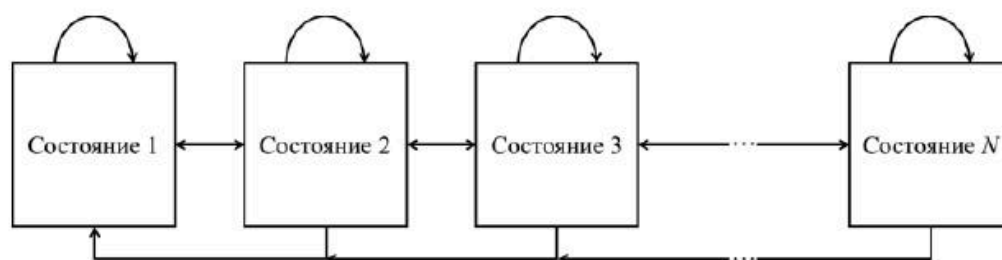


Рис. 3. Схема состояний и переходов обучающего режима

Оптимизационным порождающим паттерном является объектный пул (Object pool). Он используется для управления созданием и уничтожением объектов, особенно когда это является ресурсоемким процессом. Объектный пул позволяет повторно использовать созданные объекты, что может значительно повысить производительность приложения, что особенно важно в контексте VR. Новые версии Unity (2021+) имеют встроенный класс `ObjectPool` для его реализации [5]. Объектный пул был успешно внедрен в систему повреждений рук (рис. 4): экземпляры «ран», создаваемые при нанесении повреждений, возвращаются в пул после истечения своего жизненного цикла.

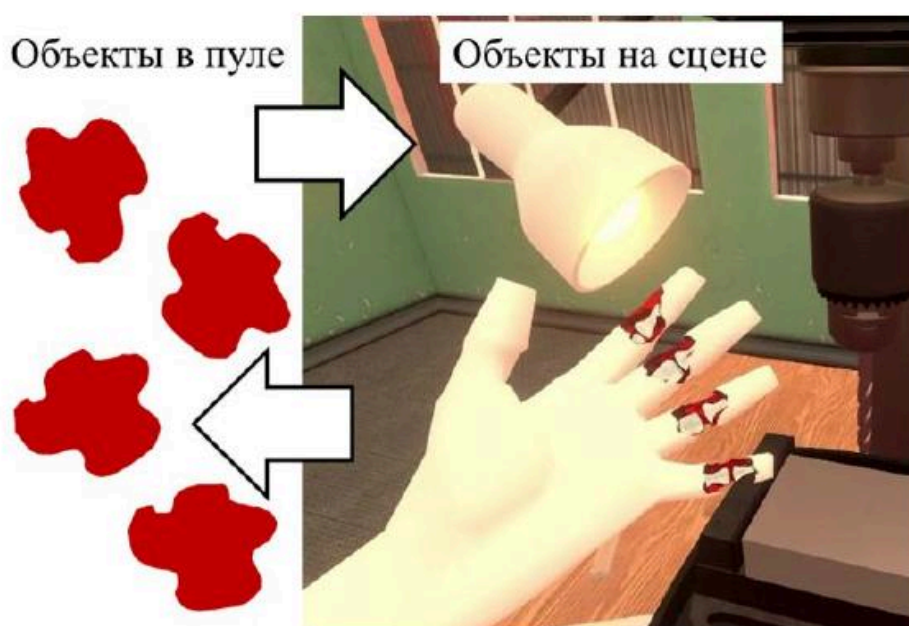


Рис. 4. Демонстрация повреждения виртуальных рук

Обсуждение полученных результатов. Таким образом, применение вышеописанных паттернов позволило повысить масштабируемость и производительность VR-приложения. Начальная версия тренажера-симулятора включала один станок с ознакомительным режимом, но впоследствии была расширена за счет добавления еще двух станков и соответствующих обучающих режимов. Использование шаблона «Наблюдатель» значительно упростило обмен сигналами между различными компонентами системы. Паттерн «Машина состояний» обеспечил гибкость и расширяемость обучающих режимов. Применение шаблона «Объектный пул» позволило избежать падения частоты кадров при создании большого количества виртуальных объектов на сцене.

В ходе проектирования рассматривались и другие распространенные для Unity паттерны [5–7]: «Модель-Представление-Контроллер» (Model-View-Controller, MVC), «Внедрение зависимостей» (Dependency injection, DI), «Стратегия» (Strategy), «Команда» (Command), «Посредник» (Mediator) и др.

Однако в данном симуляторе их применение является нецелесообразным и неудобным из-за избыточности и особенностей VR-приложений (например, отсутствия экранного интерфейса и ограниченного использования баз данных).

Заключение. Таким образом, было рассмотрено применение паттернов проектирования при разработке обучающего виртуального тренажера-симулятора металлорежущих станков на платформе Unity. Их внедрение позволило сделать программный код приложения более расширяемым, производительным и гибким. В дальнейшем планируется внедрение паттерна «Сущность-компонент-система» (Entity-Component-System, ECS), реализация механизма сохранения состояния сцены, а также интеграция системы искусственного интеллекта.

Список источников

- [1] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. *Паттерны объектно-ориентированного проектирования*. Санкт-Петербург, Питер, 2021, 448 с.
- [2] Nystrom R. Game Programming Patterns. URL: <https://gameprogrammingpatterns.com/contents.html> (дата обращения 30.03.2025).
- [3] Харитонов Л.С., Раскаткин Я.А., Эварт Т.Е. *Виртуальный тренажер-симулятор металлорежущих станков «Эсиман» версии 1*. Свидетельство о государственной регистрации программы для ЭВМ № 2025613960 РФ, 2025.
- [4] Create modular game architecture in Unity with ScriptableObjects. Unity Technologies, 2023, 75 p. URL: <https://www.createvideogiochi.it/wp-content/uploads/2023/12/create-modular-game-architecture-in-unity-with-scriptableobjects.pdf> (accessed March 30, 2025).
- [5] Baron D. Game Development Patterns with Unity. Packt Publishing, 2021, 232 p.
- [6] Level up your code with design patterns and SOLID. Unity Technologies, 2024, 147 p. URL: <https://unity.com/resources/design-patterns-solid-ebook> (accessed March 30, 2025).
- [7] Level up your code with game programming patterns. Unity Technologies, 2022, 99 p. URL: <https://unity.com/resources/level-up-your-code-with-game-programming-patterns> (accessed March 30, 2025).