МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и информационных технологий

РЕАЛИЗАЦИЯ ФИЗИЧЕСКОГО ДВИЖКА С DOD И ECS

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Павлова Егора Дмитриевича

Научный руководитель
доцент, к. т. н. _____ М. В. Хамутова

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2026

Структура и объем работы

Выпускная квалификационная работа состоит из введения, двух глав, заключения, списка литературы и приложений.

Краткое содержание работы

Первая глава посвящена анализу существующих архитектурных подходов и математического аппарата физического моделирования.

В разделе 1.1 рассмотрена классификация физических движков по целевому назначению, архитектурному подходу, размерности и модели лицензирования. Отмечена тенденция перехода от объектно-ориентированного программирования к data-oriented архитектурам.

В разделе 1.2 выполнен анализ особенностей современных процессорных архитектур. Рассмотрен эффект Memory Wall — расхождение между темпами роста производительности процессоров и скорости доступа к памяти. Описана иерархия кэш-памяти (L1, L2, L3), принципы пространственной и временной локальности, конвейеризация выполнения инструкций и работа предсказателя ветвлений. Показано, что косвенные переходы при виртуальных вызовах провоцируют ошибки предсказания и остановки конвейера. Выделены три фундаментальные проблемы объектно-ориентированных реализаций: неэффективное использование кэша из-за разброса объектов по памяти, ложное разделение данных в многопоточных средах и неоптимальность формата Array of Structures для SIMD-вычислений.

В разделах 1.3–1.6 проведён детальный обзор существующих физических движков. Box2D проанализирован как эталонный представитель объектно-ориентированного подхода; выявлены системные недостатки, связанные с низкой локальностью данных и сложностью распараллеливания. Bullet Physics рассмотрен как пример переходной архитектуры; показано, что гибридный подход не позволяет полностью реализовать преимущества ни одной из парадигм. PhysX проанализирован как промышленное решение с поддержкой GPU-ускорения; отмечена обязательная организация данных в формате Structure of Arrays для эффективной работы в GPU-режиме. Unity Physics на платформе DOTS рассмотрен как наиболее последовательное воплощение компонентно-ориентированного подхода.

В разделе 1.7 изложен математический аппарат. Рассмотрены кинематика

и динамика твёрдого тела, уравнения Ньютона-Эйлера, тензор инерции. Описаны численные методы интегрирования: метод Эйлера, интегрирование кватерниона ориентации. Детально рассмотрены алгоритмы обнаружения столкновений: ограничивающие объёмы (AABB, OBB), пространственное хеширование для широкой фазы, теорема о разделяющей оси для ориентированных параллелепипедов, алгоритм Гилберта-Джонсона-Кирти с опорной функцией. Изложены методы разрешения столкновений: импульсный метод, метод последовательных импульсов, стабилизация Баумгарте, ограничение трения конусом Кулона.

Вторая глава посвящена описанию программной архитектуры и ключевых алгоритмов разработанного движка.

В разделе 2.1 обоснован выбор технологического стека: язык Rust (безопасность памяти без сборщика мусора, гарантии отсутствия гонок данных), библиотека ash для низкоуровневого доступа к Vulkan, winit для создания окон, cgmath для математических операций, dear-imgui для отладочного интерфейса.

Архитектурная структура приложения выглдяит следующим образом:

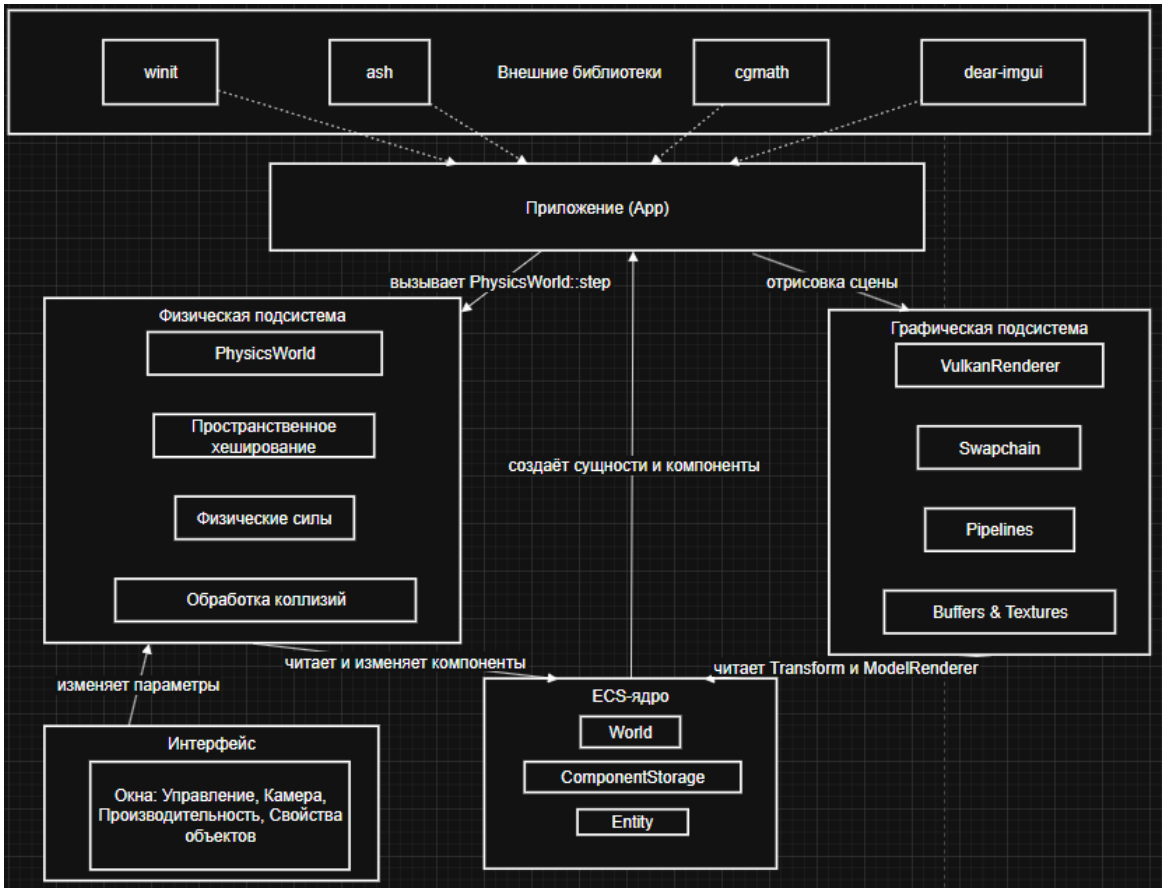

Рисунок 1 – Архитектурная схема приложения

В разделе 2.2 представлена архитектура ECS. Сущность (Entity) - это лег-

ковесный идентификатор с атомарным счётчиком. ComponentStorage<T> реализует разреженное множество: плотный массив данных, HashMap для связи сущностей с индексами, вспомогательный вектор для итерации. Вставка выполняется за амортизированное O(1), удаление — через swap-remove за O(1). Итерация обходит массив последовательно, обеспечивая оптимальную загрузку кэш-линий. World агрегирует хранилища компонентов по TypeId.

В разделе 2.3 описана физическая подсистема. Компоненты: Transform (позиция, ориентация, масштаб), RigidBody (масса, инерция, скорости, флаги), Collider (сфера или параллелепипед, статичность). Главный цикл шага включает интегрирование сил, обнаружение столкновений, тёплый старт, позиционную коррекцию, пробуждение тел, скоростной солвер, интегрирование скоростей и проверку сна. Широкая фаза на пространственном хешировании имеет сложность O(n). Узкая фаза диспетчеризует проверки по типам коллайдеров: сфера-сфера (сравнение расстояния с суммой радиусов), сфера-OBB (ближайшая точка в локальном пространстве), OBB-OBB (теорема о разделяющей оси, 15 осей). Решатель контактов реализует метод последовательных импульсов: вычисление относительной скорости в точке контакта, эффективной массы, нормального импульса с Baumgarte-стабилизацией и накоплением, тангенциальных импульсов с ограничением конусом Кулона. Тёплый старт использует дискретизацию локальных координат для построения устойчивых идентификаторов контактов.

В разделе 2.4 описана интеграция с Vulkan. Модуль vulkan.rs обеспечивает создание графического конвейера, управление памятью и синхронизацию. Система рендеринга реализована как ECS-система: обход сущностей с Transform и ModelRenderer, формирование TRS-матрицы, передача данных через push-константы.

В разделе 2.5 описан интерфейс на dear-imgui: окна управления параметрами симуляции, менеджер сцен, настройки камеры, информация о выбранном объекте, мониторинг производительности. Выбор объекта — трассировка луча с аналитическим решением пересечения со сферой.

В разделе 2.6 приведены демонстрационные сцены: падение сфер, башня из объектов, пирамида из параллелепипедов.

Виды демонстационных сцен приложения приведён ниже:

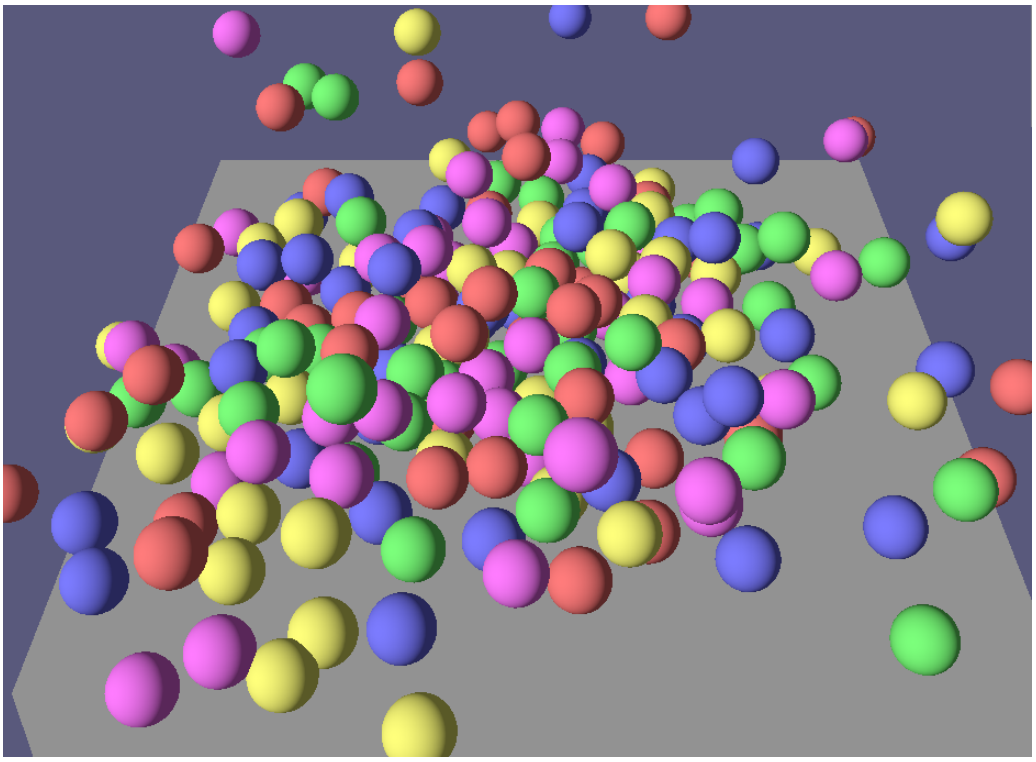

Рисунок 2 – Большое количество объектов

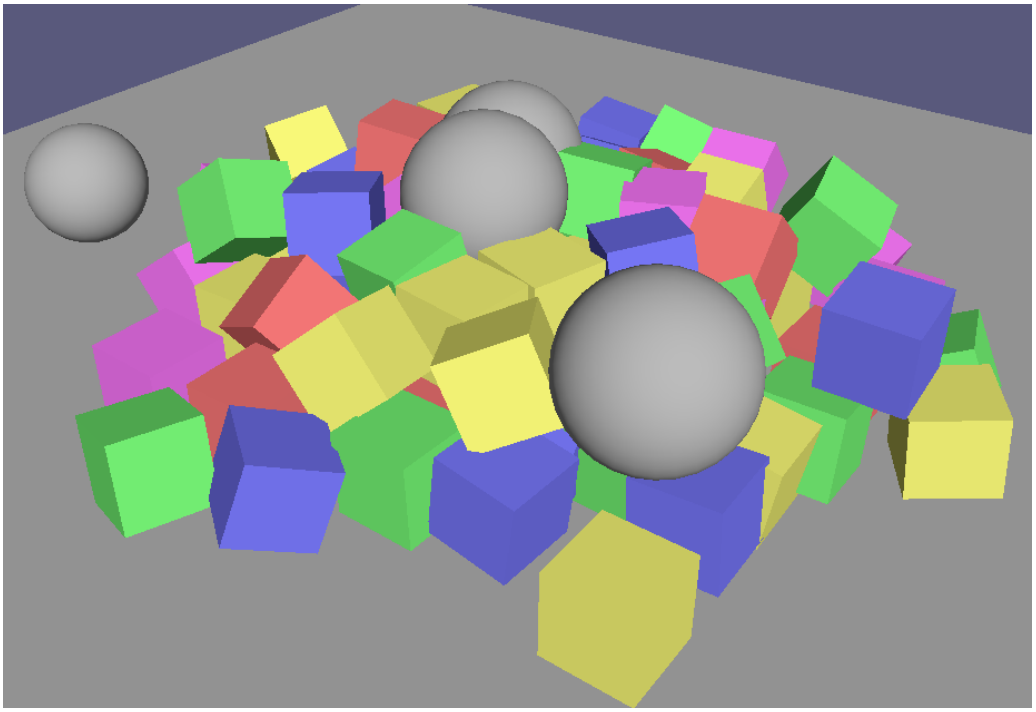

Рисунок 3 – Большое количество объектов

ЗАКЛЮЧЕНИЕ

1. Был проведён анализ существующих архитектур физических движков, выявлены фундаментальные ограничения объектно-ориентированного подхода и обоснован переход к Data-Oriented Design.
2. Разработана собственная реализация ECS на Rust с хранением компонентов в непрерывных массивах по принципу Structure of Arrays, обеспечивающая последовательный доступ к памяти и отсутствие виртуальных вызовов.
3. Реализована физическая подсистема, включающая пространственное хеширование для широкой фазы, теорему о разделяющей оси для узкой фазы и метод последовательных импульсов с тёплым стартом, стабилизацией Баумгарте и ограничением трения конусом Кулона.
4. Выполнена интеграция с Vulkan через ash, создан отладочный интерфейс на dear-imgui с возможностью изменения параметров симуляции в реальном времени.
5. Созданы демонстрационные сцены, подтверждающие работоспособность архитектуры и масштабируемость при увеличении числа объектов.

Дальнейшее развитие работы может быть связано с реализацией многопоточного планировщика ECS-систем, внедрением SIMD-оптимизаций и расширением набора типов коллайдеров.