МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра Математического и компьютерного моделирования

Разработка симулятора спортивного ориентирования на основе интеграции картографических данных

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы

направление 09.03.03 — Прикладная информатика

механико-математического факультета

Афанасьева Олега Андреевича

Научный руководитель
доцент, к.э.н., доцент Ю.В Мельникова

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2025

Введение. Симулятор спортивного ориентирования представляет собой специализированную информационно-тренировочную систему, ориентированную на моделирование процесса прохождения дистанции по карте с учётом топографических и пространственных условий. Он разрабатывается с целью поддержки тренировочного процесса спортсменов, повышения их подготовки, а также расширения возможностей участия в мероприятиях по ориентированию в условиях ограниченного доступа к реальной местности.

Современное развитие цифровых технологий, в частности в области картографии, геоинформационных систем (ГИС) и компьютерной визуализации, создаёт предпосылки для внедрения виртуальных тренажёров, имитирующих реальные условия соревнований. В основе симулятора лежит обработка и интерпретация картографических данных, позволяющая формировать виртуальную среду, максимально приближенную к условиям реального рельефа и объектов местности. Согласно концепции цифровой трансформации спорта, развитие инструментальных средств подготовки и моделирования соревновательной деятельности становится важным направлением научных и практических исследований. Особенно актуально это в контексте спортивного ориентирования, где значительную роль играют пространственное мышление, навигационные навыки и способность быстро принимать решения на местности.

Интеграция картографических данных в основу симулятора делает его эффективным инструментом подготовки спортсменов, способствует расширению доступности соревнований и поддерживает создание единого цифрового пространства спортивного ориентирования. Разработка подобного продукта особенно актуальна на фоне растущего интереса к цифровым двойникам местности и виртуальным тренажёрам в сферах образования, военной подготовки и спорта.

Целью данной работы является разработка симулятора спортивного ориентирования с интеграцией картографических данных, который может быть использован как в индивидуальной тренировке, так и при проведении дистанционных соревнований.

Для достижения указанной цели поставлены следующие задачи:

- Проанализировать теоретические и технические аспекты разработки систем симуляции и ориентирования.
- Сформировать техническое задание и обосновать целесообразность разработки.
- Реализовать программный прототип симулятора, учитывающий требования к точности отображения местности, функциональности взаимодействия с картой и пользовательскому опыту.

Первый раздел работы посвящен изучению теоретических основ разработки симулятора. Процесс проектирования был структурирован на несколько последовательных этапов.

На первом этапе проведено исследование особенностей симулируемого вида спорта. Изучено понятие симулятора, выполнен анализ вида спорта, его базовых механик, истории возникновения, правил и условий проведения. Рассмотрены различные модификации дисциплины, включая лонг, классику и спринт.

На втором этапе осуществлен анализ конкурентных решений на рынке симуляторов. Изучены существующие продукты, такие как Virtual-O, Orienteering Simulator, Catching Features. Проведено сопоставление их характеристик, выявлены сильные и слабые стороны.

На третьем этапе выполнена предварительная разработка архитектуры программного комплекса. Сформулировано предположение о необходимости построения системы на основе трёх ключевых модулей: Heightmap Generator, Map Generator и Core Simulator. Определены функции и задачи каждого модуля.

Heightmap Generator — программный модуль, предназначенный для генерации карт высот (Heightmap) на основе исходных данных о горизонталях рельефа. Получаемая карта высот представляет собой двумерное изображение, в котором интенсивность цвета каждого пикселя кодирует абсолютную высоту соответствующей точки местности.

Map Generator (или Level Generator) — система, осуществляющая построение трёхмерной модели местности. Входными данными для данной подсистемы служат карта высот, сформированная Heightmap Generator, а также информация о статических объектах ландшафта. На выходе формируется

пространственная сцена, пригодная для визуализации и дальнейшего взаимодействия.

Core Simulator — модуль симуляции спортивного ориентирования, реализующий интеграцию с результатами генерации местности. Он предоставляет инструментарий для проектирования дистанций, размещения контрольных пунктов и проведения виртуальных стартов в рамках смоделированной геопространственной среды.

На четвертом этапе описана необходимость разработки компонента Heightmap Generator, сформулированы его основные требования. Обоснованы причины отказа от использования данных с LIDAR и спутниковых снимков при генерации высотных карт.

На пятом этапе рассмотрены необходимость и требования к компоненту Map Generator. Представлено описание его функциональности, а также уточнено место Map Generator в общей архитектуре симулятора.

На шестом этапе подробно рассмотрен компонент Core Simulator. Описаны его функции, задачи и требования, предъявляемые к его реализации.

Во **втором разделе** данной работы проводится изучение существующих инструментов, фреймворков и технологий, потенциально пригодных для реализации поставленных задач.

На первом этапе был проведён анализ инструментов, фреймворков и технологий для Heightmap Generator. В результате анализа в качестве основного языка программирования выбран Python. Его применение обосновано богатой экосистемой библиотек для обработки изображений и данных (таких как PIL/Pillow, NumPy, Pandas), а также наличием инструментов для генерации высотных карт на основе алгоритмов процедурной генерации (например, шум Перлина). Python обеспечивает быструю разработку прототипов, высокую гибкость алгоритмов и широкую возможность адаптации под особенности карт спортивного ориентирования.

На втором этапе был проведён анализ инструментов и технологий для Level Generator. В качестве основной платформы выбран Unreal Engine, благодаря наличию встроенных средств для работы с высотными картами (Landscape Editor), мощных инструментов ручной и автоматизированной генерации сцен, а также поддержке модульной архитектуры, рефлексии через

Unreal Header Tool и двухуровневого подхода к программированию (C++ для низкоуровневых систем и Blueprints для высокоуровневой логики). Unreal Engine обеспечивает необходимую производительность, визуальное качество и удобство интеграции пользовательских данных, что критически важно для создания реалистичных и гибко настраиваемых трёхмерных карт.

На третьем этапе был рассмотрен вопрос реализации Simulator. Также в качестве основной платформы выбран Unreal Engine, что обеспечивает полную совместимость с ранее созданными картами и сценами. Анализ показал, что объединение всех модулей в единый проект нецелесообразно из-за высокой ресурсоёмкости движка и требований к аппаратному обеспечению. Поэтому принято решение о разделении проекта на три самостоятельных компонента: Heightmap Generator, Level Generator и Simulator, что повышает гибкость распространения и удобство использования готового продукта.

В **третьем разделе** работы представлена разработка основных компонентов симулятора спортивного ориентирования, включающая Heightmap Generator, Level Generator и Simulator.

На первом этапе рассмотрена разработка Heightmap Generator. Произведён выбор формата для работы, проанализированы особенности форматов BNA и OMAP. BNA – формат векторной графики. OMAP – внутренний формат приложения Open Orientiring Map. Выявлены проблемы разработки, включая ошибки в данных, отсутствие некоторых символов и неоднозначность трактовки информации. В качестве решения предложено использование функций типа Filter, исправляющих данные в объектах типа Line. Filter - Функция, преобразующая или исправляющая данные (например, удаление ошибок или корректировка линий). Представлена структура алгоритма, включающая основной цикл работы программы, графический интерфейс и специализированные компоненты Utility Widget. Utility Widget - Интерфейсный элемент, предоставляющий дополнительные функции или настройки. Детально рассмотрено устройство интерфейса пользователя и типы фильтров. Даны определения двум основным подходам к рендерингу высотных карт. Рендеринг - процесс визуализации данных или сцены на экране. Приведены методы оптимизации производительности программы. Продемонстри-

рованы результаты работы генератора Heightmap, в соответсвии с рисунком 1 и рисунком 2.

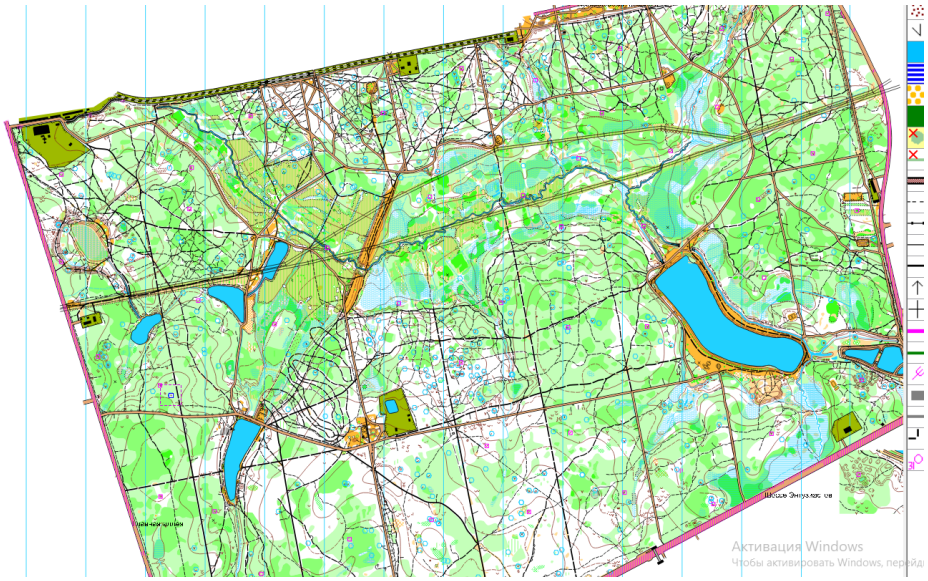

Рисунок 1 — Результат с тестовой картой 1

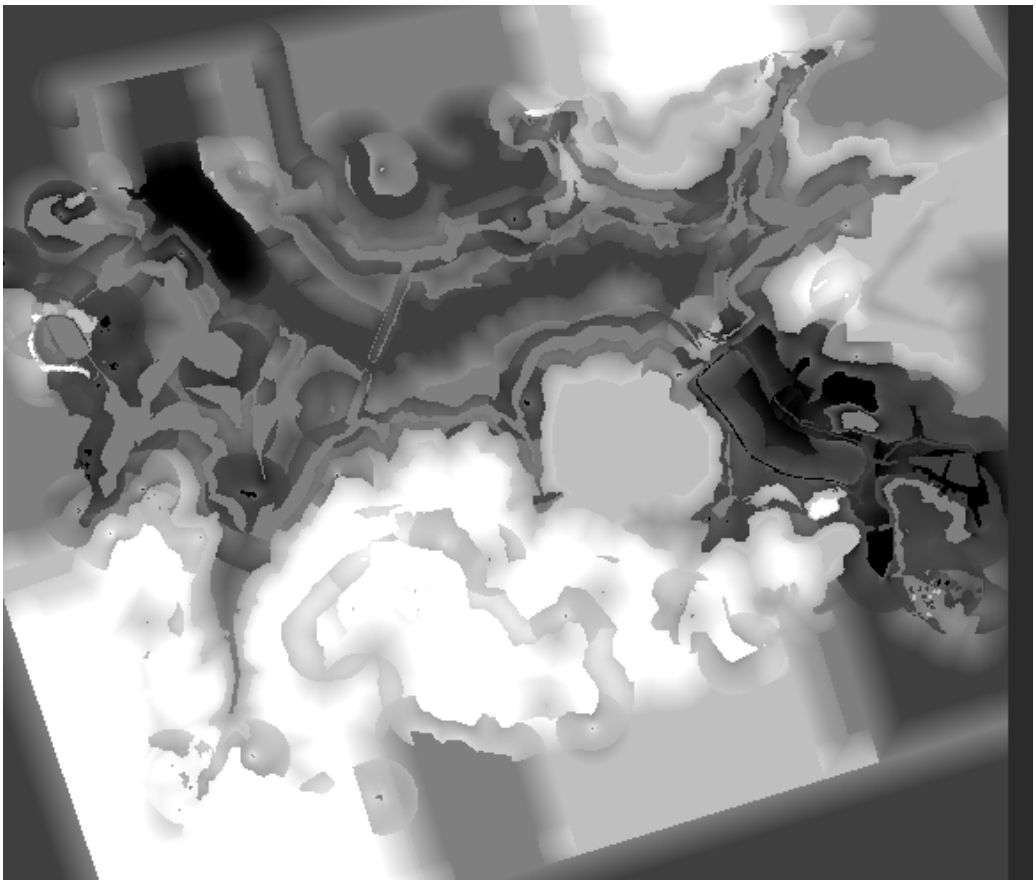

Рисунок 2 — Результат с тестовой картой 2

На втором этапе рассмотрена разработка генератора карт. В качестве основы импорта пользовательских типов данных выбран формат UASSET. Формат хранения ассетов (ресурсов) в Unreal Engine. Описано использование двух ключевых технологий генерации — PCG и Blueprint Brush. PCG (Procedural Content Generation) – Процедурная генерация контентаАвтоматическое создание содержимого (например, карт, объектов) с помощью алгоритмов. Blueprint Brush – Кисть на основе Blueprint Инструмент для рисования или изменения ландшафта, реализованный через визуальные скрипты Unreal Engine. Проанализирован модуль PCG: выделены его преимущества, выявлены недостатки, рассмотрены модификации, внесённые в систему для её расширения. Аналогичным образом рассмотрен модуль Blueprint Brush. В ходе интеграционных тестов обнаружены существенные недостатки существующей системы, обусловленные особенностями родительской системы Landscape. Landscape - Система создания и редактирования больших игровых территорий в Unreal Engine. В результате был реализован переход на собственную систему SimpleLandscape - упрощенную систему ланшафта, устраняющую выявленные недостатки. Представлена новая архитектура, описаны её преимущества, а также разработанные новые кисти, обеспечивающие функциональность старой системы. Произведена унификация старой системы Brush. Разработан специализированный виджет для упрощённой настройки генератора для не обладающих серьезным опытом в нем. Отмечены текущие недостатки инфраструктуры, такие как каскадная генерация кэша и длительность операций его создания. Проанализированы проблемы использования Procedural Mesh (динамически создаваемая 3D-сетка, используемая для генерации геометрии на лету) в качестве основного элемента генерации. Представлены результаты работы генератора уровней, в соответсвии с рисунком 3 и рисунком 4.

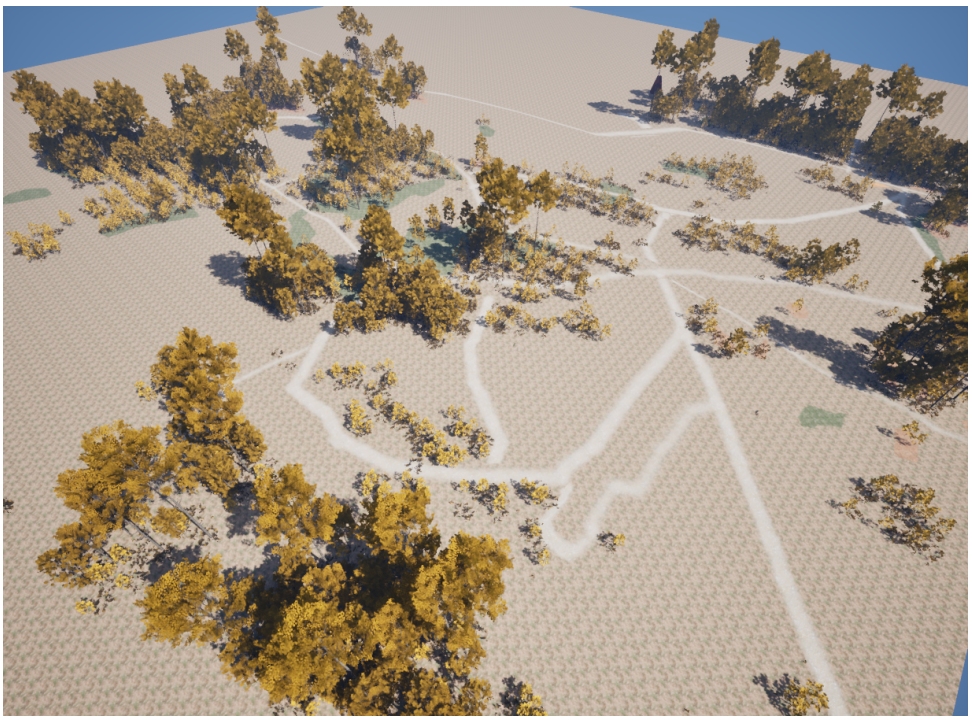

Рисунок 3 — Пример итоговой генерации элементов без Default PCG

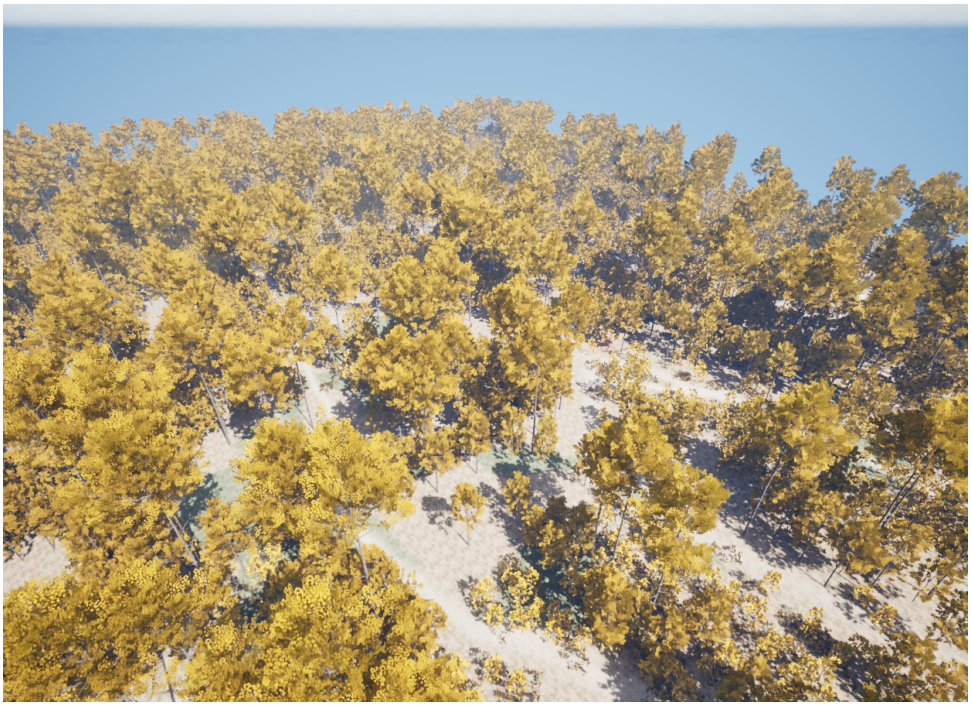

Рисунок 4 — Пример итоговой генерации элементов с Default PCG

На третьем этапе рассмотрена разработка компонента Simulator. Сформулированы основные характеристики симулятора, включая поддержку локального мультиплеера, физическую составляющую карт, методы повышения

зрелищности и интеграцию компьютерных противников. Проанализированы применяемые подходы к архитектуре проекта, описаны функции различных объектов и классов. Дано обоснование разделения сущностей Character, Player State и Player Controller. Эти три игровые сущности предствляют игрока в архитектуре Unreal Engine, но разделяют его логику и данные. Дано обоснование выбора скелетного Mesh в качестве основы для анимаций. Mesh - 3д модель, с костной архитектурой. Создан пользовательский интерфейс на базе UMG (инструмент для создания интерфейсов), являющийся основой для локаций. Разработан процесс импорта формата DLC (обьект, представляющий собой массив данных, расширяющий первоначальный продукт) для загрузки карт. Представлены результаты системы UMG, в соответсвии с рисунком 5 и общей игровой системы в соответсвии с рисунком 6.

Рисунок 5 — Пример фотографии с камеры игрока

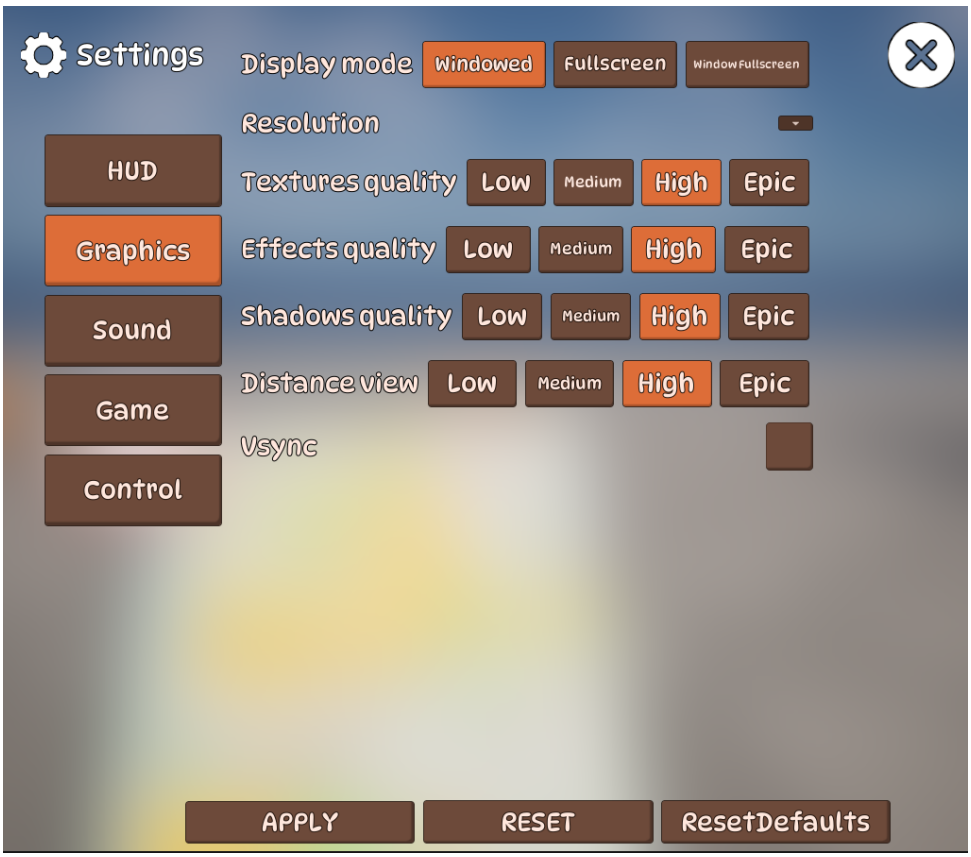

Рисунок 6 — Settings UI

В четвёртом разделе описана оптимизация в трёх направлениях: рендеринг, память, время исполнения.

Для рендеринга была выбрана система Nanite (система виртуальной геометрии Unreal Engine), обеспечивающая эффективную обработку высокополигональных моделей и совместимость с Lumen (система глобального освещения Unreal Engine) и Virtual Shadow Map (система теней с улучшенной производительностью и качеством.). Основной проблемой стала растительность, для которой отключали вершинный шейдер (программа для GPU, управляющие визуализацией объектов.) на дистанции, настраивали LodBias (параметр, влияющий на выбор качества модели в зависимости от расстояния), оптимизировали текстуры и убирали малозаметные объекты вдали. Отображение сплайнов на высоте ограничили дистанцией визуализации.

Оптимизация памяти включала использование Soft References (ссылка на объект, не мешающая его выгрузке из памяти) и соблюдение ограничений текстур для эффективного кеширования. Проблемы с World Partition (система загрузки и управления большими игровыми мирами) возникли из-за

необходимости полной загрузки объектов и закрытости его внутреннего кода, решение проблемы отложено.

Оптимизация времени исполнения разделялась на редактор и Runtime. В редакторе меры были вспомогательными; в Runtime (период, когда программа или игра работает, в отличие от редактора) основной упор сделан на эффективный код, минимальное использование Blueprints (позволяет создавать игровую логику без написания кода) и ограничение потоков. Для ускорения поиска компонентов Spline применялся Quadtree (структура данных для эффективного поиска объектов в пространстве), для оптимизации симулятора — каскадная модель событий и переписывание тяжёлого кода на C++.

На пятом этапе выполнен анализ системы пакетирования в Unreal Engine и проведено пакетирование, в рамках которого была осуществлена оптимизация ассетов (моделей, текстур, звуков) с приведением их к форматам, сбалансированным по качеству и размеру. Выполнена компиляция кода на C++ и шейдеров, настройка механизмов стриминга (динамической загрузки ресурсов по мере необходимости) данных для эффективной загрузки ресурсов в рантайме. Проведено структурирование кода с учетом требований различных типов сборок (Debug, Development, Shipping) Debug, Development, Shipping - это типы компиляции: отладочный, разработческий и финальный (для публикации). Реализовано с вынесением редакторской логики в отдельные модули или ограждением через препроцессорные директивы. Исключен код, не совместимый с финальной сборкой. Пакетирование успешно завершено без критических ошибок, получены рабочие программы для платформ Windows и Linux.

Заключение. В ходе работы:

1. Были изучены теоретические аспекты разработки симулятора спортивного ориентирования с интеграцией картографических данных.
2. Было разработано техническое задание для создания симулятора спортивного ориентирования.
3. Было выполнено проектирование ключевых компонентов симулятора.

4. Результатом является симулятор спортивного ориентирования с интеграцией картографических данных. Таким образом, задачи, поставленные в работе, полностью выполнены, и цель достигнута.

Таким образом, задачи, поставленные в работе, полностью выполнены, и цель достигнута.