МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ПО УЧЕБНОЙ ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

Студент
гр. БИС–22–1 _____ А.В. Собкалов

Руководитель
Канд. хим. наук, доцент
каф. ИТС _____ Б.К. Васильев

Владивосток 2024

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАДИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на учебную ознакомительную практику

Студент гр. БИС–22–1 Собкалов Антон Вячеславович

1 Тема работы: Разработка приложения для визуализации цифровых рельефов средствами OpenGL.

2 Срок сдачи работы: 13.07.2024.

3 Содержание задания

3.1 Решение практических задач

Задача 1.

Изучить OpenGL, а также способы его использования для визуализации 3D моделей. Выбрать библиотеку для создания пользовательского интерфейса. Изучить распространённые форматы, используемые для хранения данных о ландшафте, должна быть возможность визуализировать несколько форматов высотных карт.

Задача 2.

С использованием языка программирования C++, создать приложение с пользовательским интерфейсом для визуализации ландшафта, представляемого в различных форматах файлов. Приложение должно предоставлять возможность выбора файла, а также возможность двигаться внутри сцены для удобного исследования ландшафта. Обязательно обеспечить версию для Linux.

3.2 Оформление и защита отчета

Отчет по учебной исследовательской практике должен содержать: титульный лист; индивидуальное задание; содержание; цель и задачи; описание выполненных заданий (в соответствии с методическим руководством и требованиями преподавателя); выводы и предложения; список использованных источников; графический материал (схемы, графики, технологические карты).

Календарный план–график работ прилагается к отчету отдельным листом.

Требования к оформлению отчетов.

Отчёт предоставляется в печатном виде с выполнением требований нормоконтроля и состоит из следующих разделов:

Введение. Во введении обосновывается цель и задачи прохождения практики.

В разделе 1 выполняется краткий обзор литературы по теме индивидуального задания, подбирается необходимый математический инструментарий (аппарат), производится его анализ и разрабатывается алгоритм решения задачи.

В разделе 2 осуществляется обоснование выбора среды реализации разработанного алгоритма.

В разделе 3 описывается процесс кодирования в выбранной среде и языке программирования и основные элементы управления создаваемым приложением.

В разделе 4 описывается процесс тестированию и отладки программного кода.

Заключение. В заключении обобщается изложенный в отчёте материал, делаются выводы.

Объем отчёта составляет 26 страниц.

Отчёт по практике оформляется в соответствии с «Требованиями к оформлению текстовой части выпускных квалификационных работ, курсовых работ (проектов), рефератов, контрольных работ, отчётов по практикам, лабораторным работам (СК–СТО–ТР–04–1.005–2015)». Для оформления графического материала блок–схем, алгоритмов использовать ГОСТ 19.701–90 который распространяется на условные обозначения (символы) в схемах алгоритмов, программ, данных и систем и устанавливает правила выполнения схем, применяемых для отображения различных видов задач обработки данных и средств их решения.

Руководитель
Канд. хим. наук, доцент
каф. ИТС_____Б.К. Васильев

Задание получил: _____А.В. Собкалов

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАДИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

КАЛЕНДАРНЫЙ ПЛАН–ГРАФИК (ДНЕВНИК)

прохождения учебной ознакомительной практики студента ВВГУ

Студент Собкалов Антон Вячеславович направляется для прохождения учебной ознакомительной практики «Учебная ознакомительной практика» в ВВГУ, кафедра ИТС, г. Владивосток.

С __10.06.2024__ по __13.07.2024__ г.

№ п/п	Содержание выполняемых работ по программе	Сроки выполнения		Заключение и оценка руководителя	Подпись руководителя
		Начало	Окончание		
1	**Задание 1.** Анализ поставленной задачи	10.06.2024	17.06.2024		
2	**Задание 2.** Сбор и анализ информации	18.06.2024	19.06.2024		
3	**Задание 3.** Разработка решения поставленных задач	20.06.2024	10.07.2024		
4	**Задание 4.** Оформить отчет и документы практики в печатном и электронном виде и представить на защиту	11.07.2024	13.07.2024		

Согласовано:

Студент–практикант _____ Собкалов А.В.

подпись

Руководитель от кафедры _____ Васильев Б.К.

подпись

дата

Содержание

Введение

В рамках данной учебной практики необходимо разработать приложение для визуализации и исследования трехмерных ландшафтов, представленных в различных форматах файлов. Для достижения этой цели будут изучены современные графические технологии и методы их применения.

Ключевым элементом разработки станет изучение графической библиотеки OpenGL. OpenGL предоставляет широкий спектр возможностей для создания, отрисовки и управления трехмерными моделями. Будут рассмотрены основные принципы работы с OpenGL, а также способы ее использования для визуализации сложных 3D моделей. Особое внимание будет уделено методам рендеринга, управлению камерой, освещению, шейдерам и другим важным аспектам, необходимым для реалистичного отображения трехмерных сцен.

Параллельно с изучением OpenGL будет выбрана библиотека для создания пользовательского интерфейса приложения. Одним из вариантов является библиотека FLTK (Fast Light Toolkit), которая предоставляет простой и интуитивно понятный способ создания графического интерфейса. FLTK известна своей легкостью в использовании и высокой производительностью. Это позволит обеспечить удобное взаимодействие пользователя с программой.

Кроме того, будут изучены распространенные форматы, используемые для хранения данных о ландшафте, такие как GeoTIFF, RAW, JPEG, CSV и другие. GeoTIFF — это формат растровых географических изображений, основанный на стандарте TIFF. RAW — это необработанный формат данных, часто используемый в цифровой фотографии. JPEG — популярный формат сжатия изображений, а CSV — формат табличных данных, разделенных запятыми. Изучение этих форматов позволит обеспечить возможность визуализации нескольких типов высотных карт, расширяя возможности приложения и обеспечивая расширенный функционал.

Объединив полученные знания, будет создано приложение на языке C++ с использованием множества библиотек, позволяющее пользователю выбирать файл с ландшафтом, а также перемещаться внутри трехмерной сцены для удобного исследования и изучения рельефа. Особое внимание будет уделено оптимизации производительности и плавности анимации, а также четкому отображению модели, чтобы обеспечить комфортное взаимодействие с приложением.

Визуализация ландшафта является важной частью дизайна и планирования архитектурных проектов. Она позволяет архитекторам и заказчикам увидеть, как будет выглядеть участок до начала работ, создавая эффект присутствия и позволяя оценить будущую архитектурную планировку ландшафта. Визуализация ландшафта может включать в себя создание трехмерных изображений местности, обладающих фотографической реалистичностью, что помогает

клиенту понимать свои конечные ожидания от проекта и исполнителю наилучшим образом презентовать свои идеи.

Создание приложения для визуализации и исследования трехмерных ландшафтов позволит пользователю выбирать файл с ландшафтом, перемещаться внутри трехмерной сцены и изучать рельеф. Это приложение будет использовать современные графические технологии, такие как OpenGL, и удобный пользовательский интерфейс, реализованный на кроссплатформенном языке C++. В результате выполнения данного проекта будет создано приложение, позволяющее визуализировать и исследовать трехмерные ландшафты, представленные в различных форматах файлов, с использованием современных графических технологий и удобного пользовательского интерфейса.

Таким образом, разработка приложения для визуализации и исследования трехмерных ландшафтов с использованием современных графических технологий является важной и актуальной задачей в рамках данной учебной практики. Изучение и применение библиотеки OpenGL, FLTK и различных форматов данных позволит создать функциональное и высококачественное приложение, которое будет востребовано в сфере архитектуры и дизайна. Полученные знания и навыки также могут быть применены в других областях, требующих визуализации трехмерных моделей и сцен.

1. Карты высот

1.1 История

Карты высот, отображающие рельеф местности, имеют долгую историю развития. Первые примитивные карты создавались вручную, без использования специальных инструментов для точных измерений, поэтому обладали большой погрешностью. Одни из самых ранних известных карт высот датируются 3–м тысячелетием до нашей эры и были найдены на территории современного Ирака.

Эти древние карты представляли собой глиняные таблички с нанесенными на них контурами гор и долин. Они использовались для ориентирования и планирования торговых маршрутов. Со временем картографы научились более точно изображать рельеф, используя методы послойной окраски и штриховки. Такие карты появились в Древнем Китае, Греции и Риме.

В средневековой Европе карты высот были редкостью, так как основное внимание уделялось навигационным картам морских побережий. Лишь в эпоху Возрождения интерес к картографии ландшафтов возродился. Леонардо да Винчи, Герард Меркатор и другие ученые того времени создавали карты с изображением рельефа, используя методы послойной окраски и гравировки.

С развитием технологий появились новые способы создания карт высот. В 19 веке стали применяться топографические съемки с использованием теодолитов и нивелиров. Это позволило создавать более точные карты с изображением горизонталей – линий равных высот. Такие карты широко использовались в военном деле, строительстве и геологии.

Современные карты высот могут создаваться вручную с помощью простых программ рисования или редакторов ландшафта, которые генерируют визуальную картинку местности в формате 3D. Существуют специальные программы, такие как Aerialod, которые преобразуют импортированную карту высот в 3D–ландшафт. Aerialod принимает стандартные форматы карт высот: PNG, ASC, DTM и другие.

Для генерации карт высот также используются алгоритмы программирования. Например, можно задать каждой точке случайное значение высоты, а затем добавить холмы и ямы, используя специальные функции. Для сглаживания ландшафта применяются алгоритмы размытия, усредняющие значения высот соседних точек.

Современные технологии позволяют создавать карты высот с беспрецедентной точностью. Спутниковые системы, такие как SRTM и ASTER GDEM, позволяют получать данные о рельефе с разрешением до 30 метров. Эти данные используются для создания цифровых моделей местности, которые находят применение в различных областях, таких как картография, моделирование ландшафтов, планирование инфраструктуры и даже в компьютерных играх.

Таким образом, карты высот прошли путь от примитивных ручных чертежей до сложных алгоритмов генерации 3D–ландшафтов. Развитие технологий позволило создавать все более точные и детализированные карты, которые находят широкое применение в современном мире. От древних глиняных табличек до спутниковых снимков – история карт высот отражает прогресс человечества в познании и изображении окружающего мира.

1.2 Карта высот

Карты высот – это двухмерные изображения, которые используются для хранения и визуализации информации о рельефе местности. Они представляют собой матрицу значений высот, где каждому пикселю соответствует определенная высота над уровнем моря или другой выбранной отсчетной поверхности.

Обычно карты высот хранятся в виде 8–битных изображений, где значения пикселей варьируются от 0 (самая низкая точка) до 255 (самая высокая точка). Более широкий диапазон высот может быть представлен с помощью 16–битных или 32–битных форматов. Для визуализации карты высот используется градиент серого цвета, где темные оттенки соответствуют низким участкам, а светлые – высоким.

Карты высот могут создаваться различными способами:

Вручную, с помощью графических редакторов, где пользователь рисует или редактирует ландшафт.

Автоматически, с использованием алгоритмов процедурной генерации, которые создают карты высот на основе заданных параметров.

На основе данных топографических съемок, спутниковых снимков или других источников, содержащих информацию о рельефе.

Карты высот находят широкое применение в различных областях:

– Картография и геология – для визуализации рельефа местности.

– Моделирование ландшафтов – в компьютерной графике и видеоиграх.

– Планирование инфраструктуры – для анализа местности при строительстве дорог, трубопроводов и т.д.

– Военное дело – для анализа местности, планирования операций и навигации.

Таким образом, карты высот являются важным инструментом для представления и анализа рельефа местности, находя применение в самых разных сферах человеческой деятельности.

1.3 Алгоритм создания приложения

Первым шагом является установка необходимого программного обеспечения для разработки приложения визуализации ландшафта. Это включает в себя:

1. Установка компилятора C++ g++. g++ является стандартным компилятором C++ в Unix–подобных системах, таких как Linux и macOS. Он необходим для компиляции исходного кода C++ в исполняемый файл.

2. Установка библиотек OpenGL и FLTK. OpenGL (Open Graphics Library) – это стандартная библиотека для создания 2D и 3D графики. FLTK (Fast Light Toolkit) – кроссплатформенная библиотека для создания графических пользовательских интерфейсов (GUI).

3. Настройка проекта в Visual Studio Code. Visual Studio Code (VS Code) – популярная интегрированная среда разработки (IDE), которая поддерживает разработку на C++. Для настройки проекта в VS Code необходимо:

– Создать новую папку для проекта.

– Открыть папку в VS Code.

– Установить расширения для C++, OpenGL и FLTK.

– Настроить компилятор и линковщик в settings.json.

После установки всех необходимых компонентов и настройки IDE можно приступать к разработке приложения.

Создание основного окна приложения

Используя библиотеку FLTK, создается основное окно приложения со следующими элементами:

1. Меню. Меню позволяет пользователю выполнять различные действия, такие как загрузка файлов, настройка отображения и управление камерой.

2. Область отображения ландшафта. Это основная область окна, где будет визуализироваться загруженный ландшафт с использованием OpenGL.

Определяются основные элементы интерфейса, такие как:

– Кнопки для загрузки файлов в форматах RAW, TIFF, GeoTIFF, JPEG и CSV.

– Реализация загрузки и обработки данных.

Создаются функции для чтения файлов в различных форматах:

1. RAW. Формат RAW содержит только данные о высотах без дополнительной информации. Функция чтения RAW–файла должна определять размер карты и загружать данные в память.

2. TIFF и GeoTIFF. Форматы TIFF и GeoTIFF содержат данные о высотах и дополнительную информацию, такую как географические координаты и метаданные. Функции чтения этих форматов должны извлекать необходимую информацию.

3. JPEG. Формат JPEG используется для хранения текстур, которые могут быть наложены на ландшафт. Функция чтения JPEG–файла загружает текстурные данные.

4. CSV. Формат CSV (Comma–Separated Values) представляет данные в табличном виде. Функция чтения CSV–файла должна парсить данные и извлекать высоты и другую необходимую информацию.

После чтения данных из файлов необходимо обработать их для использования в визуализации. Это включает в себя:

1. Извлечение высот из данных.

2. Обработку текстур (при наличии).

3. Обработку метаданных (географические координаты, масштаб и т.д.).

Визуализация ландшафта с использованием OpenGL

Для визуализации ландшафта с использованием OpenGL необходимо:

1. Создание 3D–сцены. Создается 3D–сцена, в которой будет отображаться ландшафт. Определяются размеры сцены, положение камеры и источники света.

2. Создание сетки высот. На основе данных, загруженных из файлов, создается сетка высот. Каждая вершина сетки имеет координаты (X, Y, Z), где Z – высота в данной точке.

3. Применение текстур. Если доступны текстурные данные, они накладываются на сетку высот для улучшения визуального представления ландшафта.

4. Применение освещения. Применяются различные техники освещения, такие как ambient, diffuse и specular, для создания более реалистичного отображения ландшафта.

5. Другие эффекты. Могут быть применены дополнительные эффекты, такие как туман, отражения, тени и т.д., для улучшения визуального качества.

Реализация управления камерой

Для управления камерой используются функции FLTK для обработки ввода пользователя с клавиатуры и мыши. Реализуются следующие функции:

1. Перемещение камеры. Пользователь может перемещать камеру вперед, назад, влево и вправо с помощью клавиш или стрелок.

2. Вращение камеры. Пользователь может вращать камеру, нажимая и перетаскивая мышью по экрану.

3. Масштабирование камеры. Пользователь может приближать или удалять камеру с помощью колеса прокрутки мыши или специальных клавиш.

Положение и ориентация камеры обновляются в реальном времени, чтобы пользователь мог свободно исследовать ландшафт.

Тестирование и оптимизация

Для обеспечения качества и производительности приложения необходимо провести тщательное тестирование и оптимизацию:

1. Тестирование с различными наборами данных. Приложение должно быть протестировано с различными наборами данных, включая большие и сложные ландшафты, для проверки стабильности и корректности работы.

2. Тестирование сценариев использования. Должны быть протестированы различные сценарии использования, такие как загрузка файлов, навигация по ландшафту, измерение параметров и т.д.

3. Оптимизация производительности. Для обеспечения плавного отображения ландшафта даже для больших наборов данных необходимо оптимизировать код, используя техники, такие как LOD (Level of Detail), фрустум–кулинг и параллельную обработку данных.

4. Тестирование на различных платформах. Приложение должно быть протестировано на различных операционных системах и аппаратных платформах для обеспечения кроссплатформенной совместимости.

Таким образом, разработка приложения визуализации ландшафта включает в себя установку необходимого ПО, создание основного окна приложения, реализацию загрузки и обработки данных, визуализацию ландшафта с использованием OpenGL, управление камерой, добавление дополнительных функций и тщательное тестирование и оптимизацию. Каждый из этих этапов важен для создания качественного и функционального приложения.

2. Выбор среды разработки и языка программирования

2.1 Язык программирования

C++ является отличным выбором для создания приложения, которое визуализирует ландшафт с использованием OpenGL и FLTK, по следующим причинам:

1. Производительность и низкоуровневый доступ

C++ является компилируемым языком, что обеспечивает высокую производительность приложения. Он предоставляет прямой доступ к низкоуровневым системным функциям, что критично для работы с графическими библиотеками, такими как OpenGL.

2. Интеграция с OpenGL

OpenGL является стандартной библиотекой для 3D–графики, тесно интегрированной с C++. Это позволяет эффективно использовать возможности OpenGL для создания высокопроизводительных визуализаций ландшафта.

3. Возможности FLTK

FLTK – это кроссплатформенная библиотека пользовательского интерфейса, также написанная на C++. Она предоставляет богатый набор виджетов и инструментов для создания интуитивно понятного графического интерфейса для приложения, визуализирующего ландшафт.

4. Кроссплатформенность

C++ является кроссплатформенным языком, что позволяет разрабатывать приложение, которое будет работать на различных операционных системах, таких как Windows, macOS и Linux. Это важно для обеспечения широкой доступности и совместимости вашего приложения.

5. Управление памятью и ресурсами

C++ предоставляет развитые средства управления памятью и ресурсами, что критично для приложений, работающих с большими объемами данных, как в случае визуализации ландшафта. Это помогает избежать утечек памяти и обеспечить стабильную работу приложения.

6. Расширяемость и модульность

Объектно–ориентированная природа C++ позволяет создавать модульную и расширяемую архитектуру приложения. Это упрощает добавление новых функций, таких как поддержка дополнительных форматов данных или улучшение алгоритмов визуализации.

7. Сообщество и инструменты

C++ имеет обширное сообщество разработчиков и богатую экосистему инструментов, библиотек и фреймворков. Это облегчает разработку, отладку и поддержку приложения, визуализирующего ландшафт.

В целом, C++ является превосходным выбором для создания приложения, визуализирующего ландшафт с использованием OpenGL и FLTK. Его производительность, интеграция с графическими библиотеками, кроссплатформенность и развитые средства управления ресурсами делают его идеальным языком для таких задач.

При выборе языка программирования для создания приложения, визуализирующего ландшафт с использованием OpenGL и FLTK, C++ имеет ряд преимуществ по сравнению с другими популярными языками, такими как Python, Java и C#.

C++ или Python

Производительность: C++ обеспечивает более высокую производительность, особенно при работе с графикой и обработке больших объемов данных, что важно для визуализации ландшафта.

Низкоуровневый доступ: C++ предоставляет более прямой доступ к системным функциям и библиотекам, таким как OpenGL, что упрощает интеграцию и оптимизацию.

Управление памятью: C++ предлагает более гибкие и эффективные средства управления памятью по сравнению с автоматическим управлением памятью в Python.

C++ или Java

Производительность: C++ превосходит Java в производительности, особенно в областях, требующих низкоуровневого доступа, таких как графические приложения.

Интеграция с OpenGL: C++ имеет более тесную интеграцию с OpenGL, что упрощает разработку графических приложений.

Кроссплатформенность: Несмотря на то, что Java также является кроссплатформенным языком, C++ может обеспечить более эффективную кроссплатформенность, особенно при работе с системно–зависимыми библиотеками, такими как FLTK.

C++ или C#

Производительность: C++ и C# имеют сравнимую производительность, но C++ может иметь небольшое преимущество, особенно при работе с низкоуровневыми системными функциями.

Интеграция с OpenGL: C++ имеет более прямую интеграцию с OpenGL, в то время как интеграция с OpenGL в C# требует использования дополнительных библиотек.

Кроссплатформенность: C++ обладает более широкой кроссплатформенностью, поскольку C# в основном ориентирован на платформу .NET, в то время как C++ может работать на различных операционных системах без существенных ограничений.

2.2 Выбор библиотек

FLTK (Fast Light Toolkit)

Кроссплатформенность: FLTK поддерживает множество операционных систем, включая Windows, macOS и Linux, что позволяет создавать приложения, которые будут работать на различных платформах.

Легкость в использовании: FLTK имеет простой и интуитивно понятный API, что упрощает разработку графического интерфейса для приложения.

Быстрая загрузка: FLTK известен своей быстрой загрузкой и низким потреблением ресурсов, что важно для приложений, которые должны работать с большими объемами данных.

TIFF (Tagged Image File Format)

Поддержка растровых изображений: TIFF является популярным форматом для хранения растровых изображений, что позволяет загружать и отображать изображения в приложении.

Компрессия и качество: TIFF поддерживает различные алгоритмы компрессии, что позволяет сохранять изображения с высоким качеством и небольшим размером.

Версионирование: TIFF имеет версионирование, что позволяет поддерживать обратную совместимость с различными версиями формата.

JPEG (Joint Photographic Experts Group)

Компрессия и качество: JPEG известен своей эффективной компрессией и качеством изображений, что позволяет загружать и отображать изображения с высоким качеством.

Поддержка цветовых пространств: JPEG поддерживает различные цветовые пространства, что позволяет загружать и отображать изображения с различными цветовыми профильами.

GeoTIFF (Georeferenced TIFF)

Геореквизитация: GeoTIFF позволяет добавлять геореквизитацию к изображениям, что позволяет отображать изображения в географическом контексте.

Поддержка координатных систем: GeoTIFF поддерживает различные координатные системы, что позволяет загружать и отображать изображения с различными системами координат.

CSV (Comma Separated Values)

Простота и универсальность: CSV является простым и универсальным форматом для хранения табличных данных, что позволяет загружать и отображать данные в различных форматах.

Поддержка различных приложений: CSV поддерживается множеством приложений и библиотек, что позволяет интегрировать данные в различные системы.

LJPEG (Lossless JPEG)

Компрессия и качество: LJPEG является форматом, который поддерживает компрессию без потерь качества, что позволяет загружать и отображать изображения с высоким качеством.

Поддержка цветовых пространств: LJPEG поддерживает различные цветовые пространства, что позволяет загружать и отображать изображения с различными цветовыми профильями.

LGLFW (Light OpenGL Framework)

Сокращение кода: LGLFW является легкой и простой в использовании библиотекой для OpenGL, что позволяет сократить код и упростить разработку графических приложений.

Поддержка OpenGL: LGLFW поддерживает различные версии OpenGL, что позволяет создавать приложения, которые будут работать на различных платформах.

LGLEW (Light OpenGL Extension Wrangler)

Сокращение кода: LGLEW является легкой и простой в использовании библиотекой для загрузки и использования OpenGL–экстеншенов, что позволяет сократить код и упростить разработку графических приложений.

Поддержка OpenGL–экстеншенов: LGLEW поддерживает различные OpenGL–экстеншены, что позволяет создавать приложения, которые будут использовать различные функции и возможности OpenGL.

В целом, выбор этих библиотек был обусловлен их кроссплатформенностью, простотой в использовании, поддержкой различных форматов данных и графических функций, а также их способностью обеспечить высокую производительность и качество визуализации ландшафта.

2.3 Пользовательский интерфейс

При разработке пользовательского интерфейса на C++ для приложения, визуализирующего ландшафт, существует несколько популярных библиотек и фреймворков, которые можно рассмотреть в качестве альтернатив FLTK.

Qt – это мощный и всесторонний фреймворк с богатым набором виджетов и инструментов. Он предоставляет широкие возможности для создания сложных и функциональных графических интерфейсов. Qt позволяет разрабатывать кроссплатформенные приложения, которые работают на различных программных и аппаратных платформах, таких как Linux, Windows, macOS, Android или встроенные системы, с минимальными изменениями в исходном коде. При этом приложения, созданные с помощью Qt, являются нативными с нативными функциями и производительностью.

Qt в настоящее время разрабатывается компанией The Qt Company, публичной компанией, и Qt Project в рамках Open Source, в которую входят индивидуальные разработчики и организации, стремящиеся продвигать Qt. Qt доступен по коммерческим и открытым лицензиям (GPL 2.0, GPL 3.0 и LGPL 3.0).

Тем не менее, Qt является более сложным и ресурсоемким по сравнению с FLTK, и требует больше времени на изучение. Для приложения, визуализирующего ландшафт, такая сложность может быть избыточной, и более простой в использовании FLTK может быть предпочтительным выбором.

wxWidgets (ранее wxWindows) – это библиотека виджетов и инструментов для создания кроссплатформенных графических интерфейсов (GUI). wxWidgets обеспечивает единообразный интерфейс на различных операционных системах. Некоторые известные программы, написанные с использованием wxWidgets, включают в себя Code::Blocks, FileZilla и Audacity.

Тем не менее, wxWidgets считается более громоздкой и менее оптимизированной по сравнению с FLTK. Для приложения, визуализирующего ландшафт, более легковесная и производительная FLTK может быть более подходящим выбором.

GTK+ – это популярный и мощный фреймворк, особенно на платформе Linux. Он предоставляет богатый набор инструментов для создания графических интерфейсов. Однако GTK+ в первую очередь ориентирован на платформу Linux и может быть сложнее в использовании на Windows и macOS. Для кроссплатформенного приложения, которое должно работать одинаково хорошо на различных операционных системах, FLTK может быть более подходящим выбором.

В сравнении с этими библиотеками, FLTK обладает рядом преимуществ, которые делают ее наиболее подходящим выбором для приложения, визуализирующего ландшафт:

Кроссплатформенность: FLTK работает на Windows, Linux и macOS, обеспечивая единообразный интерфейс на различных операционных системах. Это позволяет создавать приложения, которые будут работать на широком спектре платформ без необходимости значительных изменений в коде.

Легковесность и производительность: FLTK отличается небольшим размером и низким потреблением ресурсов, что важно для приложений, работающих с большими объемами данных, необходимыми для визуализации ландшафта. Более легкий и оптимизированный FLTK позволяет эффективно обрабатывать и отображать большие наборы данных высотной карты.

Простота: FLTK имеет простой и интуитивно понятный API, что упрощает разработку графического интерфейса приложения. Это позволяет быстрее создавать и итерировать над интерфейсом, сосредоточившись на основных функциях приложения.

Интеграция с OpenGL: FLTK имеет встроенную поддержку OpenGL, что делает ее идеальным выбором для приложений, визуализирующих ландшафт. Встроенная поддержка OpenGL упрощает реализацию высокопроизводительной 3D–визуализации ландшафта с использованием этой популярной графической библиотеки.

Статическая компоновка: FLTK может быть статически скомпонована, что упрощает распространение и установку приложения. Возможность статической компоновки позволяет создавать автономные исполняемые файлы, которые можно легко распространять и устанавливать на различных системах без необходимости в отдельных библиотеках.

Таким образом, FLTK предлагает оптимальное сочетание кроссплатформенности, производительности, простоты использования и интеграции с OpenGL, что делает ее наилучшим выбором для разработки приложения, визуализирующего ландшафт, по сравнению с другими популярными библиотеками, такими как Qt, wxWidgets и GTK+.

Кроме того, FLTK является свободной и открытой программной библиотекой, распространяемой под лицензией GNU Lesser General Public License (LGPL) с дополнительным условием, разрешающим статическую компоновку приложений с несовместимыми лицензиями. Это делает ее доступной и гибкой для использования в различных проектах.

В заключение, при разработке приложения для визуализации ландшафта на C++ FLTK является наиболее подходящим выбором благодаря своей кроссплатформенности, легковесности, простоте использования, интеграции с OpenGL и возможности статической компоновки. Эти преимущества делают FLTK оптимальным решением для создания эффективных и кроссплатформенных приложений для визуализации ландшафта.

3. Процесс написания кода

1. Подготовка и загрузка данных высотной карты

Первым шагом является предоставление пользователю возможности выбрать файл с высотной картой. Для этого используется функция Fl_Native_File_Chooser, которая открывает диалоговое окно выбора файла, нативное для системы пользователя. Пользователь может выбрать файл в форматах raw, tiff, csv, jpeg. После выбора файла определяется его размер с помощью функций GetImgSize, GetRawSize и GetCsvSize в зависимости от формата. Эти функции анализируют содержимое файла и возвращают ширину и высоту карты.

Далее создается двумерный массив HeightMap для хранения высотных данных. Размер массива определяется на основе размера высотной карты, полученного на предыдущем шаге. Затем данные из файла загружаются в массив HeightMap с помощью функций LoadImgFile, LoadRawFile и LoadCsvFile в зависимости от формата файла.

Для обработки изображений и получения размера карты а также ее данных используется библиотека CImg для обработки изображений, а также std::ifstring для обработки текстовых файлов.

Для удобства дальнейшей работы двумерный массив HeightMap преобразуется в одномерный массив heightMap. Это позволяет более эффективно использовать данные в процессе визуализации.

2. Настройка OpenGL и создание окна

Следующим шагом является инициализация библиотеки GLFW для создания окна и контекста OpenGL. Создается окно с заданными размерами и настройками, такими как возможность изменения размера окна. Также инициализируется библиотека GLEW для работы с функциями OpenGL.

3. Компиляция и связывание шейдеров

Для визуализации ландшафта необходимо создать и скомпилировать вершинный и фрагментный шейдеры. Вершинный шейдер отвечает за преобразование вершин, а фрагментный шейдер – за расчет цвета каждого фрагмента. Шейдеры связываются в единую программу, которая будет использоваться в дальнейшем для отрисовки.

4. Создание вершин и индексов для визуализации ландшафта

Для построения поверхности ландшафта необходимо создать вершины с соответствующими координатами (x, y, z) и нормалями. Для каждой ячейки высотной карты генерируются четыре вершины, образующие два треугольных многоугольника. Координаты вершин вычисляются на основе значений высот в массиве heightMap и генерируются с помощью цикла и одномерного массива данных.

Вычисление нормалей для каждой вершины производится с использованием разностей высот соседних ячеек. Это позволяет обеспечить корректное освещение поверхности. Нормали добавляются к вершинам в вектор vertices.

Также создаются индексы для построения треугольников, формирующих поверхность ландшафта. Эти индексы добавляются в вектор indices.

5. Создание и заполнение буферов OpenGL

Для эффективной отрисовки ландшафта необходимо создать буферы OpenGL для хранения вершин и индексов. Создаются буферы VBO (Vertex Buffer Object) для вершин и EBO (Element Buffer Object) для индексов.

Данные вершин и индексов загружаются в соответствующие буферы. Также создается вертекс массив объект VAO (Vertex Array Object) для связывания вершин, нормалей и индексов.

6. Обработка ввода пользователя и управление камерой

Для обеспечения интерактивности приложения реализуется обработка ввода пользователя. Это включает в себя обработку нажатий клавиш (W, S, A, D, стрелки вверх/вниз, влево/вправо) для перемещения и вращения камеры, а также клавиши C, F, Esc для отвязки/привязки мышки, открытие окна в полноэкранный режим и выход из программы соответственно.

Функция processInput обрабатывает ввод пользователя и обновляет положение и ориентацию камеры. Для этого используются переменные cameraPos, cameraFront, cameraUp и cameraRight, которые хранят текущее состояние камеры.

Также реализуется обработка движения мыши для вращения камеры. Для этого используется функция mouse_callback, которая обновляет углы поворота камеры (yaw и pitch) в соответствии с движением мыши.

Функция updateCameraVectors обновляет направление взгляда камеры на основе текущих значений yaw и pitch.

7. Визуализация ландшафта

Заключительным этапом является визуализация ландшафта. В основном цикле приложения выполняется отрисовка ландшафта с использованием шейдерной программы и загруженных данных.

Для отрисовки используются буферы, созданные на предыдущем этапе, и шейдерная программа, скомпилированная ранее. Применяются различные техники освещения (ambient, diffuse, specular) для более реалистичного отображения ландшафта.

Ambient освещение создает общий фон, diffuse освещение отвечает за диффузное отражение света от поверхности, а specular освещение добавляет блики на поверхности. Эти техники освещения позволяют добиться более реалистичного и выразительного отображения ландшафта.

Таким образом, данный код реализует полный процесс загрузки высотной карты, сгенерированный по изображениям в RAW, JPEG, TIFF и CSV форматах, настройки OpenGL, создания геометрии ландшафта и его визуализации с возможностью управления камерой и другими дополнительными функциями. Это позволяет пользователю интерактивно исследовать и просматривать трехмерный ландшафт, созданный на основе высотных данных.

4. Тестирование кода

Тестирование данного кода можно разделить на несколько основных этапов, чтобы обеспечить его надежность и корректность работы.

1. Тестирование загрузки данных высотной карты:

Первым шагом является тщательное тестирование загрузки данных высотной карты из различных форматов файлов (RAW, TIFF, CSV). Необходимо проверить, что:

Пользователь может успешно выбрать файл с помощью диалогового окна.

Функции GetImgSize, GetRawSize и GetCsvSize корректно определяют размер высотной карты для каждого формата.

Функции LoadImgFile, LoadRawFile и LoadCsvFile успешно загружают данные в двумерный массив HeightMap.

Преобразование двумерного массива HeightMap в одномерный массив heightMap выполняется корректно.

Для этого следует протестировать загрузку файлов различных размеров и форматов, в том числе граничные случаи, чтобы убедиться в отсутствии ошибок.

2. Тестирование создания и настройки OpenGL:

Следующим этапом является тестирование создания окна и настройки OpenGL. Необходимо проверить, что:

Инициализация GLFW и GLEW выполняется успешно.

Создание окна с заданными размерами и настройками (например, возможность изменения размера) работает корректно.

Контекст OpenGL настраивается правильно и готов к дальнейшей работе.

Для этого можно запустить приложение и убедиться, что окно отображается корректно, а OpenGL–функции доступны и работают как ожидается.

3. Тестирование компиляции и связывания шейдеров:

Важным этапом является тестирование компиляции и связывания вершинного и фрагментного шейдеров. Необходимо проверить, что:

Шейдеры компилируются без ошибок.

Шейдерная программа создается и связывается корректно.

Шейдерная программа может быть успешно использована для визуализации ландшафта.

Для этого можно вывести информацию о статусе компиляции и связывания шейдеров, а также проверить, что визуализация ландшафта выполняется корректно.

4. Тестирование создания вершин и индексов:

Важно протестировать процесс создания вершин и индексов для визуализации ландшафта. Необходимо проверить, что:

Вершины создаются с правильными координатами и нормалями.

Индексы формируют корректные треугольники для построения поверхности.

Вычисление нормалей выполняется корректно и обеспечивает правильное освещение.

Для этого можно визуально проверить отображение ландшафта и убедиться, что он выглядит правильно.

5. Тестирование обработки ввода пользователя и управления камерой:

Заключительным этапом является тестирование обработки ввода пользователя и управления камерой. Необходимо проверить, что:

Обработка нажатий клавиш (W, S, A, D, стрелки, Esc) работает корректно и перемещает, и вращает камеру как ожидается.

Обработка движения мыши работает корректно и вращает камеру в соответствии с движением.

Функция updateCameraVectors обновляет направление взгляда камеры правильно.

Для этого можно проверить управление камерой в различных сценариях и убедиться, что она ведет себя ожидаемым образом.

Таким образом, всесторонне протестировав все основные аспекты кода, можно обеспечить его надежность и корректность работы. Это позволит создать стабильное и функциональное приложение для визуализации ландшафта.

Заключение

Данный проект представляет собой реализацию визуализации трехмерного ландшафта на основе высотной карты. Основные результаты и выводы по проекту можно сформулировать более подробно.

Успешная загрузка и обработка данных высотной карты является ключевым аспектом проекта. Код поддерживает загрузку высотных данных из различных форматов файлов, таких как raw, tiff и csv. Функции GetImgSize, GetRawSize, GetCsvSize, LoadImgFile, LoadRawFile и LoadCsvFile обеспечивают корректное определение размера карты и загрузку данных в двумерный массив HeightMap. Это позволяет работать с высотными данными различного происхождения и формата. Преобразование двумерного массива в одномерный массив heightMap выполняется успешно для дальнейшего использования в визуализации.

Эффективная настройка OpenGL и создание окна являются важными шагами для подготовки к визуализации. Инициализация GLFW и GLEW позволяет создать окно с заданными размерами и настройками, готовое для визуализации ландшафта. Возможность изменения размера окна и полноэкранный режим реализованы корректно, что обеспечивает гибкость и адаптивность приложения.

Успешная компиляция и связывание шейдеров является необходимым условием для визуализации. Вершинный и фрагментный шейдеры компилируются без ошибок и связываются в единую программу для использования в визуализации. Шейдеры отвечают за преобразование вершин и расчет цвета фрагментов соответственно, что является ключевым элементом для отображения ландшафта.

Корректное создание вершин и индексов для визуализации ландшафта является важным этапом. Для каждой ячейки высотной карты генерируются вершины с соответствующими координатами и нормалями. Индексы формируют треугольники, образующие поверхность ландшафта. Вычисление нормалей выполняется правильно, обеспечивая корректное освещение поверхности.

Эффективное создание и заполнение буферов OpenGL позволяет оптимизировать процесс визуализации. Буферы VBO, EBO и VAO создаются и заполняются данными вершин, нормалей и индексов. Это позволяет эффективно отрисовывать ландшафт с использованием шейдерной программы, избегая повторной передачи данных на каждый кадр.

Реализация интерактивного управления камерой является важной функциональностью для исследования ландшафта. Обработка ввода пользователя, включая нажатия клавиш (W, S, A, D, стрелки, Esc) и движение мыши, позволяет перемещать и вращать камеру. Функции processInput и updateCameraVectors обеспечивают корректное обновление положения и ориентации камеры, что дает пользователю возможность исследовать ландшафт с разных ракурсов.

Визуализация ландшафта с применением техник освещения является заключительным этапом. В основном цикле приложения выполняется отрисовка ландшафта с использованием шейдерной программы и загруженных данных. Применяются техники ambient, diffuse и specular освещения для более реалистичного отображения поверхности. Это позволяет создать более реалистичное и привлекательное визуальное представление ландшафта.

В целом, проект демонстрирует успешную реализацию визуализации трехмерного ландшафта на основе высотной карты. Код поддерживает загрузку данных из различных форматов, эффективно настраивает OpenGL, создает вершины и индексы, управляет камерой и применяет техники освещения для получения реалистичного результата. Полученное приложение позволяет интерактивно исследовать ландшафт, созданный на основе высотных данных, и может быть использовано в различных областях, таких как геоинформационные системы, игровая индустрия или визуализация данных.

Список использованных источников

1 OpenGL Wiki: [сайт]. – URL: https://www.khronos.org/opengl/wiki/Main_Page (дата обращения: 14.06.2024)

2 Fast Light Toolkit [сайт]. – URL: https://www.fltk.org/ (дата обращения: 14.06.2024)

3 FLTK Programming Manual [сайт]. – URL: https://www.fltk.org/doc–1.3/index.html (дата обращения: 14.06.2024)

4 GitHub –g–truc/glm: OpenGL Mathematics (GLM) [сайт]. – URL: https://github.com/g–truc/glm (дата обращения 15.06.2024)

5 GitHub – glfw/glfw: A multi–platform library for OpenGL, OpenGL ES, Vulkan, window and input [сайт]. – URL: https://github.com/glfw/glfw (дата обращения 15.06.2024)

6 Лекции по компьютерной графики Ю.М.Баяковского 1999 [сайт]. – URL: https://www.graphicon.ru/oldgr/courses/cg99/index.htm (дата обращения 21.06.2024)

7 C++ OpenGL Tutorial [сайт]. – URL: https://youtu.be/_POT8K638VY?list=PLys-LvOneEETPlOI_PI4mJnocqIpr2cSHS (дата обращения 21.06.2024)

8 Алмазный – RnR Wiki [сайт]. – URL: https://rnr–wiki.ru/index.php?title=Алмазный (дата обращения 21.06.2024)

9 OpenGL (SDL. C++) tutorial [сайт]. – URL: https://youtu.be/pAHzHcUXsYA (дата обращения 22.06.2024)

10 EarthExplorer [сайт]. – URL: https://earthexplorer.usgs.gov/ (дата обращения 27.06.2024)

11 GitHub – libsdl–org/SDL: Simple Directmedia Layer [сайт]. – URL: https://github.com/libsdl–org/SDL (дата обращения 27.06.2024)

12 SDL3/FrontPage – SDL Wiki [сайт]. – URL: https://wiki.libsdl.org/SDL3/FrontPage (дата обращения 27.06.2024)

13 The CImg Library – C++ Template Image Processing Library [сайт]. – URL: https://cimg.eu/ (дата обращения 28.06.2024)