Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1
по курсу «Компьютерная графика»

Студент: Эсмедляев Ф.Р.
Группа: М80-312Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Цель работы

Целью данной лабораторной работы является изучение основ работы с графическим API для отрисовки 2D-примитивов с использованием библиотеки SFML в C++. Так- же изучены базовые 2D-трансформации (перемещение, масштабирование, поворот) и алгоритмы построения 2D-кривых.

Задача

В рамках данного задания реализована программа, которая рисует круг с фиксированным радиусом. Пользователь может перемещать круг по экрану, масштабировать его и вращать вокруг центра с использованием клавиатуры и мыши. Дополнительно визуализируется путь, по которому перемещается центр круга.

Метод решения

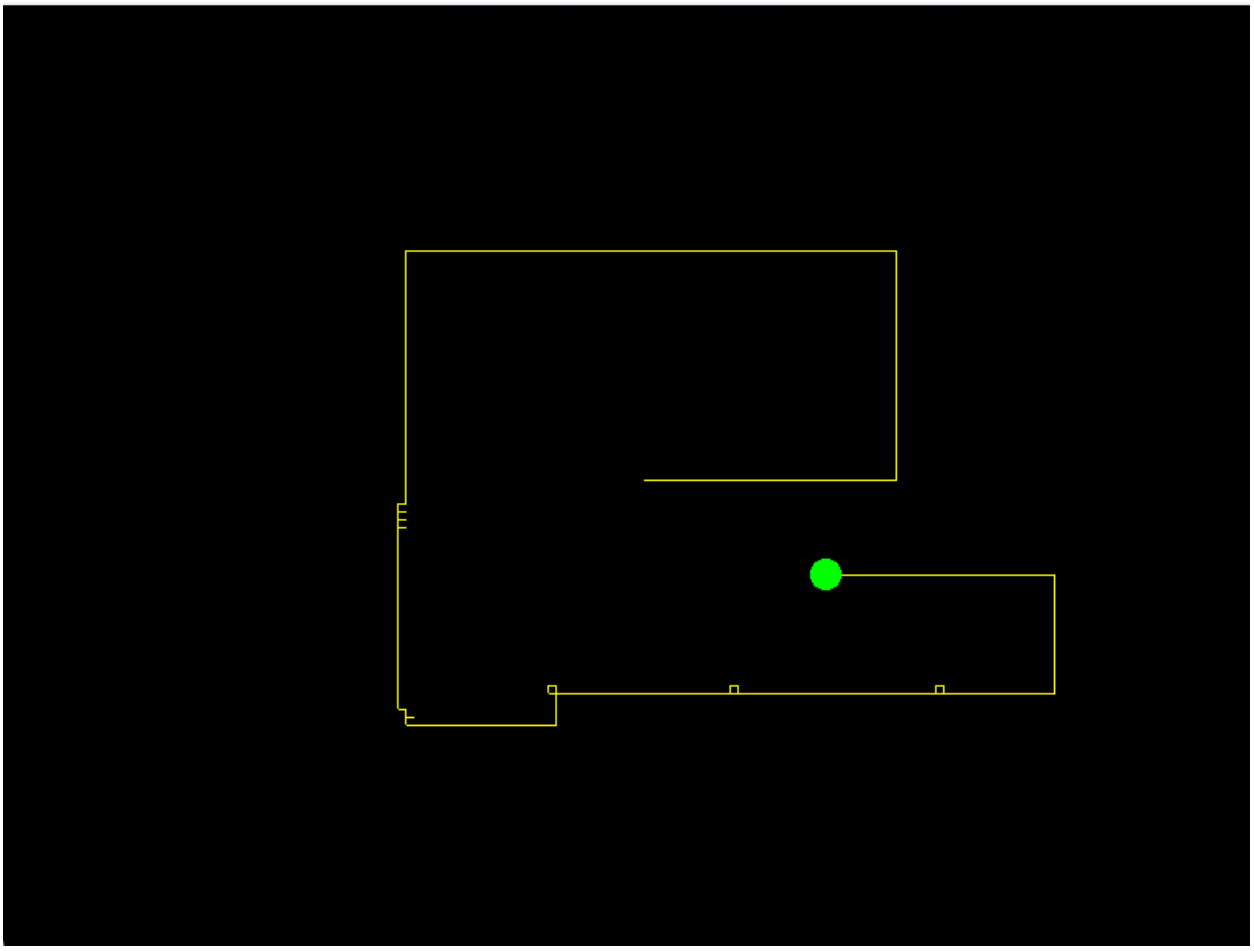
Для выполнения задания использовались язык программирования C++ и библиотека SFML. Основные шаги решения:

1. **Инициализация окна**: Создается окно размером 800x600 пикселей для отображения графики.
2. **Отрисовка круга**: Используется класс `sf::CircleShape`, который позволяет создавать круги с заданным радиусом. Круг изначально центрируется для удобства вращения.
3. **Управление кругом**:
   - Перемещение осуществляется с помощью клавиш `W`, `A`, `S`, `D`.
   - Масштабирование — клавишами `Up` и `Down`.
   - Вращение — клавишами `Left` и `Right`.
4. **Визуализация пути**: Для каждой позиции центра круга сохраняются координаты, и на экране рисуется линия движения.
5. **Цикл обновления**: Программа обновляет изображение в цикле с учетом пользовательских команд для трансформаций круга.

Основной код программы

Программа написана на языке C++ с использованием библиотеки SFML. Основные функции программы:

- Создание окна с заданными параметрами.
- Реализация механики управления перемещением, масштабированием и вращением круга.
- Построение и отображение пути перемещения центра круга.



Результаты работы

В ходе работы над проектом были реализованы базовые операции с 2D-графикой, включая перемещение, масштабирование и вращение объекта. Кроме того, была добавлена визуализация движения объекта по экрану в виде линии, что добавило анимации и интерактивности программе.

Выводы

В процессе выполнения лабораторной работы я изучил основы работы с графикой и трансформациями на языке C++ с использованием библиотеки SFML. Я также освоил применение базовых анимационных и трансформационных техник, таких как перемещение, масштабирование и вращение, что будет полезно при разработке простых 2D-игр или визуализаций.

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №2
по курсу «Компьютерная графика»

Студент: Эсмедляев Ф.Р.
Группа: М80-312Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Цель работы

Цель этой лабораторной работы — изучить основы 3D-графики, включая создание простых 3D-объектов, их проекцию на 2D-плоскость, а также освоить работу с матрицами перспективной и ортографической проекций.

Задача

В рамках данного задания реализована программа, которая строит 3D-куб. Пользователь может вращать куб вокруг его осей, а также наблюдать изменения нормалей для каждой грани в зависимости от поворота. Куб отображается на экране с использованием перспективной проекции. Дополнительно реализована визуализация нормалей и возможность изменения углов вращения.

Метод решения

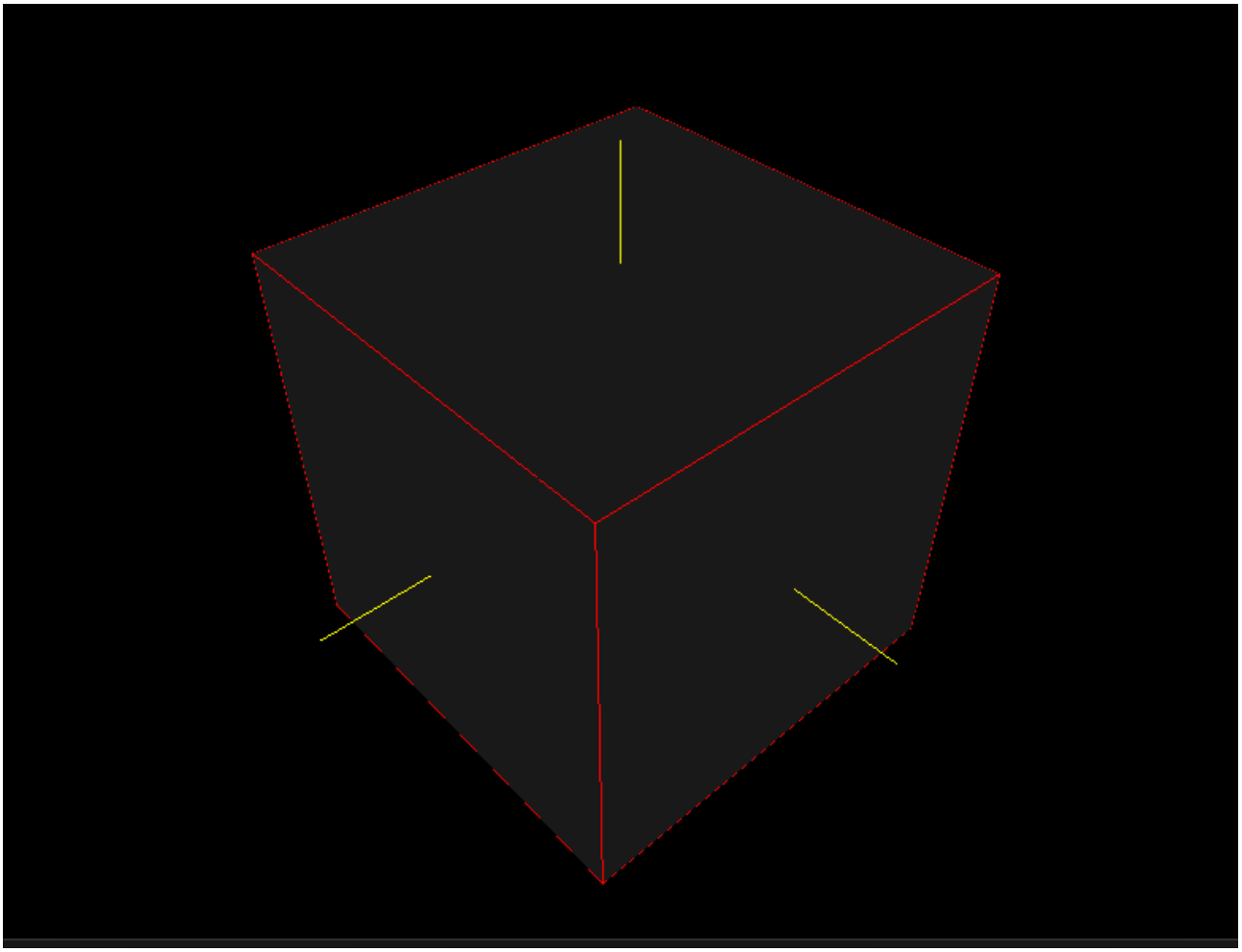
Для выполнения задания использовались язык программирования C++ и библиотека SFML. Основные шаги решения:

1. **Инициализация окна**: Создается окно размером 800x600 пикселей с использованием SFML для отображения графики. Активируется режим глубины для корректного отображения 3D-объектов.
2. **Отрисовка куба**:
    - Куб создается с использованием OpenGL. Вершины и грани определены в мировых координатах.
    - Визуализируются рёбра куба и нормали к его граням.
    - Нормали отображаются в виде желтых линий, выходящих из центра каждой грани.
3. **Управление кубом**:
    - Вращение вокруг осей X и Y осуществляется с помощью клавиш W, A, S, D.
    - Повороты реализованы путем изменения углов вращения и применения соответствующих трансформаций с использованием матриц OpenGL.
4. **Перспективная проекция**: Используется функция glFrustum для установки перспективной проекции. Камера фиксируется перед кубом, создавая эффект глубины.
5. **Цикл обновления**: Программа постоянно обновляет изображение в

цикле, реагируя на команды пользователя и перерисовывая куб с учетом текущих параметров вращения.

Основной код программы

Программа написана на языке C++ с использованием библиотеки SFML и OpenGL. Основные функции программы:
• Создание окна с заданными параметрами, поддерживающего 3D-графику.
• Реализация механики управления вращением куба вокруг осей X и Y с использованием клавиш клавиатуры.
• Построение и отображение 3D-куба с визуализацией его рёбер и нормалей к граням.
• Настройка перспективной проекции для реалистичного отображения глубины.



Результаты работы

В ходе работы над проектом были реализованы базовые операции с 3D-

графикой, включая построение, отображение и вращение куба вокруг осей X и Y. Кроме того, была добавлена визуализация нормалей к граням куба, что улучшило понимание ориентации объекта в пространстве. Реализация перспективной проекции добавила реалистичности и глубины отображению.

Выводы

В процессе выполнения лабораторной работы я изучил основы работы с 3D-графикой и трансформациями на языке C++ с использованием библиотек SFML и OpenGL. Я освоил построение объектов, применение перспективной проекции, а также визуализацию нормалей и управление вращением, что будет полезно при разработке 3D-программ или визуализаций сложных объектов.

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3
по курсу «Компьютерная графика»

Студент: Эсмедляев Ф.Р.
Группа: М80-312Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Цель работы

Цель этой лабораторной работы — изучить основы работы с камерой в 3D-пространстве, включая управление её положением и направлением, а также освоить базовые трансформации объектов: перемещение, поворот и масштабирование.

Задача

В рамках данного задания реализована программа, которая строит 3D-куб. Пользователь может вращать куб вокруг осей X, Y и Z независимо друг от друга, используя клавиатуру. Куб отображается на экране с использованием перспективной проекции. Дополнительно реализована плавная анимация вращения с возможностью изменения скорости вращения.

Метод решения

Для выполнения задания использовались язык программирования C++ и библиотека SFML с OpenGL для создания и управления 3D-кубом. Основные шаги решения:

1. **Инициализация окна**:
    - Создается окно размером 800x600 пикселей с использованием SFML для отображения графики. Для корректного отображения 3D-объектов активируется режим глубины через glEnable(GL_DEPTH_TEST), что позволяет правильно отобразить объекты в трехмерном пространстве.
2. **Отрисовка куба**:
    - Куб создается с использованием OpenGL. Вершины куба задаются в мировых координатах, а его рёбра рисуются с помощью команды glBegin(GL_LINES), что позволяет отобразить его как каркас.
    - Грани куба создаются с помощью glBegin(GL_QUADS), каждая грань имеет нормаль, которая вычисляется с использованием функции glNormal3f. Нормали отображаются как ориентированные вектора, что помогает визуализировать ориентацию каждой грани.
3. **Управление кубом**:
    - Вращение куба вокруг осей X, Y и Z осуществляется с помощью клавиш W, S (для вращения по оси X), A, D (для оси Y), и Q, E (для оси Z). Каждая из этих клавиш изменяет соответствующую скорость вращения.

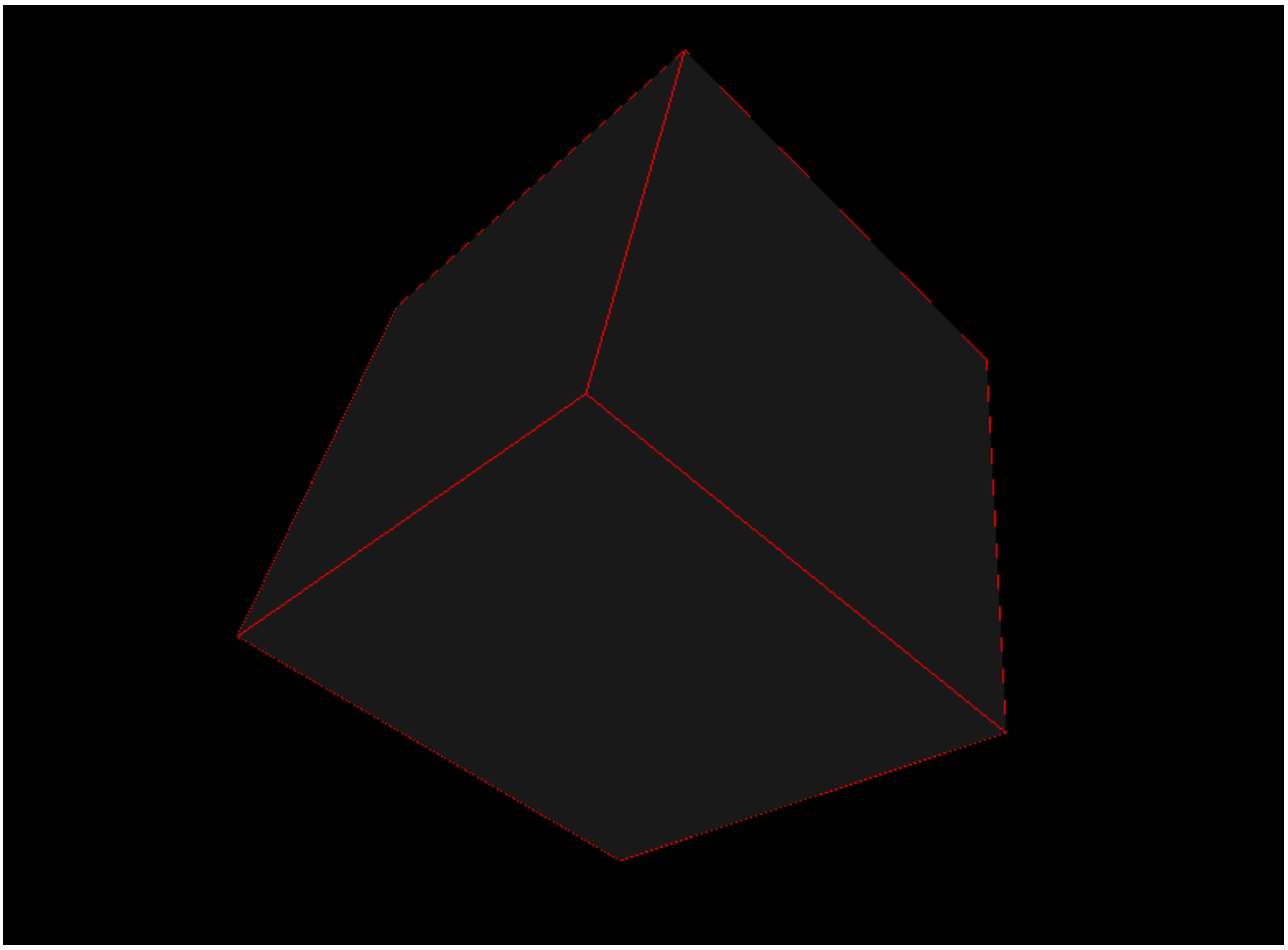
- Повороты куба реализованы через изменения углов вращения для каждой оси и применения соответствующих трансформаций с использованием функции glRotatef, которая вращает объект в 3D-пространстве.

4. **Перспективная проекция**:
   - Для создания эффекта глубины используется функция glFrustum, которая настраивает перспективную проекцию, принимая во внимание поле зрения, аспекты экрана, а также ближнюю и дальнюю плоскости отсечения.
   - Камера фиксируется перед кубом на расстоянии, создавая эффект глубины при изменении углов и масштаба.
5. **Цикл обновления**:
   - Программа использует главный цикл, в котором постоянно обновляются параметры вращения куба. Каждый раз происходит перерисовка изображения с актуальными значениями углов вращения и отображением новых положений нормалей.
   - В цикле также обрабатываются события с клавиатуры для изменения скорости вращения, а также сброса параметров при нажатии клавиши Space.

Основной код программы

Программа реализована на языке C++ с использованием библиотеки SFML и OpenGL. Основные функции программы:

- **Создание окна** с заданными параметрами с использованием SFML.
- **Отрисовка 3D-куба** с рёбрами и гранями, используя OpenGL. Грани отображаются с нормалями для правильного освещения.
- **Управление вращением** куба. Реализована независимая настройка скорости вращения по осям X, Y и Z через клавиши W, S, A, D, Q, E.
- **Плавная анимация** вращения с возможностью изменять скорость вращения.
- **Сброс углов вращения** и скорости при нажатии клавиши Space.



Результаты работы

В ходе работы над проектом был реализован куб в 3D-пространстве с возможностью вращения вокруг осей X, Y и Z. Программа позволяет пользователю динамически изменять скорость вращения этих осей с клавиатуры. Дополнительно реализована плавная анимация вращения, а также возможность сбросить все настройки вращения в исходное состояние.

Выводы

В процессе выполнения лабораторной работы я освоил основные принципы работы с 3D-графикой и базовыми трансформациями на языке C++ с использованием OpenGL через SFML. Я научился создавать 3D-объекты, управлять их вращением и реализовывать плавные анимации. Полученные знания будут полезны при разработке более сложных графических приложений и игр, а также в области визуализаций 3D-данных.

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4
по курсу «Компьютерная графика»

Студент: Эсмедляев Ф.Р.
Группа: М80-312Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Цель работы

Цель этой лабораторной работы — изучить основы работы с освещением в 3D-пространстве, используя различные типы источников света, а также освоить написание шейдеров для реализации освещения объектов в сцене с применением простейших моделей освещения.

Задача

В рамках данного задания реализована программа, которая строит 3D-сферу. Пользователь может наблюдать различные эффекты освещения, применяя метод гладкого затенения (Gouraud Shading), при котором освещение рассчитывается на вершинах объекта и интерполируется по его поверхности. Также реализована возможность динамического переключения между методами Gouraud Shading и Flat Shading для наглядного сравнения этих эффектов. В работе используются как вершинный, так и фрагментный шейдер для достижения желаемого результатаим мсар.

Метод решения

Для выполнения задания использовались язык программирования C++ и библиотеки SFML и OpenGL для построения и отображения 3D-сферы с применением методов затенения Gouraud и Flat Shading. Основные шаги решения:

1. **Инициализация окна:**
    - Создается окно размером 800x600 пикселей с использованием SFML для отображения графики. Для корректного отображения 3D-объектов активируется режим глубины с помощью glEnable(GL_DEPTH_TEST), что позволяет корректно отображать объекты в трехмерном пространстве.
    - Инициализация библиотеки GLEW для работы с расширениями OpenGL, что необходимо для использования новых функций OpenGL.
2. **Генерация сферы:**
    - Для построения сферы используется функция generateSphere, которая генерирует вершины и индексы для сферы, заданной радиусом, количеством делений по долготе и широте.
    - Вершины сферы вычисляются через сферические координаты, и для каждой вершины рассчитываются нормали, направленные от центра сферы. Эти нормали будут использоваться для расчета

освещения.

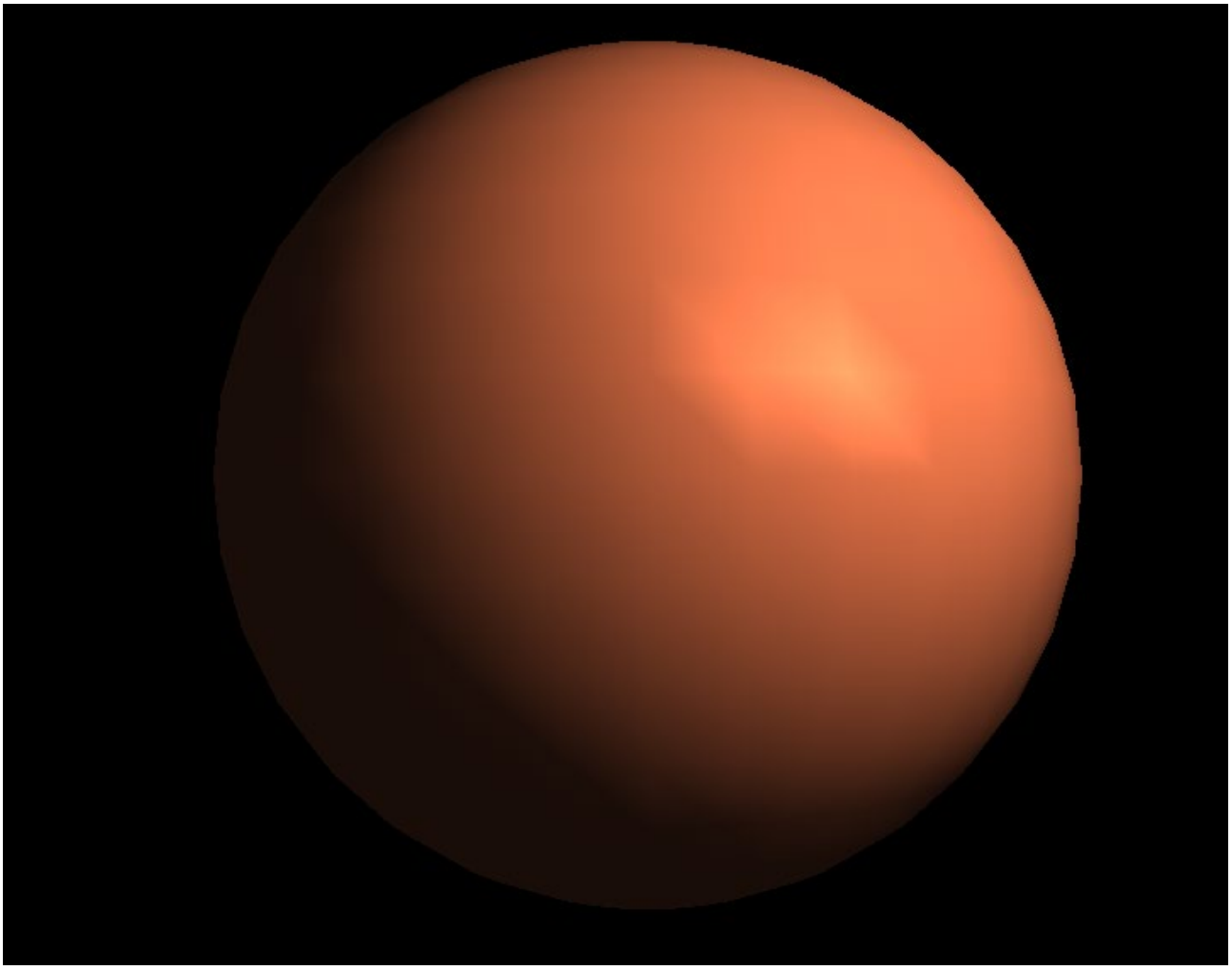
3. **Создание и компиляция шейдеров:**
    - Для реализации различных типов затенения создаются два набора шейдеров:
        - Вершинный шейдер для **Gouraud Shading** рассчитывает освещение на каждой вершине, а затем интерполирует результат по поверхности.
        - Вершинный шейдер для **Flat Shading** рассчитывает освещение для одной вершины и применяет его ко всем вершинам грани.
    - Для каждого типа затенения также создается фрагментный шейдер, который отображает итоговый цвет на поверхности сферы.
4. **Визуализация сферы:**
    - Сфера отображается с использованием VAO (Vertex Array Object), VBO (Vertex Buffer Object) и EBO (Element Buffer Object). Эти объекты используются для хранения данных о вершинах, нормалях и индексах для эффективной передачи данных в OpenGL.
    - Вершины передаются в виде массива данных в OpenGL, и используется команда glDrawElements для отрисовки сферической сетки.
5. **Переключение между шейдерами:**
    - Программа поддерживает динамическое переключение между **Gouraud Shading** и **Flat Shading** при нажатии клавиши пробела. Это позволяет сравнивать два различных подхода к затенению: плавное (Gouraud) и плоское (Flat).
    - В зависимости от выбранного шейдера используется соответствующий набор вершинных и фрагментных шейдеров, которые передаются в OpenGL для дальнейшего выполнения.
6. **Управление трансформациями:**
    - Для отображения сферы в 3D-пространстве используются матрицы трансформации (model, view, projection), которые создаются с использованием библиотеки GLM.
    - Камера настроена с помощью функции glm::lookAt, чтобы сфера располагалась перед камерой, и используется перспективная проекция с помощью функции glm::perspective.

7. **Цикл обновления:**
    - Программа работает в основном цикле, где постоянно обновляются параметры трансформации и освещения. При каждом обновлении происходит перерисовка сцены с учетом актуальных значений матриц и освещения.
    - Обрабатываются события с клавиатуры, позволяя переключать шейдеры, а также контролировать поведение сцены в реальном времени.
8. **Очистка ресурсов:**
    - По завершению работы программы освобождаются ресурсы, такие как VAO, VBO, EBO, для корректного завершения работы OpenGL.

Основной код программы

Программа реализована на языке C++ с использованием библиотеки SFML и OpenGL. Основные функции программы:

- **Создание окна и инициализация OpenGL**: окно размером 800x600 пикселей создается с использованием SFML. Включен режим глубины для правильного отображения 3D-объектов.
- **Генерация 3D-сферы**: с использованием OpenGL создается сфера с заданным радиусом, количеством долгот и широт. Вершины и индексы для отображения сферы сохраняются в векторах.
- **Реализация шейдеров**: используются два шейдера — для **Gouraud Shading** и **Flat Shading**. Вершинный и фрагментный шейдеры рассчитывают освещение на основе позиции источника света, нормалей и камеры.
- **Переключение между шейдерами**: при нажатии клавиши Space происходит переключение между Gouraud и Flat шейдерами, что позволяет изменять способ отображения освещенности.
- **Рендеринг сцены**: каждый кадр сцена очищается, затем применяется выбранный шейдер, и происходит рендеринг сферы с передачей матриц трансформации, а также параметров источника света и камеры.
- **Основной цикл**: программа обрабатывает события закрытия окна и клавишу для переключения шейдеров, обновляет изображение на экране.



Результаты работы

В ходе выполнения проекта была реализована программа, создающая 3D-модель сферы с использованием OpenGL и SFML. Реализованы два способа освещения: **Gouraud Shading** (освещение вычисляется на вершинах) и **Flat Shading** (освещение вычисляется для каждой грани). Программа позволяет переключаться между этими режимами освещения в реальном времени с помощью клавиши Space. Световой источник, камера и параметры материала учитываются при расчете освещения. Сфера плавно отображается и корректно освещается в зависимости от выбранного метода.

Выводы

В процессе выполнения лабораторной работы я освоил основы работы с OpenGL, включая генерацию геометрии, настройку шейдеров и управление освещением. Ознакомился с принципами Gouraud и Flat освещения, а также их влиянием на визуальное восприятие 3D-объектов. Эти знания являются важной основой для разработки графических приложений, визуализаций и игр, где требуется качественная работа с 3D-графикой и реалистичным освещением.

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №5
по курсу «Компьютерная графика»

Студент: Эсмедляев Ф.Р.
Группа: М80-312Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Цель работы

Цель этой лабораторной работы — изучить основы техники трассировки лучей (Ray Tracing) для создания реалистичной 3D-графики. В рамках работы вы реализуете алгоритмы, позволяющие рассчитывать физически корректные отражения, преломления, тени и взаимодействие света с объектами в сцене. Выполнение лабораторной работы способствует пониманию основ рендеринга на базе работы с лучами света и созданию визуально правдоподобных 3D-сцен.

Задача

В рамках данного задания реализована программа, которая строит сцену, состоящую из одной сферы и плоскости. Освещение рассчитывается с использованием метода трассировки лучей, основанного на модели затенения Фонга (Phong Shading). Эта модель учитывает диффузное, зеркальное и окружающее освещение, обеспечивая реалистичное отображение света на поверхности объектов.

Дополнительно реализована возможность изменения параметров модели освещения Фонга, что позволяет наблюдать различные визуальные эффекты и оценить влияние параметров на качество рендеринга.

Метод решения

Для выполнения задачи использовались язык программирования C++ и библиотеки SFML и GLM для реализации алгоритма трассировки лучей с использованием модели затенения Фонга. Основные шаги решения включают:

1. **Инициализация окна:**
    - Создается окно размером 800x600 пикселей с использованием SFML. Оно используется для отображения сгенерированной сцены.
    - Устанавливается частота кадров (60 FPS) для обеспечения плавности анимации.
2. **Описание основных объектов:**
    - **Сфера:** Представлена как структура Sphere, включающая положение, радиус и цвет. Для расчета пересечений лучей со сферой используется математическое решение квадратного уравнения.
    - **Плоскость:** Представлена как структура Plane, включающая точку на плоскости, нормаль и цвет. Пересечение луча с плоскостью рассчитывается с использованием скалярного произведения.

3. **Реализация модели затенения Фонга:**
    - В функции phongShading рассчитываются три компонента освещения:
        - **Окружающее освещение (Ambient):** Представляет равномерный свет, добавляющий базовую яркость объектам.
        - **Диффузное освещение (Diffuse):** Основано на угле падения света на поверхность объекта.
        - **Зеркальное освещение (Specular):** Рассчитывается как отражение света относительно поверхности, чтобы добавить блеск.
    - Итоговый цвет комбинирует все три компонента.
4. **Трассировка лучей:**
    - Функция traceRay отвечает за определение объектов, с которыми пересекается луч, и вычисление цвета пикселя.
    - Луч пересекается со всеми объектами сцены, а ближайшая точка пересечения используется для расчета освещения.
    - Если пересечений нет, пикселю назначается цвет фона.
5. **Создание сцены:**
    - Сцена состоит из одной сферы и плоскости. Их параметры (положение, радиус, цвет) задаются в начале программы.
    - Источник света располагается в трехмерном пространстве и используется для вычислений освещения.
6. **Управление параметрами освещения:**
    - Параметры модели Фонга (интенсивности окружающего, диффузного, зеркального освещения, а также степень блеска) изменяются в реальном времени с помощью клавиш:
        - **Q/A:** Увеличение/уменьшение окружающего освещения.
        - **W/S:** Увеличение/уменьшение диффузного освещения.
        - **E/D:** Увеличение/уменьшение зеркального освещения.
        - **R/F:** Увеличение/уменьшение степени блеска.
7. **Растеризация и отрисовка:**

- Для каждого пикселя экрана вычисляется направление луча, исходящего из камеры.
- Функция traceRay возвращает цвет пикселя, который затем записывается в объект sf::Image.
- Готовое изображение отображается в окне с использованием sf::Texture и sf::Sprite.

8. **Цикл обновления:**
   - В основном цикле программы обрабатываются события, обновляются параметры освещения и генерируется новое изображение сцены.
   - Пользователь может изменять параметры освещения, наблюдая влияние на рендеринг сцены.
9. **Очистка ресурсов:**
   - После завершения работы окна программа завершает свою работу, освобождая все задействованные ресурсы, связанные с SFML.

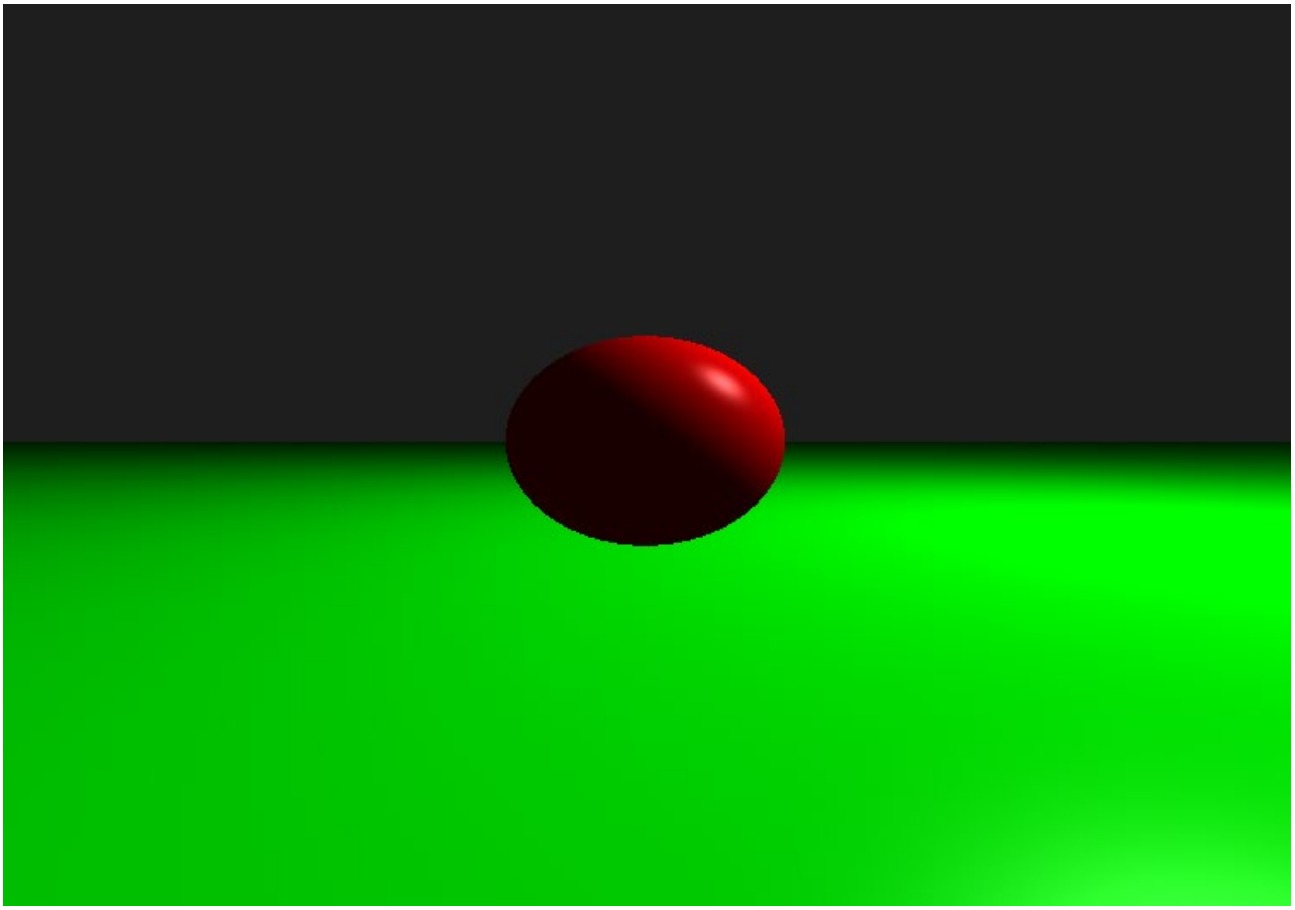
Основной код программы

Программа реализована на языке C++ с использованием библиотеки SFML и OpenGL. Основные функции программы:

- **Создание окна и инициализация OpenGL**: окно размером 800x600 пикселей создается с использованием SFML. Включен режим глубины для правильного отображения 3D-объектов.
- **Генерация 3D-сферы**: с использованием OpenGL создается сфера с заданным радиусом, количеством долгот и широт. Вершины и индексы для отображения сферы сохраняются в векторах.
- **Реализация шейдеров**: используются два шейдера — для **Gouraud Shading** и **Flat Shading**. Вершинный и фрагментный шейдеры рассчитывают освещение на основе позиции источника света, нормалей и камеры.
- **Переключение между шейдерами**: при нажатии клавиши Space происходит переключение между Gouraud и Flat шейдерами, что позволяет изменять способ отображения освещенности.
- **Рендеринг сцены**: каждый кадр сцена очищается, затем применяется выбранный шейдер, и происходит рендеринг сферы с передачей матриц

трансформации, а также параметров источника света и камеры.

- **Основной цикл**: программа обрабатывает события закрытия окна и клавишу для переключения шейдеров, обновляет изображение на экране.



Результаты работы

В ходе выполнения лабораторной работы была реализована программа, которая создает 3D-сцену, содержащую сферу и плоскость. Для отображения сцены использовалась техника трассировки лучей с применением модели затенения Фонга. Программа позволяет рассчитывать окружающее, диффузное и зеркальное освещение, а также отображает реалистичные тени и блики.

Реализована возможность динамического изменения параметров модели освещения Фонга:

- Окружающее освещение (Ambient Strength).
- Диффузное освещение (Diffuse Strength).
- Зеркальное освещение (Specular Strength).
- Степень блеска (Shininess).

Программа наглядно демонстрирует влияние этих параметров на визуализацию сцены, позволяя пользователю оценить эффекты освещения в реальном времени.

Выводы

В процессе выполнения лабораторной работы я освоил основные принципы работы алгоритма трассировки лучей и модели затенения Фонга. Изучены и реализованы:

- Алгоритмы пересечения лучей с примитивами (сферой и плоскостью).
- Методы расчета нормалей и освещения с учетом углов падения и отражения света.
- Использование библиотек SFML и GLM для реализации графических сцен и математических операций.

Работа показала, как изменение параметров освещения влияет на реалистичность сцены, и подчеркнула важность правильного расчета освещения в 3D-графике. Полученные знания закладывают фундамент для дальнейшего изучения и реализации более сложных графических систем и алгоритмов визуализации.

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №6
по курсу «Компьютерная графика»

Студент: Эсмедляев Ф.Р.
Группа: М80-312Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Цель работы

Цель этой лабораторной работы — разработать основу для собственного игрового движка, объединяя ранее изученные техники работы с 2D- и 3D-графикой, освещением, шейдерами и трассировкой лучей. В ходе работы вы создадите простой движок, способный отрисовывать сцены в реальном времени, с поддержкой управления камерой, объектами и освещением. Выполнение лабораторной работы способствует развитию навыков интеграции графических технологий и созданию базового инструмента для разработки игр.

Задача

1. Создать базовую архитектуру игрового движка с поддержкой 3D-сцен.
2. Реализовать систему рендеринга с использованием изученных подходов: рендеринг через растризацию и трассировку лучей.
3. Настроить камеру и систему управления ею.
4. Реализовать поддержку базовых игровых объектов и взаимодействие с ними.
5. Обеспечить работу с несколькими типами источников света.
6. Оптимизировать производительность движка (по возможности).

Метод решения

Для выполнения лабораторной работы использовались язык программирования C++ и технологии OpenGL. Основные шаги решения следующие:

1. **Инициализация проекта:**
   - Проект конфигурируется через CMakeLists.txt.
   - Подключаются зависимости, включая библиотеку ImGui (для пользовательского интерфейса) и JSON.
2. **Работа с 3D-объектами:**
   - В папке assets/objects хранятся 3D-модели в формате .obj, включая фигуры: сфера, куб, цилиндр, конус, обезьяна и плоскость.
3. **Использование шейдеров:**
   - Шейдеры расположены в папке assets/shaders:
     - default.vert — вершинный шейдер для обработки вершин.
     - default.frag — фрагментный шейдер для расчета цвета пикселей.
4. **Кодовая база:**
   - Основной код программы находится в main.cpp и организован с использованием классов:
     - **Классы для рендеринга:**

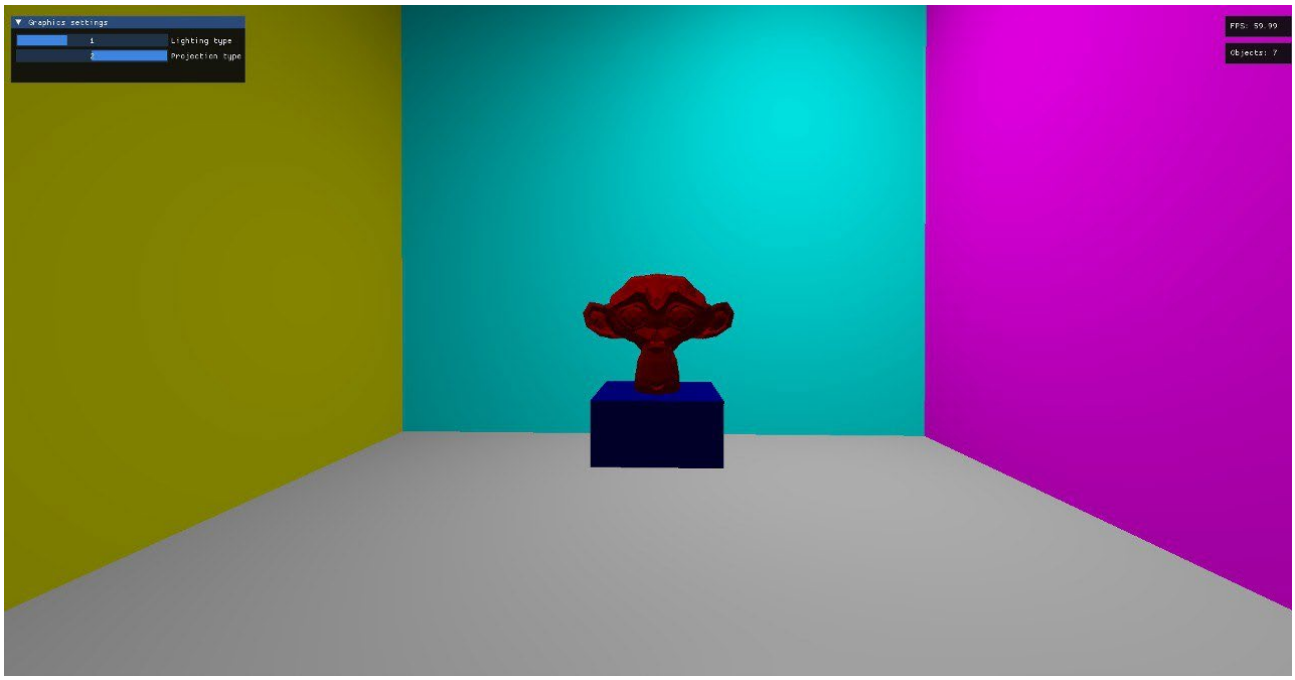
        - Mesh (для работы с объектами),
        - Scene (для управления сценой),
        - Camera (для управления камерой).
    - **Классы для работы с OpenGL:**
        - VAO, VBO, shaderClass.
5. **Отображение и взаимодействие:**
    - Используется библиотека ImGui для взаимодействия с пользователем. Реализованы элементы интерфейса для работы с объектами и их свойствами.
6. **Цикл рендеринга:**
    - Отображение сцены обновляется в основном цикле программы:
        - Камера, матрицы трансформации и освещение применяются ко всем объектам.
        - Отрисовка осуществляется через OpenGL с использованием VAO и VBO.
7. **Тестирование:**
    - В файле config.json могут быть сохранены параметры сцены для настройки начального состояния.

Основной код программы

Программа реализована на языке C++ с использованием OpenGL и вспомогательных библиотек (ImGui для интерфейса и GLM для работы с матрицами). Основные функции программы:

- **Создание окна и инициализация OpenGL:**
    - Окно создается с использованием GLFW. Задается начальное разрешение 1280x720 пикселей.
    - Включается режим глубины (GL_DEPTH_TEST) для корректной отрисовки 3D-объектов, чтобы ближние элементы перекрывали дальние.
- **Загрузка 3D-моделей:**
    - В папке assets/objects хранятся 3D-модели (cube.obj, sphere.obj, monkey.obj и другие), которые загружаются и используются для отображения.
    - Модели разбиваются на вершины и индексы для передачи в OpenGL.
- **Реализация шейдеров:**
    - Вершинный шейдер (default.vert) выполняет преобразование координат и расчет нормалей.
    - Фрагментный шейдер (default.frag) отвечает за освещение и окрашивание пикселей, используя источник света и нормали объекта.
- **Управление камерой:**

    - Камера реализована в классе Camera с поддержкой перемещения и вращения.
    - Для управления используются стандартные функции GLM (glm::lookAt, glm::perspective) для построения вида и проекции.
- **Отрисовка сцены:**
    - Каждый кадр обновляется текущая позиция объектов, камеры и освещения.
    - Используются VAO и VBO для хранения информации о вершинах, нормалях и текстурах объектов.
    - Выполняется вызов glDrawElements для отрисовки всех объектов сцены.
- **Пользовательский интерфейс:**
    - Библиотека ImGui используется для настройки параметров объектов, таких как масштаб, положение и цвет.
    - Реализовано динамическое управление освещением и переключение между моделями через интерфейс.
- **Основной цикл:**
    - Программа обрабатывает ввод с клавиатуры и мыши для управления камерой и объектами.
    - Постоянно обновляются матрицы трансформации и параметры шейдеров для отображения актуальной сцены.
    - При завершении работы все ресурсы (буферы, VAO, текстуры) освобождаются.



Результаты работы

В результате выполнения лабораторной работы была создана программа, которая позволяет визуализировать 3D-сцены с использованием OpenGL и взаимодействовать с ними через интерфейс. Реализованы следующие функции:

- Отрисовка объектов (сфера, куб, обезьяна и другие) с использованием моделей в формате .obj.
- Использование двух типов шейдеров для визуализации сцены: базовый расчет освещения и текстурирование.
- Управление камерой для перемещения и изменения угла обзора.
- Настройка параметров объектов и освещения через интерфейс на основе ImGui.
- Визуализация сцены в реальном времени с учетом изменений в положении камеры, источника света и объектов.

Программа успешно отрисовывает сцены с реалистичным освещением и предоставляет гибкость в настройке параметров объектов и камеры.

Выводы

В ходе выполнения лабораторной работы были изучены и применены на практике основы работы с OpenGL, включая загрузку моделей, настройку шейдеров и использование VAO/VBO для управления геометрией. Освоено использование библиотеки ImGui для реализации пользовательского интерфейса.

Эти навыки помогут в дальнейшем при разработке 3D-приложений, создании игровых движков и графических визуализаций. Проект подчеркнул важность структурированного подхода к работе с графическими API, а также позволил понять взаимодействие между геометрией, шейдерами и освещением.