Московский авиационный институт

(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

Лабораторные работы

по курсу «Компьютерная графика»

Студент: Меркулов Ф.А.

Группа: М8О-307Б-21

Преподаватель: Филиппов Г.Б.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2023

Лабораторная работа №1

Тема: Построение изображений 2D-кривых.

Задача: Написать и отладить программу, строящую изображение заданной замечательной кривой.

Вариант №2: $(x^2 + y^2)^2 = a^2(x^2 - y^2)$

1 Решение

При написании программы, строящей изображение замечательной кривой $(x^2 + y^2)^2 = a^2(x^2 - y^2)$ были использованы библиотеки glew.h, glfw3.h, cmath и freeglut.h.

1. <glew.h>: относится к библиотеке GLEW, что расшифровывается как OpenGL Extension Wrangler Library. Это кросс-платформенная open-source библиотека, которая используется в программировании с OpenGL. Основная функция GLEW - облегчить доступ и использование расширений OpenGL, предоставляемых различными графическими драйверами.
2. <glfw3.h>: является частью библиотеки GLFW, которая расшифровывается как Graphics Library Framework. GLFW - это портативная библиотека для управления окнами, вводом и событиями, предназначенная для использования с OpenGL и другими графическими API. Она широко используется в области компьютерной графики для создания и управления окнами, а также для обработки ввода от пользователя (например, клавиатуры и мыши).
3. <cmath>: это заголовочный файл в стандартной библиотеке языка программирования C++. Он предоставляет различные математические функции и является C++ версией заголовочного файла <math.h>, который использовался в языке C.
4. <freeglut.h>: является частью библиотеки FreeGLUT, которая представляет собой альтернативу библиотеке OpenGL Utility Toolkit (GLUT). FreeGLUT разработана как более свободная и функционально обогащенная версия GLUT, предназначенная для создания и управления окнами в приложениях, использующих OpenGL.

Внутри цикла while (!glfwWindowShouldClose(window)) я сначала нарисовал и подписал оси белым цветом, а поверх них нарисовал график функции $(x^2 + y^2)^2 = a^2(x^2 - y^2)$.

Рисование осей. В начале отрисовываются горизонтальная (будущая абсцисса) и вертикальная (будущая ордината) линии белыми цветами, затем наносятся масштабные линии после чего подписываются масштабные лини и оси и наконец отрисовываются стрелочки на осях.

Рисование замечательной кривой. Уравнение Лемнискаты Бернулли имеет вид: $(x^2 + y^2)^2 = 2c^2(x^2 - y^2)$, и оно имеет решение относительно y:

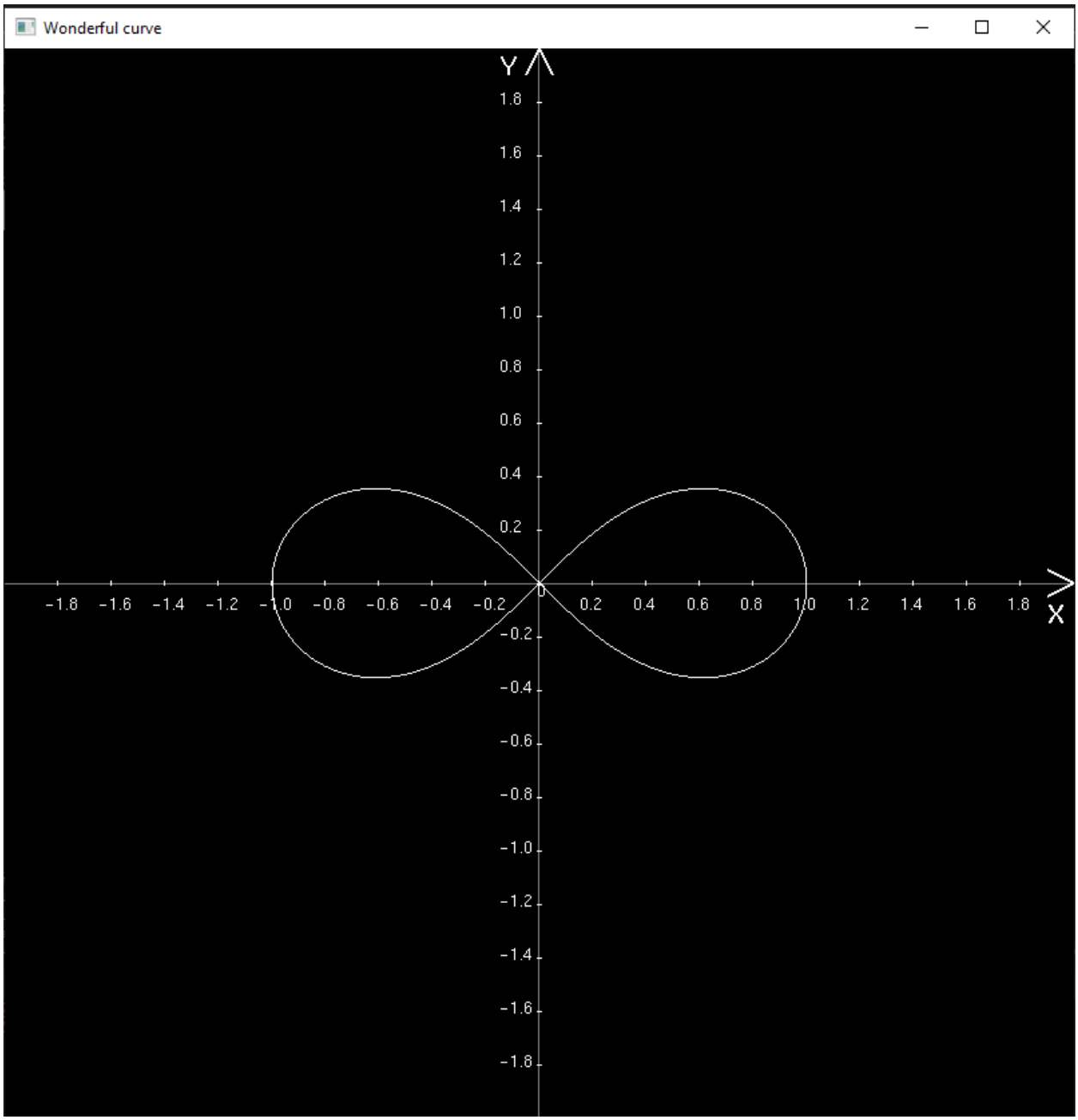
$$y = \pm\sqrt{\sqrt{c^4 + 4x^2c^2} - x^2 - c^2}$$

Можем заметить, что наша замечательная кривая $(x^2 + y^2)^2 = a^2(x^2 - y^2)$ имеет абсолютно такую же структуру, единственное надо произвести замену $a^2 = 2c^2$, то есть $c^2 = \frac{a^2}{2}$ из-за чего получаем уравнение относительно y:

$$y = \pm\sqrt{\sqrt{(\frac{a^2}{2})^2 + 4x^2(\frac{a^2}{2})} - x^2 - (\frac{a^2}{2})}$$

В итоге для отрисовки замечательной кривой используется шаг $iterator = 0.0001$, по которому перебираются всевозможные x в промежутке $[-a; a]$ и для каждого x высчитываются y по формуле выше и таким образом высчитываются всевозможные точки, которые соединяются и получается замечательная кривая

Результат работы программы:



2 Вывод

Программа представляет собой комплексную лабораторную работу, в которой реализована визуализация 2D-кривых с использованием библиотек OpenGL и GLFW. Основной фокус работы направлен на создание графического интерфейса для отображения замечательной кривой.

Использование OpenGL и GLFW позволяет создать окно приложения и обеспечивают инструменты для рисования 2D-графиков. Наличие подписанных осей и лучей упрощает ориентацию в пространстве, а также позволяет легче воспринимать форму и характеристики кривой. Регулировка масштаба и центрирование графика обеспечивают лучшее визуальное восприятие и адаптацию под разные размеры окна.

Лабораторная работа №2

Тема: Каркасная визуализация выпуклого многогранника. Удаление невидимых линий.

Задача: Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

Вариант №2: Правильный октаэдр

1 Решение

Для реализации фигуры «правильный октаэдр», я использовал библиотеки и инструменты Qt, что позволило мне эффективно создать и визуализировать эту трёхмерную геометрическую форму, а также я реализовал вращение фигуры, изменение масштаба и удаление невидимых линий.

В проекте для визуализации фигуры "правильный октаэдр" была использована мощная и гибкая среда Qt. Программа структурирована с использованием нескольких классов, каждый из которых выполняет свою специфическую функцию, обеспечивая чёткое разделение логики визуализации и управления интерфейсом.

1. Класс MainWindow:
    - Отвечает за создание и управление главным окном приложения.
    - Инициализирует объект Octahedron (октаэдр) и управляет его визуализацией.
    - Включает обработку событий, таких как рисование (через paintEvent), и события мыши для интерактивного управления вращением фигуры.
2. Обработка событий:

paintEvent используется для отрисовки октаэдра. Происходит масштабирование, применение матрицы вращения (rotateMatrix) и отрисовка граней с учётом перспективы.

3. События мыши (mousePressEvent, mouseMoveEvent, mouseReleaseEvent) обрабатываются для реализации интерактивного вращения октаэдра в пространстве.

4. Класс Octahedron:

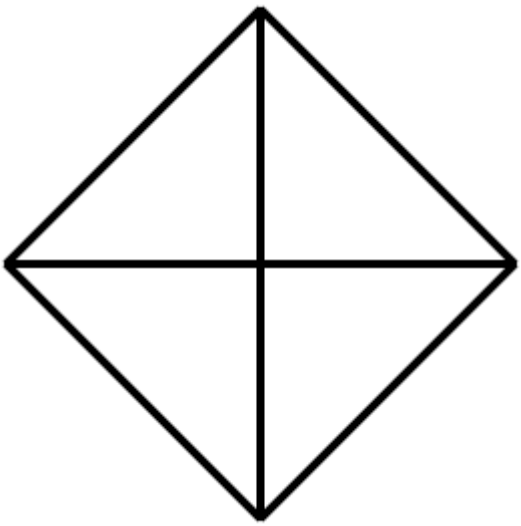
Описывает геометрию правильного октаэдра, включая вершины и грани. Использует QVector для хранения координат вершин (vertexes) и информации о гранях (planes).

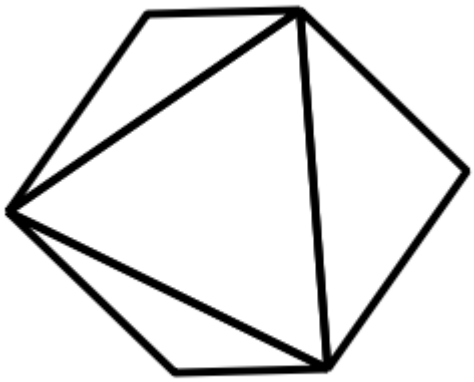
5. Рисование фигуры:
    - Используется QPainter для отрисовки граней октаэдра.
    - Применяются математические преобразования для корректного отображения фигуры в трёхмерном пространстве.
    - Отдельно обрабатываются передние и задние грани для корректного визуального представления глубины.
6. Интерактивное управление:

Пользователь может вращать фигуру мышью, что достигается путём обновления матрицы вращения в ответ на движения мыши.

Эта программа демонстрирует применение объектно-ориентированного подхода в разработке графических приложений и эффективное использование возможностей Qt для создания интерактивных 3D-визуализаций.

Примеры работы программы:





2 Вывод

В данной лабораторной работе была успешно выполнена задача визуализации правильного октаэдра с использованием Qt. Основная цель проекта заключалась в создании точной трехмерной модели октаэдра и её интерактивной визуализации.

Проект эффективно демонстрирует структурированное представление октаэдра, включающее вершины и грани, что значительно облегчает понимание его геометрической структуры. Визуализация осуществляется через детализированный рисунок граней октаэдра, что позволяет точно отобразить его форму и пространственное расположение. Особое внимание уделено интерактивности: пользователь может вращать модель мышью, что обеспечивает более глубокое погружение в изучение фигуры и её свойств.

Использование Qt как инструмента для реализации проекта подчеркивает мощь этой среды в создании комплексных графических интерфейсов и визуализаций. Поддержка математических операций над матрицами и векторами в Qt значительно упрощает выполнение требуемых геометрических преобразований и визуальное представление трехмерных объектов.

Лабораторная работа №3

Тема: Основы построения фотореалистичных изображений.

Задача: Используя результаты Л.Р.No2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света.

Вариант №2: Прямой эллиптический цилиндр

1 Решение

Решение задачи аппроксимации прямого эллиптического цилиндра выпуклым многогранником на Qt было реализовано следующим образом:

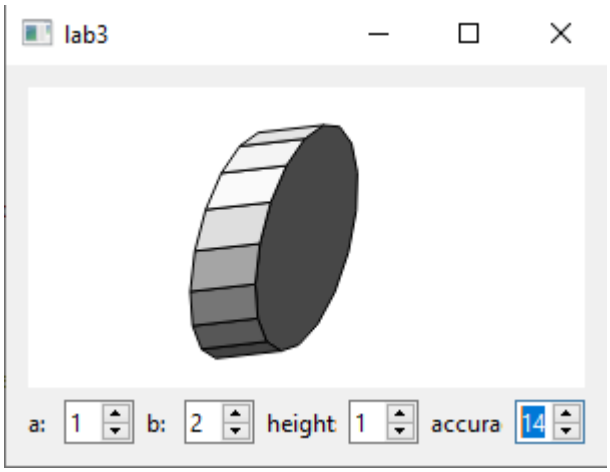
1. Классы и структура программы:
    - Figure: Класс, отвечающий за генерацию вершин цилиндра. Вершины делятся на три группы: базовые (нижние), боковые и верхние.
    - MainWindow: Главный класс интерфейса, включает в себя элементы управления для изменения параметров фигуры (радиусов, высоты, точности) и обработку событий мыши для вращения и масштабирования фигуры.
    - Surface: Виджет, который отвечает за отрисовку фигуры. Использует QPainter для визуализации многогранника с учетом удаления невидимых линий и поверхностей.
2. Реализация аппроксимации:
    - Вершины цилиндра генерируются с учетом заданных параметров a и b (радиусы эллипса, лежащего в основаниях цилиндра) и высоты. Количество вершин на основаниях и боковых гранях определяется параметром точности.
    - Для боковых граней создаются пары вершин: каждая пара соединяет соответствующие верхнюю и нижнюю точки.
3. Взаимодействие с пользователем:

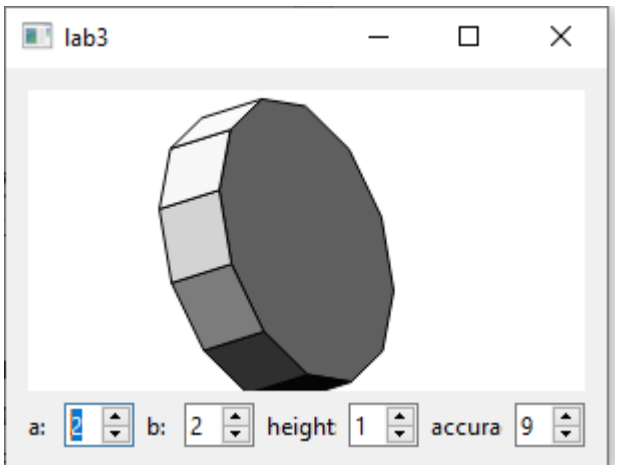
- Пользователь может изменять параметры a, b, высоту и точность аппроксимации с помощью элементов управления в MainWindow.
- Вращение фигуры осуществляется путем перетаскивания мыши, что изменяет матрицу вращения rotateMatrix.
- Обновление фигуры и её перерисовка происходят при изменении параметров или вращении.

4. Отрисовка многогранника:
   - Используется QPainter для создания путей (QPainterPath), представляющих грани многогранника.
   - Для каждой грани определяется, является ли она видимой из текущего ракурса, и применяется соответствующая обработка для удаления невидимых линий.
   - Применяется простая модель закраски, учитывающая один источник света, для создания эффекта объемности фигуры. Решение о том в какой цвет цвет закрасить ту или иную грань принимается от коэффициента k, который проверяет как соотносится грань к источнику света и в зависимости от этого делает либо более светлой, либо более тусклой закраску.

Эта реализация демонстрирует комплексный подход к созданию трехмерных объектов в программном обеспечении, включая геометрическую генерацию, интерактивное управление и визуализацию с применением основ компьютерной графики и программирования.

Примеры работы программы:





2 Вывод

В ходе данной лабораторной работы была успешно реализована визуализация и аппроксимация прямого эллиптического цилиндра с использованием технологии Qt. Основным достижением является создание выпуклого многогранника, который аппроксимирует заданную фигуру с задаваемой пользователем степенью точности. Это достигнуто путем эффективного генерирования вершин и граней на основе входных параметров, таких как размеры и точность.

Эта работа не только демонстрирует глубокое понимание принципов компьютерной графики и программирования, но и отражает практическую значимость и применимость этих знаний для решения конкретных задач в области визуализации данных. Результаты проекта могут быть использованы как основа для дальнейших исследований и разработок в сфере 3D-графики и программирования интерактивных приложений.

Лабораторная работа №4-5

Тема: Ознакомление с технологией OpenGL.

Задача: Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.No3, изобразить заданное тело (то же, что и в л.р. No3) с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

Вариант №2: Прямой эллиптический цилиндр

1 Решение

Создание графического приложения с использованием OpenGL для визуализации прямого эллиптического цилиндра включало несколько ключевых этапов:

1. Структурирование данных:
    - Векторная структура vec3 использовалась для представления трехмерных точек и векторов.
    - Функции для операций над векторами (+, -, *) были определены для упрощения геометрических расчётов.
2. Построение модели цилиндра:
    - Функция build_figure генерирует вершины для каждой грани цилиндра исходя из параметров a, b (радиусы эллипса) и шага da, которые пользователь может изменять в диалоговом режиме.
    - Вершины для верхнего и нижнего оснований цилиндра также создаются в этой функции.
3. Отрисовка фигуры:

Функция draw_figure используется для отрисовки полигонов цилиндра. Она также обрабатывает нормали к граням для корректной работы освещения.

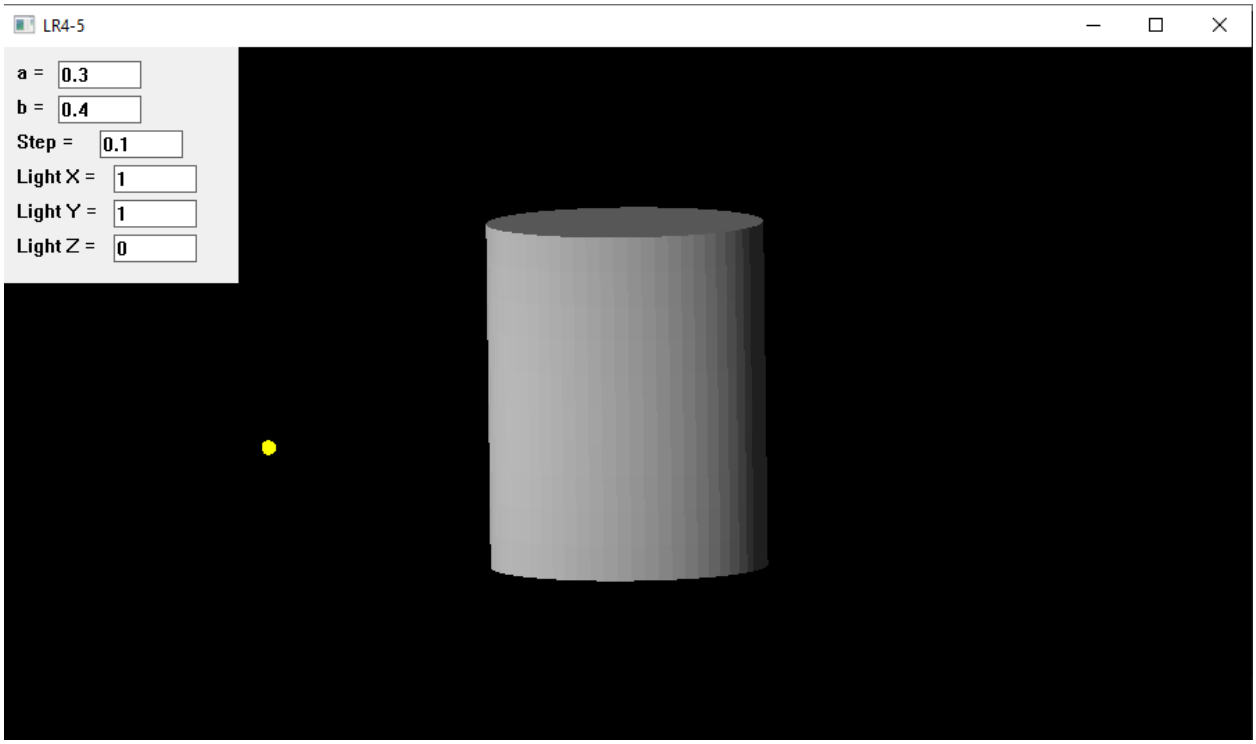
4. Взаимодействие с пользователем:
    - Ввод данных осуществляется через дополнительные окна, создаваемые с помощью WinAPI.

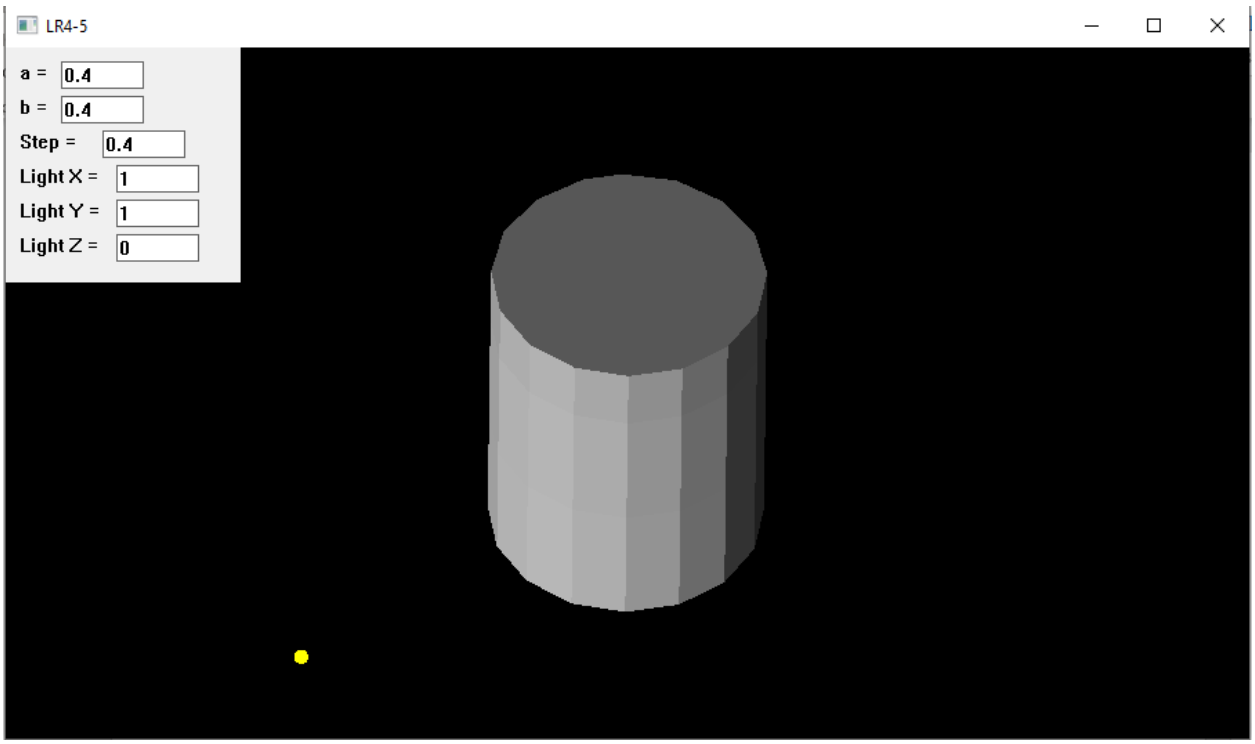
- Обработка событий мыши реализована для вращения фигуры в пространстве.

5. Интеграция освещения:
    - Реализована простая модель освещения на основе GLSL. Позиция источника света задаётся пользователем.
    - Применяется освещение с учетом нормалей к граням для создания реалистичного эффекта освещённости.
6. Обработка визуализации:
    - Установка матриц проекции и вида для корректного отображения объекта в пространстве.
    - Вращение фигуры и масштабирование обеспечиваются через преобразования матриц.

Этот проект демонстрирует глубокое понимание работы с OpenGL и визуализацией трехмерных объектов, а также интеграцию пользовательского интерфейса на основе WinAPI для более гибкого управления параметрами модели и освещения.

Примеры работы программы:





2 Вывод

В рамках данной работы была успешно реализована задача создания графического приложения с использованием OpenGL для визуализации прямого эллиптического цилиндра. Проект продемонстрировал эффективное применение OpenGL для построения и отображения трехмерных объектов, а также использование GLSL для реализации простой модели освещения.

Основные достижения работы включают:

1. **Точная аппроксимация цилиндра**: Цилиндр был аппроксимирован с возможностью динамического управления степенью точности, что позволяет детально изучить его геометрию.

2. **Интерактивное взаимодействие**: Реализованы функции вращения и масштабирования многогранника, обеспечивающие интерактивность и улучшенное пользовательское взаимодействие.

3. **Удаление невидимых линий и поверхностей**: Применение алгоритмов для удаления невидимых частей повышает качество и реалистичность визуализации.

4. **Пользовательский интерфейс**: Интегрированный интерфейс с использованием WinAPI позволяет динамически менять параметры освещения и отражающие свойства материала, добавляя глубину и гибкость в управление визуализацией.

Эта работа является отличным примером того, как OpenGL может быть использован для создания сложных и интерактивных 3D приложений, демонстрируя возможности и гибкость этой технологии в области компьютерной графики. Результаты этого проекта могут быть использованы как основа для дальнейших исследований и разработок в сфере 3D-моделирования и визуализации.

Лабораторная работа №6

Тема: Создание шейдерных анимационных эффектов в OpenGL 2.1

Задача: Для поверхности, созданной в л.р. No5, обеспечить выполнение следующего шейдерного эффекта:

Вариант №2: Прямой эллиптический цилиндр

1 Решение

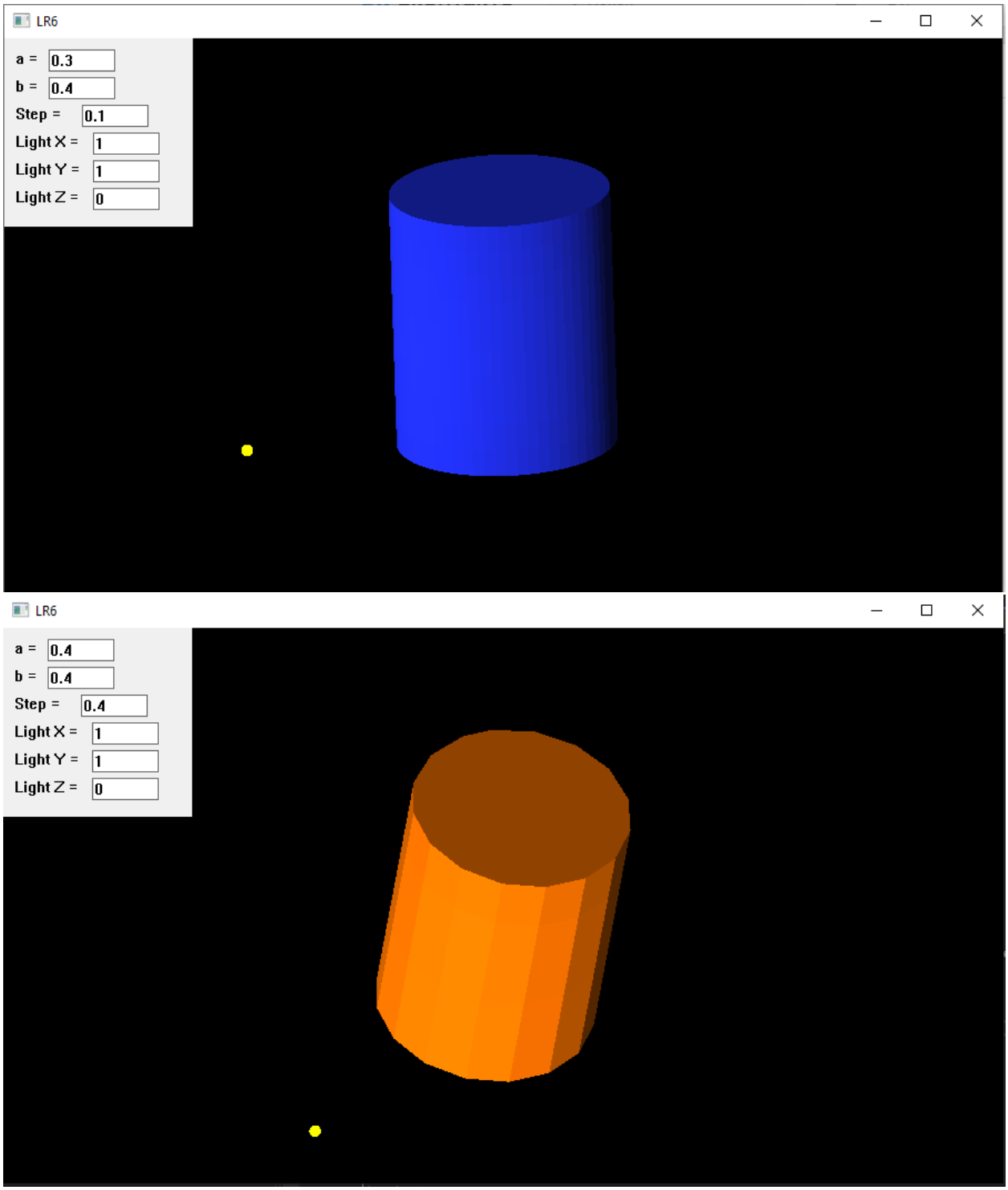
Для создания шейдерного анимационного эффекта в OpenGL 2.1 на прямом эллиптическом цилиндре были внесены следующие дополнения и изменения:

1. Временная зависимость цвета:
    - Использование времени с момента запуска программы для изменения цвета каждой грани цилиндра. Это достигается за счет вычисления time с использованием стандартной библиотеки chrono.
    - Цвет каждой грани цилиндра изменяется во времени с использованием функции синуса для каждого канала цвета (красного, зеленого и синего). Фазы для зеленого и синего цветов смещены, что создает динамический эффект изменения цвета.
2. Расчет цвета:
    - Для каждого канала цвета (red, green, blue) используются синусоидальные функции с разными фазами. Это создает плавное изменение цветов в диапазоне от 0 до 1.
    - Изменение цвета зависит от времени, что приводит к анимационному эффекту.
3. Отрисовка фигуры:
    - В цикле перебираются все грани фигуры. Для каждой грани вычисляется нормаль для правильного отображения освещения.
    - Цвет каждой грани задается динамически перед началом ее отрисовки с помощью glColor3d.
    - Грани отрисовываются с использованием вызовов glBegin(GL_POLYGON) и glEnd().

Это изменение в функции draw_figure позволяет создать визуально привлекательный анимационный эффект, где цвета поверхности цилиндра плавно меняются во времени, создавая иллюзию движения цвета по

поверхности фигуры. Этот эффект демонстрирует, как можно использовать временные параметры и математические функции для создания динамических визуальных эффектов в OpenGL.

Примеры работы программы:



2 Вывод

В ходе данной лабораторной работы был успешно реализован шейдерный анимационный эффект для прямого эллиптического цилиндра в среде OpenGL 2.1. Основной акцент был сделан на создании динамического изменения цвета поверхности цилиндра в зависимости от времени, используя синусоидальные функции для расчета цветовых компонентов.

Основные достижения работы:

1. **Динамическое изменение цвета**: Реализация плавного изменения цвета поверхности цилиндра, основанного на времени, демонстрирует глубокое понимание работы с шейдерами в OpenGL.
2. **Использование времени в шейдерах**: Интеграция времени с момента запуска программы для создания анимационных эффектов показывает, как можно манипулировать графическими эффектами в реальном времени.
3. **Визуальный эффект**: Созданный анимационный эффект обеспечивает визуально привлекательное и динамическое отображение, улучшая эстетическую привлекательность графического приложения.

Эта работа подчеркивает, как творческие и технические аспекты программирования могут сочетаться для создания впечатляющих визуальных эффектов. Реализация такого рода анимационных эффектов в OpenGL открывает двери для более сложных и интерактивных визуализаций в будущих проектах, включая игры и симуляции.

Лабораторная работа №7

Тема: Построение плоских полиномиальных кривых.

Задача: Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

Вариант №2: Сегмент кубического сплайна по конечным точкам и касательным

1 Решение

В данной лабораторной работе использовалась библиотека GLFW для создания и визуализации кубического сплайна. Вот основные элементы программы:

1. Структуры данных:
    - vec3: Структура для представления трехмерной точки или вектора.
    - SplinePoint: Содержит позицию и касательный вектор для точки сплайна.
    - CubicSpline: Описывает кубический сплайн через две точки SplinePoint.
2. Функции для работы с векторами:

Определены операторы для основных операций над векторами, таких как сложение, вычитание и векторное произведение.

3. Вычисление точек сплайна:

CalculateSplinePoint: Вычисляет точку на сплайне в зависимости от параметра t (в диапазоне от 0 до 1).

4. Отрисовка сплайна:

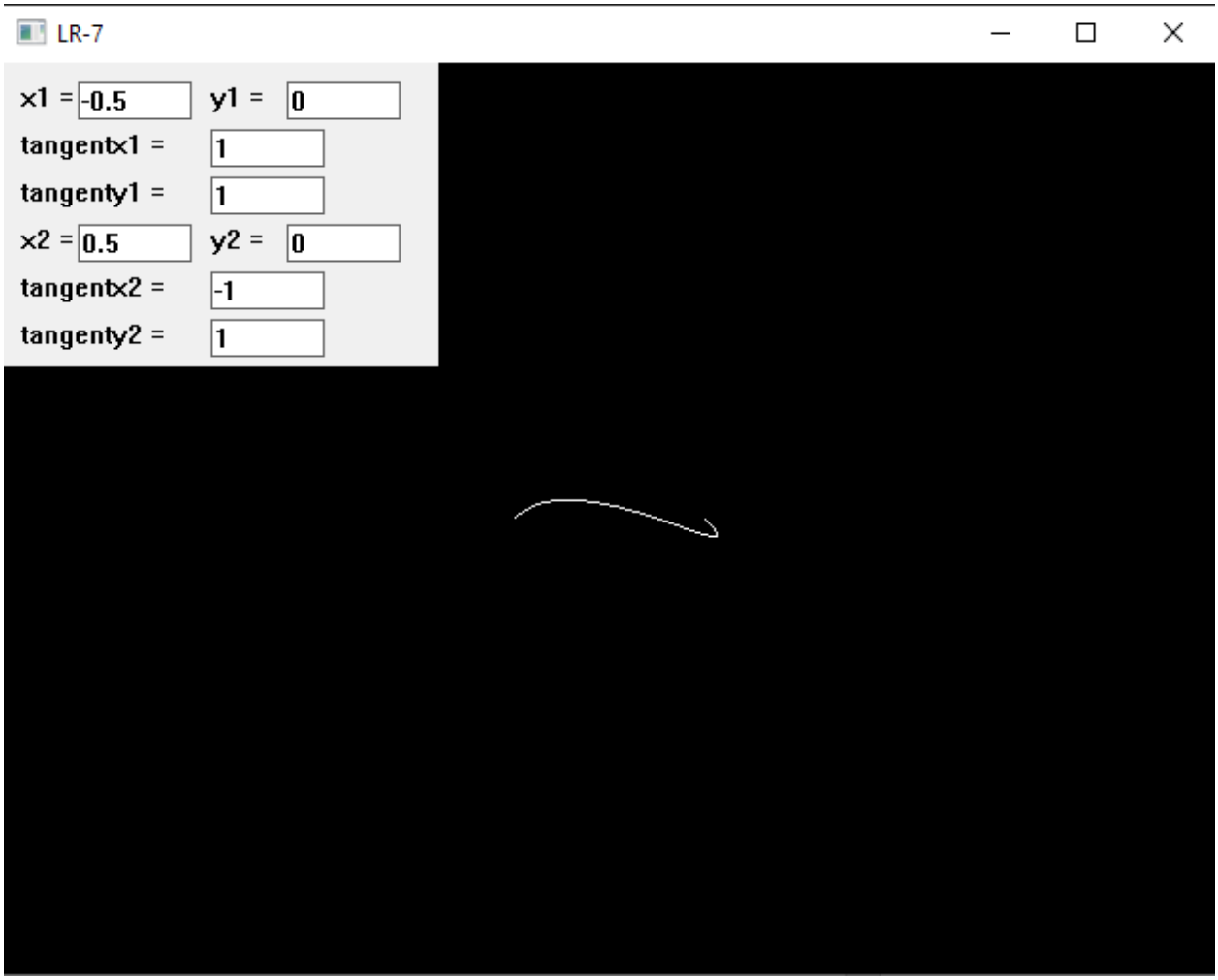
DrawSpline: Отрисовывает сплайн, вычисляя его точки в цикле и соединяя их линиями.

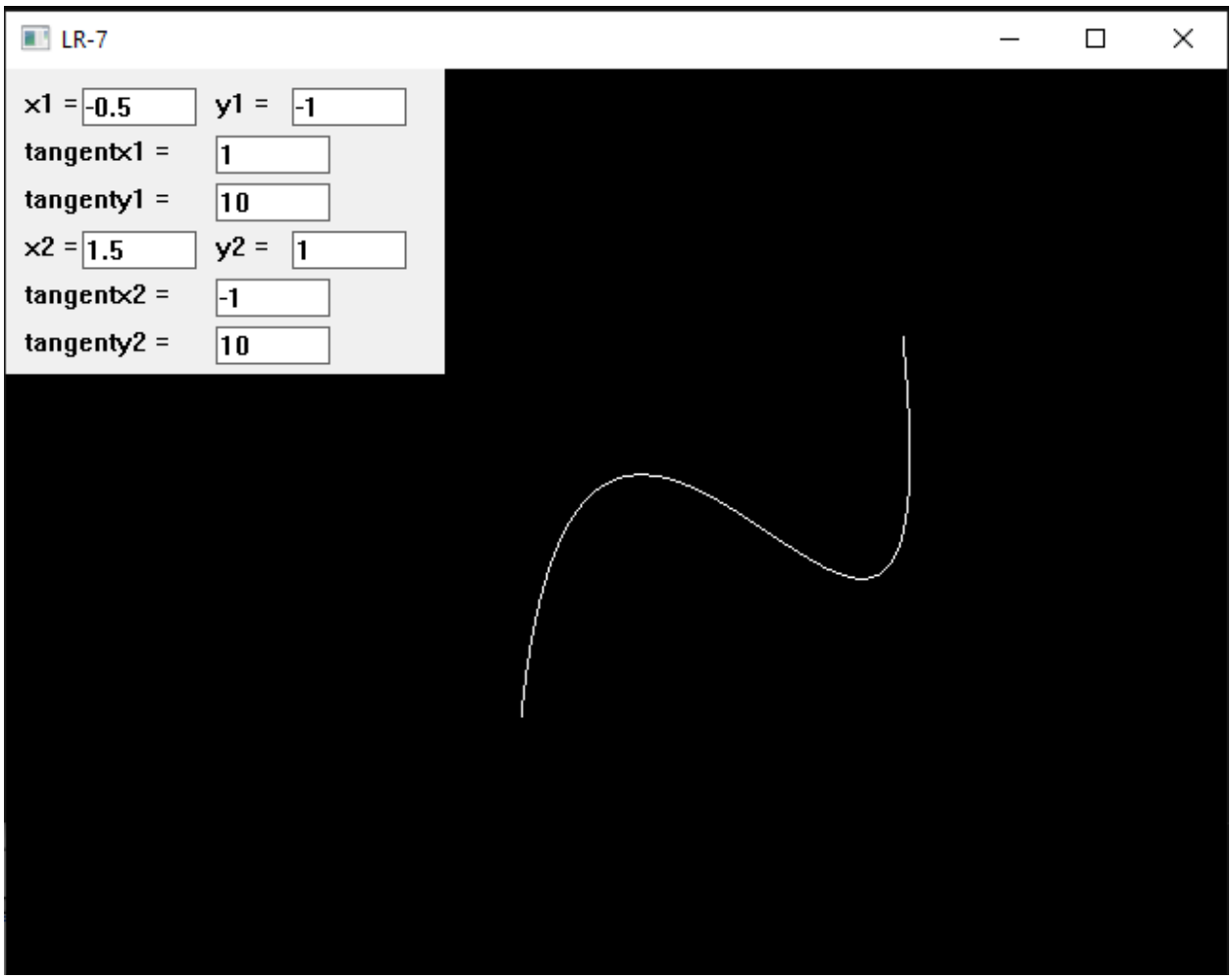
5. Взаимодействие с пользователем:
    - Используется обработка событий мыши для изменения положения сплайна.
    - Параметры сплайна и касательных задаются в диалоговых окнах, создаваемых через WinAPI.
6. Главный цикл приложения:

В цикле обновляются параметры сплайна в зависимости от ввода пользователя и происходит его отрисовка.

Эта программа демонстрирует, как можно создать интерактивное графическое приложение для визуализации и модификации кубического сплайна. Она показывает применение математических принципов для генерации полиномиальных кривых и интеграцию с пользовательским интерфейсом для динамического управления этими кривыми.

Примеры работы программы:





2 Вывод

В ходе данной лабораторной работы была успешно разработана программа для построения и визуализации кубического сплайна, основанного на конечных точках и касательных векторах. Программа позволяет динамически изменять позиции точек и касательные векторы, обеспечивая тем самым гибкое управление формой кривой.

Эта лабораторная работа подчеркивает важность сочетания математических методов и компьютерной графики для создания сложных и интерактивных визуализаций. Полученные навыки и знания могут быть применены в различных областях, включая разработку игр, анимацию и графическое моделирование.