**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Компьютерная графика" №1**

Студент: Кострюков Е. С.

Группа: М8О-307Б-22

Москва, 2024

# **Основы 2D-графики и трансформаций**

Цель лабораторной работы:

В данной лабораторной работе вам предстоит научиться работать с графическим API для отрисовки 2D-примитивов, освоить основные 2D-трансформации (перемещение, масштабирование, поворот) и изучить алгоритмы построения 2D-кривых.

Вариант 5. Построение кубической кривой Безье

Постройте кубическую кривую Безье, используя четыре контрольные точки.

Контрольные точки должны быть отображены на экране и иметь возможность перемещения.

Кривая должна обновляться в реальном времени при изменении положения любой контрольной точки.

Дополнительно: Реализуйте анимацию, где кривая плавно изменяет свою форму в зависимости от времени.

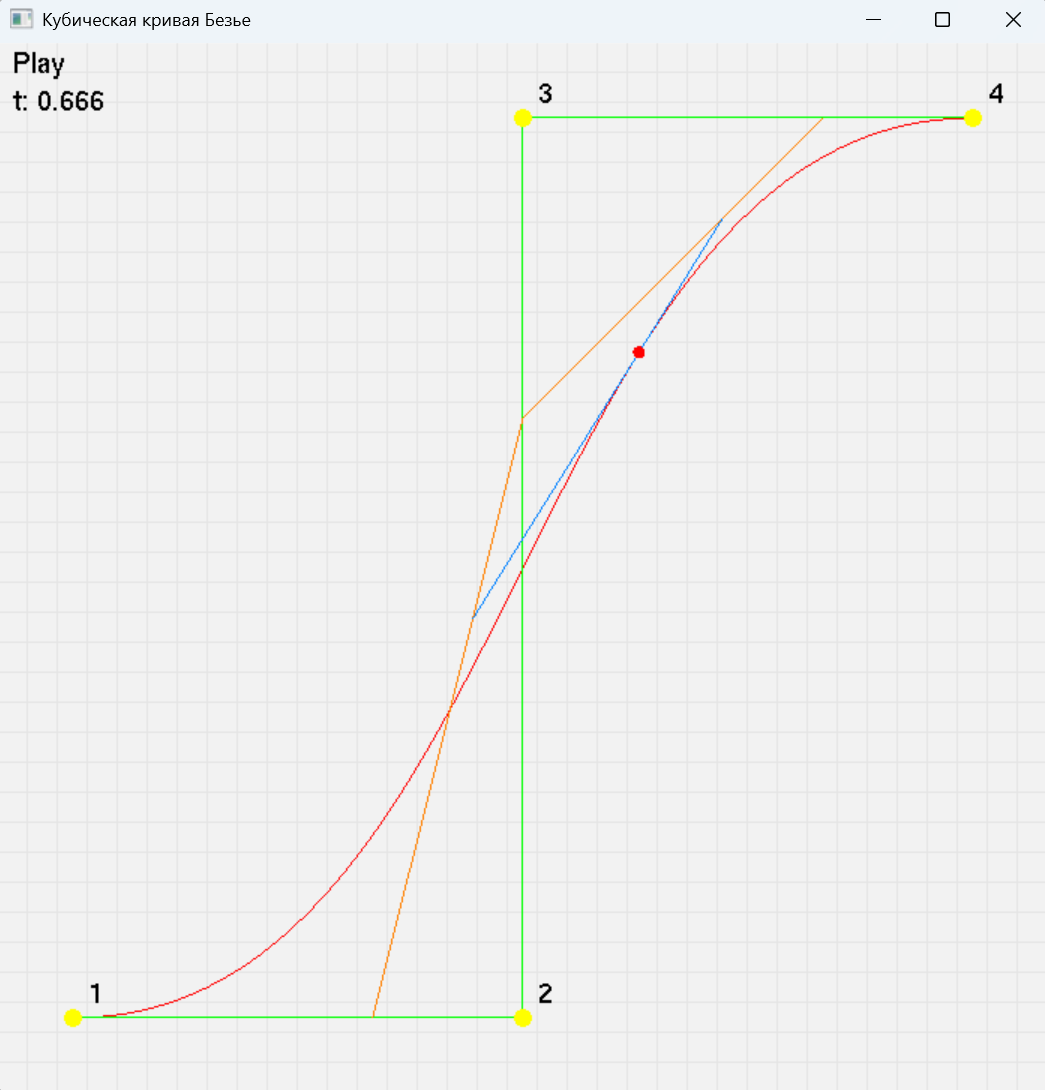
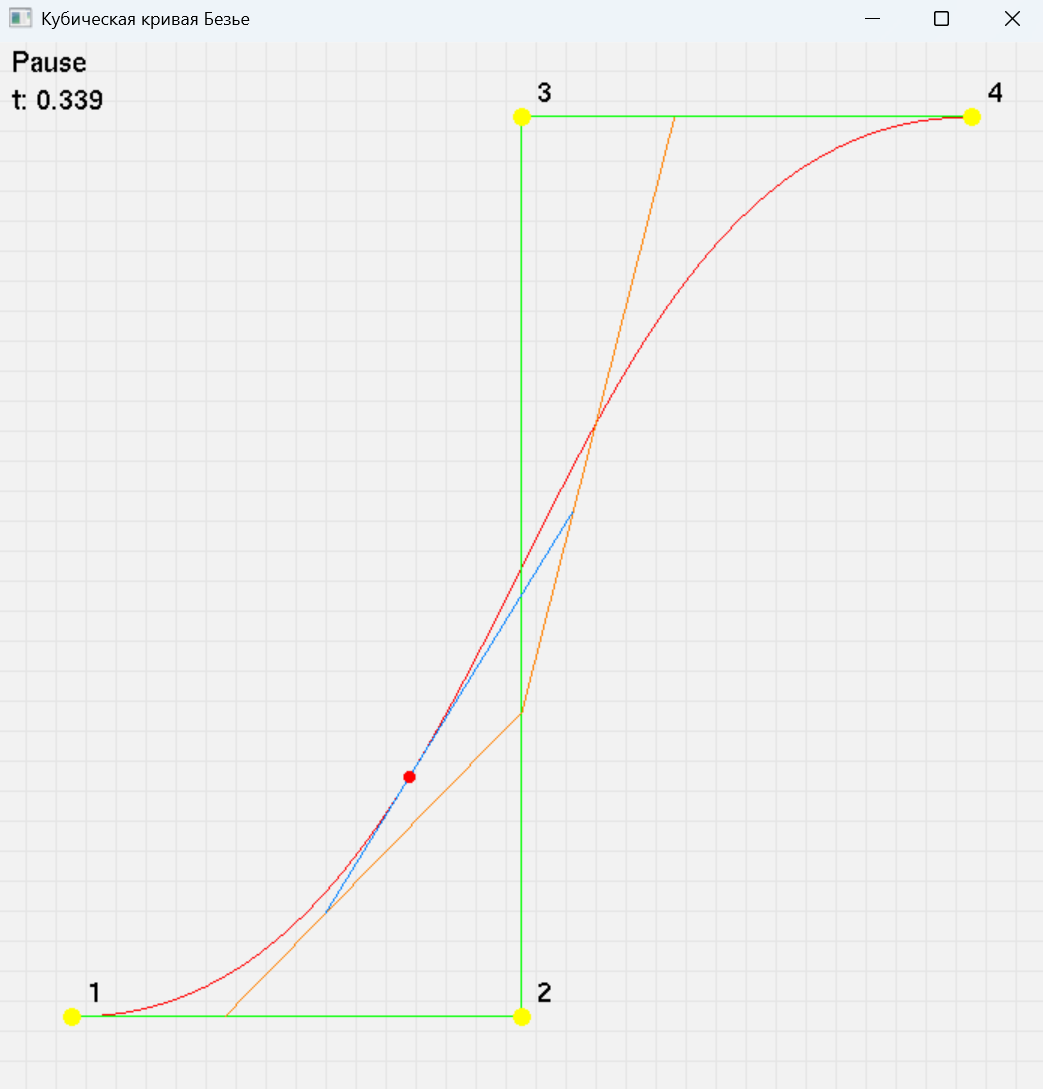
# **Решение**

Программа реализует визуализацию кубической кривой Безье с возможностью интерактивного изменения контрольных точек и анимации кривой. Основная логика программы заключается в вычислении промежуточных точек и отображении кривой через линейную интерполяцию между ними.

Функция calculateIntermediatePoints вычисляет новые промежуточные точки между двумя соседними точками, используя линейную интерполяцию. Эти точки используются для построения кривой на каждом уровне. Функция drawHelperLines рисует вспомогательные линии, соединяющие все контрольные точки и промежуточные точки, а также рекурсивно строит линии на разных уровнях интерполяции, визуализируя процесс построения кривой.

Функция drawBezierCurve отрисовывает саму кривую Безье, вычисляя новые промежуточные точки для различных значений параметра t, который изменяется в ходе анимации. Для отображения интерфейса используется функция drawUI, которая показывает текущее состояние анимации (воспроизводится или на паузе) и значение параметра t.

Обработчики событий мыши и клавиш позволяют пользователю взаимодействовать с программой: перетаскивать контрольные точки и запускать или приостанавливать анимацию кривой. Таймер, реализованный в функции timer, обновляет значение t и перерисовывает сцену, создавая эффект анимации. Инициализация OpenGL происходит в функции init, которая настраивает проектирование и начальные параметры для рендеринга.



# **Вывод**

Лабораторная работа позволяет продемонстрировать знания работы с OpenGL и базовой графикой. Расчеты трансформаций выполняются вручную, что соответствует требованиям. Программа подтверждает понимание линейной алгебры и ее применения в графике, а также базовых принципов визуализации. Возможность управления через клавиатуру делает приложение интерактивным, улучшая пользовательский опыт.

**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Компьютерная графика" №2**

Студент: Кострюков Е. С.

Группа: М8О-307Б-22

Москва, 2024

# **Основы 3D-графики и проекция**

Цель лабораторной работы:

В этой лабораторной работе вы познакомитесь с основами 3D-графики: построением простых 3D-объектов, проекцией на 2D-плоскость, а также научитесь работать с матрицами перспективы и ортографической проекции.

Вариант 5. Построение сферического объекта с возможностью масштабирования

Постройте сферу в 3D-пространстве.

Реализуйте возможность масштабирования сферы (увеличение и уменьшение радиуса).

Примените перспективную проекцию для корректного отображения.

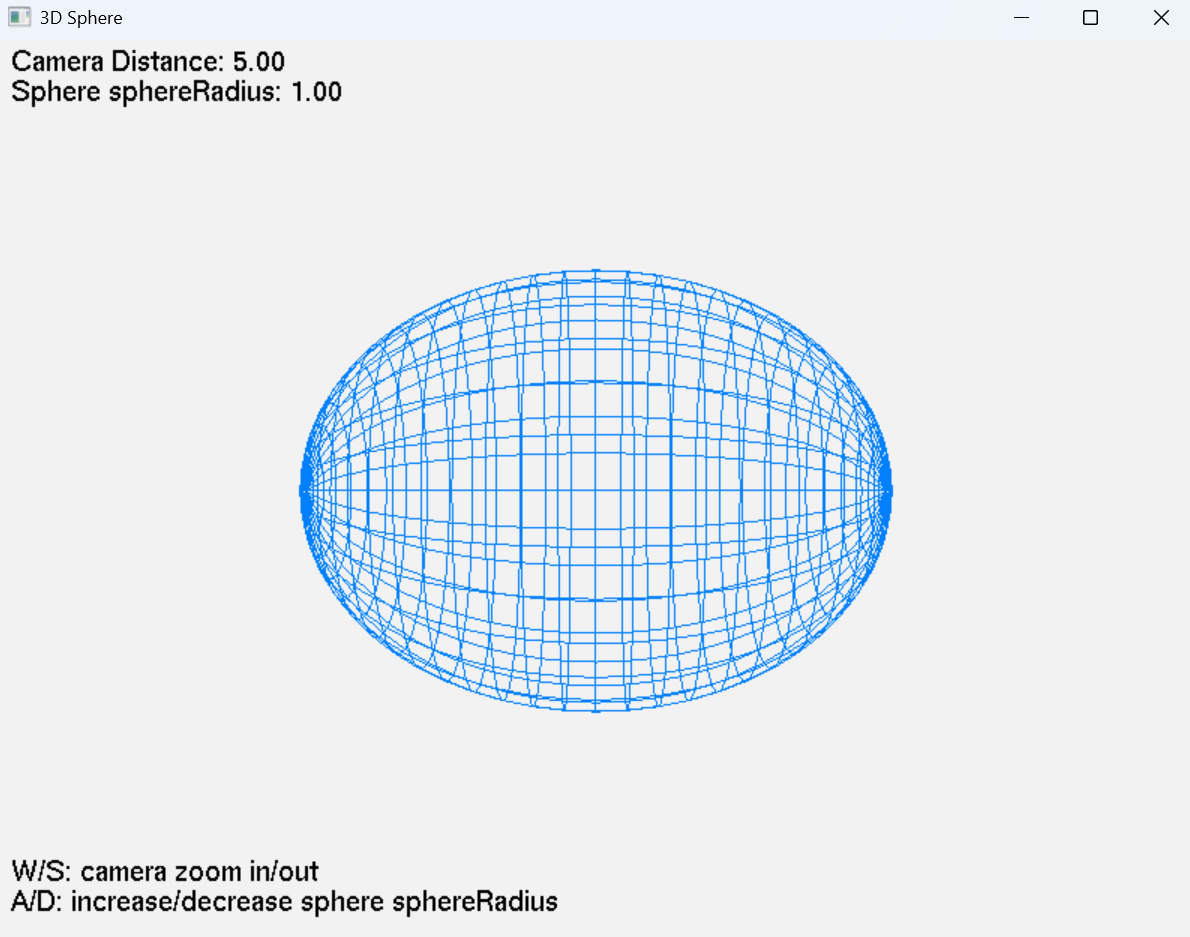
Дополнительно: Добавьте возможность перемещения камеры вокруг сферы с изменением расстояния до неё.

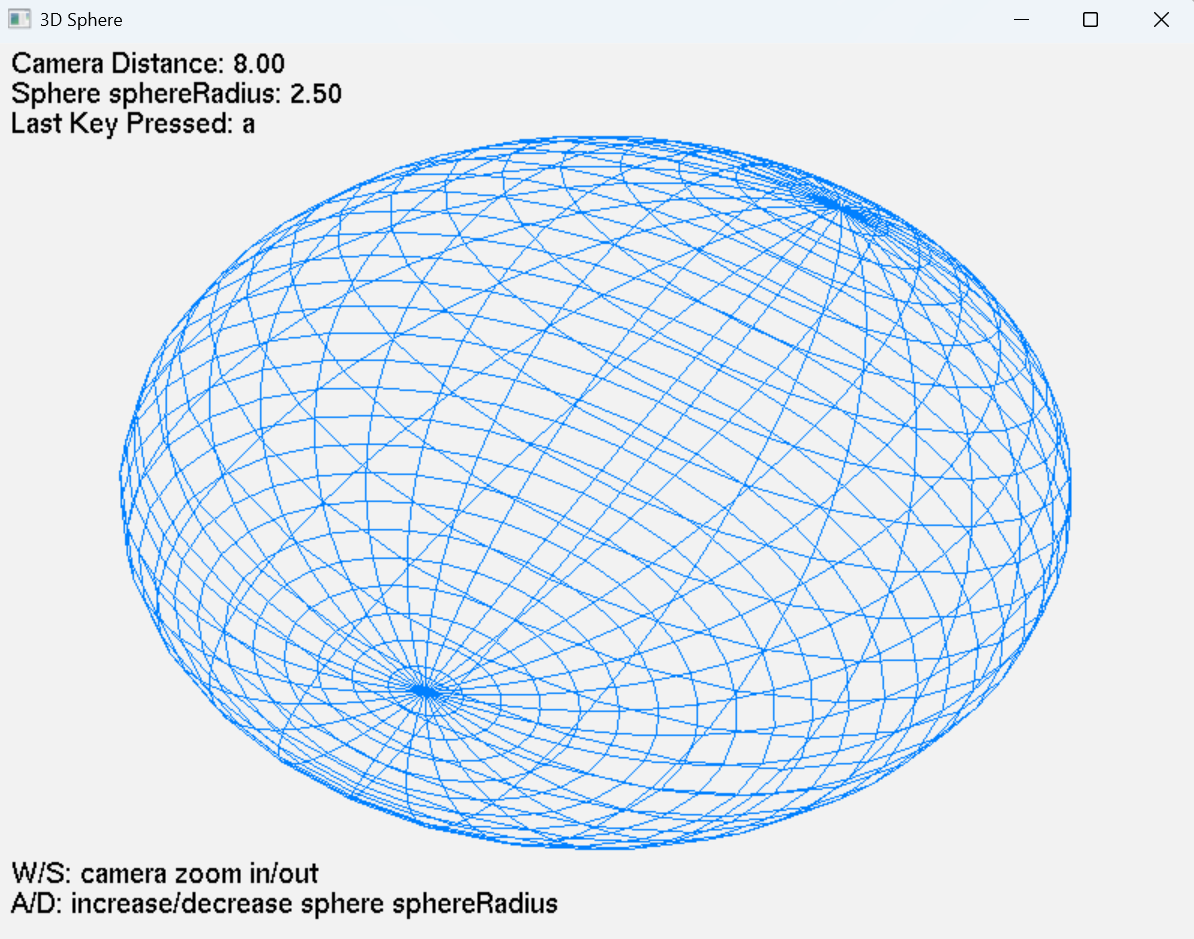
# **Решение**

Программа строит 3D-сферу, используя библиотеку OpenGL, и отображает её в перспективной проекции. В функции display осуществляется отрисовка сферы с применением камеры, которая может изменять своё положение с помощью клавиш и движения мыши. Функция updateCamera обновляет параметры камеры, используя gluLookAt для задания её позиции и направления взгляда, а сама сфера отрисовывается с помощью функции drawSphere, которая масштабирует объект в зависимости от заданного радиуса.

Управление осуществляется через клавиатуру и мышь: клавиши W и S отвечают за увеличение и уменьшение расстояния камеры, а клавиши A и D изменяют радиус сферы. Движение мыши используется для изменения углов обзора камеры, что позволяет свободно вращать сцену. В процессе работы отображаются текстовые подсказки и текущие значения параметров, таких как расстояние до камеры и радиус сферы.

Главный цикл программы запускается с помощью glutMainLoop, который обеспечивает постоянное обновление сцены. Инициализация OpenGL происходит в функции initOpenGL, где настраиваются параметры проекции и активируется тест глубины для корректного отображения 3D объектов.





# **Вывод**

Лабораторная работа позволила изучить основы работы с 3D-графикой в OpenGL, включая управление камерой и отрисовку 3D объектов. Полученные навыки демонстрируют, как математические методы используются для изменения углов обзора и расстояния до объектов, а также как взаимодействие пользователя с клавишами и мышью влияет на отображение сцены. Программа подтверждает понимание принципов перспективной проекции и динамического обновления параметров камеры для создания интерактивных 3D-изображений.

**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Компьютерная графика" №3**

Студент: Кострюков Е. С.

Группа: М8О-307Б-22

Москва, 2024

# **Камера и базовые 3D-трансформации**

Цель лабораторной работы:

В этой лабораторной работе вы научитесь работать с камерой в 3D-пространстве, управлять её положением и направлением, а также освоите базовые трансформации (перемещение, поворот и масштабирование) объектов в 3D.

Вариант 5. Масштабирование объекта с учетом позиции камеры

Постройте пирамиду в 3D-пространстве.

Реализуйте возможность масштабирования пирамиды в реальном времени.

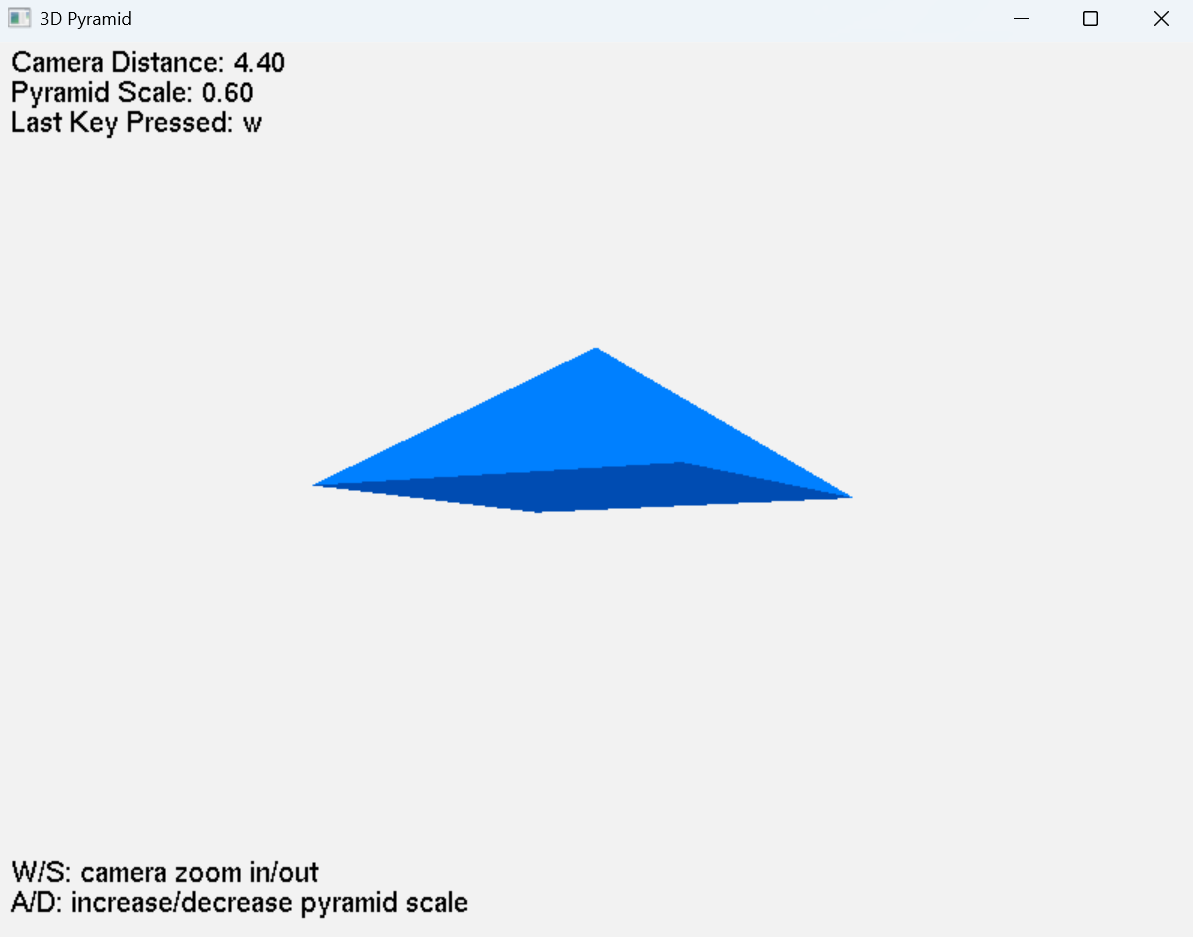
При изменении масштаба пирамиды камера должна оставаться неподвижной, но сцена должна корректно отрисовываться.

Дополнительно: Реализуйте ограничение минимального и максимального масштаба объекта.

# **Решение**

# Программа создаёт 3D-сцену с пирамидой и реализует управление камерой с помощью клавиатуры и мыши. Для управления масштабом пирамиды используется клавиша A/D, а для изменения расстояния камеры — клавиши W/S. Функция display отвечает за отрисовку сцены, где обновляется камера с использованием gluLookAt и отображается пирамида. Позиция камеры и её углы обновляются в функции updateCamera, которая учитывает текущие параметры камеры. Ввод с клавиатуры обрабатывается в функции keyboard, а движение мыши для изменения углов обзора — в функции motion. Для постоянного обновления сцены используется двойная буферизация с помощью glutSwapBuffers.





# **Вывод**

Выполнение лабораторной работы позволило изучить основы работы с OpenGL для создания и визуализации 3D-объектов, а также взаимодействие с пользователем через клавиатуру и мышь. Реализованное управление камерой и масштабированием объекта демонстрирует, как математические преобразования и взаимодействие с пользователем влияют на отображение 3D-сцены. Работа позволила углубить знания в области обработки ввода, настройки камеры и применения трансформаций для создания интерактивных графических приложений.

**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Компьютерная графика" №4**

Студент: Кострюков Е. С.

Группа: М8О-307Б-22

Москва, 2024

# **Освещение и работа с шейдерами**

Цель лабораторной работы:

В этой лабораторной работе вы научитесь работать с освещением в 3D-пространстве, используя различные типы источников света, и освоите основы написания шейдеров. Вы реализуете освещение объектов в сцене с использованием простейших моделей освещения и настроите эффекты при помощи вершинных и фрагментных шейдеров.

Вариант 5. Плоское затенение (Flat Shading)

Постройте куб в 3D-пространстве.

Реализуйте плоское затенение (Flat Shading), при котором каждый полигон объекта имеет один цвет, основанный на нормали грани.

Используйте только вершинный шейдер для выполнения освещения.

Дополнительно: Реализуйте возможность переключения между плоским и гладким затенением (Gouraud Shading).

# **Решение**

Программа отображает 3D-куб с возможностью переключения между двумя методами затенения: плоским (flat) и гладким (Gouraud). Вот краткое описание работы:

1. Шейдеры:

- Вершинный шейдер вычисляет позицию вершин, их нормали, а также направление света для освещения. Для Gouraud шейдинга вычисляется цвет в вершинах, а для плоского — в фрагментном шейдере.

- Фрагментный шейдер либо применяет плоский шейдинг, рассчитывая цвет на основе нормалей, либо использует цвет, вычисленный в вершинах для Gouraud шейдинга.

2. Куб:

- Куб инициализируется с вершинами и нормалями для каждой стороны. Буферы вершин (VBO) и индексов (EBO) загружаются в OpenGL.

- Вершины и нормали используются для определения геометрии куба и его освещения.

3. Вращение и управление:

- Мышь используется для вращения куба, изменяя углы вращения по осям X и Y.

- Клавиша 'f' переключает между плоским шейдингом и шейдингом Gouraud.

4. Матрицы преобразований:

- Для отрисовки используется модельная матрица (вращение), видовая матрица (перемещение камеры) и проекционная матрица (перспектива).

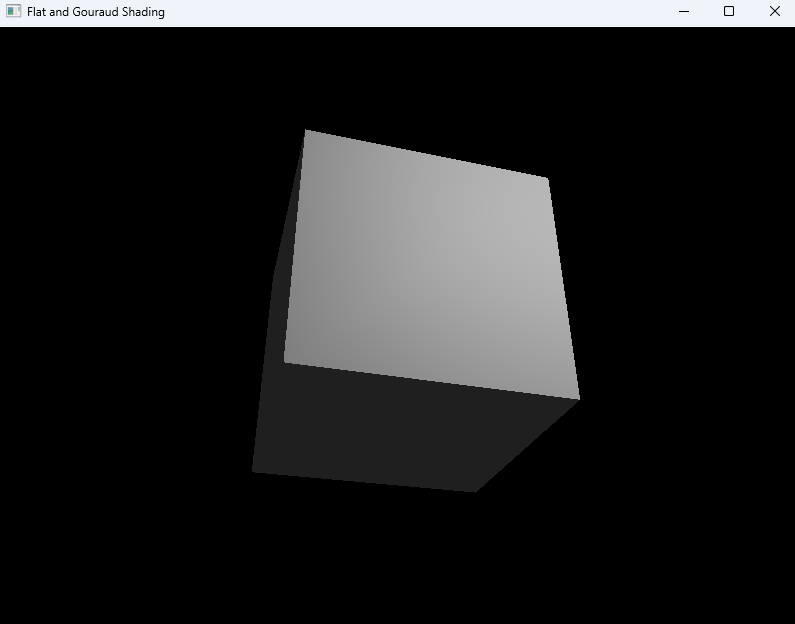
5. Освещение:

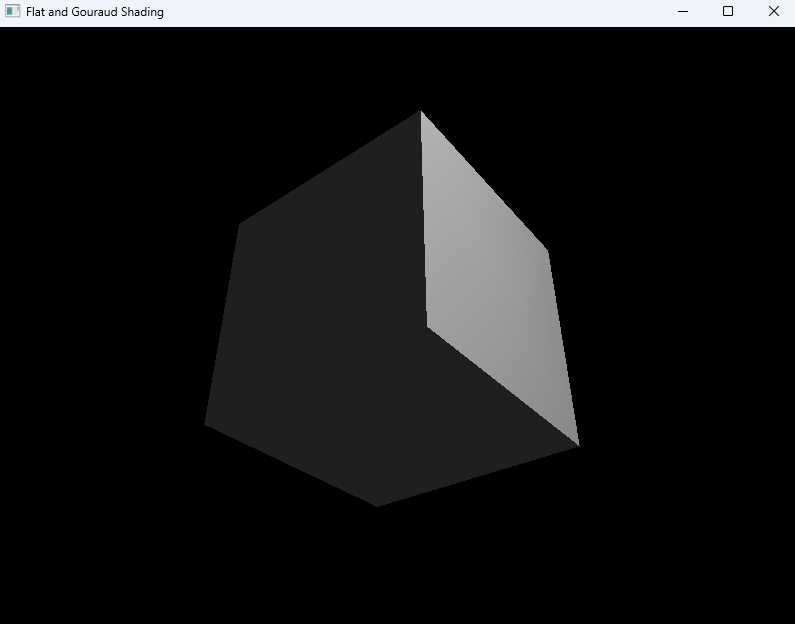
- Программой задано освещение с направленным источником и амбиентным освещением.

6. Основной цикл:

- В главной функции запускается цикл отрисовки, обработка ввода и управление отображением.

В результате программа отображает вращающийся куб, где можно выбирать между плоским и Gouraud шейдингом для разных визуальных эффектов.





# **Вывод**

В ходе выполнения лабораторной работы изучена реализация методов шейдинга в OpenGL, а также основы работы с 3D-объектами и освещением. Реализованы два метода шейдинга: плоский и Gouraud, что позволило продемонстрировать различные подходы к освещению и отображению поверхностей в 3D-графике. Взаимодействие света и объекта на сцене было визуализировано с использованием нормалей и различных матричных преобразований для вращения и отображения куба. Управление вращением куба с помощью мыши обеспечило интерактивность работы. Работа углубила понимание шейдеров, а также показала важность различных методов освещения для создания реалистичных визуальных эффектов в 3D-пространстве.

**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Компьютерная графика" №5**

Студент: Кострюков Е. С.

Группа: М8О-307Б-22

Москва, 2024

# **Трассировка лучей (Ray Tracing)**

Цель лабораторной работы:

В этой лабораторной работе вы научитесь работать с техникой трассировки лучей для создания реалистичной 3D-графики. Вы реализуете алгоритм Ray Tracing, который позволяет рассчитывать физически корректные отражения, преломления, тени и свет в сцене. Лабораторная работа подводит к пониманию основ рендеринга, работающего с лучами света, а также к созданию реалистичных сцен.

Вариант 5. Реалистичное глобальное освещение

Постройте сцену с кубом и сферой.

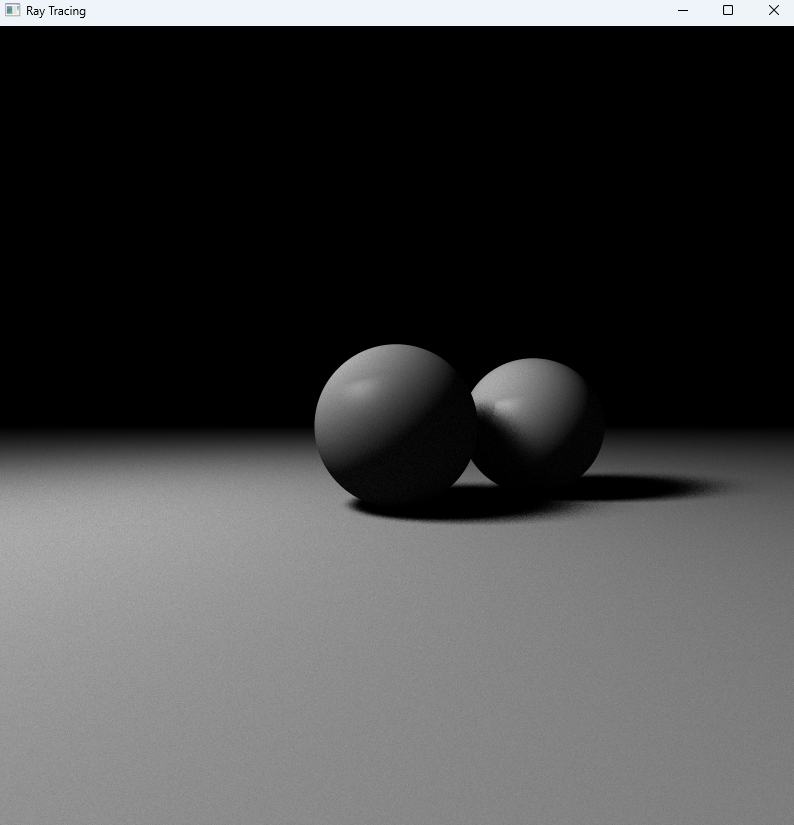
Реализуйте глобальное освещение (Global Illumination) с помощью трассировки лучей.

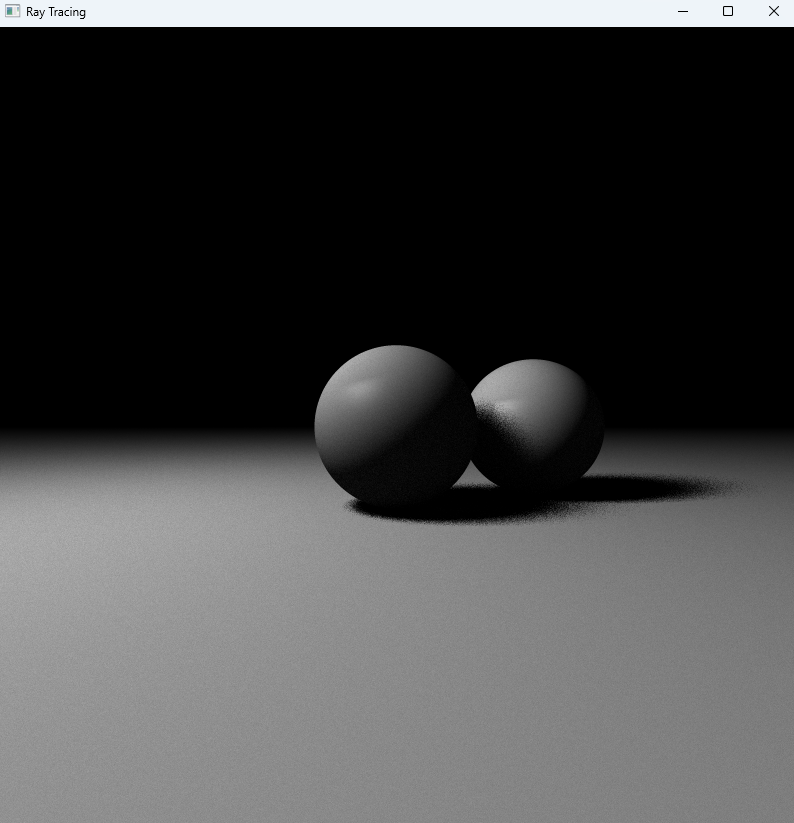
Обеспечьте взаимодействие объектов со светом, отскакивающим от других поверхностей (интерфлексии света).

Дополнительно: Реализуйте управление количеством "прямых" и "косвенных" лучей для точной настройки уровня освещения.

# **Решение**

Программа реализует сцену с использованием алгоритма трассировки лучей, включающую сферу, плоскость и различные эффекты освещения, такие как отражения, косвенное освещение и мягкие тени. Функция trace отвечает за трассировку лучей, определяя пересечения с объектами (сферой или плоскостью), вычисляя цвет пикселя с учетом освещения и отражений. Мягкие тени достигаются с помощью распределённой трассировки: случайные смещения источника света генерируют несколько теневых лучей, а итоговый цвет пикселя зависит от количества затенённых лучей. Функция renderScene строит первичные лучи для каждого пикселя, используя координаты экрана, и отрисовывает пиксели с помощью OpenGL. Параметры сцены, такие как положение объектов и источника света, а также глубина отражений и косвенного освещения задаются в коде.





# **Вывод**

В ходе выполнения лабораторной работы была реализована сцена с использованием алгоритма трассировки лучей, включающая объекты, такие как сферы и плоскости, с учетом эффектов освещения, отражений, косвенного освещения и мягких теней. Были изучены основные методы трассировки лучей, включая определение пересечений лучей с объектами, а также различные методы вычисления освещения, такие как модель Фонга для диффузного и зеркального освещения, а также использование случайных смещений для создания мягких теней. Работа позволила углубить знания в области компьютерной графики, научиться применять алгоритмы рендеринга в 3D-сценах и освоить методы визуализации, учитывающие различные физические эффекты освещения.

**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Компьютерная графика" №6**

Студент: Кострюков Е. С.

Группа: М8О-307Б-22

Москва, 2024

# **Основы 2D-графики и трансформаций**

Цель лабораторной работы:

В этой лабораторной работе вы будете разрабатывать основу для собственного игрового движка. Это завершающая работа, в которой вы объедините все изученные техники работы с 2D и 3D графикой, освещением, шейдерами и трассировкой лучей. В результате выполнения лабораторной работы у вас должен получиться простой движок, который может отрисовывать сцены в реальном времени, с возможностью работы с камерой, объектами и освещением.

Основные задачи:

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).

# **Решение**

Программа представляет собой игру-головоломку, в которой игроку предстоит пройти через лабиринт, чтобы найти выход.

- Файл main отвечает за запуск игры и управление основными процессами.

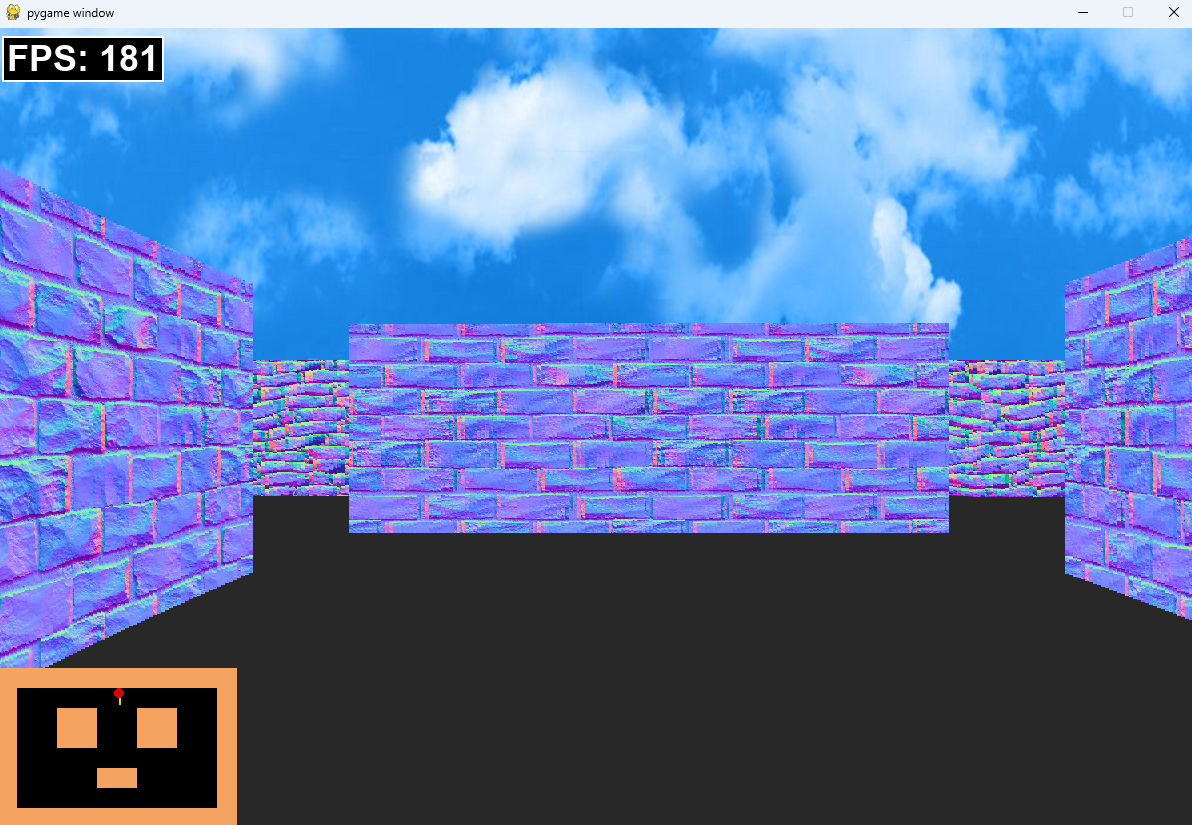
- Файл settings содержит настройки игры: параметры графики, константы, и конфигурации текстур.

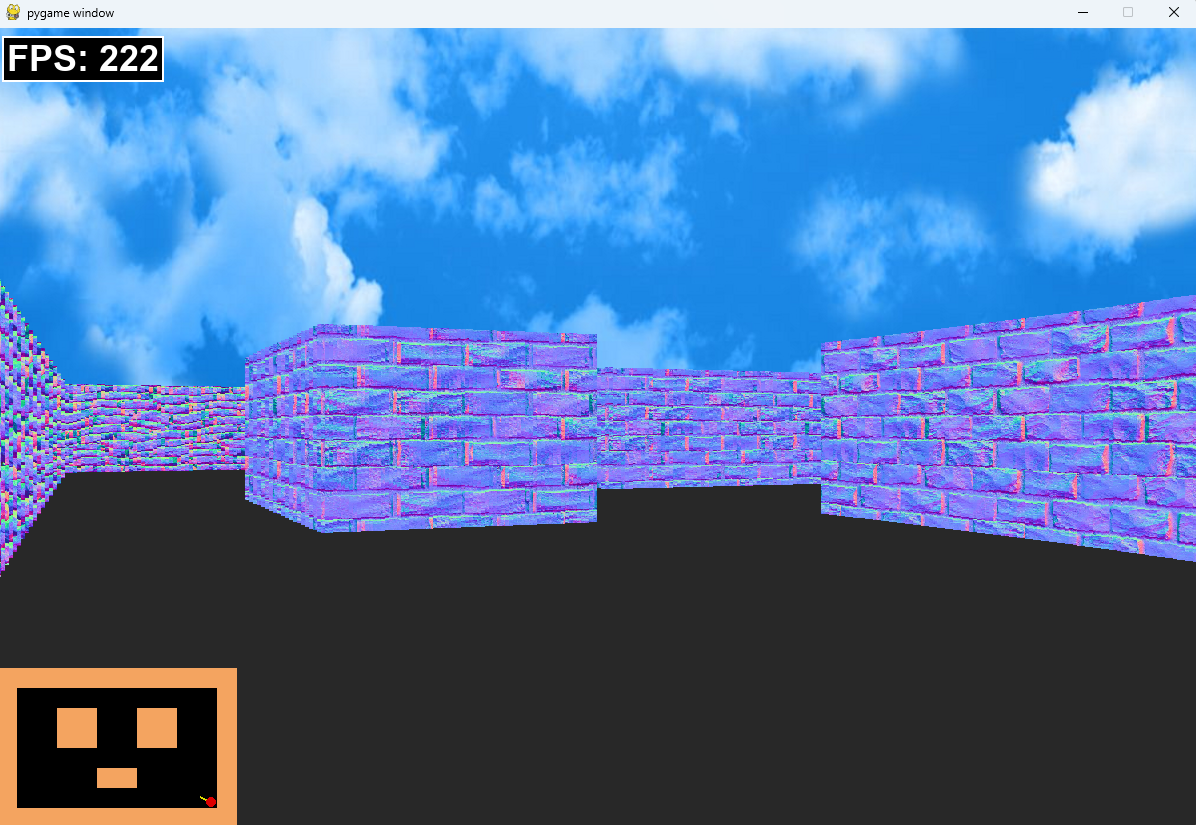
- Файл player реализует функционал игрока, включая управление с клавиатуры и обработку действий.

- Файл map отвечает за структуру лабиринта, хранение карты и связывание её элементов с текстурами.

- Файл drawing занимается визуализацией игры: рисует окружение, интерфейс и фон.

- Файл ray\_casting реализует технологию построения графики, создавая псевдотрёхмерное изображение лабиринта.

Игра сосредоточена на исследовании и поиске пути к выходу, создавая атмосферу, ориентированную на логику и ориентацию в пространстве.



# **Вывод**

В рамках этой лабораторной работы мной был разработан игровой движок и создана игра-головоломка. Используя ранее изученные концепции, я реализовал технологию ray casting для отображения псевдотрёхмерной графики. Дополнительно были добавлены текстуры, а игровая карта была построена в виде лабиринта. Этот опыт позволил мне углубить знания в области разработки игр и подготовки к будущим проектам.