Лабораторная работа №6
Построение и визуализация трёхмерных объектов
Реализация трёхмерных преобразований и простейших проекций

Цель работы: закрепление теоретического материала и практическое освоение основных методов и алгоритмов трёхмерной визуализации, включая построение каркасных моделей, применение аффинных преобразований (масштабирование, перенос, вращение) и построение ортографических проекций на координатные плоскости.

Задачи:

- Разработать приложение на C++ с использованием библиотеки OpenGL для визуализации трёхмерной каркасной модели буквы (первой буквы фамилии студента).
- Реализовать интерактивное управление объектом:
- Масштабирование (+, -)
- Перенос по осям (WASD, Q, E)
- Вращение вокруг осей X, Y, Z (X, Y, Z)
- Вращение вокруг произвольной оси (1, 1, 1)
- Организовать переключение между режимами отображения:
- 3D-вид с перспективной проекцией
- Ортографические проекции на плоскости OXY, OXZ, OYZ
- Одновременный вывод всех четырёх видов
- Реализовать интерактивное управление камерой (перетаскивание мышью для изменения ракурса).
- Отображать текущую матрицу преобразования объекта.

OpenGL (C++)

OpenGL — это кросс-платформенный API для рендеринга 2D- и 3D-графики. В данной работе используются следующие компоненты OpenGL: *GLUT (OpenGL Utility Toolkit)* — библиотека для создания окон, обработки ввода и управления событиями, *GLU (OpenGL Utility Library)* — дополнительные утилиты для работы с камерой, проекциями и примитивами.

Использованный в работе подход демонстрирует классическую архитектуру графического приложения OpenGL: объект описывается геометрией в объектном пространстве, последовательно преобразуется через систему матриц, проецируется на 2D-плоскость и отображается в заданной области окна.

Подход к созданию трехмерной модели

В данной работе используется *каркасное представление (wireframe)* трёхмерного объекта. Этот подход основан на определении модели через

набор вершин и соединяющих их рёбер, что является одним из наиболее фундаментальных способов представления 3D-геометрии.

Преимущества:

- Минимальные вычислительные затраты
- Позволяет видеть внутреннюю структуру объекта
- Идеально для отладки и инженерной визуализации

Структура данных модели:

Вершины хранятся в векторе векторов vertices, где каждая вершина представляет собой тройку координат (x, y, z);

Рёбра определяются как пары индексов в массиве вершин в векторе edges.

Алгоритм отрисовки:

1. Активируется режим рисования линий: glBegin(GL_LINES)
2. Для каждого ребра из списка edges извлекаются координаты соответствующих вершин
3. Последовательно вызывается glVertex3f() для начала и конца каждого ребра
4. Завершается режим рисования: glEnd()

Выполнение трёхмерных преобразований

Система координат и иерархия преобразований

Иерархия пространств в OpenGL:

1. Активируется режим рисования линий: glBegin(GL_LINES)
2. Для каждого ребра из списка edges извлекаются координаты соответствующих вершин
3. Последовательно вызывается glVertex3f() для начала и конца каждого ребра
4. Завершается режим рисования: glEnd()

В работе используется классический стек матриц OpenGL, где преобразования накапливаются.

Преобразования применяются в обратном порядке относительно их вызова в коде. Порядок фактического применения в матричных терминах: Масштабирование → Вращение → Перенос. Каждое преобразование соответствует умножению текущей матрицы на матрицу преобразования:

- Перенос: T = [[1,0,0,tx], [0,1,0,ty], [0,0,1,tz], [0,0,0,1]]
- Вращение вокруг X: Rx = [[1,0,0,0], [0,cos(a),-sin(a),0], [0,sin(a),cos(a),0], [0,0,0,1]]
- Масштабирование: S = [[sx,0,0,0], [0,sy,0,0], [0,0,sz,0], [0,0,0,1]]

Система проекций

Перспективная проекция (для 3D-вида)

Создаёт эффект перспективы, где удалённые объекты кажутся меньше. Параметры: 70° - угол обзора по вертикали, (w/h) - соотношение сторон, 0.1 и 500 - расстояния до ближней и дальней плоскостей отсечения

Ортографическая проекция (для 3D-вида)

Сохраняет параллельность линий, без эффекта перспективы. Используется для инженерных чертежей.

Настройка камеры

Для ортографических проекций *камера позиционируется перпендикулярно соответствующей плоскости: OXY (вид сверху)*: eyeZ = centerZ + 30, *OXZ (вид спереди)*: eyeY = centerY + 30, *OYZ (вид сбоку)*: eyeX = centerX + 30

Для 3D-вида используется *камера с привязкой к объекту*: (всегда смотрит на центр объекта, может перемещаться по сфере вокруг объекта, сохраняет ориентацию "верх-низ")

Интерактивное управление

Каждая клавиша изменяет соответствующий параметр преобразования:

- +/- - масштабирование (умножение/деление на 1.1)
- WASD - перенос по осям X и Y
- QE - перенос по оси Z
- xyz/XYZ - вращение вокруг соответствующих осей (±5°)
- RT - вращение вокруг произвольной оси (±10°)

Методы геометрических преобразований

Геометрические преобразования — это математические операции, которые изменяют положение, ориентацию или размеры объектов в пространстве, сохраняя при этом их основные геометрические свойства. В компьютерной графике используются преимущественно аффинные преобразования, которые сохраняют параллельность прямых, но могут изменять расстояния и углы.

В данной работе реализованы три фундаментальных типа преобразований:

1. *Жёсткие (изометрические) преобразования* — сохраняют расстояния и углы (перенос, вращение)
2. *Подобные преобразования* — сохраняют углы, но изменяют расстояния пропорционально (масштабирование)
3. *Аффинные преобразования* — сохраняют параллельность, но изменяют расстояния и углы произвольно

Все преобразования в работе реализованы с использованием *однородных координат (homogeneous coordinates)*, что позволяет представить все преобразования в виде умножения матриц 4×4.

Масштабирование

Масштабирование изменяет размеры объекта относительно начала координат или заданного центра.

Матрица масштабирования относительно начала координат:

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Особенности реализации

1. Равномерное масштабирование — все три коэффициента одинаковы
2. Накопительный эффект — каждое нажатие умножает текущий масштаб
3. Относительно начала координат — объект "растёт" из центра координат

Перенос

Перенос перемещает объект в пространстве без изменения его формы или ориентации.

Матрица переноса:

$$\begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Вращение

Вращение поворачивает объект вокруг заданной оси на определённый угол. *Вращение вокруг оси X:*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Вращение вокруг оси Y:

$$\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Вращение вокруг оси Z:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Вращение вокруг оси (1, 1, 1) (Матрица вращения Родрига):

$R = I + \sin(\theta)*K + (1-\cos(\theta))*K^2$

Где K — матрица векторного умножения:

$$K = \begin{bmatrix} 0 & -uz & uy \\ uz & 0 & -ux \\ -uy & ux & 0 \end{bmatrix}$$

Почему важен порядок преобразований

Основная математическая причина важности порядка преобразований заключается в том, что умножение матриц некоммутативно.

Это означает, что если у нас есть два преобразования — например, перенос (T) и вращение (R), то результат их композиции зависит от порядка:

T × R — сначала перенос, потом вращение

R × T — сначала вращение, потом перенос

Эти два результата не равны между собой, и каждый имеет совершенно разный геометрический смысл.

Вывод

В ходе выполнения лабораторной работы была успешно разработана и реализована интерактивная система трёхмерной визуализации с полным набором геометрических преобразований. Работа наглядно продемонстрировала важность порядка преобразований в компьютерной графике и показала, как теоретические знания о матричных преобразованиях применяются на практике для создания интерактивных графических приложений.

Разработанная система может быть расширена в следующих направлениях:

1. Усовершенствование визуализации
2. Расширение функциональности преобразований
3. Улучшение пользовательского интерфейса
4. Расширение форматов данных
5. Сетевое взаимодействие