Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Визуализация тел вращения заданных кривой Безье»

Студент	ИУ7-52Б		Н. Ф. Нисуев
	(Группа)	(Подпись, дата)	(И. О. Фамилия)
Руководитель курсовой работы			Н. Н. Мартынюк
		(Подпись, дата)	(И. О. Фамилия)
Консультант			
		(Подпись, дата)	(И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Кривые Безье и тела вращения являются элементами в компьютерной графике и проектировании, используемыми для создания плавных и точных поверхностей объектов.

Кривая Безье — тип кривой, предложенный в 60-х годах XX века независимо друг от друга Пьером Безье из автомобилестроительной компании «Renault» и Полем де Кастельжо из компании «Citroen», где применялись для проектирования кузовов автомобилей. Кривая Безье является частным случаем многочленов Бернштейна, описанных русским математиком Сергеем Натановичем Бернштейном в 1912 году. Впоследствии это открытие стало одним из важнейших инструментов систем автоматизированного проектирования и программ компьютерной графики [1].

Целью работы: разработка программного обеспечения, которое позволяет пользователю генерировать тела вращения с помощью кривой Безье, выбирать цвет тела вращения, расположение источника света и камеры.

Для достижения поставленной цели необходимо решить следующие задачи:

— изучение методов генерации кривой Безье и тел вращения;

— анализ существующих алгоритмов создания кривой Безье и тел вращения;

— выбор подходящих алгоритмов для решения поставленной задачи;

— проектирование архитектуры и графического интерфейса программы;

— реализация структур данных и алгоритмов для работы с кривой Безье и телами вращения;

— описание структуры разрабатываемого ПО;

— написание программы и тестирование;

— исследование производительности программы при работе с телами вращения.

1 Аналитическая часть

В текущем разделе анализируются доступные алгоритмы, предназначенные для генерации кривой Безье, генерации и визуализации тел вращения. Также обосновывается выбор предложенного метода и описываются ограничения, в которых будет функционировать создаваемое программное обеспечение.

1.1 Описание методов генерации кривой Безье

Алгоритмы для генерации кривых Безье различаются по подходам и областям применения. Кривые Безье, основанные на полиномиальной интерполяции контрольных точек, широко используются в графике, анимации и моделировании.

1.1.1 Алгоритм де Кастельжо

Алгоритм де Кастельжо — один из самых известных методов генерации кривых Безье. Он основан на линейной интерполяции между контрольными точками и повторном применении интерполяции для получения точек на кривой.

Принцип работы: Линейная интерполяция начинается между двумя соседними точками, создавая новые точки. Эти новые точки затем снова интерполируются, и так до тех пор, пока не останется одна точка — точка на кривой для данного значения параметра t.

Преимущества: Простота реализации, возможность динамического изменения кривой при перемещении контрольных точек;

Недостатки: Неэффективен для кривых высокого порядка, так как количество операций растёт экспоненциально с увеличением числа точек.

1.1.2 Полиномиальная форма Бернштейна

Кривые Безье могут быть определены с использованием полиномов Бернштейна, которые основаны на весовых коэффициентах для контрольных точек:

$$B(t) = \sum_{i=0}^{n} P_i \cdot B_i^n(t),$$

где $B_i^n(t)$ — полиномы Бернштейна степени n.

Принцип работы: Полиномы Бернштейна для каждой контрольной точки взвешивают вклад этой точки в общую форму кривой на основе параметра t.

Преимущества: Полиномиальная форма дает аналитическое представление кривой и удобна для математических преобразований.

Недостатки: Сложность вычисления может увеличиваться для кривых высокого порядка.

1.1.3 Выбор метода

Для интерактивных приложений с реальным временем (например, графические редакторы) наиболее подходит алгоритм де Кастельжо, благодаря своей простоте и гибкости.

1.2 Описание методов генерации тел вращения

Алгоритмы для генерации тел вращения играют важную роль в компьютерной графике и моделировании. Они используются для построения трёхмерных объектов, полученных путём вращения двумерных профилей (контуров) вокруг оси.

1.2.1 Метод дисков

Этот метод основан на разбиении поверхности вращаемого тела на малые диски, которые представляют сечения объекта.

Принцип работы: Двумерная кривая (профиль) разбивается на множество малых сегментов вдоль оси вращения. Каждый сегмент вращается на малый угол вокруг оси, формируя диск или кольцо. Соединение всех этих дисков создаёт объект.

Преимущества: Простота реализации и возможность получения достаточно точного результата при большом количестве сегментов.

Недостатки: Для получения гладкой поверхности требуется использование большого количества сегментов, что увеличивает вычислительную нагрузку.

1.2.2 Метод треугольников

Этот метод использует триангуляцию поверхности для построения тела вращения. Он применим как для полигональных моделей, так и для параметрически описанных кривых.

Принцип работы: Профиль разбивается на вершины, которые затем вращаются вокруг оси, формируя трёхмерные координаты. Получившиеся вершины соединяются треугольниками или полигонами, образуя сетку, которая описывает поверхность тела вращения.

Преимущества: Высокая точность и гибкость, возможность контроля плотности сетки для точной визуализации.

Недостатки: Могут возникнуть сложности с производительностью при работе с очень плотными сетками.

1.2.3 Параметризация поверхности

Этот метод использует параметрическое описание как кривой (профиля), так и угла вращения, чтобы получить трёхмерные координаты всех точек на поверхности тела.

Принцип работы: Профиль описывается как параметрическая кривая (например, кривая Безье или B-сплайн). Каждая точка на кривой преобразуется в трёхмерные координаты путём её вращения вокруг оси на определённый угол. Параметры могут включать радиус и угол вращения.

Преимущества: Позволяет легко изменять форму и параметры объекта, например, радиус, высоту и угол вращения.

Недостатки: Вычисления могут быть сложными при работе с кривыми высокого порядка и требуют точного контроля параметризации.

1.2.4 Выбор метода

Если тело вращения задаётся кривой Безье и осью, то наилучший метод для генерации такого тела будет метод треугольников. Это позволит получить гладкую поверхность тела и обеспечить гибкость управления точностью сетки.

1.3 Описание методов визуализации тел вращения

Визуализация тел вращения требует не только генерации геометрии объекта, но и правильного удаления невидимых ребер, отображения света и цвета для создания реалистичной трёхмерной сцены. Рассмотрим основные методы визуализации тел вращения, которые включают обработку локальной модели освещения и работу с цветом поверхности [2].

1.3.1 Алгоритмы удаления невидимых рёбер

Удаление невидимых рёбер и поверхностей играет важную роль в оптимизации визуализации трёхмерных объектов, позволяя скрывать части объектов, которые не видны наблюдателю, и, таким образом, снижать вычислительную нагрузку. Существуют несколько основных алгоритмов для удаления невидимых рёбер [3].

Z-буфер

Z-буферизация является одним из наиболее распространённых методов для удаления невидимых поверхностей. В этом алгоритме создаётся буфер глубины (или Z-буфер), в котором хранится значение глубины z для каждого пикселя изображения. При рендеринге каждой новой поверхности её глубина z_{new} сравнивается со значением, уже находящимся в буфере z_{buffer}. Если $z_{\text{new}} < z_{\text{buffer}}$, то новый пиксель отображается, и буфер обновляется:

$$z_{\text{buffer}} = z_{\text{new}}$$

иначе поверхность отбрасывается.

Преимущества: Простота реализации и возможность использования в реальном времени.

Недостатки: Требует дополнительной памяти для хранения буфера глубины и может вызывать артефакты, если глубина не представлена с высокой точностью.

Обратная трассировка лучей

Обратная трассировка лучей моделирует процесс попадания лучей в камеру, отслеживая каждый луч до его источника, чтобы определить видимые поверхности. Для каждого пикселя на экране строится луч, проходящий через его координаты и направленный в сцену. Лучи проверяются на пересечения с объектами сцены, и видимым становится объект, находящийся на минимальной глубине z_{min} вдоль луча:

$$z_{\text{min}} = \min(z_i)$$

где z_i — глубина пересечения луча с i-м объектом. Этот метод учитывает отражения и преломления, что делает его идеальным для высококачественной визуализации.

Преимущества: Обеспечивает высокую точность визуализации с учётом сложных эффектов освещения, отражений и преломлений.

Недостатки: Очень высокая вычислительная сложность, поэтому метод применяется в основном для рендеринга статичных изображений, а не в реальном времени.

Алгоритм Робертса

Алгоритм Робертса выполняет удаление невидимых рёбер на основе ориентации граней объекта. Для каждой грани объекта рассчитывается нормаль $\mathbf{N} = (N_x, N_y, N_z)$, и определяется, направлена ли она к камере. Если скалярное произведение нормали $\mathbf{N}$ и вектора камеры $\mathbf{V} = (V_x, V_y, V_z)$ положительно, то грань видима:

$$\mathbf{N} \cdot \mathbf{V} > 0$$

Если это условие выполняется, рёбра грани считаются видимыми, и они включаются в рендеринг. Алгоритм хорошо подходит для полигональных объектов.

Преимущества: Позволяет эффективно исключать невидимые рёбра для полигональных моделей, снижая нагрузку на процесс рендеринга.

Недостатки: Может быть сложен в реализации для объектов со сложными формами и изогнутыми поверхностями.

1.3.2 Закраски

Закраска поверхностей играет ключевую роль в визуальном восприятии объектов на сцене, влияя на их реалистичность и восприятия. Будут рассмотрены простая закраска, закраска по Гуро и закраска по Фонгу. Примеры рассматриваемых закрасок можно увидеть на рисунке 1.1;

Простая закраска

Простая закраска или метод плоской закраски (flat shading) заключается в том, что каждому полигону присваивается один цвет. Цвет вычисляется на основе нормали и источника света, причём освещение рассчитывается только один раз для всей поверхности.

Принцип работы:

1. Для каждого полигона вычисляется нормаль.
2. Расчёт освещения производится в одной точке.
3. Результирующий цвет применяется ко всему полигону.

Преимущества: высокая скорость расчётов.

Недостатки: низкая реалистичность изображения.

Закраска по Гуро

Метод закраски по Гуро (Gouraud shading) позволяет достичь более плавных переходов между полигонами, благодаря интерполяции цвета между вершинами.

Принцип работы:

1. Для каждой вершины полигона вычисляется нормаль.
2. На основе нормали и освещения определяется цвет в вершинах.
3. Цвета интерполируются вдоль рёбер и внутри полигона для получения плавного градиента.

Преимущества: плавные переходы между полигонами.

Недостатки: может не учитывать локальные изменения освещения внутри полигона, а также блики могут выглядеть размазанными.

Закраска по Фонгу

Метод закраски по Фонгу (Phong shading) обеспечивает высокую реалистичность за счёт интерполяции нормалей и расчёта освещения в каждой точке поверхности.

Принцип работы:

1. Нормали интерполируются между вершинами внутри полигона.
2. Освещение рассчитывается для каждой точки полигона на основе интерполированных нормалей.
3. Цвета интерполируются вдоль рёбер и внутри полигона для получения плавного градиента.

Преимущества: реалистичная передача бликов и теней.

Недостатки: высокая вычислительная сложность и долгий рендеринг.

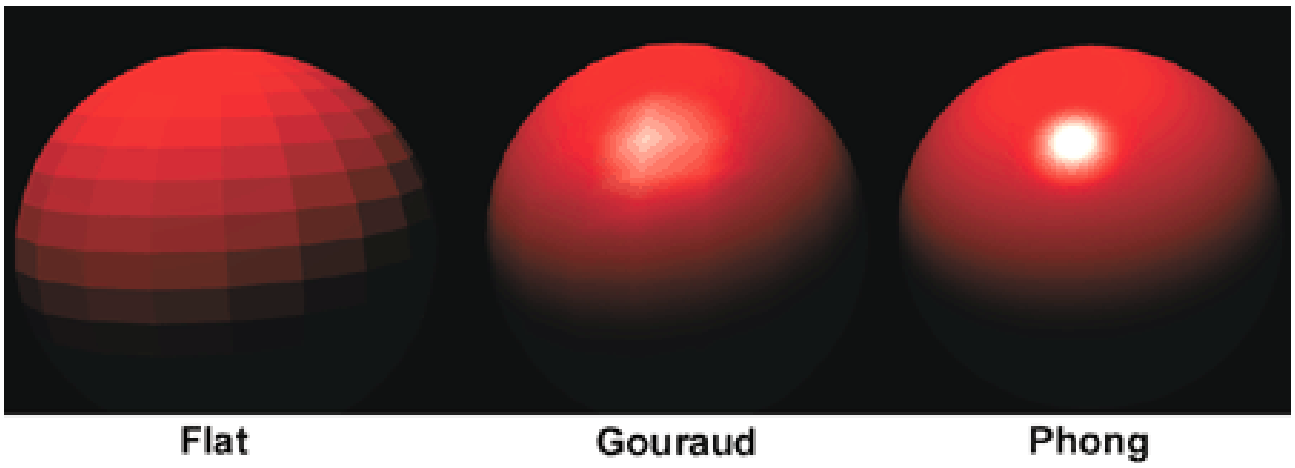

Рисунок 1.1 – Примеры закрасок

1.3.3 Модели освещения

Модель освещения Ламберта

Модель освещения Ламберта описывает диффузное отражение света, при котором интенсивность отражённого света пропорциональна косинусу угла между направлением света и нормалью к поверхности. Формула имеет вид:

$$I = I_0 k_a + I_t k_d \cos\theta, \quad 0 \leq \theta \leq \frac{\pi}{2},$$

где:

— I — интенсивность отражённого света;

— I_0 — интенсивность рассеянного света;

— k_a — коэффициент диффузного отражения рассеянного света ($0 \leq k_a \leq 1$);

— I_t — интенсивность точечного источника света;

— k_d — коэффициент диффузного отражения для направленного света ($0 \leq k_d \leq 1$);

— θ — угол между направлением света и нормалью к поверхности.

Эта модель предполагает, что поверхность отражает свет равномерно во всех направлениях, а интенсивность освещения зависит от угла падения света.

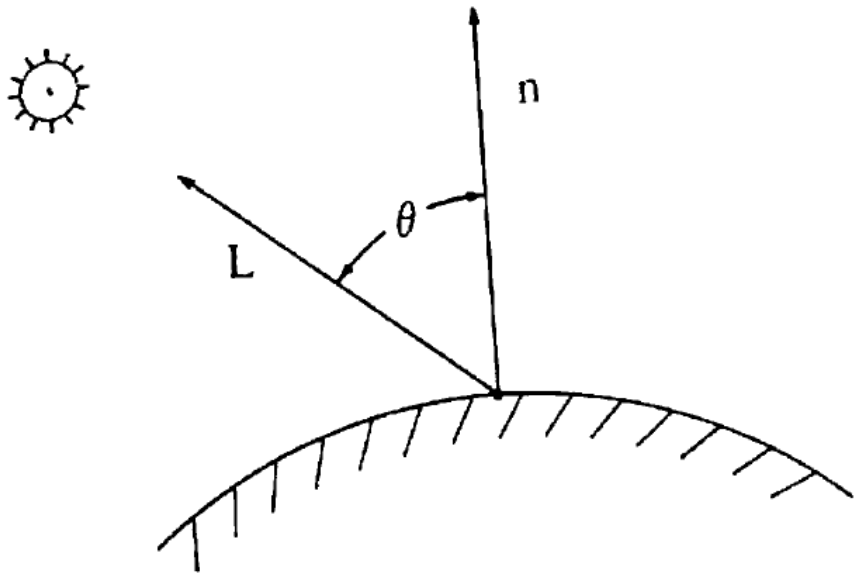

Рисунок 1.2 – Схема модели освещения Ламберта

Преимущества: Такой подход хорошо подходит для диффузных и матовых поверхностей.

Недостатки: не учитывает бликов и подходит только для равномерно освещённых объектов.

Модель освещения Фонга

Модель освещения Фонга расширяет модель Ламберта за счёт добавления зеркального отражения, описывающего поведение света на глянцевых поверхностях. Она состоит из трёх компонентов освещения: фонового, диффузного и зеркального. Формула имеет вид:

$$I = I_0 k_a + \frac{I_t k_d \cos\theta}{d + K} + I_t k_s \cos^n \alpha,$$

где:

— I — общая интенсивность света;

— I_0 — интенсивность фонового рассеянного света;

— k_a — коэффициент фонового освещения;

— I_t — интенсивность направленного источника света;

— k_d — коэффициент диффузного отражения;

— $\cos\theta$ — косинус угла между нормалью к поверхности и направлением света;

— d — расстояние от источника света до объекта;

— K — произвольная постоянная для уменьшения интенсивности с расстоянием;

— k_s — коэффициент зеркального отражения;

— $\cos\alpha$ — косинус угла между направлением отражённого света и направлением наблюдателя;

— n — степень блеска, управляющая шириной блика (чем больше n, тем уже блик).

Модель Фонга описывает, как свет распределяется на поверхности: диффузный компонент учитывает матовое отражение, а зеркальный — глянцевый блик, зависящий от угла обзора и гладкости поверхности.

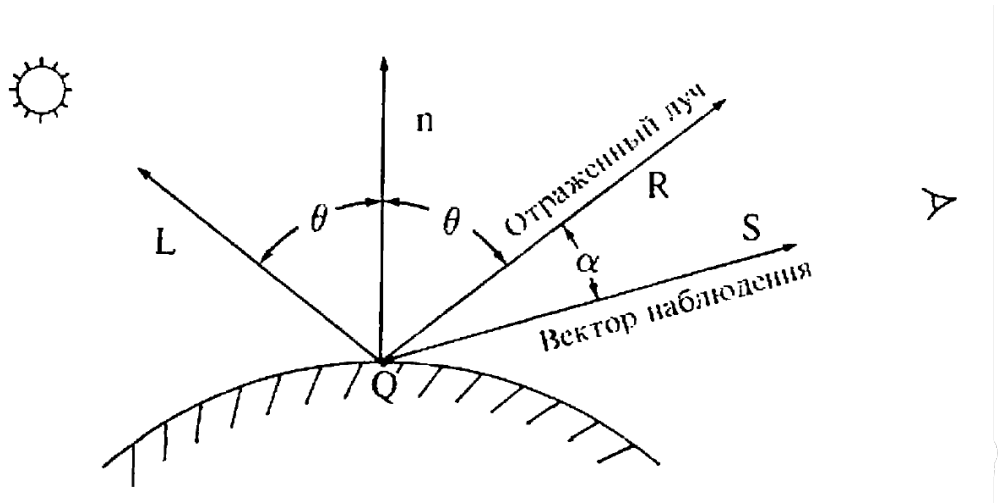

Рисунок 1.3 – Схема модели освещения Фонга

Преимущества: реалистичное освещение, подходящее для сложных сцен и объектов.

Недостатки: сложность вычислений.

1.3.4 Выбор методов

Для визуализации тел вращения оптимальным выбором являются Z-буфер, модель освещения Ламберта и закраска по Фонгу. Такое сочетание методов обеспечит высокое качество визуализации с учётом света, тени и цвета, а также на выходе получится реалистичное изображение.

ВЫВОД

В данном разделе проанализированы доступные алгоритмы, предназначенные для генерации кривой Безье, генерации и визуализации тел вращения. Также обоснован выбор предложенных методов и описаны ограничения, в которых будет функционировать создаваемое программное обеспечение.

2 Конструкторская часть

В данном разделе представлены требования к программному обеспечению, рассмотрены структуры данных и алгоритмы, выбранные для построения сцены.

2.1 Требования к программному обеспечению

Программа должна предоставлять следующий функционал:

— Задание кривой Безье;

— Генерация тела вращения, заданного кривой Безье;

— Изменение положения камеры;

— Изменение цвет тела вращения;

— Задание источника света и его положения;

— Сохранение сцену с телом вращения в изображение.

2.2 Описание структур данных

Для реализации работы программы были разработаны следующие структуры данных:

1. Кривая Безье содержит:
    — 2 основные точки, через которые проходит кривая;
    — массив опорных точек, задающих кривизну;
    — методы генерирующие точки кривой.
2. Сцена содержит:

— трехмерную модель объекта;

— камеру;

— источник света.

3. Трехмерная модель объекта содержит:

— массив вершин;

— массив полигонов;

— массив нормалей;

— массив цветов полигонов.

4. Полигон содержит:

— индексы тройки вершин из списка вершин, образующие грань.

5. Камера содержит:

— расстояние до объекта;

— угол на плоскости O_{xz};

— угол на плоскости O_{zy}.

6. Источник света содержит:

— позицию в пространстве;

— информацию о наличии света;

— интенсивность света.

2.3 Алгоритм генерации тела вращения

Алгоритм генерации тела вращения на рисунке 2.1.

Вход: упорядоченные точки на кривой, абсцисса оси вращения, количество сегментов.

Выход: полигональная модель.

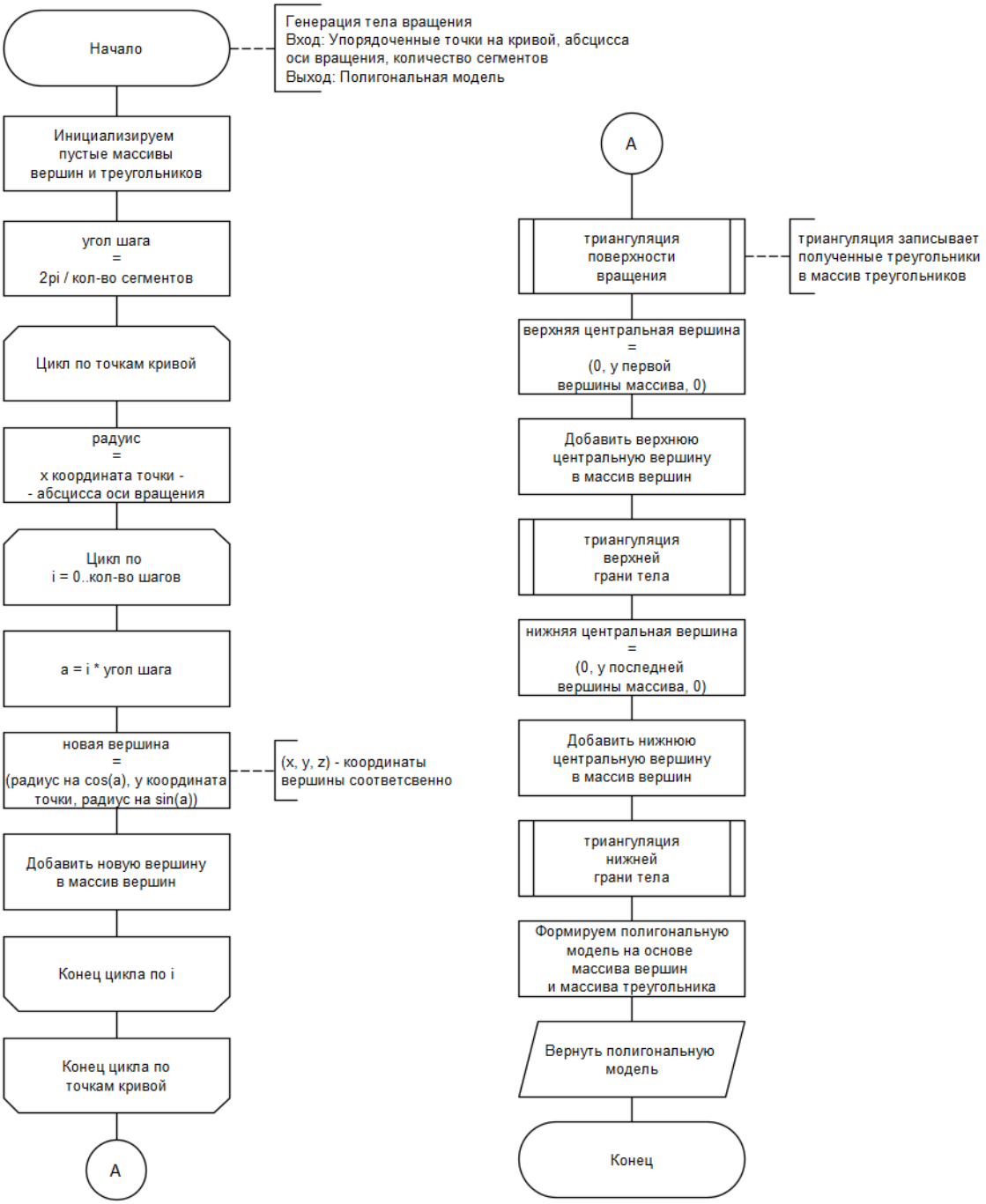

Рисунок 2.1 – Алгоритм генерации тела вращения

2.4 Общий алгоритм построения изображения

Алгоритм генерации изображения представлен на рисунке 2.2.

Вход: тело вращения и источник света, переведенные в пространство

камеры.

Выход: изображение трехмерной сцены в виде матрицы пикселей.

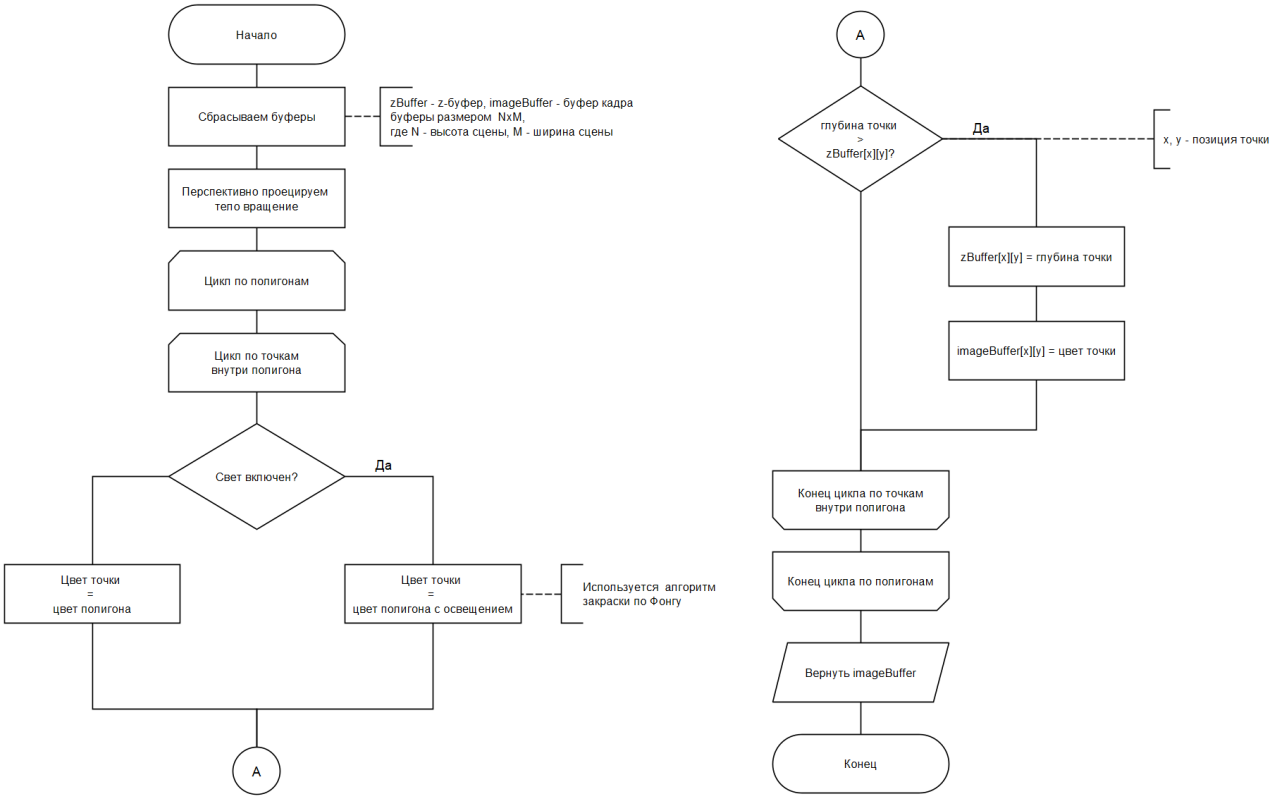

Рисунок 2.2 – Общий алгоритм построения изображения

2.5 Описание алгоритма Z-буфера

Z-буфер (или глубинный буфер) — это метод, используемый для скрытия невидимых поверхностей и определения, какие пиксели объекта должны отображаться на экране при наложении трёхмерных объектов. Z-буфер работает на основе хранения значений глубины (Z-координат) для каждого пикселя сцены.

1. Буфер кадра c_{buf} заполняется фоновым цветом, z-буфер z_{buf} заполняется $-\infty$.

2. Для каждого полигона сцены:

   — вычисляется глубина $z(x, y)$ для каждого пикселя полигона;

— Сравнивается глубина пикселя со значением глубины в z-буфере. Если $z(x, y) > z_{buf}(x, y)$, то $z_{buf}(x, y) = z(x, y)$ и $c_{buf} = colour$.

3. Вернуть буфер кадра.

Схема алгоритма z-буфера представлена на рисунке 2.3.

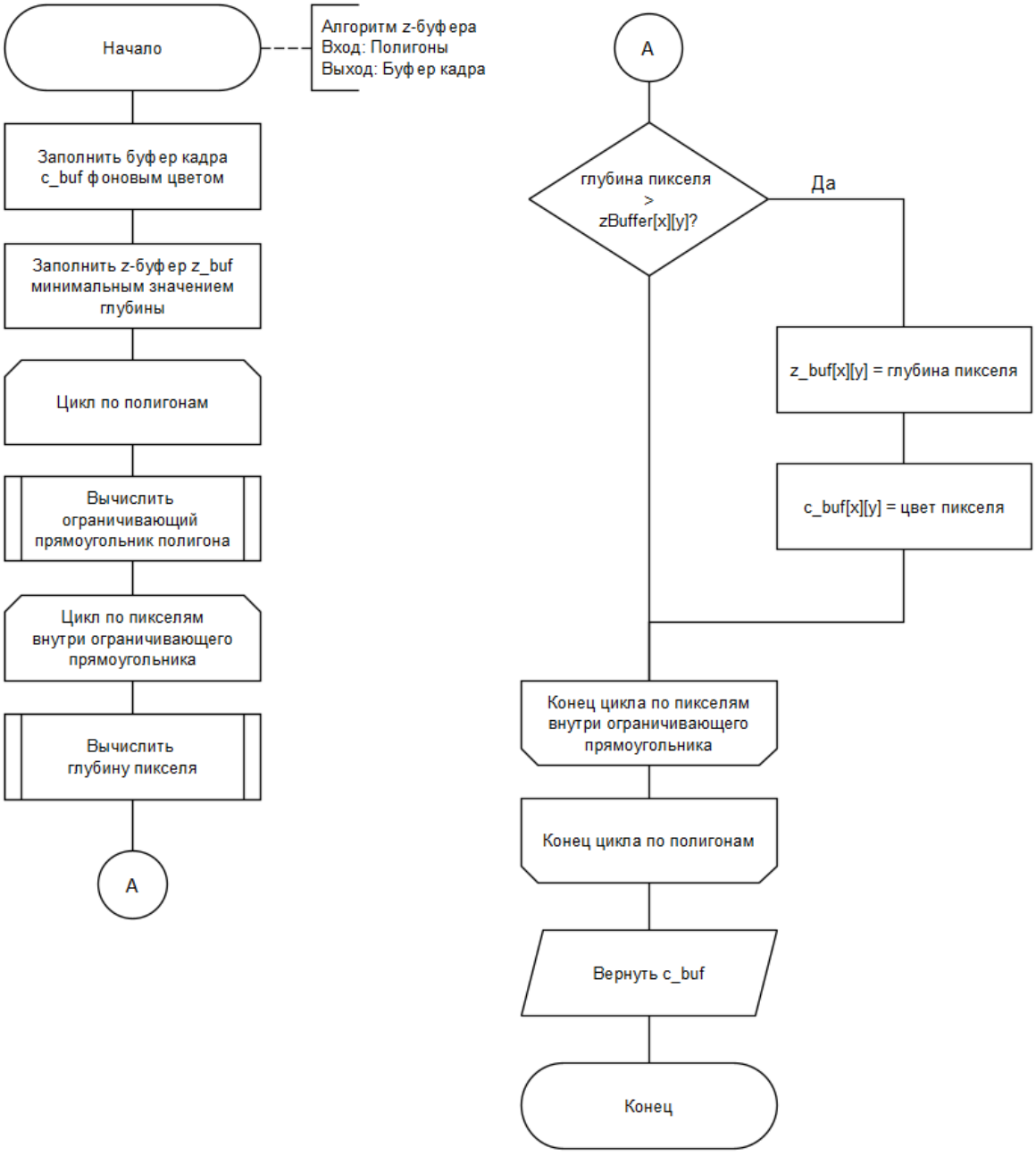

Рисунок 2.3 – Схема алгоритма z-буфера

ВЫВОД

В данном разделе были представлены требования к программному обеспечению, рассмотрены структуры данных и алгоритмы, выбранные для построения сцены.

3 Технологическая часть

В данной части рассматривается выбор средств реализации, описывается структура классов программы и приводится интерфейс программного обеспечения.

3.1 Средства реализации

Для реализации программного обеспечения выбран язык C++ [4]. Выбор обусловлен скоростью выполнения и наличием опыта работы с ним, также язык представляет весь необходимый функционал для решения поставленной задачи и поддерживает объектно-ориентированную модель разработки.

Для реализации пользовательского интерфейса программного обеспечения выбран фреймворк *Qt* [5], который содержит в себе средства, позволяющие работать напрямую с пикселями.

Средой разработки был выбран *QtCreator* [6], который обладает всем необходимым функционалом для написания, отладки программ, и создания графического пользовательского интерфейса.

Для сборки программного обеспечения использовалась утилита *qmake* [7].

3.2 Структура классов

Диаграмма классов представлена на рисунке 3.1.

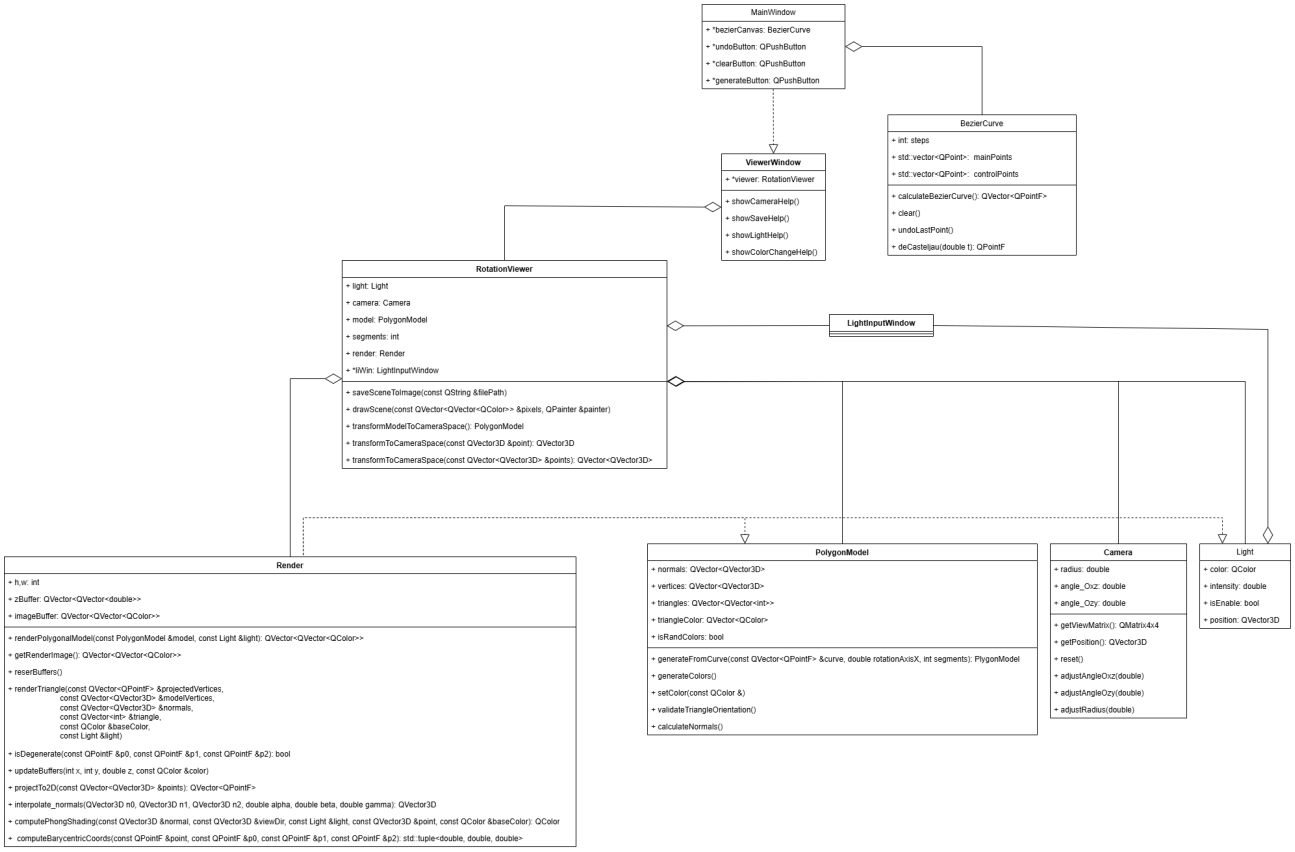

Рисунок 3.1 – UML диаграмма классов

3.3 Интерфейс программного обеспечения

Интерфейс программы представлен на рисунках 3.2 — 3.6.

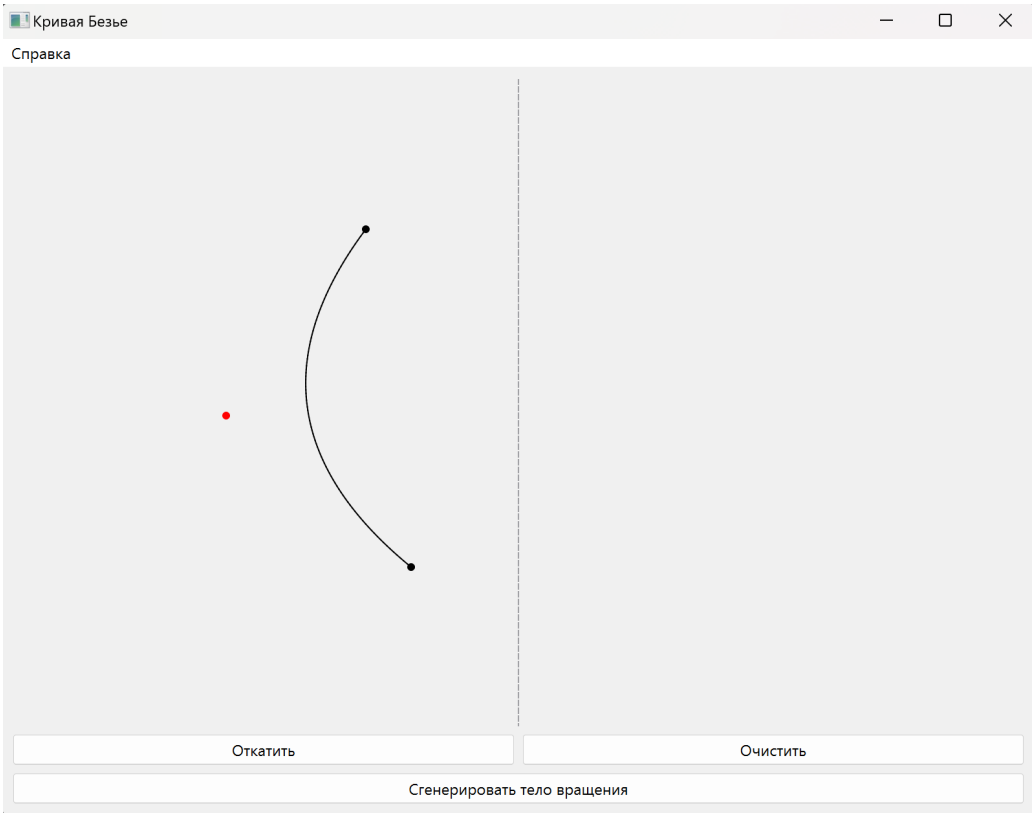

Рисунок 3.2 – Окно ввода кривой Безье

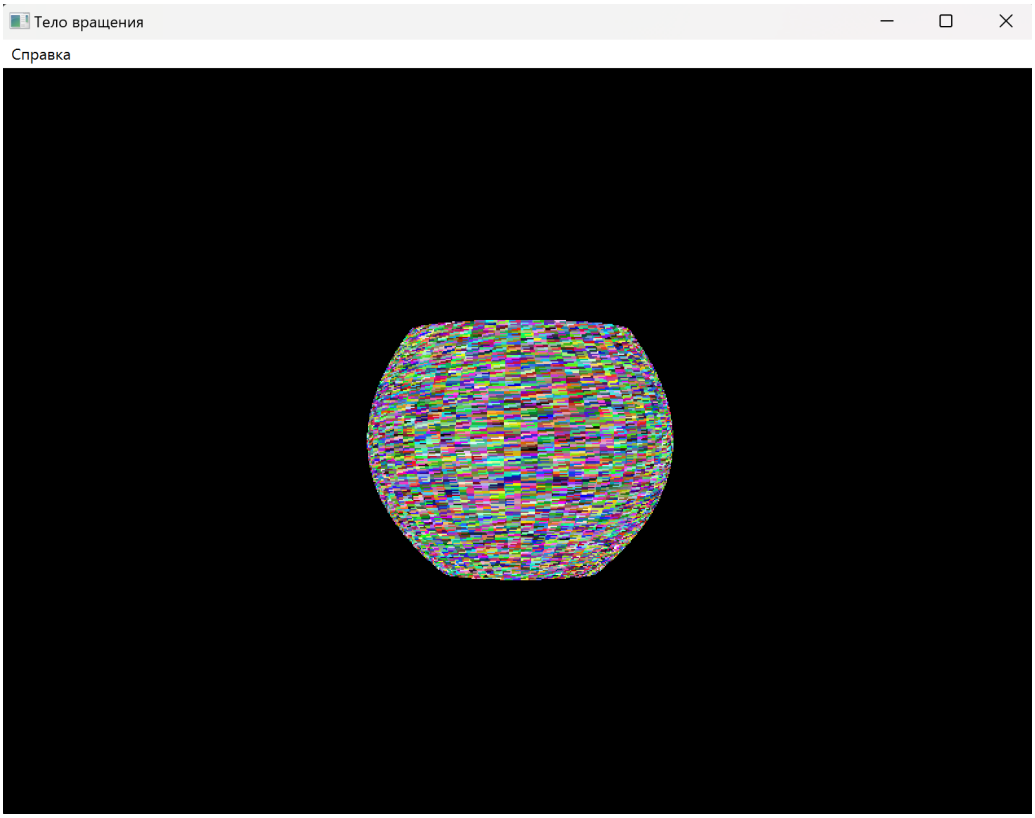

Рисунок 3.3 – Окно с телом вращения

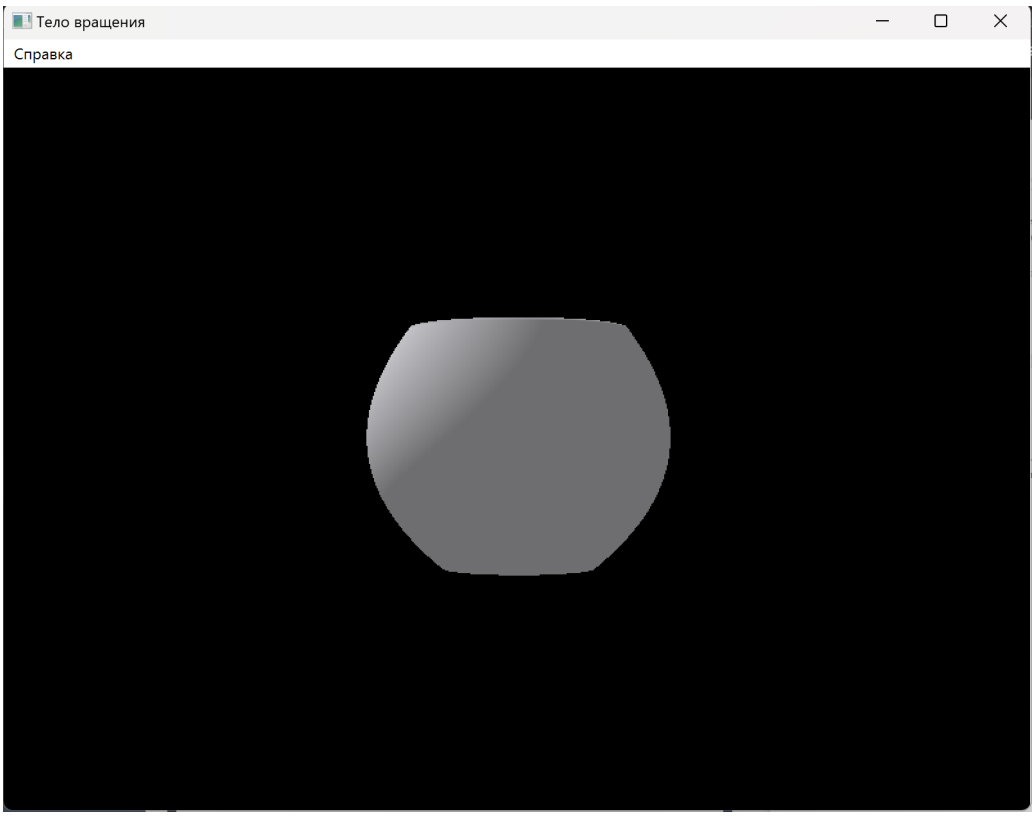

Рисунок 3.4 – Окно с освещенным телом вращения

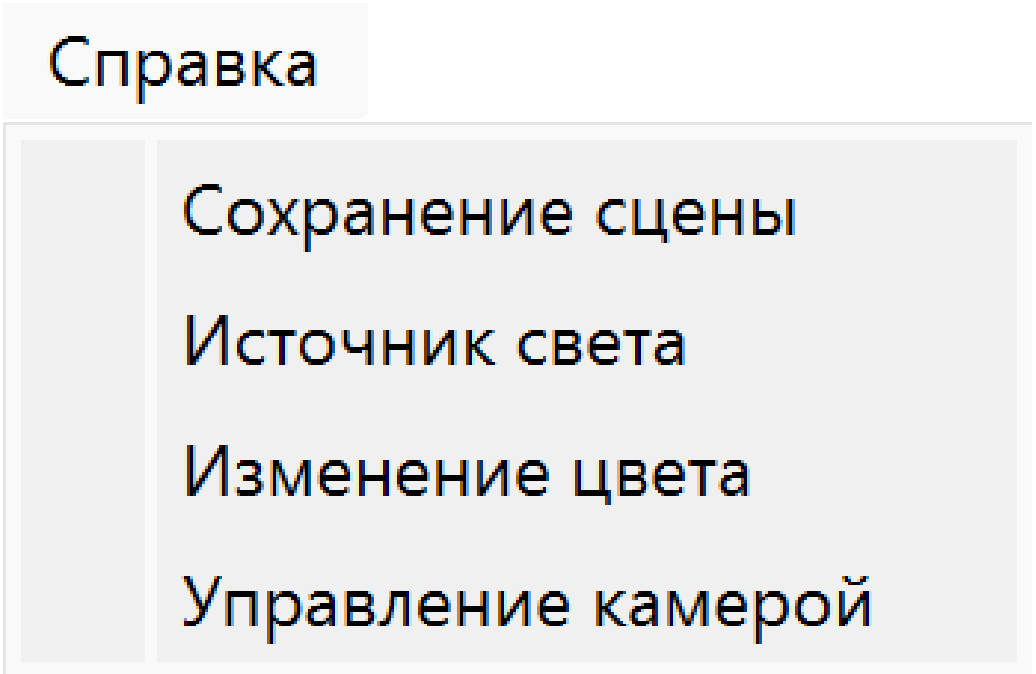

Рисунок 3.5 – Справочная информация

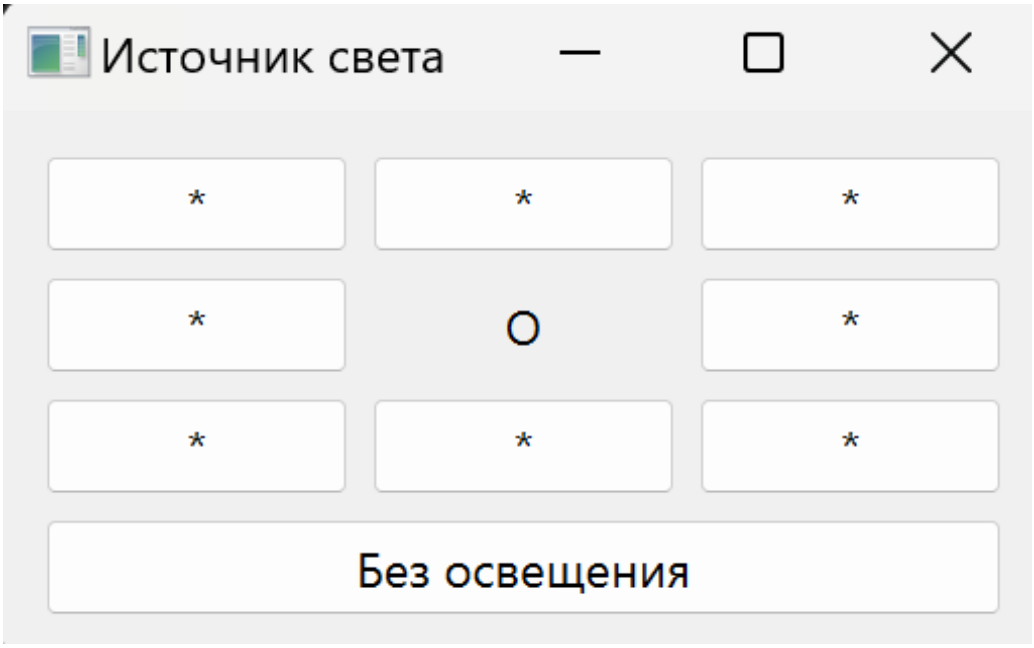

Рисунок 3.6 – Ввод положения камеры

4 Исследовательская часть

В данном разделе приведены технические характеристики устройства, на котором проводилось измерение времени работы программного обеспечения, а также результаты замеров времени.

4.1 Технические характеристики

Характеристики используемого оборудования:

— операционная система — Windows 11 Home;

— память — 16 Гб;

— процессор — 12th Gen Intel(R) Core(TM) i7-12700H @ 2.30 ГГц [8].

4.2 Цель исследования

Целью исследования является определение зависимости времени генерации тела вращения от количества сегментов и от количества точек на кривой.

4.3 Результаты замеров

Замеры времени проводились на одной кривой и с использованием библиотеки *QElapsedTimer* [9]. Каждое значение получено путем взятия среднего из 20 измерений. Зависимость времени генерации тела вращения от количества точек на кривой представлено на рисунке 4.1 и замерялась с константным числом сегментов, равным 50. Зависимость времени генерации тела вращения от количества сегментов представлено на рисунке 4.2 и замерялась с константным числом точек на кривой, равным 50.

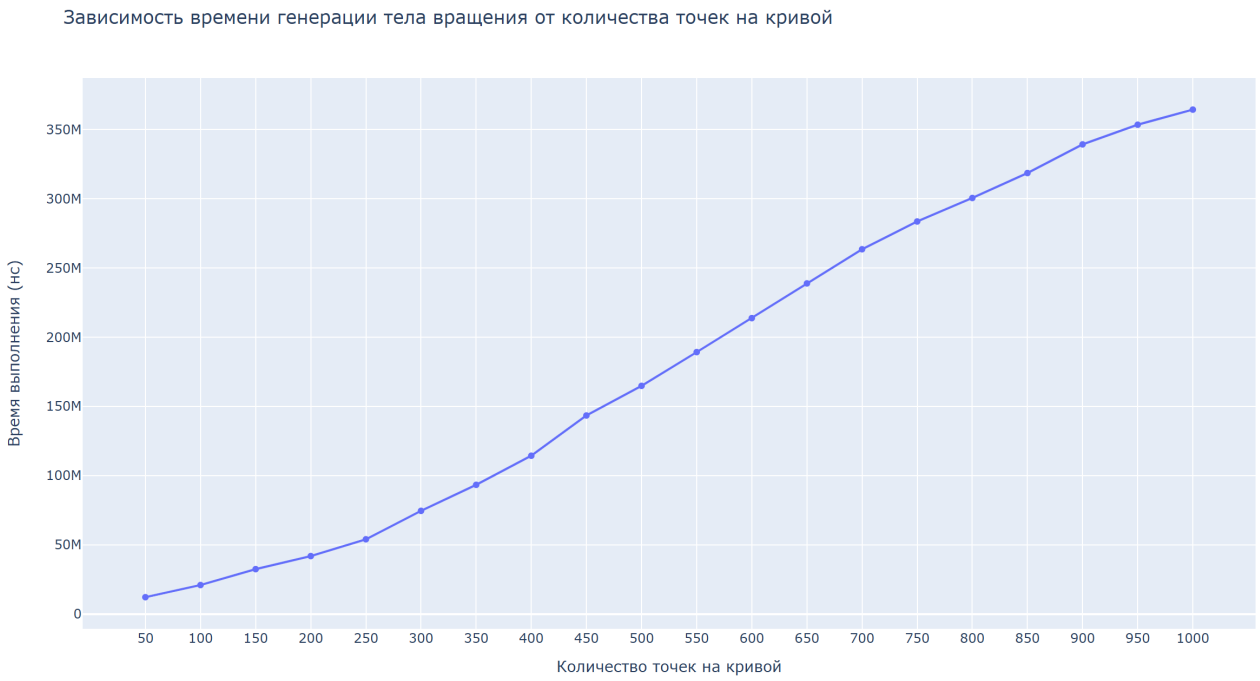

Рисунок 4.1 – Зависимость времени генерации тела вращения от количества точек на кривой

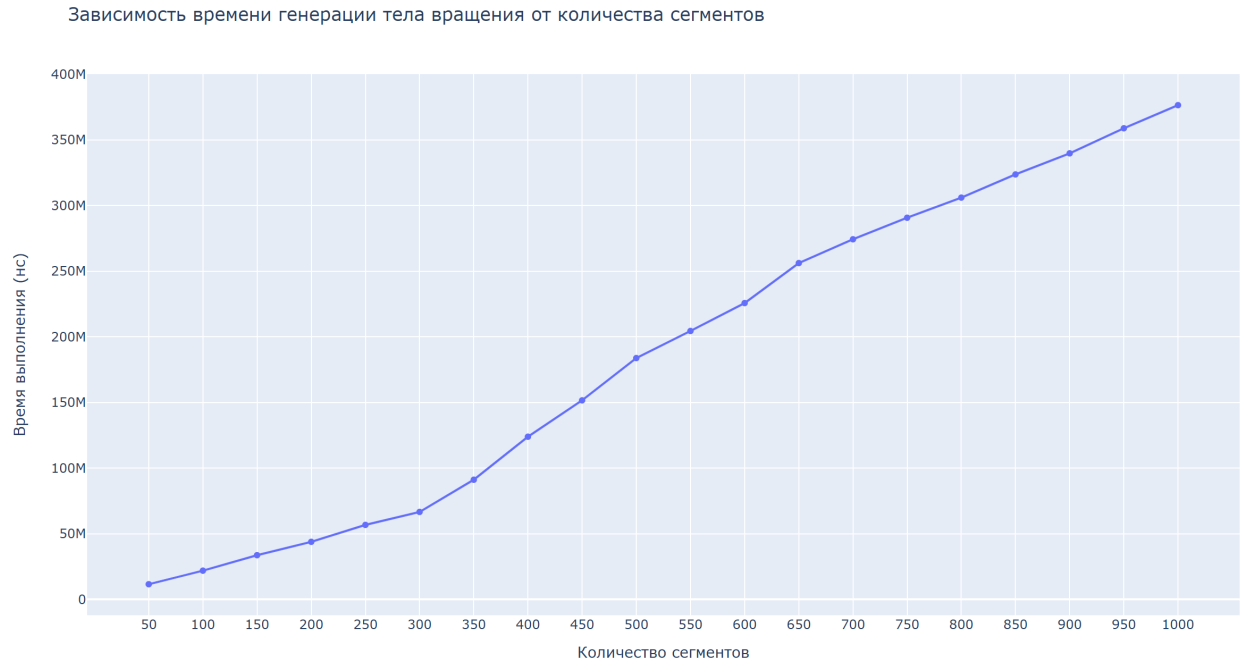

Рисунок 4.2 – Зависимость времени генерации тела вращения от количества сегментов

4.4 Результат исследования

По результатам исследования можно сделать вывод, что зависимость времени от количества точек на кривой и от количества сегментов имеет линейный характер.

ВЫВОД

В данном разделе были приведены технические характеристики устройства, на котором проводилось измерение времени работы программного обеспечения, а также результаты замеров времени.

ЗАКЛЮЧЕНИЕ

В результате курсовой работы было разработано программное обеспечение, которое позволяет пользователю генерировать тела вращения с помощью кривой Безье, выбирать цвет тела вращения, расположение источника света и камеры.

В ходе выполнения данной работы были решены следующие задачи:

— изучены методов генерации кривой Безье и тел вращения;

— проанализированы существующие алгоритмы создания кривой Безье и тел вращения;

— выбраны подходящие алгоритма для решения поставленной задачи;

— спроектированы архитектуры и графического интерфейса программы;

— реализованы структуры данных и алгоритмы для работы с кривой Безье и телами вращения;

— описаны структуры разрабатываемого ПО;

— написана программы и тестирование;

— исследована производительности программы при работе с телами вращения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. From Bézier to Bernstein [Электронный ресурс]. — Режим доступа: `https://www.ams.org/publicoutreach/feature-column/fcarc-bezier` (дата обращения: 09.12.2024).
2. *Роджерс Д.* Алгоритмические основы машинной графики. — Рипол Классик, 1989.
3. *Емельянова Т. В.*, *Аминов Л. А.*, *Емельянов В. А.* Реализация алгоритма удаления невидимых граней // Актуальные проблемы военно-научных исследований. — 2021. — № 2. — С. 37—44.
4. *Stroustrup B.* An overview of the C++ programming language // Handbook of object technology. — 1999. — С. 72.
5. Qt Documentation [Электронный ресурс]. — Режим доступа: `https://doc.qt.io/` (дата обращения: 08.12.2024).
6. Qt Creator Documentation [Электронный ресурс]. — Режим доступа: `https://doc.qt.io/qtcreator/` (дата обращения: 08.12.2024).
7. qmake Manual [Электронный ресурс]. — Режим доступа: `https://doc.qt.io/qt-6/qmake-manual.html` (дата обращения: 08.12.2024).
8. Intel® Core™ i7-12700H Processor [Электронный ресурс]. — Режим доступа: `https://ark.intel.com/content/www/us/en/ark/products/132228/intel-core-i7-12700h-processor-24m-cache-up-to-4-70-ghz.html` (дата обращения: 08.12.2024).
9. QElapsedTimer Class [Электронный ресурс]. — Режим доступа: `https://doc.qt.io/qt-6/qelapsedtimer.html` (дата обращения: 08.12.2024).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе

Презентация содержит 16 слайдов.