



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:

*«Визуализатор распространения звуковых волн в
замкнутом пространстве»*

Студент ИУ7-51Б
(Группа)

(Подпись, дата)

Постнов С. А.
(Фамилия И. О.)

Руководитель курсовой работы

(Подпись, дата)

Кузнецова О. В.
(Фамилия И. О.)

2023 г.

СОДЕРЖАНИЕ

| | |
|---|-----------|
| ВВЕДЕНИЕ | 4 |
| 1 Аналитический раздел | 5 |
| 1.1 Описание предметной области | 5 |
| 1.2 Формализация объектов сцены | 6 |
| 1.3 Методы описания моделей на сцене | 6 |
| 1.4 Выбор алгоритма удаления невидимых линий и поверхностей . | 8 |
| 1.4.1 Алгоритм, использующий Z-буфер | 9 |
| 1.4.2 Алгоритм Варнока | 9 |
| 1.5 Выбор модели освещения | 11 |
| 1.5.1 Модель Ламберта | 11 |
| 1.5.2 Модель Фонга | 11 |
| 1.6 Выбор метода закраски | 12 |
| 1.6.1 Метод простой закраски | 12 |
| 1.6.2 Метод закраски по Гуро | 13 |
| 1.7 Вывод | 13 |
| 2 Конструкторский раздел | 14 |
| 2.1 Требования к программному обеспечению | 14 |
| 2.2 Разработка алгоритмов | 14 |
| 2.2.1 Общий алгоритм решения поставленной задачи | 14 |
| 2.2.2 Алгоритм изображения звуковой волны | 16 |
| 2.2.3 Алгоритм простой закраски | 17 |
| 2.2.4 Алгоритм, использующий Z-буфер | 17 |
| 2.2.5 Алгоритм вычисления освещенности по модели Фонга . . | 19 |
| 2.3 Выбор типов и структур данных | 20 |
| 2.4 Вывод | 20 |
| 3 Технологический раздел | 21 |
| 3.1 Выбор средств реализации | 21 |
| 3.1.1 Выбор графического интерфейса для работы в трехмер- ном пространстве | 21 |
| 3.1.2 Выбор языка программирования | 21 |

| | | |
|---|--|-----------|
| 3.1.3 | Выбор среды разработки | 22 |
| 3.2 | Описание интерфейса пользователя | 22 |
| 3.3 | Демонстрация работы программы | 27 |
| 3.4 | Диаграмма классов | 29 |
| 3.5 | Вывод | 29 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | | 31 |

ВВЕДЕНИЕ

Компьютерная графика играет важную роль в современной технологической среде, находя применение в различных сферах, от разработки инженерного проектирования до компьютерных анимаций в мультфильмах. В наше время цифровое графическое представление достигло максимальной реалистичности, что требует огромных мощностей [1].

Целью данной курсовой работы является разработка ПО, которое позволит визуализировать процесс распространения звуковых волн в замкнутом пространстве.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) проанализировать предметную область;
- 2) формализовать объекты сцены;
- 3) рассмотреть известные методы описания моделей на сцене, подходы и алгоритмы удаления невидимых линий и поверхностей, модели освещения, методы закраски;
- 4) спроектировать программное обеспечение для визуализации распространения звуковой волны;
- 5) выбрать средства реализации;
- 6) исследовать характеристики разработанного программного обеспечения.

1 Аналитический раздел

1.1 Описание предметной области

Компьютерная графика изначально зародилась как эффективное и мощное средство связи между человеком и вычислительной машиной. Использование графической формы представления информации, организация диалога между человеком и компьютером с использованием визуальных образов позволили существенно увеличить скорость обработки информации человеком, что привело к повышению эффективности исследований и разработок в самых различных областях науки и техники. Одним из направлений компьютерной графики является визуализация звуковых волн [2; 3].

Звук — физическое явление, представляющее собой распространяющиеся в виде упругих волн механические колебания в твердой, жидкой или газообразной среде.

Изучение природы звуковых волн является важной частью на пути к пониманию и представлению более сложных физических процессов, что достигается путем применения методов визуализации.

1.2 Формализация объектов сцены

Объектами сцены будут являться:

- 1) замкнутое пространство — помещение прямоугольной формы, состоящее из четырех стен, пола и потолка и имеющее константный размер;
- 2) преграды — объекты кубической формы;
- 3) точечный источник звуковых волн — материальная точка в пространстве, характеризующаяся следующими свойствами:
 - источник звука излучает звуковые волны во все стороны равномерно;
 - источник не имеет предпочтительного направления для распространения звука;
 - звук распространяется от источника радиально, образуя сферическую волну.
- 4) источник света — материальная точка, задающая параметры освещенности объектов сцены;
- 5) камера — материальная точка в пространстве, которая определяет обзор и параметры визуализации.

1.3 Методы описания моделей на сцене

В компьютерной графике выделяют три основных вида описания моделей на сцене [4]:

- 1) каркасные — дают представление о поверхности объекта, но описывают его только дискретными элементами каркаса (точки или линии);
- 2) поверхностные — также дают представление о поверхности объекта, но несут информацию обо всех точках, принадлежащих этой поверхности;
- 3) твердотельные — объекты в виде сплошных тел, т. е. в виде сочетания всех точек занимаемого моделью объема.

В данном курсовом проекте выдвинуты следующие критерии к методам описания моделей на сцене [5]:

- 1) сложность обработки;
- 2) количество информации о модели.

Вывод

В таблице 1.1 приведены результаты сравнения методов описания моделей на сцене, исходя из выдвинутых критериев [6].

Таблица 1.1 – Сравнительная таблица для методов описания объектов на сцене

| | <i>Каркасная</i> | <i>Поверхностная</i> | <i>Твердотельная</i> |
|------------------------------|-----------------------|---|---|
| <i>Сложность обработки</i> | $O(k \cdot n)$ | $O(n^2)$ | $O(n^3)$ |
| <i>Количество информации</i> | Только вершины модели | Каркасная модель, дополненная информацией о всех точках поверхности | Поверхностная модель, дополненная информацией о внутренних точках |

В качестве реализуемого метода был выбран поверхностный способ описания объектов сцены, так как он обладает оптимальной сложностью обработки $O(n^2)$ и при этом предоставляет всю необходимую информацию для решения поставленных задач.

1.4 Выбор алгоритма удаления невидимых линий и поверхностей

Для построения реалистичных изображений необходимо предусмотреть возможность удаления невидимых линий и поверхностей.

Выделяют два основных вида таких алгоритмов [7]:

- 1) алгоритмы, работающие в пространстве объекта: использующий Z-буфер, Варнока;
- 2) алгоритмы, работающие в пространстве изображения: Робертса.

В данном курсовом проекте выдвинуты следующие критерии к алгоритмам удаления невидимых линий и поверхностей [8]:

- 1) система координат, с которой работают алгоритмы;
- 2) объем вычислений.

В таблице 1.2 приведены результаты сравнения алгоритмов, работающих в пространстве объекта и в пространстве изображения [8]

Таблица 1.2 – Сравнительная таблица для алгоритмов, работающих в пространстве объекта и в пространстве изображения

| | <i>Алгоритмы, работающие в пространстве объекта</i> | <i>Алгоритмы, работающие в пространстве изображения</i> |
|--------------------------|---|--|
| <i>Система координат</i> | Алгоритмы работают с физической системой координат | Алгоритмы работают с экранной системой координат |
| <i>Объем вычислений</i> | Растет, как квадрат числа объектов n : $O(n^2)$ | Растет, как число объектов n , умноженное на число пикселей k : $O(n \cdot k)$ |

Далее рассматриваются только алгоритмы, работающие в пространстве изображения, так как они обладают линейной сложностью $O(n)$ (при фиксированном количестве пикселей на экране), а также работают с экранной системой координат.

1.4.1 Алгоритм, использующий Z-буфер

Алгоритм был предложен Эдом Кэтмулом и представляет собой обобщение буфера кадра. Обычный буфер кадра хранит в пространстве изображения коды цвета для каждого пикселя.

Идея алгоритма удаления поверхностей с Z-буфером состоит в том, чтобы для каждого пикселя дополнительно хранить величину глубины или координату Z. Когда очередной пиксель заносится в буфер кадра, происходит сравнение значения его Z-координаты с координатой Z пикселя, который уже имеется в буфере. Атрибуты нового пикселя и его Z-координата заносятся в буфер, если он ближе к наблюдателю, т. е. если Z-координата нового пикселя больше, чем координата старого.

Главное преимущество алгоритма заключается в его простоте, однако для его реализации требуется большой объем памяти. Кроме решения общей задачи удаления невидимых линий и поверхностей он позволяет достаточно просто вычислять изображение сечения трехмерного объема плоскостью с произвольной координатой $Z_{сеч}$, что предоставляет возможность обрабатывать сцену любой сложности [9; 10].

1.4.2 Алгоритм Варнока

Алгоритм работает в пространстве изображения и анализирует область на экране дисплея (окно) на наличие в них видимых элементов:

- если в окне нет изображения, то оно просто закрашивается фоном;
- если в окне имеется элемент, то проверяется, достаточно ли он прост для визуализации;
- если объект сложный, то окно разбивается на более мелкие, для каждого из которых выполняется тест на отсутствие и/или простоту изображения.

Рекурсивный процесс разбиения может продолжаться до тех пор пока не будет достигнут предел разрешения экрана [7].

Алгоритм может быть реализован в двух вариантах в зависимости от решаемой задачи:

- 1) удаление невидимых линий, в результате чего получается контурное изображение элементов сцены;

2) удаление невидимых поверхностей.

Реализация двух подходов различается в определении цвета областей, которые являются частями изображения каких-либо граней. Если область содержит пиксели, относящиеся к одной грани, и не содержит границ полигонов, то при удалении невидимых поверхностей визуализируемое изображение принимает значение цвета этой грани. Для случая удаления невидимых линий такая область считается «пустой», то есть не содержащей элементов изображения [9].

Вывод

В качестве реализуемого алгоритма удаления невидимых линий и поверхностей был выбран алгоритм, использующий Z-буфер в силу его простоты и универсальности (в виде обработки сцен любой сложности).

1.5 Выбор модели освещения

Для корректного наблюдения за визуализацией распространения волн существует необходимость в выборе модели освещения.

Выделяют две основные модели освещения [11]:

- 1) модель Ламберта;
- 2) модель Фонга.

1.5.1 Модель Ламберта

В общем виде модель освещения Ламберта состоит из суммы фоновой и диффузной компонент:

$$I = I_a + I_d = m_a \cdot L_a + m_d \cdot k_d \cdot L_d \quad (1.1)$$

где, I_a — интенсивность фоновой составляющей,
 I_d — интенсивность диффузной составляющей.

Модель Ламберта является одной из самых простых моделей освещения. Данная модель очень часто используется как часть других моделей, поскольку практически в любой другой модели освещения можно выделить диффузную составляющую. Данная модель может быть очень удобна для анализа свойств других моделей. Она является существенной частью модели Фонга [11].

1.5.2 Модель Фонга

В 1975 Фонг предложил модель освещения достаточно гладких поверхностей. Эта модель давно стала классикой и до сих пор остается самой популярной в компьютерной графике. В общем виде модель освещения Фонга состоит из суммы фоновой, диффузной и зеркальной составляющей и представляется формулой 1.2 [11]:

$$I = I_a + I_d + I_s \quad (1.2)$$

где, I_a — интенсивность фоновой составляющей,
 I_d — интенсивность диффузной составляющей,
 I_s — интенсивность зеркальной составляющей.

Вывод

Исходя из потребности в качественном и детальном наблюдении за визуализацией распространения волн, была выбрана модель освещения Фонга, являющаяся более сложной и классической по сравнению с моделью Ламберта.

1.6 Выбор метода закраски

Для идентификации и визуально корректного отображения объектов сцены необходимо выбрать алгоритм закраски. Выделяют три основных вида методов закраски [12]:

- 1) простая закраска;
- 2) закраска по Гуро;
- 3) закраска по Фонгу.

Далее будут рассмотрены только метод простой закраски и метод закраски по Гуро, так как метод закраски по Фонгу требует больших вычислительных затрат, которые не являются необходимыми [9].

1.6.1 Метод простой закраски

Алгоритм простой закраски вызывает расчет по модели освещения только 1 раз, в одной контрольной точке, которая может быть как вершиной примитива, так и его центром. Полученный таким образом цвет применяется ко всем пикселям примитива.

Этот алгоритм используется в том случае, когда важно не качество изображения, а производительность и относительно небольшие вычислительные затраты [11].

1.6.2 Метод закрашки по Гуро

Метод Гуро — закрашка, согласно которой цвет примитива рассчитывается лишь в вершинах, а затем линейно интерполируется по его поверхности, что значительно снижает вычислительные затраты.

Закрашка граней по методу Гуро осуществляется в четыре этапа [12]:

- 1) вычисление нормали к каждой грани;
- 2) определение нормали в вершинах многогранника путем усреднения нормали по всем полигональным граням, которым принадлежит вершина;
- 3) вычисление значения интенсивности освещения в вершинах при помощи нормали в вершинах;
- 4) закрашивание каждого многоугольника путем линейной интерполяции значений интенсивности в вершинах.

Вывод

Исходя из потребности только в идентификации объектов сцены и быстродействии, в качестве метода закрашки был выбран метод простой закрашки.

1.7 Вывод

В данном разделе были формализованы объекты сцены, а также рассмотрены методы описания моделей на сцене. Кроме того, были рассмотрены алгоритмы удаления невидимых линий и поверхностей, модели освещения и методы закрашки.

Были выбраны следующие алгоритмы, модели и методы:

- 1) в качестве алгоритма удаления невидимых линий и поверхностей — алгоритм, использующий Z-буфер;
- 2) в качестве модели освещения — модель освещения Фонга;
- 3) в качестве метода закрашки — метод простой закрашки.

2 Конструкторский раздел

В данном разделе представлены требования к программному обеспечению и формальное описание алгоритмов, выбранных в аналитическом разделе.

2.1 Требования к программному обеспечению

Разработанная программа должна обладать следующим функционалом:

- 1) перемещение, поворот и масштабирование сцены;
- 2) добавление и удаление объектов сцены;
- 3) добавление и удаление точечных источников звуковых волн;
- 4) добавление и удаление единственного точечного источника освещения.

2.2 Разработка алгоритмов

В данном разделе будут представлены схемы алгоритмов для решения поставленных задач.

2.2.1 Общий алгоритм решения поставленной задачи

Общий алгоритм решения поставленных задач представлен на рисунке 2.1. На вход подаются точки моделей (преград), камеры, волны (если она есть). На выходе получается сцена в текущий момент времени.

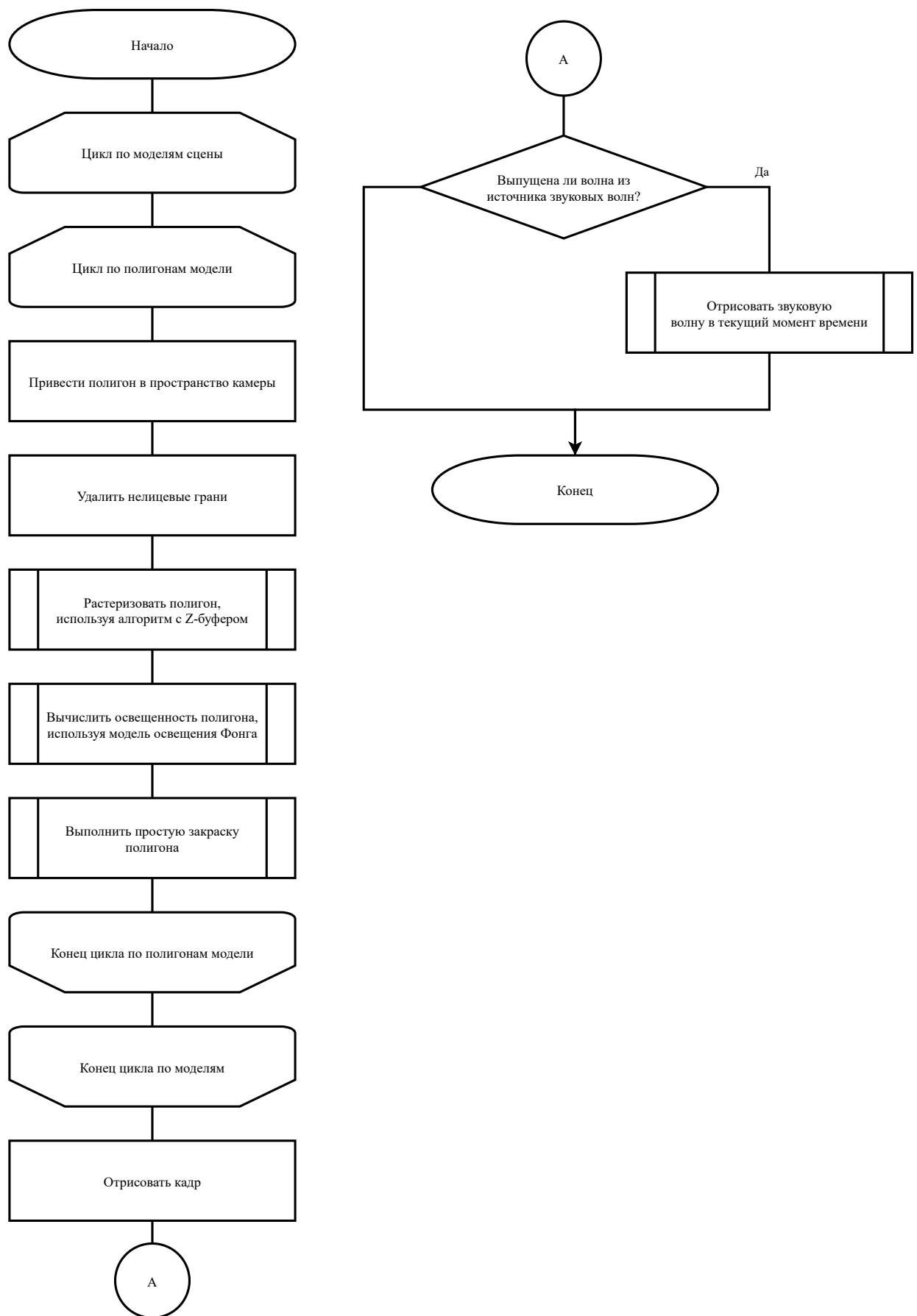


Рисунок 2.1 – Схема общего алгоритма решения поставленной задачи

2.2.2 Алгоритм изображения звуковой волны

Схема алгоритма изображения звуковой волны представлена на рисунке 2.2. На вход подаются точки, образующие волну, и их скорости. На выходе получается сцена с изображенной звуковой волной.

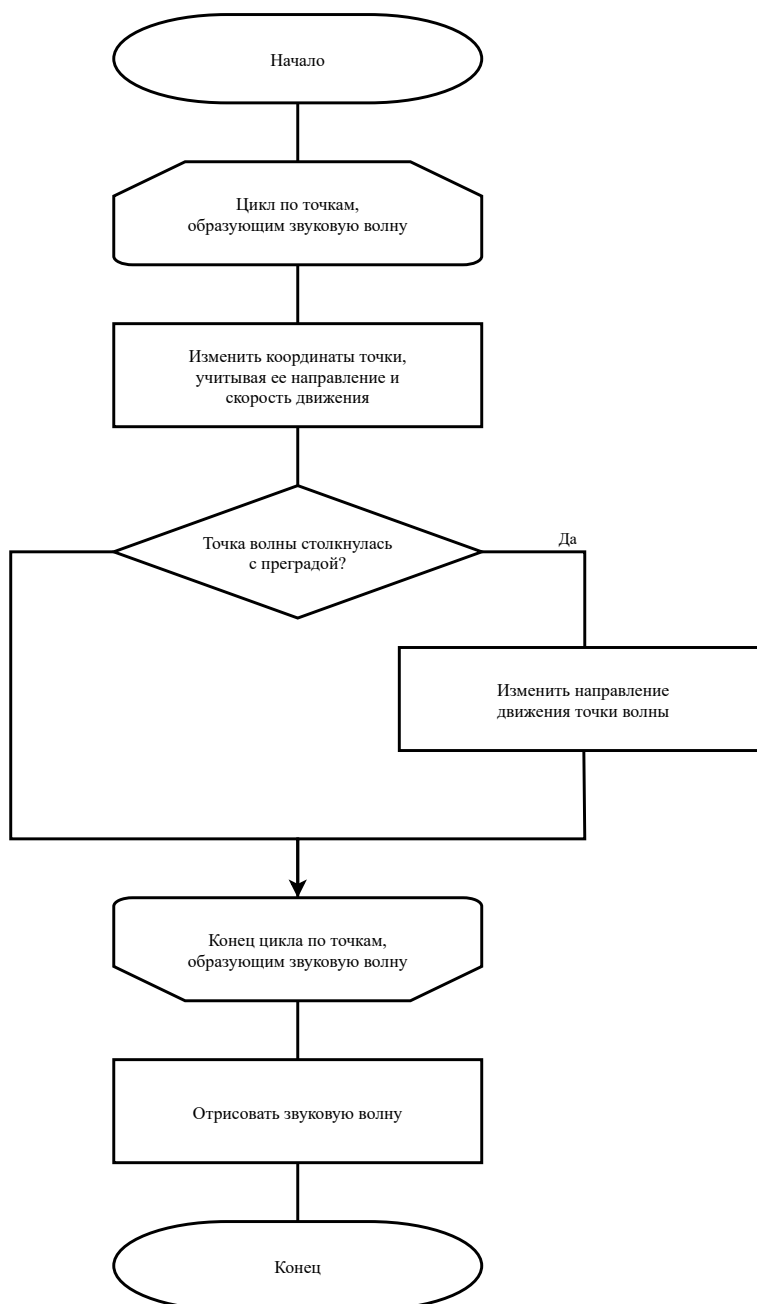


Рисунок 2.2 – Схема алгоритма изображения звуковой волны

2.2.3 Алгоритм простой закрашки

Схема алгоритма простой закрашки представлена на рисунке 2.3. На вход подаются точки модели (преграды) и ее цвет. На выходе получается сцена с закрашенной моделью.

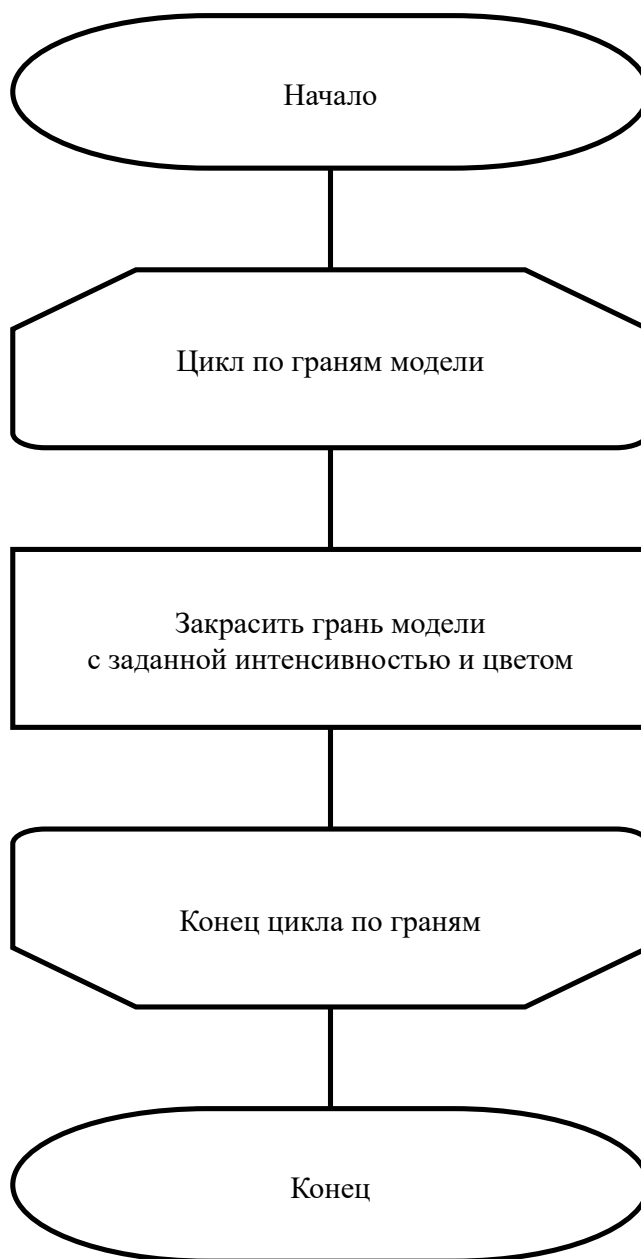


Рисунок 2.3 – Схема алгоритма простой закрашки

2.2.4 Алгоритм, использующий Z-буфер

Схема алгоритма, использующего Z-буфер, представлена на рисунке 2.4. На вход подаются точки модели сцены, буфер кадра. На выходе получается сцена без невидимых линий [13].

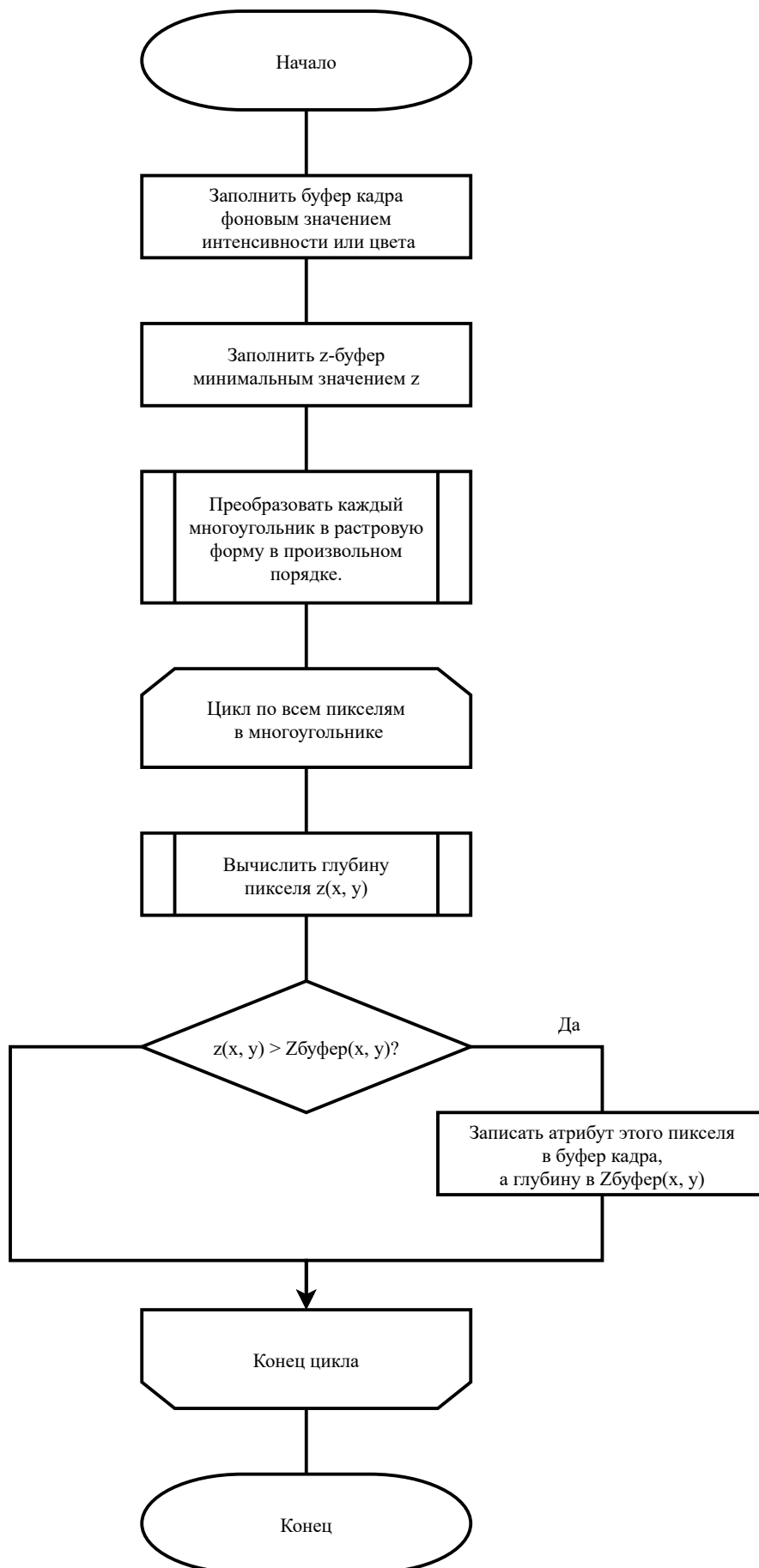


Рисунок 2.4 – Схема алгоритма, использующего Z-буфер

2.2.5 Алгоритм вычисления освещенности по модели Фонга

Схема алгоритма вычисления освещенности по модели Фонга представлена на рисунке 2.5. На вход подаются точки модели сцены и ее цвет, положение источника света. На выходе получается освещенная модель [11].

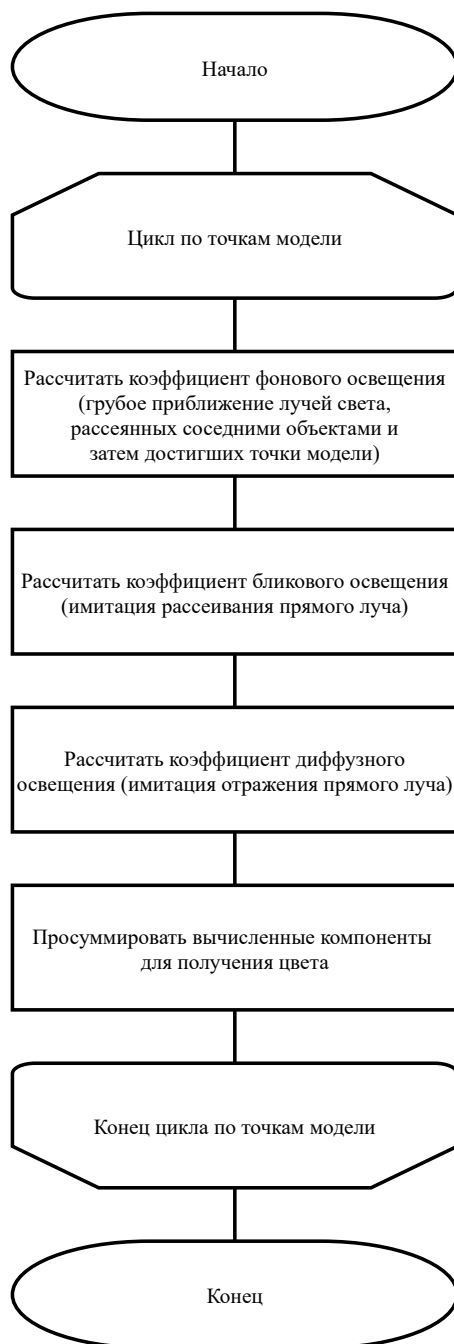


Рисунок 2.5 – Схема алгоритма вычисления освещенности по модели Фонга

2.3 Выбор типов и структур данных

Для решения поставленных задач курсового проекта необходимо определить представление объектов разрабатываемой программы. В таблице 2.1 представлены объекты и выбранные для них типы и структуры данных.

Таблица 2.1 – Таблица объектов и выбранных типов и структур данных

| <i>Объект</i> | <i>Типы и структуры данных</i> |
|----------------------------------|---|
| Точка в трехмерном пространстве | Вектор из трех элементов: x, y, z |
| Скорость точки | Вектор из трех элементов: sx, sy, sz |
| Вершина | Точка в трехмерном пространстве |
| Нормаль | Вектор из трех элементов: nx, ny, nz |
| Направление | Вектор из трех элементов: dx, dy, dz |
| Грань | Структура данных, содержащая: массив из 6 вершин и нормаль |
| Звуковая волна | Структура данных, содержащая: массив граней и массив скоростей точек |
| Препятствие | Массив граней |
| Замкнутое пространство (комната) | Массив граней |
| Источник звука | Точка в трехмерном пространстве |
| Камера | Структура данных, содержащая: направление и точку в трехмерном пространстве |

2.4 Вывод

В данном разделе были сформированы требования к программному обеспечению. Также были выбраны типы и структуры данных и описаны следующие алгоритмы:

- общий алгоритм построения сцены;
- алгоритм изображения звуковой волны;
- алгоритм простой закраски;
- алгоритм, использующий Z-буфер;
- алгоритм вычисления освещенности по модели Фонга.

3 Технологический раздел

В данном разделе будут выбраны средства реализации, которые будут использованы при разработке программного обеспечения. Также будут продемонстрированы интерфейс и работа программы.

3.1 Выбор средств реализации

Далее будет обоснован выбор языка программирования, графического интерфейса для работы в трехмерном пространстве и среды разработки.

3.1.1 Выбор графического интерфейса для работы в трехмерном пространстве

В качестве графического интерфейса для работы в трехмерном пространстве был выбран OpenGL в силу следующих причин [14]:

- 1) рендеринг на графическом процессоре (GPU) осуществляется быстрее, чем на центральном процессоре (CPU) [15];
- 2) большое сообщество разработчиков, а также множество материалов в открытом доступе.

3.1.2 Выбор языка программирования

В качестве языка программирования для разработки программного обеспечения был выбран C++ в силу следующих причин:

- 1) язык C++ изучался в рамках курса «Объектно-Оrientированное Программирование»;
- 2) язык C++ обладает высокой производительностью;
- 3) наличие библиотек для удобной работы с OpenGL и создания оконных приложений: glad, GLFW, GLEW [16].

3.1.3 Выбор среды разработки

В качестве среды для разработки программного обеспечения была выбрана Visual Studio 2022 в силу следующих причин [17]:

- 1) поддержка отладки графических приложений, а также большое количество инструментов для поиска и анализа ошибок;
- 2) удобная интеграция с языком C++;
- 3) наличие расширений для гибкой интеграции с OpenGL.

3.2 Описание интерфейса пользователя

При запуске разработанного ПО, на экране появляется вспомогательное окно «Главное меню» для взаимодействия с программой. Меню содержит следующие разделы, представленные на рисунке 3.1:

- «Препятствия» — раздел для управления препятствиями сцены;
- «Освещение» — раздел для управления освещением;
- «Источник звука» — раздел для управления источниками звуковых волн.



Рисунок 3.1 – Окно «Главное меню» для взаимодействия с программой

Раздел «Препятствия», представленный на рисунке 3.2, предоставляет пользователю возможность создавать препятствия, указывая при этом их параметры: цвет, позицию, масштаб и угол поворота. Также предусмотрена возможность удаления отдельных препятствий.

▼ Главное меню

▼ Препятствия

▶ Освещение

▶ Источник звука

Цвет препятствия

R: 51 G: 77 B:102

Позиция

0.000 X

0.000 Y

0.000 Z

Масштаб

1.000 RX

1.000 RY

1.000 RZ

Угол поворота (в градусах)

0 deg SX

0 deg SY

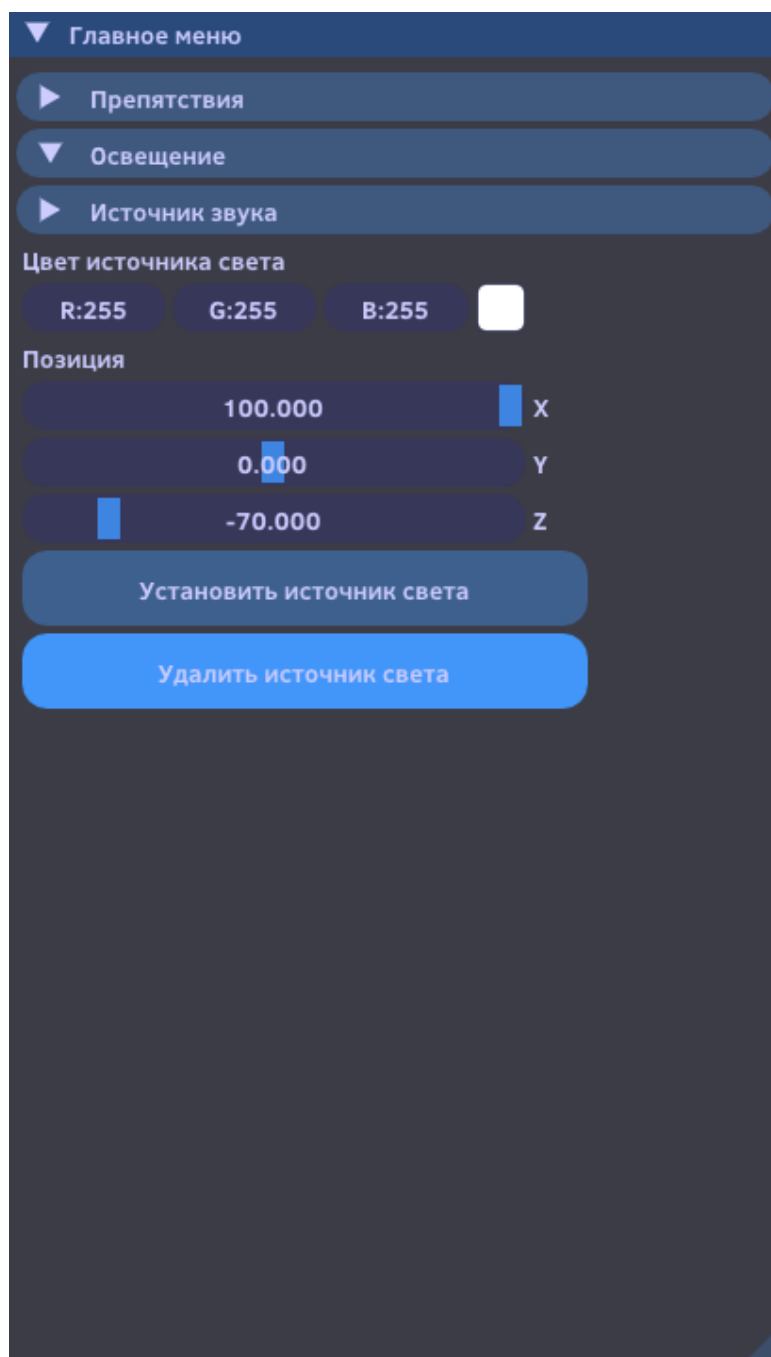
0 deg SZ

Разместить препятствие

Удалить препятствие

Рисунок 3.2 – Раздел «Препятствия»

Раздел «Освещение», представленный на рисунке 3.3, предоставляет пользователю возможность задавать положение источника света и его цвет. Также предусмотрена возможность удаления источника света.



▼ Главное меню

▶ Препятствия

▼ Освещение

▶ Источник звука

Цвет источника света

R:255 G:255 B:255

Позиция

X 100.000

Y 0.000

Z -70.000

Установить источник света

Удалить источник света

Рисунок 3.3 – Раздел «Освещение»

Раздел «Источник звука», представленный на рисунке 3.4, предоставляет пользователю возможность устанавливать источники звуковых волн и задавать скорость распространения звука из этих источников, после чего запускать процесс визуализации путем нажатия кнопки «Испустить волну из источников звука». Также предусмотрена возможность удаления отдельных источников звуковых волн.

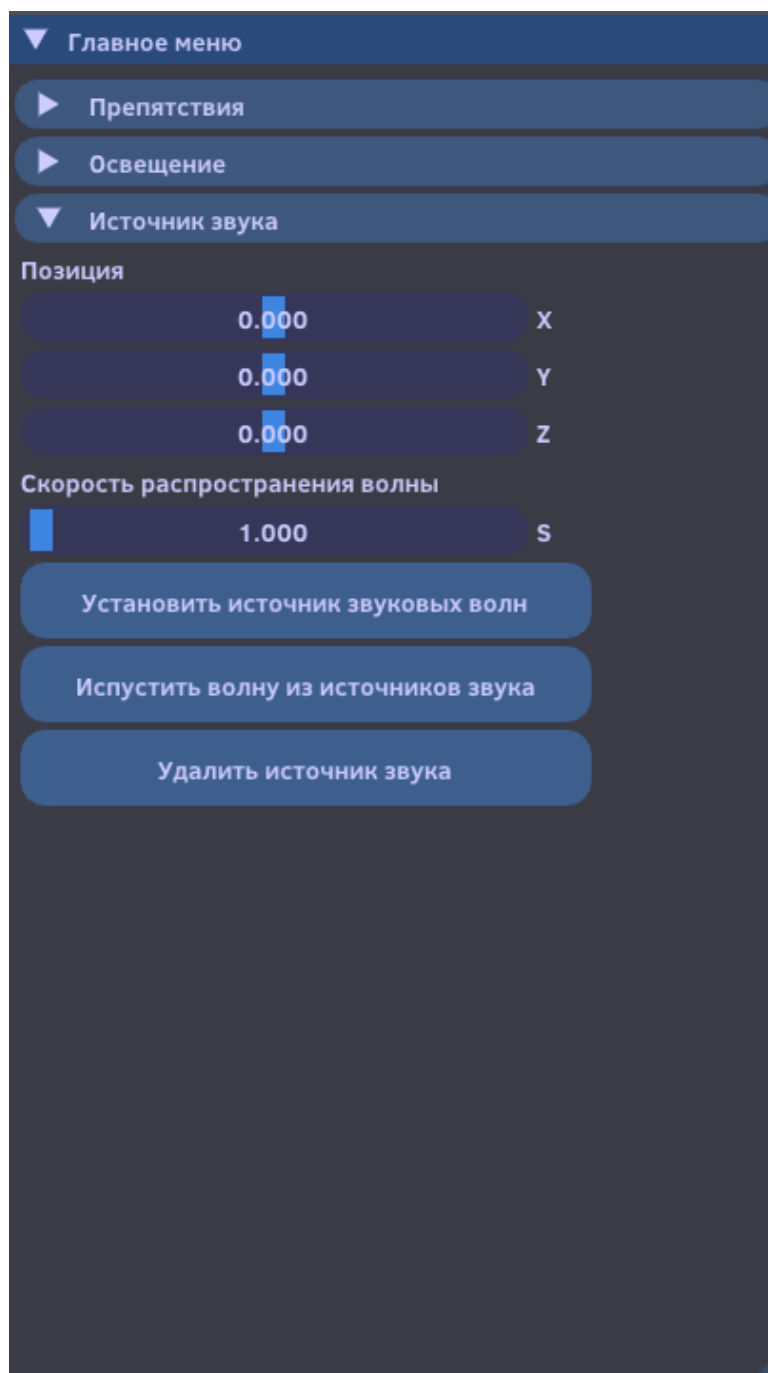


Рисунок 3.4 – Раздел «Источник звука»

3.3 Демонстрация работы программы

На рисунке 3.5 продемонстрирована работа программы при испускании нескольких звуковых волн из источников звука.

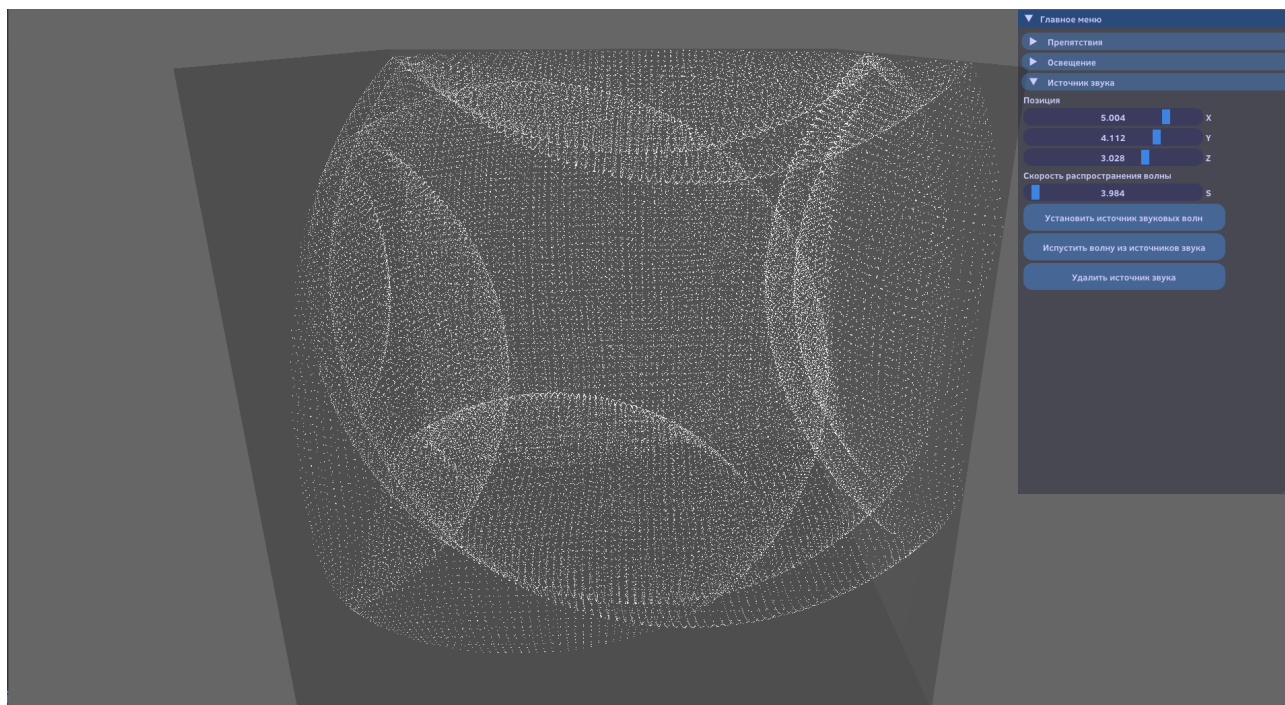


Рисунок 3.5 – Демонстрация работы программы при испускании нескольких звуковых волн из разных источников

На рисунке 3.6 продемонстрирована работа программы при испускании нескольких звуковых волн из источников звука, которые отражаются от установленного препятствия.

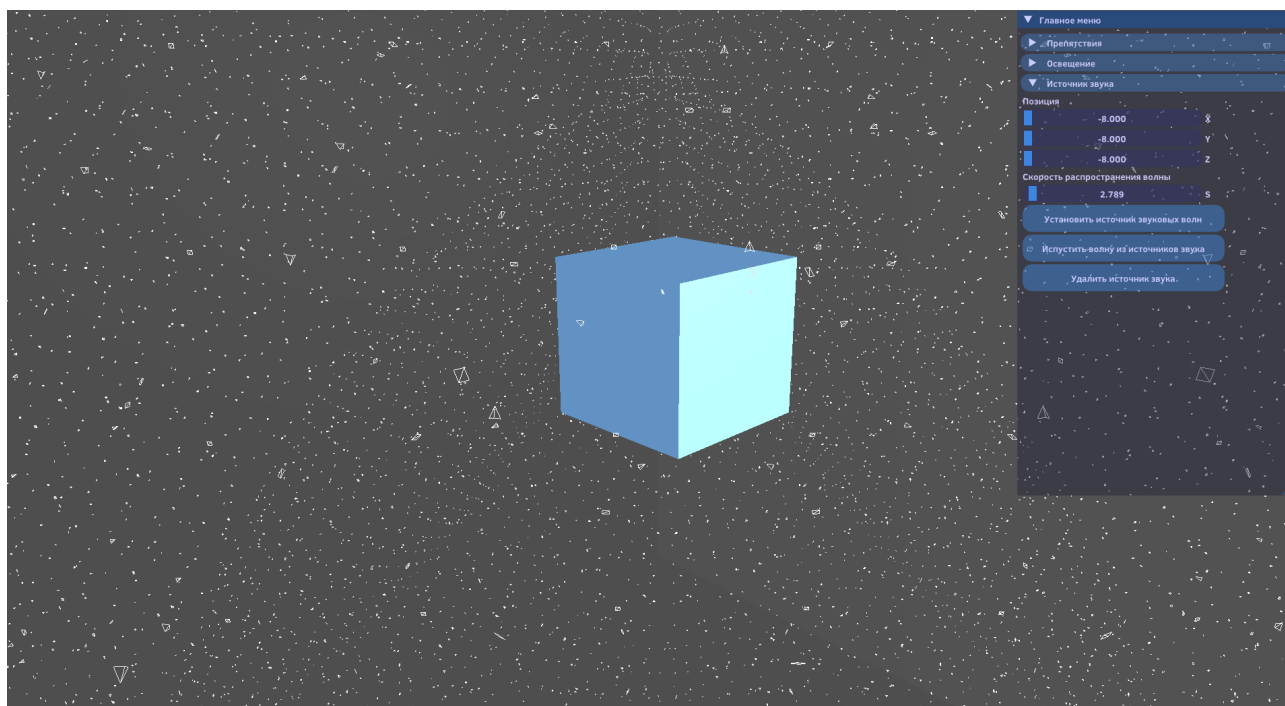


Рисунок 3.6 – Демонстрация работы программы при испускании нескольких звуковых волн из разных источников

3.4 Диаграмма классов

На рисунке 3.7 представлена диаграмма классов разработанного программного обеспечения.

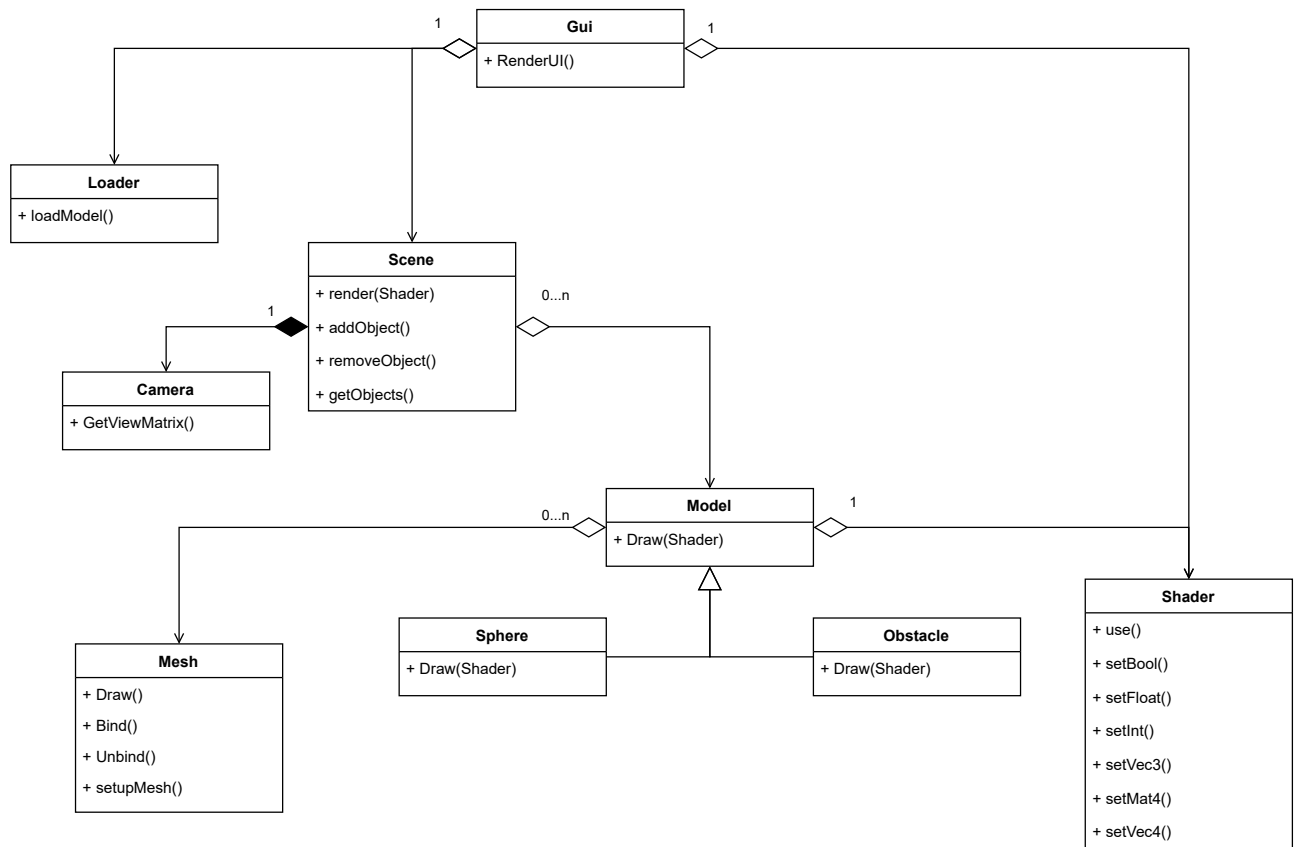


Рисунок 3.7 – Диаграмма классов разработанного программного обеспечения

3.5 Вывод

В данном разделе были выбраны средства для разработки программного обеспечения:

- 1) графический интерфейс для работы в трехмерном пространстве;
- 2) язык программирования;
- 3) среда разработки.

Также был описан интерфейс пользователя, приведены примеры работы программы и представлена диаграмма классов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Е. Ю. Цыбина.* КОМПЬЮТЕРНАЯ ГРАФИКА. СФЕРЫ ПРИМЕНЕНИЯ И ПЕРСПЕКТИВЫ РАЗВИТИЯ. // . — 2022. — С. 4.
2. *А. Г. Волобой.* Автореферат на тему «Программные технологии автоматизации построения реалистичных изображений». // . — 2012. — С. 35.
3. *Ю. Д. Кишиневская, Д. К. Иеронов.* ВИЗУАЛИЗАЦИЯ ЗВУКА И ЕГО ДЕЙСТВИЕ НА СТРУЮ ЖИДКОСТИ. // . — 2018. — С. 8.
4. *Ю. Н. Косников.* Поверхностные модели в системах трехмерной компьютерной графики. // . — 2007. — С. 60.
5. *«ЛЭТИ» С.-П. государственный электротехнический университет.* Конспект лекций по компьютерной графике. // . — 2014. — С. 6.
6. *С. А. Роменский, С. И. Ротков.* Формирование трехмерной каркасной модели в проблеме преобразования чертежноконструкторской документации на бумажном носителе в электронную модель объекта. // . — 2020. — С. 16.
7. Удаление скрытых линий и поверхностей. — [Электронный ресурс]. — Режим доступа: <https://algotlist.ru/graphics/delinvis.php> (дата обращения: 29.09.23).
8. Сравнительный анализ алгоритмов удаления невидимых линий и поверхностей, работающих в пространстве изображения. — [Электронный ресурс]. — Режим доступа: <https://novainfo.ru/article/3958> (дата обращения: 05.10.23).
9. *А. А. Головин.* БАЗОВЫЕ АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ. // . — 2016. — С. 18.
10. *Н. И. Витиска, Н. А. Гуляев, И. Г. Данилов В. В. С.* ОПТИМИЗАЦИЯ ПРЯМОЙ ОБЪЕМНОЙ ВИЗУАЛИЗАЦИИ С ПРОГРАММИРУЕМЫМ УПРАВЛЕНИЕМ КАЧЕСТВА. // . — 2020. — С. 82.
11. *А. Г. Задорожный.* МОДЕЛИ ОСВЕЩЕНИЯ И АЛГОРИТМЫ ЗАТЕНЕНИЯ В КОМПЬЮТЕРНОЙ ГРАФИКЕ. // . — 2020. — С. 80.
12. *Ю. А. Иванова.* Лекция на тему «Методы закраски». // . — 2019. — С. 51.

13. Электронный учебник «Компьютерная графика». — [Электронный ресурс]. — Режим доступа: <https://ychebnikkompgrafblog.wordpress.com/> (дата обращения: 25.10.23).
14. The Industry's Foundation for High Performance Graphics. — [Электронный ресурс]. — Режим доступа: <https://www.opengl.org/> (дата обращения: 23.11.23).
15. *М. К. Буза*. Высокоэффективные вычисления с применением графических процессоров. // . — 2015. — С. 5.
16. The Industry's Foundation for High Performance Graphics. — [Электронный ресурс]. — Режим доступа: https://www.khronos.org/opengl/wiki/OpenGL_Loading_Library (дата обращения: 24.11.23).
17. Visual Studio 2022. — [Электронный ресурс]. — Режим доступа: <https://visualstudio.microsoft.com/ru/vs/> (дата обращения: 24.11.23).