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»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Клименко Алексей Константинович
фамилия, имя, отчество

Группа ИУ7-45Б

Тип практики технологическая

Название предприятия МГТУ им. Н. Э. Баумана

Студент _____ **Клименко А. К.**
подпись, дата *фамилия, и.о.*

Руководитель практики _____ **Куров А. В.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2021 г.

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)

Кафедра «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

ЗАДАНИЕ
на прохождение производственной практики

на предприятии _____МГТУ им. Н. Э. Баумана, каф. ИУ7_____

Студент _____Клименко Алексей Константинович ИУ7-45Б_____

Во время производственной практики студент должен:

1. Спроектировать программу синтеза изображения трехмерной сцены, состоящей из полигональных объектов и источников света, с возможностью динамической смены параметров отрисовки.
2. Предоставить пользователю возможность добавлять на сцену объекты из готового набора примитивов, который включает в себя сферу, цилиндр, конус и тор.
3. Предоставить пользователю возможность загружать полигональные модели из файла.
4. Предоставить пользователю возможность сохранять полученное изображение сцены в файл.

Дата выдачи задания «15» июня 2021 г.

Руководитель практики от кафедры _____ _____Куров А. В._____
подпись, дата *фамилия, и.о.*

Студент _____ ___Клименко А. К._____
подпись, дата *фамилия, и.о.*

Содержание

ВВЕДЕНИЕ

В современном мире всё более и более востребованным становится использование компьютерной графики в разных сферах деятельности. Она используется в компьютерном моделировании, компьютерных играх, а также в системах автоматизированного проектирования. В связи с нарастающей потребностью к реалистичному изображению трехмерных объектов, появляется необходимость создавать программное обеспечение, способное учитывать широкий спектр оптических явлений при отрисовке изображения.

В компьютерной графике алгоритмы создания реалистичного изображения требуют особых затрат ресурсов, тратится много времени и памяти для получения результата. Затраты времени на синтез изображения не всегда позволяют удобно и эффективно подобрать параметры отрисовки.

Цель данной работы – создать программное обеспечение, которое позволило бы синтезировать изображение трехмерной сцены с настраиваемыми параметрами ее объектов в реальном времени и сменой режимов быстродействия.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. описать трехмерную сцену и объекты, из которых она состоит;
2. анализ и видоизменение существующих алгоритмов отрисовки, визуализирующих трехмерную сцену;
3. реализовать выбранные алгоритмы;
4. разработать программное обеспечение, позволяющее в реальном времени менять параметры сцены, а также параметры отрисовки.

1. Аналитический раздел

В данном разделе проводится обзор существующих способов описания трехмерных моделей, алгоритмов построения трехмерного изображения. Осуществляется анализ существующих моделей освещения и алгоритмов закраски полигонов.

1.1. Описание сцены

В компьютерной графике сцена является неотъемлемым компонентом, содержащим в себе всю информацию, необходимую для синтеза изображения. Поэтому определение ее структуры является одним из важнейших и первостепенных шагов в создании программы.

Сцена будет представлена структурой, содержащей в себе коллекцию из объектов и коллекцию из источников освещения. Размеры этих коллекций будут ограничены лишь объемом оперативной памяти.

1.2. Описание моделей трехмерных объектов в сцене

Существует множество способов представления трехмерных объектов в сцене. Различным алгоритмам отрисовки требуется разная информация о самом объекте.

1.2.1. Функциональное моделирование

Одним из простейших способов задания трехмерных объектов можно назвать функциональное моделирование. Возможны несколько вариантов:

1. Поверхность такого объекта задается системой уравнений:

$$\begin{cases} x = x(u, v) \\ y = y(u, v) \\ z = z(u, v) \end{cases}$$

В этом случае каждая точка на поверхности объекта из плоской системы координат (u, v) преобразуется в мировое пространство (x, y, z).

2. Внутренний объем такого объекта выражается неравенством:

$$f(x, y, z) > 0$$

Данное неравенство справедливо для всех точек внутри объекта.

Особенности данного представления:

- Функциональное моделирование позволяет получить максимальный уровень детализации при отрисовке, наилучшую сглаженность формы объекта.
- Достаточно сложная форма может иметь довольно простое параметрическое представление, из чего следует меньший расход памяти при хранении представления объекта.
- Трансформации над объектом производятся очень просто. Достаточно применить преобразования к описывающим объект уравнениям или неравенствам.
- Наложение текстур в случае с внутренне-определёнными объектами является затруднительным.
- Процесс отыскания параметрического представления невыпуклых многогранников может быть затруднительным.

1.2.2. Полигональная сетка

Другим представлением трехмерных объектов в сцене является представление в виде полигональной сетки. Структура полигональной модели представлена на рисунке 1.1.

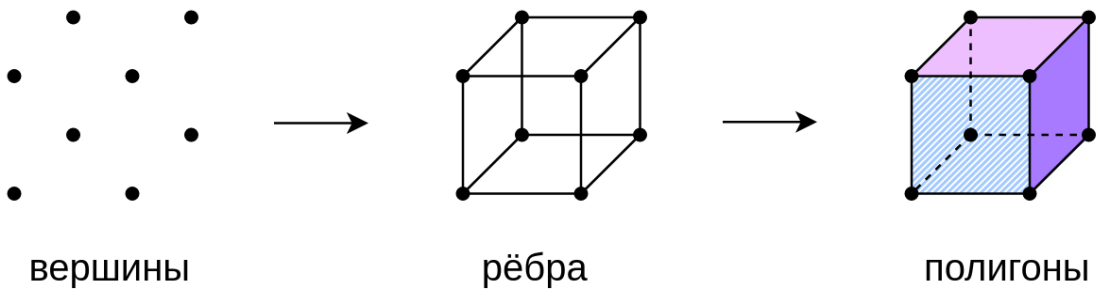

Рисунок 1.1 - Структура полигональной модели

Полигональная сетка состоит из набора отдельных полигонов, которые в свою очередь являются многоугольниками, ребра которых соединяют вершины сетки [1]. Полигональная сетка является аппроксимацией поверхности представляемого объекта. От её плотности, т.е. от количества полигонов на единицу объема, зависит качество изображения объекта.

Так как полигоны в сети могут представляться вне зависимости друг от друга, данный способ позволяет с меньшей сложностью описать объекты нетривиальной формы в сцене. Перечислим особенности данного представления:

- Просто создать топологию поверхности.
- Преобразования над объектом сводятся к преобразованию полигональной сетки, что в свою очередь раскладывается на независимые преобразования над отдельными полигонами.
- Наложение текстур на поверхность является несложной процедурой.
- Повышение плотности сетки приводит к увеличению объема затрачиваемой памяти для ее хранения.

1.2.3. Воксельная сетка

Наконец, есть еще один способ представления трёхмерных объектов - представление воксельной сеткой. Объем - это трехмерный массив кубических элементов (вокселей), представляющих единицы 3D пространства [2].

Воксельная сетка позволяет непосредственно представить объем объекта, его внутреннее содержание. Выделим особенности данного представления:

- Просто представляются объекты сцены.
- Легко выполняются отсечения частей объекта. Достаточно сделать часть вокселей прозрачными.
- Текстурирование тесно связано с воксельной сеткой. Нельзя изменить текстуру объекта без изменения воксельной сетки.
- Поворот и масштабирование объекта не могут быть осуществлены без перерасчета сетки.
- Чем больше объем объекта, тем больше затрачивается памяти на хранение его представления.

1.2.4. Вывод

Для решения поставленных задач лучше всего подходит метод с использованием полигональной сетки для представления поверхностей объектов, так как он рассчитан на широкий круг задач моделирования, позволяет легко модифицировать положение объекта в пространстве, а также прост в реализации.

1.3. Алгоритмы построения трехмерного изображения

Проведем сравнительный анализ алгоритмов построения трехмерного изображения для выбора наиболее подходящего. Целевыми показателями, на которые следует обратить внимание являются: зависимость скорости работы от числа изображаемых объектов на сцене, требуемый объем памяти, а также возможность распараллеливания процесса работы алгоритма.

1.3.1. Алгоритм художника

Принцип данного алгоритма заключается в том, что объекты на экране рисуются в порядке от дальних к ближним, при этом происходит наложение - ближние объекты рисуются поверх дальних. Этот алгоритм удобен в случаях, когда процесс распознавания видимых частей объекта является более

затратным, чем собственно его полная отрисовка. Но при этом этот алгоритм производит лишнюю работу по отрисовке невидимых частей объектов.

1.3.2. Алгоритм Варнока

Алгоритм Варнока работает по принципу “Разделяй и властвуй” и разбивает окно на подокна в случаях, когда его отрисовка нетривиальна. Такие разбиения могут продолжаться до тех пор, пока размер окна не станет равным единице растеризации или пока в окне не останется всего одной простой для отрисовки фигуры [3].

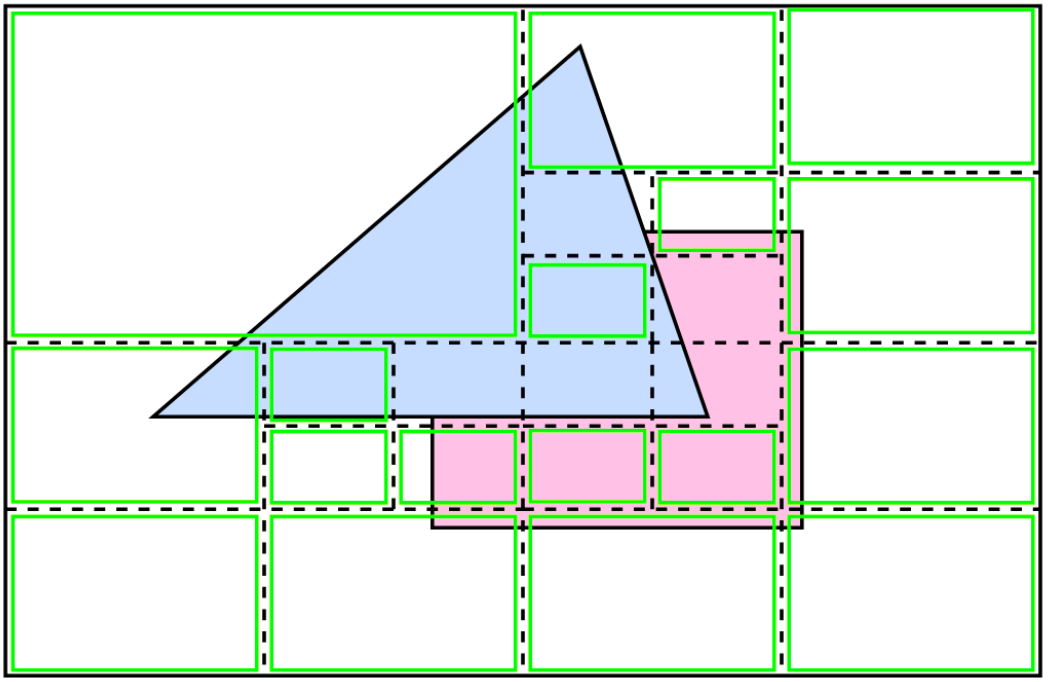

Рисунок 1.2 - Разбиение окна алгоритмом Варнока

В алгоритме Варнока и его вариациях делается попытка извлечь преимущество из того факта, что большие области изображения однородны.

1.3.3. Алгоритм Вейлера-Азертона

Алгоритм Вейлера-Азертона позволяет выполнить отсечение невидимой части объекта перед его отрисовкой по контурам загораживающих его объектов. Этот прием позволяет не делать лишнюю работу при отрисовке частично видимых объектов. На рисунке 1.3 показано разбиение многоугольников.

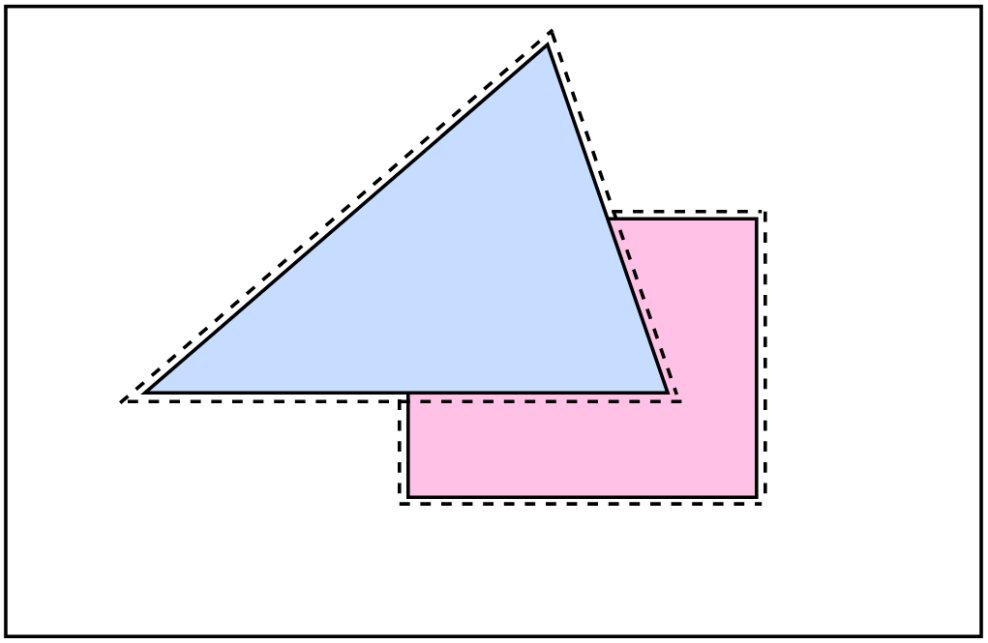

Рисунок 1.3 - Разбиение многоугольников по алгоритму Вейлера-Азертона

1.3.4. Алгоритм с использованием Z-буфера

Алгоритм с использованием Z-буфера базируется на том, что вместо того, чтобы определять очередность в которой должны отрисовываться объекты, для каждого пикселя при отрисовке сохраняется дополнительный атрибут - глубина, использование которого гарантирует корректное отображение пикселя, соответствующего наиболее близкому объекту.

1.3.5. Алгоритм обратной трассировки лучей

Алгоритм обратной трассировки лучей базируется на физическом представлении света и законах геометрической оптики. Он позволяет с высокой степенью точности моделировать различные оптические эффекты.

1.3.6. Вывод

Для программной реализации отображения сцены на экран было решено выбрать алгоритм трассировки лучей как основной для создания реалистического изображения, а при временной отрисовке использовать алгоритм Z-буфера. Сравнение алгоритмов растеризации приведено в таблице 1.1.

Таблица 1.1 - Сравнение алгоритмов растеризации

Алгоритм	**Необходимость предварительной сортировки объектов**	**Объем необх. доп. памяти**[1]	**Возможность параллельного выполнения**
Художника	есть	O(1)	нет
Варнока	есть	O(1)	есть
Вейлера-Азертона	есть	O(N)	нет
Z-буфера	нет	O(p)	нет
Обратная трассировка лучей	нет	O(1)	есть

1.4. Методы закраски

При использовании алгоритмов, основанных на закрашивании полигонов, следует подобрать модель освещения в сцене, т.к. это напрямую влияет на изменение цвета в пределах одного полигона.

1.4.1. Простая закраска

При реализации простой закраски для каждой отдельно взятой грани объекта используется информация об источниках света для расчета цвета, в который впоследствии закрашивается эта грань. Простая закраска хорошо подходит для граненых объектов, которым не требуется сглаживание.

1.4.2. Закраска по Гуро

Закраска по Гуро учитывает кривизну поверхности и использует бикубическую интерполяцию для расчета цвета очередного пикселя. Интерполируемым значением является цвет граничных пикселей. Данный вид закраски позволяет с достаточной степенью точности изображать гладкие кривые поверхности [4].

[1] N - число объектов в сцене, p - число пикселей растра

1.4.3. Закраска по Фонгу

Закраска по Фонгу также учитывает искривленные поверхности, но за счёт интерполяции не цвета пикселя, а вектора нормали, для которого и происходит перерасчет значения интенсивности цвета пикселя. Такой метод позволяет гораздо точнее учитывать блики на поверхности объектов, а также позволяет корректно обрабатывать случаи, когда источник света расположен вблизи освещаемого объекта [4].

1.4.4. Вывод

Для закраски полигонов было решено реализовать все три метода и предоставить пользователю возможность изменять метод закраски в процессе работы программы. Простой метод закраски может быть использован для быстрого, но менее реалистичного синтеза изображения, в то время как с помощью остальных методов закраски изображение синтезируется значительно медленнее, но имеет более реалистичный вид.

2. Конструкторский раздел

Данный раздел посвящен конструированию приложения, а также описанию алгоритмов и методов, которые будут применяться в разрабатываемой программе.

2.1. Требования к программе

Создаваемая программа должна предоставлять следующий набор функций:

- Загрузка заготовленных полигональных моделей из файлов
- Добавление в сцену объектов из заготовленного набора примитивов
- Независимое трансформирование положения объектов в сцене
- Навигация по сцене с помощью перемещения и поворота главной камеры
- Просмотр и изменение специфических свойств объектов в сцене
- Сохранение текущего изображения сцены в файл на диске
- Сохранение созданной сцены в файл на диске
- Загрузка готовой сцены из файла

2.2. Описание основных алгоритмов

В данном подразделе будут приведены описания используемых алгоритмов с представлением соответствующих схем алгоритмов.

2.2.1. Алгоритм визуализации трехмерной сцены с использованием Z-буфера

1. Инициализировать буфер кадра и z-буфер

2. Цикл по всем объектам в сцене

2.1. Цикл по всем полигонам объекта

2.1.1. Определение массива видимых частей полигона

2.1.2. Если массив видимых частей пуст, то переход к 2.1, иначе переход к 2.1.3

2.1.3. Цикл по видимым частям полигона

2.1.3.1. Отобразить часть полигона в буфер

2.4. Отобразить буфер кадра на экран

3. Конец

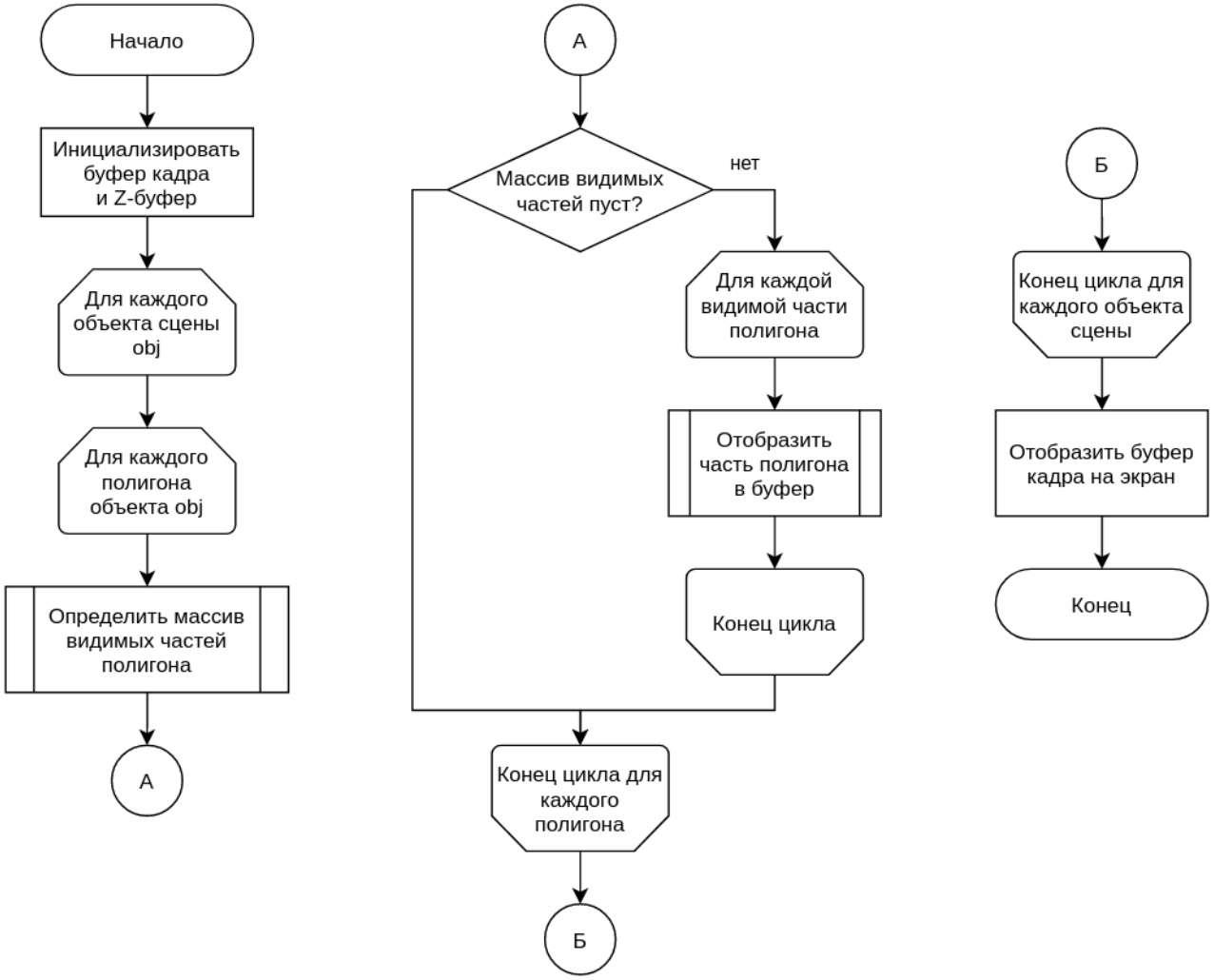

Рисунок 2.1 - Схема алгоритма визуализации сцены с использованием Z-буфера

2.2.2. Алгоритм отображения части полигона в буфер

1. Рассчитать xLeft, xRight - точки пересечения левой и правой границ полигона с первой сканирующей строкой, пересекающей полигон

2. Рассчитать приращения dxLeft, dxRight

3. Цикл по y от yStart до yEnd

3.1. Рассчитать глубину z левой точки пересечения и приращение dz

3.2. Цикл по x от xLeft до xRight

3.2.1. Рассчитать color(x, y) - цвет точки полигона (x, y)

3.2.2. zPrev := значение в Z-буфере по индексу (x, y)

3.2.3. Если z < zPrev, то переход к 3.2.4, иначе переход к 3.3

3.2.4. Буфер кадра (x, y) := color

3.2.5. Z-буфер (x, y) := z

3.3. xLeft := xLeft + dxLeft

3.4. xRight := xRight + dxRight

3.5. z := z + dz

4. Конец

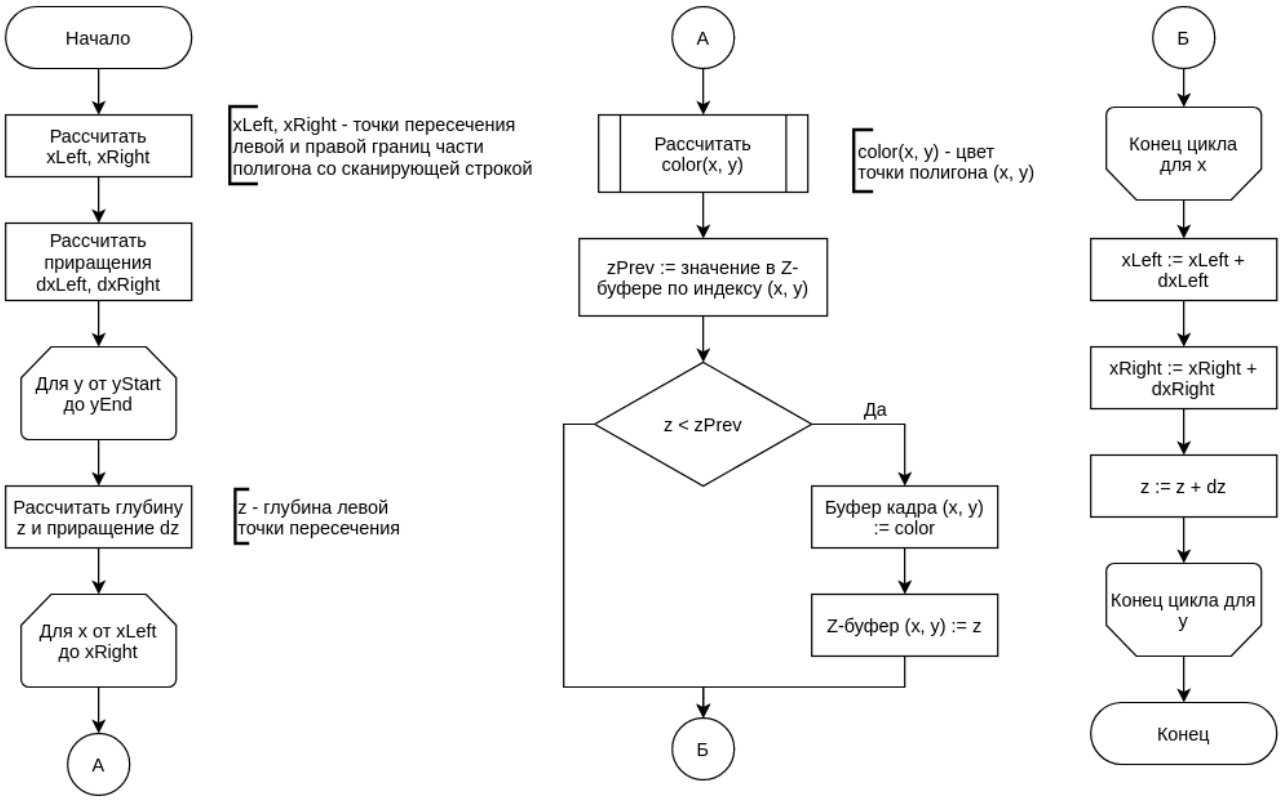

Рисунок 2.2 - Схема алгоритма отображения части полигона в буфер

2.2.3. Алгоритм обратной трассировки лучей

Для всех пикселей на экране следует выполнять следующие действия:

1. Вычислить параметры луча для текущего пикселя

2. Цикл по всем видимым объектам в сцене

2.1. Если луч пересекает объект, то переход к 2.2, иначе переход к 2.6

2.2. Вычислить точку пересечения луча с объектом

2.3. Если эта точка ближе ближайшей, то переход к 2.4, иначе переход к 2.6

2.4. Присвоить ближайшему объекту текущий объект

2.5. Присвоить ближайшей точке текущую точку пересечения

2.6. Конец цикла

3. Если ближайший объект найден, то переход к 4, иначе переход к 5

4. Вычислить цвет пикселя согласно глобальной модели освещения

5. Конец

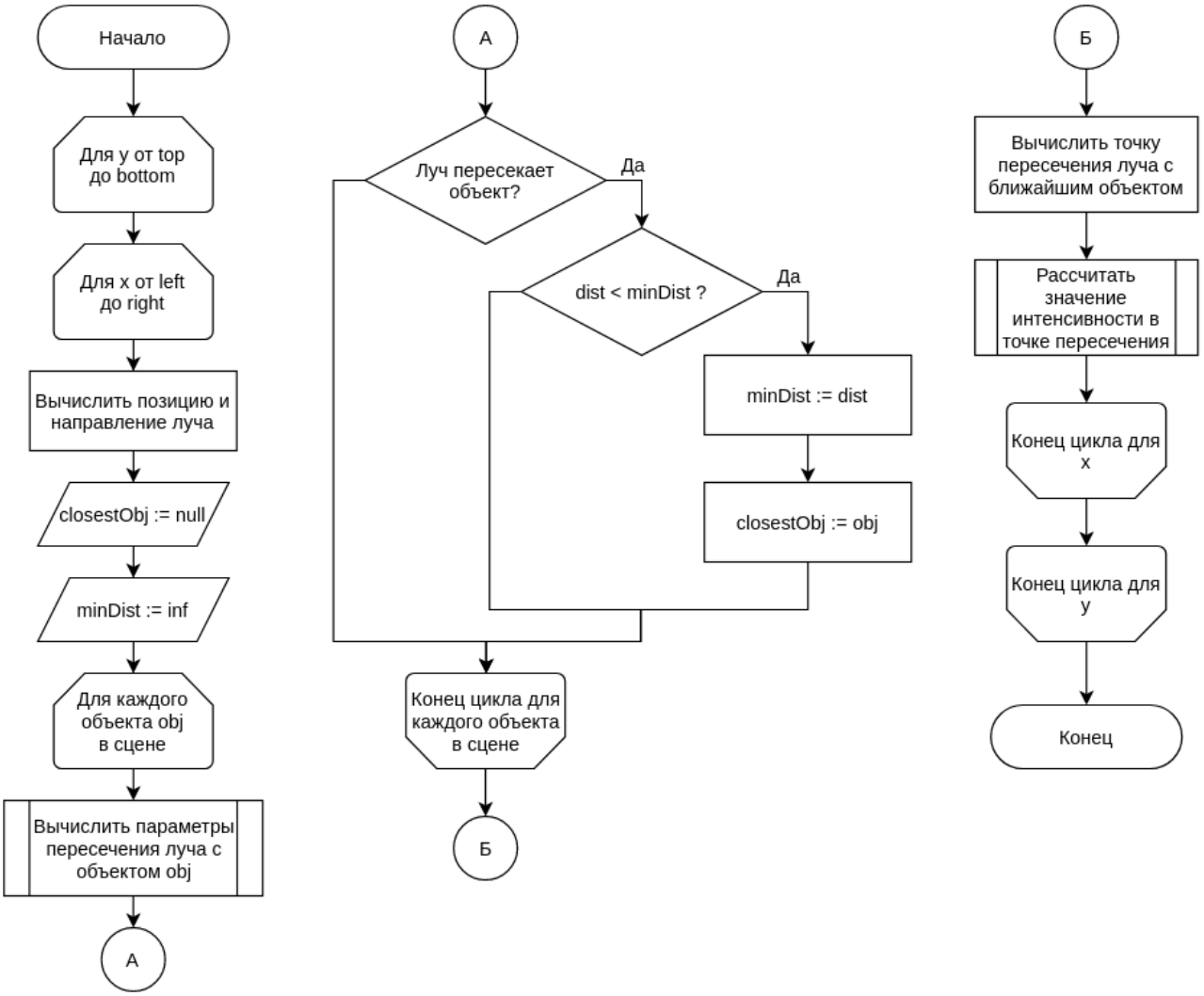

Рисунок 2.3 - Схема алгоритма трассировки лучей

2.2.3. Метод освещения

Для реализации освещения в сцене будет использоваться глобальная модель освещенности, а для создания теней - алгоритм трассировки лучей от поверхности отображаемого полигона до всех источников освещения.

2.3. Выбор используемых типов и структур данных

Математические абстракции:

- Четырехкомпонентный вектор
- Матрица 4x4
- Луч. Состоит из двух векторов - начала и направления
- Прямоугольная двухмерная область

Примитивы растеризации:

- Пиксель (в формате ARGB32)
- Цель отрисовки. Содержит буфер из пикселей, в который происходит отрисовка, и размеры окна - ширина и высота.

Сцена и объекты сцены:

- Вершина полигона. Состоит из позиции и вектора нормали.
- Полигон. Состоит из трех вершин полигона.
- Полигональная модель. Состоит из массива полигонов.
- Источник света. Содержит тип (точечный/направленный), вектор положения/направления, интенсивность и цвет.
- Камера. Содержит матрицу проецирования на экранную плоскость.
- Сцена из объектов. Состоит из списка полигональных моделей и списка источников света.

3. Технологический раздел

Данный раздел посвящен выбору средств для программной реализации разработанного метода, описанию основных моментов разработки программного обеспечения.

3.1. Выбор языка программирования и среды разработки

Для написания программы в качестве языка программирования был выбран язык C++ по следующим причинам:

1. Имеет статическую типизацию, является компилируемым языком программирования;
2. Позволяет напрямую работать с памятью, а также контролировать все обращения к менеджеру памяти;
3. Имеет мощный механизм шаблонов;
4. Позволяет писать программы как в структурном стиле, так и в объектно-ориентированном;

В качестве среды разработки была выбрана “CLion” по следующим причинам:

1. Кроссплатформенность среды и разрабатываемого ПО;
2. Данная среда разработки использует утилиту CMake для сборки проектов;

3.2. Схема доменов

Программа состоит из трех доменов: прикладного, архитектурного и интерфейсного. Прикладной домен отвечает за процесс визуализации сцены, интерфейсный - за взаимодействие с пользователем, а архитектурный служит связующим звеном между двумя вышеупомянутыми доменами.

Рисунок 3.1 - Схема взаимодействия доменов

3.3. Прикладной домен

Прикладной домен реализуется в структурном стиле, так как такой подход обеспечивает наибольшую скорость работы и простоту реализации, а также в дальнейшем не нуждается в частых модификациях.

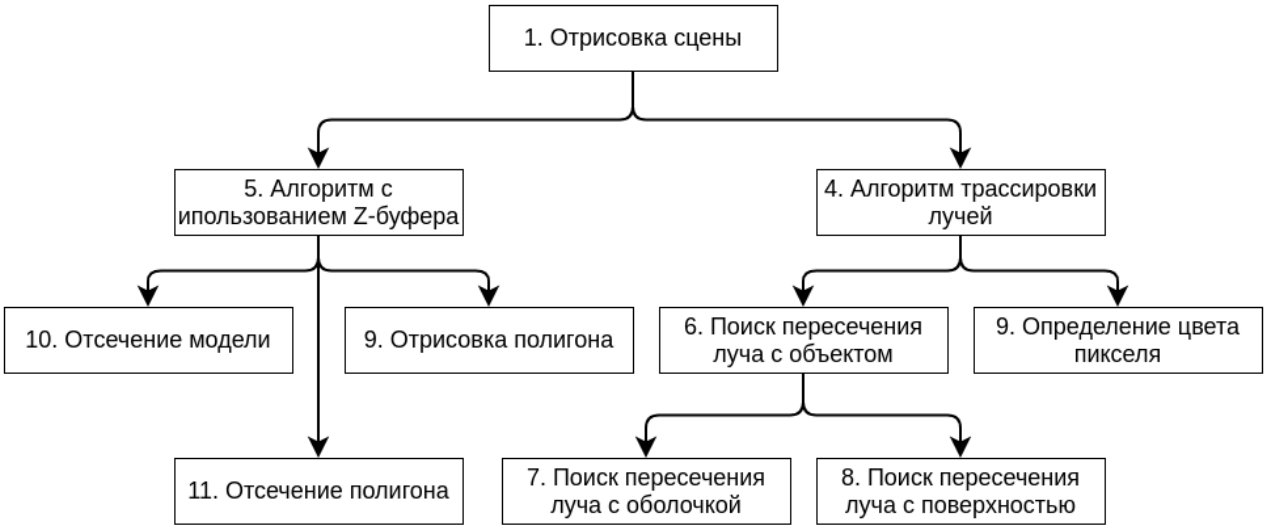

Рисунок 3.2 - Иерархия функций прикладного домена

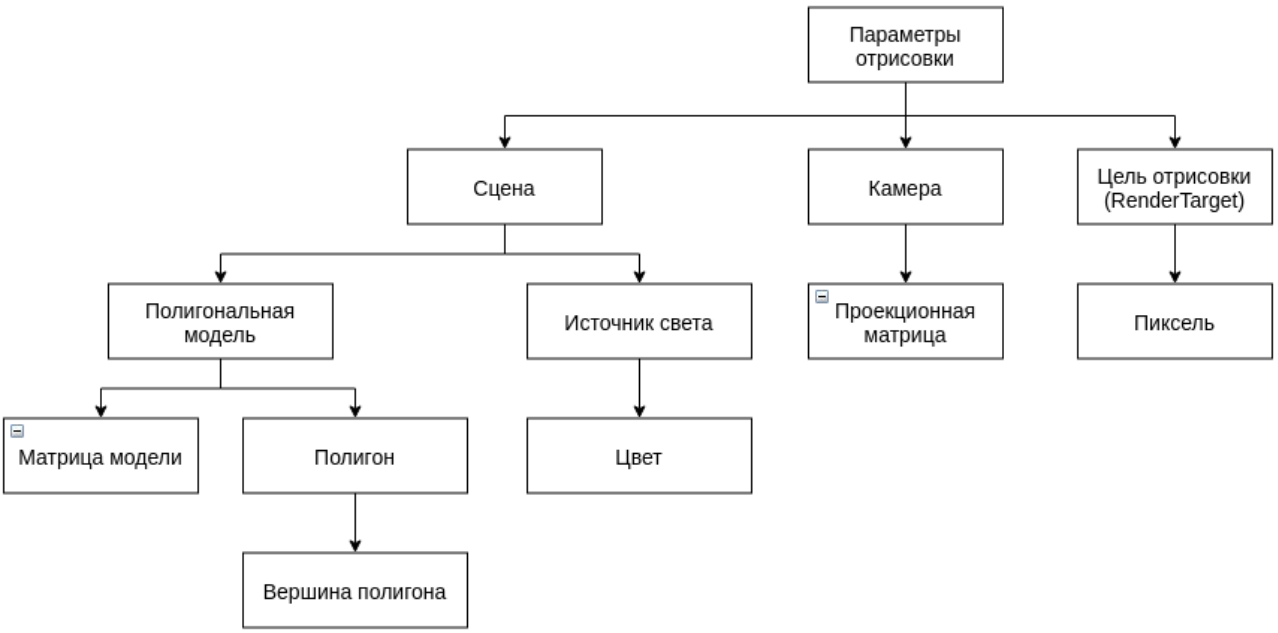

Рисунок 3.3 - Иерархия по данным

3.4. Архитектурный домен

Архитектурный домен реализуется согласно объектно-ориентированному подходу с использованием паттернов проектирования. Информационная модель архитектурного домена представлена на рисунке 3.4.

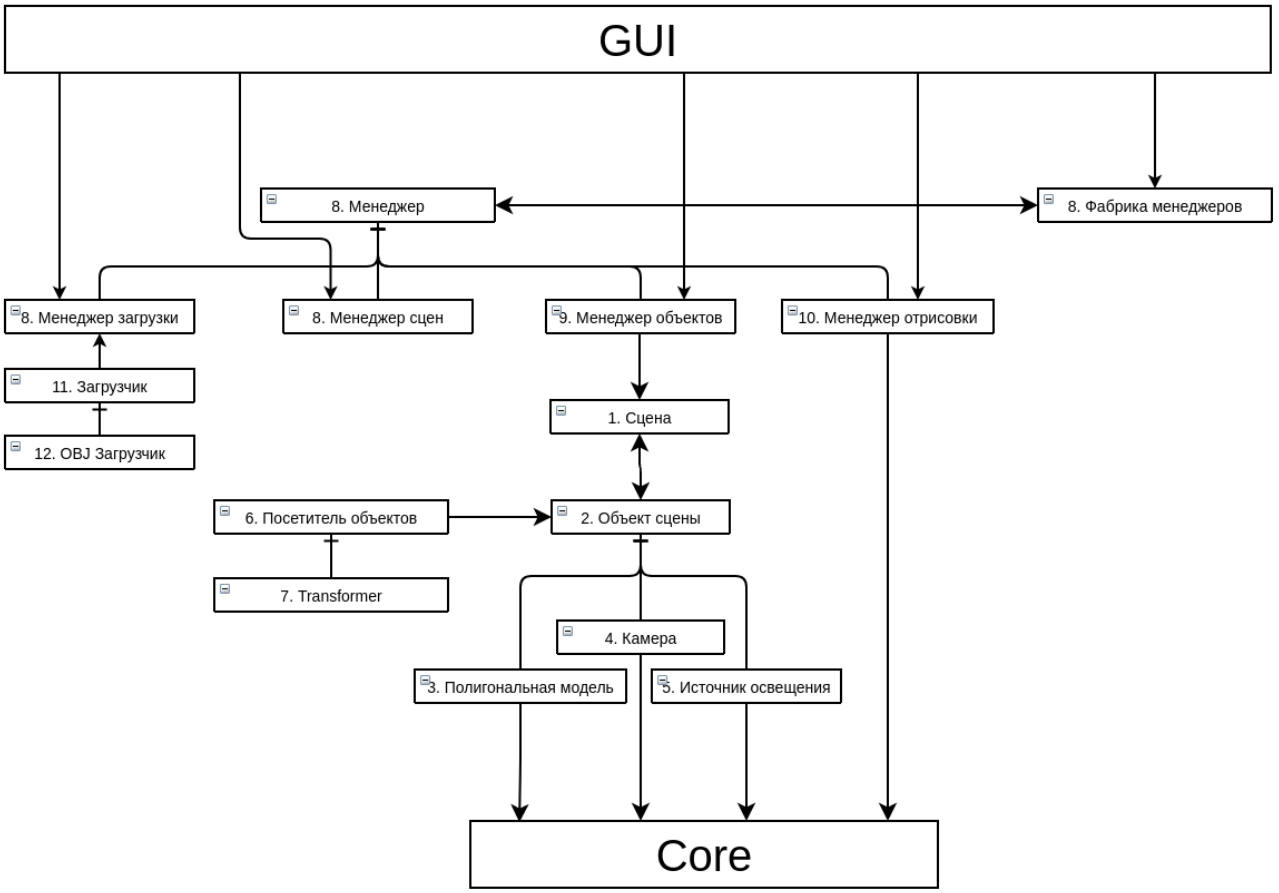

Рисунок 3.4 - Информационная модель архитектурного домена

3.5. Интерфейсный домен

Интерфейсный домен реализуется с использованием открытой библиотеки Qt. Результат разработки программного обеспечения представлен в приложении А.

ЗАКЛЮЧЕНИЕ

В ходе работы был выполнен анализ существующих алгоритмов построения трехмерного изображения и алгоритмов, используемых в разрабатываемом программном обеспечении. Были выдвинуты более подробные требования к программному обеспечению, обоснован выбор инструментов разработки. В основной части были выбраны типы и структуры данных, выделены математические абстракции и описано представление объекта сцены. Была продумана архитектура приложения.

СПИСОК ЛИТЕРАТУРЫ

1. Colin Smith, On Vertex-Vertex Meshes and Their Use in Geometric and Biological Modeling [Электронный ресурс] // algorithmic botany. URL: http://algorithmicbotany.org/papers/smithco.dis2006.pdf. (Дата обращения: 13.08.2021)
2. Рендеринг объемов в реальном времени: [Электронный ресурс] // Открытые системы СУБД. URL: https://www.osp.ru/os/1996/05/178968 (Дата обращения: 13.08.2021)
3. Алгоритм Варнока: [Электронный ресурс] // Компьютерная графика - А. Ю. Дёмин, А. В. Кудинов. URL: http://compgraph.tpu.ru/warnock.htm (Дата обращения: 14.08.2021)
4. Модели затенения. Плоская модель. Затенение по Гуро и Фонгу: [Электронный ресурс] // Компьютерная графика теория, алгоритмы, примеры на C++ и OpenGL. URL: http://compgraphics.info/3D/lighting/shading_model.php (Дата обращения: 18.08.2021)

ПРИЛОЖЕНИЕ А

Интерфейс реализации программного обеспечения

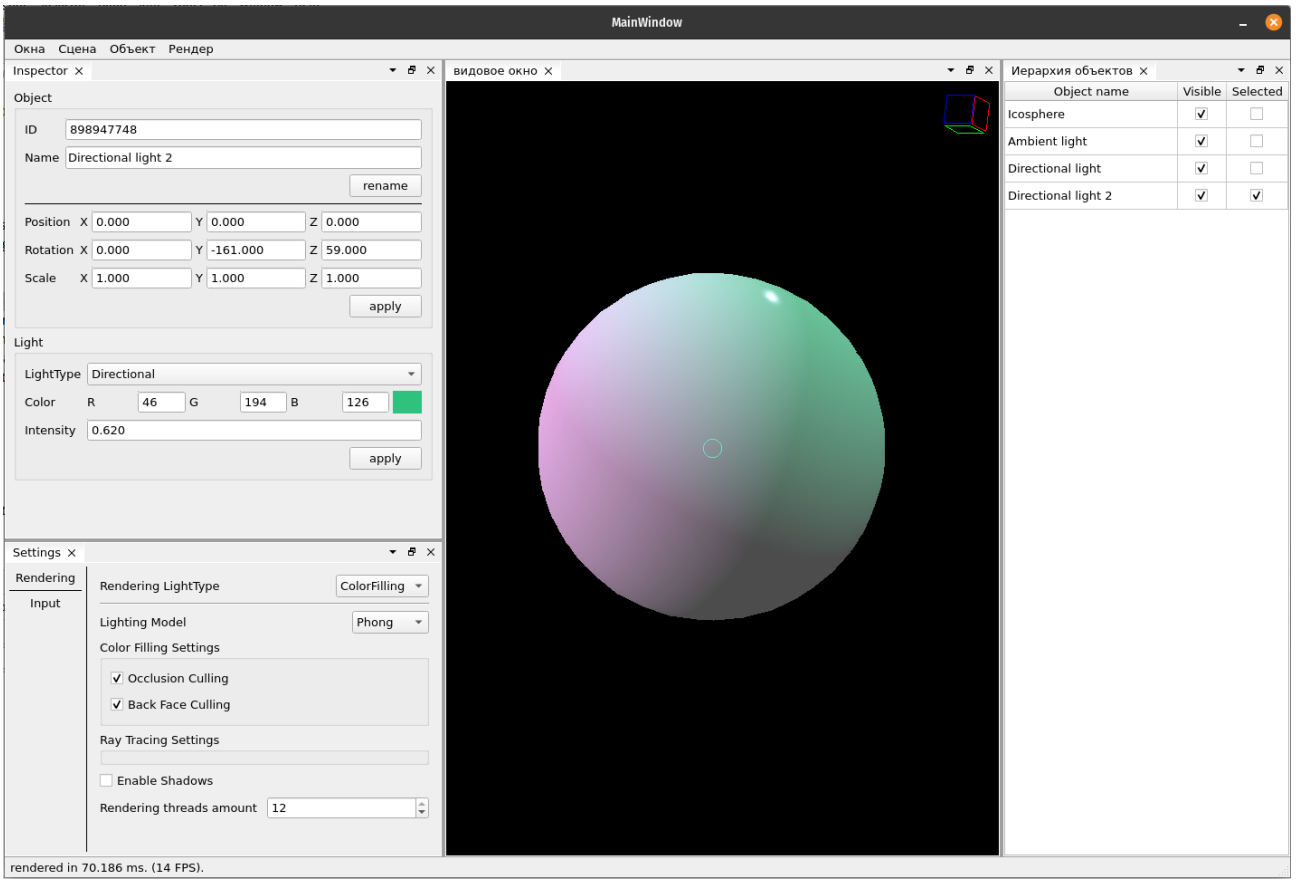

Рисунок А.1 - Изображение сферы с использованием закраски по Фонгу. В сцене установлено три источника освещения

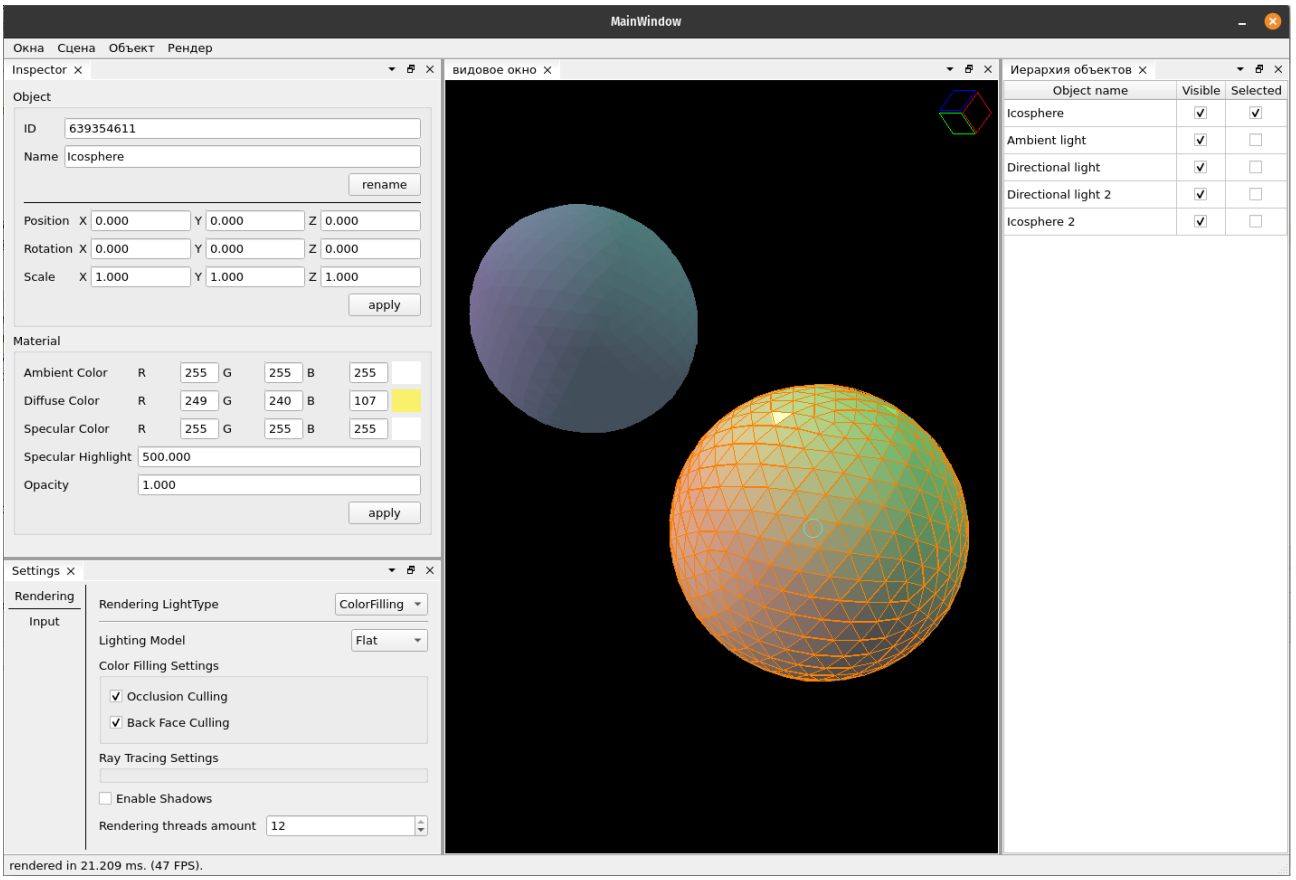

Рисунок А.2 - Режим редактирования параметров объектов. Использование однотонной закраски для каждого полигона

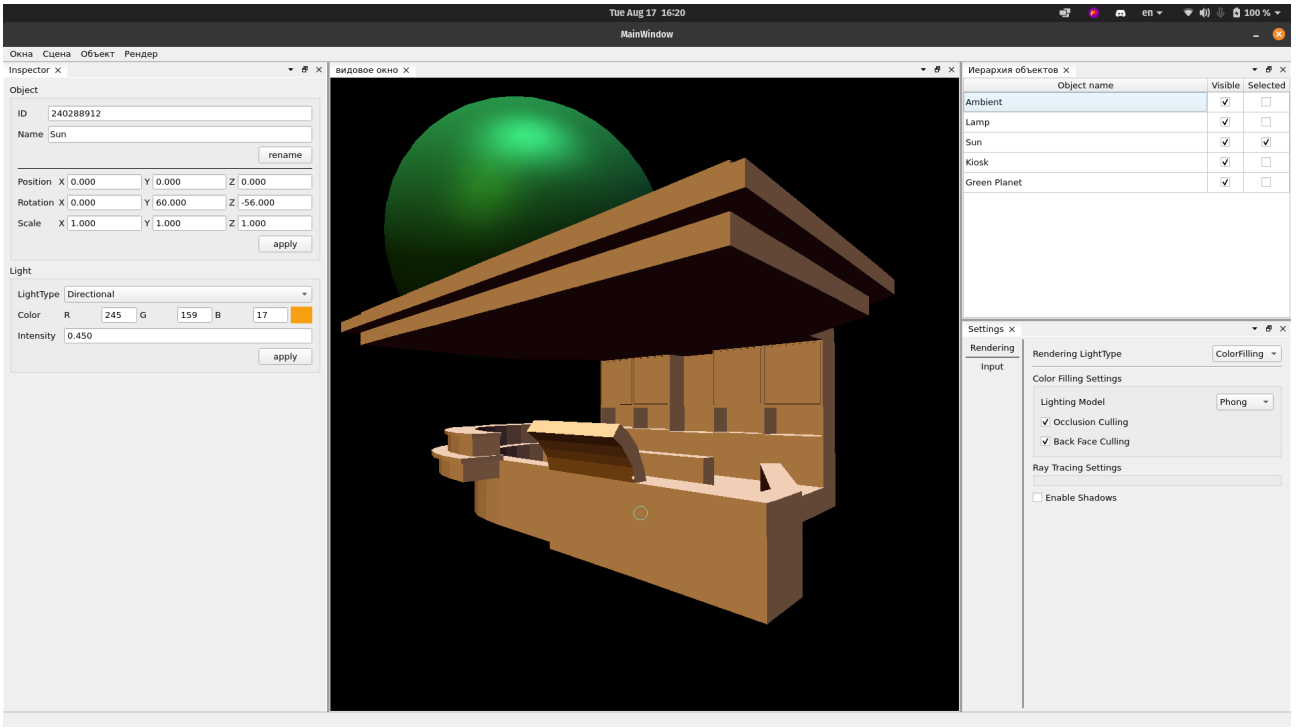

Рисунок А.3 - Изображение загруженной модели

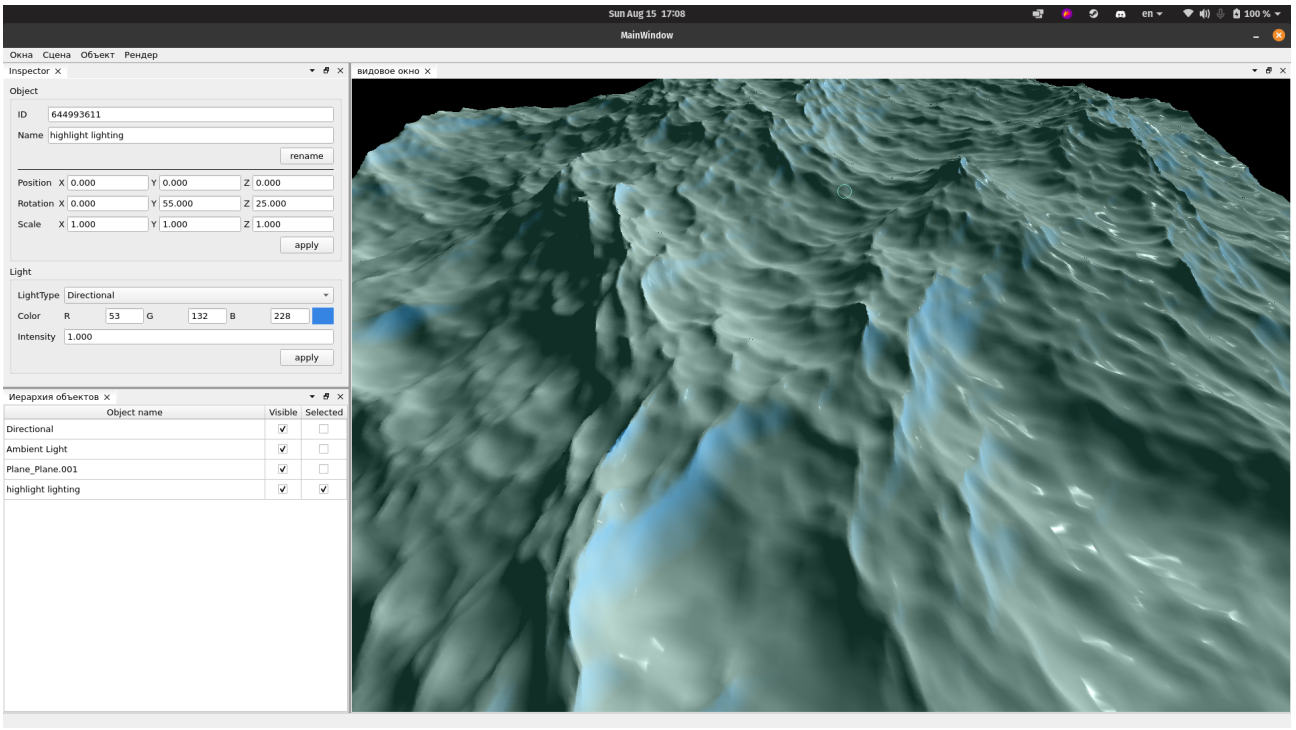

Рисунок А.4 - Синтез изображения высокополигональной модели. Поверхность воды