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

«Программа для визуализации модели солнечной системы»

Студент Маслюков П. В.

Руководитель курсовой работы Павельев А. .

Москва — 2023 г.

ДЛЯ ТЗ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютерное моделирование используется в различных сферах, от научных исследований и инженерных разработок до экономического анализа и проектирования городской инфраструктуры. Компьютерные модели позволяют создавать симуляции и визуализации сложных процессов и событий. Использование компьютерных моделей облегчает и упрощает их исследование благодаря возможности проведения вычислительных экспериментов. Это особенно полезно, когда проведение реальных экспериментов осложнено финансовыми, физическими ограничениями или может дать непредсказуемые результаты

Целью курсовой работы является разработка программы, которая будет позволять визуализировать модель солнечной системы. Для достижения поставленной цели требуется выполнить следующие задачи:

- выбрать алгоритмы трехмерной графики, необходимые для визуализации модели солнечной системы;
- реализовать выбранные алгоритмы;
- описать и разработать программное обеспечение для визуализации модели;
- исследовать зависимость производительности от количества вершин сферы.

1 Аналитическая часть

В данном разделе будут формализованы объекты сцены и описаны алгоритмы трехмерной графики, необходимые для поставленной задачи.

1.1 Формализация объектов сцены

Сцена состоит из определенного набора объектов.

1. Объекты сцены – звезда и планеты солнечной системы.
2. Камера характеризуется своим пространственным положением и направлением просмотра.

1.2 Способ задания трехмернерных моделей

Существуют разные способы задания трехмерных моделей:

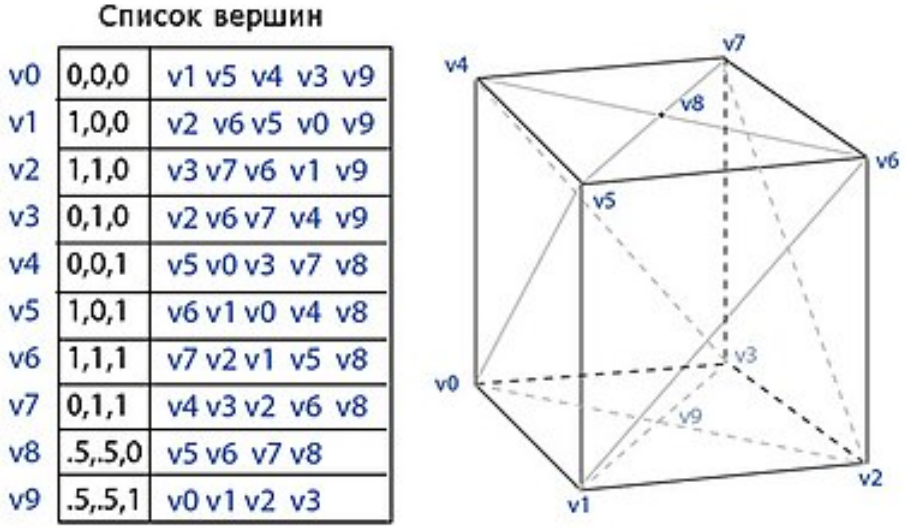
- Сеточная модель - модель, созданная из сетки вершин и ребер, которые определяют форму объекта. Эта модель может быть использована для создания простых геометрических форм или сложных объектов;
- Поверхностная модель - модель, которая включает в себя информацию о поверхности объекта, такую как текстуры, цвета и материалы. Эти модели обычно используются для создания реалистичных изображений и визуализации объектов[1];
- Твердотельная модель - модель, в которой к информации о поверхностях добавляется информация о том, с какой стороны расположен материал. Это достигается путем указания направления внутренней нормали.

Для решения поставленной задачи не подойдет сеточная модель, так как такое представление может приводить к неправильному восприятию форм объекта, также эта модель представления не применима к сферическим объектам. Твердотельная модель также не подойдет, так как по поставленной задачи нет необходимости знать из какого материала будет выполнен тот или иной объект и с какой стороны расположен материал. Поэтому для решения поставленной задачи была выбрана поверхностная модель.

1.2.1 Алгоритм построения поверхностной модели

Поверхностной моделью называется совокупность ограничивающих модель поверхностей, называемых полигонами[1]. Существует разные способы хранения информации о полигонах объекта:

- Вершинное представление описывает объект как множество вершин, соединённых с другими вершинами. На рисунке 1.1 приведен пример вершинного представления.



Список вершин		
v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,0	v5 v6 v7 v8
v9	.5,.5,1	v0 v1 v2 v3

Рисунок 1.1 – Вершинное представление

- Список граней представляет объект как множество граней и вершин,

составляющих грань. На рисунке 1.2 приведен пример представления объекта с использованием списка граней.

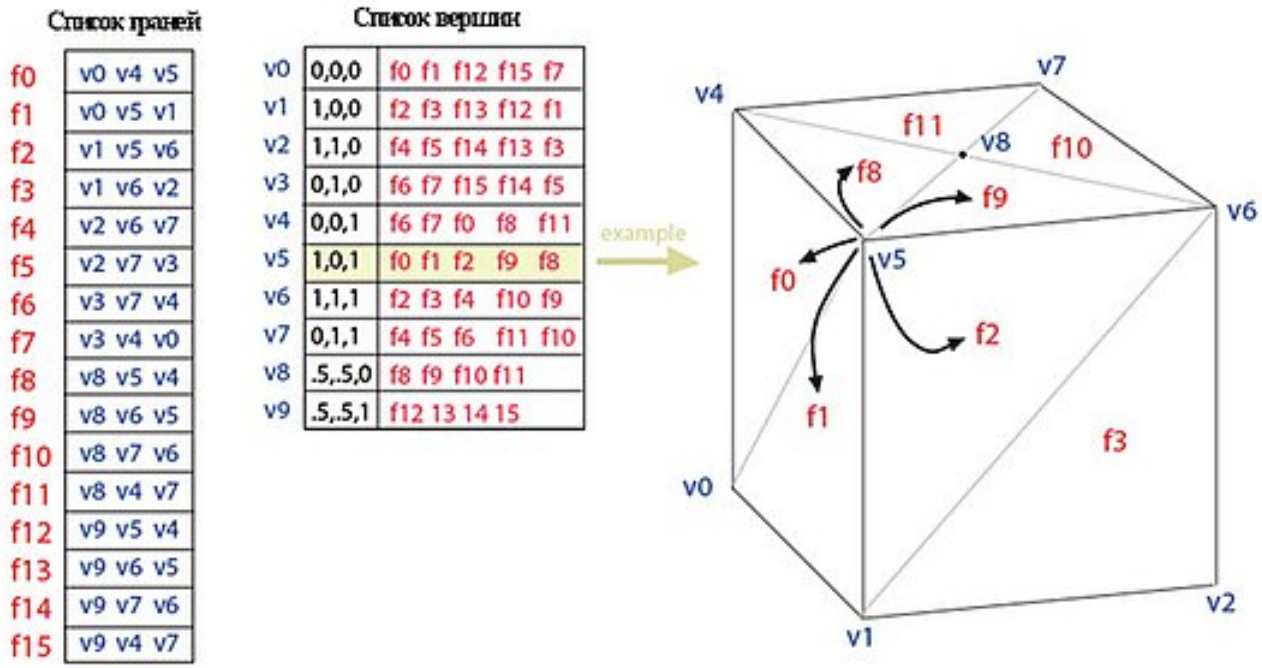


Рисунок 1.2 – Представление списком граней

- "Крылатое"представление явно хранит информацию о вершинах, гранях и рёбрах объектов. На рисунке 1.3 приведен пример "крылатого"представления.

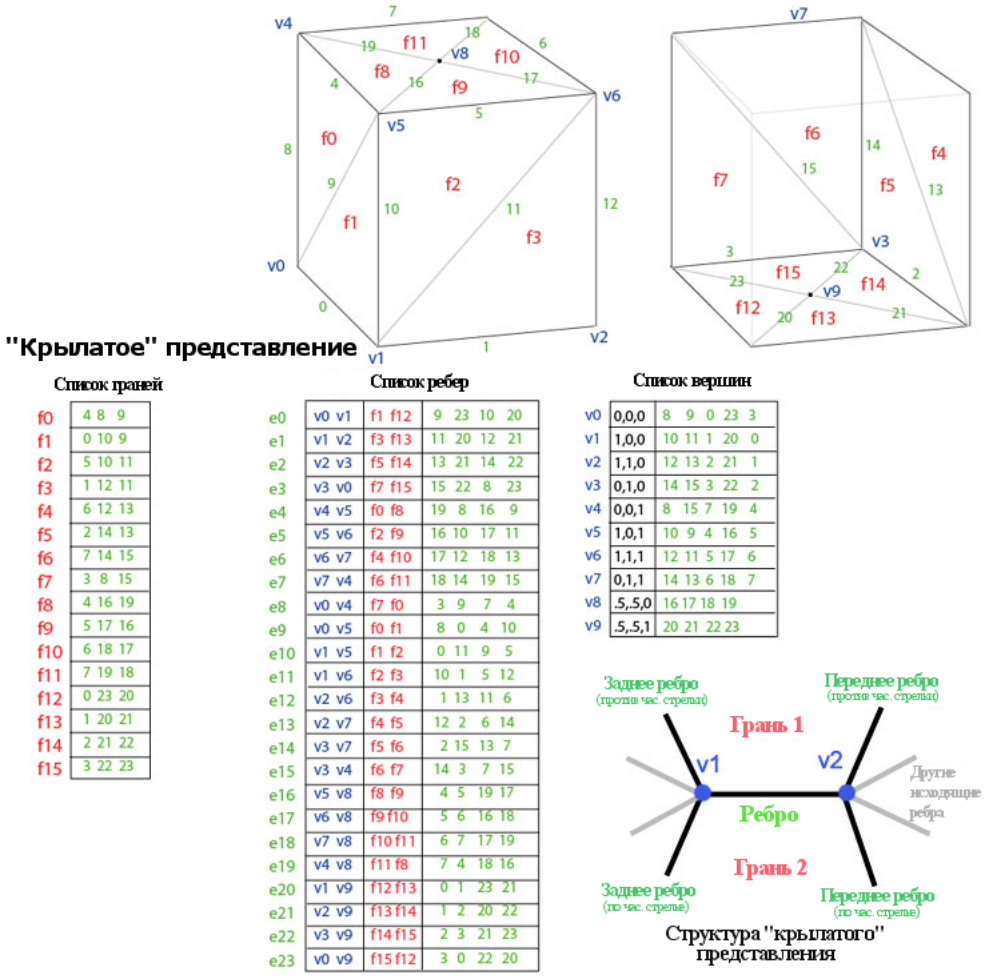


Рисунок 1.3 – "Крылатое"представление

Вершинное представление не подходит для задачи визуализации планетарной системы так как для построения изображения, используя вершинное представление необходимо обойти все вершины, что значительно затрудняет работу и увеличивает время выполнения.

Наиболее удобным способом построения поверхностной модель является список граней т.к. в представленной задачи нет необходимости хранить информацию о ребрах объектов.

1.3 Анализ алгоритмов удаления невидимых ребер

Основная задача при создании реалистичного изображения состоит в удалении объектов или их частей, которые перекрываются другими

объектами и становятся невидимыми для наблюдателя.

1.3.1 Алгоритм Робертса

Идея алгоритма состоит в том, чтобы для каждого полигона проверить, видим он или нет с учетом его местоположения и ракурса наблюдателя[2]. Для этого алгоритм выполняет следующие шаги:

- Сортирует полигоны сцены по глубине;
- Для каждого полигона определяет, виден он или нет. Для этого сравнивает его глубину с глубиной других полигонов в сцене. Если полигон перекрывается другим полигоном, то он считается невидимым, и его рёбра удаляются;
- Повторяет предыдущий шаг для каждого полигона в сцене;
- Отображает видимые полигоны на экране.

Однако алгоритм Робертса имеет свои ограничения. Например, если два полигона находятся на одной глубине, то может возникнуть проблема их обратного удаления. Также алгоритм требует большого объема вычислительной работы, особенно при большом количестве полигонов в сцене.

1.3.2 Алгоритм Варнока

Основная идея алгоритма Варнока заключается в построении абстрактной прямоугольной сетки над трехмерной сценой[3]. Эта сетка разбивает пространство на регулярные ячейки и позволяет распределить видимые и невидимые ребра на основе их положения внутри ячеек. Алгоритм осуществляет обход графа. Процесс обхода начинается с некоторой стартовой точки сцены и проходит через все ребра, определяя их видимость. На каждом этапе обхода алгоритм выполняет проверку каждого ребра на видимость. Если ребро видимо, оно добавляется в список видимых ребер. Если же ребро невидимо, оно не добавляется в список и не отображается на экране.

Алгоритм Варнока имеет ряд преимуществ, таких как эффективность, возможность работы с большими объемами данных и простота реализации. Однако он также имеет некоторые ограничения, такие как проблемы с обработкой повторных видений, сложность работы с динамическими сценами и сложность визуализации прозрачных объектов.

1.3.3 Алгоритм Трассировки лучей

Главная идея, лежащая в основе алгоритма трассировки лучей, заключается в том, что наблюдатель видит любой объект посредством испускаемого неким источником света, который падает на этот объект и затем согласно законам оптики некоторым путем доходит до глаза наблюдателя[4]. В методе построения траекторий лучей происходит от всех источников света ко всем точкам всех объектов сцены, лучи отражаются и преломляются или проходят сквозь и в результате достигает наблюдателя. Данные лучи называются первичными. Если объект не является отражающим или прозрачным, то траектория луча на этой точке обрывается. Основным недостатком алгоритма является излишне большое число рассматриваемых лучей, приводящее к существенным затратам вычислительных мощностей, так как лишь малая часть лучей достигает точки наблюдения. Данный алгоритм подходит для генерации статических сцен и моделирования зеркального отражения, а так же других оптических эффектов как отражение, преломление и т.д.

1.3.4 Алгоритм, использующий Z-буфер

В основе алгоритма лежит концепция буфера глубины или Z-буфера. Это двумерный массив, количество элементов которого соответствует размерам экрана. Каждый элемент этого массива представляет собой значение глубины определенной точки экрана. В начале работы алгоритма все значения глубины в Z-буфере устанавливаются в максимальное значение, что означает, что ни один объект не находится на экране. При отрисовке каждого объекта алгоритм проверяет каждую точку объекта и сравнивает ее

глубину с значением, хранящимся в соответствующем элементе Z-буфера. Если глубина точки меньше значения в Z-буфере, это означает, что текущая точка находится ближе к наблюдателю и должна отображаться на экране. В таком случае значение глубины буфера обновляется. Процесс проверки точек и обновления буфера глубины выполняется для каждого объекта на сцене.

Алгоритм Z-буфера дает возможность реализовать эффекты перекрывания объектов, при этом сохраняя правильное отображение их глубины на экране[2]. Это обеспечивает реалистичное и естественное представление трехмерных сцен для наблюдателя.

1.3.5 Выбор оптимального алгоритма удаления невидимых ребер

Таблица 1.1 – Сравнение алгоритмов удаления ребер и поверхностей

Характеристика	АВ	АТ	АР	АБ
Сложность алгоритма	$O(S \cdot n)$	$O(n^2)$	$O(n^2)$	$O(S \cdot n)$
Работа со сферами	да	да	да	да
Требуется проверка пересечния лучей с объектами сцены	да	да	да	нет

Обозначения: АВ – алгоритм Варнока, АТ – алгоритм трассировки лучей, АР – алгоритм Робертса, АБ – алгоритм Z-буфера.

Наиболее подходящим алгоритмом является алгоритм с Z-буфером, так как он более эффективен для рендеринга в режиме реального времени, потому что он не требует проверки пересечений лучей со всеми объектами сцены. Данный алгоритм имеет линейную зависимость от числа объектов, что приведет к оптимальной работе программы.

1.4 Анализ методов освещения

В компьютерной графике модель освещения используется для определения того, как свет взаимодействует с объектами 3D-сцены и как они будут выглядеть на экране. Модель освещения включает в себя различные

типы источников света, а также материалы объектов, их отражательные свойства и расчет освещения для определения цвета и яркости каждого пикселя. Главным требованием, выдвигаемым в моей программе к методам освещения является быстродействие.

1.4.1 Модель Фонга

Одним из основных типов моделей освещения является модель Фонга. Она включает в себя три основных компонента: освещение по Фонгу, отражение на постоянном цвете и диффузное отражение[5]. Освещение по Фонгу рассчитывает отраженный свет от источника света и добавляет его к объекту. Отражение на постоянном цвете добавляет равномерный фоновый свет ко всей сцене. Диффузное отражение учитывает разброс света на поверхности объекта в зависимости от угла падения света

На рисунке 1.4 приведена модель Фонга. N — нормаль к поверхности, L — направление света, R — отраженное направление света, V — направление наблюдателя.

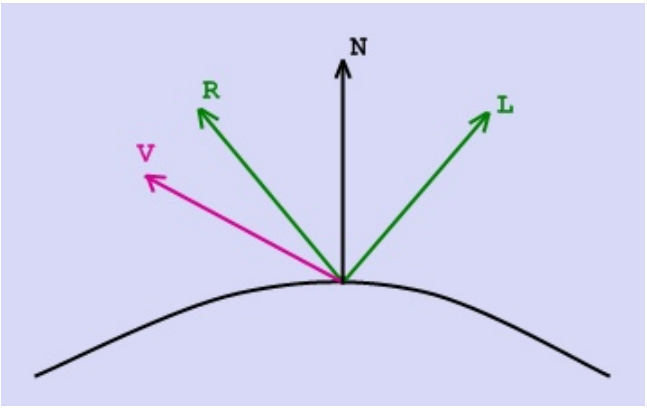


Рисунок 1.4 – Модель Фонга

1.4.2 Модель Ламберта

Еще одна распространенная модель освещения - модель Ламберта. Она учитывает только диффузное отражение света и не учитывает отражение на постоянном цвете или отражение по Фонгу[5]. Поэтому эта модель

используется, когда необходимо выполнить быстрые вычисления освещения.

На рисунке 1.5 приведена модель Ламберта. N — нормаль к поверхности, L — направление света.

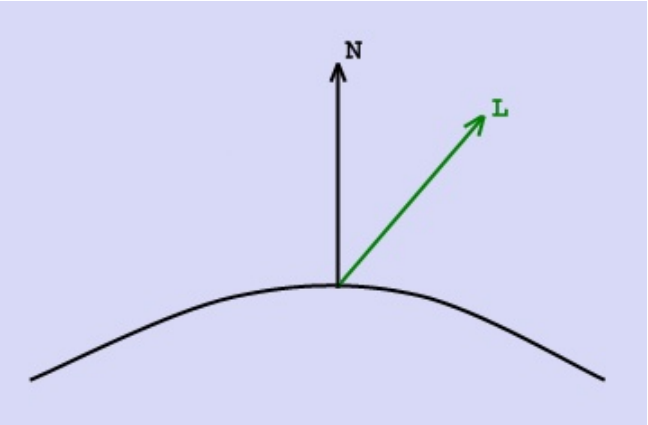


Рисунок 1.5 – Модель Ламберта

1.4.3 Выбор оптимальной модели освещения

В представленной задачи объекты не обладают отражающими свойствами, поэтому было решено выбрать модель освещения Ламберта.

1.5 Анализ методов закраски

В компьютерной графике существует несколько методов закраски, которые используются для заполнения фигур цветом или текстурой.

1.5.1 Закраска Ламберта

Закраска Ламберта основана на моделировании диффузного освещения и принципе Ламберта, который описывает взаимодействие света с материалом. Этот метод используется для достижения реалистичных эффектов без блеска и отражений. Алгоритм закраски Ламберта рассчитывает интенсивность освещения в каждой точке поверхности объекта на основе

угла между нормалью поверхности и направлением света. Чем больше этот угол (то есть чем более поверхность повернута в сторону источника света), тем ярче будет освещение этой точки. Процесс закраски Ламберта прост и вычислительно эффективен.

На рисунке 1.6 приведен пример закраски Ламберта.

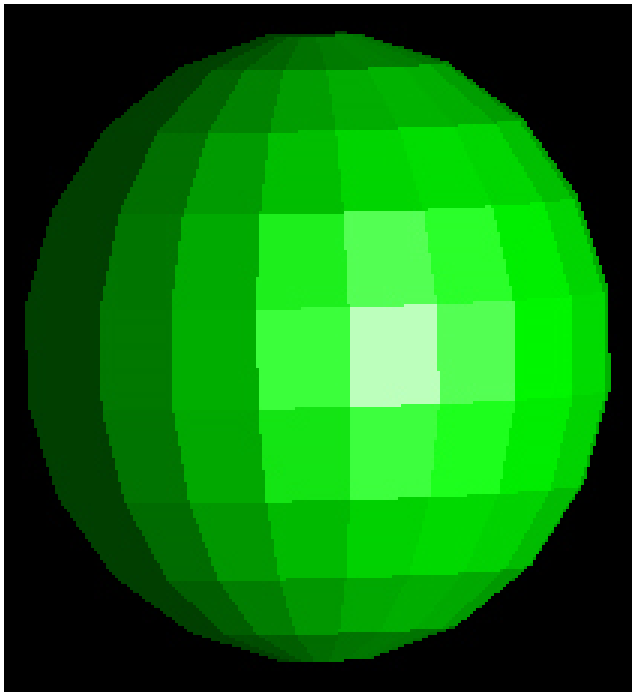


Рисунок 1.6 – Закраска Ламберта

1.5.2 Закраска по Фонгу

Закраска по Фонгу использует модель освещения, состоящую из трех компонентов: диффузного, зеркального и окружающего освещения. Этот метод позволяет достичь более реалистичных результатов, включая блеск и отражения. Алгоритм закраски по Фонгу включает вычисление нормали в каждой точке поверхности объекта и ориентации к наблюдателю источника света. Затем рассчитывается освещение каждой точки с использованием угла между нормалью поверхности и направлением света, а также углом между направлением наблюдения и отраженным лучом [6].

На рисунке 1.7 приведен пример закраски Фонга.

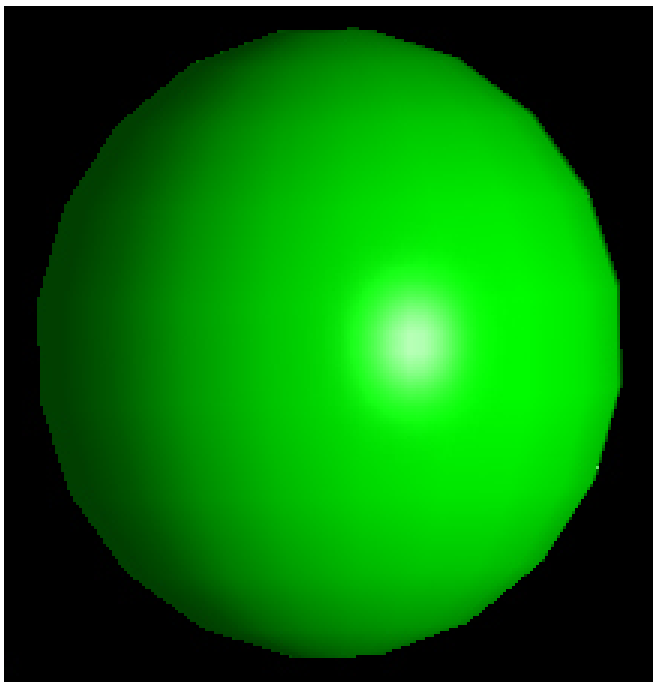


Рисунок 1.7 – Закраска Фонга

1.5.3 Закраска по Гуро

Закраска по Гуро схожа с моделью Фонга, но применяется для закраски вершин многоугольников, а не отдельных точек поверхности объекта. Это делает алгоритм более эффективным по сравнению с алгоритмом Фонга. Алгоритм закраски по Гуро заключается в вычислении интенсивности освещения в вершинах многоугольника. Затем эти значения интенсивности интерполируются между вершинами, чтобы определить интенсивность освещения для каждого из пикселей внутри многоугольника. Этот метод обеспечивает более плавные переходы интенсивности между вершинами и может использоваться для создания более реалистичных и плавных поверхностей[6].

На рисунке 1.8 приведен пример закраски Гуро.

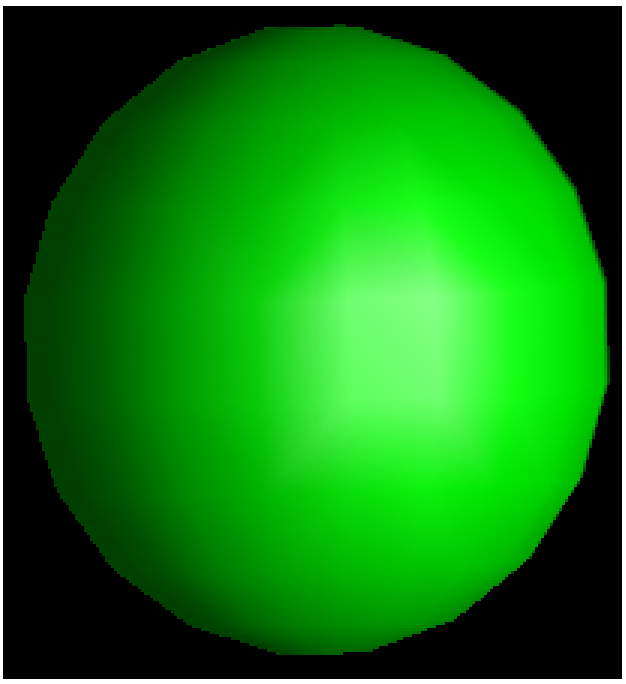


Рисунок 1.8 – Закраска Гуро

1.5.4 Выбор оптимального алгоритма закраски

В таблице 1.2 приведено сравнение алгоритмов закрашивания полигонов.

Таблица 1.2 – Сравнение алгоритмов закрашивания полигонов

Характеристика	ЗЛ	ЗГ	ЗФ
Учёт зеркальных и матовых бликов	нет	да	да
Высокие вычислительные затраты	нет	да	да
Применимость к произвольным выпуклым объектам	да	да	да

Обозначения: ЗЛ – алгоритм закраски Ламберта, ЗГ – алгоритм закраски по Гуро, ЗФ – алгоритм закраски по Фонгу.

Так как главным критерием для выбора алгоритма закраски является быстродействие, для решения поставленной задачи был выбран алгоритм закраски Ламберта.

1.6 Описание движения планет

Для визуализации анимации движения необходимо рассчитать скорость движения планет, а также изменение их координат. Для этого необходимо воспользовать законом всемирного тяготения.

$$F_i^t = \sum_{j=0}^{n} G \cdot \frac{m_i \cdot m_j}{R_{ij}^2}, j \neq i, \tag{1.1}$$

где n – количество объектов, m – масса объекта, R_{ij} – расстояние между объектами[7].

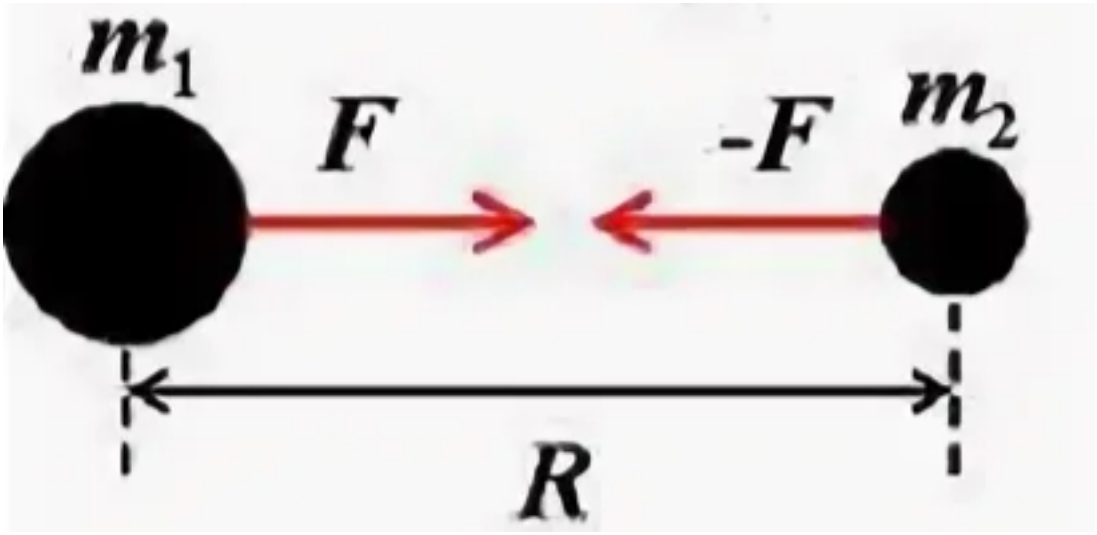


Рисунок 1.9 – Иллюстрация взаимодействия двух тел

Скорость движения планет считается по формуле (1.2).

$$\vec{v_i^t} = \frac{F_i^{t-1} \cdot dt}{m_i}, \tag{1.2}$$

где m – масса объекта, F – сила тяготения, dt – временная разница.

Из этой формулы можно вывести формулу для рассчета координат следующего положения планеты.

$$\vec{r_i^t} = \vec{v_i^{t-1}} \cdot dt + \vec{r_i^{t-1}}. \tag{1.3}$$

1.7 Вывод

В этом разделе было исследовано предметное направление, были изучены алгоритмы, используемые для удаления невидимых граней, методы освещения и методы закрашивания поверхностей. Также был описан и обоснован выбор используемых методов и алгоритмов. Для решения поставленной задачи была выбрана поверхностная модель со списком граней, алгоритм Z-буфера для удаления невидимых рёбер и поверхностей, а также метод закраски Ламберта.

2 Конструкторская часть

В данном разделе будет описан общий алгоритм решения задачи, представлены схемы используемых алгоритмов. Также будут описаны используемые типы и структуры данных.

2.1 Алгоритм решения поставленной задачи

На рисунке 2.1 приведена функциональная модель визуализации движения планет солнечной системы.

Рисунок 2.1 – Диаграмма IDEF0 нулевого уровня

На вход поступают масса, радиус, скорости планет, анимацию вращения которых необходимо визуализировать. Используя выбранные алгоритмы закраски, удаления невидимых ребер и модель освещения строится растровая анимация движения планет.

2.2 Схемы алгоритмов

На рисунке 2.2 приведена общая схема решения.

Рисунок 2.2 – Общий алгоритм решения задачи построения изображения

На рисунке 2.3 приведена схема построения поверхностной модели.

Рисунок 2.3 – Алгоритм построения поверхностной модели

На рисунке 2.4 приведена схема алгоритма закраски.

Рисунок 2.4 – Алгоритм закраски Ламберта

На рисунке 2.5 приведена схема алгоритма, использующего Z-буфер.

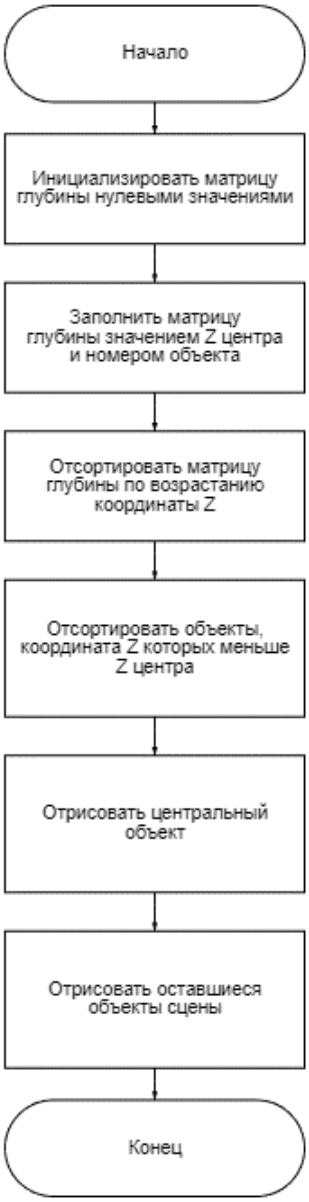


Рисунок 2.5 – Алгоритм использующий Z-буфер

2.3 Описание используемых типов и структур данных

В таблице 2.1 представлены типы и структуры данных, которые будут реализованы для разрабатываемого ПО.

Таблица 2.1 – Используемые типы и структуры данных

Сущность	Структура
Точка в пространстве	Point
Вектор	Vector
Грань с индексами вершин	Edge
Сфера с координатами центра, радиусом, массивом точек	Sphere
Планета с координатами центра, радиусом, цветом и списком интенсивности	Planet
Звезда с координатами центра, радиусо, и списком источников света	Star

2.4 Вывод

В данном разделе был представлены алгоритмы, используемые для решения задачи визуализации солнечной системы, представлены их схемы. Также были описаны используемые типы и структуры данных.

3 Технологическая часть

В данном разделе будут указаны средства реализации, а также описаны основные модули программного обеспечения.

3.1 Средства реализации

Для выполнения поставленной задачи был выбран язык программирования C++, так как данный язык программирования объектно-ориентирован[8], что дает возможность представлять объекты сцены в виде объектов классов используя абстрактные классы и наследование. Также данный язык программирования обеспечивает все необходимые возможности для решения задачи, и имеет фреймворк QT с встроенным графическим редактором QT Design, для создания интерфейса.[9].

3.2 Сведения о модулях программы

Основные модули программы:

- main.cpp - точка входа в приложение;
- planeta.cpp, planeta.h - описание и методы взаимодействия с планетами;
- star.cpp, star.h - описание и методы взаимодействия со звездой;
- sphere.cpp, sphere.h - описание и методы взаимодействия со сферическими объектами;

3.3 Вывод

В данном разделе были указаны средства реализации, а также описаны основные модули программного обеспечения.

4 Исследовательская часть

В данном разделе будут приведены примеры работы программы, а также поставлено исследование по сравнению производительности программы.

4.1 Демонстрация работы программы

На рисунке 4.1 и 4.1 приведены результаты работы программы.

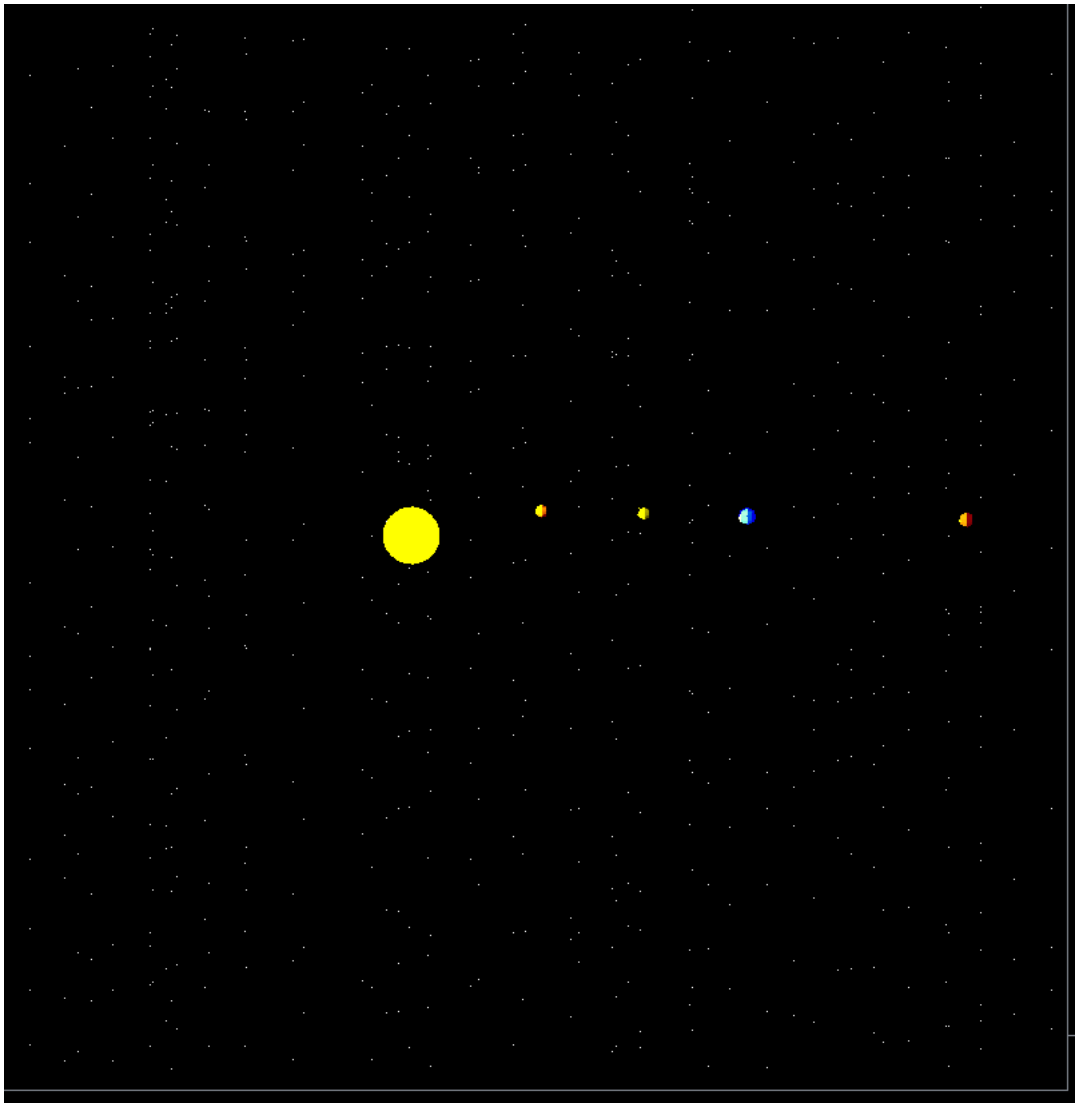


Рисунок 4.1 – Пример работы программы (вид 1)

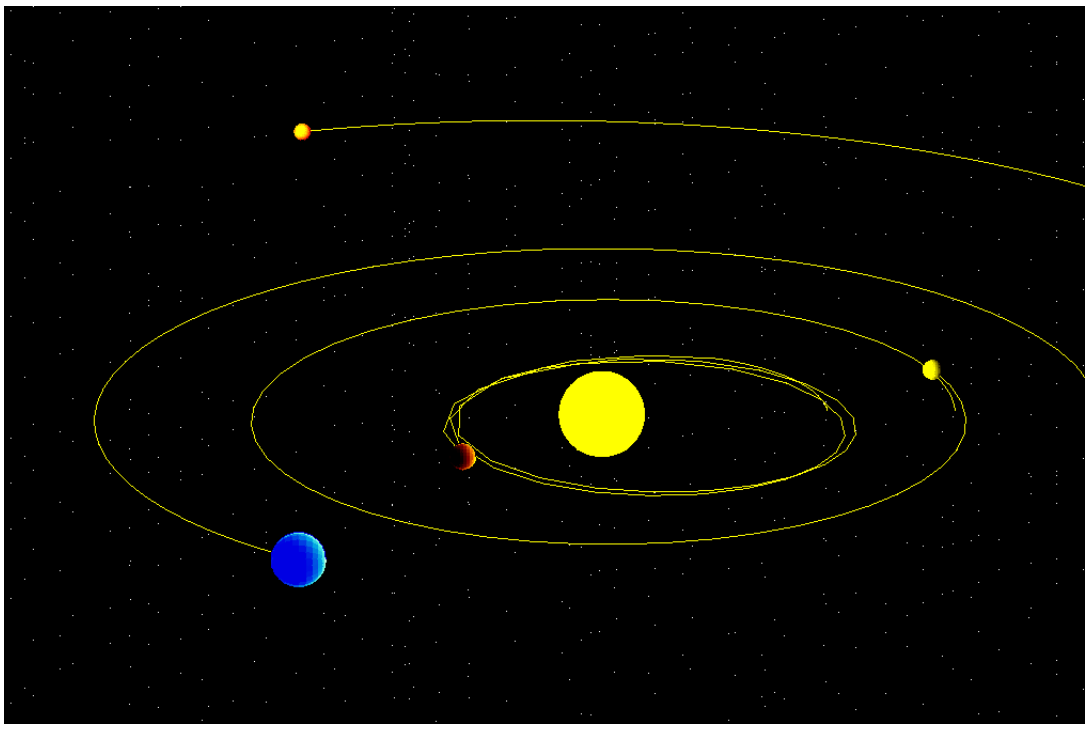


Рисунок 4.2 – Пример работы программы (вид 2)

4.2 Постановка исследования

4.2.1 Цель исследования

Целью исследования является проведение тестирования производительности при создании сцен различной нагруженности. Нагрузка будет меняться в зависимости от параметра, опеределяющего количество вершин между левой и верхней вершинами сфер планет. Оцениваться производительность будет мерой количества кадров в секунду (Frames Per Second, FPS, к/с), которое получается при работе приложения при данной загруженности.

4.2.2 Технические характеристики

Тестирование проводилось на устройстве со следующими техническими характеристиками:

- операционная система Windows 10 pro[10];
- память 32 Гб;
- процессор Intel(R) Core(TM) i5-12400 12th Gen 2.50 ГГц.
- видеокарта NVIDIAGeForce RTX 3050 with 8 Gb GDDR6.

Тестирование проводилось на компьютере, включенном в сеть электропитания. Во время тестирования компьютер был нагружен только встроенными приложениями окружения.

4.3 Результаты исследования

Результаты эксперимента приведены в таблице 4.1. Также на рисунке 4.3 представлен график изменения FPS в зависимости от количества вершин между левой и верхней вершинами сфер планет.

Таблица 4.1 – Зависимость производительности от размера

Параметр сферы, кол-во вершин	Производительность, к/с
2	28
4	21
6	16
8	14
10	10
12	6
14	4
16	3
18	1

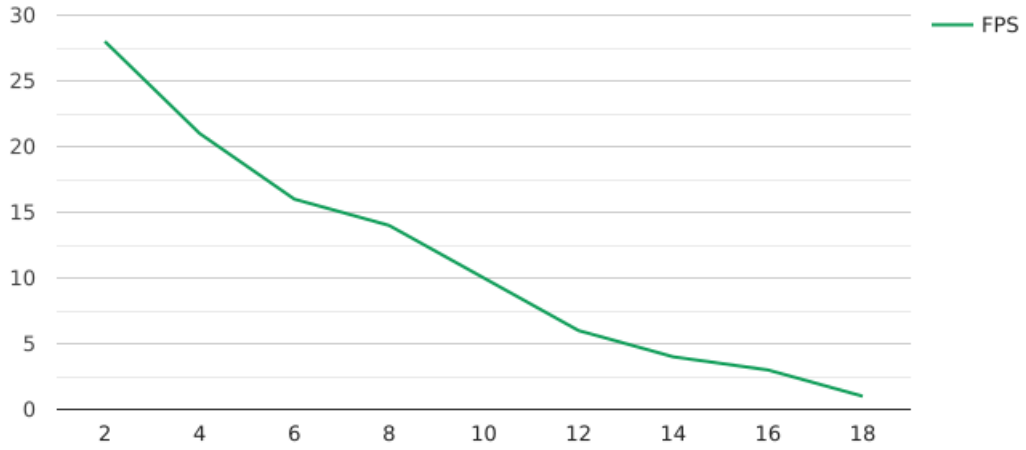


Рисунок 4.3 – Зависимость производительности от количества вершин между левой и верхней вершинами сфер планет

4.4 Выводы из исследовательского раздела

Как видно из результатов, количество кадров в секунду уменьшается экспоненциально при линейном увеличении количества вершин. Рендер сферы с большим количеством вершин, является трудной задачей: при параметре сферы равном 2 программа выдает 28 FPS, в то время как при параметре равном 18 получается лишь 1 кадр в секунду.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано программное обеспечение, визуализирующее движение планет солнечной системы. Данное программного обеспечение позволяет запускать анимацию движения планет, изменять мастштаб относительно Солнца в центре. В качестве дальнейшего развития программы можно предложить реализацию наложения различных текстур на материал поверхности.

Цель, поставленная перед началом работы, была достигнута. В ходе курсовой работы были решены следующие задачи:

- были описаны и выбраны алгоритмы трехмерной графики;
- были реализованы изученные алгоритмы;
- разработано программное обеспечение для визуализации модели;
- исследована зависимость производительности от количества вершин сферы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Методы представления дискретных данных [Электронный ресурс]. Режим доступа: https://www.graphicon.ru/oldgr/ru/library/multires_rep/index.html (дата обращения: 22.11.2023).

[2] Роджерс Д.Ф. Алгоритмические основы машинной графики. 1989.

[3] Warnock algorithm [Электронный ресурс]. Режим доступа: https://www.brainkart.com/article/Warnock-algorithm_5717/ (дата обращения: 20.09.2023).

[4] RayTracing — царь света и теней [Электронный ресурс]. Режим доступа: https://old.computerra.ru/206167/ (дата обращения: 12.11.2023).

[5] Простые модели освещения [Электронный ресурс]. Режим доступа: https://cgraph.ru/node/435 (дата обращения: 30.09.2023).

[6] Е. А. Снижко. Компьютерная геометрия и графика. 2005.

[7] Моделирование движения космических тел для исследования устойчивости планетарных систем двойных звёзд Пожалуйста, не забудьте правильно оформить цитату: Колпащикова, Д. Д. Моделирование движения космических тел для исследования устойчивости планетарных систем двойных звёзд [Электронный ресурс]. Режим доступа: https://moluch.ru/archive/224/52740/ (дата обращения: 07.10.2023).

[8] Руководство по языку программирования C++ [Электронный ресурс]. Режим доступа: https://metanit.com/cpp/tutorial/?ysclid=lq8g6j5pxo297336800 (дата обращения: 05.12.2023).

[9] QT Documentation [Электронный ресурс]. Режим доступа: https://doc.qt.io/qt-6/ (дата обращения: 05.12.2023).

[10] Microsoft.com [Электронный ресурс]. https://www.microsoft.com/ru-ru/software-download/windows10 (дата обращения: 27.10.2023).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе.

ПРИЛОЖЕНИЕ Б

Листинги.