**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:*

***Моделирование эффекта «Фата-моргана»***

Студент __ИУ7-55Б_____ _____ __**М. Д. Коротыч**_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ _____**К. А. Кивва**_____
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

*2021 г.*

# Содержание

# Введение

Компьютерная графика для современного человека стала одной из самых важных и неотъемлемых частей его жизни. Сейчас она используется везде: в компьютерных играх, в кинопроизводстве, в фотографии и дизайне. В связи с этим возникает проблема: очень важно, чтобы изображения, построенные на экране компьютера, были реалистичными, то есть учитывали свойства света (преломление, дифракция, рассеивание), выбранный цвет, а также другие физические свойства реальных объектов.

Предметом компьютерной графики как науки являются создание, хранение и обработка моделей и их изображений с помощью аппаратно-программных вычислительных комплексов. Таким образом, можно сказать, что компьютерная графика является разделом информатики, который занимается проблемами получения различных изображений (рисунков, чертежей, мультипликации) на компьютере. Большое количество алгоритмов призвано решить эти проблемы, однако очень часто сталкиваются с очень большими затратами ресурсов, прежде всего это память и время. Эффективность как по памяти, так и по времени очень важна, ведь на практике пользователь не готов столкнуться с медленно и плохо работающей программой.

Цель данной работы: реализовать построение трёхмерной сцены и визуализацию оптического явления «Фата-моргана». И чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1. исследование оптической модели фата-моргана;
2. исследование существующих алгоритмов построения трёхмерных изображений;
3. выбор наиболее подходящих и оптимальных алгоритмов для поставленной задачи;

4. описание структуру трёхмерной сцены, включая объекты, из которых состоит сцена, и описать выбранное физическое явление, которое будет визуализировано;

5. выбор и/или модифицирование существующих алгоритмов трёхмерной графики, которые позволяют визуализировать трёхмерную сцену;

6. разработка программного обеспечения, которое позволит отобразить трёхмерную сцену и визуализировать оптическое явление.

7. реализация данных алгоритмов для создания трёхмерной сцены;

# 1 Аналитическая часть

## 1.1 Описание эффекта

**Фа́та-морга́на** — редкая и сложная форма миража над водой (морем или океаном) (изображена на рисунке 1). Фата-моргана возникает, когда в нижних слоях атмосферы из-за разницы температур образуется несколько чередующихся слоёв воздуха различной плотности. Возникающие образы меняются и сильно искажены — они способны давать зеркальные отражения. И именно в результате отражения (а также преломления) лучей реально существующие объекты дают на горизонте или над ним по нескольку искажённых изображений. Эти миражи получили название в честь героини бретонского эпоса, которую звали Фата Моргана (с итальянского — «фея Моргана»). [1]

Рисунок 1 - Фата-моргана: корабль «плывёт» над уровнем горизонта.

### 1.1.1 Введение в эффект

Фиксируется система координат, связанная с центром земного шара, в которой ось $Z$ соответствует вертикальному направлению.

Тогда каждый прямолинейный луч обладает единственной «точкой сближения» с поверхностью воды — точкой, где расстояние от самого луча до центра сферы минимально. Проще говоря, если луч не пересекает водную поверхность, то он перпендикулярен радиус-вектору точки сближения, проведённому из центра координат системы (центр окружности на рисунке 2), и удобно рассмотреть вторую систему координат с вертикальной осью $Z'$, идущей вдоль радиус-вектора точки сближения, и осью $X'$ вдоль рассматриваемого луча. Атмосферный эффект состоит в небольшом искажении направления луча в случае, если луч долгое время проходит тонкий слой воздуха около поверхности воды.

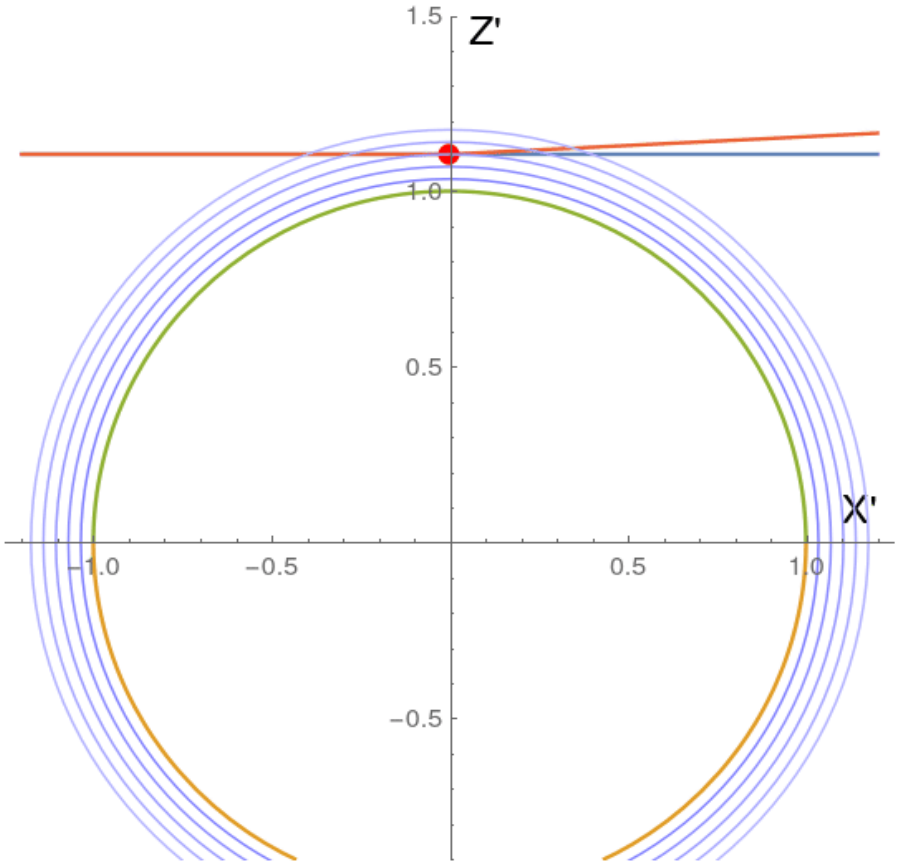


Рисунок 2 - Система координат X'Z'. Земная сфера (единичная окружность) и более горячие слои воздуха (выделены голубым цветом). Красным обозначена точка сближения луча, меняющего своё направление от первоначального (тёмно-синий)

В общем виде размер искажения угла отклонения луча от первоначального направления $d\alpha = \psi(h)$, где $h$ — расстояние от точки сближения до водной поверхности, а $\psi(h)$ – функция, определяющая характер зависимости угла отклонения от параметра $h$. [11]

Свойства:

- при возрастании $h$ значение $\psi(h)$ быстро стремится к нулю, однако при очень малых $h$ функция будет принимать существенные значения;
- $\psi(h)$ корректно определена и при некоторых отрицательных значениях h: $D(\psi) = [-h_0; +\infty)$, где $h_0$ – левая граница видимости объектов;

- однако при существенно отрицательных $h$ функция $\psi(h)$ не определена и луч не рассматривается (не попадает в поле зрения);
- в реальной практике $\psi$ может зависеть и от дополнительных параметров, таких как длина волны света (в нашем случае — цветовая компонента).

### 1.1.2 Исследование функции $\psi$

Будем исходить из концепции, что $\psi(h)$ пропорциональна времени нахождения в слое фиксированной толщины $\eta$ (константа модели).

$$\psi(h) = 0 \text{ при } h \geq \eta \tag{5}$$

Для нахождения $\psi(h)$ воспользуемся уравнением:

$$\sqrt{R_0^2 - x^2} = R_0 + h - \eta, \text{ где } R_0 \text{ - радиус Земного шара} \tag{6}$$

Находим отсюда $x$:

$$x^2 = 2R_0(\eta - h) \Rightarrow x_0 = \sqrt{2R_0(\eta - h)} \tag{7}$$

Здесь $x$ - длина сегмента луча внутри искажающей зоны воздуха

В итоге формула и график функции $\psi(h)$ выглядят следующим образом (последний изображён на рис. 3):

$$\psi(h) = c_0\sqrt{\eta - h}, \text{ где } c_0 \text{ - скорость света в вакууме} \tag{8}$$

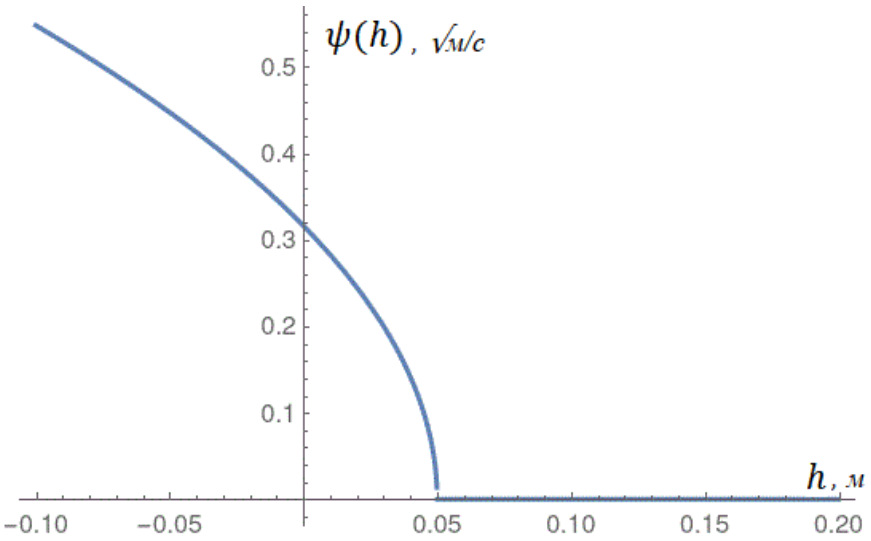


Рисунок 3 - График функции ψ(h)

Данная модель не идеальна, потому что свойства атмосферы не меняются одномоментно, при этом полученная функция не очень удобна для моделирования. Модель необходимо изменить таким образом, что в пределах слоя интенсивность искажения постепенно меняется от максимального значения до нуля.

Степень преломления: $\tau(x,h) = \frac{\eta-(\sqrt{x^2+(R_0+h)^2}-R_0)}{\eta} = 1 - \frac{h}{\eta} - \frac{x^2}{2R_0\eta}$ (9)

При $x = x_0 = \sqrt{2R_0(\eta - h)}$:

$$\tau(x,h) = 1 - \frac{h}{\eta} - \frac{2R_0(\eta-h)}{2R_0\eta} = 1 - \frac{h}{\eta} - \frac{\eta-h}{\eta} = 0 \quad (10)$$

$$\psi_2(h) = \int_{-x_0}^{x_0} \tau(x,h)dx = \int_{-x_0}^{x_0} \left(1 - \frac{h}{\eta} - \frac{x^2}{2\eta R_0}\right) dx = c_2\sqrt{(\eta-h)^3} \Rightarrow$$

$$c_2 = \frac{4\gamma\sqrt{2R_0}}{3\eta R_0}, \quad (11)$$

где $\psi_2(h)$ - скорректированная функция $\psi(h)$. Её график представлен ниже на рисунке 4.

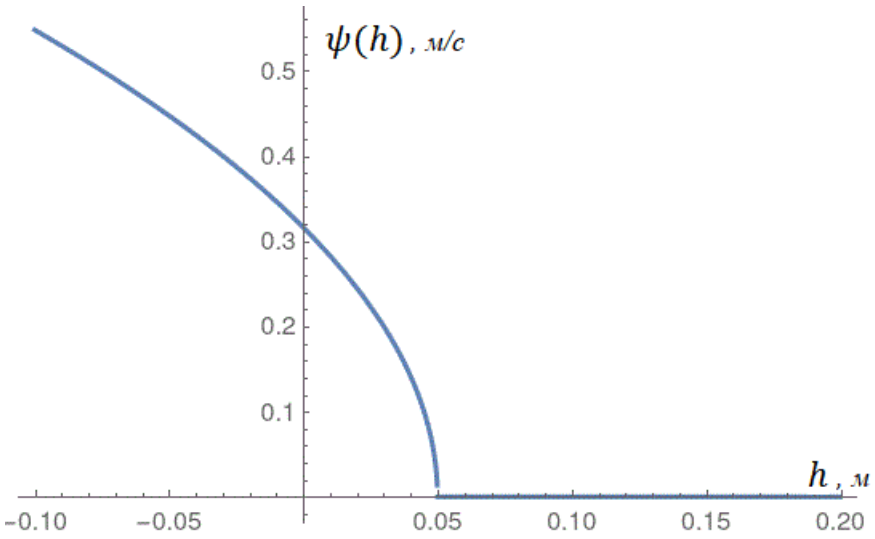


Рисунок 4 - Скорректированный график функции $\psi_2(h)$

### 1.1.3 Вычисление граничного значения $h_0$ в отрицательной области

$$\psi_2(h) = c_2\sqrt{(\eta - h)^3} \Rightarrow c_2 = \frac{4\gamma\sqrt{2R_0}}{3\eta R_0}, \quad (12)$$

где $\gamma$ - безразмерный коэффициент искажения.

Искажение воздействует на луч на расстоянии порядка

$$x_0 = \sqrt{2R_0(\eta - h)} \quad (13)$$

$$x_0 tg\ \psi_2(h) = -h \rightarrow c_2\sqrt{2R_0\sqrt{(\eta - h)^3}} = h \quad (14)$$

Выражение (14) преобразуется к виду:

$$c_2(\eta - h)^2 = -h \quad (15)$$

$$\frac{8\gamma}{3\eta}(\eta - h)^2 = -h \quad (16)$$

Корень этого квадратного уравнения:

$$h = \frac{8\gamma\eta}{3} \quad (17)$$

Таким образом, функция $\psi(h)$ определена и положительна на интервале $(-\mathrm{h};\ \eta) = (-\frac{8\gamma\eta}{3};\ \eta)$

$$\psi_2(h) = c_2\sqrt{(\eta - h)^3} \Rightarrow c^2 = \frac{\gamma\sqrt{2^5}}{3\eta\sqrt{R_0}},\ h \leq \eta \qquad (18)$$

### 1.1.4 Обращение формулы искажения

С высокой точностью можно считать, что угол искажения обратного луча равен углу прямого луча. [10]

### 1.1.5 Алгоритмическая интерпретация

В модели для простоты рассматривается только продольное искривление вдоль направления оси $X$. Атмосферный эффект искривления луча таким образом меняет направления луча в вертикальной плоскости.

1. Формируется неискажённое изображение сцены за исключением водной поверхности, которая сохраняется в виде матрицы пикселей размером $w_0 \times h_0$, где $w_0$ - ширина матрицы изображения, а $h_0$ – высота матрицы изображения; [9]

2. Создаётся новая матрица размера $w_0 \times h_0$, геометрически соответствующая матрице в пункте 1, привязанная к новому z-буферу; [11]

3. Определяется прямоугольная область матрицы, которая занята изображением воды;

4. Для каждого пикселя $(X, Y)$ новой матрицы рассматривается последовательность пикселей $(X, U)$, где $U \in [U_0; U_0 + dU]$ [5], значение $U_0$ соответствует $Y$. [12]

5. Для каждого из этих пикселей производятся следующие действия:

a. вычисляется угол искажения луча, соответствующий этому пикселю [6];
b. определяется расстояние до объекта в этом пикселе при помощи z-буфера;
c. по смещению $\Delta = U - U_0$ проверяется соответствие расстояния до объекта степени искажения ($\Delta$) [7];
d. если соответствие проверено, то отображается точка из пикселя $(X, U)$ исходного изображения, записывая в новый z-буфер;
e. если ни один из пикселей не дал эффекта, то полагается, что здесь «пустота» [8];
f. параллельно этому «вода» добавляется в каждый пиксель.

Функция, выражающая зависимость искажения угла (коэффициента преломления) от расстояния до объекта:

$$y = x_0 \frac{4}{3}(1 - \frac{h}{\eta}) \quad (19)$$

## 1.2 Алгоритмы удаления невидимых линий и поверхностей

### 1.2.1 Алгоритм, использующий z-буфер

Алгоритм, использующий *z-буфер* это один из простейших алгоритмов удаления невидимых поверхностей, работающий в пространстве изображения. [1] Идея z-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пикселя в пространстве изображения, z-буфер — это отдельный буфер глубины, используемый для запоминания координаты $z$ или глубины каждого видимого пикселя в пространстве изображения. В процессе работы глубина или значение $z$ каждого нового пикселя, который нужно занести в буфер кадра, сравнивается с глубиной того пикселя, который уже занесён в z-буфер.

Если это сравнение показывает, что новый пиксель расположен впереди пикселя, находящегося в буфере кадра, то новый пиксель заносится в этот буфер и, кроме того, производится корректировка z-буфера новым значением $z$, как показаны на рисунке 5. Если же сравнение даёт противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по $x$ и $y$ наибольшего значения функции $z(x, y)$.

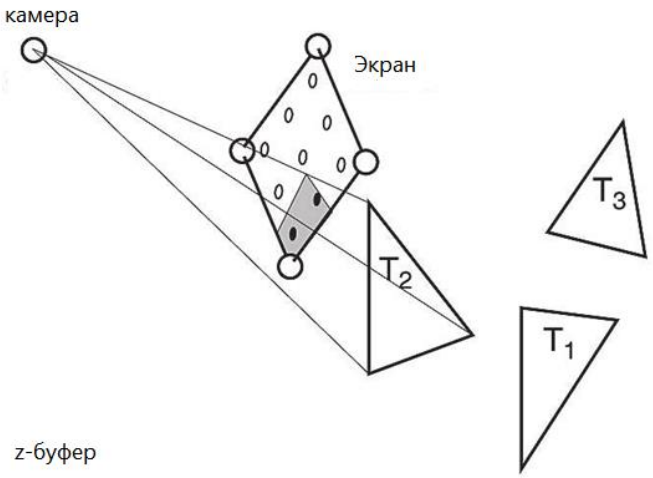


Рисунок 5 – Алгоритм z-буфера

Главное преимущество алгоритма – его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоёмкости алгоритма не более чем линейна. Элементы сцены или картинки можно заносить в буфер кадра или в z-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма – большой объём требуемой памяти. Если сцена подвергается видовому преобразованию и отсекается

до фиксированного диапазона значений координат $z$, то можно использовать z-буфер с фиксированной точностью.

### 1.2.2 Обратная трассировка лучей

Из виртуальной камеры через каждый пиксель изображения испускается луч и находится точка его пересечения с поверхностью сцены. Лучи, выпущенные из камеры, называют первичными. Пусть первичный луч пересекает некий объект. Далее необходимо определить для каждого источника освещения, видна ли из него эта точка пересечения. Если предположить, что все источники света точечные, то тогда для каждого источника света к нему испускается теневой луч из точки объекта. Это позволяет сказать, освещается ли данная точка конкретным источником. Если теневой луч находит пересечение с другими объектами, расположенными ближе, чем источник света, значит рассматриваемая точка находится в тени от этого источника и освещать её не надо. Иначе считается, что точка освещается. Тогда освещение со всех видимых источников света складывается. Далее, если материал объекта имеет отражающие свойства, из нашей точки испускается отражённый луч и для него вся процедура трассировки рекурсивно повторяется. Аналогичные действия должны быть выполнены, если материал имеет преломляющие свойства. [2]

Плюсы данного метода – фотореалистичность и чёткость получаемого изображения. Однако объём вычислений, которые надо при этом производить, – очень большой и обычно трассировка лучей работает довольно медленно. Причём, львиная доля процессорного времени затрачивается на поиск пересечений. Поэтому вопрос производительности здесь стоит на первом месте.

### 1.2.3 Алгоритм Варнока

Алгоритм Варнока является одним из примеров алгоритма, основанного на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто.

Поскольку алгоритм Варнока нацелен на обработку картинки, он работает в пространстве изображения. В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения.

Видимая часть картинной плоскости разбивается на четыре равные части и рассматривается то, каким образом могут соотноситься между собой проекции граней получившейся части картинной плоскости.

1. Проекция грани полностью накрывает область (рис 6, a).
2. Проекция грани пересекает область, но не содержится в ней полностью (рис. 6, c).
3. Проекция грани целиком содержится внутри области (рис. 6, b).
4. Проекция грани не имеет общих внутренних точек с рассматриваемой областью (рис 6, a).

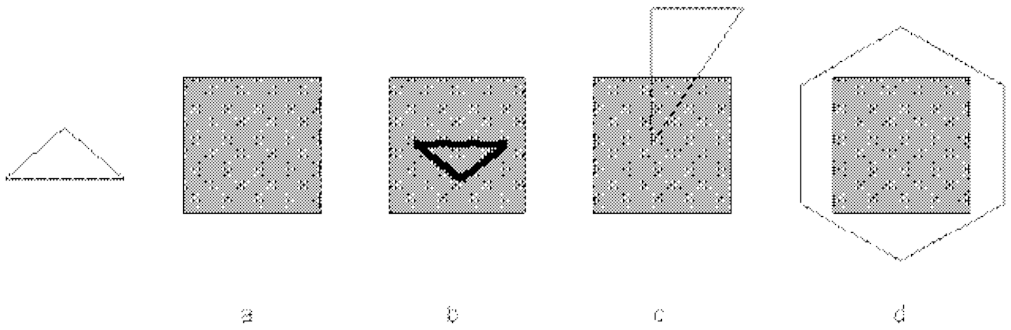


Рисунок 6 – Соотношение проекции грани с подокном

Сравнивая область с проекциями всех граней, можно выделить случаи, когда изображение, получающееся в рассматриваемой области, определяется сразу:

- Проекция ни одной грани не попадает в область.
- Проекция только одной грани содержится в области или пересекает область. В этом случае проекции грани разбивают всю область на две части, одна из которых соответствует этой проекции.
- Существует грань, проекция которой полностью накрывает данную область, и эта грань расположена к картинной плоскости ближе, чем все остальные грани, проекции которых пересекают данную область. В данном случае область соответствует этой грани.

Если ни один из рассмотренных трёх случаев не имеет места, то снова область разбивается на четыре равные части и проверяется выполнение этих условий для каждой из частей. Те части, для которых таким образом не удалось установить видимость, разбиваются снова и т. д. [3]

Плюс алгоритма Варнока – указанный способ можно применять рекурсивно. Если в окне содержится только один многоугольник и если он целиком охвачен этим окном, то его легко изобразить, не проводя дальнейшего разбиения. Однако простота критерия разбиения, а также негибкость способа разбиения приводят к тому, что количество подразбиений оказывается велико.

### 1.2.4 Алгоритм художника

Название «алгоритм художника» относится к технике, используемой многими живописцами: сначала рисуются наиболее удалённые части сцены, потом части, которые ближе. Постепенно ближние части начинают

перекрывать отдалённые части более удалённых объектов. Задача программиста при реализации алгоритма художника — отсортировать все полигоны по удалённости от наблюдателя и начать выводить, начиная с более дальних.

Преимущества этого алгоритма – его простота использования. Но есть множество его недостатков, как то:

- Алгоритм не позволяет получить корректную картину в случае взаимно перекрывающихся полигонов, как показано на рисунке 7. В этом случае следует разбить конфликтный полигон на несколько меньших;

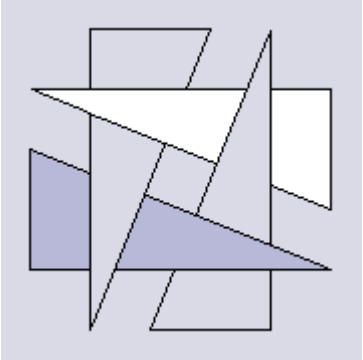

Рисунок 7 - Взаимноперекрывающиеся полигоны при любой сортировке будут выведены неверно

- Ещё одной распространённой проблемой является то, что система прорисовывает также области, которые впоследствии будут перекрыты, на что тратится лишнее процессорное время.

Поэтому целесообразно использовать алгоритм z-буфера, который можно рассматривать как развитие алгоритма художника. [4]

## 1.3 Алгоритмы освещения

### 1.3.1. Модель освещения Ламберта

Модель Ламберта моделирует идеальное диффузное освещение. Считается, что свет при попадании на поверхность рассеивается равномерно во все стороны. При расчёте такого освещения (на рисунке 8) учитывается только ориентация поверхности (нормаль $N$) и направление на источник света (вектор $L$). Рассеянная составляющая рассчитывается по закону косинусов (закон Ламберта):

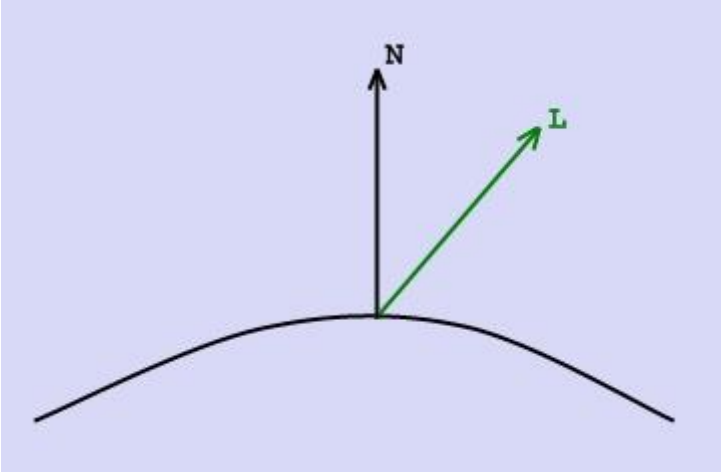


Рисунок 8 – Расчёт нормали и вектора направления света на источник

$$I_d = k_d \cos(\vec{L}, \vec{N}) i_d = k_d i_d (\vec{L} \cdot \vec{N}), \quad (20)$$

где

- $I_d$ – рассеянная составляющая освещённости в точке,
- $k_d$ – свойство материала воспринимать рассеянное освещение,
- $i_d$ – мощность рассеянного освещения,
- $L$ – направление из точки на источник,
- $N$ - вектор нормали в точке.

Модель Ламберта является одной из самых простых моделей освещения. Данная модель очень часто используется в комбинации других моделей, практически в любой другой модели освещения можно выделить диффузную составляющую. Более-менее равномерная часть освещения

(без присутствия какого-либо всплеска) как правило будет представляться моделью Ламберта с определенными характеристиками. Данная модель может быть очень удобна для анализа свойств других моделей (за счёт того, что её легко выделить из любой модели и анализировать оставшиеся составляющие). [6]

### 1.3.2 Модель освещения Фонга

Пусть заданы точечный источник света, расположенный в некоторой точке, поверхность, которая будет освещаться, и наблюдатель (точечный). Каждая точка поверхности имеет свои координаты и в ней определена нормаль к поверхности. Её освещённость складывается из трёх компонент: фоновое освещение, рассеянный свет и бликовая составляющая. Свойства источника определяют мощность излучения для каждой из этих компонент, а свойства материала поверхности определяют её способность воспринимать каждый вид освещения.

Фоновое освещение — это постоянная в каждой точке величина надбавки к освещению. Вычисляется фоновая составляющая освещения как

$$I_a = k_a i_a, \text{ где} \quad (21)$$

$I_a$ – фоновая составляющая освещённости в точке,

$k_a$ - воспринимаемость материалом фонового освещения,

$i_a$ – мощность фонового освещения.

Из формулы видно, что фоновая составляющая освещённости не зависит от пространственных координат освещаемой точки и источника. Поэтому при моделировании освещения, в большинстве случаев, не имеет смысла брать более одного фонового источника света.

Рассеянный свет при попадании на поверхность рассеивается равномерно во все стороны. При расчёте такого освещения учитывается только ориентация поверхности (нормаль) и направление на источник света (показано на рисунке 9):

$$I_d = i_d k_d cos(L, N) = i_d k_d (L \cdot N)\text{ , где} \qquad (21)$$

$I_d$ – рассеянная составляющая освещённости в точке,

$i_d$ – свойство материала воспринимать рассеянное освещение,

$k_d$ – мощность рассеянного освещения,

$L$ – направление из точки на источник,

$N$ – вектор нормали в точке.

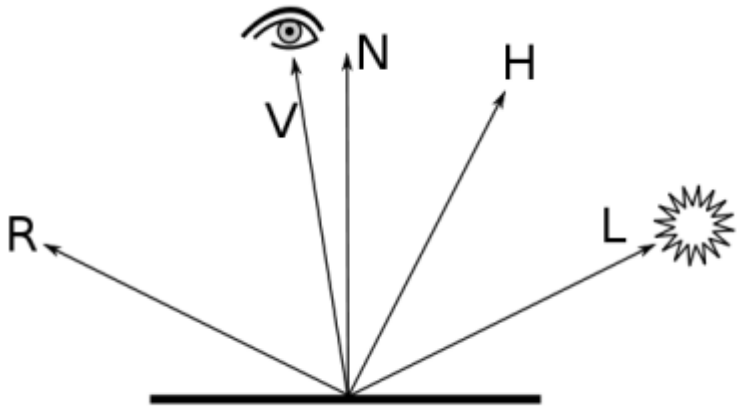


Рисунок 9 – Модель освещения Фонга

Зеркальный свет при попадании на поверхность подчиняется закону отражения света: “Отражённый и падающий лучи лежат в плоскости, содержащей перпендикуляр к отражающей поверхности в точке падения, и угол падения равен углу отражения”. Т. о. отражённая составляющая освещённости в точке зависит от того, насколько близки направления на наблюдателя и отражённого луча.

$$I_s = k_s i_s cos^{\alpha}(r, v) = k_s i_s (r, v)^{\alpha}\text{, где} \qquad (22)$$

$I_s$ - зеркальная составляющая освещённости в точке,

$k_s$ – коэффициент зеркального отражения,

$i_s$ – мощность зеркального освещения,

$r$ – направление отражённого луча,

$v$ – направление на наблюдателя,

$\alpha$ – коэффициент блеска (свойство материала).

Именно зеркальное отражение представляет наибольший интерес, но в то же время его расчёт требует больших вычислительных затрат. При фиксированном положении поверхности относительно источников света фоновая и рассеянные составляющие освещения могут быть просчитаны единожды для всей сцены, т. к. их значение не зависит от направления взгляда. [5]

Скалярное произведение $r \cdot v$ рассчитывается по формуле:

$$r \cdot v = 2(n \cdot l)(n \cdot v) - (l \cdot v) \quad (23)$$

## 1.4 Анализ методов закрашивания

### 1.4.1 Плоская закраска

Принцип простой: сперва цвет вычисляется в каждой вершине грани, затем полученные значения усредняются, и вся грань закрашивается в полученный цвет. Данная модель обладает высокой скоростью работы, но на визуализированной модели чётко заметны переходы между гранями, что в случае тел вращения будет очень заметно (видно на рисунках 10 и 11).

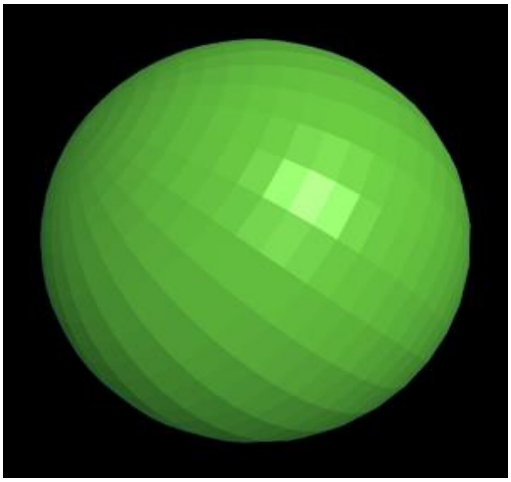

Рисунок 10 - Сфера с плоским затенением, около 2000 треугольников

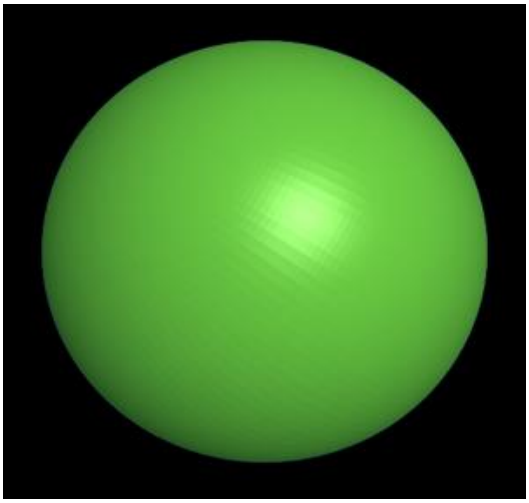

Рисунок 11 - Сфера с плоским затенением, около 32000 треугольников

### 1.4.2 Закраска Гуро

Метод Гуро основывается на определении освещённости грани в её вершинах с последующей билинейной интерполяцией получившихся величин на всю грань, что обеспечивает непрерывное изменение интенсивности при переходе от одной грани к другой без разрывов и скачков.

Преимуществом этого метода является его инкрементальный характер – грань рисуется в виде набора горизонтальных отрезков, причём интенсивность последующего пикселя отрезка отличается от

интенсивности предыдущего на величину постоянную для данного отрезка. Кроме того, при переходе от отрезка к отрезку значения интенсивности в его концах также изменяются линейно, что заметно на рисунках 12 и 13.

Применяется для реализации диффузного рассеивания, так как не обеспечивает гладкости изменения интенсивности, что так необходимо для зеркальных отражений. [7]

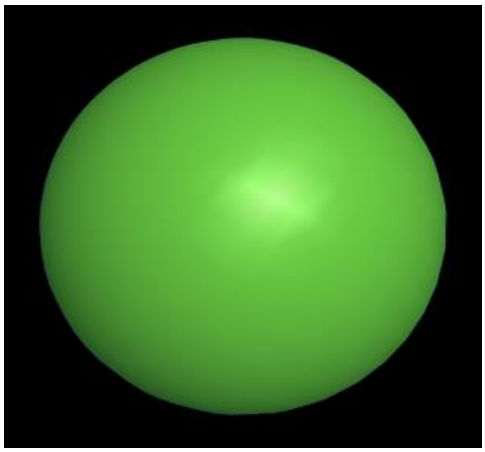

Рисунок 12 - Сфера с затенением по Гуро, около 2000 треугольников

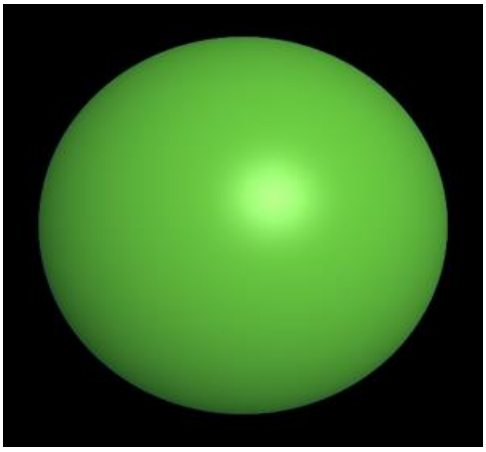

Рисунок 13 - Сфера с затенением по Гуро, около 32000 треугольников

### 1.4.3 Закраска Фонга

В отличие от метода Гуро здесь интерполируется не значение интенсивности по уже известным её значениям в опорных точках, а значение вектора внешней нормали, использующееся затем для вычисления интенсивности пикселя. Поэтому закраска Фонга требует заметно большего объёма вычислений. Однако блики отражённого света придают объектам большей реалистичности (что видно на рисунках 14 и 15). [8]

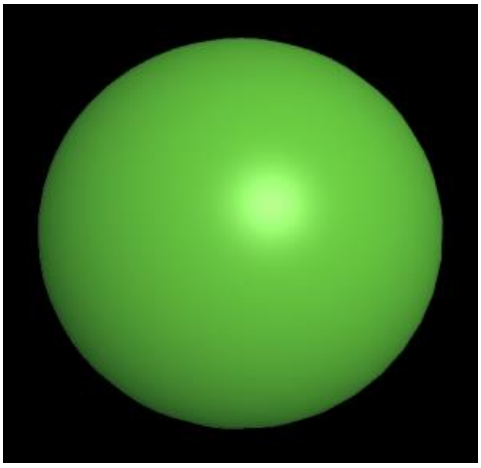

Рисунок 14 - Сфера с затенением по Фонгу, около 2000 треугольников

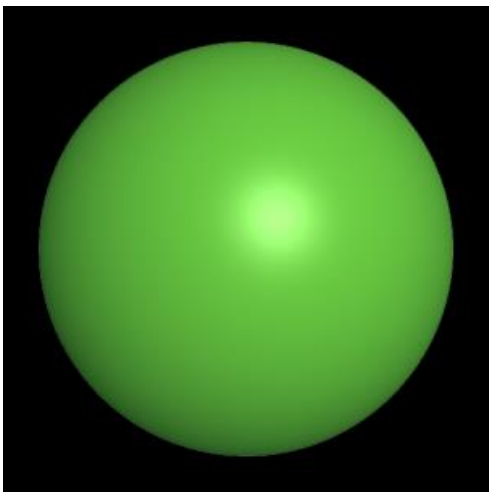

Рисунок 15 - Сфера с затенением по Фонгу, около 32000 треугольников

## 1.5 Вывод

Для работы лучше использовать закраску Фонга, так как она обеспечивает высокий уровень реалистичности за счёт отражаемого света. Для удаления поверхностей будет применён алгоритм z-буфера в следствие его простоты и удобства.

Также был исследован и описан алгоритм моделирования явления фата-моргана при отрисовке сцены.

# 2 Конструкторская часть

В данном разделе рассмотрены требования к программе и алгоритмы визуализации сцены и погодных явлений.

## 2.1 Требования к программе

Программа должна давать возможность:

- визуализировать сцену
- выбирать из предложенного набора желаемый объект
- наблюдать эффект фата-моргана
- менять физические параметры среды

## 2.2 Общий алгоритм визуализации

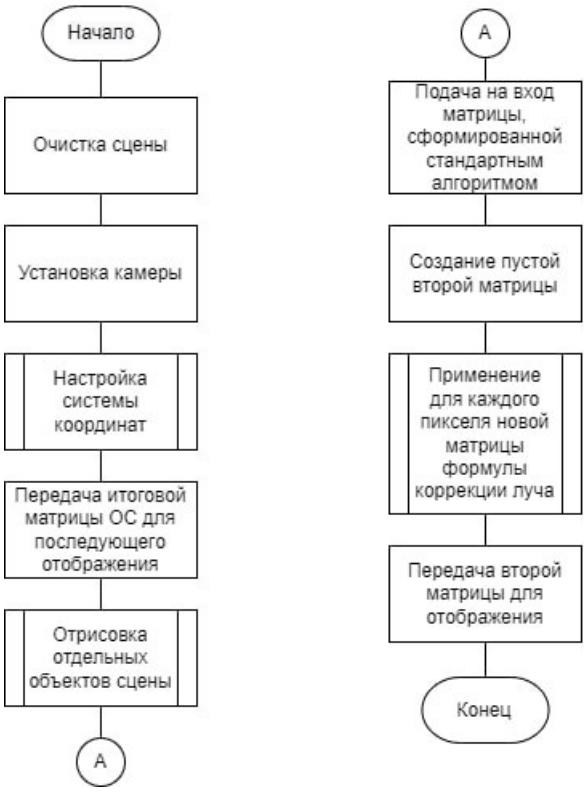


Рисунок 16 – Схема общего алгоритма визуализации

В процессе выполнения как первого, так и второго шага алгоритма, изображённого на рисунке 16, необходимо выполнить несколько дополнительных задач, как то: расчёт цвета пикселя с учётом

освещённости, отрисовка графических примитивов (в моём случае это треугольники), а также растеризация с применением z-буфера.

## 2.3 Общий алгоритм отрисовки отдельных объектов сцены

Основные этапы работы алгоритма применения эффекта фата-морганы к объекту на сцене представлены ниже на рисунке 17.


Рисунок 17 – Алгоритм применения эффекта к объекту

# 3 Технологическая часть

Языком программирования был выбран язык C#, так как это объектно-ориентированные язык программирования, что позволит:

- управлять объектами сцены в качестве непосредственно программируемых объектов с их свойствами и параметрами,
- что в свою очередь повышает читаемость кода и его сопровождаемость;
- реализовать в полной мере механизмы наследования, полиморфизма и т. д.

Средой программирования была выбрана Visual Studio 2019, потому что:

- она бесплатна для некоммерческого использования;
- многофункциональна (отладка, управление проектами, поддержка систем контроля версий и т. д.)
- позволяет работать с Windows Forms (API, отвечающий за графический интерфейс пользователя. Упрощает доступ к элементам MS Windows обёрткой для Win32-кода)

## 3.1 Структура программы

### 3.1.1 Структура и состав основных классов

На рисунке 18 представлена диаграмма классов.

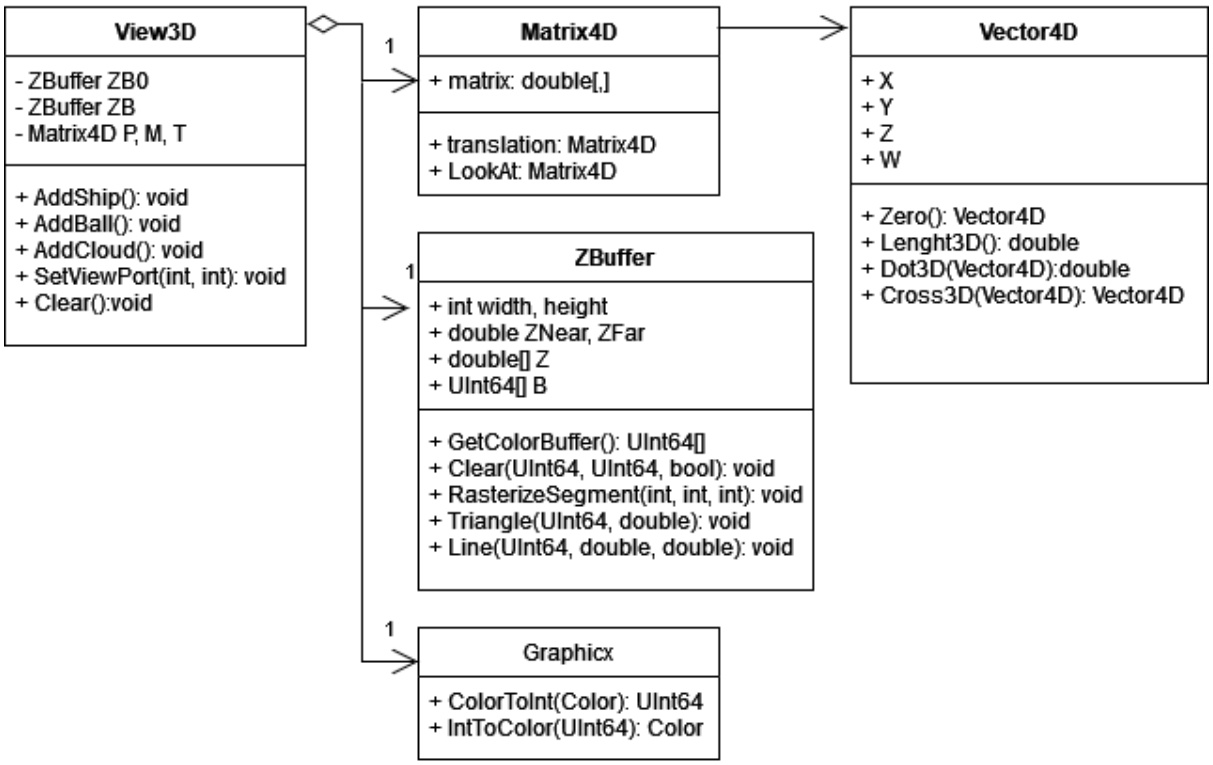


Рисунок 18 – Диаграмма классов программы

- `Graphicx` – класс, переводящий цвет пикселя в целое число и наоборот;
- `View3D` – класс, представляющий наблюдателя. Содержит z-буфер и его граничные значения, матрицы преобразования (объектив), модельную матрицу (расположение наблюдателя);
- `ZBuffer` – класс z-буфера. Хранит информацию о буферах цвета и глубины, имеет методы растеризации сегмента и различных примитивов, очистки, получения цвета пикселя в буфере;
- `Matrix4D` – класс, содержащий 4-мерную матрицу и соответствующие методы;
- `Vector4D` – класс, содержащий 4-мерный вектор и соответствующие методы.

### 3.1.2 Структура и состав файлов

- `Program.cs` - главная точка входа в приложение;
- `Form1.cs` – интерфейс приложения;

- `Graphicx.cs` – содержит класс `Graphicx`;
- `Matrix4D.cs` – содержит класс `Matrix4D`;
- `Vector4D.cs` – содержит класс `Vector4D`;
- `View3D.cs` – содержит класс `View3D`;
- `ZBuffer.cs` – описание z-буфера.

## 3.2 Интерфейс программы

На рисунке 19 представлено окно программы. Интерфейс представляет сцену и различные её настройки: добавление объектов на сцену (меню добавления на рис. 20), изменение физических параметров (коэффициент преломления), изменение положения обзора, а также очистку сцены и очистку лога.

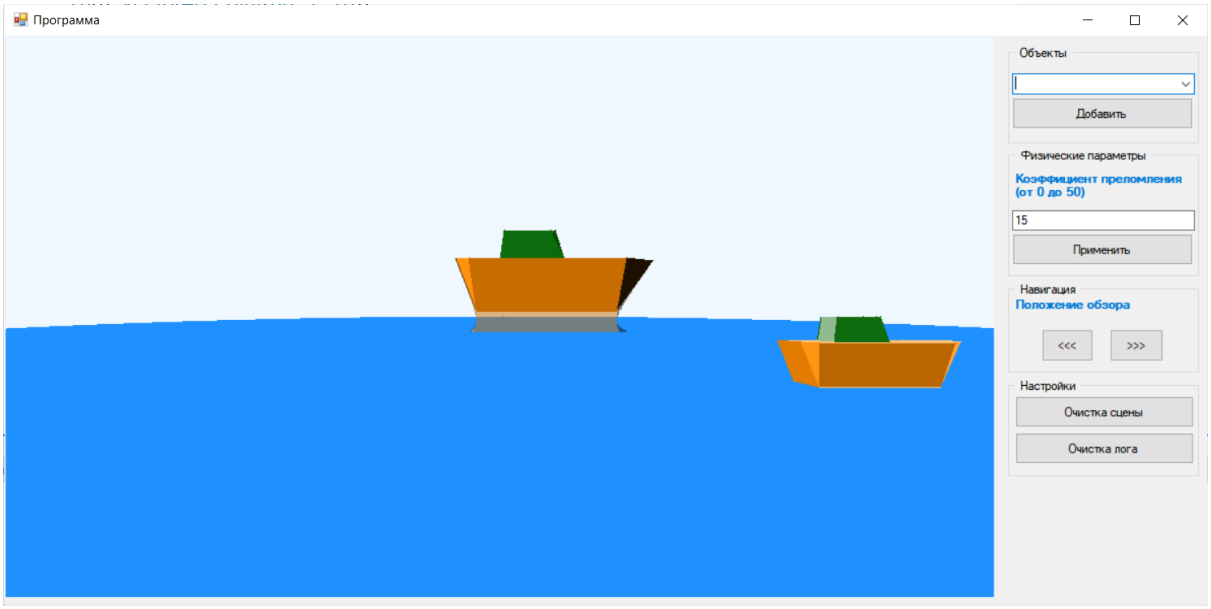


Рисунок 19 – Пример сцены

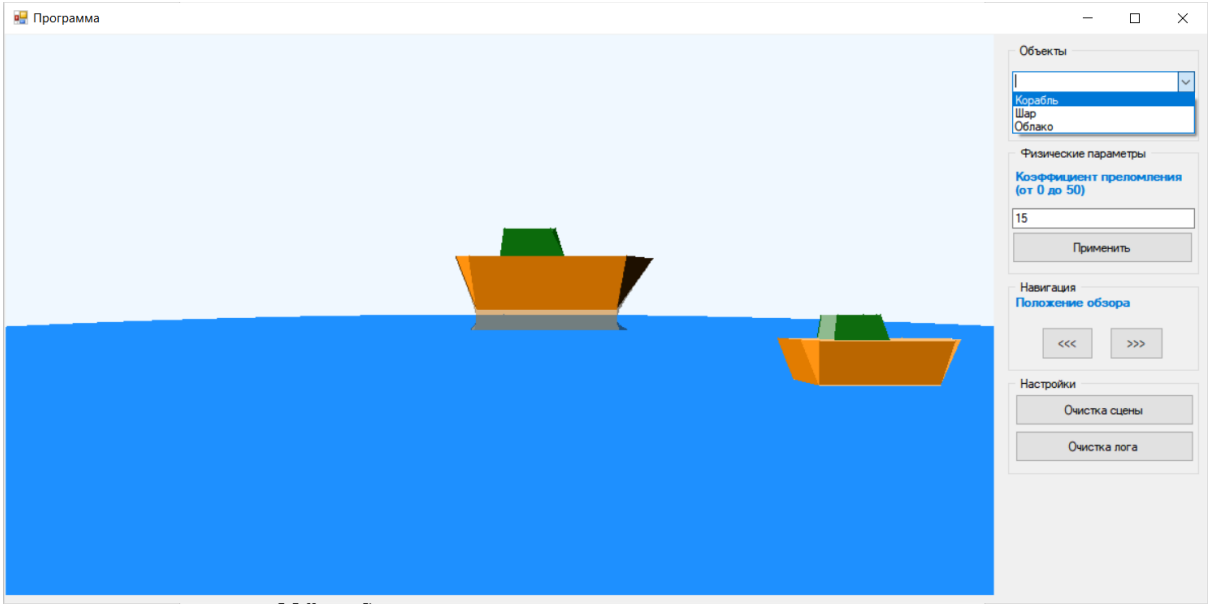


Рисунок 20 – Развёрнутое меню выбора добавляемого объекта

## 3.3 Вывод

В разделе были представлены структуры, состав классов и интерфейс программы.

# 4 Экспериментальная часть

## 4.1 Исследование зависимости времени генерации изображения от коэффициента преломления

Суть эксперимента заключается в наблюдении зависимости изменения среднего времени генерации изображения от значения коэффициента преломления.

Сцена состояла из фона (водная поверхность и небо), а также их объектов, добавляемых на эту сцену. Объект «корабль» состоял из 17 полигонов.

Эксперимент проводился на компьютере со следующими характеристиками:

- Intel® Core™ i7-8550U;
- 4 ядра;
- 8 логических процессоров;
- 8 ГБ ОЗУ.

Результаты представлены в таблице 1 и на рисунке 21.

Таблица 1 – Зависимость среднего времени генерации от коэффициента преломления

| Коэффициент преломления | Время генерации изображения, нс |
|---|---|
| 0 | 26114700 |
| 2,5 | 41592400 |
| 5 | 42747600 |
| 7,5 | 43917300 |
| 10 | 45002500 |
| 12,5 | 45443900 |
| 15 | 45560600 |
| 17,5 | 45541100 |
| 20 | 46338200 |
| 22,5 | 46832000 |
| 25 | 47483600 |
| 27,5 | 48444100 |
| 30 | 48439800 |
| 32,5 | 49633000 |
| 35 | 50210500 |
| 37,5 | 51309100 |
| 40 | 52213000 |
| 42,5 | 55708200 |
| 45 | 64867700 |
| 50 | 66789020 |

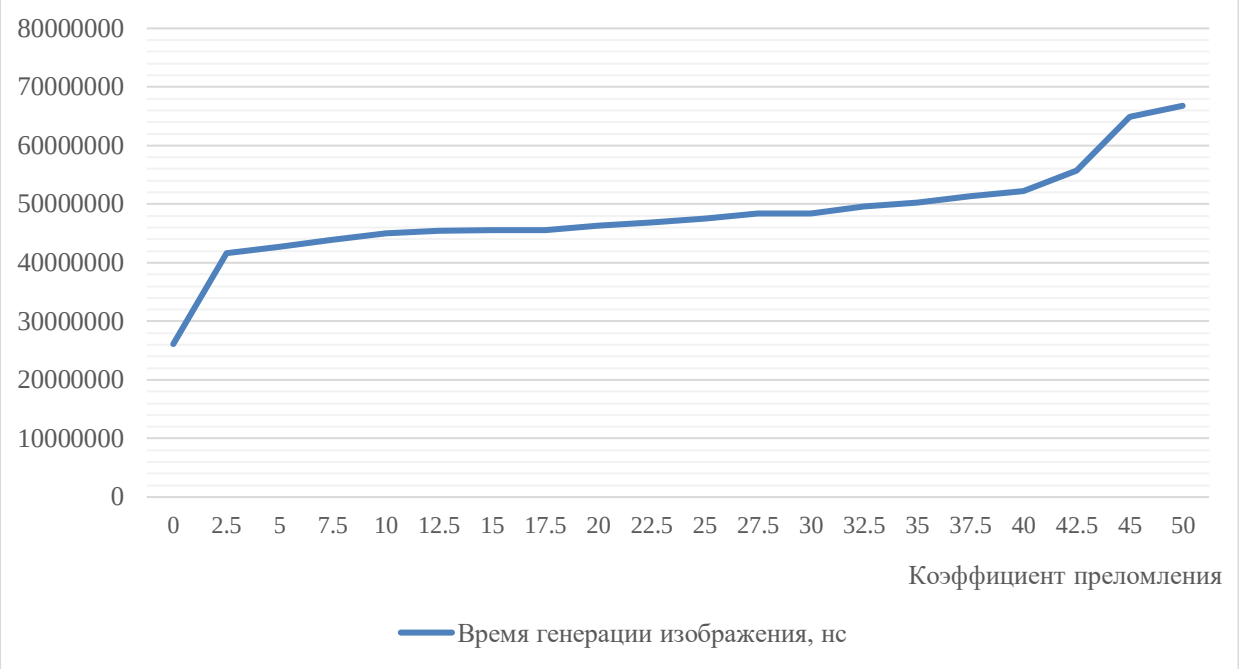


Рисунок 21 – График зависимости среднего времени генерации изображения от коэффициента преломления

Таблица и график показывают, что с увеличением коэффициента преломления линейно увеличивается и среднее время генерации объекта из-за многократного пересчёта координат нужных пикселей, а также таких «дорогостоящих» математических операций, как возведение в степень, деление вещественных чисел. Математической логикой алгоритма обусловлены и резкие скачки графика при коэффициентах преломления с 0 до 5 и с 40 до 50. При коэффициенте больше нуля нет необходимости считать искажение координаты – оно будет нулевым. Если же коэффициент будет слишком большим, то соответствующие пиксели необходимо – кроме того, что надо обсчитать их искажения – «смешать» с пикселями фона (неба например).

Чем больше коэффициент преломления и чем дальше объект расположен от наблюдателя, тем сильнее объект будет искажаться.

# Заключение

Во время выполнения курсовой работы была исследована оптическая модель явления фата-моргана, исследованы существующие алгоритмы построения трёхмерных изображений, выбраны наиболее подходящие и оптимальные алгоритмы для поставленной задачи, описана структура трёхмерной сцены, включая объекты, из которых состоит сцена, и описано выбранное физическое явление, которое было визуализировано, реализованы соответствующие алгоритмы для создания трёхмерной сцены, разработано программное обеспечение, которое позволило отобразить трёхмерную сцену и визуализировать оптическое явление.

Программа реализована таким образом, что пользователь может добавлять новые объекты на сцену, изменять физические параметры, от которых зависит наблюдаемый эффект.

В ходе выполнения экспериментальной части было установлено, что время генерации изображения растёт линейно с ростом коэффициента преломления; вычисление эффекта довольно сильно зависит от медленных операций возведения в степень и деления вещественных чисел. Количество затраченного времени на отрисовку изображения при определённых значениях коэффициента резко увеличивается за счёт «предельных» особенностей алгоритма.

При возможном развитии проекта появится необходимость: улучшить отрисовку добавляемых объектов с полноценными реалистичными текстурами, добавить смену времени суток, улучшить алгоритм эффекта фата-морганы.

# Список литературы


1. Сушинова, Яна. Что такое фата-моргана. Аргументы и факты. [Электронный портал] / В. Шукшин. – Электрон. текстовые дан. – Москва: [б.и.], 2022. Режим доступа: https://aif.ru/dontknows/eternal/chto_takoe_fata-morgana
2. Бекназарова, Саида. Удаление невидимых линий. Медиаобразовательный портал. [Электронный ресурс] / С. Бекназарова. – Электрон. текстовые дан. – Ташкент: [б.и.], 2012. - Режим доступа: http://cb.mediaedu.uz/cabinet.php?page=courses&urok=90
3. Фролов Владимир, Фролов Александр. Обратная трассировка лучей. Ray-tracing. [Электронный ресурс] / А. Фролов. – Электрон. текстовые дан. – Москва: [б.и.], 2007. – Режим доступа: http://ray-tracing.ru/articles164.html, свободный
4. Дёмин А. Ю., Кудинов А. В. Компьютерная графика. Алгоритм Варнока (Warnock). [Электронный ресурс] / А. Ю. Дёмин – Электрон. текстовые дан. – Томск: [б.и.], 2005. – Режим доступа: http://compgraph.tpu.ru/warnock.htm, свободный
5. Кантор И. Удаление невидимых частей. Алгоритм художника. [Электронный ресурс] / И. Кантор – Электрон. текстовые дан. – Москва: [б.и.], 2005. – Режим доступа: http://algolist.ru/graphics/3dfaq/articles/32.php, свободный
6. Кулагин, Денис. Закон Ламберта. Модель отражения Фонга. Модель отражения Блинна-Фонга. Компьютерная графика. Теория, алгоритмы, примеры на C++ и OpenGL [Электронный ресурс]. / Д. Кулагин – Электрон. текстовые дан. – Москва: [б.и.] – Режим доступа:

https://compgraphics.info/3D/lighting/phong_reflection_model.php, свободный

7. Кругленко Александр. Компьютерная графика. Простые модели освещения. [Электронный ресурс]. / А. Кругленко – Электрон. текстовы дан. – Москва: [б.и.] – Режим доступа: http://grafika.me/node/344, свободный
8. Савкина, Елена. Закраска методом Гуро. Электронный учебник «Компьютерная графика». [Электронный ресурс] / Е. Савкина – Электрон. текстовые дан. – Москва: [б.и.], 2009. – Режим доступа: http://openedo.mrsu.ru/catalog/Tehnicheskie/2009/Savkina2/resources/resource_53/content/screen29.htm, свободный
9. Joey de Vries. Camera. LearnOpenGL. [Электронный ресурс]. / Joey de Vries - Электрон. текстовые дан. - Режим доступа: https://learnopengl.com/Getting-started/Camera
10. D. Gutierrez, Simulation of atmospheric phenomena [текст] / D. Gutierrez, F. J. Seron, A. Munoz, O. Anson // Elsevier. - 2006. - №30. - С. 997-1000.
11. D. Gutierrez, Rendering Ghost Ships and Other Phenomena in Arctic Atmospheres [текст] / D. Gutierrez, F. J. Seron, A. Munoz, O. Anson // Instituto de Investigación en Ingeniería de Aragón. - 2006. - №30. - С. 997-1000.
12. D. Hughes. What does a perspective projection matrix look like in OpenGL? StackExchange. [Электронный ресурс]. / D. Hughes - Электрон. текстовые дан. - Режим доступа: https://gamedev.stackexchange.com/questions/120338/what-does-a-perspective-projection-matrix-look-like-in-opengl.