|  |  |
| --- | --- |
|  | **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 Козлова Ирина Васильевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИУ7-42Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_технологическая\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия\_\_\_\_\_\_\_МГТУ им. Н. Э. Баумана, каф. ИУ7\_\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_** Козлова И. В.**\_\_\_**

*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 (кафедра ИУ7)

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Козлова Ирина Васильевна ИУ7-42Б\_\_\_\_\_

(фамилия, имя, отчество; инициалы; индекс группы)

Во время прохождения производственной практики студент должен:

1. Начать разработку программного обеспечения для визуализации удара молнии.

2. Решить вопрос о способе представления объектов, проанализировать и выбрать алгоритмы для их обработки.

3. Закрепить знания и навыки, полученные в ходе аудиторных занятий по пройденным курсам.

Дата выдачи задания « \_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Руководитель практики от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись, дата) (Фамилия И.О.)

Оглавление

[Введение 4](#_Toc81861099)

[1. Аналитическая часть 5](#_Toc81861100)

[1.1 Описание объектов сцены 5](#_Toc81861101)

[1.2 Анализ и выбор формы задания трехмерных моделей 5](#_Toc81861102)

[1.3 Анализ способа задания поверхностных моделей 6](#_Toc81861103)

[1.4 Анализ и выбор алгоритма удаления невидимых ребер и поверхностей 7](#_Toc81861104)

[1.5 Анализ и выбор модели освещения 13](#_Toc81861105)

[2. Конструкторская часть 16](#_Toc81861106)

[2.1 Общий алгоритм решения поставленной задачи 16](#_Toc81861107)

[2.2 Алгоритм обратной трассировки лучей 16](#_Toc81861108)

[2.3 Алгоритм генерации молнии 18](#_Toc81861109)

[2.4 Модель освещения Ламберта 20](#_Toc81861110)

[2.5 Генерация дома 20](#_Toc81861111)

[3. Технологическая часть 21](#_Toc81861113)

[3.1 Выбор языка программирования и среды разработки 21](#_Toc81861114)

[3.2 Структура классов программы 22](#_Toc81861115)

[3.3 Диаграмма классов 23](#_Toc81861116)

[3.4 Интерфейс программного обеспечения 23](#_Toc81861117)

[Заключение 25](#_Toc81861118)

[Список использованных источников 26](#_Toc81861119)

# Введение

В современном мире компьютерная графика используется достаточно широко. Типичная область ее применения – это кинематография и компьютерные игры.

На сегодняшний день большое внимание уделяется алгоритмам получения реалистичного изображения. Такие алгоритмы являются одними из самых затратных по времени, потому что они должны предусматривать множество физических явлений, таких как преломление, отражение, рассеивание света. Для создания еще более реалистичного изображения также учитывается дифракция, вторичное, троичное отражение света, поглощение.

Можно заметить, что чем качественнее мы получаем изображение на выходе алгоритма, тем больше времени и памяти мы используем для синтеза. Это и становится проблемой при создании динамической сцены, так как на каждом временном интервале необходимо производить расчеты заново.

Целью данной работы является обоснование выбора алгоритмов, которые можно использовать для получения реалистичного изображения удара молнии, их практическая реализация и адаптация (при необходимости) к условиям решаемой задачи.

# Аналитическая часть

## 1.1 Описание объектов сцены

Сцена состоит из источника света, молнии, дома и плоскости земли.

Источник свет представляет собой материальную точку, испускающую лучи света во все стороны (если источник расположен в бесконечности, то он имеет направление). В моей программе источником света будет молния.

Молния представляет собой ломаную линию, которая имеет начало и конец, а также несколько ветвей.

Дом – сооружение, для которого пользователь должен задать этажность, а также указать, в каких окнах включен свет.

Плоскость земли – это некая ограничивающая плоскость. Предполагается, что под такой плоскостью не расположено никаких объектов. Располагается на максимальной координате по оси У.

## 1.2 Анализ и выбор формы задания трехмерных моделей

Отображением формы и размеров объектов являются модели.

Обычно используются три формы задания моделей:

1. Каркасная (проволочная) модель

Одна из простейших форм задания модели, так как мы храним информацию только о вершинах и ребрах нашего объекта. Недостаток данной формы состоит в том, что модель не всегда точно передает представление о форме объекта.

1. Поверхностная модель

Такой тип модели часто используется в компьютерной графике. Поверхности можно задавать разными способами: либо аналитически, либо задавать участки поверхности, как поверхность того или иного вида (использовать полигональную аппроксимацию). Недостаток данной формы состоит в том, что мы не знаем с какой стороны находится материал.

1. Объемная (твердотельная) модель

Данная форма отличается от поверхностной тем, что у нас есть информация о том, где расположен материал. Это делается с помощью указания направления внутренней нормали.

Таким образом, можно сделать вывод о том, что для решение данной задачи нам подойдут поверхностные модели, так как каркасные модели могут привести к неправильному восприятию формы, а объемные модели будут излишеством, так как будут тратить больше памяти.

## 1.3 Анализ способа задания поверхностных моделей

Также необходимо определить каким образом лучше всего задавать поверхностные модели:

Аналитическим способом. Этот способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра;

Полигональной сеткой. Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной компьютерной графике.

Для более верного выбора также следует перечислить способы хранения информации о сетке:

* Список граней. Объект – это множество граней и множество вершин. В каждую грань входят как минимум 3 вершины;
* «Крылатое» представление. Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые её касаются;
* Полурёберные сетки. То же «крылатое» представление, но информация обхода хранится для половины грани;
* Таблица углов. Таблица, хранящая вершины. Обход заданной таблицы неявно задаёт полигоны. Такое представление более компактно и более производительно для нахождения полигонов, но, в связи с тем, что вершины присутствуют в описании нескольких углов, операции по их изменению медленны.
* Вершинное представление. Хранятся лишь вершины, которые указывают на другие вершины. Простота представления даёт возможность проводить над сеткой множество операций.

Стоит отметить, что одним из решающих факторов в выборе способа задания модели в данном проекте является скорость выполнения преобразований над объектами сцены.

При реализации программного продукта наиболее удобным представлением является модель, заданная полигональной сеткой – это поможет избежать проблем при описании сложных моделей. При этом способ хранения полигональной сетки – список граней, так как он предоставляет явное описание граней, что поможет при реализации алгоритма удаления невидимых рёбер и поверхностей. Также этот способ позволит эффективно преобразовывать модели, так как структура будет включать в себя список вершин.

## 1.4 Анализ и выбор алгоритма удаления невидимых ребер и поверхностей

Перед выбором алгоритма удаления невидимых ребер выделим несколько свойств, которыми должен обладать выбранный алгоритм, чтобы обеспечить оптимальную работу и реалистичное изображение.

Свойства:

* Алгоритм может работать как в объектном пространстве, так и в пространстве изображений.
* Алгоритм должен быть достаточно быстрым и использовать мало памяти.
* Алгоритм должен иметь высокую реалистичность изображения.

**Алгоритм, использующий Z-буфер**

Суть данного алгоритма – это использование двух буферов: буфера кадра, в котором хранятся атрибуты каждого пикселя, и Z-буфера, в котором хранятся информация о координате Z для каждого пикселя.

Первоначально в z-буфере находятся минимально возможные значения , а в буфере кадра располагаются пиксели, описывающие фон. Каждый многоугольник преобразуется в растровую форму и записывается в буфер кадра.

В процессе подсчета глубины нового пикселя, он сравнивается с тем значением, которое уже лежит в z-буфере. Если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то он заносится в буфер кадра и происходит корректировка z-буфера.

Для решения задачи вычисления глубины каждый многоугольник описывается уравнением При многоугольник для наблюдателя вырождается в линию.

Для некоторой сканирующей строки , поэтому имеется возможность рекуррентно высчитывать для каждого :

Получим*: ,* так как

При этом стоит отметить, что для невыпуклых многогранников предварительно потребуется удалить нелицевые грани.

Положительными моментами в этом алгоритме являются:

* простота реализации;
* оценка трудоемкости линейна.

Недостаток:

* сложная реализация прозрачности;
* большой объем требуемой памяти.

Вывод:

Данный алгоритм не подходит для решения поставленной задачи, так как требует большой объем памяти, что не удовлетворяет требованиям.

**Алгоритм обратной трассировки лучей**

Суть данного алгоритма состоит в том, что наблюдатель видит объект с помощью испускаемого света, который согласно законам оптики доходит до наблюдателя некоторым путем. Отслеживать пути лучей от источника к наблюдателю неэффективно с точки зрения вычислений, поэтому наилучшим способом будет отслеживание путей в обратном направлении, то есть от наблюдателя к объекту.

Положительными моментами в этом алгоритме являются:

* высокая реалистичность синтезируемого изображения;
* работа с поверхностями в математической форме;
* вычислительная сложность слабо зависит от сложности сцены.

Недостаток:

* производительность.

Вывод:

Данный алгоритм не отвечает главному требованию – скорости работы, но при некоторой адаптации можно добиться большей скорости работы.

**Алгоритм Робертса**

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами.

Алгоритм выполняется в 3 этапа:

1. Этап подготовки исходных данных. На данном этапе должна быть задана информация о телах. Для каждого тела сцены должна быть сформирована матрица тела . Размерность матрицы - , где – количество граней тела.

Каждый столбец матрицы представляет собой четыре коэффициента уравнения плоскости , проходящей через очередную грань.

Таким образом, матрица тела будет представлена в следующем виде:

Матрица тела должна быть сформирована корректно, то есть любая точка, расположенная внутри тела, должна располагаться по положительную сторону от каждой грани тела. В случае, если для очередной грани условие не выполняется, соответствующий столбец матрицы надо умножить на .

2. Этап удаления рёбер, экранируемых самим телом.

На данном этапе рассматривается вектор взгляда .

Для определения невидимых граней достаточно умножить вектор на матрицу тела . Отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

3. Этап удаления невидимых рёбер, экранируемых другими телами сцены.

На данном этапе для определения невидимых точек ребра требуется построить луч, соединяющий точку наблюдения с точкой на ребре. Точка будет невидимой, если луч на своём пути встречает в качестве преграды рассматриваемое тело.

Положительным моментом этого алгоритма являются:

* работа в объектном пространстве;
* высокая точность вычисления.

Недостатки:

* рост сложности алгоритма – квадрат числа объектов;
* тела сцены должны быть выпуклыми (усложнение алгоритма, так как нужна будет проверка на выпуклость);
* сложность реализации.

Вывод:

Алгоритм Робертса не подходит для решения поставленной задачи по следующей причине:

* высокая сложность реализаций как самого алгоритма, так и его модификаций, отсюда низкая производительность.

**Алгоритм художника**

Данный алгоритм работает аналогично тому, как художник рисует картину – то есть сначала рисуются дальние объекты, а затем более близкие. Наиболее распространенная реализация алгоритма – сортировка по глубине, которая заключается в том, что произвольное множество граней сортируется по ближнему расстоянию от наблюдателя, а затем отсортированные грани выводятся на экран в порядке от самой дальней до самой ближней. Данный метод работает лучше для построения сцен, в которых отсутствуют пересекающиеся грани.

Положительным моментом данного алгоритма является:

* требование меньшей памяти, чем, например, алгоритм Z-буффера.

Недостатки

* недостаточно высока реалистичность изображения;
* сложность реализации при пересечения граней на сцене.

Вывод:

Данный алгоритм не отвечает главному требованию – реалистичность изображения. Также алгоритм художника отрисовывает все грани (в том числе и невидимые), на что тратится большая часть времени.

**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.

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 трех случаев не имеет места, то снова разбиваем область на четыре равные части и проверяем выполнение этих условий для каждой из частей. Те части, для которых таким образом не удалось установить видимость, разбиваем снова и т. д.

Преимущества:

* меньшие затраты по времени в случае области, содержащий мало информации.

Недостатки:

* алгоритм работает только в пространстве изображений;
* большие затраты по времени в случае области с высоким информационным содержимым.

Вывод:

Данный алгоритм не отвечает требованию работы как в объектном пространстве, так и в пространстве изображений, а также возможны большие затраты по времени работы.

**Вывод**

Для удаления невидимых линий выбран алгоритм обратной трассировки лучей. Данный алгоритм позволит добиться максимальной реалистичности и даст возможность смоделировать распространение света в пространстве, учитывая законы геометрической оптики. Данный алгоритм можно модернизировать, добавив в него обработку новых световых явлений. Также этот алгоритм позволяет строить качественные тени с учетом большого числа источников. Стоит отметить тот факт, что алгоритм трассировки лучей не требователен к памяти, в отличие, например, от алгоритма Z-буфера.

## 1.5 Анализ и выбор модели освещения

Физические модели материалов стараются аппроксимировать свойства некоторого реального материала. Такие модели учитывают~~ся~~ особенности поверхности материала или же поведение частиц материала.

Эмпирические модели материалов устроены иначе, чем физически обоснованные. Данные модели подразумевают некий набор параметров, которые не имеют физической интерпретации, но которые позволяют с помощью подбора получить нужный вид модели.

Рассмотрим эмпирические модели, а конкретно модель Ламберта и модель Фонга.

**Модель Ламберта**

Модель Ламберта моделирует идеальное диффузное освещение, то есть свет при попадании на поверхность рассеивается равномерно во все стороны. При такой модели освещения учитывается только ориентация поверхности (N) и направление источника света (L) (рисунок 1).

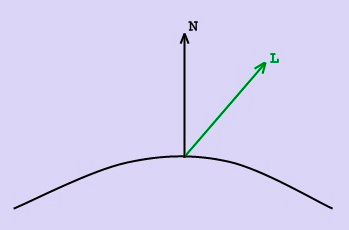


Рисунок 1. Направленность источника света

Эта модель является одной из самых простых моделей освещения и очень часто используется в комбинации с другими моделями. Она может быть очень удобна для анализа свойств других моделей, за счет того, что ее легко выделить из любой модели и анализировать оставшиеся составляющие.

**Модель Фонга**

Это классическая модель освещения. Модель представляет собой комбинацию диффузной и зеркальной составляющих. Работает модель таким образом, что кроме равномерного освещения на материале могут появляться блики. Местонахождение блика на объекте определяется из закона равенства углов падения и отражения. Чем ближе наблюдатель к углам отражения, тем выше яркость соответствующей точки.

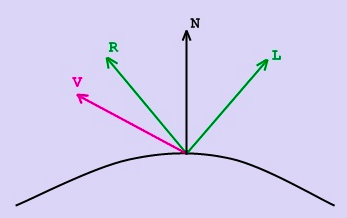


Рисунок 2 Направление источника света, отраженного луча и наблюдателя

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения (рисунок 2). Нормаль делит угол между лучами на две равные части. L – направление источника света, R – направление отраженного луча, V – направление на наблюдателя.

**Вывод**

Для освещения была выбрана модель Ламберта из-за своей простоты по сравнению с моделью Фонга. Для расчета данных модели Ламберта необходимо меньше вычислений, а значит меньше времени.

# 2. Конструкторская часть

## 2.1 Общий алгоритм решения поставленной задачи

1. Задать объекты сцены (дом, окна, в которых включен свет).
2. Задать ионизацию облака.
3. Рассчитать координаты конца молнии, и в зависимости от этого рассчитать куда ударит молния.
   1. Если конец молнии находится в области дома, то молния ударит в молниезащиту дома.
   2. Если конец молнии находится вне области дома, то молния ударит в землю в случае, если молния-лидер.
4. Изобразить молнию.

## 2.2 Алгоритм обратной трассировки лучей

Алгоритмы трассировки лучей на сегодняшний день считаются наиболее мощными при создании реалистичных изображений.

Изображение формируется из-за того, что свет попадает в камеру. Выпустим из источников света множество лучей (первичные лучи). Часть этих лучей “улетит” в свободное пространство, а часть попадет на объекты. На них лучи могут преломляться и отражаться. При этом часть энергии луча поглотится. Преломленные и отраженные лучи образуют новое поколение лучей. Далее эти лучи опять же преломятся, отразятся и образуют новое поколение лучей. В конечном итоге часть лучей попадет в камеру и сформирует изображение.

Существуют алгоритмы, работающие по такому ~~алгоритму~~ принципу, но они крайне неэффективны, так как большинство лучей, которые исходят из источника, не попадают в камеру. А приемлемая картинка получается, если трассировать большое число лучей, что займет очень много времени. Данный алгоритм называется прямой трассировкой лучей.

Метод обратной трассировки лучей позволяет значительно сократить перебор световых лучей. В этом методе отслеживаются лучи не от источников, а из камеры. Таким образом, трассируется определенное число лучей, равное разрешению картинки.

Предположим, что у нас есть камера и экран, находящийся на расстоянии h от нее (рисунок 3). Разобьем экран на квадратики. Дальше будем по очереди проводить лучи из камеры в центр каждого квадратика (первичные лучи). Найдем пересечение каждого такого луча с объектами сцены и выберем среди всех пересечений самое близкое к камере. Далее, применив нужную модель освещения, можно получить изображение сцены. Это самый простой метод трассировки лучей. Он позволяет лишь отсечь невидимые грани.

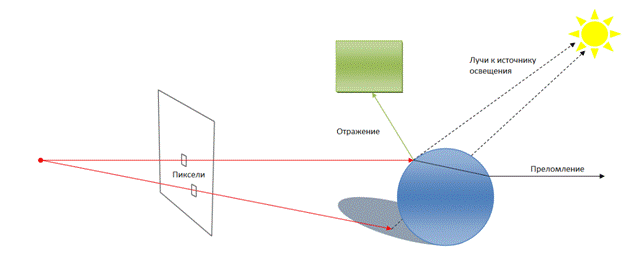


Рисунок 3 Трассировка лучей

Но если надо смоделировать такое явление как отражение, то необходимо из самого близкого пересечения пустить вторичные лучи. Например, если поверхность отражает свет, и она идеально ровная, то необходимо отразить первичный луч от поверхности и пустить по этому направлению вторичный луч. Если же поверхность неровная, то необходимо пустить множество вторичных лучей.

Для уменьшения времени работы алгоритма в моей программе необходимо испускать лучи из камеры не по всей сцене, а в отдельные ее участки. Во-первых, лучи пускаются во все сегменты молнии для того, чтобы узнать, где находится молния за домом или перед ним, также лучи пускаются в черные окна, чтобы затем пустить из них вторичный луч и понять отражается в них молния или нет. Таким образом можно получить большой выигрыш по времени.

## 2.3 Алгоритм генерации молнии

Первоначально случайным образом задаются две координаты на молнии, ее конец и начало. По данным двум точками строим прямую, путем вычитания из координат конца координаты начала молнии, также находим расстояние от нее до громоотвода. Если расстояние это меньше нужного, то нужно поменять координаты конца на координаты вершины громоотвода.

Существует два вида молнии:

1. Обычная молния – это молния, которая не доходит до объекта или земли (рисунок 4).
2. Молния лидер – это молния, которая доходит до какого-то объекта либо до земли (рисунок 5).

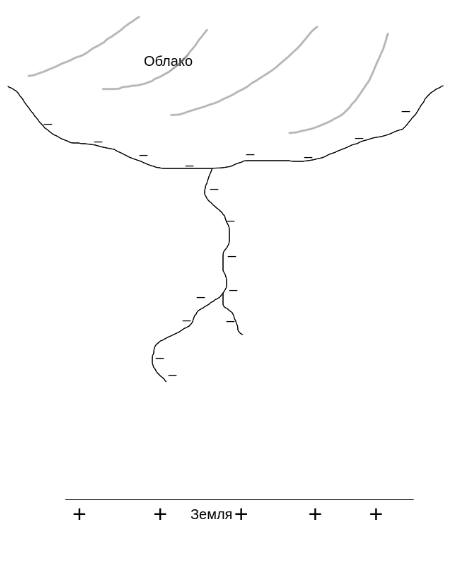


Рисунок 4 Обычная молния

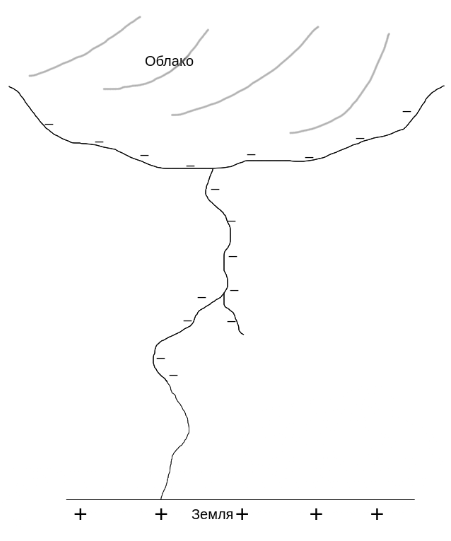


Рисунок 5 Молния-лидер

Далее генерацию молнии можно разделить на два случая.

1. Молния бьет в землю – в данном случае молния имеет более непредсказуемый характер и может себя вести произвольно.
2. Молния бьет в громоотвод – в данном случае молния движется от начала удара до вершины громоотвода с небольшими колебаниями.

Каждую итерацию каждый сегмент делится пополам, с небольшим сдвигом центральной точки.

Чтобы создать ветви, когда разделяем сегмент молнии, вместо добавления двух сегментов надо добавить три. Третий сегмент – это продолжение молнии в направлении первого с небольшим отклонением.

На каждом сегменте с вероятностью в 1% появляется побочная ветвь, которая строится по таким же законам, как и главная в том случае, если молния бьет в землю. Для каждой такой побочной ветви генерируется угол на который она повернута относительно главной ветви. Длина побочного сегмента зависит от того, в каком месте молнии он~~а~~ появляется: чем ближе к концу, тем короче он~~а~~ будет.

## 2.4 Модель освещения Ламберта

Данная модель вычисляет цвет поверхности в зависимости от того как на нее светит источник света. Согласно данной модели, освещенность точки равна произведению силы источника света и косинуса угла, под которым он светит на точку.

## 2.5 Генерация дома

Дом удобнее генерировать с помощью массива точек, ограничивающих сторону дома.

Дом состоит из 5 объектов – 4 стороны и крыша. Они задаются путем задания координат для каждой стороны. Для каждой координаты задается три параметра – координаты X, Y, Z. Высота дома зависит от этажности. После задания данных параметров создаются и накладываются окна.

Каждое окно, как и сторона дома, ограничено массивом точек. Для каждого окна задаются ограничивающие его 4 точки. Задаются они путём задания трёх координат для каждой стороны. В зависимости от количества этажей создаются окна. На каждый этаж приходится по 8 окон.

Также создается громоотвод (молниезащита дома).

# 3. Технологическая часть

## 3.1 Выбор языка программирования и среды разработки

Существует множество языков, а также сред программирования, многие из которых обладают достаточно высокой эффективностью, удобством и простотой в использовании. Для разработки данной программы был выбран язык C#. Данный выбор обусловлен следующими факторами:

* Этот язык предоставляет программисту широкие возможности реализации самых разнообразных алгоритмов. Он обладает высокой эффективностью и большим набором стандартных классов и процедур.
* C# является полностью объектно-ориентированным. Он позволяет использовать множественное наследование, абстрактные и параметризованные классы.
* ~~Так как~~ трехмерные объекты, также как и математические абстракции, естественным образом представляются в виде объектов классов, что позволяет легко и эффективно организовывать их взаимодействие, при этом сохраняется читаемый и легко изменяемый код.

В качестве среды разработки была выбрана Visual Studio 2019. Некоторые факторы по которым была выбрана данная среда:

* Включает весь основной функционал: параллельная сборка, отладчик, поддержка точек останова, сборки и т.д.
* Разработчики имеют возможность расширить любой функционал, включая компиляцию, отладку.
* Работает с интерфейсом Windows Forms, который очень удобен в использовании, а также позволяет без проблем создавать приложения.
* Данная среда разработки бесплатна для студентов.

## 3.2 Структура классов программы

Так как при написании программы используется язык C#, а это объектно-ориентированный язык, то особое внимание уделено структуре классов.

Условно классы в программе можно разделить на несколько групп по выполняемым функциям.

* Математические абстракции
  + Trace - структура, в которую входит точка пересечения луча с объектом расстояние от камеры до точки пересечения.
  + Ray - трехмерный луч, задающийся точкой начала луча, направляющим вектором.
  + Comparator - помогает сравнивать сегменты молнии.
* Вспомогательные классы свойств трехмерных объектов
  + Texture - абстрактный класс с основными свойствами.
  + SimpleTexture - обеспечивает загрузку из файла текстуры, ее интерпретацию на простую поверхность.
* Трехмерные объекты
  + House – реализует работу с домом, генерацию, трехмерные преобразования.
  + Lightning – реализует работу с молнией, генерацию, трёхмерные преобразования.
* Источники света
  + Shadow - класс, позволяющий работать с тенями.
* Сцена
  + Scene - характеризует набор объектов и их свойств.
* Алгоритмы визуализации
  + Highlight - отрисовка вспышки.
  + Lightning - отрисовка молнии.
  + House - отрисовка дома.
  + Shadow - отрисовка тени.
  + Texture - отрисовка текстуры.
  + Window - отрисовка окна.
* Интерфейс пользователя
  + Взаимодействие с интерфейсом происходит через диалоговые окна, которые в свою очередь взаимодействуют с классом Scene.

## 3.3 Диаграмма классов

Диаграмма классов для данной работы представлена на рисунке 6.

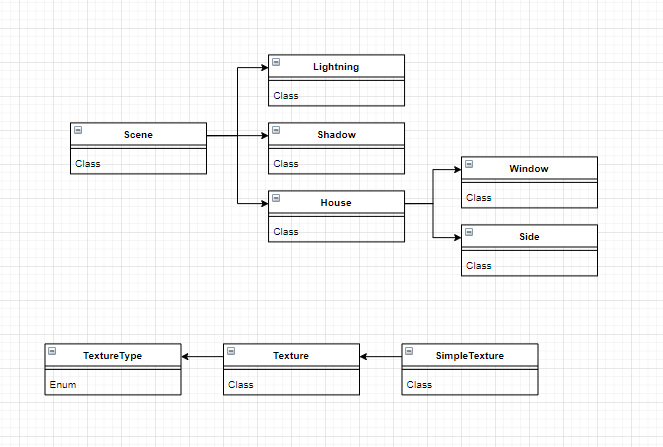


Рисунок 6 Диаграмма классов

## 3.4 Интерфейс программного обеспечения

Пуск/Стоп - кнопка для запуска/остановки генерации молнии.

Рендер - кнопка для создании изображения.

Включить и выключить свет - кнопки, которые включают и выключают свет во всех окнах.

Изменить количество этажей - кнопка, которая изменяет количество этажей в доме. Меняет количество этажей у дома в зависимости от введённого значения, при это генерируя новую сцену.

Также есть возможность взаимодействовать с программой при помощи стрелок вправо/влево на клавиатуре или при помощи мышки (нажать на сцену и не отпуская кнопку плавно ввести курсор вправо/влево).

Интерфейс представлен на рисунке 7.

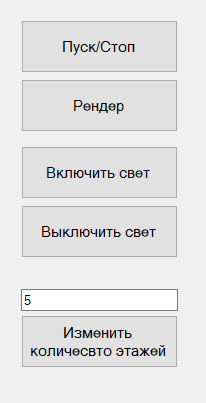


Рисунок 7 Интерфейс

# 

# Заключение

Во время выполнения поставленной задачи были проанализированы основные способы представления и задания трехмерных моделей, а также рассмотрены и основные алгоритмы удаления невидимых линий и методы освещения. Проанализированы достоинства и недостатки представленных алгоритмов и выбраны наиболее оптимальные для решения поставленной задачи.

Проделанная работа помогла закрепить полученные навыки в области компьютерной графики и проектирования программного обеспечения. Реализация программы позволяет легко и быстро добавить необходимые новые объекты (классы). Пользователь может задавать необходимые параметры для дома, а также ионизацию облака.

Стоит отметить, что при выполнении данной работы удалось изучить язык C#, а также познакомиться и изучить возможности среды разработки Visual Studio 2019.

# Список использованных источников

1. Дёмин А.Ю., Основы компьютерной графики: учебное пособие – Томск: Изд-во Томского политехнического университета, 2011. – 191 с.

2. Авдеева С.М., Куров А.В. Алгоритмы трехмерной машинной графики: учебное пособие. - М.: Издательство МГТУ им. Н.Э. Баумана, 1996. - 60 с., ил.

3. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. – М.: Диалог-МИФИ, 1995. – 288 с.

4. Удаление скрытых линий и поверхностей [Электронный ресурс]. – Режим доступа: <https://algolist.manual.ru/graphics/delinvis.php> (дата обращения 02.07.21)

5. Роджерс Д., Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.— 512 с.: ил.