|  |  |
| --- | --- |
|  | **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.*

**Индивидуальное задание:**

Разработать программное обеспечение для трехмерной визуализации удара молнии рядом с домом в ночное время и работы молниезащиты. Выбрать методы построения реалистичных изображений и освещения от молнии.

Оглавление

[Введение 4](#_Toc81473699)

[1.Аналитическая часть 5](#_Toc81473700)

[1.1 Описание объектов сцены 5](#_Toc81473701)

[1.2 Анализ и выбор формы задания трехмерных моделей 5](#_Toc81473702)

[1.3 Анализ способа задания поверхностных моделей 6](#_Toc81473703)

[1.4 Анализ и выбор алгоритма удаления невидимых ребер и поверхностей 7](#_Toc81473704)

[1.5 Анализ и выбор модели освещения 10](#_Toc81473705)

[2. Конструкторская часть 12](#_Toc81473706)

[2.1 Общий алгоритм решения поставленной задачи 12](#_Toc81473707)

[2.2 Алгоритм обратной трассировки лучей 12](#_Toc81473708)

[2.3 Алгоритм генерации молнии 14](#_Toc81473709)

[2.4 Модель освещения Ламберта 16](#_Toc81473710)

[2.5 Генерация дома 16](#_Toc81473711)

[3. Технологическая часть 17](#_Toc81473712)

[3.1 Выбор языка программирования и среды разработки 17](#_Toc81473713)

[3.2 Структура классов программы 18](#_Toc81473714)

[3.3 Диаграмма классов 19](#_Toc81473715)

[3.4 Интерфейс программного обеспечения 19](#_Toc81473716)

[Заключение 21](#_Toc81473717)

[Список использованных источников 22](#_Toc81473718)

# Введение

В современном мире компьютерная графика является одной из важнейших частей нашей жизни. Она используется для наглядного описания и отображения данных, для создания красочных эффектов в кино и в компьютерных играх.

На сегодняшний день большое внимание уделяется алгоритмам получения реалистичного изображения. Такие алгоритмы являются одними из самыми затратными 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.

Цель моей практики - это выбор оптимальных алгоритмов или модификация уже существующих, а также реализация этих алгоритмов для получения реалистичного изображения удара молнии.

# Аналитическая часть

## 1.1 Описание объектов сцены

Сцена состоит из:

Источника света - представляет собой материальную точку, из которой исходят 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 соединяет начало и конец, а также имеет разветвленности.

Дом - сооружение, у которого пользователь должен задать этажность, а также в каких окнах свет включен.

Плоскость земли - 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. Главное назначение модели - правильно отображать форму и размеры определенного объекта.

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:

Аналитический способ - это способ задание модели характеризуется описанием объекта, доступное в неявной форме. То есть необходимо будет вычислять некую дополнительную функцию, чтобы получить визуальное представление модели.

С помощью полигональной сетки - это 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:

* “Крылатое” представление - это способ, в котором каждая точка ребра указывает на две грани, две вершины, четыре ребра. Данный способ позволяет обрабатывать поверхность за постоянное время, но требует очень большие объемы памяти.
* Вершинное представление - это способ, в котором представлены лишь вершины, которые указывают на другие вершины. Информация о гранях и ребрах выражена неявно, но простота позволяет множество операций.
* Список граней - это способ, в котором описание граней происходит при помощи указателей в списке вершин.
* Полуреберные сетки - это способ похожий на “крылатое” представление, отличие лишь в том, что мы делаем обход лишь половины грани.

Наиболее удобным способом хранения моей сцены - это список граней, так как можно эффективно преобразовывать данные, а также представление позволяет явный поиск вершин граней, которые окружают вершину.

## 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, 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.

Изначально в Z-буфере находятся минимальные значения Z, а в буфере кадра - информация о пикселях, которые описывают фон. Глубина каждого нового пикселя при подсчете сравнивается со значением, которое уже есть в Z-буфере. В случае, если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то информация о нем заносится в буфер кадра и происходит редактирование Z-буфера.

Положительной стороной данного алгоритма является простота реализация, экономия времени, так как нет сортировки.

Недостатком - большой объем требуемой памяти, а также сложная реализация эффектов прозрачности.

**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. Алгоритм называется “обратной трассировкой” из-за того, что эффективнее выходит отслеживать лучи, которые идут от наблюдателя к объекту.

Положительным моментом в этом алгоритме является возможность использования в параллельных вычислительных системах.

Недостатком же является то, что требуется большое количество необходимых вычислений.

**Алгоритм Робертса**

Данный алгоритм решает задачу удаления невидимых ребер и граней только с выпуклыми телами, работает только в объектном пространстве.

У алгоритма три этапа выполнения. Этап 0 - подготовка данных. Этап 1 - удаление ребер, экранируемых самим телом. Этап 2 - удаление невидимых ребер, экранируемых другими телами сцены.

Положительным моментом в этом алгоритме является высокая точность вычислений.

Недостатком же является то, что все тела должны быть выпуклыми, а также

**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 - сортировк 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 является то, что данный алгоритм требует памяти меньше, чем, например, алгоритм Z-буфера.

Недостатком же является то, что у него недостаточно высока реалистичность изображения.

**Алгоритм Варнока**

Данный алгоритм работает в пространстве изображения. Основная задача - это рекурсивное разбиения экрана. То есть нам необходимо понять, что изображается в очередном окне, а также понять, надо ли нам это изображать или нет. Это и является одновременно положительной и отрицательной стороной алгоритма. Скорость данного алгоритма зависит от количества пересечения объектов.

**Вывод**

Для удаления невидимых линий выбран алгоритм обратной трассировки лучей. Данный алгоритм позволит добиться максимальной реалистичности, а также он позволяет 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 обоснования модели материалов стараются аппроксимировать свойства некоторого реального материала. Такие модели учитываются 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).

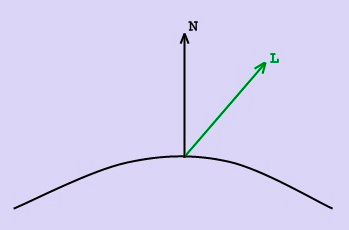


Рисунок 1Направленность источника света

Эта модель является одной из самых простых моделей освещения, 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 яркость соответствующей точки повышается.

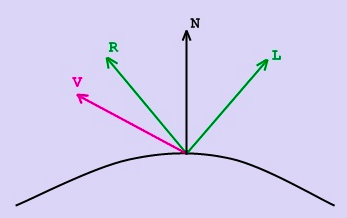


Рисунок 2 Направление источника света, отраженного луча и наблюдателя

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точки 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 это 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, 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.

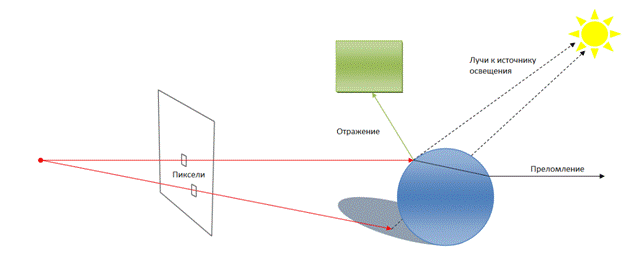


Рисунок 3 Трассировка лучей

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

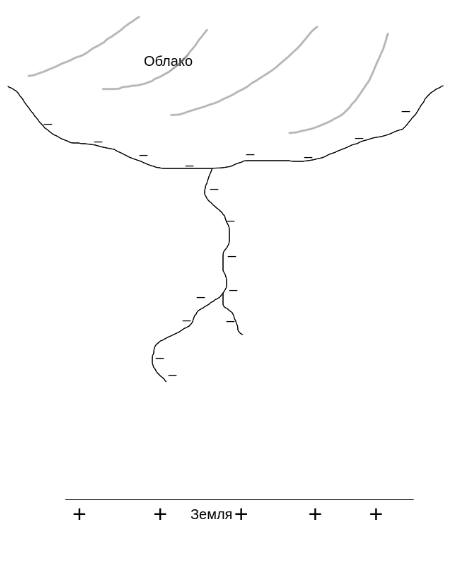


Рисунок 4 Обычная молния

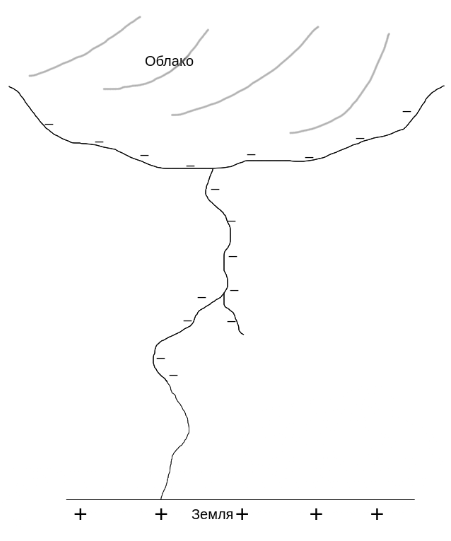


Рисунок 5 Молния-лидер

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, что 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 она появляется, чем ближе к концу, тем короче она 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, ограничивающий сторону дома.

Дом состоит из 5 сторон - 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 массивов точек. Для каждого окна задаются 4 точки, по три координаты для каждой точки. В среднем на каждый этаж приходится около 8 этажей.

Также создается громоотвод (молниезащита дома).

# 3. Технологическая часть

## 3.1 Выбор языка программирования и среды разработки

Существует множество языков, а также сред программирования, многие 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.

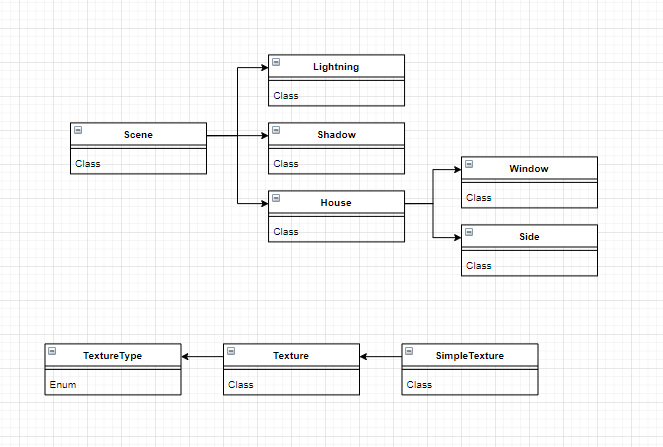


Рисунок 6 Диаграмма классов

## 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, также рассмотрены и основные алгоритмы удаления невидимых линий и методы освещения. Рассмотрены достоинства и недостатки представленных алгоритмов и выбраны наиболее оптимальные для решения поставленной задачи.

Проделанная работа помогла закрепить полученные навыки в области компьютерной графике 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.