Оглавление

Введение

На сегодняшний день компьютерная графика является одним из самых быстрорастущих сегментов в области информационных технологий.

Область ее применения очень широка и не ограничивается художественными эффектами: в отраслях техники, науки, медицины и архитектуры трехмерные графические объекты используются для наглядного отображения разнообразной информации и презентации различных проектов.

Алгоритмы создания реалистичных изображений требуют особого внимания, поскольку они связаны с внушительным объемом вычислений, требуют больших компьютерных ресурсов и затратны по времени. Для создания качественного изображения объекта следует учесть не только оптические законы, но и расположение источника света, фактуры поверхностей.

Основным направлением в развитии компьютерной графики является ускорение вычислений и создание более качественных изображений.

Цель рабочей пояснительной записки — разработка, реализация, описание программного обеспечения, генерирующее изображение стержня помещенного в цилиндр с жидкостью.

Задачи рабочей пояснительной записки:

- описать структуру трехмерной сцены, включая объекты, из которых она состоит;
- проанализировать существующие алгоритмы построения изображения и обосновать выбор тех из них, которые в наибольшей степени подходят для решения поставленной задачи;
- проанализировать и выбрать варианты оптимизации ранее выбранного алгоритма удаления невидимых линий;
- реализовать выбранные алгоритмы;
- разработать программное обеспечение для отображения сцены и визуализации стержня в цилиндре, наполненного жидкостью;

1 Аналитический раздел

В этом разделе будут представлены описание объектов, а также обоснован выбор алгоритмов, которые будут использован для ее визуализации.

1.1 Описание объектов сцены

Сцена состоит из источника света, цилиндра, жидкости, стержня и плоскости.

Источник света представляет собой материальную точку, пускающую лучи света во все стороны (если источник расположен в бесконечности, то лучи идут параллельно). Источником света в программе является вектор.

Цилиндр — это тонкостенный прозрачный объект, в котором располагается два других объекта - жидкость, стержень.

Жидкость — это объект, который тоже является прозрачным тонкостенным цилиндром.

Стержень — это непрозрачный прямоугольный параллелепипед. Служит для того чтобы отобразить на экране преломление твердого тела в жидкости.

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

1.2 Обоснование выбора 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 модели состоит в том, что она не всегда точно передает представление о форме объекта.

2. Поверхностная модель.

   Поверхностная модель объекта — это оболочка объекта, пустая внутри. Такая информационная модель содержит данные только о внешних геометрических параметрах объекта. Такой тип модели часто используется в компьютерной графике. При этом могут использоваться различные типы поверхностей, ограничивающих объект, такие как полигональные модели, поверхности второго порядка и др.

3. Объемная (твердотельная) модель.

   При твердотельном моделировании учитывается еще материал, из которого изготовлен объект. То есть у нас имеется информация о том, с какой стороны поверхности расположен материал. Это делается с помощью указания направления внутренней нормали.

При решении данной задачи будет использоваться объемная модель. Этот выбор обусловлен тем, что каркасные модели могут привести к неправильному восприятию формы объекта, а поверхностные модели не подходят, так как важен материал из которого сделаны объекты сцены.

1.3 Задание объемных моделей

После выбора модели, необходимо выбрать лучший способ представления объемной модели.

Аналитический способ — 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.

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 представлением является аналитический способ, так как все объекты в сцене являются простыми геометрическими фигурами.

1.4 Выбор алгоритма удаления невидимых ребер и поверхностей

Перед выбором алгоритма удаления невидимых ребер необходимо выделить несколько свойств, которыми должен обладать выбранный алгоритм, чтобы обеспечить оптимальную работу и реалистичное изображение, а именно:

- алгоритм должен использовать как можно меньше памяти;
- алгоритм должен обладать небольшой относительно аналогичных алгоритмов трудоемкостью;
- алгоритм должен выдавать реалистичное изображения.

1.4.1 Алгоритм, использующий Z-буфер

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

Первоначально в Z-буфере находятся минимально возможные значения Z, 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 [1].

Для решения задачи вычисления глубины Z каждый многоугольник описывается уравнением $ax + by + cz + d = 0$. При $c = 0$ многоугольник для наблюдателя вырождается в линию.

Для некоторой сканирующей строки y=const, поэтому имеется возможность рекуррентно высчитывать z' для каждого $x' = x + dx$: $z' - z = -\frac{ax'+d}{c} + \frac{ax+d}{c} = \frac{a(x-x')}{c}$.

Получим $z' = z - \frac{a}{c}$, так как $x - x' = dx = 1$.

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

Преимущества

- простота реализации;
- оценка трудоемкости линейна.

Недостатки

- сложная реализация прозрачности;
- большой объем требуемой памяти.

Данный алгоритм не подходит для решения поставленной задачи, так как требует большой объем памяти, и плохо работает с прозрачными объектами что не удовлетворяет требованиям.

1.4.2 Алгоритм Робертса

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

Алгоритм выполняется в 3 этапа.

Этап подготовки исходных данных

На данном этапе должна быть задана информация о телах. Для каждого тела сцены должна быть сформирована матрица тела V. Размерность матрицы - $4 * n$, где n – количество граней тела.

Каждый столбец матрицы представляет собой четыре коэффициента уравнения плоскости $ax + by + cz + d = 0$, проходящей через очередную грань.

Таким образом, матрица тела будет представлена в следующем виде

$$V = \begin{pmatrix} a_1 & a_2 & \dots & a_n \\ b_2 & b_2 & \dots & b_n \\ c_2 & c_2 & \ddots & c_n \\ d_2 & d_2 & \dots & d_n \end{pmatrix} \tag{1.1}$$

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 -1.

Этап удаления рёбер, экранируемых самим телом

На данном этапе рассматривается вектор взгляда $E = \{0, 0, -1, 0\}$. Для определения невидимых граней достаточно умножить вектор E на матрицу тела V. Отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

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 [3].

Преимущества

- 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.

Данный алгоритм не подходит для решения поставленной задачи из-за высокой сложности реализации как самого алгоритма, так и его модификаций, отсюда низкая производительность.

1.4.3 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 [4].

Преимущества

- требование меньшей памяти, чем, например, алгоритм Z-буфера.

Недостатки

- недостаточно высокая реалистичность изображения;
- сложность реализации при пересечения граней на сцене.

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

1.4.4 Алгоритм Варнока

Алгоритм Варнока [5] 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, то 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, то 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.

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

1.4.5 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.

Предполагается, что сцена уже преобразована в пространство изображения, а точка, в которой находится наблюдатель, находится в бесконечности на положительной полуоси Z, и поэтому световые лучи параллельны этой же оси. При этом каждый луч проходит через центр пикселя растра до сцены. Траектория каждого луча отслеживается для определения факта пересечения определенных объектов сцены с этими лучами. При этом необходимо проверить пересечение каждого объекта сцены с каждым лучом, а пересечение с Z_{min} представляет видимую поверхность для данного пикселя [2].

Преимущества

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

Недостатки

- производительность.

Данный алгоритм подходит для реализации реалистичных прозрачных объектов

1.4.6 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. Также этот алгоритм позволяет строить качественные тени с учетом большого числа источников, и в данном алгоритме удобнее всего реализовывать прозрачные и матовые объекты, что позволит получить достаточно реалистичное изображение. 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 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.

В данной работе следует делать выбор из эмпирических моделей, а конкретно из модели Ламберта и модели Фонга.

1.5.1 Модель Ламберта

Модель Ламберта [6] 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). Иллюстрация данной модели представлена на рисунке 1.1.

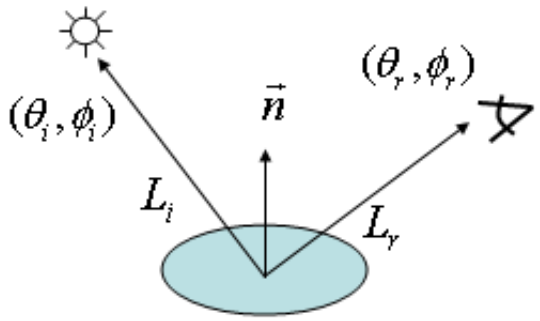

Рисунок 1.1 – Модель освещения Ламберта

Эта модель является одной из самых простых моделей освещения и очень часто используется в комбинации с другими моделями.

1.5.2 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 [6].

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения (рисунок 1.2). Нормаль делит угол между лучами на две равные части. L – направление источника света, R – направление отраженного луча, V – направление на наблюдателя.

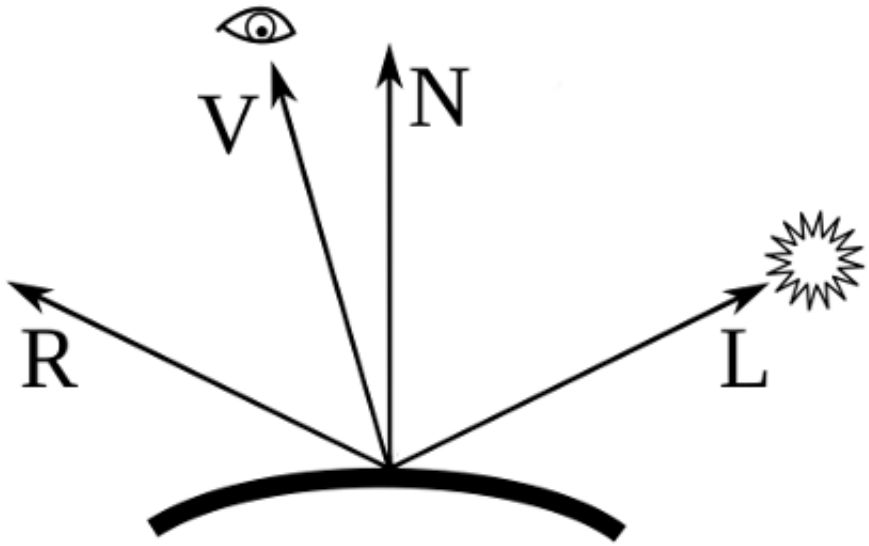

Рисунок 1.2 – Модель освещения Фонга

1.5.3 Вывод

Для освещения выбрана модель Фонга, так как изображение должно быть наиболее приближенным к реальности.

1.6 Вывод

В данном разделе был проведен анализ алгоритмов удаления невидимых линий и модели освещения, которые возможно использовать для решения поставленных задач. В качестве ключевого алгоритма выбран алгоритм обратной трассировки лучей, который будет реализован в рамках данного курсового проекта.

2 Конструкторский раздел

В данном разделе будут рассмотрены требования к программе и алгоритмы визуализации сцены и молнии.

2.1 Общий алгоритм решения поставленной задачи

1. Задать объекты сцены (цилиндр, жидкость, стержень).
2. Разместить источник света.
3. С помощью обратной трассировки лучей визуализировать обстановку.

2.2 Алгоритм обратной трассировки лучей

Алгоритмы трассировки лучей на сегодняшний день считаются наиболее приближен к реальности при создании изображений.

Изображение формируется за счет того, что отраженный свет попадает в камеру. Выпускается множество лучей (первичные) из источников света. Часть этих лучей “улетает” в свободное пространство, а часть попадает на объекты сцены, преломляясь и отражаясь, при том часть энергии лучей поглощается в зависимости от физических свойств материала объекта. Преломленные и отраженные лучи продолжают взыскиваться до предела выставленным в программе. В конечном счете часть лучей попадает в камеру и формирует изображение. Данный алгоритм называется прямой трассировкой лучей, но он крайне неэффективен, так как большинство лучей, исходящих из источника света, не попадают в камеру. Оптимизированный алгоритм трассировки лучей называется обратный. В данном алгоритме лучи отслеживаются из камеры, а не из источников света. Таким образом, программно обрабатывается меньшее количество лучей.

Предполагается, что есть камера и экран (рисунок 2.1), находящийся на определенном расстоянии от нее. Экран разбивается на пиксели, далее поочередно испускаются лучи из камеры в центр каждого пиксела. Находится пересечение каждого луча с объектами сцены и выбирается среди всех пересечении наиболее близкое к камере. Далее, применяется модель освещения и получается изображение сцены. Это самый простой метод трассировки лучей, который отсекает невидимые грани.

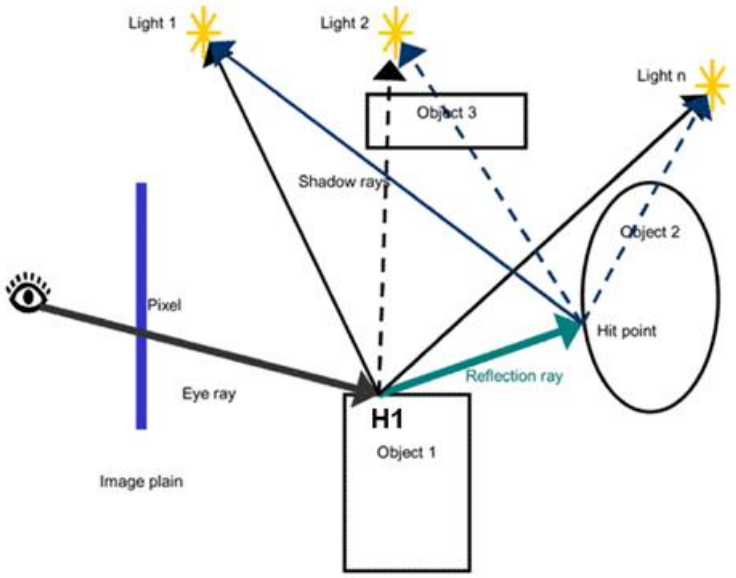

Рисунок 2.1 – Пример работы алгоритма обратной трассировки лучей

В усложненном алгоритме учитывается отражение, для этого необходимо из самого близкого пересечения пустить вторичные лучи.

2.3 Алгоритм преломления лучей в прозрачных объектах

Чтобы изобразить прозрачный объект, в его материале должны задаваться отражение и преломление (рисунок 2.2).

Если поверхность обладает отражающими свойствами, то строится вторичный луч отражения. Направление луча определяется по закону отражения (геометрическая оптика) равна

$r = i - 2 \cdot n \cdot (n \cdot i)$.

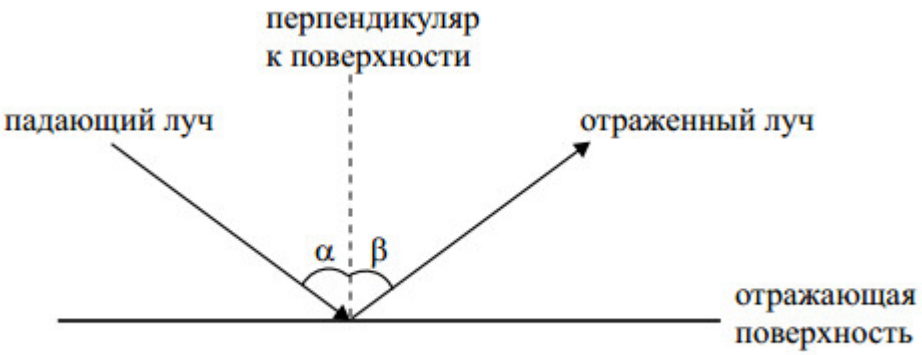

Рис. 8.

Рисунок 2.2 – Направление луча по закону отражения

Если же поверхность прозрачна, то строится луч прозрачности. Для определения направления луча используется закон преломления (рисунок 2.3).

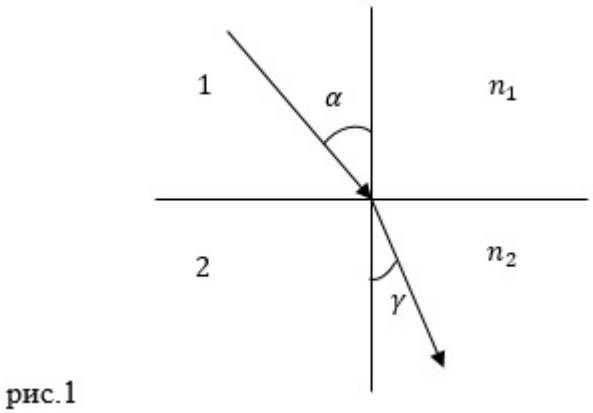

рис.1

Рисунок 2.3 – Направление луча по закону преломления

2.4 Алгоритм генерации молнии

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

Существует два вида молнии.

1. Обычная молния – это молния, которая не доходит до объекта или земли (рисунок 2.4).

2. Молния-лидер – это молния, которая доходит до какого-то объекта либо до земли (рисунок 2.5).

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 она появляется: чем ближе к концу, тем короче она будет.

Пример генерации молнии без побочных сегментов (ветвей) представлен на рисунке 2.6 и с побочными сегментами (ветвями) – на рисунке 2.7.

2.5 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 [6].

$$I_d = k_d cos(L, N) i_d, \tag{2.1}$$

где:

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

2.6 Визуализация изображения дома

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

Дом состоит из 5 объектов – 4 сторон и крыши. 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).

Габаритные размеры дома, а именно ширина и длина задаются константами. Пользователь может изменить количество этажей дома, от которого зависит его высота.

2.7 Выбор используемых типов и структур данных

Для разрабатываемого ПО необходимо реализовать следующие типы и структуры данных.

1. Сцена - список объектов.

```
private Shadow _s; // Тень
private readonly PictureBox _picture; // Поле для отрисовки
private readonly Point3D _center; // Центр картины
private House _house; // Дом
private Lightning _light; // Молния
public int Kol; // Количество этажей
```

2. Объекты сцены - набор вершин и граней.

   - Дом;

     ```
     public readonly Side[] S; // Стороны
     public Window[][] W; // Окна
     private readonly Point3D _center; // Центр дома
     private readonly int _kol; // Количество этажей
     ```

   - Молния;

     ```
     public readonly List<Point3D> Model; // Главная ветвь
     public readonly List<List<Point3D>> SubModels;
     // Массив побочных ветвей
     ```

   - Тень;

     ```
     // Массив многоугольников, которые рисуют тени
     private readonly PointF[][] _sh;
     ```

   - Окно;

     ```
     public readonly Point3D[] Points; // Массив вершин
     public int Light; // Свет включен/выключен
     ```

3. Источник света - положение и направление света.

4. Цвет - вектор из трех чисел (синий, красный, зеленый).

```
Color.FromArgb(0x40, 0x23, 0x16) // Пример представления
цвета
```

5. Математические абстракции:

   (a) точка - хранит положение, задается координатами x, y, z;

   (b) вектор - хранит направление, задается x, y, z;

   (c) многоугольник - хранит вершины, нормаль, цвет.

2.8 Вывод

В данном разделе были подробно рассмотрены алгоритмы, которые будут реализованы, и приведена схема алгоритма обратной трассировки лучей, указан способ оптимизации данного алгоритма для решения поставленной задачи и описаны используемые структуры.

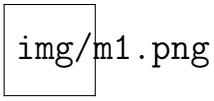

Рисунок 2.4 – Обычная молния

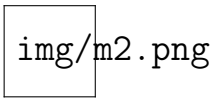

Рисунок 2.5 – Молния-лидер

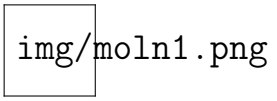

Рисунок 2.6 – Генерация молнии без побочных ветвей

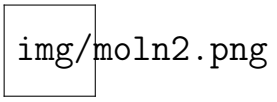

Рисунок 2.7 – Генерация молнии с побочными ветвями

3 Технологический раздел

В данном разделе будут приведены требования к программному обеспечению, средства реализации и листинга кода.

4 Исследовательская часть

Заключение

Список использованных источников

[1] Алгоритм Z-буфера. [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 09.10.2021).

[2] Трассировка лучей из книги Джефа Проузиса [Электронный ресурс]. Режим доступа: https://www.graphicon.ru/oldgr/courses/cg99/notes/lect12/prouzis/raytrace.htm (дата обращения: 28.10.2021).

[3] Алгоритм Робертса. [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 09.10.2021).

[4] Алгоритм, использующие список приоритетов (алгоритм художника). [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 09.10.2021).

[5] Алгоритм Варнока. [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 09.10.2021).

[6] Модели освещения. [Электронный ресурс]. Режим доступа: https://devburn.ru/2015/09/ (дата обращения: 09.10.2021).