|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 Якуба Дмитрий Васильевич

*фамилия, имя, отчество*

Группа ИУ7-43Б

Тип практики стационарная

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | Якуба Д. В. | | *подпись, дата* | *фамилия, и.о.* |   Студент: |  |
| |  |  | | --- | --- | |  | Куров А. В. | | *подпись, дата* | *фамилия, и.о.* |   Руководитель практики: |

Название предприятия МГТУ им. Н.Э. Баумана, каф. ИУ7

Оценка

*Москва, 2020 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)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_ИУ7\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_И.В.\_Рудаков\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине Компьютерная графика

Студент группы ИУ7-53Б

Якуба Дмитрий Васильевич

(Фамилия, имя, отчество)

Тема курсового проекта Трехмерный планировщик выставочных стендов

Направленность КП (учебный, исследовательский, практический, производственный, др.)

учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к \_4\_ нед., 50% к \_7\_ нед., 75% к \_11 нед., 100% к \_14 нед.

***Задание*** Разработать программное обеспечение для визуализации и редактирования площадки и интерьера выставочных стендов. Разработать интерфейс, который позволит выбирать из предложенного набора элементы декора, представленные в виде объемных моделей, и расставлять их по сетке сцены, заданной пользователем. Проектируемый программный продукт должен предоставлять возможность размещения источников света, а также возможность просмотра сцены для разных положений наблюдателя.

***Оформление курсового проекта:***

Расчётно-пояснительная записка на 25-30 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

На защиту проекта должна быть предоставлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс, результаты проведенных исследований.

Дата выдачи задания « » 20 г.

|  |  |  |
| --- | --- | --- |
| **Руководитель курсового проекта** | (Подпись, дата) | Н. В. Новик  (И.О. Фамилия) |
| **Студент** | (Подпись, дата) | Д. В. Якуба  (И.О. Фамилия) |

Оглавление

[Введение 3](#_Toc55127762)

[1. Аналитическая часть 4](#_Toc55127763)

[1.1 Формализация объектов синтезируемой сцены 4](#_Toc55127764)

[1.2 Анализ способов задания трёхмерных моделей 6](#_Toc55127765)

[1.3 Анализ способов задания поверхностных моделей 7](#_Toc55127766)

[1.4 Анализ алгоритмов удаления невидимых рёбер и поверхностей 9](#_Toc55127767)

[Алгоритм Робертса 9](#_Toc55127768)

[Алгоритм, использующий z-буфер 11](#_Toc55127769)

[Алгоритм обратной трассировки лучей 13](#_Toc55127770)

[1.5 Анализ алгоритмов построения теней 15](#_Toc55127771)

[2. Конструкторская часть 17](#_Toc55127772)

[2.1 Общий алгоритм решения поставленной задачи 17](#_Toc55127773)

[2.2 Алгоритм z-буфера 17](#_Toc55127774)

[2.3 Модифицированный алгоритм z-буфера 17](#_Toc55127775)

[2.4 Представление данных в программном обеспечении 18](#_Toc55127776)

[3. Технологическая часть 19](#_Toc55127777)

[3.1 Выбор языка программирования и среды разработки 20](#_Toc55127778)

[3.2 Поля данных реализуемых классов 21](#_Toc55127779)

[3.3 Интерфейс программного обеспечения 22](#_Toc55127780)

[Заключение 25](#_Toc55127781)

[Список использованных источников 26](#_Toc55127782)

[Приложение 27](#_Toc55127783)

# Введение

На настоящий момент большинство компаний, занимающихся тем или иным видом деятельности, связанным с обустройством или декорированием помещений, для согласования технических заданий с клиентом часто обращаются к компьютерному графическому моделированию предмета заказа. Такой подход позволяет как можно чётче обозначить детали проекта, и реже допускать ситуации, в которых становится известно, что исполнитель неправильно воспринял мысль и идею заказчика, когда заказ уже находится на стадии завершения проектирования.

Некоторую реалистичность предоставляемого изображения важно учитывать при реализации средств предварительного показа. Это связанно с тем, что для исполнителя очень важно исключить возникновение правок на поздних стадиях выполнения работы. Для построения реалистичного изображения, которое позволит понять пользователю концепцию декораций, потребуется учитывать невидимость рёбер объектов сцены по отношению к наблюдателю и освещение отдельных участков рабочей плоскости. При этом не стоит вносить в рассмотрение, например, рассеивание, интерференцию, дифракцию, отражения света или передачу цвета объектов. Связано это с тем, что на поздних этапах согласования проекта более качественное изображение, в котором больше уделено внимания деталям, синтезируют в специализированном программном обеспечении.

Цель работы – реализовать программное обеспечение для визуализации площадки и интерьера выставочных стендов.

Для достижения поставленной цели потребуется:

1) формализовать объекты синтезируемой сцены и описать список доступных к размещению на сцене моделей интерьера;

2) выбрать или модифицировать существующие алгоритмы компьютерной графики для визуализации синтезируемой сцены;

3) реализовать выбранные алгоритмы визуализации синтезируемой сцены;

4) разработать программный продукт для визуализации и редактирования площадки выставочного стенда и трехмерных объектов, расположенных на ней.

# Аналитическая часть

## 1.1 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов:

* Площадка выставочного стенда – правильный параллелепипед с заданной сеткой, по которой расставляются модели интерьера. Объекты располагаются только на одной из сторон площадки. При инициализации сцены её левый верхний угол расположен у начала экранных координат, также она параллельна плоскости XY. Размеры границ задаются количеством квадратов (ячеек) по ширине и длине площадки, причём размер ячеек – константная величина, определяемая внутри программы;
* Объекты интерьера – модели, которые занимают ячейки сетки или их часть. Каждая модель представляет собой набор граней, описываемых точками в пространстве, которые соединены рёбрами. Все доступные модели поставляются вместе с программным обеспечением, внесение новых моделей пользователем в базу не предусмотрено. Имеется возможность изменить положение модели на сцене: переместить, повернуть. Также можно выбрать длину, ширину или высоту модели, при этом единица измерения – занимаемые ячейки (или их часть) площадки выставочного стенда.

Опираясь на примеры визуализации выставочных стендов (см. Приложение) в список доступных объектов интерьера входят:

* + Стол. Является плоскостью, параллельной плоскости выставочного стенда, расположенной на прямоугольном параллелепипеде;
  + Высокий стол (барная стойка). Является плоскостью, параллельной плоскости выставочного стенда, расположенной на четырех прямоугольных параллелепипедах;
  + Стул. Является двумя перпендикулярными плоскостями, одна из которых параллельна плоскости выставочного стенда. Эти плоскости расположены на четырех прямоугольных параллелепипедах;
  + Барный стул. Является двумя перпендикулярными плоскостями, одна из которых параллельна плоскости выставочного стенда. Эти плоскости расположены на на прямоугольной призме;
  + Диван. Сложная фигура, представляемая прямоугольным параллелепипедом с вычтенной 1/4 подобной частью;
  + Растение в горшке в качестве элемента декора. Занимает одну клетку плоскости сцены. Является набором плоскостей, образующими в отдельности цветочный горшок, стебель и некоторую растительность;
  + Подиум для представления выставочных предметов заказчика. Является прямоугольным параллелепипедом;
  + Экран для проектора. Является плоскостью, перпендикулярной плоскости выставочного стенда, расположенной на правильной призме с боковыми рёбрами, перпендикулярными той же плоскости. Стойка всегда расположена по середине, относительно размеров экрана;
  + Плазменный телевизор. Является прямоугольным параллелепипедом, изначально расположенным над плоскостью;
  + Шкаф. Является прямоугольным параллелепипедом с двумя примыкающими к нему меньшими по размеру параллелепипедами;
  + Стеллаж с полками. Является несколькими параллельными плоскостями, расположенными на четырёх правильных призмах. Количество параллельных плоскостей зависит от заданной высоты модели;

В тех пунктах, где не указано обратного, длина и высота модели задается пользователем по количеству занимаемых клеток области сцены;

* Источники света – точка пространства, подобная точке положения наблюдателя. Принимает ортогональную проекцию визуализируемой сцены из своего положения с некоторым ограниченным обзором. В зависимости от расположения источника и направления распространения лучей света, определяется тень от объектов, расположенных на сцене. Положение источника света задаётся относительно текущей точки наблюдения в градусах по осям X и Y.

## 1.2 Анализ способов задания трёхмерных моделей

Модели являются отображением формы и размеров объектов. Основное назначение модели – правильно отображать форму и размеры определённого объекта [1].

В основном используются следующие три формы моделей:

1) каркасная (проволочная) модель. В данной модели задаётся информация о вершинах и рёбрах объекта. Это одна из простейших форм задания модели, но она имеет один существенный недостаток: модель не всегда однозначно передаёт представление о форме объекта;

2) поверхностные модели. Этот тип модели часто используется в компьютерной графике. Поверхность может описываться аналитически, либо задаваться другим способом (например, отдельными участками поверхности, задаваемыми в качестве участков поверхности того или иного вида). При этом вложенные криволинейные поверхности можно представлять в упрощённом виде, выполняя, например, полигональную аппроксимацию: такая поверхность будет задаваться в виде поверхности многогранника. Данная форма имеет свой недостаток: отсутствует информация о том, с какой стороны поверхности находится материал;

3) твердотельные (объёмные) модели. Отличие данной формы задания модели от поверхностной формы состоит в том, что в объёмных моделях к информации о поверхностях добавляется информация о том, где расположен материал. Это можно сделать путём указания направления внутренней нормали.

Таким образом, можно сделать вывод, что для решения задачи не подойдёт каркасная форма, так как такое представление будет приводить к неправильному восприятию заказчиком форм моделей, а также не подойдут и объёмные модели, так как на этапе проектирования обстановки выставочного стенда исполнителю совершенно не важно, из какого материала будет выполнен тот или иной объект сцены. Методом исключения приходим к выбору поверхностной формы модели.

## 1.3 Анализ способов задания поверхностных моделей

Поверхностные модели задаются [2]:

* 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, рёбер и граней, определяющих форму объекта в трёхмерном пространстве.

Также существует множество различных способов хранения информации о сетке [2]:

1. список граней. Объект – это множество граней и множество вершин. В каждую грань входят как минимум 3 вершины;
2. «крылатое» представление. Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые её касаются;
3. полурёберные сетки. То же «крылатое» представление, но информация обхода хранится для половины грани;
4. 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;
5. 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. Способ хранения полигональной сетки при этом – 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.

При этом недостаточная эффективность преобразований геометрии объектов, сопутствующая такой форме представления, на программный продукт не повлияет, так как задание геометрии тел предусмотрены только на этапе выбора специфичных моделей, для которых можно определить длину, ширину или высоту. То есть начальная геометрия тел будет задаваться определённым алгоритмом, что разрешит описанную проблему.

## 1.4 Анализ алгоритмов удаления невидимых рёбер и поверхностей

Решать поставленную задачу удаления можно как в объектном пространстве (в мировой системе координат), так и в пространстве изображения (в экранных координатах).

Обозначим свойства, которыми должен обладать алгоритм, для оптимальной работы реализуемого программного обеспечения:

* Алгоритм должен быть достаточно быстрым при работе с множеством объектов сцены, чтобы пользователь не ожидал долгой загрузки изображения;
* Алгоритм может работать в любом пространстве (скорость важнее точности)

Рассмотрим алгоритмы для решения поставленной задачи.

### Алгоритм Робертса

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами [3].

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 . Для проведения проверки следует взять точку, расположенную внутри тела. Координаты такой точки можно получить путём усреднения координат всех вершин тела.

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. Если тело является преградой, то луч должен пройти через тело. Если луч проходит через тело, то он находится по положительную сторону от каждой грани тела.

Для того, чтобы определить, подходит ли данный алгоритм для решения поставленной задачи, рассмотрим его преимущества и недостатки.

Преимущество:

* Алгоритм работает в объектном пространстве, точность вычислений высокая.

Недостатки:

* Теоретический рост сложности алгоритма – квадрат числа объектов.

Для решения данной проблемы достаточно воспользоваться модифицированными реализациями, например, с использованием габаритных тестов или сортировки по оси z;

* Все тела сцены должны быть выпуклыми. Данная проблема также приводит к усложнению алгоритма, так как потребуется прибегнуть к проверке объектов на выпуклость и их разбиению на выпуклые многоугольники.

Вывод:

Алгоритм Робертса не подходит для решения поставленной задачи по следующим причинам:

* От программного обеспечения не требуется той точности визуализации объектов, которую предоставляет алгоритм;
* На сцене может находиться множество объектов, что замедлит скорость работы алгоритма. Это не удовлетворяет преподнесённым требованиям к скорости выполнения алгоритма. Особенные проблемы также могут наблюдаться при наличии множества невыпуклых тел на сцене;
* Реализация модификаций, позволяющих приблизить рост сложности алгоритма к линейной, очень трудозатратна.

### Алгоритм, использующий z-буфер

Данный алгоритм работает в пространстве изображения. Используется два буфера: буфер кадра, в котором хранятся атрибуты каждого пикселя в пространстве изображения, и z-буфер, куда помещается информация о координате z для каждого пикселя [4].

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.

Рассмотрим преимущества и недостатки описанного алгоритма.

Преимущества:

* Простота реализации;
* Оценка вычислительной трудоемкости алгоритма линейна;
* Экономия вычислительного времени, так как элементы сцены не сортируются.

Недостатки:

* Большой объём требуемой памяти.
* Реализация эффектов прозрачности сложна.

Вывод:

Алгоритм отвечает главному требованию - скорости работы с множеством объектов. Также простота алгоритма позволит достаточно быстро реализовать данный алгоритм и, что важнее, в полной мере отладить его.

При этом следует отметить, что само изображение будет относительно малых размеров, что приведёт к некритично большим затратам памяти для выполнения данного алгоритма.

### Алгоритм обратной трассировки лучей

Наблюдатель видит объект посредством испускаемого источником света, который падает на этот объект и согласно законам оптики некоторым путём доходит до глаза наблюдателя. 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 [5].

Предполагается, что сцена уже преобразована в пространство изображения, а точка, в которой находится наблюдатель, находится в бесконечности на положительной полуоси , 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 определённых объектов сцены с этими лучами. При этом необходимо проверить пересечение каждого объекта сцены с каждым лучом, а пересечение с представляет видимую поверхность для данного пикселя.

Если же точка наблюдателя находится не в бесконечности, то есть в рассмотрении фигурирует перспективная проекция, то предполагается, что сам наблюдатель по-прежнему находится на положительной полуоси , а сам растр при этом перпендикуляром оси . Задача будет состоять в том, чтобы построить одноточечную центральную проекцию на картинную плоскость.

Определения пересечений происходит с помощью погружения объектов в некоторую выпуклую оболочку – например, сферическую. Поиск пересечения с такой оболочкой происходит проще: достаточно проверить превосходит ли радиус сферы-оболочки расстояние от центра этой сферы до луча.

Предположим, что некоторая прямая проходит через две точки :

Компоненты при этом:

Таким образом минимальное расстояние от этой прямой до некоторой точки :

Где параметр , который определяет ближайшую точку :

Если , где – радиус сферической оболочки, то луч не может пересечься с объектом.

Рассмотрим преимущества и недостатки описанного алгоритма.

Преимущества:

* Высокая реалистичность синтезируемого изображения;
* Работа с поверхностями в математической форме;
* Вычислительная сложность слабо зависит от сложности сцены.

Недостаток:

* Производительность.

Вывод:

Алгоритм не отвечает главному требованию – скорости работы. Также от реализуемого продукта не требуется высокой реалистичности синтезируемого изображения и возможности работы с поверхностями, заданными в математической форме. Указанные факты говорят о том, что обратная трассировка лучей не подходит для решения поставленной задачи.

Таким образом, в качестве алгоритма удаления невидимых рёбер и поверхностей был выбран алгоритм с использованием z-буфера.

## 1.5 Анализ алгоритмов построения теней

При использовании алгоритма обратной трассировки лучей, рассмотренного ранее, построение теней происходит по ходу выполнения алгоритма: пиксел будет затенён, если испускаемый луч попадает на объект, но не попадает в источник света [5]. Данный алгоритм не подходит для решения поставленной задачи, так как при проведении анализа алгоритмов удаления невидимых рёбер он не был выбран в качестве реализуемого.

В качестве алгоритма удаления невидимых линий и поверхностей в предыдущем подпункте был выбран алгоритм с использованием z-буфера, поэтому наилучшим вариантом будет модификация указанного метода путём добавления вычисления теневого z-буфера из точки наблюдения, совпадающей с источником света [6].

Такой подход позволит не усложнять структуру программы, а также избежать проблем адаптации двух различных методов друг к другу, а, следовательно, уменьшить время отладки алгоритма.

## Вывод

Были рассмотрены способы задания трёхмерных моделей и выбрана поверхностная форма задания моделей. Также были рассмотрены алгоритмы удаления невидимых рёбер: алгоритм, использующий z-буфер, алгоритм Робертса и алгоритм обратной трассировки лучей. В качестве реализуемого был выбран алгоритм z-буфера, отвечающего двум требованиям: быстрая работа с множеством объектов сцены и преобладание скорости над точностью. Были рассмотрены алгоритмы построения теней. В качестве реализуемого был выбран алгоритм, использующий теневые карты, так как уже было указано, что в работе будет использоваться z-буфер.

# 2. Конструкторская часть

В разделе будет определён список требований к программному обеспечению, общий алгоритм решения поставленной задачи, алгоритм z-буфера и его модификация с применением теневого буфера.

## 2.1 Требования к программному обеспечению

Программа должна предоставлять доступ к следующему функционалу:

1. создание сцены с заданным размером;
2. растеризация сцены;
3. поворот, перемещение и масштабирование визуализируемой сцены;
4. добавление и удаление объектов сцены;
5. изменение положения объектов на сцене;
6. добавление и удаление источников света на сцене.

## 2.1 Общий алгоритм решения поставленной задачи

Рассмотрим алгоритм визуализации выставочного стенда.

1. Задать размеры области размещения объектов

2. Разместить объекты сцены

3. С помощью модифицированного алгоритма, использующего z-буфер, определить падающие от объектов сцены тени и визуализировать обстановку, основываясь на текущем положении наблюдателя

## 2.2 Алгоритм z-буфера

1. Всем элементам буфера кадра присвоить фоновое значение

2. Инициализировать z-буфер минимальным значением глубины

3. Для каждого многоугольника сцены в произвольном порядке:

3.1 Для каждого пикселя, который принадлежит многоугольнику вычислить его глубину

3.2 Сравнить вычисленную глубину пикселя со значением, которое

находится в z-буфере:

Если , то и

4. Вывести итоговое изображение

## 2.3 Модифицированный алгоритм z-буфера

1. Для каждого направленного источника света:

1.1 Инициализировать теневой z-буфер минимальным значением глубины

1.2 Определить теневой z-буфер для источника

2. Выполнить алгоритм z-буфера для точки наблюдения. При этом, если некоторая поверхность оказалась видимой относительно текущей точки наблюдения, то проверить, видима ли данная точка со стороны источников света.

Для каждого источника света:

2.1 Координаты рассматриваемой точки линейно преобразовать из вида наблюдателя в координаты на виде из рассматриваемого источника света

2.2 Сравнить значение со значением :

Если , то пиксел высвечивается с учётом его затемнения, иначе точка высвечивается без затемнения

## 2.4 Используемые типы и структуры данных

Для реализации программного обеспечения потребуется реализовать типы и структуры данных, представленные в Таблице 1.

Таблица 1 Представление данных в программном обеспечении

|  |  |
| --- | --- |
| Данные | Представление |
| Область размещения объектов | Количество заполняемых ячеек области по ширине и длине и таблица свободных ячеек |
| Точка трехмерного пространства | Координаты по осям x, y, z |
| Полигон | Три точки трехмерного пространства |
| Объект сцены | Список полигонов |
| Источник света | Угол по осям X и Y относительно текущей точки наблюдения |
| Пользовательский интерфейс | Библиотечные классы |

## Вывод

На основе теоретических данных, полученных из аналитического раздела, были описаны общий алгоритм решения задачи, алгоритм z-буфера и модифицированный алгоритм z-буфера, а также были определены используемые типы и структуры данных.

# 3. Технологическая часть

В данном разделе будет обусловлен выбор языка программирования и среды разработки, а также рассмотрены структура и состав классов, сведения о модулях программы и пользовательский интерфейс.

## 3.1 Выбор языка программирования и среды разработки

При написании программного продукта использовался язык C++. Данный выбор обусловлен следующими факторами:

1) данный язык преподавался в рамках курса Объектно-Ориентированного Программирования;

2) высокая вычислительная производительность. В аналитической части была обозначена важность скорости исполнения кода;

3) язык поддерживает объектно-ориентированную парадигму программирования. Данный фактор позволить привести объекты сцены к объектам классов, а также пользоваться шаблонами проектирования. В свою очередь описанные факты дадут возможность писать читаемый и эффективный код;

4) доступность. Большое количество учебной литературы позволит быстро и эффективно писать код.

При написании программы будет задействована среда разработки QT Creator. Данный выбор обусловлен следующими факторами:

1) основы работы с данной средой разработки преподаётся в рамках курса Программирования на Си;

2) QT Creator позволяет работать с расширением Qt Design, которое позволит создать удобный и надёжный интерфейс для программного продукта в сжатые сроки.

## 3.2 Структура и состав классов

В данном разделе на рис. 3.1, рис 3.2, рис. 3.3 предоставлена структура и состав классов в программном обеспечении.

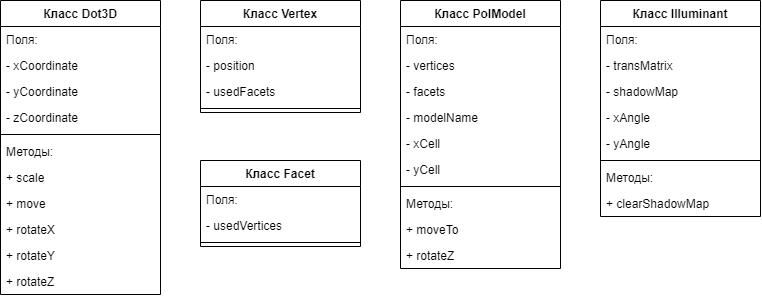


Рис. 3.1 структура классов Dot3D, Vertex, Facet, PolModel, Illuminant

Dot3D – класс точки трехмерного пространства. Хранит координаты в пространстве, владеет методами преобразований точки.

Vertex – класс вершины. Хранит координаты точки вершины и номера граней, в которых она задействована.

Facet – класс грани. Хранит номера задействованных в грани вершин.

PolModel – класс полигональной модели. Хранит множество вершин и граней, образующих модель, её имя и координаты ячейки, по которой она расположена. Владеет методами перемещения и вращения по оси Z модели.

Illuminant – класс источника света. Хранит теневую карту источника, матрицу преобразований и соответствующие ей углы для перемещения текущей точки обзора в точку размещения источника. Владеет методом очистки собственной теневой карты.

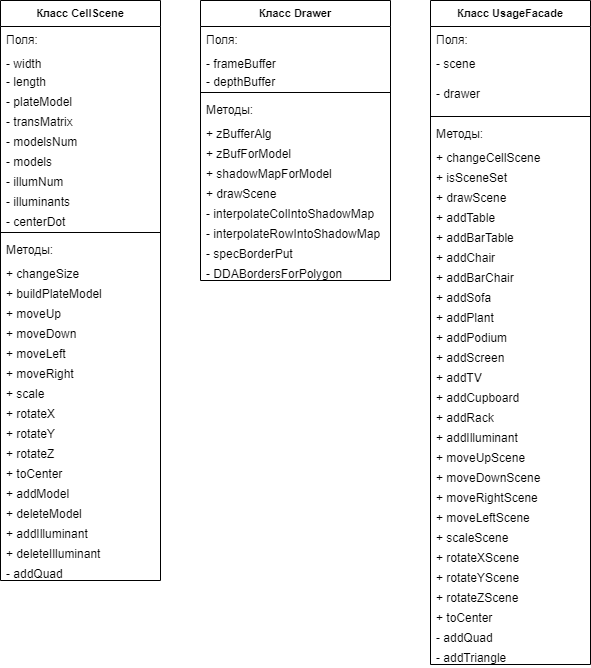


Рис. 3.2 структура классов CellScene, Drawer и UsageFacade

CellScene – класс рабочей области сцены. Хранит длину и ширину сцены, модель, на которой размещаются модели интерьера, матрицу преобразований для перехода в текущую точку наблюдения, множество моделей интерьера и источников света, размещённых на сцене, а также точку центра выполнения преобразований. Владеет методами выполнения преобразований над сценой, построения области размещения моделей, добавления и удаления объектов интерьера и источников света.

Drawer – класс, отвечающий за растеризацию сцены. Хранит буфер кадра и буфер глубины. Владеет методами алгоритма теневого z-буфера и формирования объекта для отображения рисунка в главном приложении.

UsageFacade – класс, выполненный по структурному шаблону проектирования фасад. Отвечает за взаимодействие пользователя с программным обеспечением. Хранит формируемую сцену и класс, отвечающий за растеризацию сцены. Владеет методами добавления моделей на сцену и преобразований сцены.

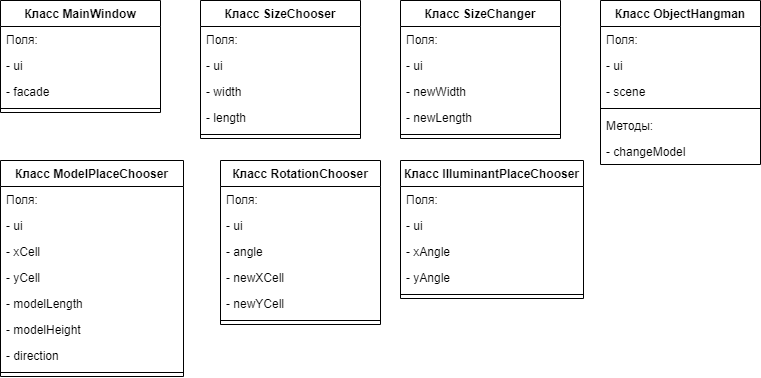


Рис. 3.3 структура классов MainWindow, illuminantPlaceChooser, modelPlaceChooser, sizeChanger, sizeChooser, ChangeChooser

MainWindow – точка входа в программу.

SizeChooser – класс, связанный с интерфейсов выбора размера новой сцены. Хранит в себе значений длины и ширины создаваемой сцены.

SizeChanger – класс, связанный с интерфейсом выбора новых размеров сцены. Хранит в себе значения новой длины и ширины сцены.

ModelPlaceChooser – класс, связанный с интерфейсом выбора точки расположения добавляемой модели, а также её длины, высоты и ориентации в пространстве.

ChangeChooser – класс, связанный с интерфейсом изменения присутствующих на сцене моделей интерьера. Хранит новые координаты размещения модели и угол поворота относительно текущего положения.

IlluminantPlaceChooser – класс, связанный с интерфейсом выбора расположения добавляемого источника света. Хранит углы поворота относительно осей X и Y для перехода из точки наблюдения в точку расположения источника света.

ObjectHangman – класс, связанный с интерфейсом работы с объектами сцены. Хранит сцену, с которой в данный момент работает. Владеет методом изменения модели.

## 3.3 Сведения о модулях программы

main.cpp – главная точка входа в приложение;

mainwindow.cpp, mainwindow.h – описание и реализация главного окна приложения;

mainwindow.ui – форма пользовательского интерфейса главного окна приложения;

placechooser.cpp, placechooser.hpp – описание и реализация класса PlaceChooser;

placechooser.ui – форма пользовательского интерфейса класса PlaceChooser;

rotationchooser.cpp, rotationchooser.hpp – описание и реализация класса RotationChooser;

rotationchooser.ui – форма пользовательского интерфейса класса RotationChooser;

sizechanger.cpp, sizechanger.hpp – описание и реализация класса SizeChanger;

sizechanger.ui – форма пользовательского интерфейса класса SizeChanger;

sizechooser.cpp, sizechooser.hpp – описание и реализация класса SizeChooser;

sizechooser.ui – форма пользовательского интерфейса класса SizeChooser;

illuminantplacechooser.cpp, illuminantplacechooser.hpp – описание и реализация класса IlluminantPlaceChooser;

illuminantplacechooser.ui – форма пользовательского интерфейса класса Illuminantplacechoooser;

objecthangman.cpp, objecthangman.hpp – описание и реализация класса ObjectHangman;

objecthangman.ui – форма пользовательского интерфейса класса ObjectHangman;

usagefacade.cpp, usagefacade.hpp – описание и реализация классов UsageFacade и Drawer;

additivemathelements.cpp, additivemathelements.hpp – описание и реализация классов математических объектов.

objects.cpp, objects.hpp – описание и реализация класса полигональных моделей, источников света и сцены.

## 3.4 Интерфейс программного обеспечения

Пользовательский интерфейс программного обеспечения предоставлен на Рисунке 1.



Рисунок 1 Пользовательский интерфейс ПО

#### Список доступных объектов (Рисунок 2):

Список позволяет пользователю выбрать требующийся объект и установить его на плоскости сцены.



Рисунок 2 Список доступных объектов

#### Работа со сценой (Рисунок 3):

Программное обеспечение позволяет сохранить созданную пользователем сцену, либо загрузить её. Также для каждой сцены есть возможность изменить её параметры: количество доступных ячеек для установки объектов.



Рисунок 3 Работа со сценой

#### Работа с объектом (Рисунок 4):

Выбрав на сцене установленный объект (путём указания на него мышью в области визуализации (4)), пользователь может удалить указанный объект, либо изменить его ориентацию в пространстве.



Рисунок 4 Работа с объектом

#### Область визуализации (находится правее описанных областей на Рисунке 1):

Через данную область происходит основное взаимодействие пользователя с обстановкой сцены – выделение объектов для их дальнейшего перемещения, поворота или удаления. Также данная область отвечает за выбор точки наблюдения путём «вращения» сцены с помощью мыши.

## Вывод

Были представлены объекты, описанные классами, которые будут участвовать в реализации спроектированных алгоритмов. Также был описан интерфейс программного обеспечения.

# Заключение

Во время выполнения практики было спроектировано программное обеспечение для визуализации и редактирования площадки и интерьера выставочных стендов. Был разработан интерфейс, позволяющий выбирать из предложенного набора элементы декора, представленные в виде объёмных моделей, и расставлять их по сетке сцены, заданной пользователем. Проект предусматривает возможность размещения источников света, а также возможность просмотра сцены для разных положений наблюдателя.

При проведении работы были получены знания в области компьютерной графики и закреплены навыки проектирования программного обеспечения, а поиск оптимальных решений для эффективной работы программного обеспечения позволил улучшить навыки поиска и анализа информации.

Следует отметить, что проделанная работа позволила лучше изучить как язык программирования С++, так и среду разработки Qt Creator и её расширения.

# Список использованных источников

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

2. Набережнов Г. М. Трёхмерное моделирование полигональными сетками [Текст] / Г. М. Набережнов, Н. Н. Максимов // Казань: Казанский Государственный Технический университет им. А.Н.Туполева, 2008. – 14 с.

3. Алгоритм Робертса [Электронный ресурс]. – Режим доступа: [http://compgraph.tpu.ru/roberts.htm](http://compgraph.tpu.ru/roberts.htm%20) (дата обращения 07.07.20)

4. Алгоритм, использующий z-буфер [Электронный ресурс]. – Режим доступа: <http://compgraph.tpu.ru/zbuffer.htm> (дата обращения 07.07.20)

5. Henrik Wann Jensen, Realistic Image Synthesis Using Photon Mapping. A. K. Peters, Ltd.63 South Avenue Natick, MA, United States, 2001. - 181 с.

6. Романюк А. Н. Алгоритмы построения теней [Текст] / А. Н. Романюк, М. В. Куринный // КОМПЬЮТЕРЫ+ПРОГРАММЫ. – 2000. - № 8-9. – С. 13.

# Приложение



Рисунок 5 Пример стенда 1



Рисунок 6 План-схема стенда 1



Рисунок 7 Пример стенда 2



Рисунок 8 План-схема стенда 2

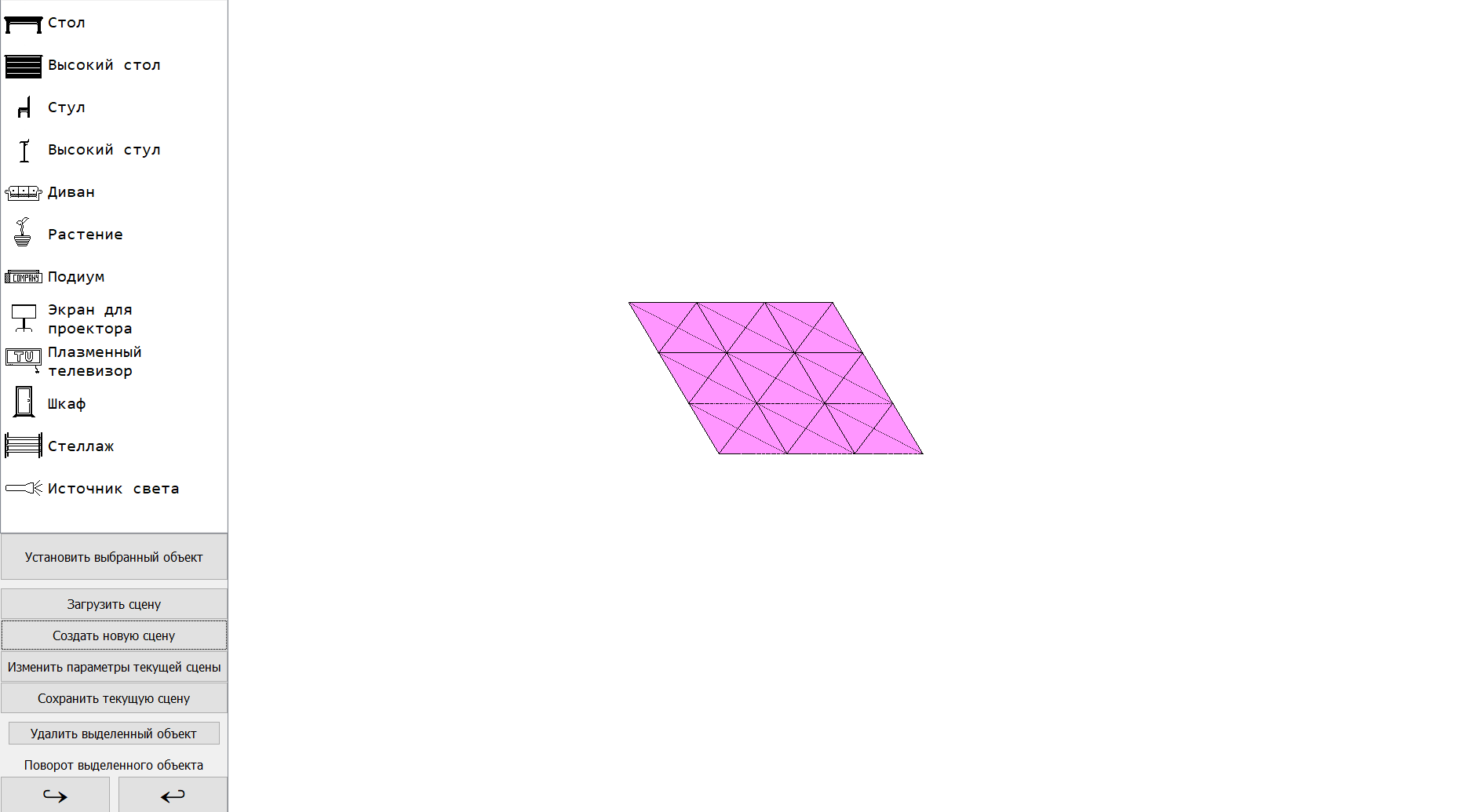


Рисунок 9 Предварительная визуализация плоскости для размещения предметов интерьера