Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:
«Трехмерный планировщик выставочных стендов»

Студент _____ _____ Д.В. Якуба
группа *подпись, дата* *и.о., фамилия*

Руководитель практики: _____ Н.В. Новик
подпись, дата *и.о., фамилия*

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.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Компьютерная Графика

Студент группы ИУ7-53Б

Якуба Дмитрий Васильевич
(Фамилия, имя, отчество)

Тема курсового проекта Трёхмерный 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. Реализовать интерфейс, который позволит выбирать из предложенного набора элементы декора, представленные в виде объемных моделей, и расставлять их по сетке сцены, заданной пользователем. Программный продукт должен предоставлять возможность размещения источников света, а также возможность просмотра сцены для разных положений наблюдателя.

Оформление курсового проекта:
Расчетно-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 _____ Н. В. Новик
(Подпись, дата) (И.О. Фамилия)

Студент _____ Д. В. Якуба
(Подпись, дата) (И.О. Фамилия)

Оглавление

Введение

Большое количество 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 дает возможность как можно четче 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 важно исключить возникновение правок на поздних стадиях выполнения работы. Для построения реалистичного изображения, которое позволит понять пользователю концепцию декораций, потребуется учитывать невидимость ребер объектов сцены по отношению к наблюдателю и освещение отдельных участков рабочей плоскости. Рассеивание, интерференция, дифракция, отражения света и передача цвета объектов не рассматриваются, так как на поздних этапах согласования проекта 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 сцены;

3) реализовать выбранные алгоритмы визуализации;

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

1. 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 Размеры границ задаются количеством квадратов (ячеек) по ширине и длине площадки, причем 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 ребрами. Все доступные модели поставляются вместе с программным обеспечением, внесение новых моделей пользователем в базу не предусмотрено. Доступен функционал изменения положения модели на сцене: переместить, повернуть. Также доступен выбор длины, ширины или высоты модели. Единица измерения пропорций модели - занимаемые ею ячейки (или их часть) площадки выставочного стенда.

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

  - Стол. Представляется плоскостью, параллельной плоскости выставочного стенда, расположенной на прямоугольном параллелепипеде;
  - Высокий стол (барная стойка). Представляется плоскостью, параллельной плоскости выставочного стенда, расположенной на четырех прямоугольных параллелепипедах;

- Стул. Представляется двумя перпендикулярными плоскостями, одна из которых параллельна плоскости выставочного стенда. Эти плоскости расположены на четырех прямоугольных параллелепипедах;
- Барный стул. Представляется двумя перпендикулярными плоскостями, одна из которых параллельна плоскости выставочного стенда. Эти плоскости расположены на прямоугольной призме;
- Диван. Представляется сложной фигурой, являющейся прямоугольным параллелепипедом с вычтенной 1/4 подобной частью;
- Растение в горшке в качестве элемента декора. Занимает одну клетку плоскости сцены. Представляется набором плоскостей, образующими в отдельности цветочный горшок, стебель и некоторую растительность;
- Подиум для представления выставочных предметов заказчика. Представляется прямоугольным параллелепипедом;
- Экран для проектора. Представляется плоскостью, перпендикулярной плоскости выставочного стенда, расположенной на прямоугольном параллелепипеде. Стойка всегда примыкает к середине экрана;
- Плазменный телевизор. Представляется прямоугольным параллелепипедом, изначально расположенным над плоскостью сцены;
- Шкаф. Представляется прямоугольным параллелепипедом с двумя примыкающими к нему меньшими по размеру параллелепипедами;
- Стеллаж с полками. Представляется несколькими параллельными плоскостями, расположенными на четырех 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 задается относительно текущей точки наблюдения последовательными поворотами по осям X и Y.

1.2 Анализ способов задания трехмерных моделей

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

В основном используются следующие три формы моделей:

1) каркасная (проволочная) модель. В данной модели задается информация о вершинах и ребрах объекта. Это одна из простейших форм задания модели, но она имеет один существенный недостаток: модель не всегда однозначно передает 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 упрощенном

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

Таким образом, можно сделать вывод, что для решения задачи не подойдет каркасная форма, так как такое представление будет приводить к неправильному восприятию заказчиком форм моделей, а также не подойдут и объемные 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, ребер и граней, определяющих форму объекта в трехмерном 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 ее касаются;
3) полуреберные сетки. То же «крылатое» представление, но информация обхода хранится для половины грани;
4) таблица углов. Таблица, хранящая вершины. Обход заданной таблицы неявно задает 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 дает возможность проводить над сеткой множество операций.

Стоит отметить, что решающий фактор в выборе способа задания модели в курсовом проекте – это скорость выполнения преобразований над объектами сцены.

Оптимальное для реализации представление - модель, заданная полигональной сеткой. Такая модель позволит избежать проблем при описании сложных объектов сцены. Способом хранения полигональной сетки был выбран список граней, так как это даст явное описание граней. Этот способ позволит эффективно преобразовывать модели, так как структура будет включать в себя список вершин.

Недостаточная эффективность преобразований геометрии объектов, сопутствующая такой форме представления, отрицательно на программном продукте на скажется, так как начальная геометрия тел будет задаваться заранее описанным алгоритмом.

1.4 Анализ алгоритмов удаления невидимых ребер 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 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:

$$V = \begin{pmatrix} a_1 & a_2 & & a_n \\ b_1 & b_2 & \cdots & b_n \\ c_1 & c_2 & & c_n \\ d_1 & d_2 & & d_n \end{pmatrix}$$

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

2. Этап удаления ребер, 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.

3. Этап удаления невидимых ребер, 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 своем 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, что 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:

- При решении поставленной задачи не требуется той точности визуализации объектов, которую предоставляет алгоритм;
- На сцене может находиться множество объектов, что замедлит скорость его работы. Это не удовлетворяет поставленным требованиям к скорости выполнения алгоритма;
- 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 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.

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 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 объем требуемой памяти;
- Реализация эффектов прозрачности сложна.

Вывод:

Алгоритм отвечает главному требованию - скорости работы с множеством объектов. Простота позволит быстро реализовать и отладить данный алгоритм.

Само изображение будет относительно малых размеров, что приведет к некритично большим затратам памяти.

Алгоритм обратной трассировки лучей

Наблюдатель видит объект посредством испускаемого источником света, который падает на этот объект и согласно законам оптики некоторым путем 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 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.

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 z, а сам растр при этом перпендикуляром оси z. 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 $P_1(x_1, y_1, z_1)$ и $P_2(x_2, y_2, z_2)$:

$$P(t) = P_1 + (P_2 - P_1)t$$

Компоненты при этом:

$$x = x_1 + (x_2 - x_1)t = x_1 + at$$

$$y = y_1 + bt$$

$$z = z_1 + ct$$

Таким образом минимальное расстояние от этой прямой до некоторой точки $P_0(x_0, y_0, z_0)$:

$$d^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2$$

Где параметр t, который определяет ближайшую точку $P(t)$:

$$t = -\frac{a(x_1 - x_0) + b(y_1 - y_0) + c(z_1 - z_0)}{a^2 + b^2 + c^2}$$

Если $d^2 > R^2$, где R – 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 ребер 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 затенен, 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 ребер он не был выбран в качестве реализуемого.

В качестве реализуемого алгоритма была выбрана модификация алгоритма с использованием z-буфера путем 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 программного продукта.

Вывод

Были рассмотрены способы задания трехмерных моделей и выбрана поверхностная форма задания моделей. Также были рассмотрены алгоритмы удаления невидимых ребер: 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 теневой z-буфер.

2. Конструкторская часть

В разделе будет определен список требований к программному обеспечению, приведён 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 $z(x, y)$

3.2 Сравнить вычисленную глубину пикселя со значением, которое находится в z-буфере:

Если $z(x, y) > z_{\text{буф}}(x, y)$, то $z_{\text{буф}}(x, y) = z(x, y)$ и $\text{цвет}(x, y) = \text{опрЦвет}$

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 (x, y, z) линейно преобразовать из вида наблюдателя в координаты (x', y', z') на виде из рассматриваемого источника света

2.2 Сравнить значение $z_{\text{тенБуф}}(x', y')$ со значением $z'(x', y')$:

Если $z'(x', y') < z_{\text{тенБуф}}(x', y')$, то пиксел высвечивается с учетом 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 2.1 Представление данных в программном обеспечении

Данные	Представление
Область размещения объектов	Количество заполняемых ячеек области по ширине и длине
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. Были определены используемые типы и структуры данных.

3. Технологическая часть

В данном разделе будет обусловлен выбор языка программирования и среды разработки, а также рассмотрена 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++ [7]. Данный выбор обусловлен следующими факторами:

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

2) высокая вычислительная производительность;

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) большое количество учебной и справочной литературы.

При написании программы будет задействована среда разработки QT Creator [8]. Данный выбор обусловлен следующими факторами:

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

2) QT Creator позволяет работать с расширением Qt Design, которое позволит создать интерфейс для программного продукта.

3.2 Структура и состав классов

На рисунках 3.1, 3.2, 3.3 предоставлена структура и состав классов в программном обеспечении.

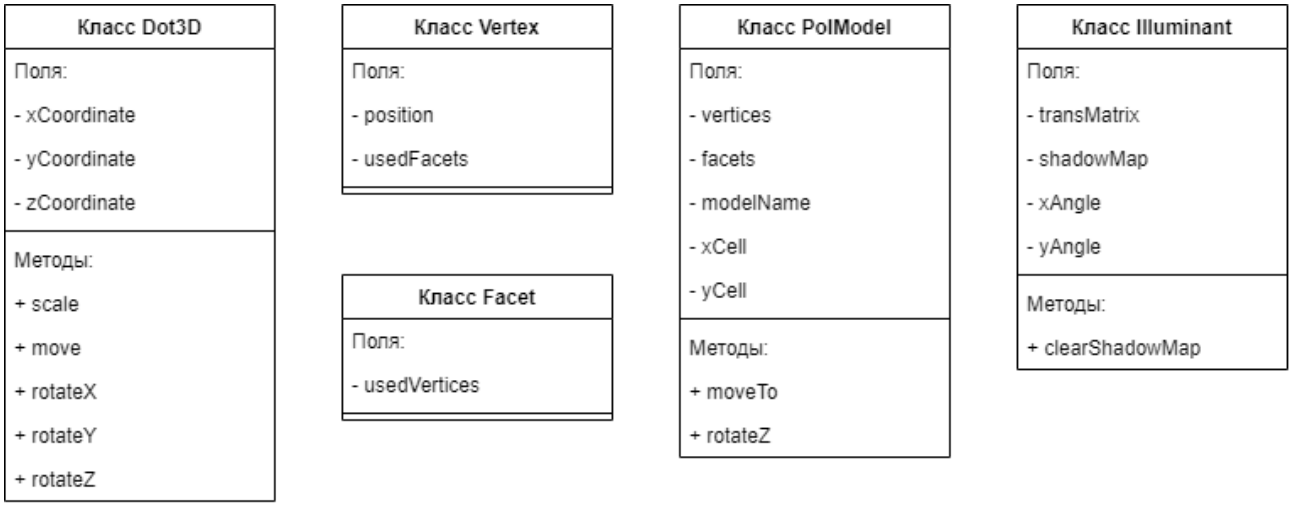

Рисунок 3.1 структура классов Dot3D, Vertex, Facet, PolModel, Illuminant

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, ее 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.

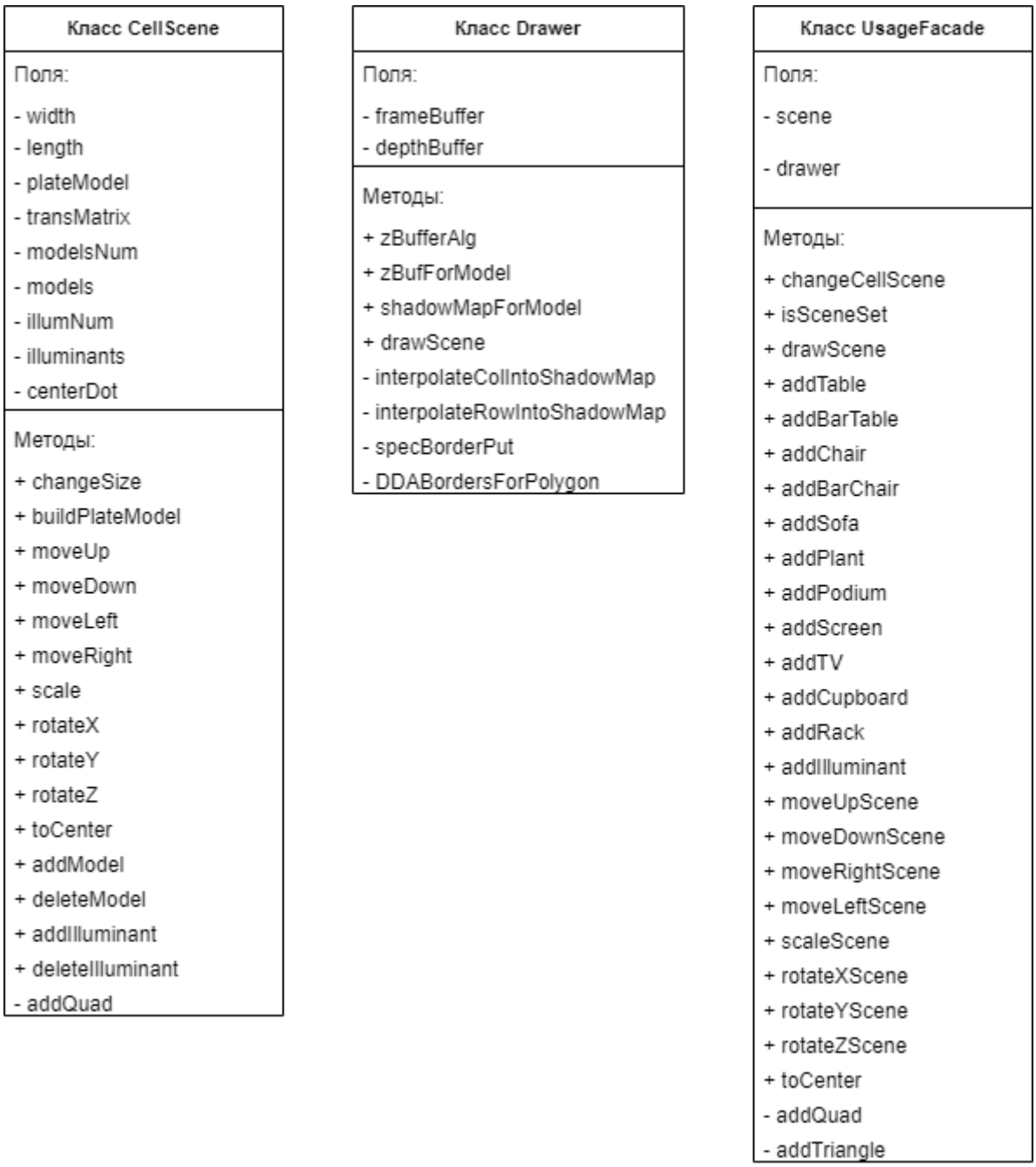

Рисунок 3.2 структура классов CellScene, Drawer и UsageFacade

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, размещенных 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.

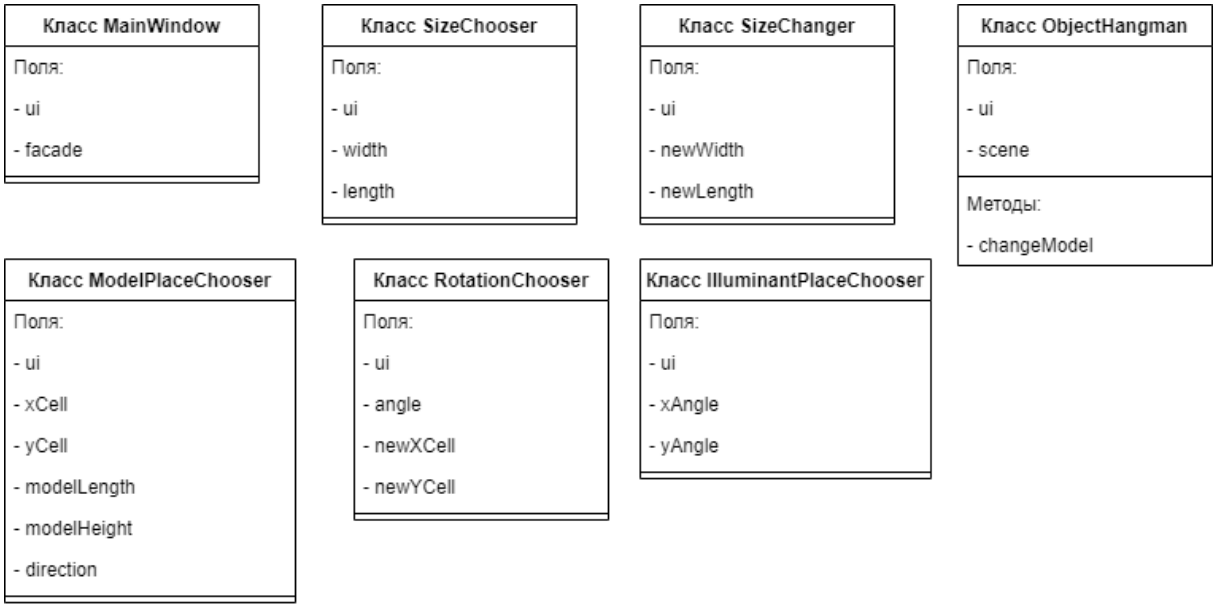

Рисунок 3.3 структура классов MainWindow, illuminantPlaceChooser, modelPlaceChooser, sizeChanger, sizeChooser, ChangeChooser

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 ее 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

Интерфейс главного окна приложения, изображенного на рисунке 3.4, включает в себя:

- Группу работы с объектами сцены. Позволяет добавлять, удалять объекты, а также изменять положение добавленных объектов;
- Группу работы со сцены. Позволяет создавать сцену, изменять параметры текущей сцены и переместить сцену в центр преобразований.

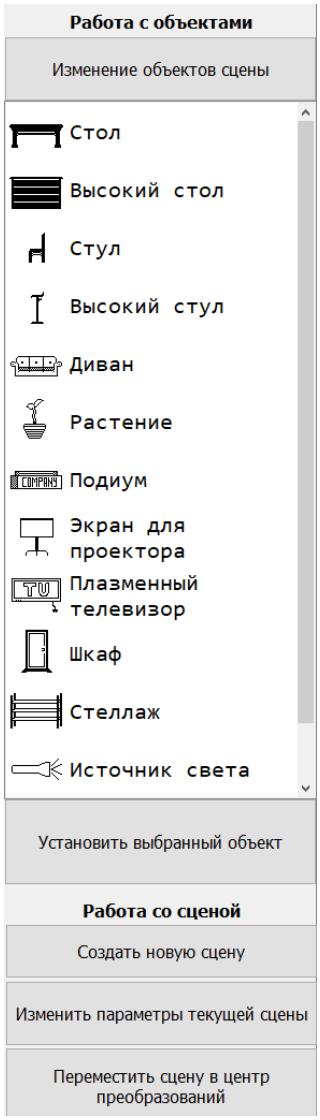

Рисунок 3.4 Интерфейс главного окна приложения

Интерфейс окна выбора размера новой сцены, изображенного на рисунке 3.5, включает в себя поля ввода для ширины и длины создаваемой сцены.

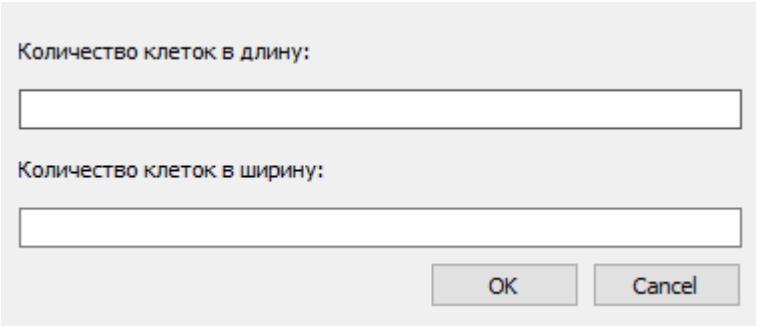

Рисунок 3.5 Интерфейс окна выбора размера новой сцены

Интерфейс окна выбора параметров добавляемой модели, изображенного на рисунке 3.6, включает в себя поля ввода положения по длине и ширине плоскости, длины и высоты, а также выбор направления распространения длинных моделей.

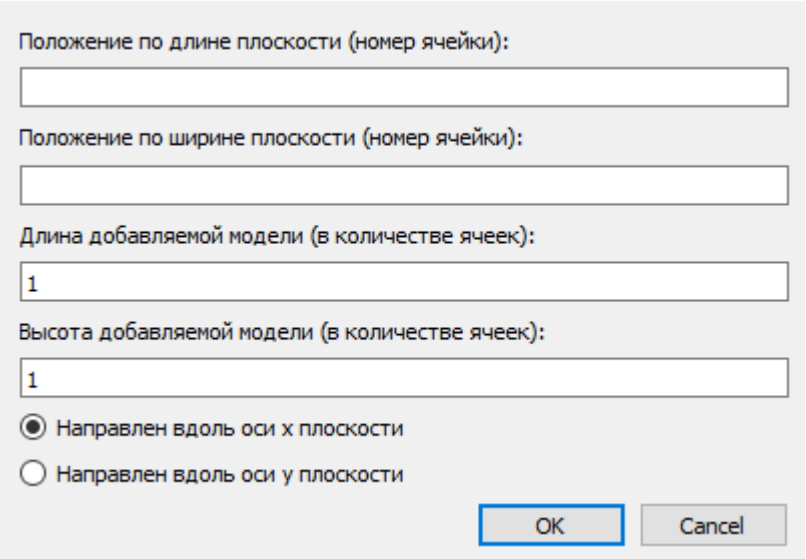

Рисунок 3.6 Интерфейс окна выбора параметров добавляемой модели

Интерфейс окна выбора параметров добавляемого источника света, изображенного на рисунке 3.7, включает в себя поля ввода углов поворота по осям X и Y относительно точки наблюдения.

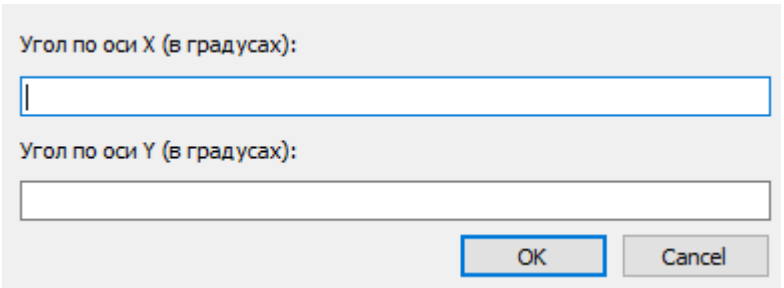

Рисунок 3.7 Интерфейс окна выбора параметров добавляемого источника света

Интерфейс окна изменения объектов сцены, изображенного на рисунке 3.7, включает в себя список объектов сцены.

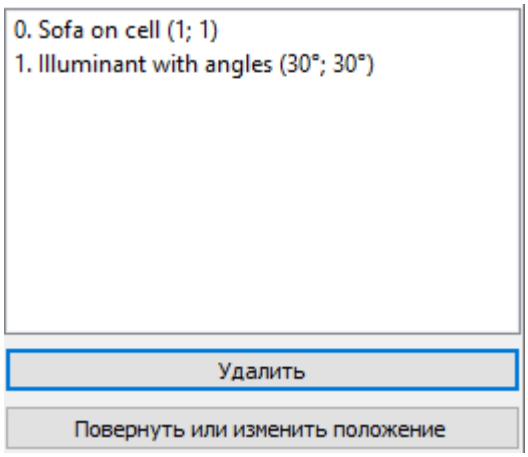

Рисунок 3.8 Интерфейс окна изменения объектов сцены

Интерфейс окна задания нового положения на сцене модели, изображенного на рисунке 3.9, включает в себя поля ввода угла поворота относительно оси Z, новую ячейку расположения по оси X и Y плоскости.

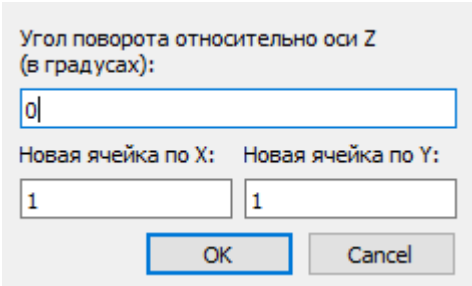

Рисунок 3.9 Интерфейс окна задания нового положения на сцене модели

Интерфейс окна изменения размеров сцены, изображенного на рисунке 3.10, включает в себя два поля ввода новой длины и ширины сцены.

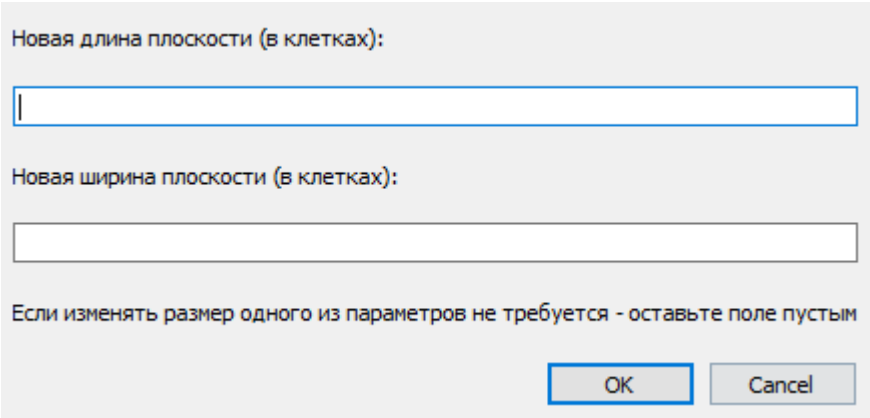

Рисунок 3.10 Интерфейс окна изменения размеров сцены

Вывод

В разделе были предоставлены структуры и состав классов, сведения о модулях и интерфейс программного обеспечения.

4. Экспериментально-исследовательская часть

В данном разделе будут приведены примеры работы программного обеспечения и поставлен эксперимент по оценке скорости работы алгоритма z-буфера без использования и с использованием библиотеки OpenMP при различных размерах визуализируемой площадки.

4.1 Пример работы программного обеспечения

На рисунке 4.1.1 приведен пример работы программы при наличии единственного источника света с заданными углами поворота в 30° по осям X и Y.

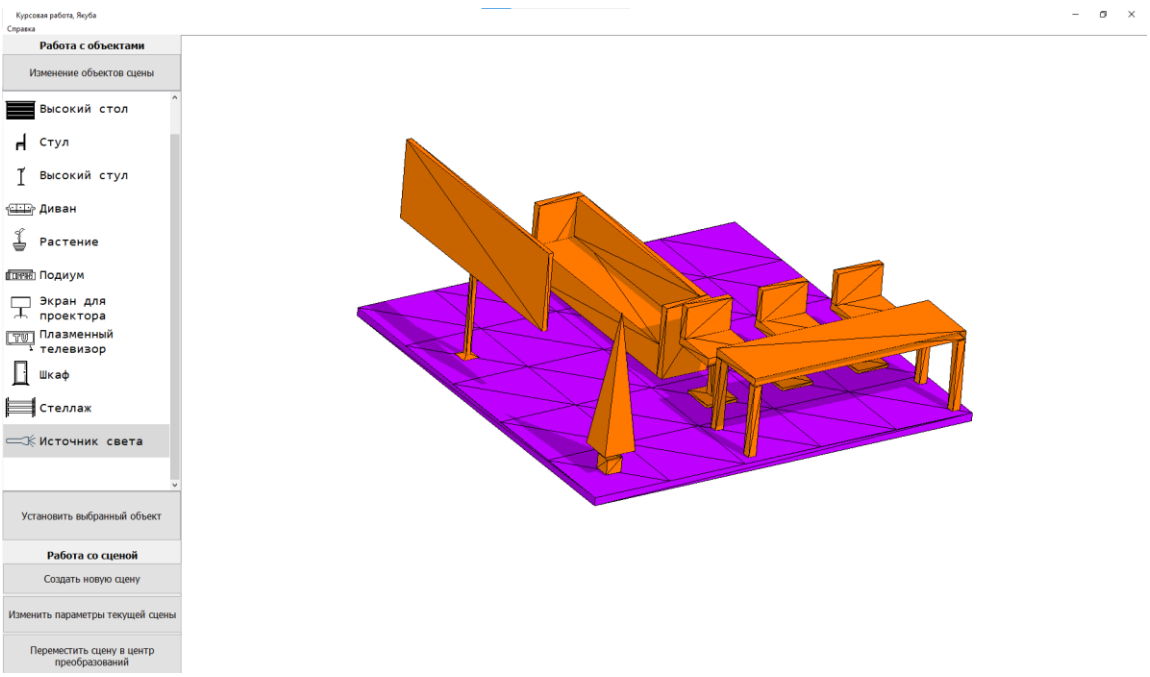

Рисунок 4.1.1 Пример работы ПО для сцены с одним источником света

На рисунке 4.1.2 приведен пример работы программы при наличии двух близко расположенных источников света. Дополнительному источнику света в качестве углов поворота заданы значениями в 20° и 30° по осям X и Y соответственно. Заметно, что в отличие от рисунка 4.1.1 тени несколько сузились.

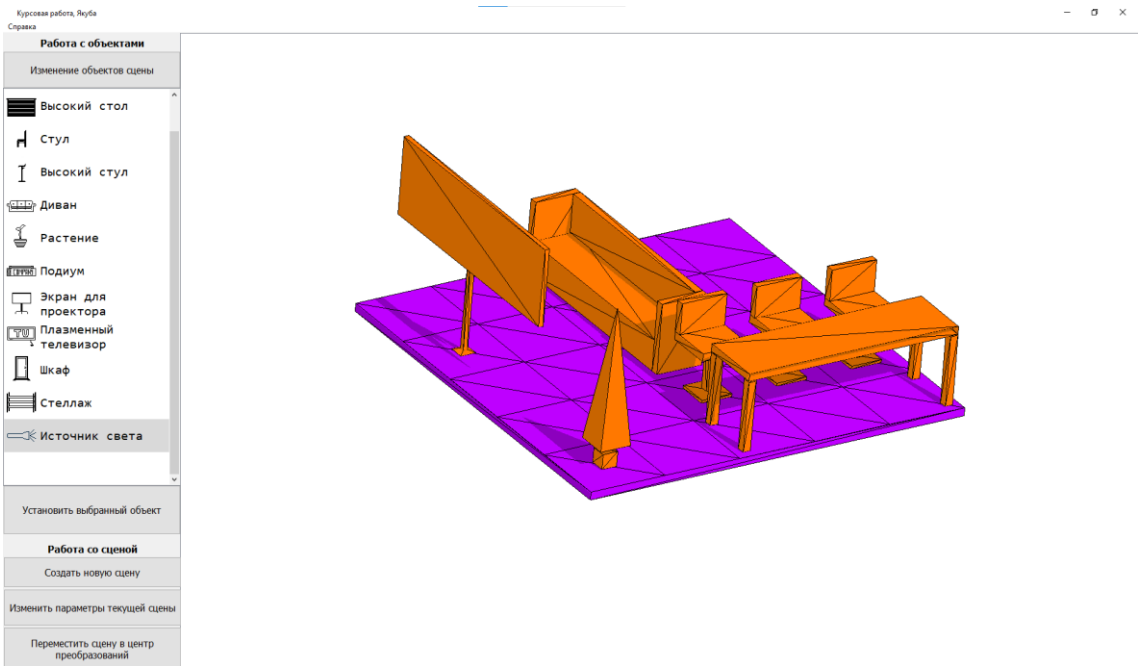

Рисунок 4.1.2 Пример работы ПО для сцены с двумя близко расположенными источниками света

На рисунках 4.1.3 и 4.1.4 приведен пример работы программы для той же сцены в разных положениях с измененным положением источника света. Для нового источника света в качестве углов поворота заданы значения в 30° и -30° по осям X и Y соответственно.

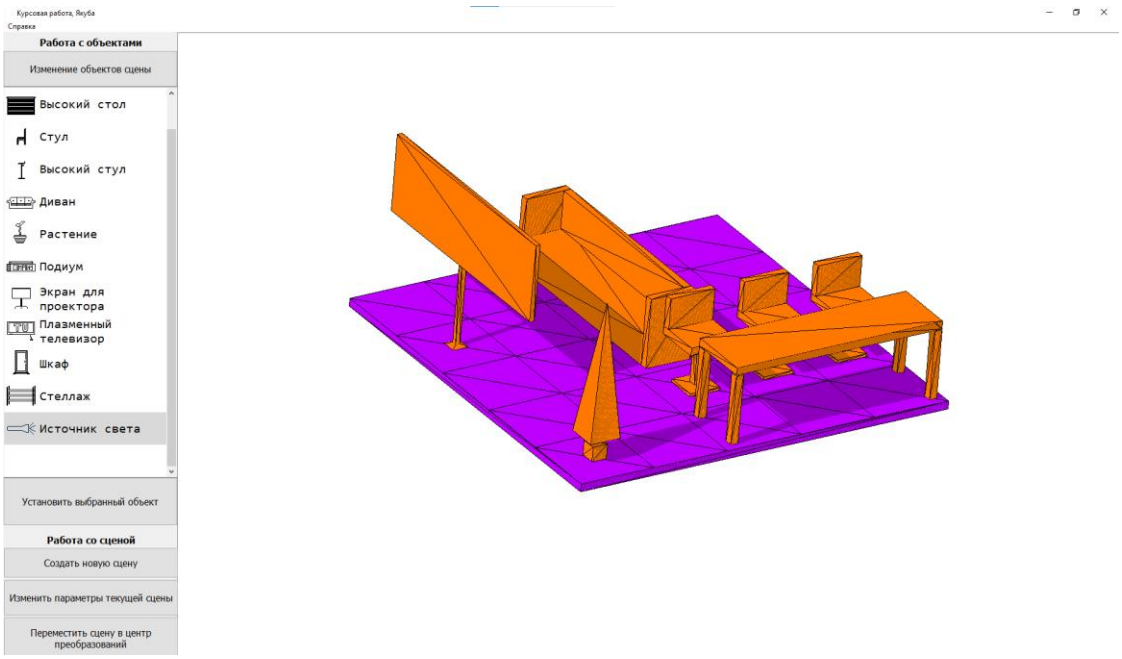

Рисунок 4.1.3 Пример работы ПО для сцены с измененным положением источника света

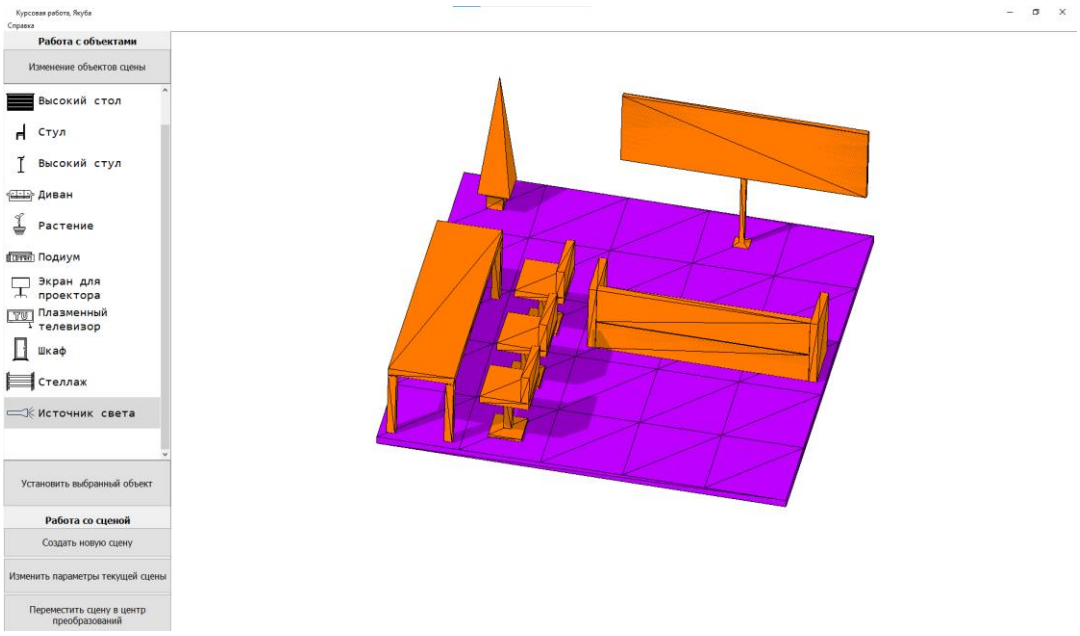

Рисунок 4.1.4 Пример работы ПО для сцены с изменённым положением источника света с другого ракурса

4.2 Постановка эксперимента

4.2.1 Цель эксперимента

Целью эксперимента является проведение оценки скорости работы программы при различных размерах визуализируемой площадки.

4.2.2 Технические характеристики

Технические характеристики ЭВМ, на котором выполнялись исследования:

- ОС: Windows 10 Pro;
- Оперативная память: 16 Гб;
- Процессор: Intel Core i7-10510U;
- Количество ядер: 4.

При проведении замеров времени ноутбук был подключен к сети электропитания.

4.2.3 Описание эксперимента

При реализации программного обеспечения в качестве механизма написания параллельно выполняющегося кода использовалась библиотека OpenMP [9]. В данном эксперименте будет показана целесообразность использования данной библиотеки.

Результаты замеров времени приведены в таблице 4.2.1. На рисунке 4.2.1 приведён график зависимости времени выполнения от размеров задаваемой плоскости сцены.

Таблица 4.2.1 Замеры времени для различного количества визуализируемых ячеек стенда

Размеры плоскости	Время выполнения без использования директив OpenMP, нс	Время выполнения с использованием директив OpenMP, нс
1x1	11 487 080	11 369 020
2x2	13 978 160	12 566 300
3x3	18 750 780	14 361 360
4x4	23 137 300	16 664 360
5x5	27 348 240	21 447 120
6x6	35 428 960	25 247 440
7x7	42 608 800	28 029 060
8x8	52 409 440	33 310 100
9x9	64 260 000	39 199 560
10x10	77 460 920	47 672 280
15x15	116 354 360	73 403 680

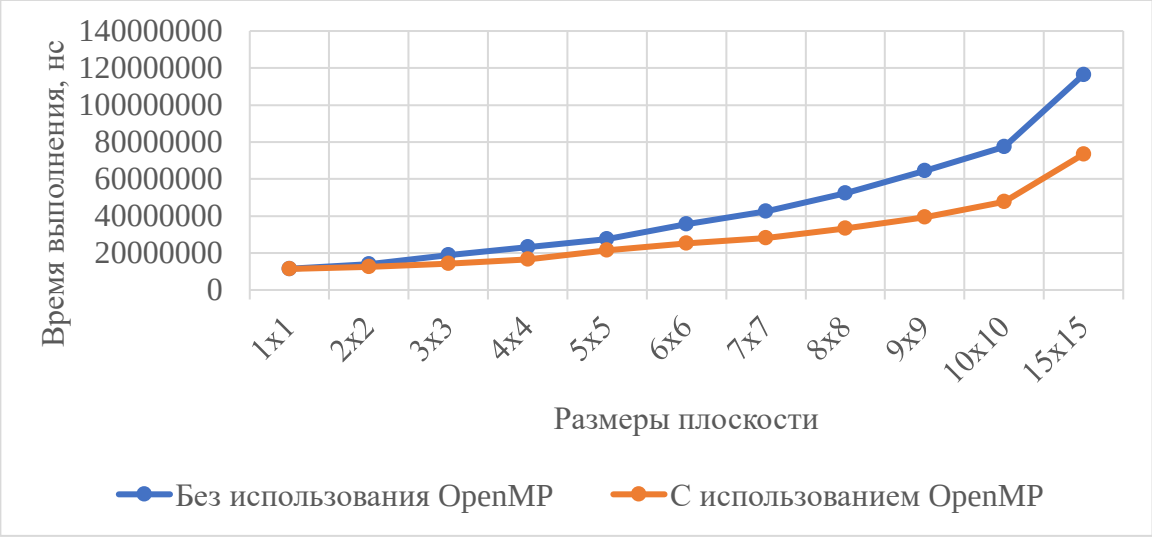

Рисунок 4.2.1 График зависимости времени выполнения от размеров задаваемой плоскости

Вывод

Из данных, предоставленных в таблице 4.2.1, видно, что реализация алгоритма с использованием функционала OpenMP не показывает себя хуже обычной реализации даже на минимальных значениях.

Из графика, предоставленного на рисунке 4.2.1 видно, что на всех рассмотренных размерах плоскости скорость выполнения алгоритма с использованием OpenMP больше, либо приблизительно равна скорости выполнения алгоритма без использования директив.

При количестве обрабатываемых клеток площадки равным 4 (случай 2x2) алгоритм с использованием OpenMP работает быстрее обычной реализации в 1.11 раз. При количестве обрабатываемых клеток равным 16 преимущество увеличивается: первый алгоритм работает в 1.39 раз быстрее. На максимально рассмотренном значении преимущество будет составлять 1.58.

В среднем по таблице алгоритм с использованием директив OpenMP будет быстрее реализации без их использования в 1.40 раз.

Из предоставленных данных видно, что параллельный алгоритм z-буфера работает быстрее обычного алгоритма в любых рассмотренных условиях.

Заключение

Во время выполнения курсового проекта было реализовано программное обеспечение для визуализации площадки и интерьера выставочных стендов.

Были формализованы объекты синтезируемой сцены. Среди алгоритмов удаления невидимых линий и поверхностей были рассмотрены алгоритм Робертса, алгоритм, использующий z-буфер, и алгоритм обратной трассировки лучей. Анализ достоинств и недостатков рассмотренных алгоритмов позволил выбрать наиболее подходящий для решения поставленной задачи.

В ходе выполнения поставленной задачи были получены знания в области компьютерной графики, закреплены навыки проектирования программного обеспечения. Были изучены возможности Qt Creator и получены навыки реализации оконных приложений. Поиск оптимальных решений для эффективной работы программного обеспечения позволил повысить навыки поиска и анализа информации.

В результате проведённой работы получено программное обеспечение, позволяющее в реальном времени выбирать из предложенного набора элементы декора, представленные в виде объемных моделей, и расставлять их по сетке сцены, заданной пользователем. Программный продукт предоставляет возможность размещения источников света, а также возможность просмотра сцены для разных положений наблюдателя.

В ходе выполнения экспериментальной части было установлено, что реализация алгоритма z-буфера с использованием директив, предоставляемых библиотекой OpenMP, позволяет ускорить реализованный алгоритм в среднем в 1.4 раза, что позволяет более быстро синтезировать изображение сцены и не заставлять пользователя ожидать большие промежутки времени между

проведением операций, таких как добавление моделей, удаление моделей, их преобразования, и преобразования сцены.

Список использованных источников

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

2. Набережнов Г. М. Трехмерное 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 (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.

7. Спецификация языка C++ [Электронный ресурс]. – Режим доступа: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3690.pdf (дата обращения 22.11.20)

8. Qt Documentation [Электронный ресурс]. – Режим доступа: https://doc.qt.io/qt-5/ (дата обращения 22.11.20)

9. OpenMP Specifications [Электронный ресурс]. – Режим доступа: https://www.openmp.org/specifications/ (дата обращения 22.11.20)

Приложение

Рисунок 1 Пример стенда

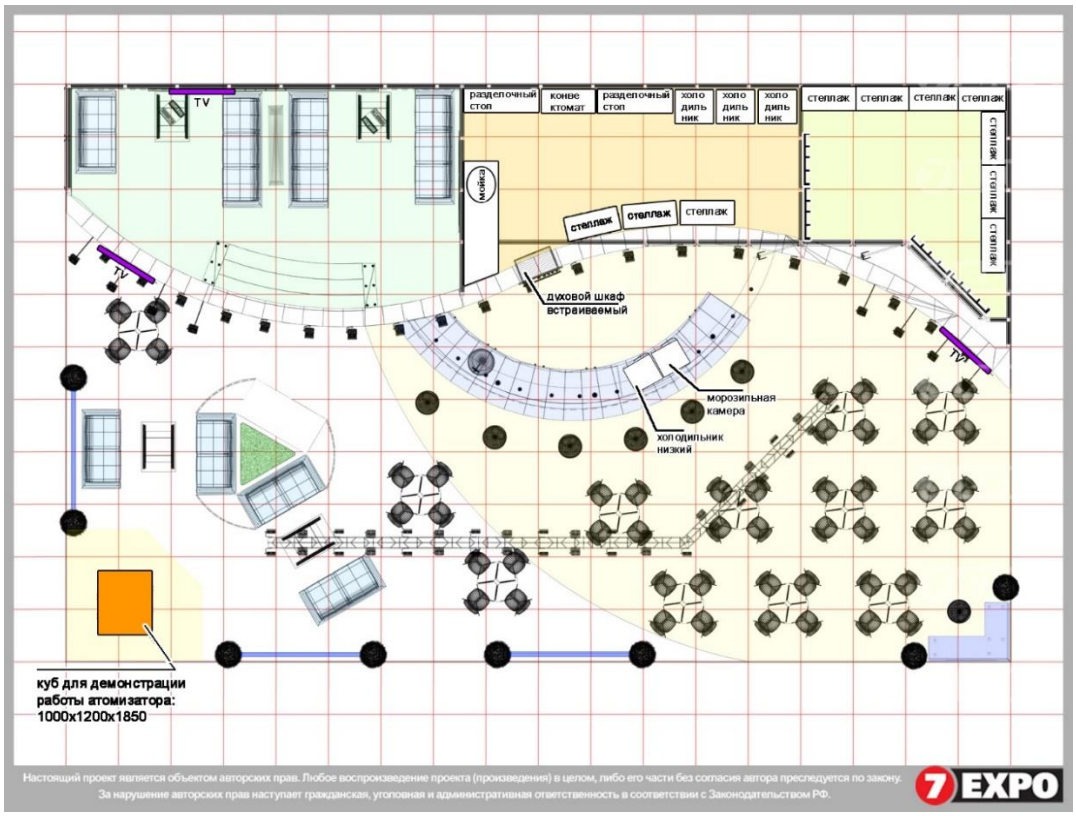

Рисунок 2 Пример стенда

Рисунок 3 Пример стенда

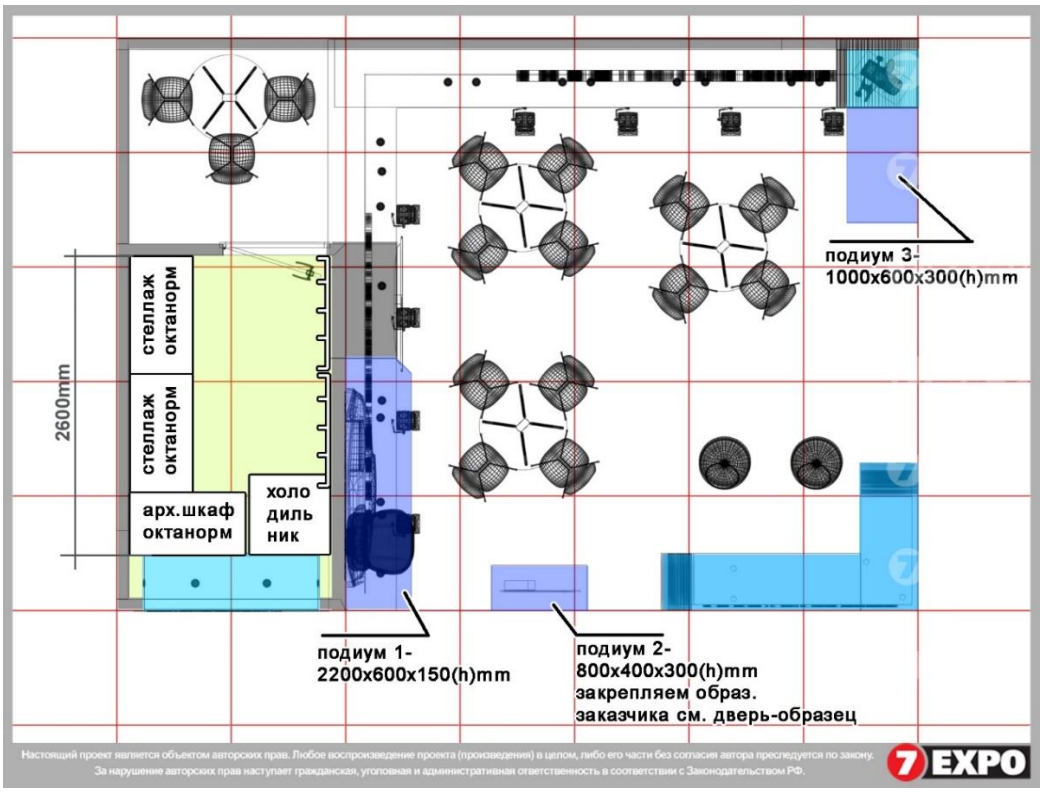

Рисунок 4 Пример стенда