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

**Индивидуальное 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 объёмных 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.

Оглавление

[Введение 4](#_Toc45454566)

[1. Аналитическая часть 5](#_Toc45454567)

[1.1 Описание объектов сцены 5](#_Toc45454568)

[1.2 Выбор формы задания трёхмерных моделей 5](#_Toc45454569)

[1.3 Выбор способа задания поверхностных моделей 6](#_Toc45454570)

[1.4 Выбор алгоритма удаления невидимых рёбер и поверхностей 8](#_Toc45454571)

[Алгоритм Робертса 8](#_Toc45454572)

[Алгоритм, использующий z-буфер 9](#_Toc45454573)

[Алгоритм обратной трассировки лучей 10](#_Toc45454574)

[1.5 Выбор алгоритма построения теней 11](#_Toc45454575)

[1.6 Определение набора необходимых моделей, поставляемых с программным обеспечением 11](#_Toc45454576)

[2. Конструкторская часть 12](#_Toc45454577)

[2.1 Общий алгоритм решения поставленной задачи 12](#_Toc45454578)

[2.2 Алгоритм z-буфера 12](#_Toc45454579)

[2.3 Модифицированный алгоритм z-буфера 13](#_Toc45454580)

[2.5 Представление данных в программном обеспечении 14](#_Toc45454581)

[3. Технологическая часть 14](#_Toc45454582)

[3.1 Выбор языка программирования и среды разработки 14](#_Toc45454583)

[3.2 Поля данных реализуемых классов 15](#_Toc45454584)

[3.3 Интерфейс программного обеспечения 16](#_Toc45454585)

[Заключение 18](#_Toc45454586)

[Список источников 19](#_Toc45454587)

[Приложения 20](#_Toc45454588)

# 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 заказа. 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 на время практики - проектирование программного обеспечения. Таким образом, необходимо выбрать оптимальные алгоритмы синтеза сцены, отвечающие вышеизложенным требованиям, создать удобный пользовательский интерфейс и реализовать выбранные алгоритмы.

# Аналитическая часть

## 1.1 Описание объектов сцены

Составляющие сцены:

* Область выставочного стенда – плоскость с заданной сеткой, на которой расставляются модели интерьера. Размеры сетки – количество квадратов (ячеек) по ширине и длине плоскости;
* Объекты интерьера – модели, которые занимают ячейки сетки или их часть. Все доступные модели поставляются вместе с программным обеспечением, внесение новых моделей в базу не предусмотрено. Имеется возможность изменить положение модели на плоскости: переместить, повернуть. Также для некоторых моделей доступна возможность выбора их длины, ширины или высоты;
* Источники света – материальные точки, испускающие свет. Положение источника задаётся трёхмерными координатами, а направление распространения света – вектором.

## 1.2 Выбор формы задания трёхмерных моделей

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

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 также 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 Выбор способа задания поверхностных моделей

Поверхностные модели задаются:

* 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:

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

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

При реализации программного продукта наиболее удобным представлением является модель, заданная полигональной сеткой – это поможет избежать проблем при описании сложных моделей. 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 Выбор алгоритма 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, при этом решая задачу только с выпуклыми телами.

Предусмотрено выполнения ряда шагов:

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 . С одной стороны – вектор взгляда наблюдателя, а с другой – вектор однородных координат точки, расположенной в бесконечности по оси .

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.

Первоначально в z-буфере находится минимально возможные значения , а в буфере кадра располагается пиксели, описывающие фон. Каждый многоугольник разлагается в растр 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.

Предполагается, что сцена уже преобразована в пространство изображения, а точка зрения находится в бесконечности на положительной полуоси и поэтому световые лучи параллельны этой же оси.

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 Выбор алгоритма построения теней

Так как в качестве алгоритма удаления невидимых линий и поверхностей в предыдущем подпункте был выбран алгоритм с использованием z-буфера, то одним из наилучших вариантов будет модификация указанного метода путём добавления вычисления теневого z-буфера из точки наблюдения, совпадающей с источником света.

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

## 1.6 Определение набора необходимых моделей, поставляемых с программным обеспечением

Опираясь на примеры визуализации выставочных стендов (см. Приложение), можно определить следующий перечень необходимых объектов сцены:

* Стол. Размеры объекта задаются пользователем по количеству занимаемых клеток области сцены;
* Высокий стол (барная стойка). Размеры объекта задаются пользователем по количеству занимаемых клеток области сцены;
* Стул. Занимает одну клетку плоскости сцены;
* Барный стул. Занимает одну клетку плоскости сцены;
* Диван. Длина объекта задаётся пользователем по количеству занимаемых клеток области сцены;
* Растение в горшке в качестве элемента декора. Занимает одну клетку плоскости сцены;
* Подиум для представления выставочных предметов заказчика. Размеры задаются пользователем по количеству занимаемых клеток плоскости сцены;
* Экран для проектора. Размеры 2 на 1 ячеек, располагается на стойке;
* Плазменный телевизор. Размеры 2 на 1 клетку, располагается на стойке;
* Шкаф. Размеры задаются пользователем по количеству занимаемых клеток плоскости сцены;
* Стеллаж с тремя полками. Длина элемента задаётся пользователем;
* Источники света. Положение задаётся координатами в трёхмерном пространстве, а направление распространения света – вектором.

При этом стоит отметить факт того, что объект, размер которого задаётся пользователем, может занимать и часть ячейки, но при этом она будет занята полностью и расположить на свободной её части другой элемент декора уже будет нельзя.

# 2. Конструкторская часть

## 2.1 Общий алгоритм решения поставленной задачи

1. Задать размеры области размещения объектов

2. Разместить объекты сцены и источники света

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 Представление данных в программном обеспечении

Ниже предоставлена таблица, описывающая данные, которые фигурируют в программном обеспечении:

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 трёхмерного пространства | Координаты по осям x, y, z |
| Объект сцены | Грани и вершины |
| Источник света | Точка трёхмерного пространства, в которой источник находится, и вектор направления распространения света |
| Вектор | Приращения единичного вектора по осям x, y, z |

# 3. Технологическая часть

## 3.1 Выбор языка программирования и среды разработки

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

1. Данный язык преподавался в рамках курса Объектно-Ориентированного Программирования;

2. Высокая вычислительная производительность. В аналитической части мною 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 Поля данных реализуемых классов

#### Математические элементы:

class **Dot2D // Точка в двухмерном пространстве**

{

private:

double xCoordinate, yCoordinate; // Координаты точки по осям X и Y

};

class **Dot3D // Точка в трёхмерном пространстве**

{

private:

double xCoordinate, yCoordinate, zCoordinate; // Координаты точки по осям X, Y, Z

};

class **MathVec // Математический вектор**

{

private:

double xInc, yInc, zInc; // Приращения вектора по каждой из осей X, Y, Z

};

#### Составляющие модели:

class **Vertex // Вершина**

{

private:

Dot3D position; // Точка расположения вершины в пространстве

std::list<size\_t> usedFacets; // Номера граней, которые содержат в себе данную вершину

};

class **Facet // Грань**

{

private:

std::list<size\_t> usedDots; // Номера точек, на которых построена грань

};

#### Объекты сцены:

class **PolModel // Полигональная модель**

{

private:

std::list<Vertex> vertices; // Вершины модели

std::list<Facet> facets; // Грани модели

};

class **illuminant // Источник света**

{

private:

Dot3D position; // Точка расположения источника в пространстве

MathVec vector; // Вектор распространения света источника

};

#### Сцена:

class **cellScene // Область размещения моделей**

{

private:

size\_t width, length; // Ширина и длина области

size\_t modelsNum; // Количество размещённых моделей на сцене

std::vector<PolModel> models; // Модели, размещённые на сцене

size\_t illumNum; // Количество размещённых источников света на сцене

std::vector<Illuminant> illuminants; // Источники, размещённые на сцене

std::vector<std::vector<bool>> cellsBoolMap; // Таблица занятости ячеек сцены

std::vector<std::vector<size\_t>> cellsModelsMap; // Таблица, описывающая номера моделей, которые занимают ячейки

};

## 3.3 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.

# 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. Полигональная сетка [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%B3%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D0%BA%D0%B0> (дата обращения 07.07.20)

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

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

5. Удаление невидимых поверхностей и линий [Электронный ресурс]. – Режим доступа: <https://sites.google.com/site/komputernaagrafika22/osnovnye-oblasti-primenenia/udalenie-nevidimyh-poverhnostej-i-linij> (дата обращения 07.07.20)

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