



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 КУРСОВОЙ РАБОТЕ*

### *НА ТЕМУ:*

«Разработка системы генерации и управления трехмерными телами  
для заполнения лунок на площадке»

Студент ИУ7-52Б  
(Группа)

И. В. Смирнов  
(Подпись, дата) (И. О. Фамилия)

Руководитель курсовой работы

А. В. Куров  
(Подпись, дата) (И. О. Фамилия)

2024 г.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>5</b>
<b>1 Аналитическая часть</b>	<b>6</b>
1.1 Описание объектов сцены . . . . .	6
1.2 Анализ способов задания моделей . . . . .	6
1.2.1 Каркасная модель . . . . .	7
1.2.2 Поверхностная модель . . . . .	7
1.2.3 Твердотельная модель . . . . .	7
1.2.4 Воксельная модель . . . . .	7
1.2.5 Выбор способа описания модели . . . . .	7
1.3 Анализ алгоритмов удаления невидимых поверхностей . . . . .	8
1.3.1 Алгоритм Робертса . . . . .	8
1.3.2 Алгоритм Z-буфера . . . . .	9
1.3.3 Алгоритм Варнока . . . . .	9
1.3.4 Алгоритм обратной трассировки лучей . . . . .	10
1.3.5 Выбор алгоритма удаления невидимых поверхностей . . . . .	11
1.4 Анализ алгоритмов закраски . . . . .	11
1.4.1 Плоская закрашка . . . . .	11
1.4.2 Закраска по Гуро . . . . .	12
1.4.3 Закраска по Фонгу . . . . .	12
1.4.4 Выбор алгоритма закраски . . . . .	13
1.5 Анализ моделей освещения . . . . .	13
1.5.1 Модель освещения Ламберта . . . . .	14
1.5.2 Модель освещения Фонга . . . . .	14
1.5.3 Модель Блинна-Фонга . . . . .	14
1.5.4 Выбор модели освещения . . . . .	15
<b>2 Конструкторская часть</b>	<b>16</b>
2.1 Требования к программному обеспечению . . . . .	16
2.2 Используемые структуры данных . . . . .	16
2.3 Алгоритм построения изображения . . . . .	18

2.4	Перевод координат в экранные . . . . .	21
2.5	Алгоритм, использующий Z-буфер . . . . .	22
2.6	Алгоритм моделирования лунок . . . . .	22
2.7	Вычисление нормалей . . . . .	23
2.8	Освещение . . . . .	24
<b>3</b>	<b>Технологическая часть</b>	<b>25</b>
3.1	Средства реализации . . . . .	25
3.2	Структура программы . . . . .	25
3.3	Схемы алгоритмов . . . . .	26
3.4	Интерфейс программного обеспечения . . . . .	29
<b>4</b>	<b>Исследовательская часть</b>	<b>32</b>
4.1	Технические характеристики . . . . .	32
4.2	Замеры времени . . . . .	32
	<b>ЗАКЛЮЧЕНИЕ</b>	<b>35</b>
	<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>36</b>
	<b>ПРИЛОЖЕНИЕ А</b>	<b>37</b>
	<b>ПРИЛОЖЕНИЕ Б</b>	<b>38</b>

## ВВЕДЕНИЕ

В современном мире компьютерная графика и моделирование активно применяются в различных областях, таких как разработка игр, архитектурное проектирование и научные исследования. Одной из актуальных задач является моделирование физических объектов и их взаимодействия с окружающей средой в виртуальном пространстве.

**Цель курсовой работы** — разработка программного обеспечения для моделирования прямоугольной площадки с лунками, соответствующими трехмерным телам (сфера, куб, параллелепипед, шестигранная призма) и возможностью генерации тел с их падением на площадку. Для достижения поставленной цели необходимо решить следующие задачи:

- описать список доступных к размещению на сцене объектов, формализовать эти объекты;
- выбрать алгоритмы компьютерной графики для визуализации сцены и объектов на ней;
- выбрать язык программирования и среду разработки;
- разработать программное обеспечение и реализовать выбранные алгоритмы визуализации;
- провести замеры временных характеристик разработанного программного обеспечения.

# 1 Аналитическая часть

В данном разделе проводится анализ существующих алгоритмов построения изображений и выбор подходящих алгоритмов для решения задачи.

## 1.1 Описание объектов сцены

Сцена состоит из следующих основных объектов:

- Площадка — прямоугольная плоская поверхность, на которой располагаются лунки для тел, которые имеют заданный размер и координаты центра;
- Тела — трехмерные объекты (сфера, куб, параллелепипед, шестигранная призма), задаваемые размером, цветом и координатами центра. Тела генерируются и размещаются над площадкой, после чего пользователь может запускать процесс их падения;
- Лунки — углубления на площадке, задаваемые размером и координатами центра. Взаимодействие тела и лунки осуществляется только при помощи падения тела внутрь углубления лунки.
- Источник света — определяет освещенность сцены, что позволяет визуализировать тени и эффекты освещения на телах и площадке. Задается положением на сцене и интенсивностью;
- Камера — используется для изменения положения обзора сцены, предоставляя пользователю возможность наблюдать процесс падения тел под разными углами. Характеризуется своим положением и направлением просмотра.

## 1.2 Анализ способов задания моделей

В компьютерной графике существуют четыре основных типа моделей для описания трехмерных объектов: каркасная, поверхностная, твердотельная и воксельная модели [1]. Они предоставляют различные способы представления объектов и позволяют достичь правильного отображения их формы и размеров на сцене.

### **1.2.1 Каркасная модель**

Каркасная модель — представляет объект как набор вершин и ребер, что позволяет экономить память. Однако, этот метод не всегда точно передает форму объекта.

### **1.2.2 Поверхностная модель**

Поверхностная модель — определяет поверхность объекта с помощью полигонов, что позволяет более точно отображать форму тел и их взаимодействие с лунками на площадке.

### **1.2.3 Твёрдотельная модель**

Твёрдотельная модель — добавляет информацию о материале объекта. Чтобы учитывать объемные свойства материала твердотельная модель содержит данные не только о поверхности объекта, но и о его внутренней структуре. Однако в данной работе она не применяется из-за высокой ресурсоемкости.

### **1.2.4 Воксельная модель**

Воксельная модель — это трехмерный растр. Воксел — это элемент объема. Подобно тому, как пикселы располагаются на плоскости 2D-изображения, так и воксели образуют трехмерные объекты в определенном объеме.

### **1.2.5 Выбор способа описания модели**

Для моделирования тел в рамках работы была выбрана поверхностная модель, так как она позволяет эффективно отображать сложные формы, такие как сфера и шестигранная призма, с учетом их взаимодействия с площадкой.

## 1.3 Анализ алгоритмов удаления невидимых поверхностей

Удаление невидимых поверхностей является фундаментальной задачей в компьютерной графике, обеспечивая корректное отображение сцены на экране. Это особенно важно при моделировании сложных объектов и их взаимодействий, как в случае с площадкой и падающими телами.

### 1.3.1 Алгоритм Робертса

Алгоритм Робертса представляет собой один из первых методов удаления невидимых поверхностей, работающий в пространстве объектов. Алгоритм выполняется в 4 этапа [2]:

- подготовка исходных данных — составление матрицы тела для каждого тела сцены;
- удаление ребер, экранируемых самим телом;
- удаление ребер, экранируемых другими телами;
- удаление линий пересечения тел, экранируемых самими телами и другими телами, связанными отношением протыкания.

#### Преимущества:

- Простота реализации для простых, выпуклых объектов;
- Точное определение видимости граней без аппроксимаций.

#### Недостатки:

- Неэффективен для сложных и невыпуклых объектов;
- Высокая вычислительная сложность при большом количестве граней;
- Не подходит для динамических сцен с изменяющимся положением объектов.

### 1.3.2 Алгоритм Z-буфера

Алгоритм Z-буфера является одним из наиболее распространенных методов удаления невидимых поверхностей в компьютерной графике. Он использует дополнительный буфер глубины (Z-буфер), где для каждого пикселя хранится информация о глубине ближайшего к наблюдателю объекта.

#### Принцип работы:

- 1) Инициализация Z-буфера максимальным значением глубины;
- 2) При отрисовке каждого полигона вычисляется глубина его точек;
- 3) Если глубина текущего пикселя больше значения в Z-буфере, пиксель отображается, и значение глубины обновляется.

#### Преимущества:

- Простота реализации и эффективность;
- Поддержка сложных сцен с пересекающимися объектами;
- Линейная зависимость от количества пикселей, а не от количества объектов.

#### Недостатки:

- Требуется дополнительная память для хранения Z-буфера;
- Возможны артефакты при ограниченной точности буфера [3].

### 1.3.3 Алгоритм Варнока

Алгоритм Варнока использует подход рекурсивного разбиения области изображения для определения видимых поверхностей. Сцена делится на квадранты, и для каждого вычисляется простейший случай видимости:

- 1) Если в области нет объектов, она закрашивается фоновым цветом;



- 2) Если область содержит один полигон, он отображается;
- 3) Если область сложная, она разбивается дальше.

**Преимущества:**

- Эффективен для сцен с разреженными объектами;
- Не требует большого объема памяти.

**Недостатки:**

- Рекурсивная природа алгоритма может привести к высокой вычислительной нагрузке;
- Не подходит для сцен с высокой детализацией на малых участках.

### **1.3.4 Алгоритм обратной трассировки лучей**

Алгоритм обратной трассировки лучей (Ray Tracing) моделирует путь лучей от наблюдателя к источникам света, определяя пересечения с объектами сцены. Он позволяет получить фотореалистичное изображение с отражениями, преломлениями и тенями.

**Преимущества:**

- Высокая реалистичность и качество изображения;
- Точное моделирование оптических эффектов.

**Недостатки:**

- Высокая вычислительная сложность;
- Необходимость значительных ресурсов, что затрудняет использование в реальном времени без аппаратного ускорения;

### 1.3.5 Выбор алгоритма удаления невидимых поверхностей

Выбор сделан в пользу алгоритма Z-буфера благодаря его простоте, что важно для отображения сцены в реальном времени, а также благодаря поддержке динамических изменений, так как тела движутся и взаимодействуют с площадкой.

## 1.4 Анализ алгоритмов закраски

Закраска поверхностей определяет визуальное восприятие объектов на сцене, влияя на реалистичность и эстетическое качество изображения.

### 1.4.1 Плоская закраска

Плоская закраска (Flat Shading) подразумевает однородное закрашивание каждого полигона одним цветом, вычисленным на основе нормали к его поверхности и источника света.

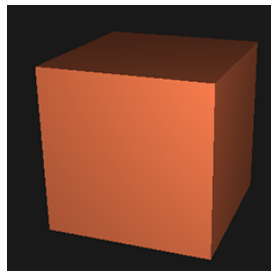


Рисунок 1.1 – Пример плоской закраски

#### Преимущества:

- Очень высокая скорость обработки;
- Простота реализации.

#### Недостатки:

- Объекты выглядят угловатыми и нереалистичными;
- Не учитываются плавные переходы света и тени между полигонами.

## 1.4.2 Закраска по Гуро

Метод Гуро (Gouraud Shading) использует интерполяцию интенсивности цвета между вершинами полигона. Интенсивности в вершинах вычисляются на основе нормалей и освещения, а затем плавно изменяются по поверхности.

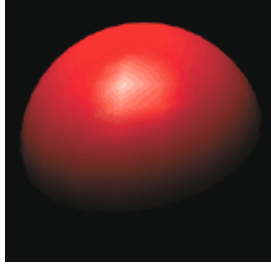


Рисунок 1.2 – Пример закрашки по Гуро

### Преимущества:

- Плавные градиенты цвета между полигонами;
- Улучшенное качество изображения по сравнению с плоской закрашкой.

### Недостатки:

- Возможна потеря детализации бликов, так как они могут не попадать на вершины;
- Сложнее реализуется, чем простая закрашка.

## 1.4.3 Закраска по Фонгу

Закраска по Фонгу (Phong Shading) интерполирует нормали между вершинами и вычисляет освещение для каждого пикселя, используя интерполированные нормали.

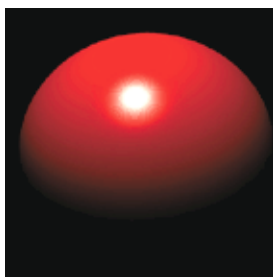


Рисунок 1.3 – Пример закраски по Фонгу

#### **Преимущества:**

- Высокая реалистичность, особенно в отображении бликов;
- Плавные и точные переходы света и тени.

#### **Недостатки:**

- Более высокая вычислительная нагрузка;
- Сложность реализации по сравнению с методом Гуро [4].

### **1.4.4 Выбор алгоритма закраски**

В курсовой работе оптимальным выбором является закраска по Гуро, так как алгоритм обеспечивает достаточную реалистичность без значительного увеличения вычислительных затрат.

## **1.5 Анализ моделей освещения**

Модель освещения определяет, как свет взаимодействует с поверхностями объектов, что существенно влияет на реализм сцены. Выделяют три основные модели освещения: модель Ламберта, модель Фонга и модель Блинна—Фонга [5].

### 1.5.1 Модель освещения Ламберта

Модель Ламберта описывает идеальное диффузное отражение света от матовой поверхности. Интенсивность отраженного света зависит от косинуса угла между нормалью к поверхности и направлением на источник света.

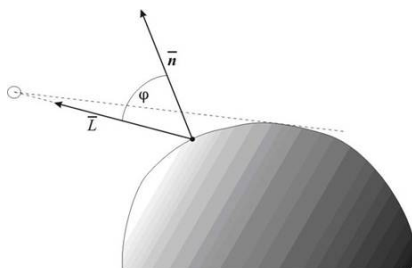


Рисунок 1.4 – Модель освещения Ламберта

### 1.5.2 Модель освещения Фонга

Модель Фонга расширяет модель Ламберта, добавляя зеркальную составляющую. Это позволяет отображать блики и более сложные эффекты освещения, делая объекты более реалистичными.

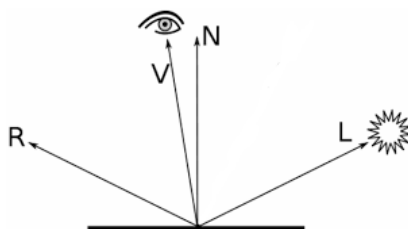


Рисунок 1.5 – Модель освещения Фонга

### 1.5.3 Модель Блинна-Фонга

В 1977 году Джеймсом Ф. Блинном была представлена модель освещения Блинна-Фонга, как дополнение к модели Фонга. Модель использует иной подход к расчету зеркальной компоненты. Вместо вектора отражения используется медианный вектор, который представляет из себя единичный вектор точно посередине между направлением обзора и направлением света. Чем ближе этот вектор к нормали поверхности, тем больше будет вклад зеркальной компоненты.

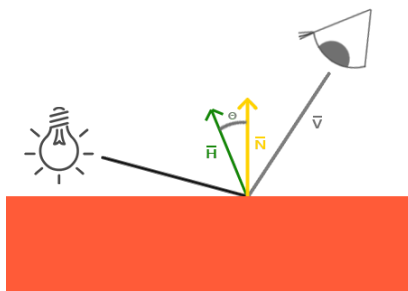


Рисунок 1.6 – Модель освещения Блинна-Фонга

#### 1.5.4 Выбор модели освещения

С учетом специфики курсовой работы, модель освещения Ламберта является наиболее подходящей. Модель обеспечивает приемлемое качество освещения, легко реализуется и интегрируется с закраской по Гуро, а также менее требовательна к вычислительным ресурсам, что важно для поддержания производительности в реальном времени.

### ВЫВОД

В данном разделе проведен анализ существующих алгоритмов построения изображений. Для решения задачи была выбрана поверхностная модель, алгоритм, использующий Z-буфер, закраска по Гуро и модель освещения Ламберта.

## 2 Конструкторская часть

В данном разделе представлены требования к программному обеспечению, рассмотрены структуры данных, алгоритмы и математические уравнения, выбранные для построения сцены.

### 2.1 Требования к программному обеспечению

Программное обеспечение должно обеспечивать следующую функциональность:

- Генерацию тел: шара, куба, параллелепипеда и шестигранной призмы, с возможностью задания их размеров, цветов и координат центра;
- Моделирование клетчатой прямоугольной площадки с лунками, соответствующими указанным телам;
- Реализацию процесса падения тел на площадку по запросу пользователя;
- Определение попадания тела в соответствующую ей лунку, вывод соответствующих сообщений и удаление тела и лунки при успешном попадании;
- Вывод сообщений при непопадании тела или попадании в чужую лунку и удаление тел из сцены;
- Возможность изменения положения камеры для обзора сцены под разными углами и ракурсами;
- Реализация источника света с возможностью задания его положения и интенсивности.

### 2.2 Используемые структуры данных

Для реализации работы программы разработаны следующие основные структуры данных:

- 1) Сцена содержит:

- Массив объектов сцены, включая модели, лунки и площадку;
- Камеру и источник освещения;
- Методы для добавления и удаления объектов со сцены.

2) Трехмерное тело включает в себя:

- Массив вершин;
- Массив граней, формирующих аппроксимацию поверхности;
- Массив нормалей к вершинам для расчета освещения;
- Цвет поверхности тела;
- Положение и вращение тела в пространстве;
- Матрицу преобразований для применения трансформаций к телу.

3) Вершина содержит:

- Положение в пространстве;
- Нормаль в данной вершине.

4) Грань модели содержит:

- Индексы вершин A, B, C в списке вершин модели, образующих грань.

5) Камера содержит:

- Положение в пространстве;
- Направление взгляда;
- Методы для изменения положения и ориентации камеры.

6) Источник освещения содержит:

- Положение в пространстве;
- Интенсивность света.

7) Для представления тел и соответствующих им лунок необходима дополнительная информация:



- Тип тела или лунки;
- Размер и другие параметры, специфичные для конкретного тела.

## 2.3 Алгоритм построения изображения

Алгоритм генерации изображения представлен в виде диаграммы, оформленной в соответствии с нотацией IDEF0 и отражающей общую декомпозицию алгоритма [6].

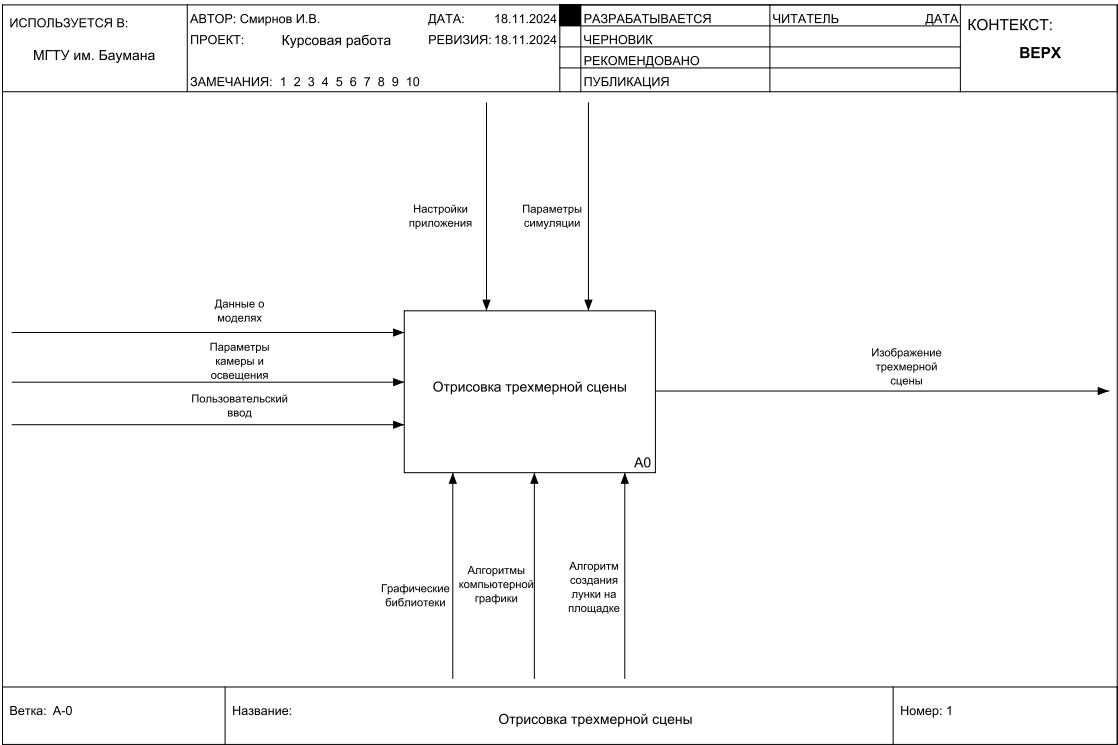


Рисунок 2.1 – Функциональная схема алгоритма построения изображения, декомпозиция верхнего уровня

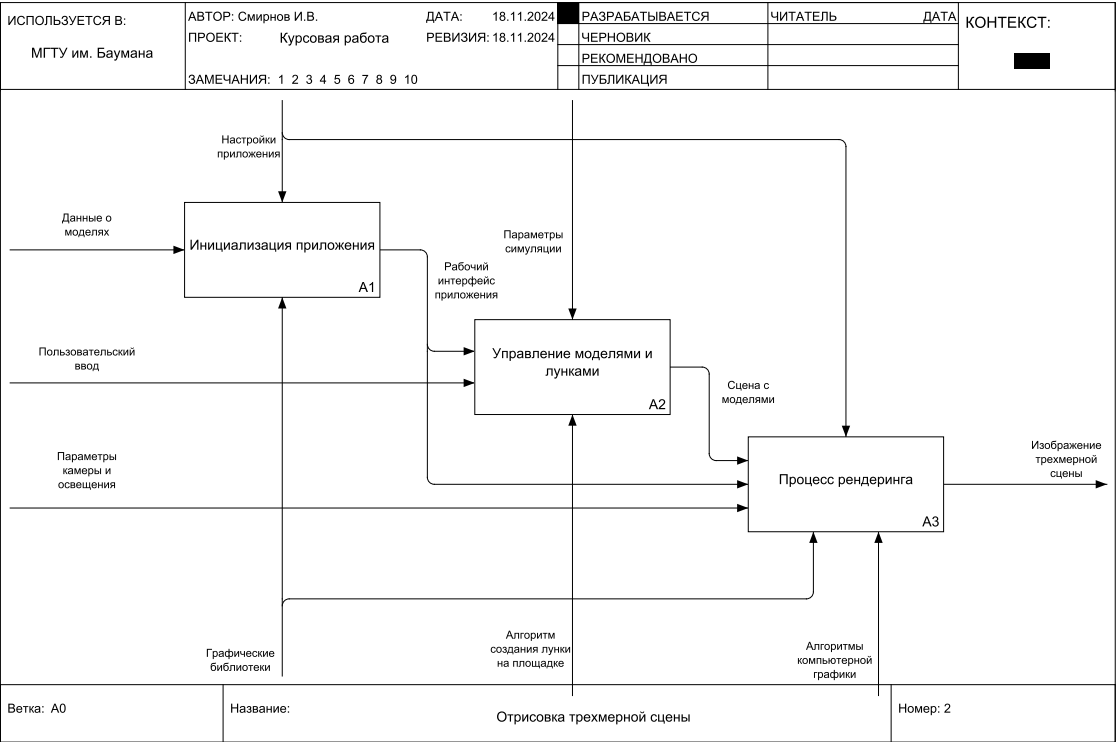


Рисунок 2.2 – Функциональная схема алгоритма построения изображения, декомпозиция уровня A0

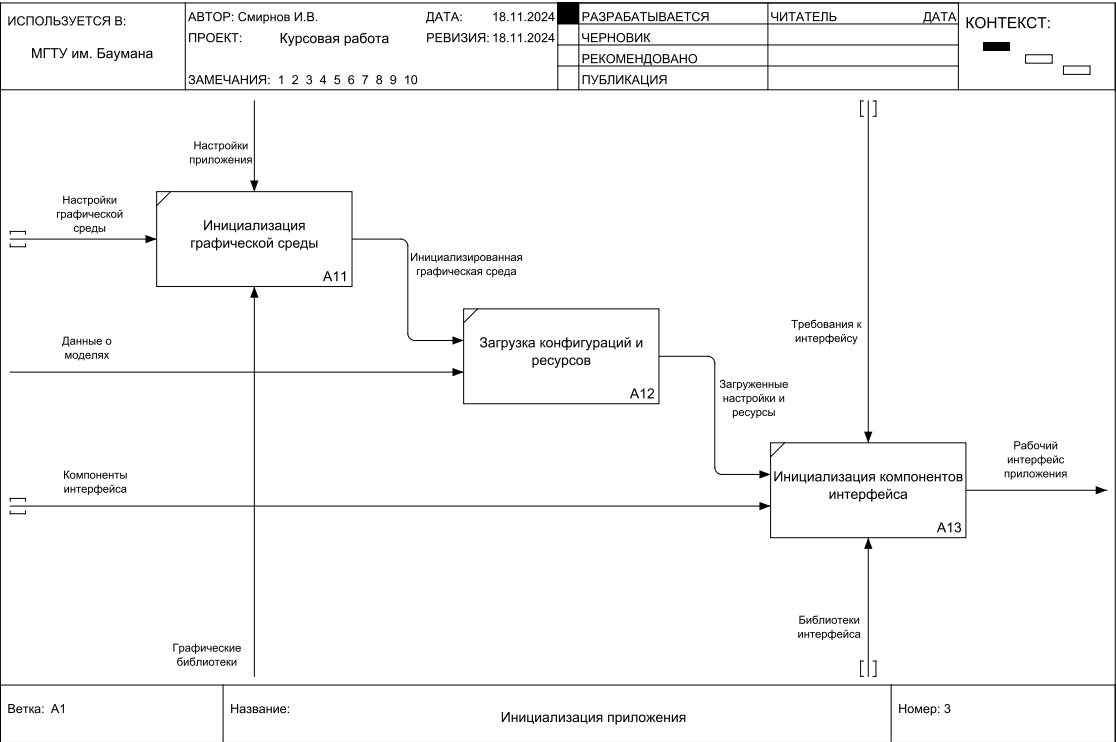


Рисунок 2.3 – Функциональная схема алгоритма построения изображения, декомпозиция уровня A1

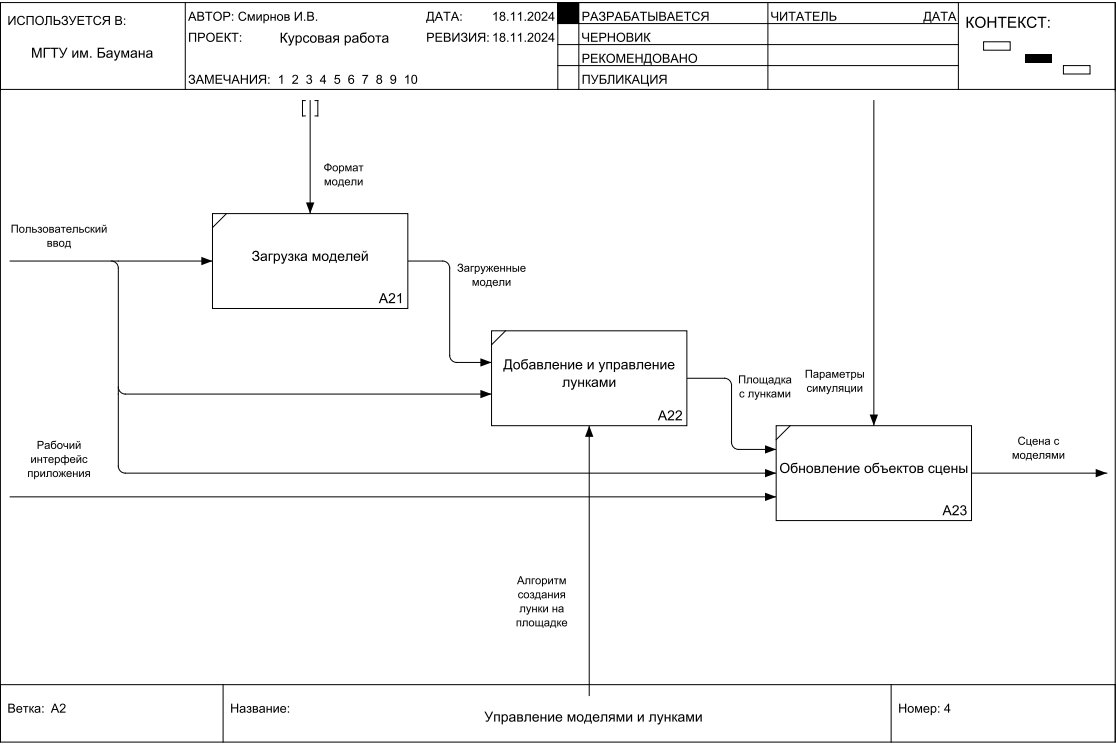


Рисунок 2.4 – Функциональная схема алгоритма построения изображения, декомпозиция уровня A2

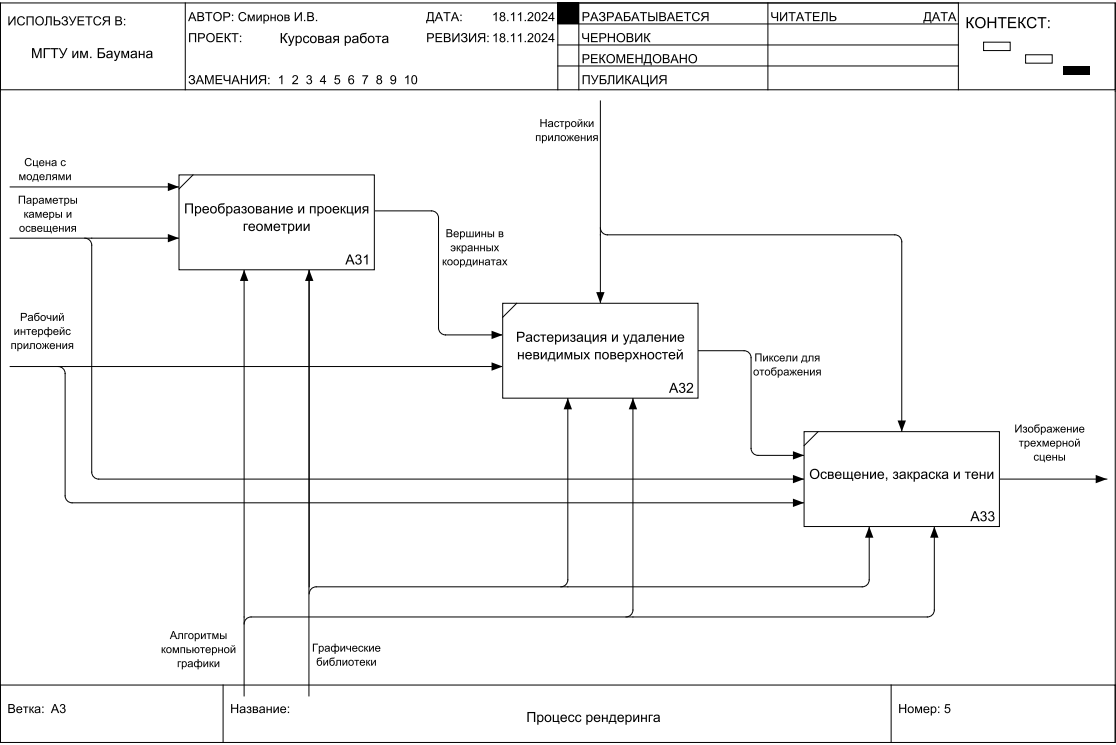


Рисунок 2.5 – Функциональная схема алгоритма построения изображения, декомпозиция уровня A3

## 2.4 Перевод координат в экранные

Для преобразования координат из мирового пространства в экранное используется следующая последовательность шагов. Сначала применяется матрица модели, включающая трансформации объекта, такие как повороты, масштабирование и перемещения. Затем применяется видовая матрица, определяющая положение и ориентацию камеры. Видовая матрица вычисляется как:

$$M_{\text{вид}} = M_{\text{пер}} \cdot M_{\text{пов}}, \quad (2.1)$$

где  $M_{\text{пер}}$  отвечает за перенос точки камеры в начало координат, а  $M_{\text{пов}}$  выравнивает оси камеры. После этого применяется проекционная матрица, которая используется для перехода от трехмерных координат к двумерным. Для перспективной проекции используется следующая матрица:

$$M_{\text{проезк}} = \begin{bmatrix} \frac{1}{\text{tg}(fov/2) \cdot \text{aspect}} & 0 & 0 & 0 \\ 0 & \frac{1}{\text{tg}(fov/2)} & 0 & 0 \\ 0 & 0 & \frac{far}{far - near} & 1 \\ 0 & 0 & -\frac{far \cdot near}{far - near} & 0 \end{bmatrix}, \quad (2.2)$$

где  $fov$  — угол обзора,  $\text{aspect}$  — соотношение сторон экрана, и  $far, near$  — расстояния до ближней и дальней плоскостей отсечения. Наконец, применяется матрица  $\text{viewport}$ , преобразующая координаты в экранные. Она задается как:

$$M_{\text{viewport}} = \begin{bmatrix} \frac{W}{2} & 0 & 0 & \frac{W}{2} \\ 0 & -\frac{H}{2} & 0 & \frac{H}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.3)$$

где  $W, H$  — ширина и высота экрана в пикселях. После применения всех матриц конечные координаты преобразуются в экранные.

## 2.5 Алгоритм, использующий Z-буфер

Алгоритм Z-буфера применяется для удаления невидимых поверхностей. На первом этапе Z-буфер инициализируется максимальными значениями глубины:

$$z_{buffer}[x][y] = z_{max}. \quad (2.4)$$

Далее, для каждого треугольника сцены его вершины проецируются в экранное пространство, и треугольник растеризуется по пикселям. Для каждого пикселя вычисляется глубина текущего треугольника по формуле:

$$z = \frac{Ax + By + C}{D}, \quad (2.5)$$

где  $A, B, C, D$  — коэффициенты плоскости треугольника. Если глубина пикселя меньше значения в Z-буфере, то значение Z-буфера и цвета пикселя обновляется:

$$z_{buffer}[x][y] = z, \quad color[x][y] = color_t. \quad (2.6)$$

## 2.6 Алгоритм моделирования лунок

Лунки моделируются путем изменения высоты и нормалей вершин сетки площадки. Для каждой лунки определяются ее тип, координаты центра в сетке  $(x_0, z_0)$  и размеры. Для сферических лунок смещение по оси  $y$  вычисляется по уравнению сферы:

$$y = -\sqrt{r^2 - (x - x_0)^2 - (z - z_0)^2}, \quad (2.7)$$

где  $(x, z)$  — координаты выбранной области сетки площадки для лунки,  $r$  — радиус лунки. Для остальных типов лунок смещение по  $y$  всегда фиксированное и вычисляется как:

$$y = -a, \quad (2.8)$$

где  $a$  — глубина лунки.

После определения новых высот для вершин внутри лунки, для шестигранных и цилиндрических лунок по границе области формируются вертикальные стенки. Алгоритм следующий:

- 1) Определяется периметр лунки: набор вершин, где лунка переходит из углубления в исходный уровень.
- 2) Для каждой пары соседних вершин на периметре (например,  $(v_1, v_2)$ ) создаются дополнительные вершины, смещенные вниз на глубину лунки, формируя четыре точки: верхние точки периметра  $(v_1, v_2)$  с  $y = 0$ , и соответствующие им нижние точки  $(v_3, v_4)$  с  $y = -a$ .
- 3) С помощью этих четырех точек формируется две треугольные грани (два треугольника), образуя вертикальную стенку.

Для шестиугольных и цилиндрических лунок стены немного смещаются наружу от центра лунки на малую величину  $e$ :

$$(x', z') = (x + e \cdot \frac{dx}{\sqrt{dx^2 + dz^2}}, z + e \cdot \frac{dz}{\sqrt{dx^2 + dz^2}}), \quad (2.9)$$

где  $dx = x - x_0$ ,  $dz = z - z_0$ . Это позволяет визуально отделить стенки лунки от основной поверхности.

## 2.7 Вычисление нормалей

Нормали необходимы для освещения и корректного отображения поверхностей. Сначала для всех вершин нормали инициализируются нулевым вектором. Для каждого треугольника вычисляется нормаль грани по следующей формуле:

$$\vec{N} = \frac{(\vec{V}_2 - \vec{V}_1) \times (\vec{V}_3 - \vec{V}_1)}{|(\vec{V}_2 - \vec{V}_1) \times (\vec{V}_3 - \vec{V}_1)|}, \quad (2.10)$$

где  $\vec{V}_1, \vec{V}_2, \vec{V}_3$  — вершины треугольника. Вычисленная нормаль добавляется к нормальям ее вершин. Затем нормали всех вершин нормализуются:

$$\vec{N}_v = \frac{\vec{N}}{|\vec{N}|}, \quad (2.11)$$

где  $\vec{N}_v$  — суммарный вектор нормали вершины.

## 2.8 Освещение

В программе используется всенаправленный источник света, расположенный в заданной точке пространства. Он испускает свет во всех направлениях, а освещение рассчитывается по модели Ламберта.

Для каждой вершины вычисляется итоговая освещенность  $I$  как сумма фонового и диффузного освещения:

$$I = I_a + I_d, \quad (2.12)$$

где  $I_a$  — фоновая составляющая (фиксированная величина), а  $I_d$  — диффузное освещение, рассчитываемое по формуле:

$$I_d = \max(0, \vec{N} \cdot \vec{L}). \quad (2.13)$$

Здесь  $\vec{N}$  — нормаль вершины, а  $\vec{L}$  — нормализованный вектор направления от вершины к источнику света. Значение  $\max(0, \vec{N} \cdot \vec{L})$  исключает отрицательные значения, которые могут возникнуть, если нормаль и направление света образуют тупой угол.

Результирующая освещенность каждой вершины интерполируется по граням треугольников, что позволяет плавно переходить между яркими и темными областями объекта.

### ВЫВОД

В данном разделе были представлены требования к программному обеспечению, рассмотрены структуры данных, алгоритмы и математические уравнения, выбранные для построения сцены.

## 3 Технологическая часть

В данной части рассматривается выбор средств реализации, описывается структура классов программы и приводится интерфейс программного обеспечения.

### 3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *C#* [7].

В качестве среды разработки была выбрана *VisualStudio 2022*. Для разработки интерфейса и работы с пикселями изображения была выбрана платформа *WindowsForms* [8].

Выбор обусловлен наличием стандартной библиотеки для работы с векторами и матрицами (*System.Numerics*), библиотеки для работы с графикой (*System.Drawing*) и *LINQ*—выражениями для работы с коллекциями, такие как список тел и список лунок. Используемые инструменты обладают полной функциональностью для разработки, профилирования и отладки необходимой программы.

### 3.2 Структура программы

Разработанная программа состоит из следующих классов. Математические классы:

- *Vector3D* — класс для работы с трехмерными векторами;
- *Matrix4x4* — класс для работы с матрицами 4x4.

Классы для работы с моделями:

- *Mesh* — класс, представляющий трехмерную модель;
- *Vertex* — класс, представляющий вершину;
- *Face* — класс, представляющий полигон (треугольник) в сетке;



- Indentation — класс, представляющий лунку (углубление) на поверхности.

Вспомогательные классы:

- Camera — класс, представляющий камеру;
- Light — класс, представляющий источник света;
- Scene — класс, представляющий сцену;
- Form — класс, представляющий интерфейс.

На рисунке 3.1 представлена диаграмма разработанных классов.

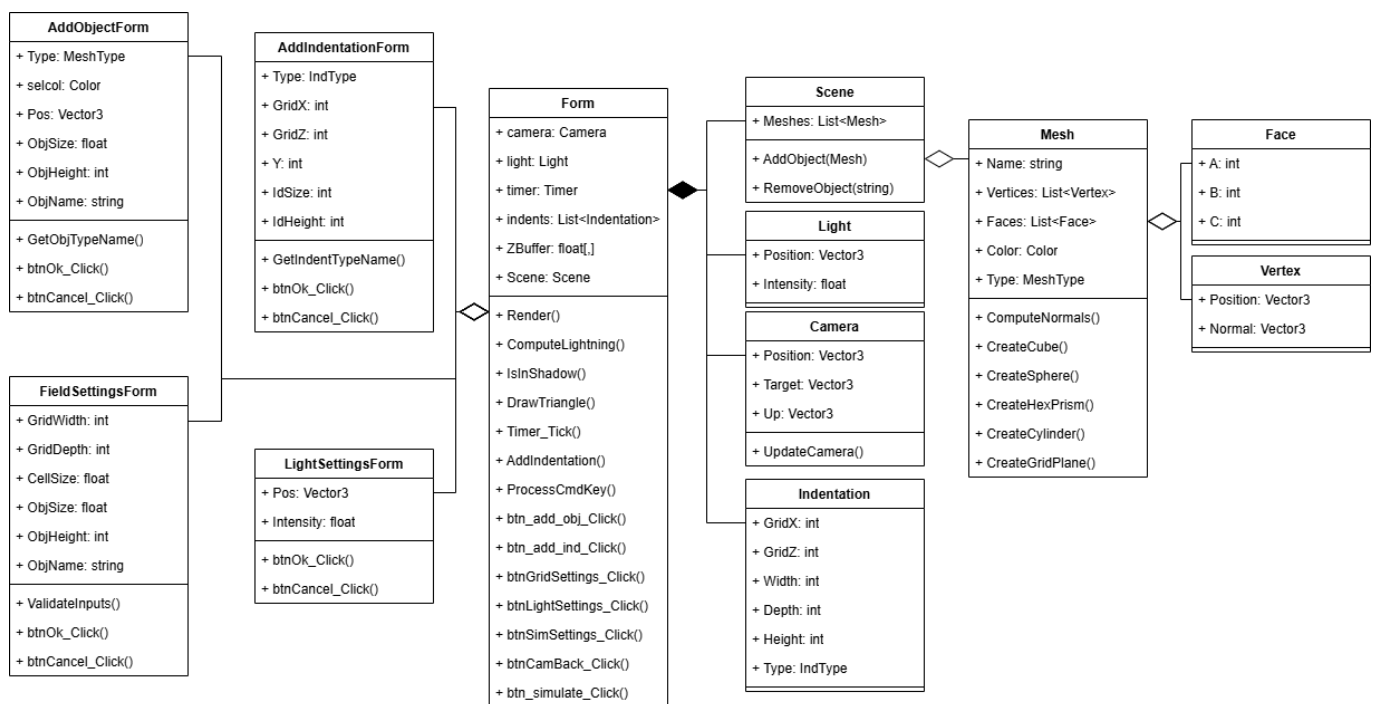


Рисунок 3.1 – Диаграмма классов программы

### 3.3 Схемы алгоритмов

Схемы алгоритмов представлены на рисунках 3.2– 3.5.

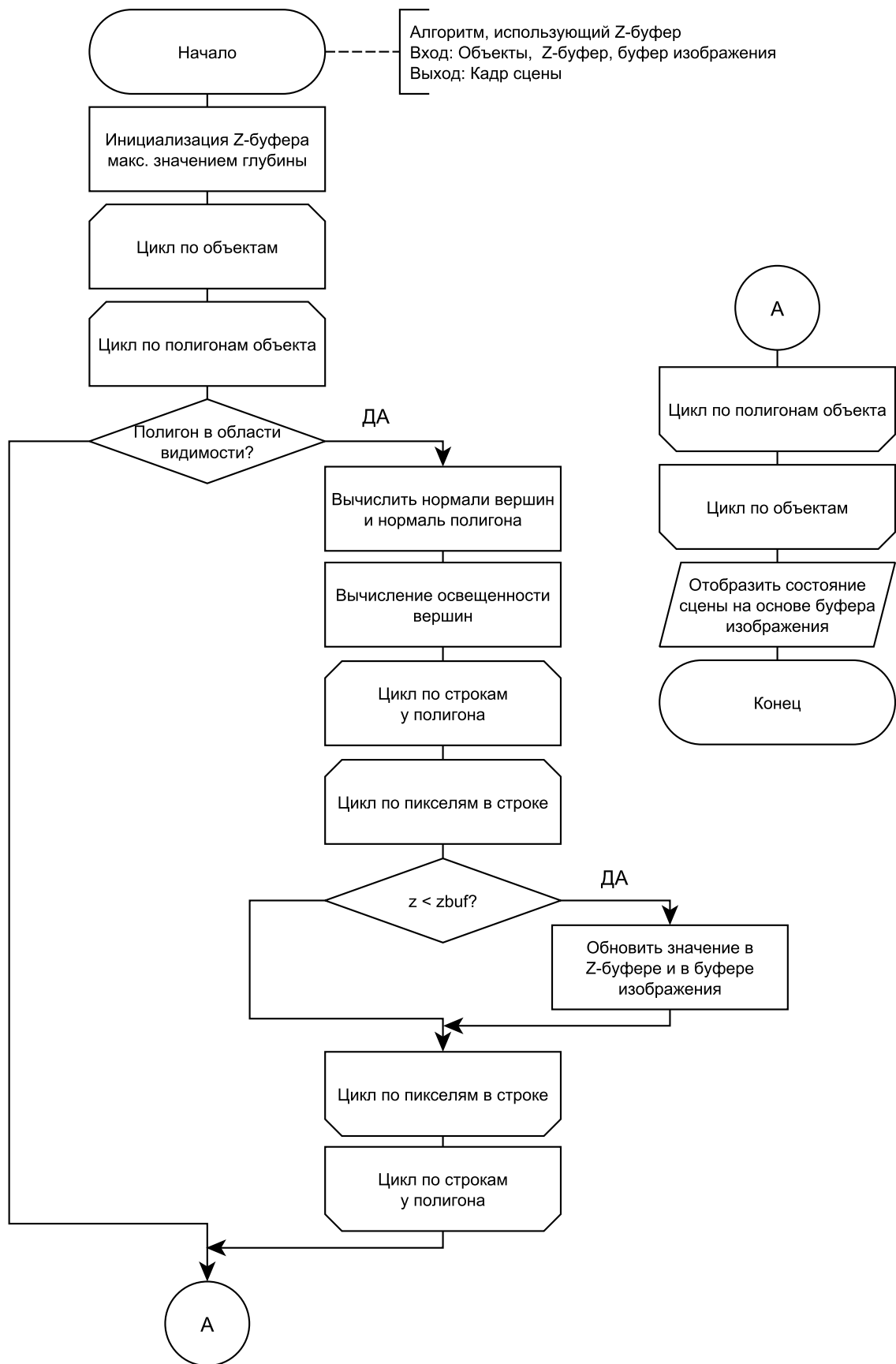


Рисунок 3.2 – Схема алгоритма, использующего Z-буфер

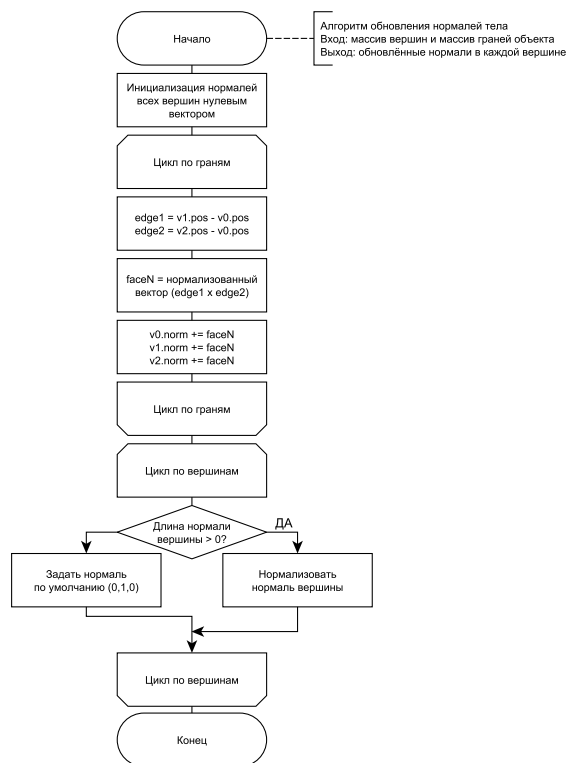


Рисунок 3.3 – Схема алгоритма обновления нормалей объекта

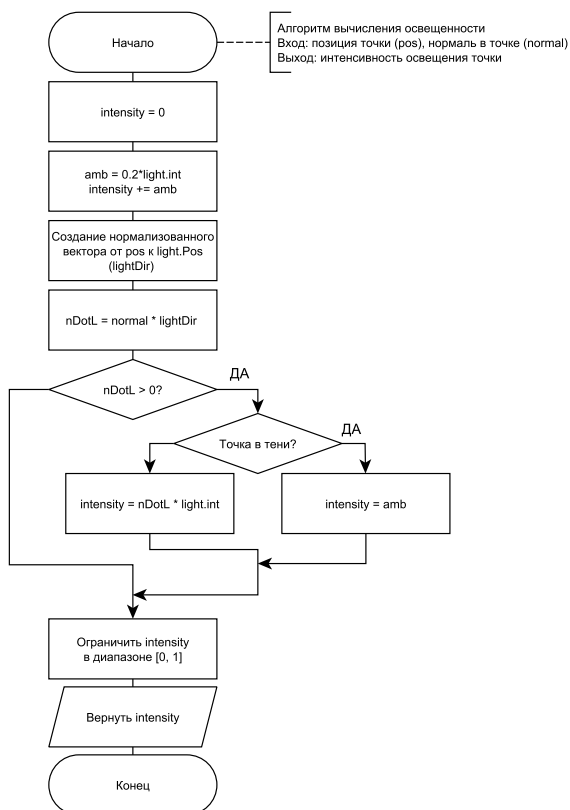


Рисунок 3.4 – Схема алгоритма вычисления освещенности

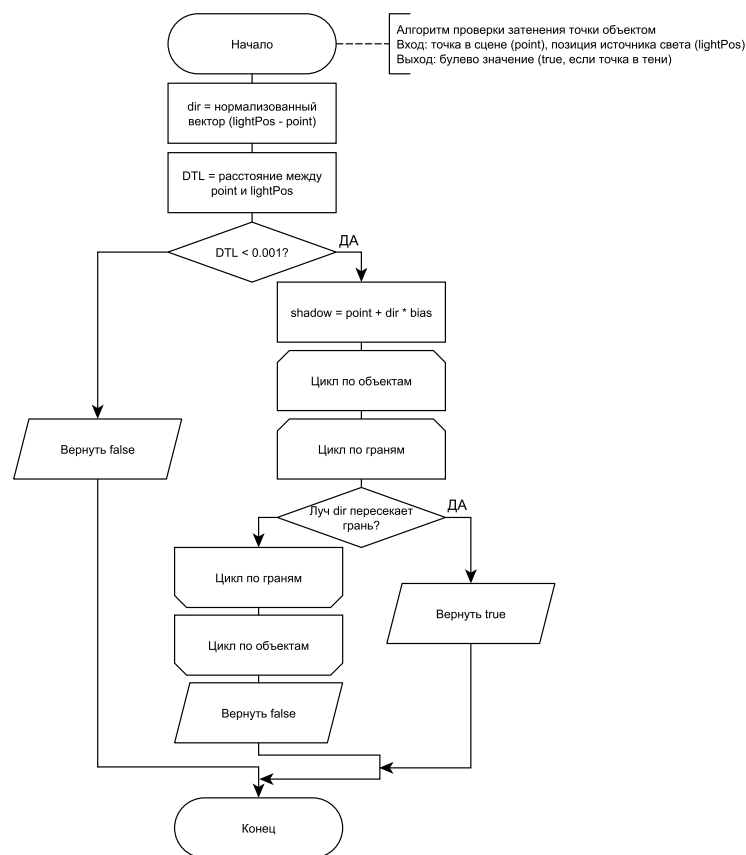


Рисунок 3.5 – Схема алгоритма проверки затенения точки объектом

## 3.4 Интерфейс программного обеспечения

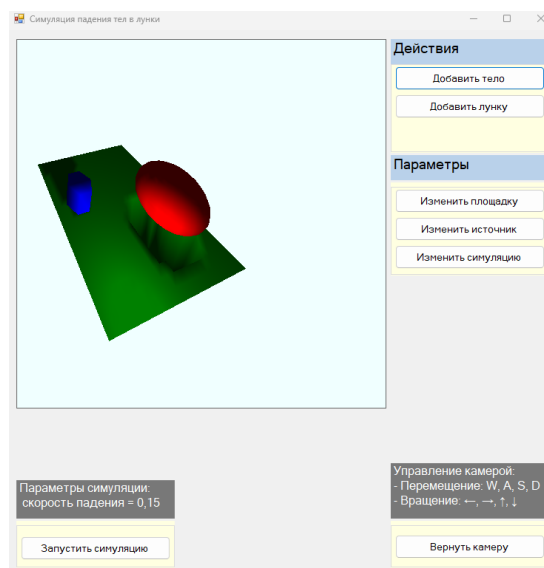
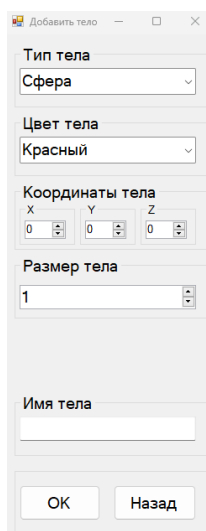


Рисунок 3.6 – Графический интерфейс программы

Главное окно программы представлено на рисунке 3.6. При запуске программы в левой части определен компонент сцены, в правой части определены

компоненты, отвечающие за изменение тел, лунок и параметров площадки, источника света и симуляции. В нижней части описаны правила пользования камерой, а также определена кнопка запуска симуляции с указанной скоростью падения.

Для создания тела пользователю необходимо в главном меню нажать кнопку «Добавить тело» и в появившемся окне указать все нужные параметры (рисунок 3.7).



Добавить тело

Тип тела  
Сфера

Цвет тела  
Красный

Координаты тела  
X Y Z  
0 0 0

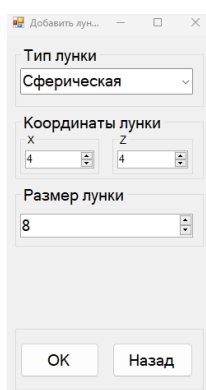
Размер тела  
1

Имя тела

OK Назад

Рисунок 3.7 – Параметры тела

Для создания лунки пользователю необходимо в главном меню нажать кнопку «Добавить лунку» и в появившемся окне указать все нужные параметры (рисунок 3.8).



Добавить лунку...

Тип лунки  
Сферическая

Координаты лунки  
X Z  
4 4

Размер лунки  
8

OK Назад

Рисунок 3.8 – Параметры лунки

Для изменения площадки пользователю необходимо в главном меню нажать кнопку «Изменить площадку» и в появившемся окне указать размер клетки и количество клеток в длину и в ширину (рисунок 3.9).

Для изменения параметров источника света пользователю необходимо в

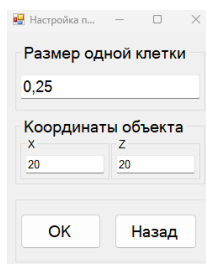


Рисунок 3.9 – Параметры площадки

главном меню нажать кнопку «Изменить источник» и в появившемся окне указать положение источника и его интенсивность (рисунок 3.10).

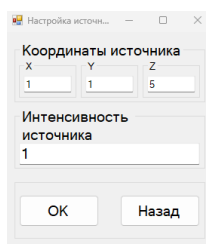


Рисунок 3.10 – Параметры источника света

Для изменения скорости падения объектов необходимо в главном меню нажать кнопку «Изменить симуляцию» и в появившемся окне указать скорость падения объектов (рисунок 3.11).

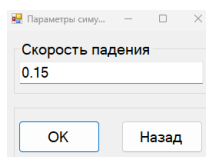


Рисунок 3.11 – Параметры симуляции

## ВЫВОД

В данном разделе были выбраны средства реализации, описаны структуры классов программы, описаны модули, а также рассмотрен интерфейс программы.

## 4 Исследовательская часть

В данном разделе приведены технические характеристики устройства, на котором проводилось измерение времени работы программного обеспечения, а также результаты замеров времени.

### 4.1 Технические характеристики

Характеристики используемого оборудования:

- Операционная система — Windows 11 Home [9].
- Память — 16 Гб.
- Процессор — Intel(R) Core(TM) i5-10300H CPU @ 2.50ГГц [10].
- Количество ядер — 4 физических и 8 логических ядер.

### 4.2 Замеры времени

Для исследования зависимости времени отрисовки сцены от числа тел/лунок на сцене использовались одинаковые типы тел/лунок для каждой серии замеров. Время отрисовки замерялось на компьютере с указанными техническими характеристиками с помощью методов встроенного класса *Stopwatch* [11]. Замеры времени проводились при добавлении различного количества тел (от 1 до 96) и лунок (от 1 до 19) и усреднялись для каждого набора экспериментов. Каждое значение получено путем взятия среднего из 5 измерений. Зависимости времени добавления тел и лунок от их количества представлены на рисунках 4.1 — 4.2.

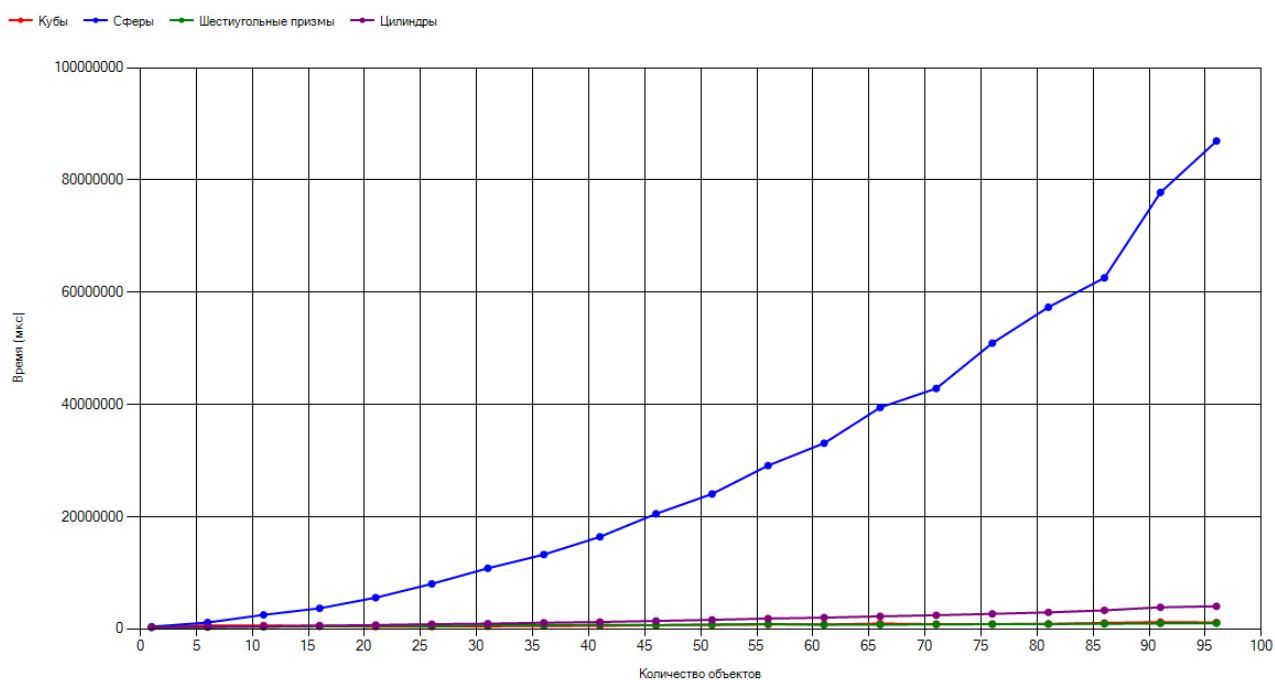


Рисунок 4.1 – График зависимости времени отрисовки от количества тел

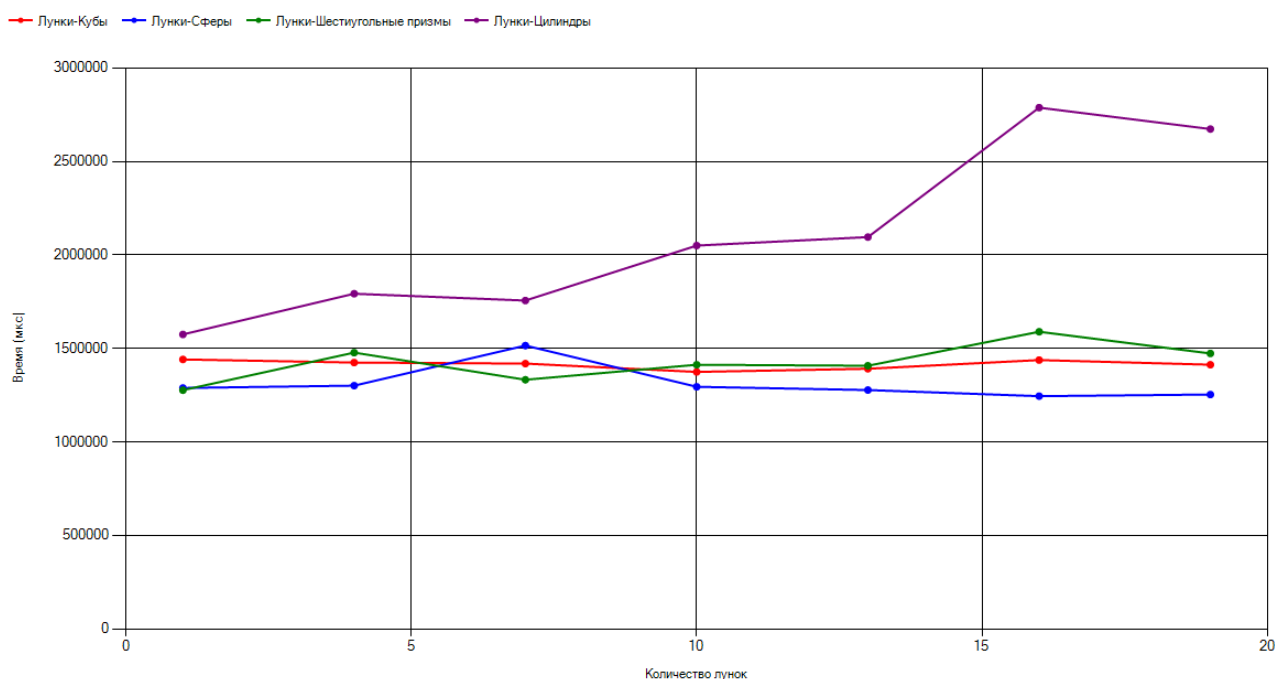


Рисунок 4.2 – График зависимости времени отрисовки от количества лунок

В результате исследования было получено, что при увеличении количества тел, отрисовка каждой из них занимает больше времени. Наиболее значительное увеличение времени наблюдается при добавлении сфер, за которыми следуют цилиндры, шестигранные призмы и кубы. Это обусловлено большим количеством



полигонов для представления каждой сферы и объемом вычислений, необходимых для их добавления и отображения. Зависимость времени отрисовки сцены от количества тел квадратична, так как алгоритм освещения с использованием алгоритма проверки затенения точки объектом имеет квадратичную сложность (см. схемы алгоритмов 3.2– 3.5).

Также было выявлено, что значительное увеличение времени наблюдается при добавлении цилиндрических и сферических лунок, что обусловлено более сложной геометрией этих форм и необходимостью выполнения дополнительных вычислений для их корректного отображения. Лунки для кубов и шестигранных призм демонстрируют стабильное увеличение времени при добавлении, что свидетельствует о меньшей сложности их обработки по сравнению с цилиндрическими или сферическими.

## **ВЫВОД**

В данном разделе приведены результаты работы программного обеспечения и результаты исследования. Результаты исследования совпали с ожидаемыми, так как время отрисовки сцены прямо пропорционально количеству объектов и лунок на сцене и сложности математических расчетов.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы поставленная цель была достигнута: было разработано программное обеспечение для моделирования клетчатой площадки с лунками, соответствующими трехмерным объектам, с возможностью генерации самих объектов и их падения на площадку.

Для достижения были решены следующие задачи:

- описан список доступных к размещению на сцене моделей и формализованы эти модели;
- проведен анализ существующих алгоритмов компьютерной графики для визуализации сцены и выбраны наиболее подходящие;
- выбраны среда реализации программного обеспечения;
- разработано программное обеспечение и реализованы выбранные алгоритмы визуализации;
- проведены замеры временных характеристик разработанного программного обеспечения.

Реализованную программу можно усовершенствовать за счет:

- распараллеливания алгоритмов для ускорения работы;
- использования графической библиотеки OpenGL для повышения производительности графической визуализации путем задействования ресурсов графического процессора;

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Порев В. Н. Компьютерная графика //СПб.: БХВ-Петербург. – 2002. – Т. 432. – С. 3.
2. Роджерс Д. Алгоритмические основы машинной графики. – Рипол Классик, 1989г.
3. Новиков И. Е. Сравнение двух алгоритмов генерации мягких теней //Программа и тез. докл. IX Всерос. конф. молодых ученых по мат. моделированию и информ. технологиям. Кемерово. – 2008г. – С. 28-30.
4. Сафина Д. Н. Исследование методов закрашивания Гуро и Фонга. – 2021г.
5. Чернявская А. Э. Простые модели освещения 3D-объектов. Особенности цифрового моделирования света //Современные вопросы науки и практики 3. – 2021г. – С. 44.
6. Бистерфельд О. А. Методология функционального моделирования IDEF0. – 2013г.
7. Hejlsberg A., Wiltamuth S., Golde P. C language specification. – Addison-Wesley Longman Publishing Co., Inc., 2003г.
8. MacDonald M. User Interfaces in C: Windows Forms and Custom Controls. – Apress, 2008г.
9. Windows 11 Home [Электронный ресурс]. URL: <https://www.officepakke.dk/products/windows-11-home> (дата обращения: 04.11.2024).
10. Intel® Core™ i5-10300H Processor [Электронный ресурс]. URL: <https://ark.intel.com/content/www/us/en/ark/products/201839/intel-core-i5-10300h-processor-8m-cache-up-to-4-50-ghz.html> (дата обращения: 04.11.2024).
11. Stopwatch Класс (System.Diagnostics) [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.diagnostics.stopwatch?view=net-8.0> (дата обращения: 06.11.2024).

## **ПРИЛОЖЕНИЕ А**

### **Презентация к курсовой работе**

Презентация содержит 14 слайдов.

## **ПРИЛОЖЕНИЕ Б**

### **Реализации алгоритмов**