



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
КАФЕДРА

Информатика и системы управления

Программное обеспечение ЭВМ и информационные технологии

КУРСОВАЯ РАБОТА

НА ТЕМУ:

Построение трёхмерного изображения объёмных каркасных и эллипсоидных объектов с использованием ASCII-графики

Студент

ИУ7-52Б

(группа)

(подпись, дата)

Поляков А.И.

(И.О. Фамилия)

Руководитель курсовой
работы

(подпись, дата)

Кострицкий А. С.

(И.О. Фамилия)

2024г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Аналитическая часть	6
1.1 Описание модели трёхмерного объекта на сцене	6
1.2 Описание способа задания трёхмерного объекта на сцене	6
1.3 Формализованная постановка задачи	6
1.4 ASCII графика	7
1.5 Алгоритм выбора символов для представления формы	8
1.6 Алгоритм выбора символов для представления яркости	9
1.7 Выбор алгоритма закраски	9
1.7.1 Простая закраска	10
1.7.2 Закраска методом Гуро	10
1.7.3 Закраска методом Фонга	10
1.7.4 Сравнение алгоритмов	11
1.8 Выбор алгоритма удаления невидимых ребер и поверхностей	11
1.8.1 Алгоритм Варнока	12
1.8.2 Алгоритм, использующий Z-буфер	12
1.8.3 Алгоритм трассировки лучей	13
1.8.4 Сравнение алгоритмов	13
1.9 Выводы из аналитической части	13
2 Конструкторская часть	14
2.1 Разработка функциональной модели (IDEF0) ПО	14
2.2 Получение информации о шрифте	15
2.3 Разработка алгоритма выбора символов для представления формы	16
2.4 Разработка алгоритма выбора символов для представления яркости	17
2.5 Разработка алгоритма закраски методом Гуро	18
2.6 Разработка алгоритма удаления невидимых ребер и поверхностей, использующего Z-буфер	19
2.7 Разработка типов и структур данных	19

2.8 Выводы из конструкторской части	20
3 Технологическая часть	21
3.1 Выбор средств реализации	21
3.2 Формат входных данных	21
3.3 Интерфейс пользователя	21
3.4 Модульное тестирование	22
3.5 Функциональное тестирование	22
3.6 Выводы из технологической части	25
4 Исследовательская часть	26
4.1 Выбор входных параметров	26
4.2 Проведение исследования	26
4.3 Выводы из исследовательской части	28
ЗАКЛЮЧЕНИЕ	29
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	30
Приложение А	32

ВВЕДЕНИЕ

В настоящее время ASCII-графика используется в текстовых интерфейсах, таких как командные оболочки и системы без графического интерфейса, в качестве средства визуального представления данных [10], а также в художественных формах и интернет-коммуникации, где текстовые символы служат основным средством передачи визуальной информации. [9]

Целью данной работы является разработка программного обеспечения для построения трёхмерного изображения объёмных объектов и эллипсоидов с использованием заданного пользователем набора ASCII-символов. Пользователь также определяет количество, яркость и расположение белых точечных источников света и набор доступных символов.

Для достижения поставленной цели требуется решить следующие задачи:

1. Проанализировать существующие методы построения трехмерных объектов и преобразования изображений в ASCII-графику.
2. Спроектировать программное обеспечение для построения трёхмерного изображения.
3. Выбрать средства реализации и реализовать спроектированное программное обеспечение.
4. Исследовать зависимость качества изображения от выбранного набора символов.

Целевой областью применения разрабатываемого в данной работе программного обеспечения будут являться командные оболочки и системы без графического интерфейса, такие как серверы. Данные системы обладают ограниченными вычислительными возможностями.

1 Аналитическая часть

В данном разделе описана модель трехмерного объекта, проведена формализованная постановка задачи, рассмотрена предметная область ASCII-графики и существующие алгоритмы визуализации ASCII-графики. Также будут рассмотрены методы закраски и удаления невидимых рёбер и поверхностей, и произведён их сравнительный анализ, по итогам которого будут выбраны наиболее подходящие для решения поставленной задачи алгоритмы.

Алгоритмы визуализации теней в данной работе рассматриваться не будут, чтобы снизить вычислительную нагрузку в связи с ограниченными графическими и вычислительными возможностями целевых систем.

1.1 Описание модели трехмерного объекта на сцене

Модели задаются следующими способами [5]:

- каркасная модель. В этой модели задается информация о вершинах и рёбрах объектов;
- поверхностная модель. Поверхность может описываться аналитически, либо может задаваться другим способом;
- объёмная модель. Эта форма модели отличается от поверхностной тем, что к информации о поверхности добавляется информация о том, с какой стороны расположен материал.

В данной работе будут визуализироваться трехмерные объемные объекты, в связи с чем была выбрана объёмная модель.

1.2 Описание способа задания трёхмерного объекта на сцене

Существует несколько способов задания трёхмерного объекта [5]:

- аналитический способ. Этот способ характеризуется описанием объекта в неявной форме, то есть для получения визуального представления нужно вычислять значения некоторой функции в различных точках пространства;
- полигональная сетка. Это совокупность вершин, рёбер и граней, которые определяют форму многогранного объекта. Гранями обычно являются треугольники, так как любой полигон можно представить в виде треугольника. [5]

Для задания объектов будет использоваться полигональная сетка, так как в данной работе будут визуализироваться различные объемные объекты, включая эллипсоиды, а любая сложная поверхность или форма может быть аппроксимирована треугольниками, что делает полигональную сетку универсальным инструментом для описания объектов.

1.3 Формализованная постановка задачи

Входные данные:

- описания объектов d_1, \dots, d_k : объект представляется в виде полигональной сетки. Для описания требуется указать координаты вершин, нормали к каждой вершине, направленные в противоположную от поверхности сторону, и грани;
- информация об источниках света ls_1, \dots, ls_m : белый источник света представляется в виде точечного объекта, заданного его координатами в пространстве и интенсивностью излучения;
- информация о наблюдателе Z : наблюдатель характеризуется его координатой на оси Z . Его взгляд всегда сонаправлен с осью oz , при этом ось oy направлена вверх, ось ox направлена влево;
- информация о шрифте F : символы ASCII, описанные векторной графикой;
- подмножество символов $CH = \{ch_1, \dots, ch_n\}$: символы ASCII, доступные для использования

Значения коэффициента фонового освещения, диффузного освещения и зеркального освещения для объектов будут одинаковы и заданы на этапе сборки.

Возвращаемые данные:

- кадр сцены, выведенный в терминал.

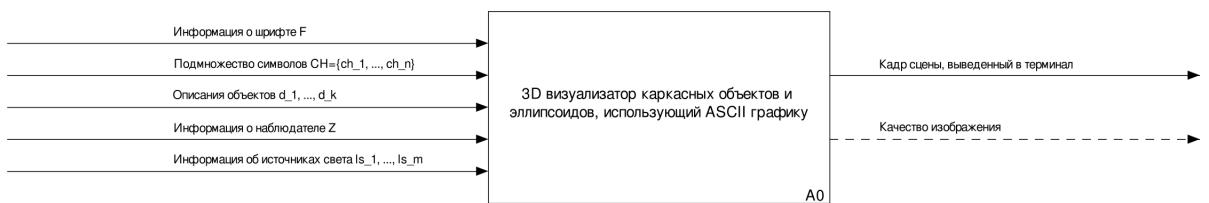


Рисунок 1.1 — Формализованная постановка задачи в нотации IDEF0

1.4 ASCII графика

ASCII графика – это форма визуального представления изображений и узоров, использующая знаки и символы из ASCII (American Standard Code for Information Interchange), американской стандартной кодировочной таблицы для печатных символов и некоторых специальных кодов. [12]

ASCII графика подразделяется на два типа, различающиеся основой: [1]

- яркость пикселей (пример на рис. 1.2);
- форма объектов (пример на рис. 1.3).

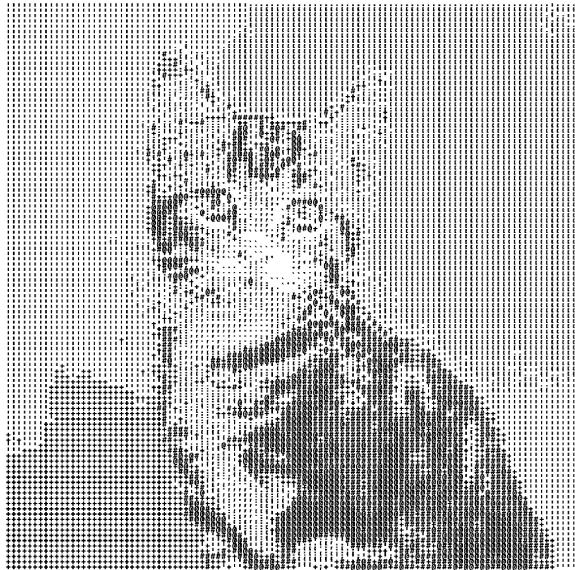


Рисунок 1.2

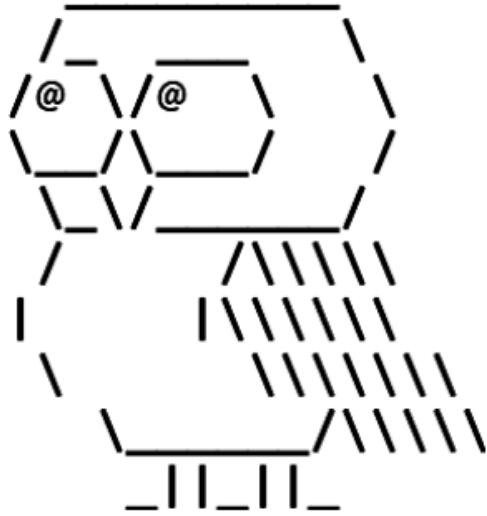


Рисунок 1.3

Графика, основанная на яркости пикселей, использует различные символы для представления различных уровней затенения, создавая иллюзию глубины и контраста, в то время как графика, основанная на форме, в первую очередь направлена на формирование узнаваемых контуров с использованием символов, с меньшим акцентом на тени или тон.

В данной работе планируется визуализировать как поверхности объектов, так и их грани, в связи с чем будут рассмотрены алгоритмы выбора символов ASCII для представления яркости пикселей для отрисовки поверхностей объемных объектов и алгоритмы выбора символов ASCII для представления формы для отрисовки рёбер каркасных объектов.

1.5 Алгоритм выбора символов для представления формы

Алгоритм выбора символов для представления формы будет использован для отрисовки ребер каркасных и визуальных границ эллипсоидных объектов.

Метрическая оценка сходства формы символа ch_i и ячейки изображения c_i [3]:

- 1) для N выбранных точек сравниваемого символа ch_i и ячейки изображения c_i строятся полярные диаграммы;
- 2) по полярным диаграммам строятся гистограммы, строки и столбцы которых соответствуют угловым (Phi) и радиальным (R) измерениям полярной диаграммы соответственно;
- 3) N полученных гистограмм представляют собой вектор $H = (h_1 \dots h_N)$, описывающий конкретный символ или ячейку. Коэффициент D , используемый для оценки их сходства высчитывается по формуле (1.2). Чем меньше коэффициент, тем больше сходство.

$$D(ch_i, c_i) = \sum_{i \in N} \|H_{ch_i, i} - H_{c_i, i}\|, \quad (1.1)$$

где ch_i - сравниваемый символ, c_i - сравниваемая ячейка, $H_{ch_i, i}$ - i -я гистограмма вектора

сравнения символа, $H_{c_i,i}$ - i-я гистограмма вектора сравнения ячейки

Пример построения гистограммы для одной из точек символа А показан на рис. (1.4)

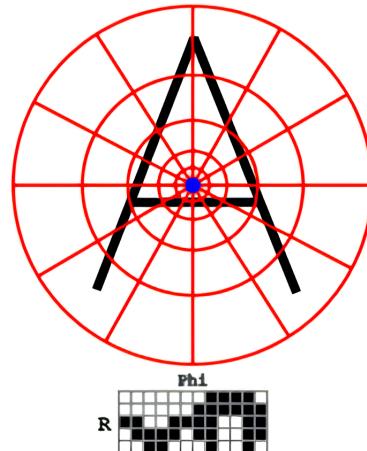


Рисунок 1.4 — Пример полярной диаграммы с гистограммой для одной из точек символа А

Для избежания многократных вычислений векторов сравнения символов, перед визуализацией объектов для всех доступных CH соответствующие им вектора будут просчитаны и сохранены в карту символов по форме MS .

1.6 Алгоритм выбора символов для представления яркости

Алгоритм выбора символов для представления яркости будет использован для отрисовки поверхностей объемных объектов.

Составление карты доступных символов по яркости MB :

- 1) отсортировать символы CH по количеству пикселей, необходимых для их отображения;
- 2) найти максимальное кол-во пикселей в символе N ;
- 3) рассчитать коэффициент яркости k для каждого символа, где k равен:

$$k = n_i/N, \text{ где } N \text{ — максимальное кол-во пикселей символа, } n_i \text{ — кол-во пикселей в } i\text{-м символе} \quad (1.2)$$

Метрическая оценка сходства яркости символа ch_i и ячейки изображения c_i [2]:

- вычисляется среднее арифметическое avg яркостей пикселей ячейки c_i ;
- из карты MB выбирается символ, для которого разница между его коэффициентом яркости k и яркостью ячейки avg наименьшая.

1.7 Выбор алгоритма закраски

Для отображения поверхностей объектов необходимо использовать алгоритмы закраски. При выборе алгоритма необходимо учитывать особенности поставленной задачи. Он должен

позволять отрисовывать поверхности объектов, освещённых несколькими точечными источниками света, произвольно расположеными в пространстве. Также важно учесть ограниченные вычислительные возможности целевых систем. Будут рассмотрены 3 основных [4] алгоритма закраски:

- 1) Простая закраска;
- 2) Закраска методом Гуро;
- 3) Закраска методом Фонга.

1.7.1 Простая закраска

Вся грань закрашивается одним уровнем интенсивности, который высчитывается по закону Ламберта [5]. При данной закраске все грани будут закрашены однотонно, не учитывая, под каким углом падает свет (см. рис. 1.5). В связи с этим алгоритм не подходит для поставленной задачи.

1.7.2 Закраска методом Гуро

Суть алгоритма заключается в интерполяции интенсивности освещения для точек внутри полигона [5].

- 1) интенсивность в вершинах полигона рассчитывается по закону Ламберта, учитывая угол падения света на поверхность;
- 2) для каждой точки P внутри полигона определяются барицентрические координаты относительно вершин полигона A, B, C ;
- 3) используя барицентрические координаты $\lambda_1, \lambda_2, \lambda_3$, интенсивность в точке P вычисляется по формуле:

$$I_P = \lambda_1 \cdot I_A + \lambda_2 \cdot I_B + \lambda_3 \cdot I_C, \quad (1.3)$$

где:

- $\lambda_1, \lambda_2, \lambda_3$ — барицентрические координаты точки P ,
- I_A, I_B, I_C — интенсивности в вершинах A, B, C ;

Закраска методом Гуро выполняет закраску полигонов, основываясь на интерполяции интенсивности освещения, что позволяет учитывать угол падения света на поверхность (см. рис. 1.5). Данный алгоритм подходит для поставленной задачи.

1.7.3 Закраска методом Фонга

Алгоритм работает похожим на алгоритм Гуро образом, однако ключевым отличием является то, что интерполируются не интенсивности в вершинах, а нормали [5]. Таким образом, закон Ламберта в данном алгоритме применяется в каждой точке, а не только в вершинах, что делает этот алгоритм более трудоёмким, однако позволяет учитывать кривизну поверхности. С учётом ограниченных вычислительных возможностей целевых систем, данный алгоритм не

является подходящим для поставленной задачи.

1.7.4 Сравнение алгоритмов

Визуальные различия алгоритмов закраски представлены на рисунке 1.5. [6]

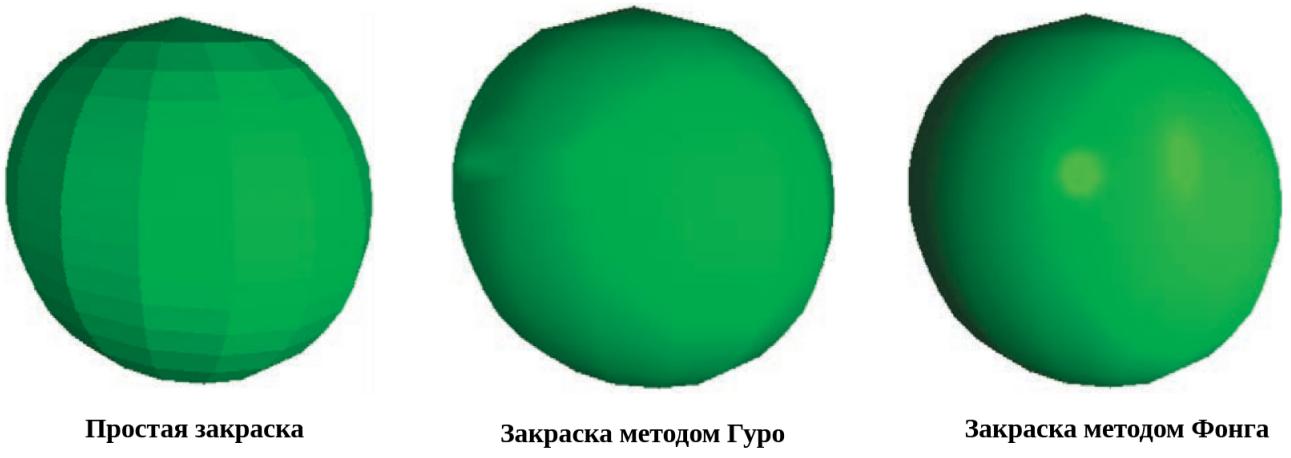


Рисунок 1.5 — Сравнение алгоритмов закраски

В таблице 1.1 приведено сравнение алгоритмов закраски:

Критерий	Простая	Метод Гуро	Метод Фонга
Количество расчетов освещения	1 на полигон	1 на вершину	1 на пиксель
Количество интерполяций	0	1 (интенсивность)	1 (нормали)
Количество операций с нормалями	1 на полигон	1 на вершину	1 на пиксель
Вычислительная сложность	$O(1)$	$O(n_{\text{полигонов}})$	$O(n_{\text{пикселей}})$

Таблица 1.1 — Сравнение методов закраски: Простая, Гуро, Фонга

Вывод:

Была выбрана закраска Гуро, так как он позволяет учитывать угол падения света на полигон, что делает изображение более реалистичным [5], а также нам важна производительность, так как целевые системы обладают ограниченными вычислительными ресурсами.

1.8 Выбор алгоритма удаления невидимых ребер и поверхностей

Задача удаления невидимых рёбер в компьютерной графике заключается в том, чтобы убрать из отображения те элементы, которые не видны наблюдателю из-за их расположения или перекрытия другими объектами.

Алгоритмы удаления невидимых линий и поверхностей делятся на работающие в:

- объектном пространстве (мировая система координат, высокая точность);
- пространстве изображений (система координат связана с дисплеем, точность ограничена разрешающей способностью дисплея).

При выборе алгоритма удаления невидимых линий необходимо учитывать особенности поставленной задачи. Важно учесть ограниченные вычислительные возможности целевых систем. Будут рассмотрены 3 основных [8] алгоритма удаления невидимых ребер и поверхностей:

- 1) Алгоритм Варнока;
- 2) Алгоритм, использующий Z -буфер;
- 3) Алгоритм прямой трассировки лучей.

1.8.1 Алгоритм Варнока

Алгоритм работает в пространстве изображений и выполняет следующие шаги [5]:

- 1) проверяется, является ли окно пустым или в окне содержатся только простые геометрические формы, такие как отрезки;
- 2) если окно не удовлетворяет указанным условиям: разделить окно на подокна и повторить анализ содержимого для каждого подокна;
- 3) повторять процесс деления до выполнения одного из условий: содержимое подокна становится достаточно простым для отображения или размер окна уменьшается до одного пикселя.

1.8.2 Алгоритм, использующий Z -буфер

Данный алгоритм работает в пространстве изображения [5] и выполняет следующие шаги:

- 1) инициализируются два буфера:
 - буфер кадра I , в котором хранятся атрибуты каждого пикселя в пространстве изображения. Размер этого буфера равен размеру изображения, то есть $width \times height$, где $width$ и $height$ — размеры изображения;
 - Z -буфер, куда помещается информация о координате z для каждого пикселя. Его размер также равен размеру изображения $width \times height$, и каждый элемент хранит значение глубины для соответствующего пикселя. Изначально заполняется максимально возможными значениями глубины.
- 2) для каждого пикселя p :
 - сравнивается глубина p с глубиной уже существующего в z -буфере.
 - если новый пиксель находится ближе к наблюдателю:
 - его глубина заносится в z -буфер;
 - значение интенсивности обновляется в I .

Основным недостатком алгоритма является значительный объем требуемой памяти для хранения двух буферов, каждый размером $width \times height$. [5]

1.8.3 Алгоритм трассировки лучей

Методы прямой и обратной трассировки лучей используются для моделирования траектории лучей от источника света до камеры с учётом их взаимодействия с объектами на пути.

— **Прямая трассировка лучей:** Отслеживает лучи от всех источников света ко всем точкам сцены, включая те, которые не попадают в камеру, что делает метод неэффективным.

— **Обратная трассировка лучей:** Более эффективный подход, отслеживающий лучи от камеры к объектам сцены и учитывающий их взаимодействие. Этот метод позволяет рассчитывать тени, отражения и преломления. Однако у него есть недостатки, такие как игнорирование вторичного освещения, высокая вычислительная сложность, резкие границы цветовых переходов и дискретность первичных лучей [5].

1.8.4 Сравнение алгоритмов

В таблице 1.2 приведено сравнение алгоритмов удаления невидимых рёбер и поверхностей:

Критерий	Варнок	Z-буфер	Трассировка лучей
Количество операций с пикселями	Зависит от сложности сцены и деления окна	1 на пиксель (сравнение и запись)	1 луч на пиксель (для обратной трассировки)
Количество буферов (занимаемая память)	1 (фреймбуфер)	2 (фреймбуфер и Z-буфер)	Нет необходимости в дополнительных буферах
Вычислительная сложность	$O(n_{\text{подокон}})$	$O(n_{\text{пикселей}})$	$O(n_{\text{лучей}})$

Таблица 1.2 — Сравнение алгоритмов: Варнока, Z-буфера, трассировки лучей

Вывод:

Был выбран алгоритм удаления невидимых ребер и поверхностей, использующий Z-буфер. Он является производительным, что делает его хорошим выбором для целевых систем.

1.9 Выводы из аналитической части

Была описана модель трехмерного объекта, проведена формализованная постановка задачи, рассмотрена предметная область ASCII-графики и существующие алгоритмы визуализации ASCII-графики. В качестве алгоритма закраски была выбрана закраска Гуро, а в качестве алгоритма удаления невидимых рёбер и поверхностей был выбран алгоритм, использующий Z-буфер.

2 Конструкторская часть

В данном разделе представлена разработка функциональной модели (IDEF0) программного обеспечения, а также разработка алгоритмов выбора символов для представления формы, выбора символов для представления яркости, закраски методом Гуро, алгоритма удаления невидимых рёбер и поверхностей, использующего Z-буфер и используемых типов и структур данных.

2.1 Разработка функциональной модели (IDEF0) ПО

На рисунках 2.1-2.3 представлена декомпозиция разрабатываемого ПО в нотации IDEF0.

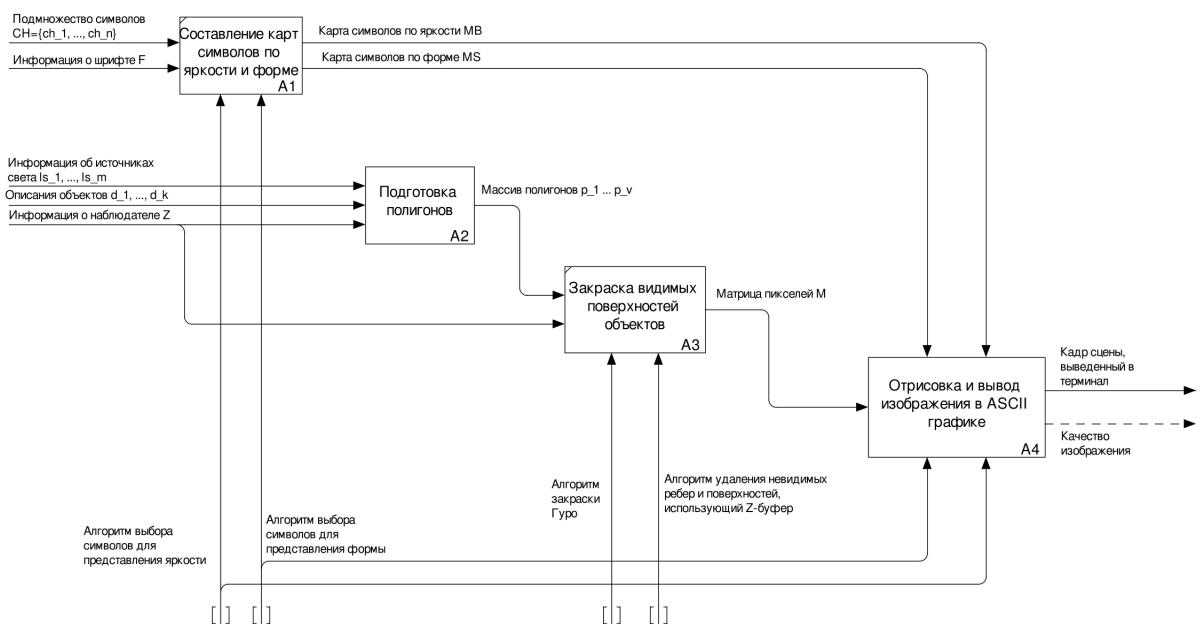


Рисунок 2.1 — 3D визуализатор каркасных объектов и эллипсоидов, использующий ASCII графику

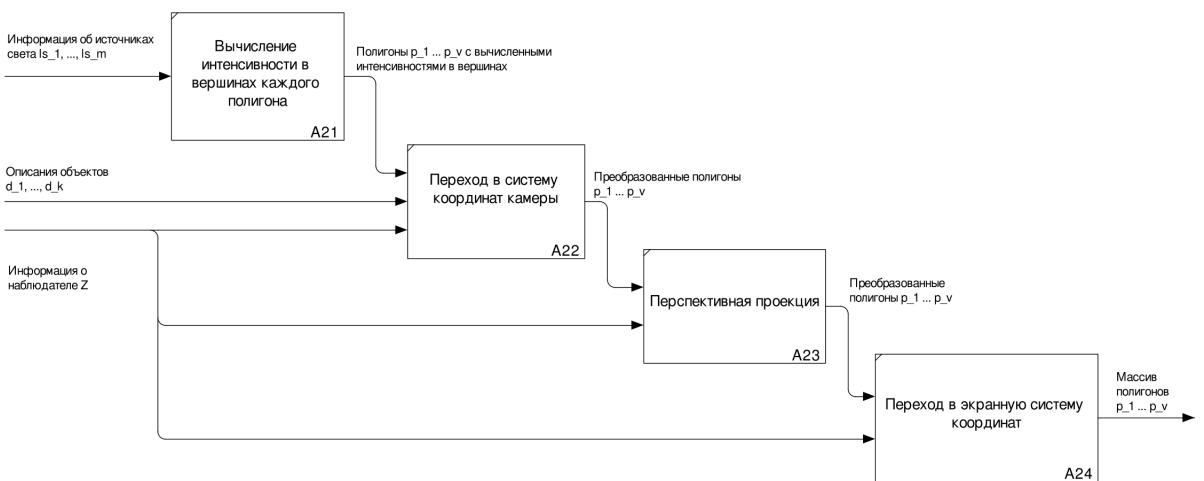


Рисунок 2.2 — Подготовка полигонов

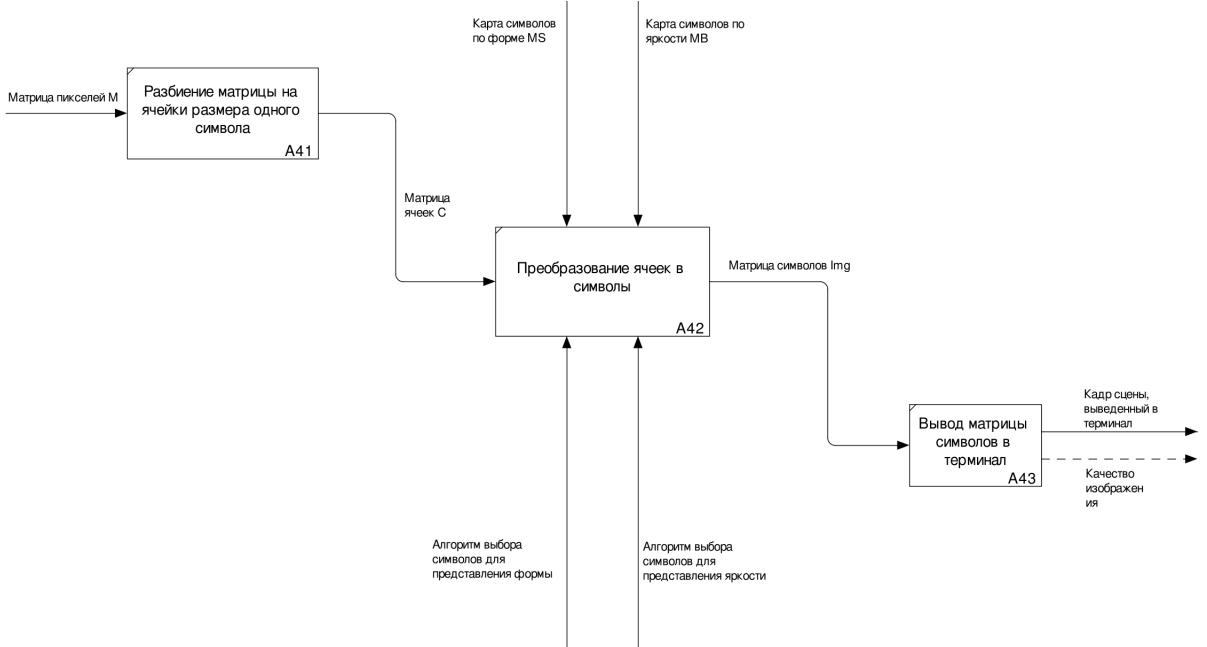


Рисунок 2.3 — Отрисовка и вывод изображения в ASCII графике

2.2 Получение информации о шрифте

Для алгоритмов выбора символов для представления формы и яркости необходима информация о используемом в терминале шрифте в виде матриц пикселей каждого символа ASCII. Для этого необходим алгоритм обработки файлов шрифта. В файлах шрифта хранится векторное представление символов, которое необходимо преобразовать в растровое. [14]

Алгоритм получения информации о шрифте:

Входные данные:

- описание шрифта F :
 - векторные представления символов ASCII $V = v_1, \dots, v_m$;
 - размер символа в терминале x на y ;
- множество символов $CH = \{ch_1, \dots, ch_n\}$.

Возвращаемые данные: карта MF матриц пикселей для каждого символа множества CH

Алгоритм:

- 1) для каждого символа ch_i из множества CH :
 - получение векторного представления v из V для символа ch_i ;
 - создание пустой матрицы размера x на y ;
 - отрисовка символа в рамках матрицы по v ;
 - запись полученной матрицы в карту MF в соответствие символу ch_i .
- 2) возврат карты MF .

2.3 Разработка алгоритма выбора символов для представления формы

Построение описательного вектора

Входные данные:

- матрица пикселей M ;
- параметры аппроксимации ctx :
 - количество разбиений ячейки по вертикали $vertParts$;
 - количество разбиений ячейки по горизонтали $horParts$;
 - сектора полярной диаграммы $phiParts$;
 - количество разбиений сектора $rParts$;
 - максимальная длина сектора $rMax$.

Возвращаемые данные: описательный вектор v

Алгоритм:

- 1) v = пустой описательный вектор;
- 2) n = ширина матрицы M ;
- 3) m = высота матрицы M ;
- 4) $nStep = n / horParts$;
- 5) $mStep = m / vertParts$;
- 6) $rStep = rMax / rParts$;
- 7) $phiStep = 2 * \pi / phiParts$;
- 8) для каждого участка матрицы (по шагам $nStep$ и $mStep$):
 - $currHyst$ = матрица размера $rParts \times phiParts$;
 - для каждого сектора полярной диаграммы (i по $phiStep$ и j по $rStep$):
 - если хотя бы один пиксель из MF находится в секторе:
 - $currHyst[i][j] = 1$;
 - иначе;
 - $currHyst[i][j] = 0$.
 - добавление $currHyst$ в $resVector$.
- 9) возврат v .

Построение карты MS символов по форме:

Входные данные:

- карта MF матриц пикселей для каждого символа множества CH ;
- параметры аппроксимации ctx :
 - количество разбиений ячейки по вертикали $vertParts$;
 - количество разбиений ячейки по горизонтали $horParts$;
 - сектора полярной диаграммы $phiParts$;
 - количество разбиений сектора $rParts$;
 - максимальная длина сектора $rMax$.

Возвращаемые данные: карта MS символов по форме

Алгоритм:

1) для каждого символа ch_i карты MF :

- построение описательного вектора v с параметрами ctx ;
- запись полученного вектора v в карту MS в соответствие символу ch_i .

2) возврат MS .

Подбор символа в соответствие ячейке:

Входные данные:

- матрица пикселей ячейки C_i ;
- карта MS символов по форме;
- параметры аппроксимации ctx :
 - количество разбиений ячейки по вертикали $vertParts$;
 - количество разбиений ячейки по горизонтали $horParts$;
 - сектора полярной диаграммы $phiParts$;
 - количество разбиений сектора $rParts$;
 - максимальная длина сектора $rMax$.

Возвращаемые данные: символ $cRes$

Алгоритм:

1) построение описательного вектора v для C_i с параметрами ctx ;

2) $cRes$ - символ;

3) $mindelt = \infty$;

4) для каждого описательного вектора cv_i символа ch_i из карты MS :

- $d = 0$;
- для каждой i -й гистограммы v :
 - $d = d + \|v[i] - cv_i[i]\|$.
- если $d < mindelt$:
 - $mindelt = d$;
 - $cRes = ch_i$.

5) возврат $cRes$.

2.4 Разработка алгоритма выбора символов для представления яркости

Построение карты MB символов по яркости:

Входные данные: карта MF матриц пикселей для каждого символа множества ChS

Возвращаемые данные: карта MB символов по яркости

Алгоритм:

1) $maxCnt = 0$

2) для каждого символа ch_i карты MF :

— cnt = количество пикселей символа ch_i

— если $cnt > maxCnt$

— $maxCnt = cnt$

3) для каждого символа ch_i карты MF :

— cnt = количество пикселей символа ch_i

— запись $cnt/maxCnt$ в карту MB в соответствие символу ch_i

4) возврат MB .

Подбор символа в соответствие ячейке:

Входные данные:

— яркость b ячейки C ;

— карта MB символов по яркости.

Возвращаемые данные: символ $cRes$

Алгоритм:

1) $cRes$ - символ;

2) $mindelt = \infty$;

3) для каждого значения яркости cb_i символа ch_i из карты MB :

— $d = |cb_i - b|$;

— если $d < mindelt$:

— $mindelt = d$;

— $cRes = ch_i$.

4) возврат $cRes$.

2.5 Разработка алгоритма закраски методом Гуро

Входные данные:

— грань f объекта, заданная вершинами $\{v_1, v_2, v_3\}$ и нормалями $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}$ в этих вершинах;;

— массив источников света ls_1, \dots, ls_m , где каждый источник ls_i задан позицией \mathbf{p}_i и интенсивностью I_i .

Возвращаемые данные: закрашенный полигон p .

Алгоритм:

1) инициализация итоговой интенсивности для каждой вершины: $I(v_i) \leftarrow k_a$, где $i = 1, 2, 3$;

2) для каждой вершины v_i с нормалью \mathbf{n}_i :

— для каждого источника света ls_j :

— $\mathbf{l}_j = \frac{\mathbf{p}_j - v_i}{\|\mathbf{p}_j - v_i\|}$;

— если $\mathbf{n}_i \cdot \mathbf{l}_j \leq 0$, перейти к следующему источнику;

— $I_d = k_d \cdot I_j \cdot \max(0, \mathbf{n}_i \cdot \mathbf{l}_j)$;

— $\mathbf{r}_j = 2(\mathbf{n}_i \cdot \mathbf{l}_j)\mathbf{n}_i - \mathbf{l}_j$;

- $I_s = k_s \cdot I_j \cdot \max(0, (\mathbf{r}_j \cdot \mathbf{v}_{obs})^n);$
- $I(v_i) += I_d + I_s.$

3) для каждого пикселя внутри треугольника f :

- $(\alpha, \beta, \gamma) = \text{ComputeBarycentric}(x, y, v_1, v_2, v_3);$
- $I(x, y) = \alpha \cdot I(v_1) + \beta \cdot I(v_2) + \gamma \cdot I(v_3);$
- закрасить пиксель (x, y) с интенсивностью $I(x, y).$

4) возврат $p.$

2.6 Разработка алгоритма удаления невидимых ребер и поверхностей, использующего Z-буфер

Входные данные:

- массив полигонов $p_1, \dots, p_v;$
- высота h и ширина w изображения.

Возвращаемые данные: матрица пикселей изображения $M.$

Алгоритм:

- 1) создание буфера изображения img - матрицы пикселей размера $h \times w;$
- 2) создание Z-буфера $zBuf$ - матрицы вещественных чисел размера $h \times w$, заполненной максимальными возможными значениями $z;$
- 3) для каждого пикселя px_i каждого полигона p_i :
 - если $(0 \leq x < w, 0 \leq y < h)$ И $(zBuf[y][x] > z):$
 - $img[y][x] = \text{пиксель};$
 - $zBuf[y][x] = z.$
- 4) возврат $img.$

2.7 Разработка типов и структур данных

- структура объекта содержит:
 - массив граней модели $Faces;$
 - центр модели $Center$ (три вещественных переменных).
- структура грани содержит:
 - массив координат вершин $Vertices;$
 - массив нормалей к вершинам $Normals.$
- структура пикселя содержит:
 - яркость $brightness$, представленную вещественным числом от 0 до 1;
 - флаг $isPolygon$, показывающий, является ли пиксель частью полигона;
 - флаг $isLine$, показывающий, является ли пиксель частью ребра.
- структура полигона содержит набор пикселей, сопоставленных координатам экрана;
- структура камеры содержит:

- координату на оси Z , задающую ее положение;
 - угол обзора камеры fov ;
 - соотношение сторон камеры $aspect$;
 - координату $zFar$ дальней плоскости обрезания;
 - координату $zNear$ ближней плоскости обрезания.
- структура источника света содержит:
- координаты положения источника света в пространстве $Position$ (три вещественных переменных);
 - интенсивность излучения $Intensity$ (вещественная переменная).

2.8 Выводы из конструкторской части

Было спроектировано программное обеспечение, разработаны алгоритмы выбора символов для представления формы, выбора символов для представления яркости, закраски методом Гуро, удаления невидимых рёбер и поверхностей с использованием Z-буфера и типы и структуры данных.

3 Технологическая часть

В данном разделе обоснован выбор средств реализации ПО, описаны формат входных данных и интерфейс пользователя и проведено функциональное тестирование.

3.1 Выбор средств реализации

В качестве языка разработки был выбран Go [15] в силу следующих причин:

- в стандартной библиотеке языка присутствует поддержка всех структур данных, выбранных по результатам проектирования;
- средствами языка можно реализовать все алгоритмы, выбранные в результате проектирования.

3.2 Формат входных данных

Пользователю необходимо передавать ПО следующие данные:

- описания объектов;
- информация об источниках света;
- информация о наблюдателе;
- информация о шрифте;
- подмножество доступных символов.

Для представления описаний объектов был выбран универсальный формат OBJ [16] в силу того, что он позволяет передавать всю информацию, необходимую ПО для представления объекта:

- список вершин;
- нормали к вершинам;
- список граней.

Информацию об источниках света пользователь указывает, вводя их координаты в пространстве и интенсивность, представленные в виде вещественных чисел.

Информацию о наблюдателе пользователь указывает, вводя его координату на оси Z .

Для представления информации о шрифте выбран формат TTF [14] в силу того, что он позволяет передавать векторные представления символов ASCII.

Доступные символы указываются пользователем в формате JSON [13] или TXT, так как оба формата позволяют передавать массивы символов.

3.3 Интерфейс пользователя

Для реализации текстового интерфейса взаимодействия была выбрана библиотека tui-go [17], так как:

- средств библиотеки достаточно для работы с растровой графикой и вывода её в терминал;

- библиотека является кроссплатформенной.

Интерфейс содержит в себе окно вывода информации и поле ввода, через которое происходит взаимодействие с ПО. Пользователь управляет программой с помощью набора команд:

- команда загрузки: l ПУТЬ_К_ФАЙЛУ;
- команда удаления: rm ID;
- команда вращения: r ID УГОЛ ОСЬ(x/y/z);
- команда перемещения: t ID tX tY tZ;
- команда масштабирования: s ID sX sY sZ;
- команда добавления источника света: ls X Y Z ЯРКОСТЬ;
- команда удаления источника света: rmls ID;
- команда перемещения камеры: mv РАССТОЯНИЕ;
- команда выхода: q.

3.4 Модульное тестирование

Модульное тестирование проводилось с помощью стандартного пакета Go – testing.

В качестве меры качества покрытия кода использовалась метрика стандартной утилиты cover входящей в состав Go, которая измеряет покрытие кода как процент от выполненных хотя бы единожды выражений [18].

Тестами были покрыты математические модули, модули матричных преобразований, отсечения и чтения объектов.

Результаты тестов с покрытием кода по модулям:

```
Running tests in internal/renderer...
ok
coverage: 29.5% of statements
Running tests in internal/object...
ok
coverage: 94.4% of statements
Running tests in internal/transformer...
ok
coverage: 36.5% of statement
```

Все модульные тесты были пройдены.

3.5 Функциональное тестирование

В рамках функционального тестирования были проанализированы отдельные изображения, полученные с помощью разработанного программного обеспечения.

Для получения изображений был реализован отдельный вариант сборки, генерирующий

изображения для заданных объектов с заданным освещением и сохраняющий их в виде текстовых файлов.

На рисунках 3.1-3.4 представлена модель сферы, освещенная источниками света, находящимися на различных позициях. Радиус сферы равен 5, центр совпадает с началом координат.

На рисунке 3.1 источник света находится в точке с координатами (10, 10, -10). Сфера освещена корректно.

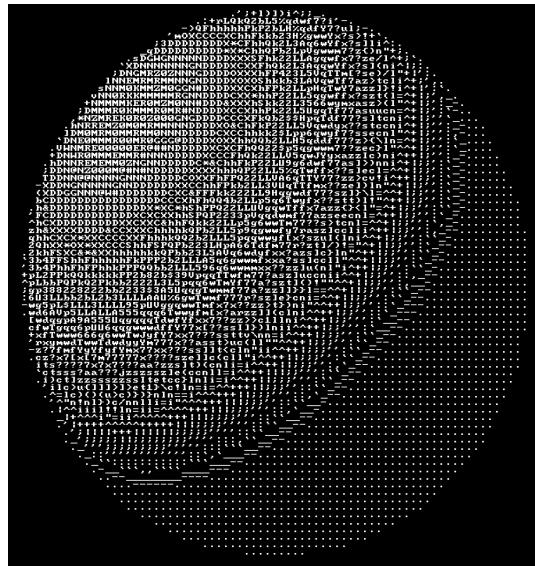


Рисунок 3.1 — Источник света находится в точке с координатами (10, 10, -10)

На рисунке 3.2 источник света находится в точке с координатами (-10, -10, 10). Сфера освещена корректно.

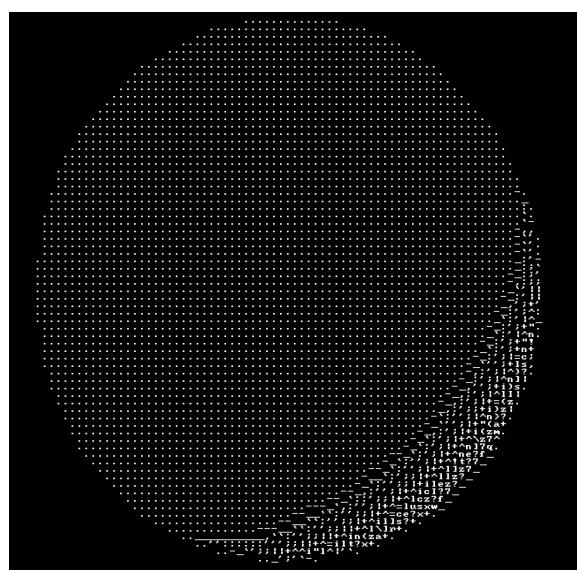


Рисунок 3.2 — Источник света находится в точке с координатами (-10, -10, 10)

На рисунке 3.3 источник света находится в точке с координатами (0, 0, 0), то есть внутри сферы. Как и ожидалось, сфера не освещена.

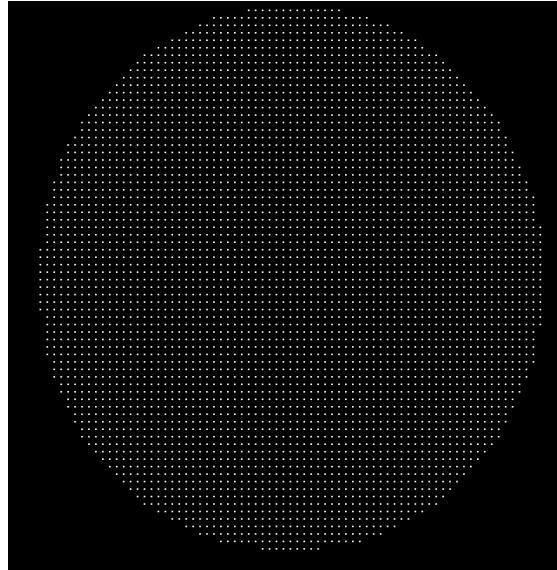


Рисунок 3.3 — Источник света находится в точке с координатами $(0, 0, 0)$

На рисунке 3.4 источник света находится в точке с координатами $(0, 0, -10)$, то есть перед сферой. Сфера освещена корректно.

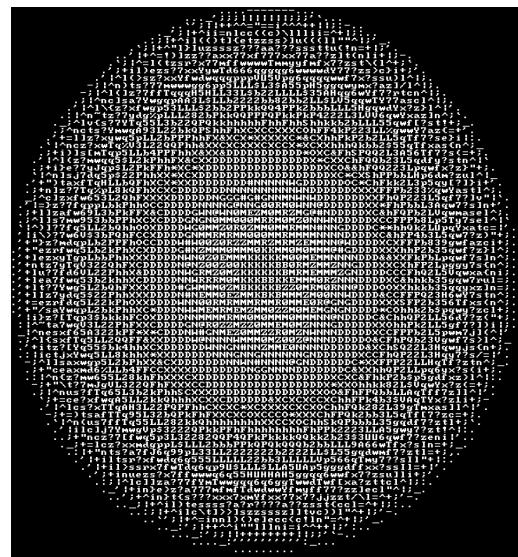


Рисунок 3.4 — Источник света находится в точке с координатами $(0, 0, -10)$

На рисунке 3.5 источник света находится в точке с координатами $(0, 0, 10)$, то есть за сферой. Как и ожидалось, освещённую часть сферы не видно.

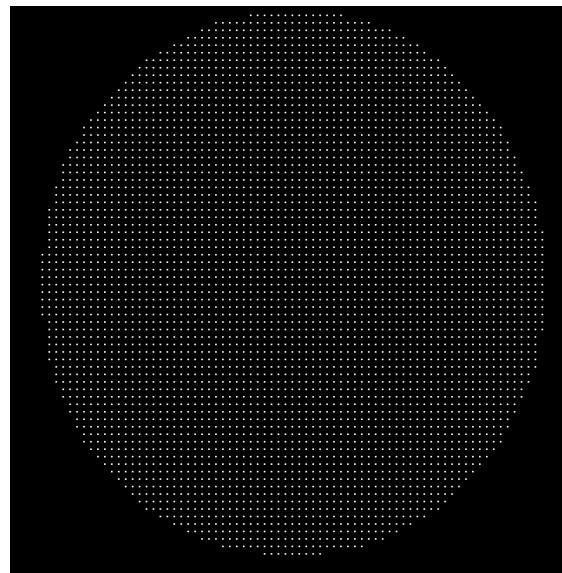


Рисунок 3.5 — Источник света находится в точке с координатами $(0, 0, 10)$

3.6 Выводы из технологической части

Было реализовано программное обеспечение для построения трёхмерного изображения объёмных объектов, а также проведено его функциональное тестирование.

4 Исследовательская часть

В данном разделе будет произведено исследование зависимости качества изображения от доступного набора символов.

Для оценки качества изображения будут исследованы две его составляющие: качество отображения освещенности и качества передачи формы.

4.1 Выбор входных параметров

Сравниваться будут два набора символов:

- 1) полный набор из 128 символов ASCII;
- 2) набор из 32 символов ASCII.

Символы для второго набора были выбраны следующим образом:

- 1) сортировка всех символов по яркости (количеству пикселей в их растровом представлении);
- 2) добавление каждого четвёртого символа в набор.

Для сравнения были построены изображения следующих объектов:

- 1) сфера с радиусом 5 и центром в начале координат, освещённая источником света (-10, -10, -10);
- 2) каркас куба с длиной ребра 2.

Оценка качества отображения освещенности объекта будет производиться по изображениям первого объекта, а оценка качества передачи формы ребер объекта будет производиться по изображениям второго.

Остальные входные данные являются фиксированными.

4.2 Проведение исследования

С выбранными входными данными были построены изображения 4.1-4.4.

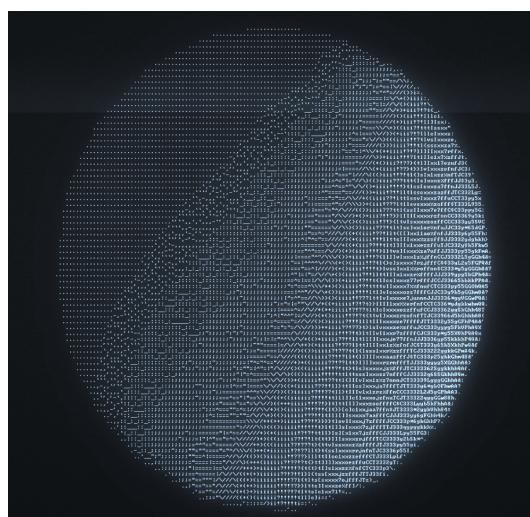


Рисунок 4.1 — Набор 1

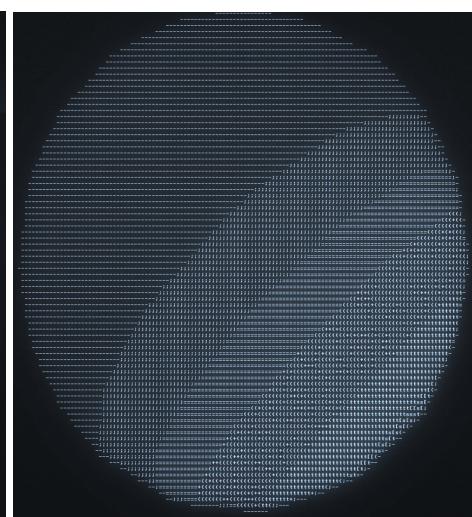


Рисунок 4.2 — Набор 2

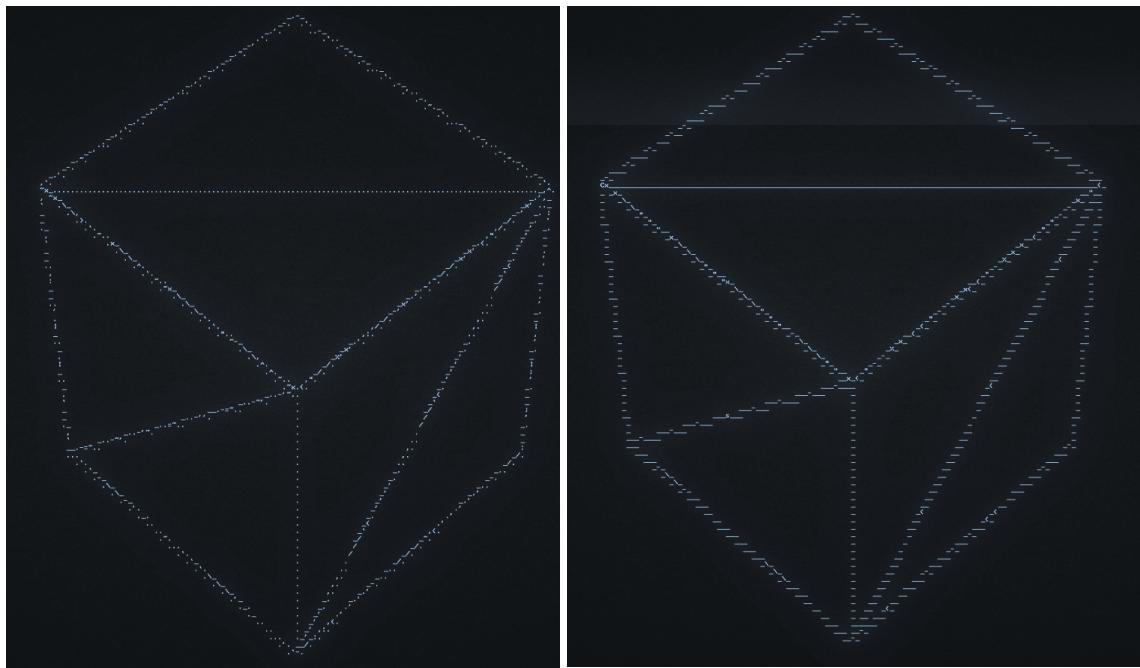


Рисунок 4.3 — Набор 1

Рисунок 4.4 — Набор 2

Данные изображения были помещены в социальный опрос со следующими вопросами:

В отношении первой фигуры, для каждого из наборов:

- 1) Кажется ли вам, что объект на изображении имеет форму сферы?
- 2) Вы видите на объекте правдоподобные блики?
- 3) Вы ощущаете глубину изображения, как если бы это был трехмерный объект?
- 4) Выберите, где, по вашему мнению, может находиться источник света (справа сверху, справа снизу, слева сверху, слева снизу).

В отношении второй фигуры, для каждого из наборов:

- 1) Кажется ли вам, что объект на изображении имеет форму куба?
- 2) Чётко ли вы видите рёбра объекта?
- 3) Все ли рёбра объекта выглядят прямыми?

Для проведения опроса использовалась платформа Yandex Forms [19]. Данный опрос был пройден 54 людьми. Результаты опроса представлены на рисунках 4.5 и 4.6.

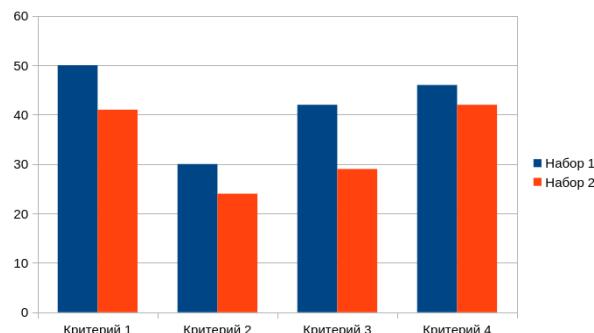


Рисунок 4.5 — Результаты опроса для фигуры 1

Из результатов следует, что сокращенный набор символов негативно повлиял на качество

отображения освещенности объекта.

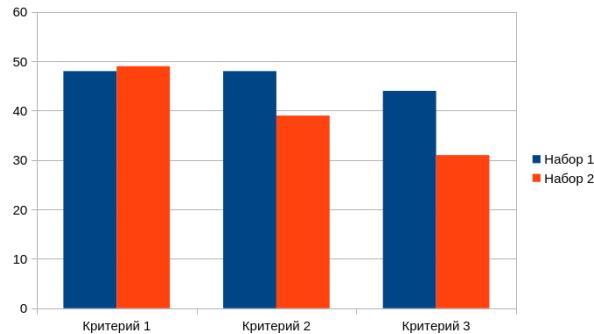


Рисунок 4.6 — Результаты опроса для фигуры 2

Из результатов следует, что сокращенный набор символов негативно повлиял на качество отображения формы рёбер объекта.

4.3 Выводы из исследовательской части

В результате проведенного исследования была установлена зависимость качества изображения от используемого набора символов. Сравнение двух наборов символов — полного набора из 128 символов ASCII и сокращенного набора из 32 символов ASCII — показало, что уменьшение количества символов негативно оказывается как на восприятии освещенности, так и на передаче формы объектов.

ЗАКЛЮЧЕНИЕ

В результате работы было разработано программное обеспечение, позволяющее строить трёхмерные изображения объёмных объектов и эллипсоидов с использованием заданного пользователем набора ASCII-символов.

Было проведено исследование, в результате которого было установлено, что уменьшение количества доступных символов негативно сказывается как на восприятии освещенности, так и на передаче формы объектов.

В ходе работы поставленная цель была достигнута, все задачи были выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Chung, Moonjun & Kwon, Taesoo. (2022). Fast Text Placement Scheme for ASCII Art Synthesis.[Статья] IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3167567.
2. Engleng, Raj & D, Ganesh. (2024). Image-to-ASCII Conversion and Preservation.[Статья] International Journal of Innovative Research in Computer and Communication Engineering. 12. 1722-1727. 10.15680/IJIRCCE.2024.1203058.
3. Xu, X., Zhang, L., Wong, T. 2010. Structure-based ASCII Art.[Статья] ACM Trans. Graph. 29, 4, Article 52 (July 2010), 9 pages. DOI = 10.1145/1778765.1778789 <http://doi.acm.org/10.1145/1778765.1778789>.
4. Иванова Ю.А. Лекция 8. Методы закраски / Иванова Ю.А. [Электронный ресурс] // Корпоративный портал ТПУ : [сайт]. — URL: <https://portal.tpu.ru/SHARED/j/JBOLOTOVA/academic/ComputerGraphics> (дата обращения: 19.11.2024).
5. Роджерс Д. Алгоритмические основы машинной графики //. — Москва: Мир, 1989.
6. Компьютерная графика. Рейтрайсинг и растеризация. — СПб.: Питер, 2022. — 224 с.: ил. — (Серия «Библиотека программиста»).
7. Shading: Flat vs. Gouraud vs. Phong [Электронный ресурс] Режим доступа: <https://www.baeldung.com/cs/shading-flat-vs-gouraud-vs-phong> (дата обращения 13.10.24)
8. SEOUL NATIONAL UNIVERSITY open courseware [Электронный ресурс] Режим доступа: <https://ocw.snu.ac.kr/sites/default/files/NOTE/3095.pdf> (дата обращения 13.10.24)
9. Anders Carlsson, A. Bill Miller Future Potentials for ASCII art [Текст] / Anders Carlsson, A. Bill Miller // CAC. Paris, France . — 2012. — № 3.
10. Jung-Wei Chen, Jiajie Zhang Comparing Text-based and Graphic User Interfaces for novice and expert users / Jung-Wei Chen, Jiajie Zhang // AMIA Annu Symp Proc. — 2007.
11. Marschner, S. and Shirley, P., 2015. Fundamentals of computer graphics (3rd ed.). CRC Press
12. Овчинникова А. А., Толоконцева А. С., Валагов Д. А. ПЕРЕКОДИРОВКА И ОБРАБОТКА СТРОК: ОТ ASCII К ANCI/DBCS И UNICODE // Теория и практика современной науки. 2017. №9 (27).

13. Канаев К.А., Фалеева Е.В., Пономарчук Ю.В. Сравнительный анализ форматов обмена данными, используемых в приложениях с клиент-серверной архитектурой // Фундаментальные исследования. – 2015. – № 2-25. – С. 5569–5572.
14. Haralambous Y. Fonts & encodings. – "O'Reilly Media, Inc. 2007.
15. GO // GO programming language URL: <https://go.dev/> (дата обращения: 24.10.2024).
16. Murray, James D., Van Ryper, William Encyclopedia of Graphics File Formats / Murray, James D., Van Ryper, William — 2. — : O'Reilly Media, Inc., 1996 — 1158 с.
17. tui — Go Terminal UI Framework // URL: <https://pkg.go.dev/github.com/hsson/go-tui> (дата обращения: 07.12.2024).
18. The cover story // The Go Programming Language URL: <https://go.dev/blog/cover> (дата обращения: 07.12.2024).
19. Yandex Forms / [Электронный ресурс] // Yandex Cloud : [сайт]. — URL: <https://yandex.cloud/ru/docs/forms/> (дата обращения: 22.12.2024).

Приложение А

Презентация к курсовой работе состоит из 22 слайдов.