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:

«Программа для игры «Тетрис» в трёхмерном пространстве»

Студент ИУ7-51Б (Группа) _____ (Подпись, дата) В. А. Гаврилюк (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) А. В. Силантьева (И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

Полимино — фигуры, составленные из одноклеточных квадратов так, что каждый квадрат примыкает хотя бы к одному соседнему, имеющему с ним общую сторону [1].

Тетрамино — четырехклеточное полимино [1].

Пространственное полимино — полимино, где вместо квадратов используются кубы [1].

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

ЭВМ — электронная вычислительная машина.

ПО — программное обеспечение.

ВВЕДЕНИЕ

Компьютерная графика — совокупность методов и средств для преобразования данных в графическую форму представления и обратно с помощью ЭВМ. Компьютерная графика находит самое широкое применение в различных отраслях науки и техники, промышленности, экономике, учебном процессе, органах управления, быту [2]. Однако сегодня одной из наиболее популярных сфер применения компьютерной графики является область разработки компьютерных игр [3]. За несколько десятков лет индустрия компьютерных игр выросла из домашнего развлечения в многомиллиардный бизнес [4].

Цель работы — разработка программного обеспечения для игры «Тетрис» в трехмерном пространстве. Для достижения поставленной цели требуется решить следующие задачи:

- формализовать объекты сцены;
- рассмотреть известные алгоритмы удаления невидимых линий и поверхностей и выбрать наиболее подходящие алгоритмы;
- спроектировать выбранные алгоритмы;
- выбрать средства реализации ПО;
- выполнить программную реализацию выбранных алгоритмов;
- разработать графический интерфейс программы;
- провести исследование затрат процессорного времени разработанного ПО.

1 Аналитический раздел

1.1 Описание объектов сцены

1.1.1 Модели трёхмерных объектов

Наибольшее распространение в компьютерной графике получили три типа моделей [5]: каркасные, поверхностные и твердотельные.

– *Каркасная модель* — каркасная конструкция в виде проволочной сетки, охватывающей объект по линиям пересечения ограничивающих его поверхностей, является простейшим и самым примитивным способом передачи формы объемного тела. Проволочные модели применяются для быстрого, не требующего детальной визуализации, эскизного изображения объектов и их габаритных оболочек [5].

– *Поверхностная модель* — модель, представляющая объект в виде тонких поверхностей, под которыми находится пустое пространство, не заполненное материалом объекта [6]. Недостатком поверхностной модели является отсутствие информации о том, с какой стороны поверхности находится материал.

– *Сплошная, объемная или твердотельная модель* — модель, охватывающая все точки внутри и на поверхности объекта [5].

1.1.2 Способы представления объектов

Современная графическая система способна с высокой скоростью выполнять операции закраски проекций плоских многоугольников, удаления невидимых поверхностей, представленных в форме множества плоских многоугольников, и наложения на них разного рода текстур. Поэтому если возникает необходимость включить в состав сцены криволинейный объект, например сферу, его зачастую стремятся с самого начала приближенно представить (аппроксимировать) множеством плоских многоугольников [7].

Такой способ представления объекта является полигональной сеткой — совокупностью связанных между собой плоских многоугольников [8].

В качестве способа представления объектов, помимо полигонального представления, могут выступать также явное или неявное аналитическое, параметрическое, воксельное представления [9].

Воксельное представление подразумевает разбиение пространства сцены на кубические или сферические ячейки, называемые вокселами, и установление состояния каждой ячейки — свободна она или занята объёмом тела. Структура данных воксельного объекта представляется трёхмерной $n \times m \times l$-матрицей битов состояний ячеек пространства сцены [5].

Для описания поверхности с помощью явной аналитической формы потребуется использовать две независимые переменные, и уравнение поверхности в явном виде будет выглядеть так [7]:

$$z = f(x, y). \tag{1.1}$$

При описании поверхности неявными функциями поверхность задается с помощью уравнения $f(X, Y, Z) = 0$, где X, Y, Z — координаты объектного пространства [6].

Поверхности, заданные в форме $X = X(u, t), Y = Y(u, t), Z = Z(u, t)$, где u, t — параметры, изменяющиеся в заданных пределах, относятся к классу параметрических. Для одной фиксированной пары значений u, t можно вычислить только положение одной точки поверхности. Для полного представления о всей поверхности необходимо с определенным шагом перебрать множество пар u, t из диапазона их изменений, вычисляя для каждой пары значение X, Y, Z в трехмерном пространстве. Любую поверхность, описанную неявно, можно представить параметрически [6].

1.1.3 Формализация объектов сцены

Объектами сцены являются игровое поле с уже поставленными кубами, активный блок, прозрачная проекция активного блока, фиксированный источник света и камера.

Блок — фигура, являющаяся элементом пространственного полимино. Блок состоит из кубов, где куб — минимальная единица объема в контексте игрового пространства. Куб может обладать скругленными гранями, при этом радиус скругления может варьироваться. На рисунке 1.1 представлены смоделированные варианты кубов с различными радиусами скругления. На

рисунке 1.2 изображены доступные варианты блоков.

Прозрачная проекция активного блока не является объектом, существующим в действительности, каким, например, является прозрачный стакан. Поэтому отбрасываемых теней проекция иметь не будет.

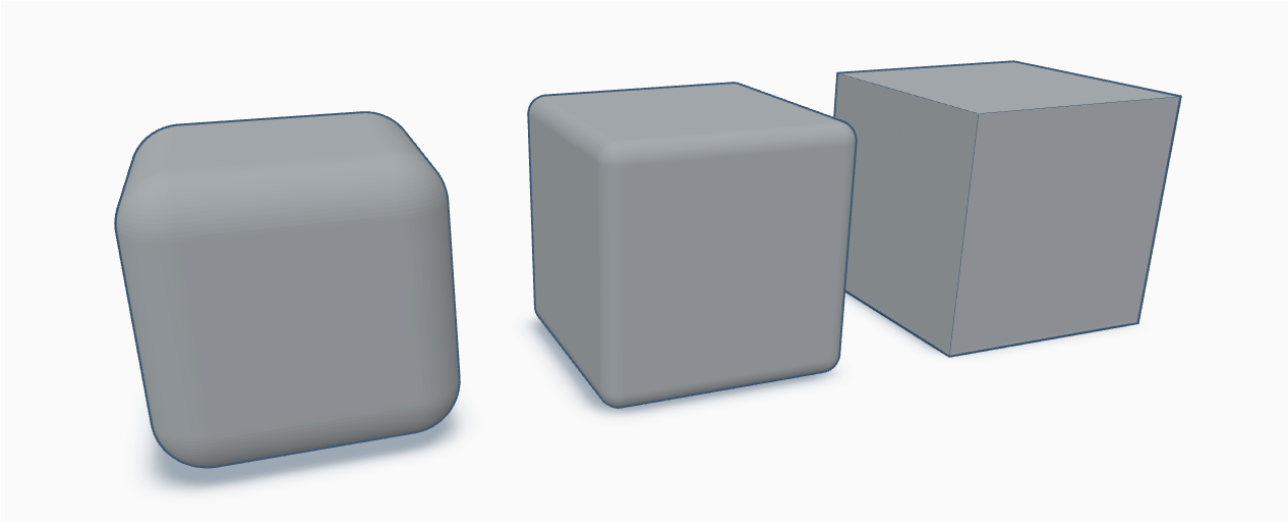

Рисунок 1.1 – Модели куба с различными радиусами скругления

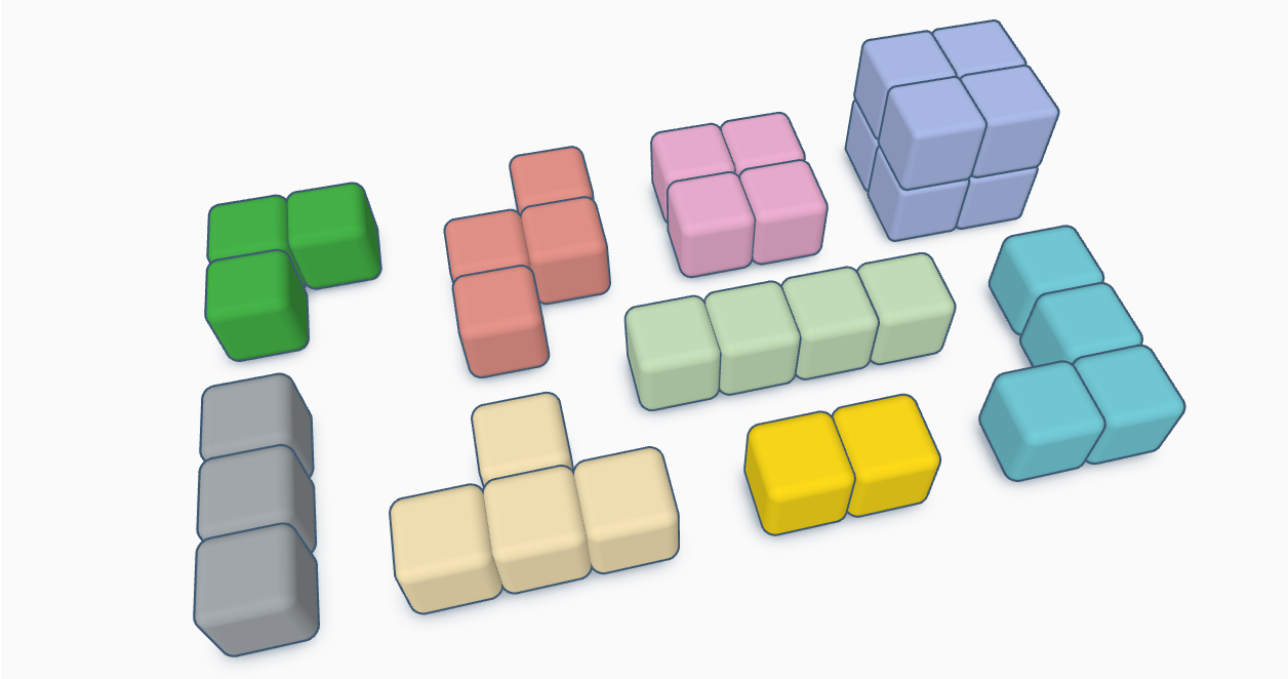

Рисунок 1.2 – Модели доступных вариантов блоков

Игровое поле представляет из себя три стороны параллелепипеда: нижнюю и две боковые стороны. Модели игрового поля вместе с блоками продемонстрированы на рисунке 1.3.

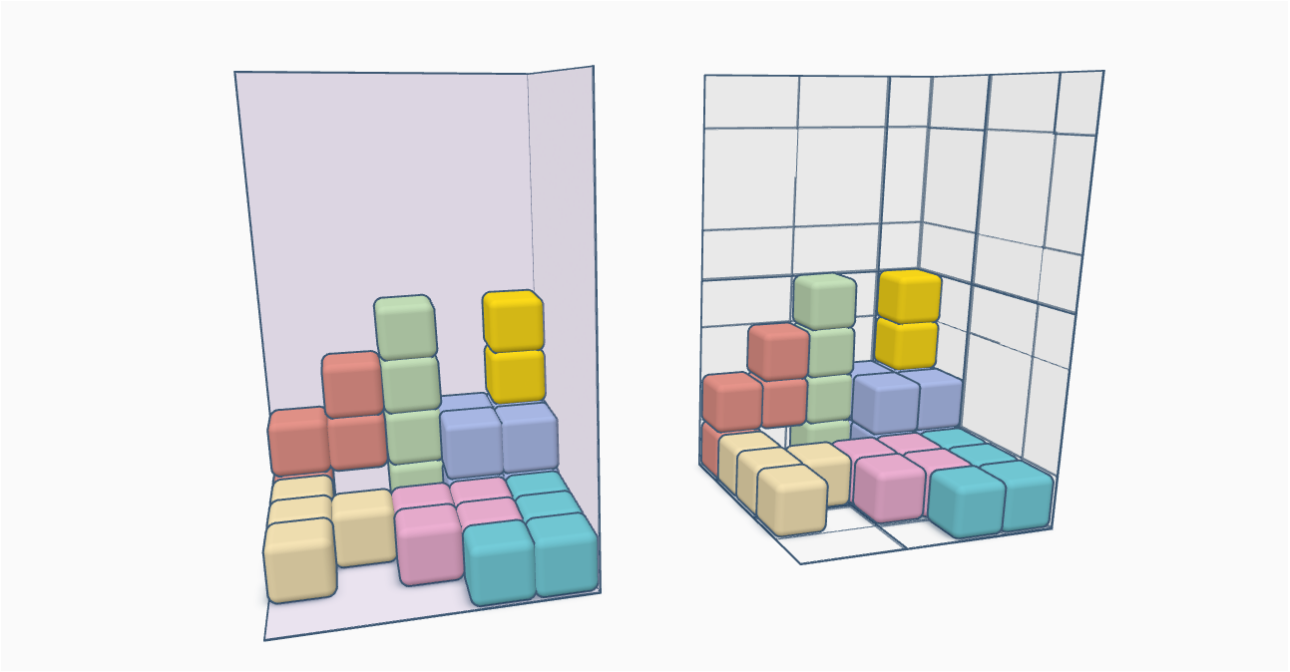

Рисунок 1.3 – Модели игрового поля вместе с блоками

Для представления объектов сцены выбрана поверхностная модель в виде полигональной сетки, т. к. в рамках поставленной задачи нет необходимости хранения информации о материале и внутренних точках объектов, а каркасная модель не всегда может корректно представлять объект. Также объекты, заданные в виде полигональной сетки, не требуют повторной генерации точек при изменениях, как в случае с аналитическими способами представления объекта. Таким образом, формально стороны игрового поля и кубы будут задаваться в виде набора граней, вершин и нормалей к ним.

Так как точность представления объектов с помощью полигональной сетки зависит от количества аппроксимирующих граней, количество аппроксимирующих граней для закругленных углов куба вдоль широтных и долготных линий будет задаваться пользователем.

Фиксированный точечный источник света формально задается положением в пространстве и интенсивностью, камера — положением в пространстве и направлением наблюдения.

1.2 Алгоритмы удаления невидимых линий и поверхностей

Алгоритмы удаления невидимых линий и поверхностей служат для определения линий рёбер, поверхностей или объёмов, которые видимы или

невидимы для наблюдателя, находящегося в заданной точке пространства [10].

Анализ видимости объектов можно производить как в исходном трёхмерном пространстве, так и на картинной плоскости. Это приводит к разделению методов на два класса [11]:

– методы, работающие непосредственно в пространстве самих объектов;

– методы, работающие в пространстве картинной плоскости, т. е. работающие с проекциями объектов [11].

Алгоритмы, работающие в объектном пространстве, имеют дело с физической системой координат, в которой описаны эти объекты. Точность результатов ограничена только точностью вычислений [10].

Алгоритмы, работающие в пространстве изображения, имеют дело с системой координат того экрана, на котором объекты визуализируются. При этом точность вычислений ограничена разрешающей способностью экрана [10].

Ниже рассмотрены наиболее известные алгоритм удаления невидимых линий и поверхностей.

1.2.1 Алгоритм Робертса

В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми. Невыпуклые тела должны быть разбиты на выпуклые части [7].

Алгоритм Робертса делится на четыре этапа:

– подготовка исходных данных, формирование матрицы объекта;

– удаление ребер, экранируемых самим телом;

– удаление невидимых ребер, экранируемых другими телами;

– удаление ребер, образованных при протыкании тел друг другом.

При общем числе граней n трудоёмкость данного алгоритма составляет $O(n^2)$ [9]. Также данный алгоритм не предполагает модификаций для отображения теней и учёта прозрачности.

1.2.2 Алгоритм, использующий z-буфер

Алгоритм z-буфера использует два буфера: z-буфер для хранения значений глубины каждого пикселя и буфер кадра для хранения атрибутов пикселей (цвет или интенсивность).

Алгоритм выполняется по следующим шагам [10]:

- заполнение буфера кадра фоновым значением интенсивности или цвета;
- инициализация z-буфера минимальным значением глубины z;
- преобразование каждого многоугольника в растровую форму;
- вычисление глубины $z(x, y)$ для каждого пикселя в многоугольнике и сравнение вычисленной глубины со значением, которое находится в z-буфере на этой позиции;
- если $z(x, y)$ больше уже записанного значения в z-буфере, то в буфер кадра записывается атрибут этого многоугольника, а в z-буфере значение обновляется на $z(x, y)$;
- в противном случае никаких действий не выполняется.

Трудоёмкость алгоритма, использующего z-буфер, равна $O(CN)$, где C — количество пикселей, N — количество граней [9]. Данный алгоритм поддерживает модификацию для работы с тенями путем добавления дополнительного теневого буфера (карты теней). Модифицированный алгоритм состоит из двух шагов [10]:

- построение сцены из точки наблюдения, совпадающей с источником;
- построение сцены с точки наблюдателя, при этом выполняя проверку на видимость каждого пикселя из точки наблюдения, совпадающей с источником.

Чтобы учесть прозрачность в z-буфере, прозрачные многоугольники следует вносить в отдельный список прозрачных многоугольников, где для каждого пикселя ($z(x, y) > z_{buffer}$) интенсивности складываются в соответствии с формулой [10]:

$$I_{bn} = I_{b0}t_{b0} + I_c t_c, \tag{1.2}$$

где I_{bn} — новое значение интенсивности, I_{b0} — старое значение интенсивности, записанное в буфере интенсивности прозрачности, I_c — интенсивность текущего многоугольника, t_{b0} — старый коэффициент прозрачности из буфера весовых коэффициентов прозрачности, t_c — коэффициент прозрачности текущего многоугольника.

Буферы интенсивности для прозрачных и непрозрачных многоугольников объединяются в соответствии с формулой:

$$I_{fb} = t_{b0}I_{b0} + (1 - t_{b0})I_{fb0}, \tag{1.3}$$

где I_{fb} — окончательная интенсивность в буфере кадра. Формулу (1.3) часто называют альфа-смешением.

Однако, так как в рамках поставленной задачи присутствует только один прозрачный объект — проекция активного блока, для отображения прозрачного объекта нет необходимости в дополнительных буферах интенсивности прозрачных объектов и весовых коэффициентов прозрачности, достаточно отобразить прозрачную проекцию после отображения всех непрозрачных объектов, используя формулу 1.3.

В силу того, что стороны игрового поля и кубы являются выпуклыми объектами, для уменьшения количества обрабатываемых невидимых граней в алгоритме с z-буфером можно применить метод отсечения заведомо невидимых (задних) граней [12].

1.2.3 Алгоритм художника

Алгоритм художника основан на предварительной сортировке граней по глубине. Сначала выводятся более дальние грани, затем (поверх них) более близкие. Фактически алгоритм художника сортирует грани по z-координате, а потом выводит их в полученном порядке [9]. Трудоемкость алгоритм художника составляет $O(n\ \log n)$, где n — количество граней [12]. Проблема данного алгоритма заключается в циклическом перекрытии граней, при котором невозможно корректно отсортировать объекты.

В алгоритм художника возможно встроить обработку прозрачных объек-

тов, однако данный алгоритм не поддерживает модификаций для отображения теней, что делает его использование невозможным в рамках данной курсовой работы.

1.2.4 Сравнение алгоритмов удаления невидимых линий и поверхностей

Ниже представлена сравнительная таблица для алгоритмов удаления невидимых линий и поверхностей, где N — количество граней, а C — количество пикселей окна отрисовки.

Таблица 1.1 – Сравнение алгоритмов удаления невидимых линий и поверхностей

Алгоритм	**Трудоёмкость**	**Поддержка теней**	**Поддержка прозрачности**
Алгоритм Робертса	$O(N^2)$	–	–
Алгоритм художника	$O(N\ \log N)$	–	+
Алгоритм, использующий z-буфер	$O(CN)$	+	+

Для удаления невидимых линий и поверхностей выбран алгоритм, использующий z-буфер, т. к. он поддерживает модификацию для отрисовки теней и прозрачных элементов.

1.3 Модели освещения

Модели освещения разделяют на два типа: глобальные и локальные [10]. Локальные модели учитывают только свет, падающий непосредственно от источника (или источников), и ориентация поверхности. В глобальных моделях учитывается также свет, отражённый от других объектов сцены или пропущенный сквозь них [10].

Основными локальными моделями освещения являются модель Ламберта и модель Фонга.

1.3.1 Модель Ламберта

Стандартная модель освещения Ламберта учитывает только диффузное отражение, при котором падающий в точку P свет равномерно рассеивается во всех направлениях верхней полусферы [9]. Видимая в точке P освещенность не зависит от положения наблюдателя. Данное свойство продемонстрировано на рисунке 1.4.

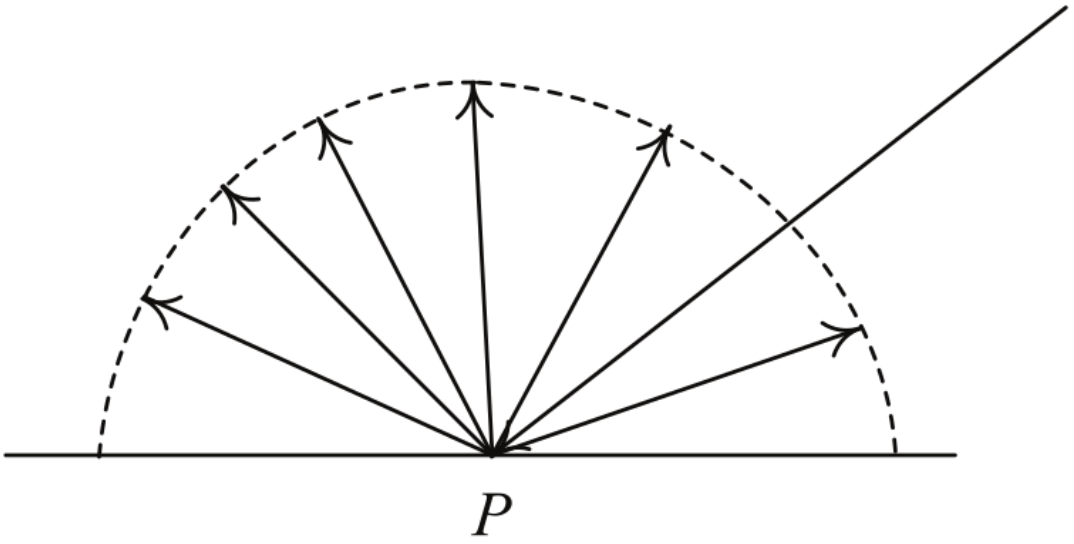

Рисунок 1.4 – Идеальное диффузное рассеивание

Стандартная модель Ламберта записывается в виде формулы (1.4

$$I = I_l k_d \cos(\theta) \quad 0 \leq \theta \leq \pi/2, \tag{1.4}$$

где I — интенсивность отраженного света, I_l — интенсивность точечного источника, k_d — коэффициент диффузного отражения ($0 \leq k_d \leq 1$), θ — угол между направлением света и нормалью к поверхности. В расширенной модели Ламберта также учитывается рассеянный свет:

$$I = I_a k_a + I_l k_d \cos(\theta) \quad 0 \leq \theta \leq \pi/2, \tag{1.5}$$

где I_a — интенсивность рассеянного света, а k_a — коэффициент диффузного отражения рассеянного света ($0 \leq k_a \leq 1$) [10].

1.3.2 Модель Фонга

В модели освещения Фонга кроме диффузной составляющей отражения и рассеянного света также учитываются зеркальные свойства отражающей поверхности:

$$I = I_a k_a + \frac{I_l}{d + K}(k_d \cos(\theta) + k_s \cos^n(\alpha)), \tag{1.6}$$

где k_s — коэффициент зеркального отражения, а α — угол между отраженным лучом и вектором наблюдения [10].

1.3.3 Сравнение моделей освещения

Таблица 1.2 – Сравнение моделей освещения

Модель освещения	**Учёт зеркальных отражений**
Модель Ламберта	–
Модель Фонга	+

Так как в рамках поставленной задачи не требуется моделировать зеркальное отражение, в качестве модели освещения выбрана расширенная модель Ламберта.

1.4 Методы закраски

Выделяют три основных метода закраски: метод постоянного закрашивания, метод Гуро и метод Фонга [11].

1.4.1 Метод постоянного закрашивания

Метод постоянного закрашивания заключается в том, что на грани берётся произвольная точка и определяется её освещённость, которая и принимается за освещённость всей грани. В качестве модели освещения обычно используются расширенная модель освещения Ламберта или модель Фонга [11].

При закраске полигональной модели результирующее изображение носит ярко выраженный полигональный характер [11]. Это связано с тем, что если рассматривать освещённость вдоль поверхности какого-либо объекта, то она

претерпевает разрывы на границах граней (фактически освещённость является кусочно-постоянной функцией) [11].

1.4.2 Метод Гуро

Метод Гуро обеспечивает непрерывность освещённости за счёт билинейной интерполяции [11]. Алгоритм закраски методом Гуро состоит из нескольких этапов [10]:

- вычисление нормалей в вершинах многоугольника;
- линейная интерполяция значений интенсивности вдоль рёбер многоугольника;
- вычисление интенсивности для каждого пикселя с помощью линейной интерполяции значений вдоль сканирующих строк.

1.4.3 Метод Фонга

В методе Фонга вместо значения интенсивности интерполируется вектор нормали [10]. Алгоритм закраски, следовательно, примет следующий вид:

- вычисление нормалей в вершинах многоугольника;
- линейная интерполяция вектора нормали вдоль рёбер многоугольника;
- вычисление вектора нормали для каждого пикселя с помощью линейной интерполяции вдоль сканирующих строк.

1.4.4 Сравнение методов закраски

В таблице 1.3 представлено сравнение методов закраски для одного полигона, представленного в виде треугольника, где N — количество пикселей, принадлежащих полигону.

Таблица 1.3 – Сравнение методов закраски для одного полигона

Метод закраски	**Наличие бликов**	**Кол-во вычислений интенсивности**
Метод постоянного закрашивания	–	1
Метод Гуро	–	3
Метод Фонга	+	N

В качестве метода закраски выбран метод Гуро, так как в рамках поставленной задачи не требуется учитывать блики, а метод постоянного закрашивания не подходит для кривых поверхностей, аппроксимированных полигональной сеткой, в силу однотонности закраски и отсутствия интерполяции интенсивности между полигонами.

1.5 Правила игры «Тетрис»

Далее приведены правила игры «Тетрис», адаптированные под трехмерное игровое поле:

- удаление слоя происходит при его полном заполнении кубами, где слой — набор ячеек игрового поля, лежащих на одной высоте;
- количество начисляемых очков равно квадрату количества слоёв, удалённых за одну постановку активного блока;
- игра прекращается, если хотя бы один из кубов достиг последнего слоя игрового поля.

Вывод

В данном разделе были рассмотрены основные модели и способы представления трехмерных объектов, методы удаления невидимых линий и поверхностей, модели освещения, методы закраски и правила игры «Тетрис».

Была выполнена формализация объектов сцены. Для представления сторон игрового поля и кубов выбрана поверхностная модель в виде полигональной сетки, т. к. в рамках поставленной задачи нет необходимости хранения

информации о материале и внутренних точках объектов, а каркасная модель не всегда может корректно представлять объект.

В качестве метода закраски выбран метод Гуро, так как в рамках поставленной задачи не требуется учитывать блики, а метод постоянного закрашивания не подходит для кривых поверхностей, аппроксимированных полигональной сеткой, в силу однотонности закраски и отсутствия интерполяции интенсивности между полигонами.

Для удаления невидимых линий и поверхностей выбран алгоритм, использующий z-буфер, т. к. он поддерживает модификацию для отрисовки теней и прозрачных элементов.

2 Конструкторский раздел

В данном разделе описана декомпозиция поставленной задачи и приведены схемы выбранных алгоритмов.

2.1 Декомпозиция задачи

На рисунках 2.1—2.2 представлена декомпозиция поставленной задачи.

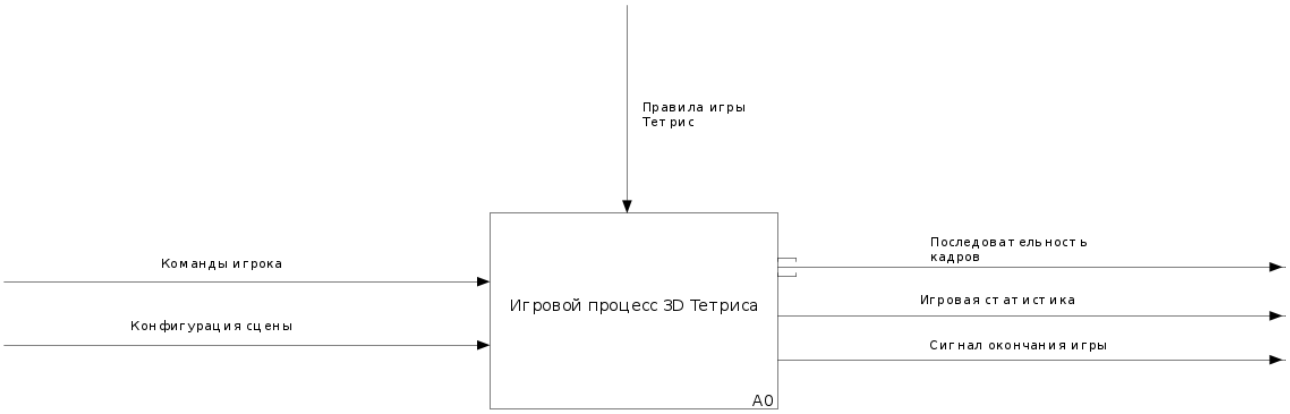

Рисунок 2.1 – Верхний (нулевой) уровень A0

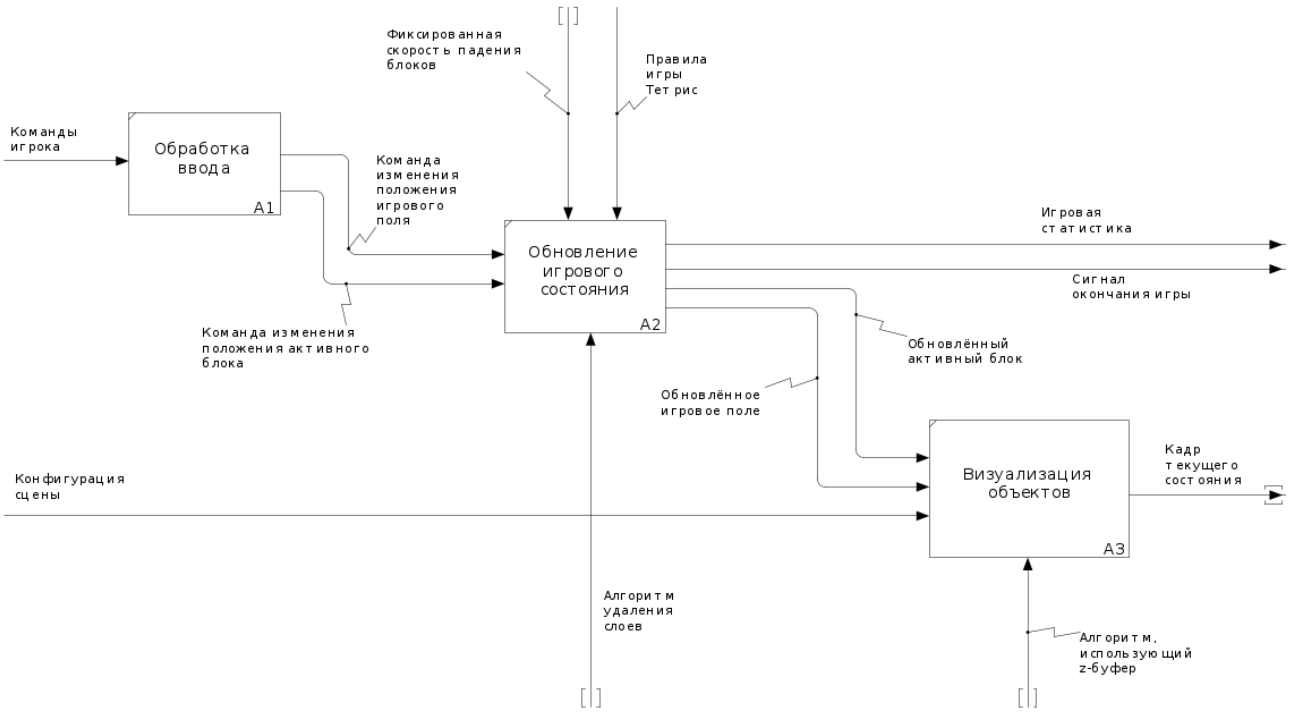

Рисунок 2.2 – Декомпозиция блока A0

2.2 Алгоритм, использующий z-буфер

На рисунке 2.3 изображена схема модифицированного алгоритма, использующего z-буфер.

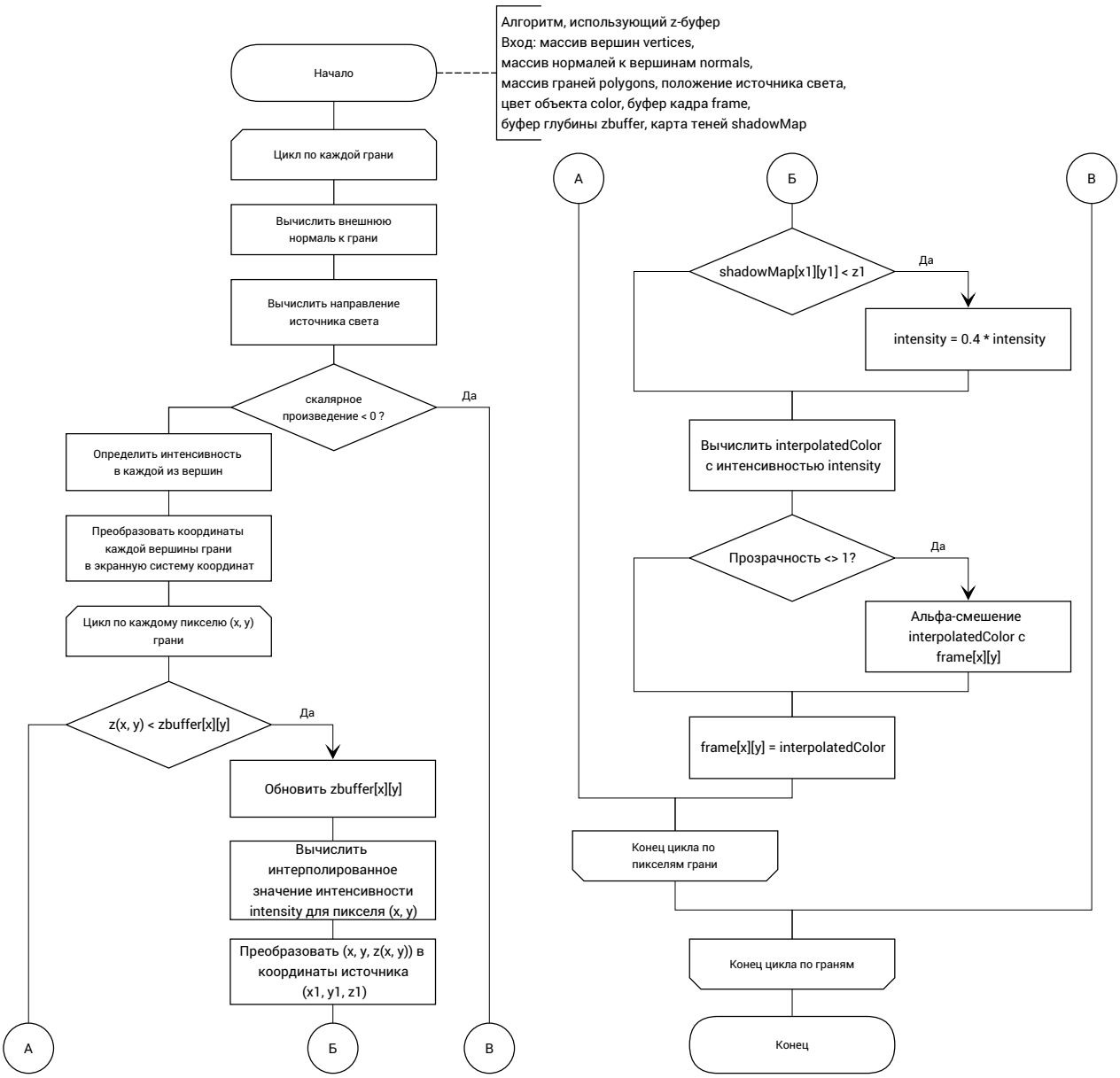

Рисунок 2.3 – Схема модифицированного алгоритма, использующего z-буфер

2.3 Алгоритм заполнения карты теней

На рисунке 2.4 изображена схема модифицированного алгоритма заполнения карты теней.

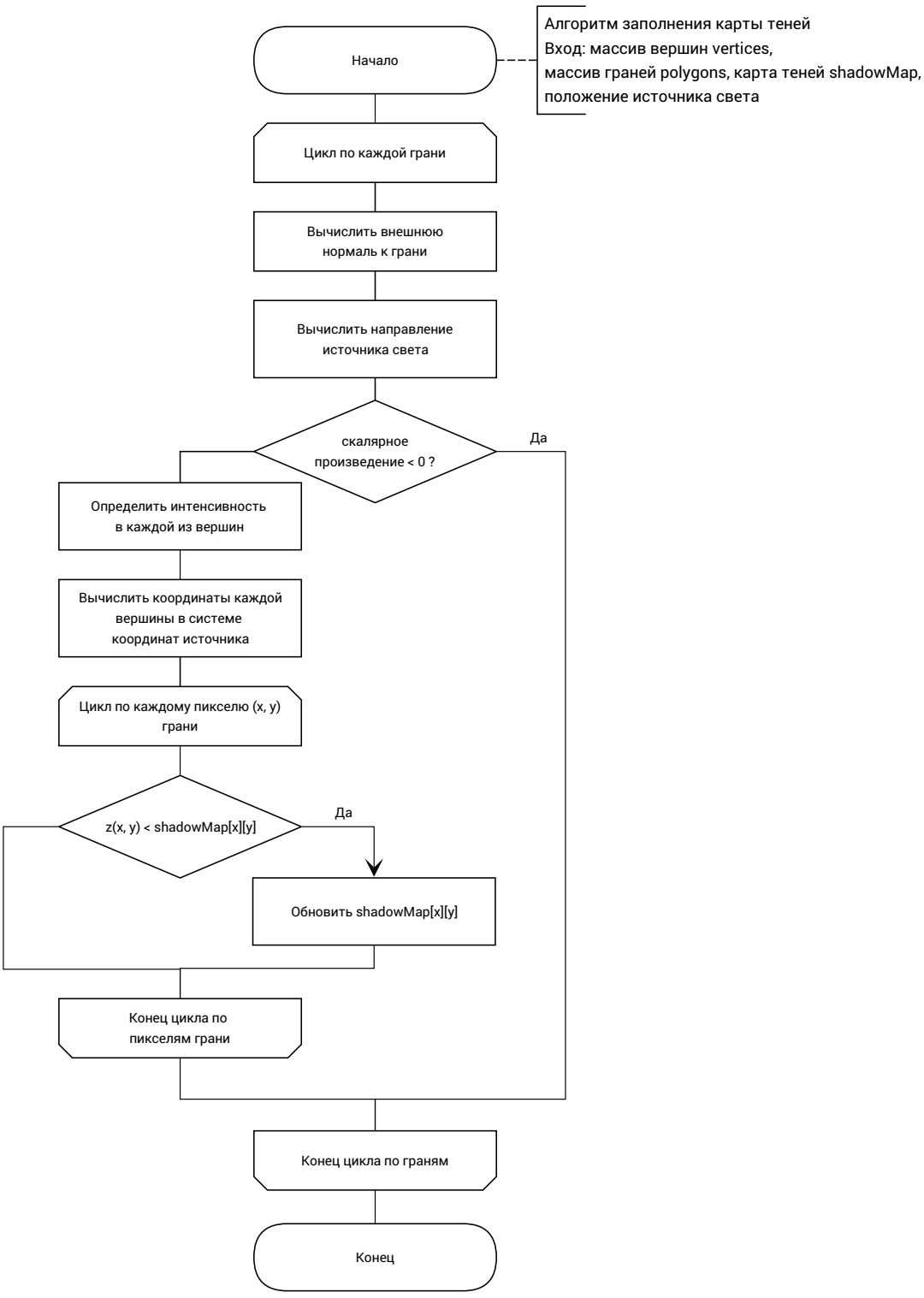

Рисунок 2.4 – Схема модифицированного алгоритма заполнения карты теней

2.4 Алгоритм отрисовки кадра

На рисунке 2.5 приведена схема алгоритма отрисовки кадра.

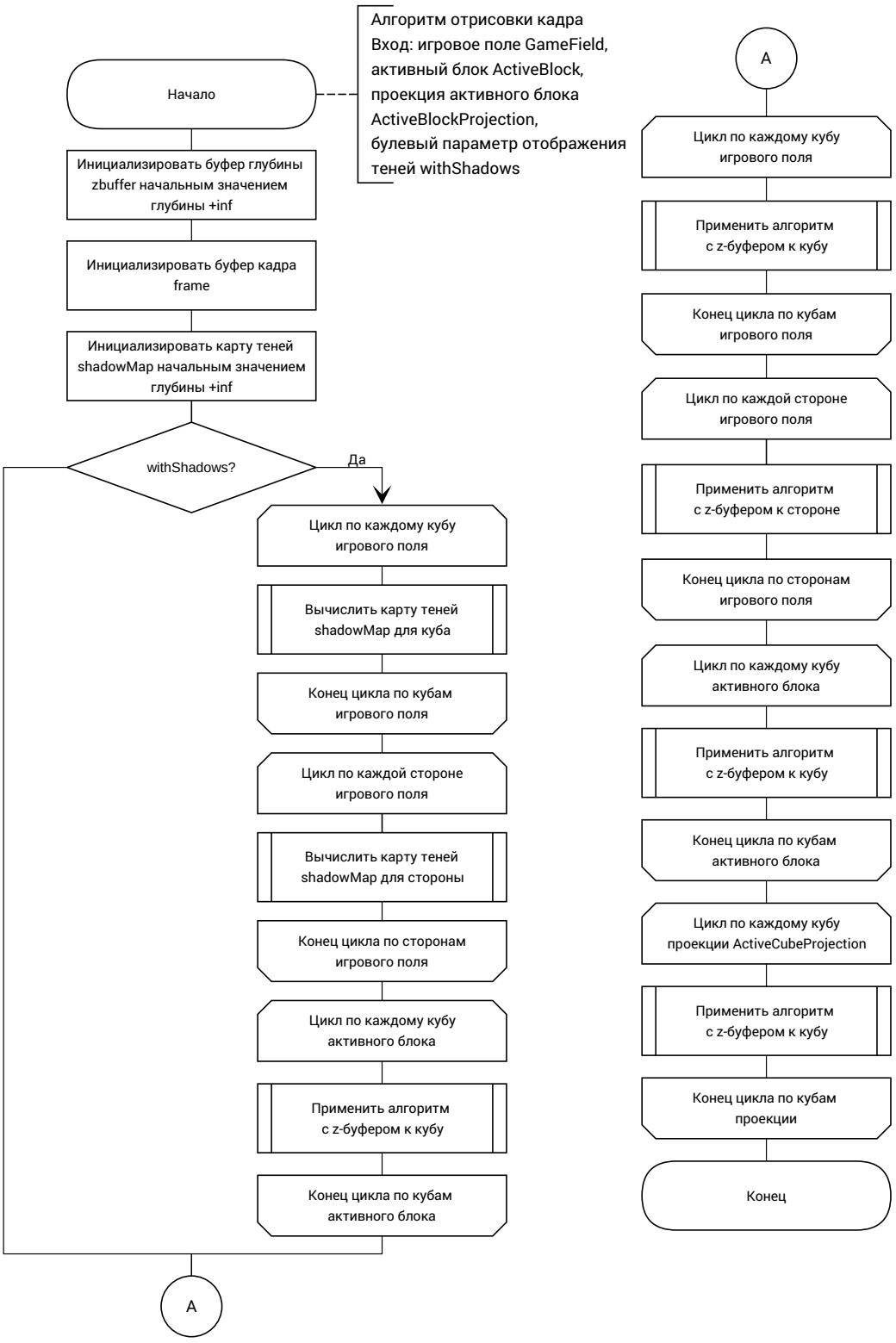

Рисунок 2.5 – Схема алгоритма отрисовки кадра

Вывод

В данном разделе выполнена декомпозиция ПО и приведены схемы алгоритмов:

– отрисовки кадра;

– модифицированного алгоритма заполнения карты теней;

– модифицированного алгоритма, использующего z-буфер.

3 Технологический раздел

3.1 Средства реализации ПО

Для реализации программного обеспечения выбран язык программирования C++, т. к. данный язык предоставляет необходимые инструменты для решения поставленной задачи: библиотеку Qt6 [13] для работы с графическими интерфейсами и встроенную библиотеку chrono для работы с временными метками, интервалами и измерениями времени.

3.2 Структура программы

Разработанная программа состоит из 9-ти классов и одной структуры.
Классы объектов сцены:

- PolygonMeshObject — абстрактный класс для объектов, представленных в виде полигональной сетки;
- Cube — класс, предоставляющий куб со скругленным углами в качестве объекта сцены;
- GameFieldSide — класс стороны игрового поля;

Классы, реализующие основную игровую логику «Тетриса»:

- Scene — дочерний класс QGraphicsScene, обрабатывающий движения мыши над сценой, для поворота игрового поля;
- Block — класс блока, используется для управления активным блоком и его проекцией;
- GameField — класс, реализующий управление игровым полем и контроль над свободными и занятыми ячейками;
- TetrisEngine — класс управления основным игровым циклом.

Класс MainWindow предоставляет графический интерфейс программы. Для отрисовки сцены используется класс Renderer.

Единственной структурой является структура Settings, необходимая для настройки объектов сцены.

3.3 Интерфейс программы

Взаимодействие пользователя с программой происходит посредством графического интерфейса, компьютерной мыши и клавиш на клавиатуре:

– клавиша «стрелка вверх» — перемещение активного блока вдоль оси Ox в направлении уменьшения значений;

– клавиша «стрелка вниз» — перемещение активного блока вдоль оси Ox в направлении увеличения значений;

– клавиша «стрелка влево» — перемещение активного блока вдоль оси Oy в направлении увеличения значений;

– клавиша «стрелка вправо» — перемещение активного блока вдоль оси Oy в направлении уменьшения значений;

– клавиша «W» — вращение блока на 90 градусов вокруг оси Ox;

– клавиша «S» — вращение блока на -90 градусов вокруг оси Ox;

– клавиша «A» — вращение блока на 90 градусов вокруг оси Oz;

– клавиша «D» — вращение блока на -90 градусов вокруг оси Oz;

– клавиша «Q» — вращение блока на 90 градусов вокруг оси Oy;

– клавиша «E» — вращение блока на -90 градусов вокруг оси Oy;

– клавиша «Space» — быстрый спуск активного блока.

Вращение игрового поля выполняется с помощью компьютерной мыши при нажатой правой кнопке мыши.

На рисунке 3.1 продемонстрирован графический интерфейс программы. В правой части графического интерфейса присутствует панель управления, которая позволяет задать радиус скругления кубов, количество аппроксимирующих граней, прозрачность проекции активного блока, размеры игрового поля, положение точечного источника света и наличие отбрасываемых теней. Также предусмотрены кнопки для остановки, возобновления текущей игры и создания новой.

Другим элементом графического интерфейса является панель игрового счёта. На ней отображаются лучший счёт за все время и текущий счет игрока. Лучший счёт сохраняется и загружается при запуске программы.

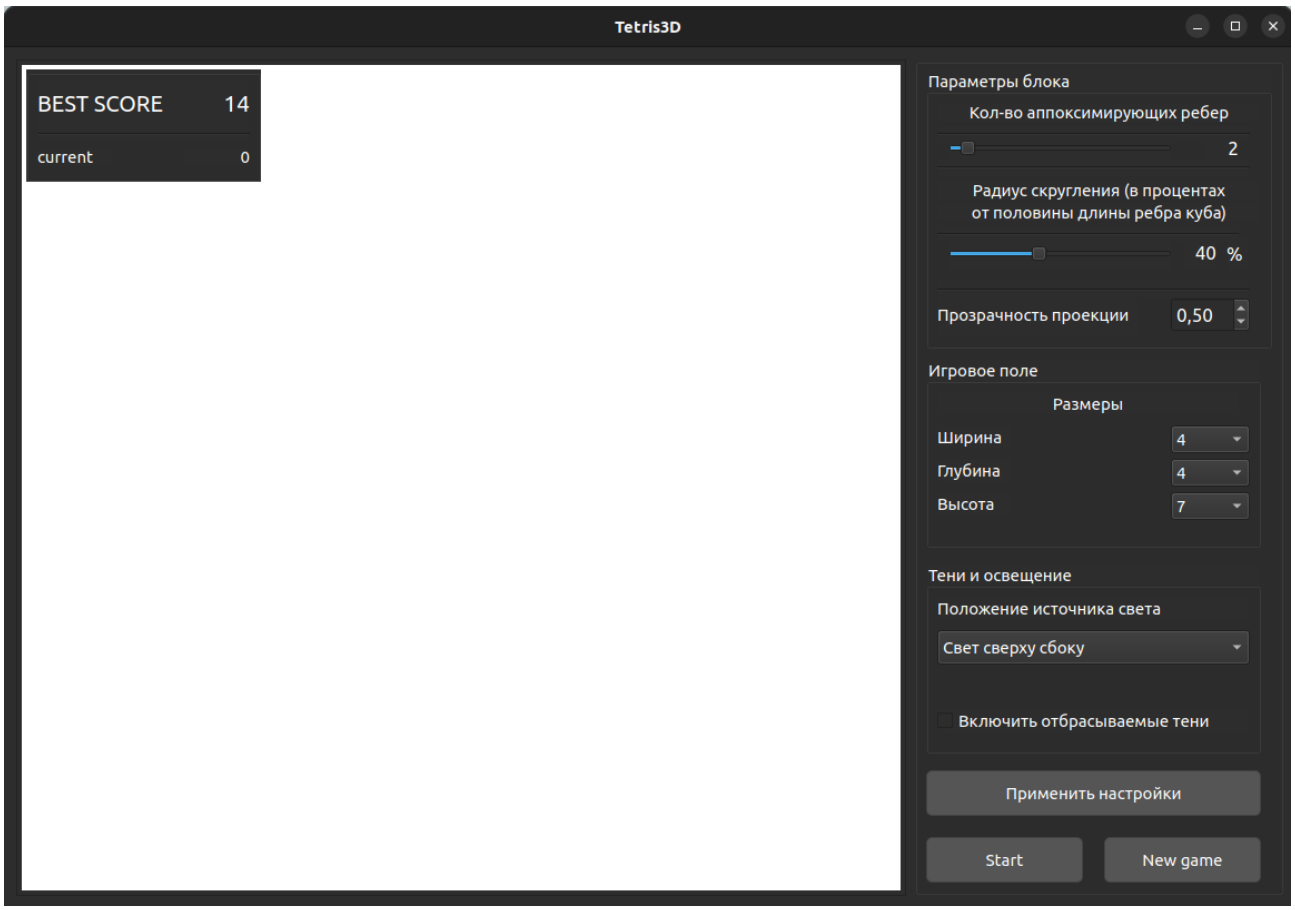

Рисунок 3.1 – Графический интерфейс программного обеспечения

3.4 Пример работы программы

На рисунках 3.2 — 3.8 представлены примеры работы программы.

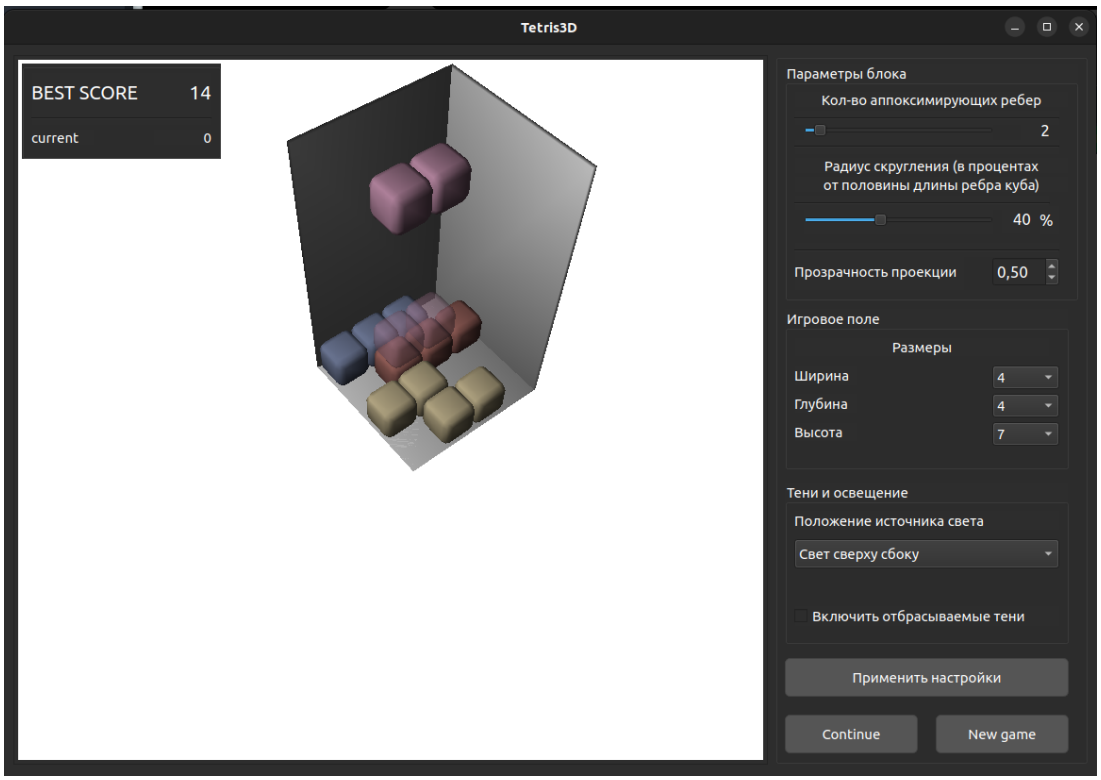

Рисунок 3.2 – Сцена без отбрасываемых теней

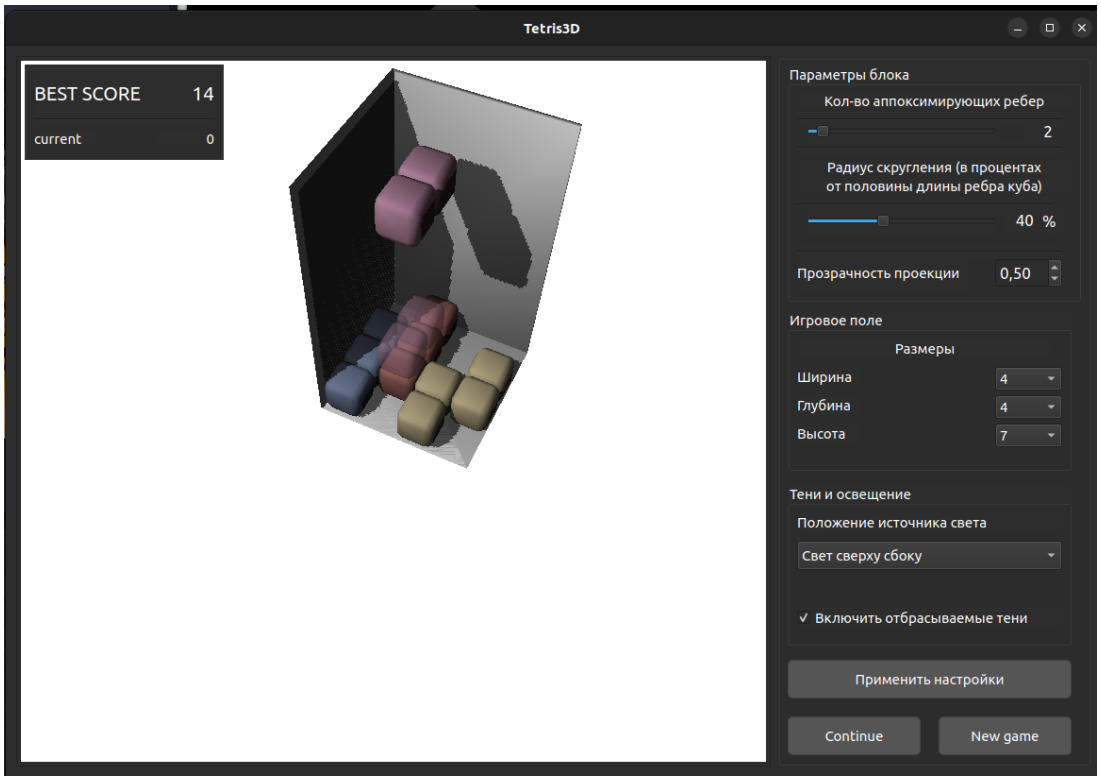

Рисунок 3.3 – Сцена с отбрасываемыми тенями, источник света находится сверху сбоку от игрового поля

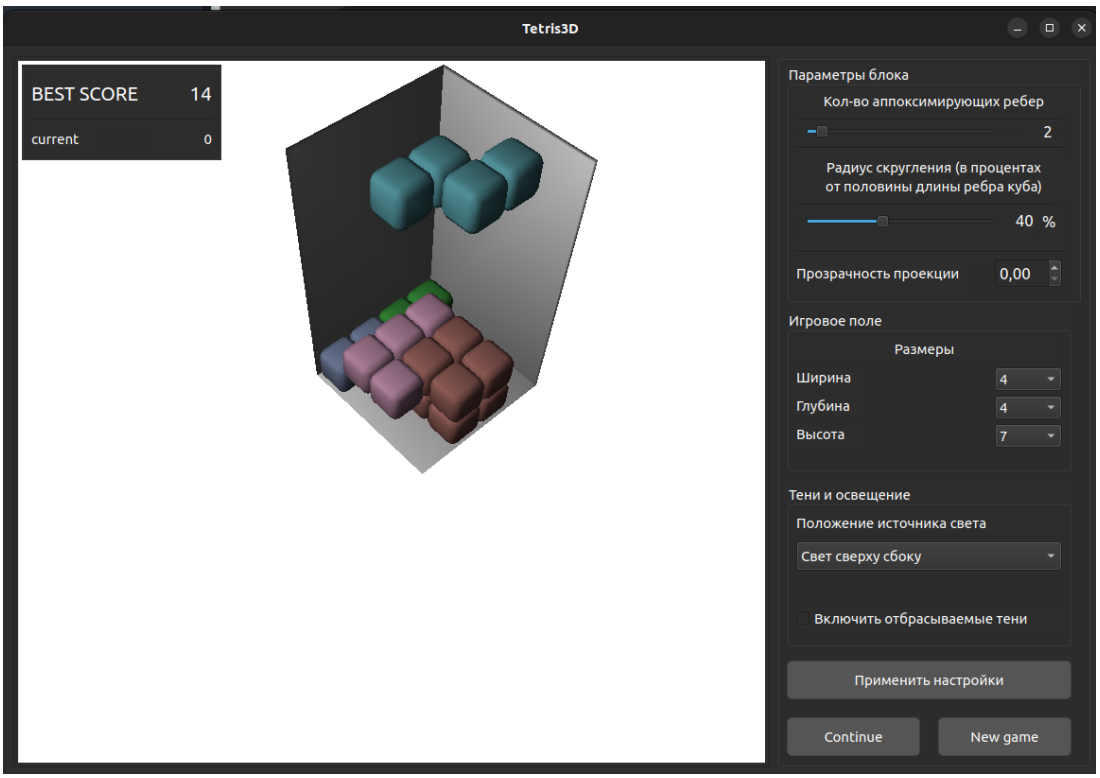

Рисунок 3.4 – Сцена с полностью прозрачной проекцией активного блока

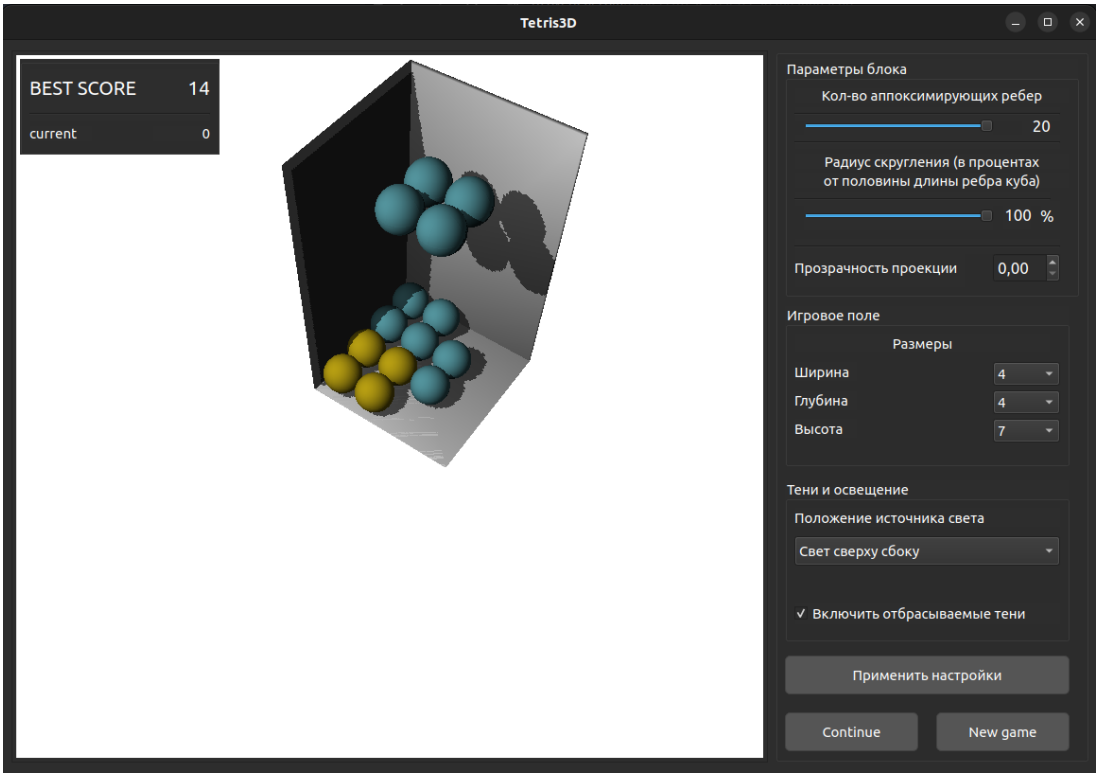

Рисунок 3.5 – Сцена с кубами с радиусом скругления, равным половине ребра

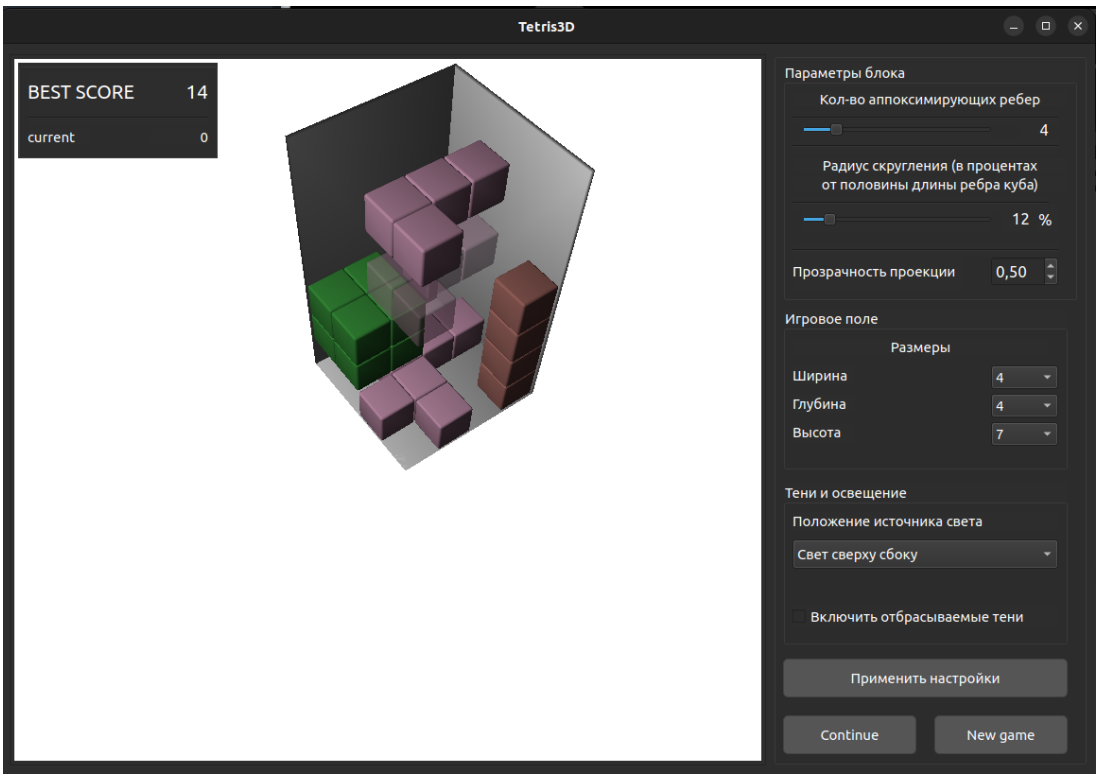

Рисунок 3.6 – Сцена с кубами с радиусом скругления, равным 12% от половины ребра

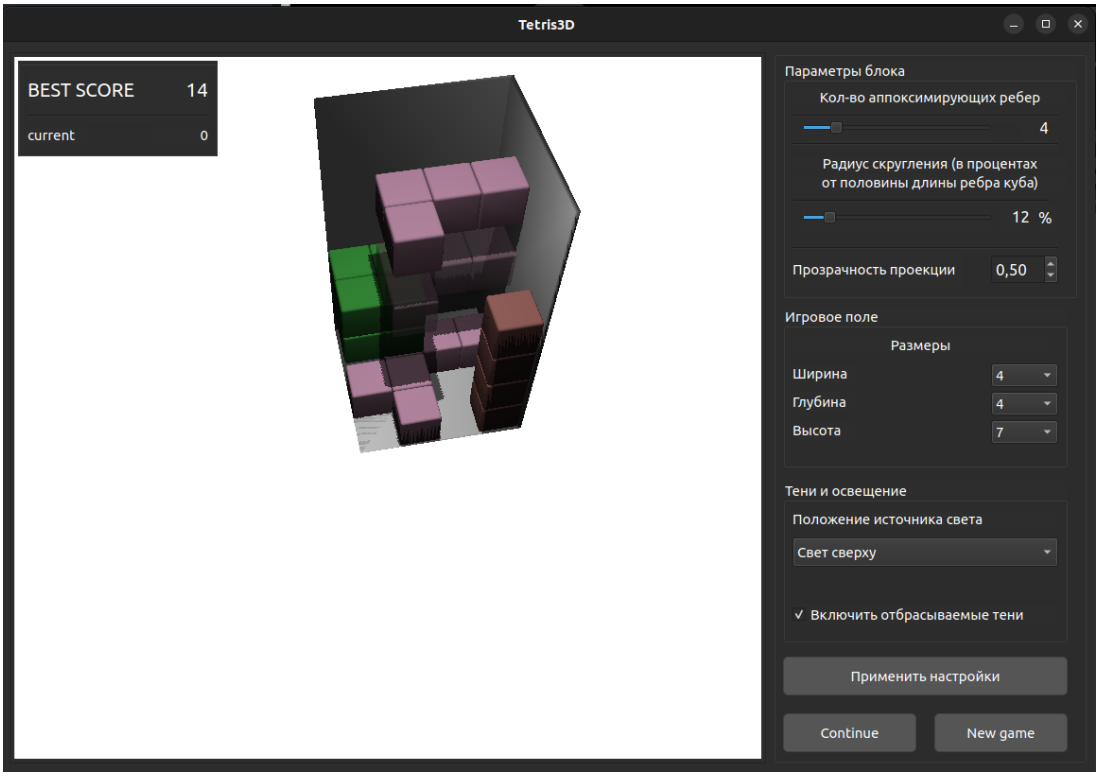

Рисунок 3.7 – Источник света находится над игровым полем

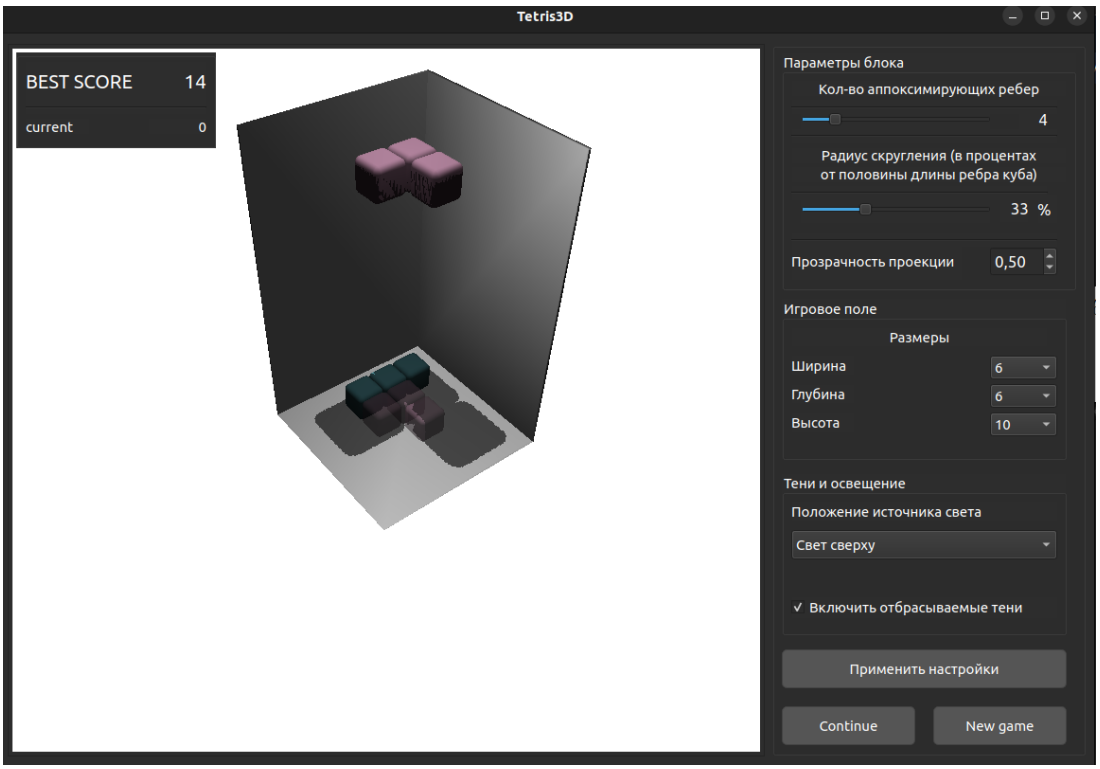

Рисунок 3.8 – Игровое поле при максимальных размерах. Источник света находится над игровым полем

3.5 Функциональное тестирование

В таблицах 3.1—3.2 приведены функциональные тесты для реализованного программного обеспечения. Все тесты пройдены успешно.

Таблица 3.1 – Тестирование управления игровым полем и блоками

№	**Описание теста**	**Ожидаемый результат**	**Тест пройден**
1	Поворот игрового поля вправо посредством движения мыши вправо	Игровое поле повернулось вправо	Да
2	Поворот игрового поля влево посредством движения мыши влево	Игровое поле повернулось влево	Да
3	Игровое поле повернуто таким образом, что ось Oy направлена влево. Перемещение блока вперед (вдоль оси Ox)	Блок переместился вперед	Да
4	Игровое поле повернуто таким образом, что ось Oy направлена влево. Перемещение блока назад (вдоль оси Ox)	Блок переместился назад	Да
5	Игровое поле повернуто таким образом, что ось Oy направлена влево. Перемещение блока влево (вдоль оси Oy)	Блок переместился влево	Да
6	Игровое поле повернуто таким образом, что ось Oy направлена влево. Перемещение блока вправо (вдоль оси Oy)	Блок переместился вправо	Да
7	Быстрый спуск активного блока нажатием клавиши «Space»	Блок опустился быстрее, чем при стандартной скорости падения	Да

Таблица 3.2 – Тестирование вращения блока и удаления слоёв

№	**Описание теста**	**Ожидаемый результат**	**Тест пройден**
8	Вращение блока вокруг оси Ox в положительном направлении	Блок повернулся вокруг оси Ox на 90 градусов в положительном направлении	Да
9	Вращение блока вокруг оси Ox в отрицательном направлении	Блок повернулся вокруг оси Ox на 90 градусов в отрицательном направлении	Да
10	Вращение блока вокруг оси Oy в положительном направлении	Блок повернулся вокруг оси Oy на 90 градусов в положительном направлении	Да
11	Вращение блока вокруг оси Oy в отрицательном направлении	Блок повернулся вокруг оси Oy на 90 градусов в отрицательном направлении	Да
12	Вращение блока вокруг оси Oz в положительном направлении	Блок повернулся вокруг оси Oz на 90 градусов в положительном направлении	Да
13	Вращение блока вокруг оси Oz в отрицательном направлении	Блок повернулся вокруг оси Oz на 90 градусов в отрицательном направлении	Да
14	Полностью заполнен один слой на игровом поле	Слой удалён, все блоки выше переместились вниз	Да
15	Полностью заполнены два слоя на игровом поле	Оба слоя удалены, все блоки выше переместились вниз	Да
16	Слои не полностью заполнены	Никакие слои не удаляются, игровое поле остаётся неизменным	Да

Вывод

В рамках данного раздела с помощью выбранных средств было реализована и протестирована программа для игры «Тетрис» в трехмерном пространстве. Также была описана структура реализованного программного обеспечения.

4 Исследовательский раздел

Данный раздел содержит описание проведённых исследований затрат процессорного времени и их условий проведения.

4.1 Условия проведения исследования

Измерение времени проводилось на ЭВМ со следующими характеристиками:

– операционная система Ubuntu 22.04.5 LTS;

– объем оперативной памяти 16 Гб;

– процессор Intel Core i7-8700K CPU 3.70Ггц × 12 [14].

Для замера времени использовались функции из встроенной библиотеки C++ chrono: *std::chrono::high_resolution_clock::now* для получения точного значения текущего времени в формате временной отметки *std::chrono::time_point*, *std::chrono::duration* для хранения интервалов времени.

4.2 Исследование №1

В первом исследовании измерялось процессорное время отрисовки кадра при различном количестве объектов на сцене. Параметры исследования:

– длина ребра куба — 50 единиц;

– радиус скругления — 10 единиц;

– количество аппроксимирующих рёбер — 5 штук;

– количество объектов сцены (кубов) изменяется от 10 до 960 включительно с шагом 50;

– количество измерений при каждом указанном количестве объектов — 20.

В таблице 4.1 представлены результаты исследования.

Таблица 4.1 – Зависимость времени отрисовки кадра от количества объектов (кубов) на сцене

№	**количество объектов (кубов), шт**	**процессорное время отрисовки (без теней), мс**	**процессорное время отрисовки (с тенями), мс**
1	10	11	19
2	60	43	72
3	110	77	130
4	160	103	162
5	210	134	196
6	260	161	227
7	310	184	259
8	360	214	292
9	410	235	324
10	460	266	428
11	510	327	525
12	560	317	485
13	610	343	514
14	660	365	544
15	710	412	578
16	760	423	622
17	810	452	622
18	860	496	656
19	910	552	703
20	960	591	729

С помощью метода полиномиальной регрессии на основе данных измерений были получены аппроксимирующие полиномы первой степени:

$$\begin{aligned} f(x) &= 0.57x + 6.19 \\ f_{shadow}(x) &= 0.75x + 37.20 \end{aligned} \tag{4.1}$$

для отрисовки без теней и с тенями соответственно. На рисунке 4.1 данные таблицы 4.1 и аппроксимирующие полиномы первой степени отображены графически.

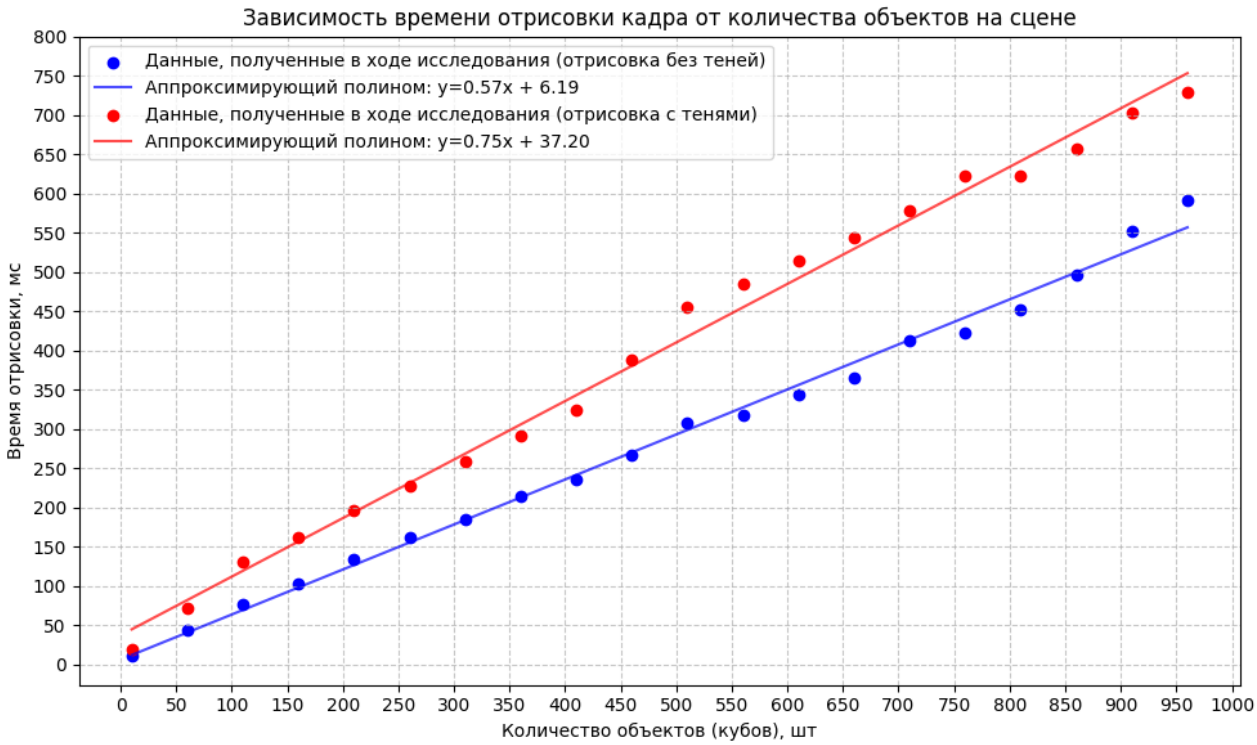

Рисунок 4.1 – Зависимость времени отрисовки кадра от количества объектов (кубов) на сцене

4.3 Исследование №2

Во втором исследовании проводилось исследование зависимости процессорного времени отрисовки кадра от количества аппроксимирующих рёбер. Параметры исследования:

– длина ребра куба — 50 единиц;

– радиус скругления — 10 единиц;

– количество аппроксимирующих рёбер изменяется от 2 до 32 с шагом 2, количество аппроксимирующих для всех объектов одинаково;

– количество объектов сцены (кубов) — 100 единиц;

– количество измерений при каждом указанном размере — 20.

В таблице 4.2 представлены результаты исследования.

Таблица 4.2 – Зависимость процессорного времени отрисовки кадра от количества аппроксимирующих рёбер

№	**Количество аппроксимирующих рёбер, шт**	**процессорное время отрисовки, мс**
1	2	642
2	4	707
3	6	798
4	8	921
5	10	1084
6	12	1280
7	14	1545
8	16	1883
9	18	2321
10	20	2832
11	22	3410
12	24	4069
13	26	4822
14	28	5646
15	30	6560
16	32	7569

С помощью метода полиномиальной регрессии на основе данных измерений был получен аппроксимирующий полином второй степени:

$$f(x) = 8.93x^2 - 79.95x + 899.19 \tag{4.2}$$

На рисунке 4.2 данные таблицы 4.2 и аппроксимирующий полином второй степени отображены графически.

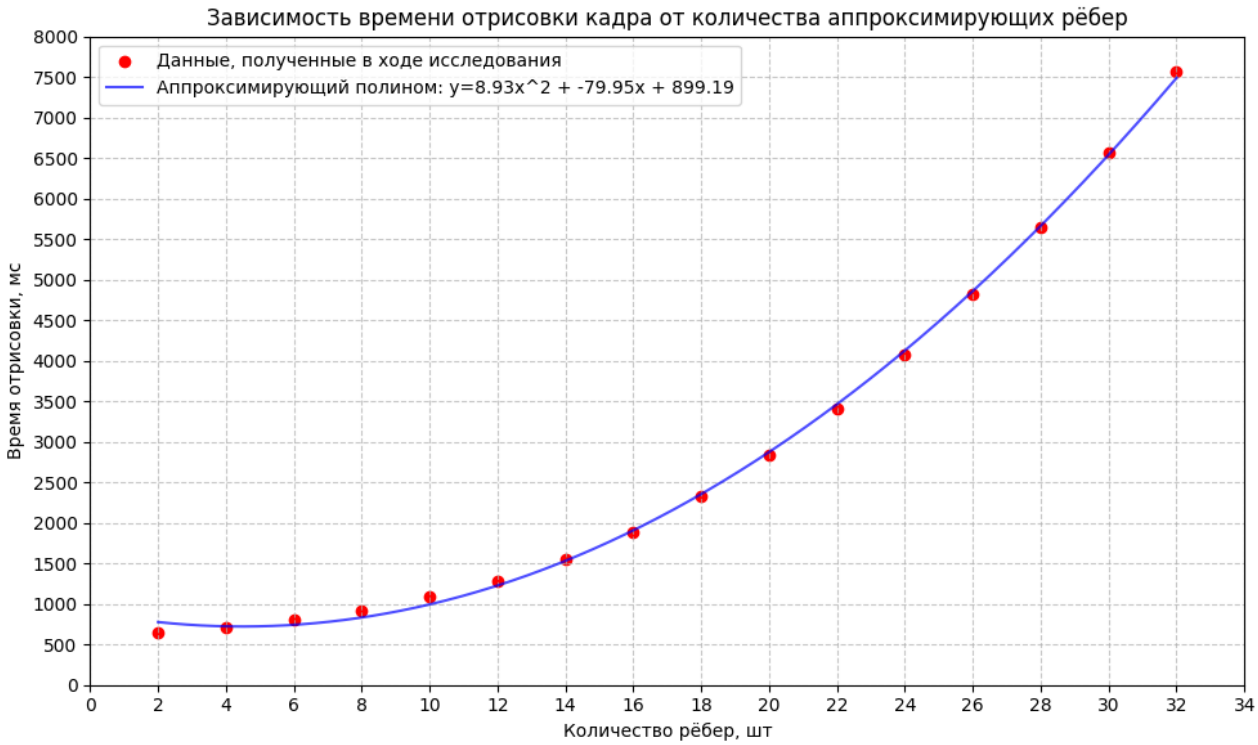

Рисунок 4.2 – Зависимость процессорного времени отрисовки кадра от количества аппроксимирующих рёбер

Вывод

В данном разделе были описаны технические характеристики ЭВМ, описаны условия проведения и результаты исследований затрат проссорного времени.

На основе результатов исследования №1 можно сделать вывод, что процессорное время отрисовки кадра с помощью алгоритма, использующего z-буфер, линейно зависит от количества объектов сцены как с отрисовкой теней, так и без неё.

На основе результатов исследования №2 можно сделать вывод, что процессорное время отрисовки кадра с помощью алгоритма, использующего z-буфер, квадратично зависит от количества аппроксимирующих рёбер.

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы были выполнены следующие задачи:

- формализованы объекты сцены;
- было рассмотрено несколько известных алгоритмов удаления невидимых линий и поверхностей, был выбран алгоритм, использующий z-буфер, как наиболее подходящий;
- спроектированы выбранные алгоритмы;
- в качестве средств реализации был выбран язык программирования C++ и библиотека Qt6;
- была выполнена программная реализация выбранных алгоритмов;
- разработан графический интерфейс программы;
- проведены исследования зависимостей процессорного времени отрисовки кадра от количества объектов на сцене и количества аппроксимирующих граней; первое исследование показало линейную зависимость, второе — квадратичную.

Все поставленные задачи выполнены. Цель курсовой работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Голомб С. В.* Полимино / пер. В. Фирсов. — Москва : Мир, 1975.
2. *Алексеев Ю. Е.*, *Куров А. В.* Компьютерная графика в среде MS VS C++: учебное пособие. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2017.
3. *Грегори Д.* Игровой движок. Программирование и внутреннее устройство. Третье издание. — СПб. : Питер, 2021. — (Для профессионалов).
4. *Доновaн Т.* Играй! История видеоигр / пер. И. Воронина. — М : Белое Яблоко, 2014.
5. *Никулин Е. А.* Компьютерная геометрия и алгоритмы машинной графики. — СПб. : БХВ-Петербург, 2003.
6. *Иванов В. П.*, *Батраков А. С.* Трехмерная компьютерная графика / под ред. Г. М. Полищук. — М. : Радио и связь, 1995.
7. *Перемитина Т. О.* Компьютерная графика: учебное пособие. — Томск : Эль Контент, 2012.
8. *Фоли Д.*, *Дэм А. вэн.* Основы интерактивной машинной графики: в 2-х книгах. Кн. 2 / пер. пер. с англ. — М. : Мир, 1985.
9. *Боресков А. В.* Программирование компьютерной графики. Современный OpenGL. — Москва : ДМК Пресс, 2019. — иллюстрации.
10. *Роджерс Д.* Алгоритмические основы машинной графики: пер. с англ. — М. : Мир, 1989.
11. *Шикин А. В.*, *Боресков А. В.* Компьютерная графика. Полигональные модели. — Москва : ДИАЛОГ-МИФИ, 2001.
12. *Гамбетта Г.* Компьютерная графика. Рейтрейсинг и растеризация. — СПб. : Питер, 2022.
13. Qt 6 Documentation. — Режим доступа: `https://doc.qt.io/qt-6/` (дата обращения: 20.12.2024).
14. Intel Core i7-8700K Processor [Электронный ресурс]. — Режим доступа: `https://ark.intel.com/content/www/us/en/ark/products/126775/intel-core-i78700k-processor-12m-cache-up-to-4-70-ghz.html` (дата обращения: 20.12.2024).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе состоит из 16 слайдов.