РЕФЕРАТ

Научно-исследовательская работа 48 с., 17 рис., 0 табл., 17 ист., 1 прил.

ЯДРО LINUX, СЕТЕВАЯ ПОДСИСТЕМА, СЕТЕВОЙ МОНИТОРИНГ, МОДИФИКАЦИЯ КОДА ЯДРА, ЗОНДИРОВАНИЕ ЯДРА, ТОЧКИ ТРАССИРОВКИ, EBPF, СЕТЕВОЙ СТЕК, FTRACE, KPROBE.

Объект исследования — сетевая подсистема ядра Linux.

Целью работы является провести анализ существующих средств мониторинга сетевой подсистемы ядра ОС Linux. Поставленная цель достигается путем рассмотрения и классификации существующих и применяемых методов сетевого мониторинга ядра Linux.

В данной работе проводились изучение принципов работы сетевой подсистемы, средств и подсистем сетевого мониторинга ядра Linux, а также сравнение и анализ методов сетевого мониторинга, рассматриваемых на примере мониторинга пути сетевого пакета в сетевой подсистеме ядра Linux.

Результат данной работы показал, что каждый метод сетевого мониторинга имеет ряд преимуществ и недостаткам и применим в зависимости от требований и ограничений к задаче.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В текущей расчетно-пояснительной записке применяется следующие сокращения и обозначения.

ОС — Операционная система.

ВВЕДЕНИЕ

В настоящее время компьютерное графическое моделирование различных объектов приобрело большую популярность и является неотъемлемой частью человеческой жизни. Она используется повсеместно: для наглядного отображения данных, в компьютерных играх, в кино для создания эффектов [1].

Компьютерная графика — собой совокупность методов и способов преобразования информации в графическое представление при помощи ЭВМ. Данная область предоставляет широкий спектр алгоритмов: от воссоздания художественных эффектов в компьютерных играх до построения трехмерных объектов при моделировании сложных технологических продуктов. Но алгоритмы компьютерной графики ресурсозатратны — чем более качественное изображение требуется получить, тем больше накладных ресурсов (времени и памяти) тратится на его синтез. Это самая главная проблема при создании реалистичных изображений и динамических сцен.

Целью данного курсового проекта является разработка редактора композиции, состоящих из многогранников, геометрические и спектральные характеристики которые задаются пользователем. Для формировании более полного представления о полученной сцене должны присутствовать возможности передвижение камеры и источника света, и изменение спектральных характеристик источника света.

Для достижения поставленной цели необходимо решить следующие задачи:

— анализ существующих алгоритмов компьютерной графики, использующие для создание реалистичной модели и трехмерной сцены;
— выбрать наиболее подходящих алгоритмов для решения поставленной задачи;
— проектирование архитектуры и графического интерфейса программы;

— выбрать средства реализации программного обеспечения;
— разработка программного обеспечения и реализация выбранных алгоритмов и структур данных;
— провести замеры временных характеристик разработанного программного обеспечения.

1 Аналитическая часть

В данной части проводится анализ объектов сцены и существующих алгоритмов построения изображений и выбор более подходящих алгоритмов для дальнейшего использования.

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из определенного набора объектов.

1) Направленный источник света представляет собой вектор направления света и принимает ортогональную проекцию визуализируемой сцены из своего положения с некоторым ограниченным положением. В зависимости от расположения источника и направления распространения лучей света, определяется тень от объекта, расположенных на сцене. Цвет свечения описывается через RGB параметры.
2) Объекты сцены – примитивные многогранники, расположенные в пространстве сцены. Каждая модель многогранника представляет собой набор граней, описываемых точками в пространстве, которые соединены ребрами и задается параметрическими (радиус нижнего и верхнего основания, количество боковых граней и высота) и спектральными характеристиками (зеркальный коэффициент, коэффициент блеска и цвет) для построения.
3) Камера характеризуется своим пространственным положением и направлением просмотра.

1.2 Способы описания трехмерных геометрических моделей на сцене

В компьютерной графики для описание трехмерных геометрических объектов существует три типа моделей: каркасная, поверхностная и твердотельная [2]. Использование моделей позволяет правильно отображать форму и размеры объектов сцены.

— *Каркасная модель* — простейший вид моделей, который содержит минимум информации только о вершинах и ребрах объектов. Это моделирование самого низкого уровня, которое имеет ряд серьезных ограничений, большинство из которых возникает из-за недостатка информации о гранях, которые заключены между ребрами. Невозможно выделить внутреннюю и внешнюю область изображения твердого объемного тела. Однако каркасная модель требует меньше памяти и затрат времени на построение, что достаточно пригодна для решения задач, не требующие информации о поверхности объекта (например, если в объекте есть отверстия). Проблемой этой модели заключается в том, что она не позволяет отличить видимые грани от невидимых. Операции по удалению невидимых линий можно выполнить только вручную с применением команд редактирования каждой отдельной линии, но результат работы нарушает каркасную конструкцию, по причине того что линии невидимы в одном случае и видимы в другом. Кроме того, каркасная модель не несет информации о поверхностях, ограничивающих форму, что обуславливает невозможность обнаружения нежелательных взаимодействий между гранями объекта [3].

— *Поверхностная модель* часто используется в компьютерной графике, кроме содержание информации о вершинах и ребрах, содержит еще информацию о поверхности. При построении поверхностной модели пред-

полагается, что технические объекты ограничены поверхностями, которые отделяют их от окружающей среды. Недостатком поверхностной модели является отсутствие информации о том, с какой стороны поверхности находится материал.

— *Твердотельная модель* отличается от поверхностной тем, что в данной модели к информации о поверхностях добавляется информация о том, с какой стороны расположен материал. Это достигается путем указания направления внутренней нормали.

Для решения поставленной задачи не подойдет каркасная модель, так как такое представление будет приводить к неправильному восприятию форм объекта. Твердотельная модель также не подойдет, так как по поставленной задачи нет необходимости знать из какого материала будет выполнен тот или иной объект и с какой стороны расположен материал. Поэтому выбор остается лишь поверхностной модели.

1.3 Способы задания поверхностных моделей

Поверхностная модель задается следующими способами [3, **?**].

— *Аналитический (параметрический) способ* характеризуется описанием модели объекта, которое доступно в неявной форме,то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра, что в свою очередь довольно удобно при просчете поверхностей вращения.

— *Полигональная сетка* характеризуется совокупностью вершин, ребер и граней, определяющих форму объекта в трехмерном пространстве.

Рассмотрим существующие способы хранения информации о полигональной сетке.

Вершинное представление описывает объект множество вершин, соеди-

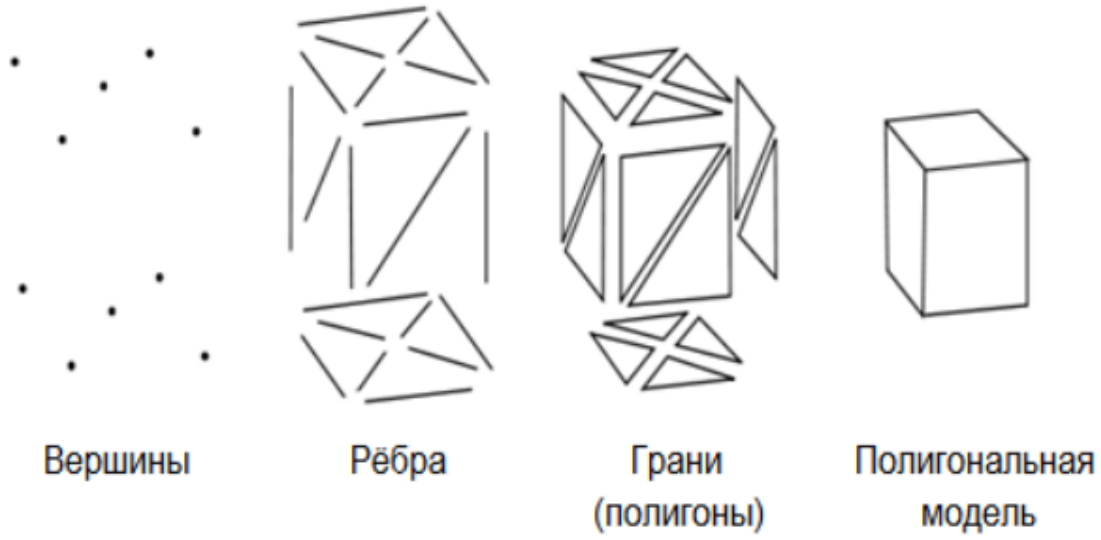

Рисунок 1 – Полигональная сетка

ненных с другими вершинами (вершины, которые указывают на другие вершины). Информация о ребрах и гранях неявно присутствует в представлении из-за чего для восстановления исходного тела необходимо обойти все вершины и составить списки граней. Кроме того, операции с ребрами и гранями выполнить нелегко [2]. Тем не менее из-за простоты представления дает возможность множество операций над сеткой. Хранение информации о сетке требует не так много памяти по сравнению с другими способами. На рисунке 2 представлен пример вершинного представления, рассмотренный на кубе.

Список вершин

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,0	v5 v6 v7 v8
v9	.5,.5,1	v0 v1 v2 v3

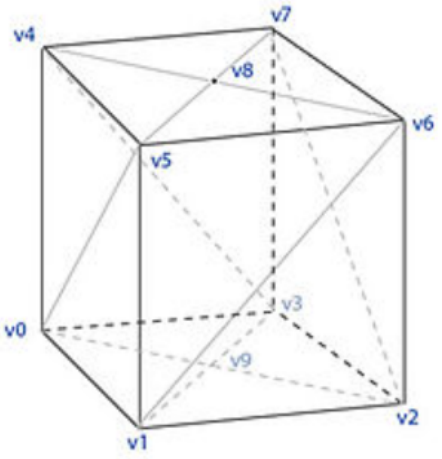

Рисунок 2 – Вершинное представление

Представление называемый списком граней представляет объект не только множеством вершин, но граней. В отличие от предыдущего способа, вершины и граны определены явно, благодаря чему нахождение соседних вершин и граней довольно проста и поиск соседних граней и вершин занимает постоянное время [2]. Также список вершин содержит список граней, связанных с каждой вершиной. Однако ребра неявны, поэтому поиск все равно необходим, чтобы найти все грани, окружающие данную грань. Другие динамические операции, такие как разделение или слияние граней, также сложны с сетками граней и вершин. Пример представления список граней представлен на рисунке 3.

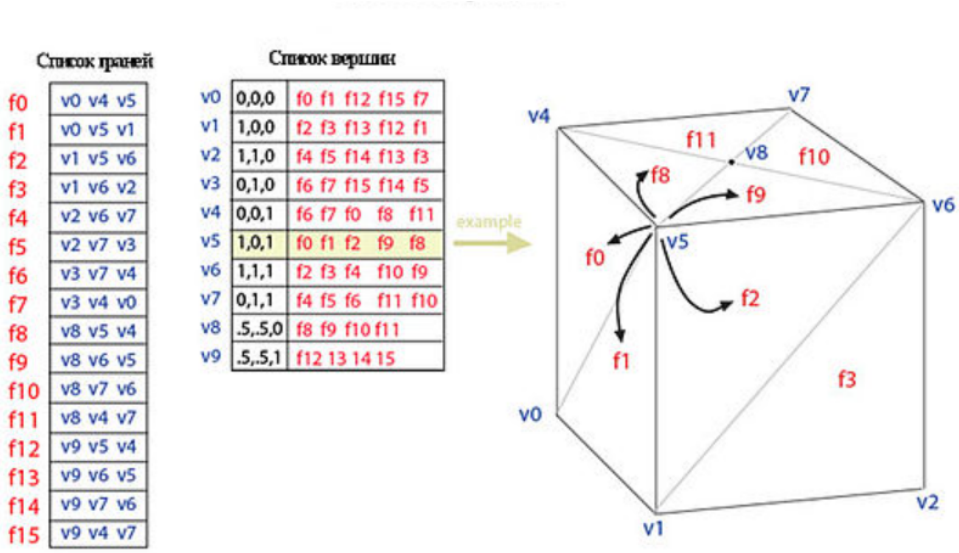

Рисунок 3 – Список граней

Представление называемый таблицей углов представляет специальной таблицей, хранящей вершины, обход которых неявно задает полигоны. Такое представление более компактно и более производительно для нахождение полигонов, но в связи с тем, что вершины присутствуют в описании нескольких углов, операции по их изменению медленны. Также угловые таблицы представляют сетки не полностью, из-за чего их требуется несколько.

«Крылатое» представление задает точки, каждая из которых указывает на две вершины, две грани и четыре ребра, которые ее касаются, благодаря

чему обход поверхностей выполняется за постоянное время. Однако метод очень требует памяти и изменения геометрических характеристик приводит к формированию списка индексов граней. Данный способ полезен для определения столкновений объектов. Пример представлен на рисунке 4.

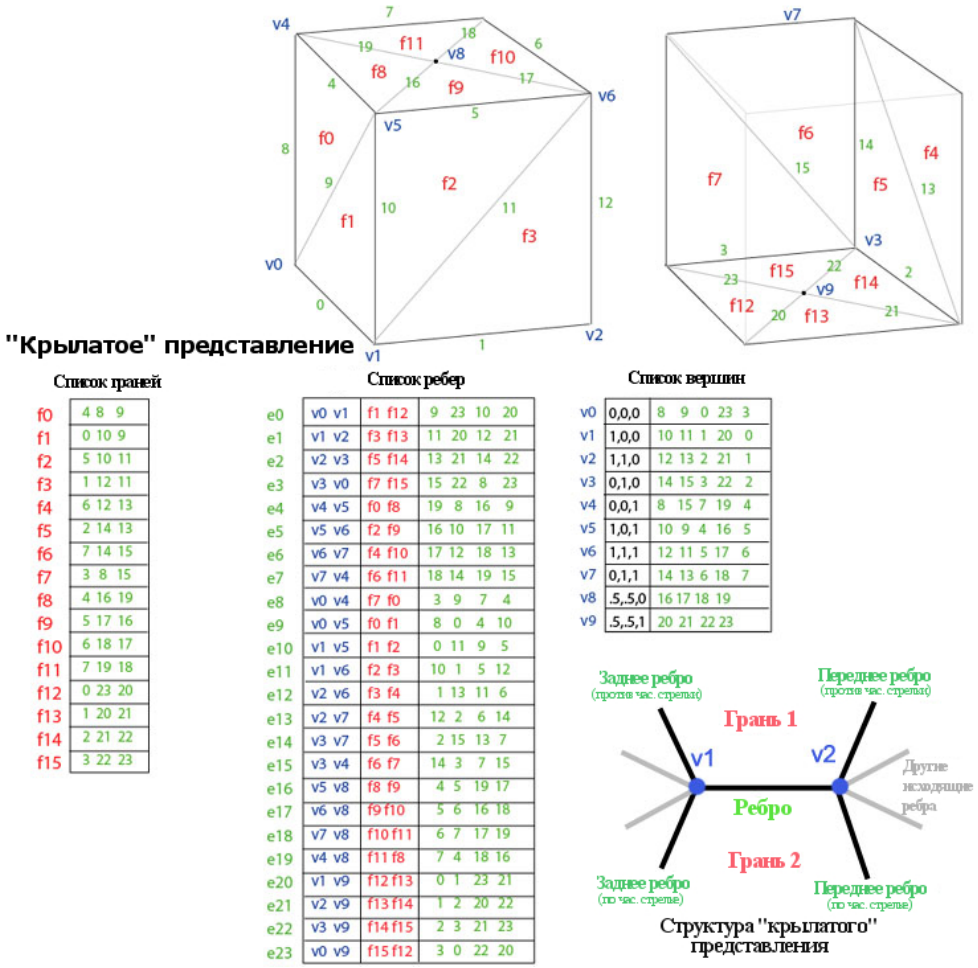

Рисунок 4 – «Крылатое» представление

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

1.4 Анализ алгоритмов удаления невидимых линий и поверхностей

Основной задачей при построении реалистичного изображения является задача удаления объектов или их частей, которые перекрываются другими объектами, то есть являются невидимыми с точки зрения наблюдателя. Выделяют две группы алгоритмов для ее решения:

— Алгоритмы, работающие в объектном пространстве. Данные алгоритмы имеют привязку к мировой или физической системе координат. Получаемые результаты ограничиваются только точностью вычислений, однако требуют большого объема вычислений, зависящего от требуемой точности и сложности поступающей на вход сцены. В эту группу входят алгоритм Робертса, алгоритм со списком приоритетов и т.д. [3]

— Алгоритмы, работающие в пространстве изображения. Данные алгоритмы предполагают привязку к системе координат экрана или картинной плоскости, на которую производится проецирование изображаемых объектов. Объем требуемых вычислений значительно меньше, чем у алгоритмов первой группы, и зависит от разрешающей способности экрана и количества объектов на сцене. Основными представителями данной группы являются алгоритм Варнака, алгоритм Z-буфера и алгоритм трассировки лучей [3].

1.4.1 Алгоритм Робертса

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами, и выполняется в 3 этапа [3].

Первый этап — подготовка исходной матрицы V, которая задает инфор-

мацию о каждой фигуре. Размерность матрицы – $4 * n$, где n – количество граней тела. Каждый столбец матрицы представляет собой четыре коэффициента уравнения плоскости $ax+by+cz+d = 0$, проходящей через очередную грань. Таким образом, матрица тела будет представлена в следующем виде:

$$V = \begin{pmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{pmatrix} \tag{1}$$

Матрица тела должна быть сформирована корректно, то есть любая точка, расположенная внутри тела, должна располагаться по положительную сторону от каждой грани тела. В случае, если для очередной грани условие не выполняется, соответствующий столбец матрицы надо умножить на -1. Для проведения проверки следует взять точку, расположенную внутри тела. Координаты такой точки можно получить путем усреднения координат всех вершин тела.

Второй этап — удаления ребер, экранируемых самим телом, где рассматривается вектор взгляда $E = \{0, 0, -1, 0\}$. Для определения невидимых граней достаточно умножить вектор E на матрицу тела V. Отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

Третий этап — удаление невидимых ребер, экранируемых другими телами сцены. Для определения невидимых точек ребра требуется построить луч, соединяющий точку наблюдения с точкой на ребре. Точка будет невидимой, если луч на своем пути встречает в качестве преграды рассматриваемое тело. Если тело является преградой, то луч должен пройти через тело. Если луч проходит через тело, то он находится по положительную сторону от каждой грани тела.

В данном алгоритме главным недостатком является его вычислитель-

ная трудоемкость равная $O(n^2)$, где n — количество объектов на сцене. Также все тела на сцене должны быть выпуклыми, что приводит к дополнительным проверкам. Однако, работа в объектном пространстве и высокая сложность вычислений обеспечивает высокую точность результата.

1.4.2 Алгоритм Варнока

Алгоритм Варнока работает в пространстве изображения и позволяет определить, какие грани или части граней объектов сцены видимы, а какие заслонены другими объектами [3, 4]. Алгоритм прелагает разбиение области изображения на более мелки окна, и для каждого такого окна определяются связанные с ней многоугольники и те, видимость которых можно определить, изображаются на сцене. На рисунке 5 представлен пример разбиние с помощью алгоритма Варнака.

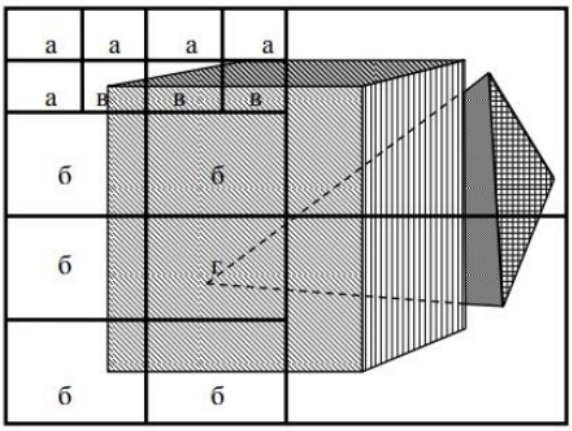

Рисунок 5 – Пример разбиения алгоритмом Варнака

В качестве граней обычно выступают выпуклые многоугольники, алгоритмы работы с ними эффективнее, чем с произвольными многоугольниками. Окно, в котором необходимо отобразить сцену, должно быть прямоугольным. Алгоритм работает рекурсивно, что является его главным недостатком, на каждом шаге анализируется видимость граней и, если нельзя «легко» опре-

делить видимость, окно делится на 4 части и анализ повторяется отдельно для каждой из частей.

1.4.3 Алгоритм, использующий Z-буфер

Алгоритм, использующий z-буфер, является одним из самых простых и широко используемых и работает в пространстве изображения. Идея z-буфер является простым обобщением идеи о буфере кадра [3]. Используется два буфера:

— буфер кадра, который используется для запоминания атрибутов (интенсивности) каждого пиксела;

— z-буфер — отдельный буфер глубины, используемый для запоминания координаты z каждого пиксела.

Первоначально в z-буфере находятся минимально возможные значения z, а в буфере кадра располагаются пиксели, описывающие фон. В процессе работы глубина z каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в z-буфер. Если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то он заносится в буфер кадра и происходит корректировка z-буфера. Затем каждый многоугольник преобразуется в растровую форму и записывается в буфер кадра, при этом не производится начального упорядочения В противном случае никаких действий не производится [3]. На рисунке 6 представлена пример работы алгоритма Z-буфера.

Основным преимущество алгоритма является простота реализации. Трудоемкость алгоритма увеличивается линейно в зависимости от количества объектов на сцене. Особая эффективность по времени достигается за счет того, что нет необходимости в сортировки объектов. Это является большим плюсом, так как на сцене может быть большое количество объектов. Однако

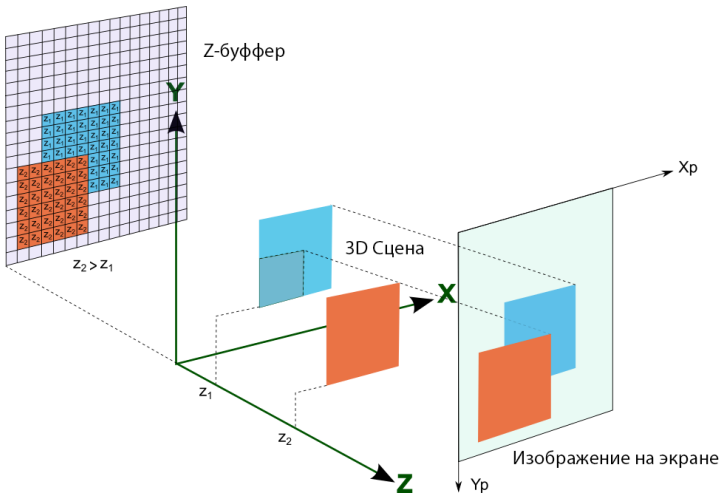

Рисунок 6 – Работа алгоритма Z-буфера

алгоритм очень требователен к памяти из-за необходимости хранить информацию о каждом пикселе и выделение памяти под два буфера. Реализация эффектов прозрачности и устранения лестничного эффекта осложнена.

1.4.4 Алгоритм прямой трассировки лучей

Главная идея, лежащая в основе алгоритма трассировки лучей, заключается в том, что наблюдатель видит любой объект посредством испускаемого неким источником света, который падает на этот объект и затем согласно законам оптики некоторым путем доходит до глаза наблюдателя. В методе прямой трассировки предполагает построения траекторий лучей от всех источников света ко всем точкам всех объектов сцены, отражаются и преломляются или проходят сквозь него и в результате достигает наблюдателя. Данные лучи называются первичными. Если объект не является отражающим или прозрачным, то траектория луча на этой точке обрывается [3, 5].

Основным недостатком алгоритма является излишне большое число рассматриваемых лучей, приводящее к существенным затратам вычислительных мощностей, так как лишь малая часть лучей достигает точки наблюдения. Данный алгоритм подходит для генерации статических сцен и модели-

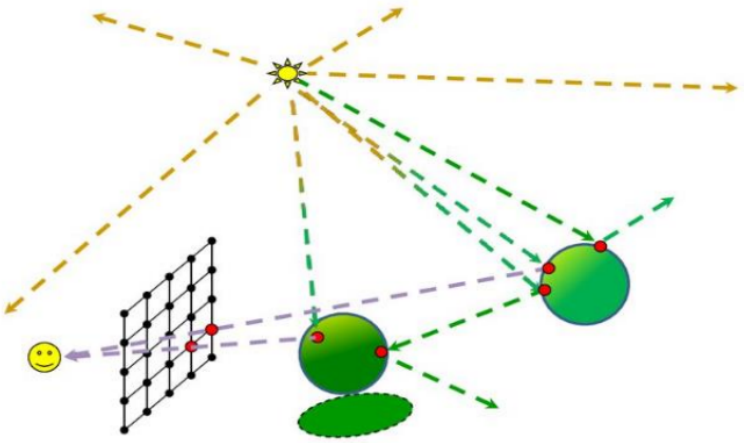

Рисунок 7 – Работа алгоритма прямой трассировки лучей

рования зеркального отражения, а так же других оптических эффектов как отражение, преломление и т.д.

1.4.5 Алгоритм обратной трассировки лучей

Наблюдатель видит объект посредством испускаемого источником света, который падает на этот объект и согласно законам оптики некоторым путем доходит до глаза наблюдателя. Отслеживать пути лучей от источника к наблюдателю неэффективно с точки зрения вычислений, поэтому наилучшим способом будет отслеживание путей в обратном направлении, то есть от наблюдателя к объекту. Лучи испускаются из камеры, пронизывая каждый пиксель сцены. После этого определяют пересечение первичного луча с объектами сцены. Первичный луч – луч, выпущенный из камеры. Если оно было установлено, вычисляют интенсивность пиксела, учитывая положения источников света [3, 5].

Считается, что наблюдатель расположен на положительной полуоси z в бесконечности, поэтому все световые лучи параллельны оси z. В ходе работы испускаются лучи от наблюдателя и ищутся пересечения луча и всех объектов сцены. В результате пересечение с максимальным значением z является

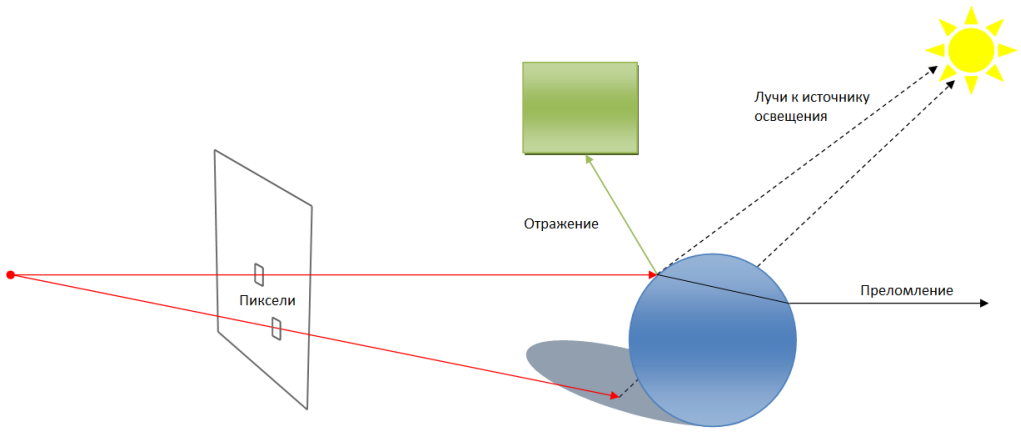

Рисунок 8 – Работа алгоритма обратной трассировки лучей

видимой частью поверхности и атрибуты данного объекта используются для определения характеристик пиксела, через центр которого проходит данный световой луч. Для расчета эффектов освещения сцены проводятся вторичные лучи от точек пересечения ко всем источникам света. Если на пути этих лучей встречается непрозрачное тело, значит, данная точка находится в тени.

Несмотря на более высокую эффективность алгоритма в сравнении с прямой трассировкой лучей, данный алгоритм считается достаточно медленным, так как в нем происходит точный расчет сложных аналитических выражений для нахождения пересечения с рассматриваемыми объектами [4].

1.4.6 Выбор оптимального алгоритма

При реализации редактора композиций из графических примитивов многогранников необходимо обеспечить плавную смену кадров при перемещении камеры и изменении объектов, поэтому алгоритм должен иметь минимальную зависимость трудоемкости алгоритма от числа объектов на сцене и использование рекурсивных вызовов. Также требуется учесть ограничение разработки курсового проекта, поэтому требуется простота реализации алгоритма. Задействования дополнительной памяти под буферы не является

весомым критерием и может быть опущен.

Таким образом, подходящий алгоритм является алгоритм с Z-буфером, который подходит для решения создания сцены с различным количеством объектов, что позволит иметь динамическую сцену. Данный алгоритм имеет линейную зависимость от числа объектом, что приведет к оптимальной работе программы. Также он простой в реализации и используется в большинстве графических движков, что приведет к быстрой скорости реализации алгоритма.

1.5 Анализ алгоритмов закраски

Методы закраски используются для затенения полигонов модели в условиях некоторой сцены, имеющей источники освещения. Учитывая взаимное положение рассматриваемого полигона и источника света, можно найти уровень освещенности по закону Ламберта [3]:

$$I_\alpha = I_0 \cdot \cos(\alpha), \tag{2}$$

где I_α - уровень освещенности в рассматриваемой точке, I_0 - максимальный уровень освещенности, а α - угол между вектором нормали к плоскости и вектором, направленным от рассматриваемой точки к источнику освещения.

Существует три основных алгоритма, позволяющих закрасить полигональную модель.

1.5.1 Простая закраска

Вся грань закрашивается одним уровнем интенсивности, который высчитывается по закону Ламберта. При данной закраске все плоскости (в том

числе и те, что аппроксимируют фигуры вращения), будут закрашены однотонно, что в случае с фигурами вращения будет давать ложные ребра [3].

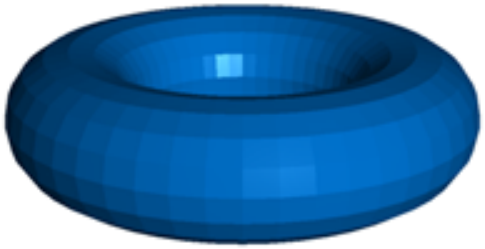

Рисунок 9 – Результат работы алгоритма простой закраски

Данный метод закраски обладает большим быстродействием, однако все пиксели грани имеют одинаковую интенсивность, и сцена выглядит нереалистично. Тем не менее, этот метод крайне прост в реализации и совершенно не требователен к ресурсам.

1.5.2 Закраска по Гуро

Метод Гуро устранить дискретность изменения интенсивности и создать иллюзию гладкой криволинейной поверхности. Он основан на интерполяции интенсивности.

Данный метод отличается от простой закраски тем, что разные точки грани закрашиваются с разными значениями интенсивности. Для это в каждой вершине грани находится вектор нормали и вычисляется значение интенсивности. Затем найденные значения интенсивности интерполируются по всем точкам примыкающих граней. На рисунке 10 представлен пример работы алгоритма закраски по Гуро.

Закрашивание граней по методу Гуро осуществляется в четыре этапа.

1) Вычисляются нормали к каждой грани.
2) Определяются нормали в вершинах. Нормаль в вершине определяется усреднением нормалей примыкающих граней.

3) На основе нормалей в вершинах вычисляются значения интенсивностей в вершинах согласно выбранной модели отражения света.
4) Закрашиваются полигоны граней цветом, соответствующим линейной интерполяции значений интенсивности в вершинах.

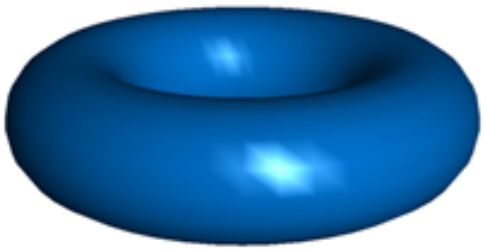

Рисунок 10 – Результат работы алгоритма закраски по Гуро

Метод Гуро применим для небольших граней, расположенных на значительном расстоянии от источника света. Если же размер грани большой, то расстояние от источника света до центра будет меньше, чем до вершин, и центр грани должен выглядеть ярче, чем ребра. Но из-за линейного закона, используемого в интерполяции, метод не позволяет это сделать, поэтому появляются участки с неестественной освещенностью

1.5.3 Закраска по Фонгу

Закраска Фонга по своей идее похожа на закраску Гуро, но ее отличие состоит в том, что в методе Гуро по всем точкам полигона интерполируется значения интенсивностей, а в методе Фонга – вектора нормалей, и с их помощью для каждой точки находится значение интенсивности [3]. На рисунке 11 представлен пример работы алгоритма закраски по Фонгу.

Этапы следующие.

1) Определяются нормали к граням.
2) По нормалям к граням определяются нормали в вершинах. В каждой точке закрашиваемой грани определяется интерполированный вектор

нормали.

3) По направлению векторов нормали определяется цвет точек грани в соответствии с выбранной моделью отражения света.

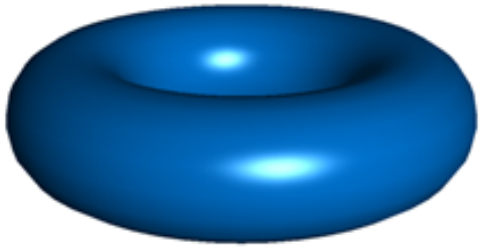

Рисунок 11 – Результат работы алгоритма закраски по Фонгу

Эта закраска требует больших вычислительных затрат, чем предыдущие, однако она позволяет достигнуть лучшей локальной аппроксимации кривизны поверхности и, следовательно, с ее помощью получается более реалистичное изображение.

1.5.4 Выбор алгоритма закраски

Алгоритм закраски Фонга требует большего числа вычислений по сравнению с другими, однако он дает наиболее реалистичное изображение, в частности зеркальных бликов. В данном курсовом проекте пользователь должен иметь возможность задавать спектральные характеристики объектов, в том числе коэффициенты зеркального отражения и блеска поверхности, поэтому для получения более реалистичной сцены используется метод закраски Фонга.

1.6 Анализ алгоритма построения теней

Для генерации теней в данном курсовом проекте используется метод теневых карт. Он основан на алгоритме Z-буфера: сначала камера устанавливается на позицию источника освещения и происходит построение сцены с занесением глубин в теневой буфер (карту теней). Все точки в этом буфере видимы с точки зрения источника, соответственно они освещены. После этого изображение строится уже с позиции наблюдателя, при этом для каждой точки буфера кадра расстояние до источника освещения сравнивается со значением из теневой карты: если это расстояние больше полученного значения из карты, то точка лежит в тени и отображается только с учетом рассеянного света. Иначе точка освещена и происходит расчет ее интенсивности со всеми составляющими освещения.

Так как используемый источник освещения является точечным, то есть излучающим свет во всех направлениях, то для генерации теней необходимо произвести построение сцены 6 раз для разных направлений камеры: вдоль и против осей OX, OY и OZ. В результате образуется так называемая кубическая карта теней, в которой содержатся значения глубин всех освещенных участков сцены.

Этот способ генерации теней требует значительного числа вычислений, однако они выполняются только при перестроении сцены, то есть при изменении положения камеры заново вычислять значения в карте теней не нужно. Таким образом метод кубической карты теней позволяет сохранить плавность подачи картинки при перемещении по сцене, поэтому используется в данном курсовом проекте.

1.7 Анализ моделей освещения

Все модели освещения делятся на два вида: глобальные и локальные. Глобальные модели учитывают возможности отражения и преломления света от объектов, не являющихся прямыми источниками освещения, поэтому они требуют значительных затрат. Например, прозрачные поверхности должны позволять видеть объекты за ними, на поверхностях возможно появление таких явлений, как рефлексы. Глобальные модели освещения основываются на том, что видимость и затенения связаны между собой: яркость точки поверхности определяется распределением яркости по всем остальным поверхностям, видимым из этой точки. Так можно моделировать шероховатость и свойства отражения реальных материалов, освещение многократно отраженным светом и связанные с ним цветовые эффекты [6].

Существуют более простые, локальные модели освещения, которые учитывают только свет от источника. Они вычисляют интенсивность, цвет и дальнейшее распределение света на поверхности, но не учитывают перенос света между объектами. Изображение формируется в результате отражения, падающего на поверхность объектов света, интенсивность и цвет которого необходимо рассчитать. Фоновое освещение определяет цвет объекта в тени или при отсутствии явных источников освещения. Его интенсивность постоянна, а также равномерно распределена по всему пространству.

Выделяют две основные модели локального освещения: модель Ламберта и модель Фонга.

1.7.1 Модель Ламберта

Модель Ламберта моделирует идеальное диффузное освещение. Считается, что свет при попадании на поверхность рассеивается равномерно во все стороны. На рисунке 12 показано, что согласно этой модели, освещенность в точке определяется только плотностью света в точке поверхности, а она линейно зависит от косинуса угла падения. При этом положение наблюдателя не имеет значения, т.к. диффузно отраженный свет рассеивается равномерно по всем направлениям [6].

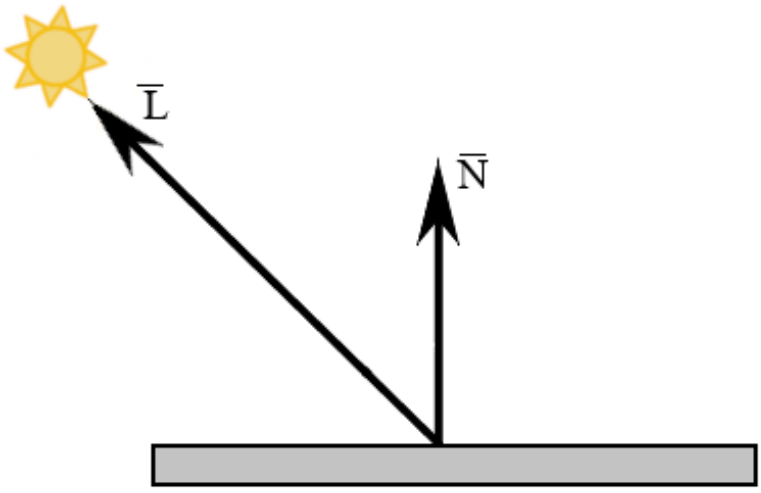

Рисунок 12 – Модель освещения Ламберта

Пусть:

— $\overrightarrow{L}$ — вектор от точки до источника;

— $\overrightarrow{N}$ — вектор нормали;

— I — результирующая интенсивность света в точке;

— I_0 — интенсивность источника;

— K_d — коэффициент диффузного освещения.

Формула расчета интенсивности имеет следующий вид:

$$I = I_0 \cdot K_d \cdot cos(\overrightarrow{L}, \overrightarrow{N}) = I_0 \cdot K_d \cdot (\overrightarrow{L}, \overrightarrow{N}) \quad (3)$$

Из формулы (5) следует главный недостаток модели Ламберта – одинаковая интенсивность во всех точках, принадлежащих одной грани.

Модель Ламберта является одной из самых простых моделей освещения. Данная модель очень часто используется в комбинации других моделей, практически в любой другой модели освещения можно выделить диффузную составляющую. Более-менее равномерная часть освещения (без присутствия какого-либо всплеска) как правило будет представляться моделью Ламберта с определенными характеристиками.

1.7.2 Модель Фонга

Модель Фонга – классическая модель освещения. Модель представляет собой комбинацию диффузной составляющей (модели Ламберта) и зеркальной составляющей и работает таким образом, что кроме равномерного освещения на материале может еще появляться блик. Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части как показано на рисунке 13.

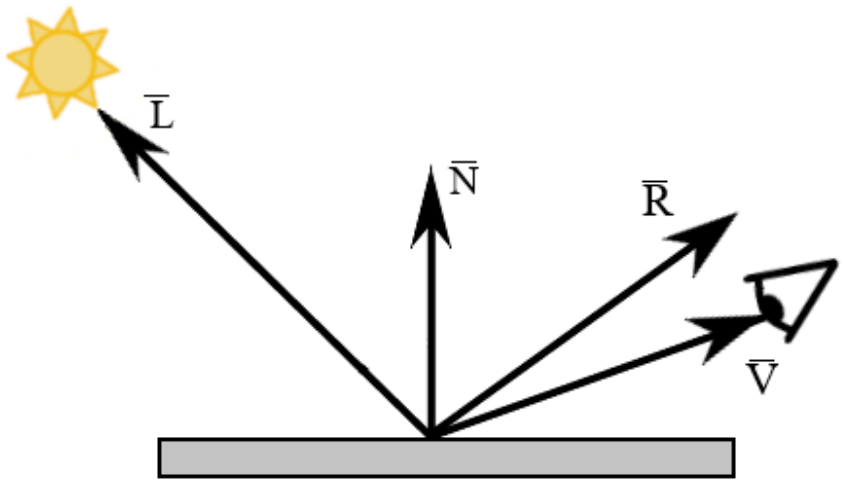

Рисунок 13 – Модель освещения Фонга

Отраженная составляющая освещенности в точке зависит от того, насколько близки направления на наблюдателя и отраженного луча. Также в модели освещения Фонга используется понятие рассеянного освещения – это константа, которая прибавляется к интенсивности в точке для придания сцене большей реалистичности.

Таким образом, согласно модели Фонга интенсивность к точке складывается будет рассчитываться по формуле:

$$I = I_a + I_d + I_s, \tag{4}$$

где I_a — диффузная составляющая, I_d — рассеянная составляющая, I_s — зеркальная составляющая.

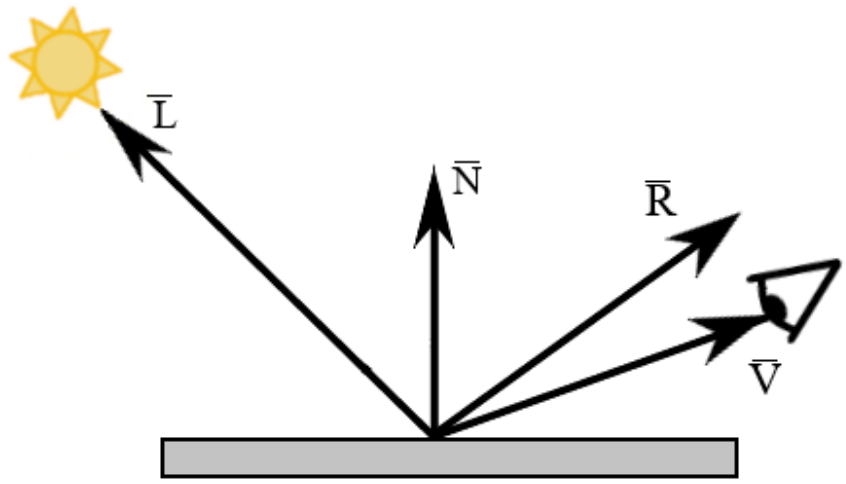

Рисунок 14 – Составляющие модели Фонга (слева направо: рассеянная, диффузная, зеркальная, суммарная)

Пусть:

— $\overrightarrow{R}$ — вектор отраженного луча;

— $\overrightarrow{V}$ — вектор от точки до наблюдателя;

— I_p — интенсивность рассеянного освещения;

— K_3 — коэффициент зеркального освещения;

— K_a — коэффициент рассеянного освещения;

— α — коэффициент блеска.

Формула для расчета интенсивности для модели Фонга имеет вид:

$$\begin{aligned} I = I_p \cdot K_a + I_0 \cdot K_d \cdot cos(\overrightarrow{L}, \overrightarrow{N}) + I_0 \cdot K_3 \cdot cos^{\alpha}(\overrightarrow{R}, \overrightarrow{V}) = \\ = I_p \cdot K_a + I_0 \cdot K_d \cdot (\overrightarrow{L}, \overrightarrow{N}) + I_0 \cdot K_3 \cdot (\overrightarrow{R}, \overrightarrow{V})^{\alpha} \end{aligned} \tag{5}$$

Вывод

Подведя итог и проанализировав все вышеописанные алгоритмы методов освещение, можно сделать вывод, что наилучшим алгоритмом для решения задачи будет метод Фонга в сочетании с закраской Фонга, так как он позволяет получить реалистичную картинку с эффектами отражения и преломления лучей.

2 Конструкторская часть

В данном разделе представлены требования к программному обеспечению, рассмотрены алгоритмы, выбранные для построения сцены

2.1 Требования к программному обеспечению

Программа должна предоставлять графический интерфейс с функционалом:

— задать параметрические и спектральные характеристики, добавляемой модели многогранника;

— изменение положение модели многогранника в пространстве;

— изменение положение камеры и направления ее взгляда в пространстве;

— изменение характеристик модели многогранника.

Разработанное программное обеспечение должно соответствовать следующим требованиям:

— источник света создается при запуске программы;

— нельзя иметь больше 1 камеры в пространстве;

— программа должна корректно обрабатывать ввод некорректных данных;

— все объекты создаются путем ввода его параметрических и спектральных характеристик.

2.2 Описание структур данных

Для формирования общего алгоритма синтеза изображения в данной программе, необходимо ввести определения использующихся в ней структур данных.

1) Сцена представляет собой список с произвольным числом моделей, объект камеры и объект источника освещения.
2) Модель многогранника включает в себя следующие данные:
   — массив вершин фигуры;
   — массив полигонов фигуры;
   — массив векторов нормалей к вершинам;
   — коэффициенты отражения и блеска поверхности;
   — цвет поверхности;
   — матрица аффинных преобразований.
3) Камера содержит:
   — положение в пространстве;
   — значения углов тангажа и рыскания,;
   — систему координат камеры, задаваемую тремя ортогональными векторами;
   — угол обзора и соотношение сторон экрана;
   — цвет поверхности;
   — границы пирамиды видимости.
4) Источник освещения включает: положение в пространстве и цвет источника.

2.3 Общий алгоритм построения изображения

Алгоритм генерации изображения представлен на рисунке 15. На вход подается геометрические характеристики моделей, камеры, источника, освещения, спектральные характеристики моделей и источника освещения. На выход выдается результат построения буфера кадра сцены.

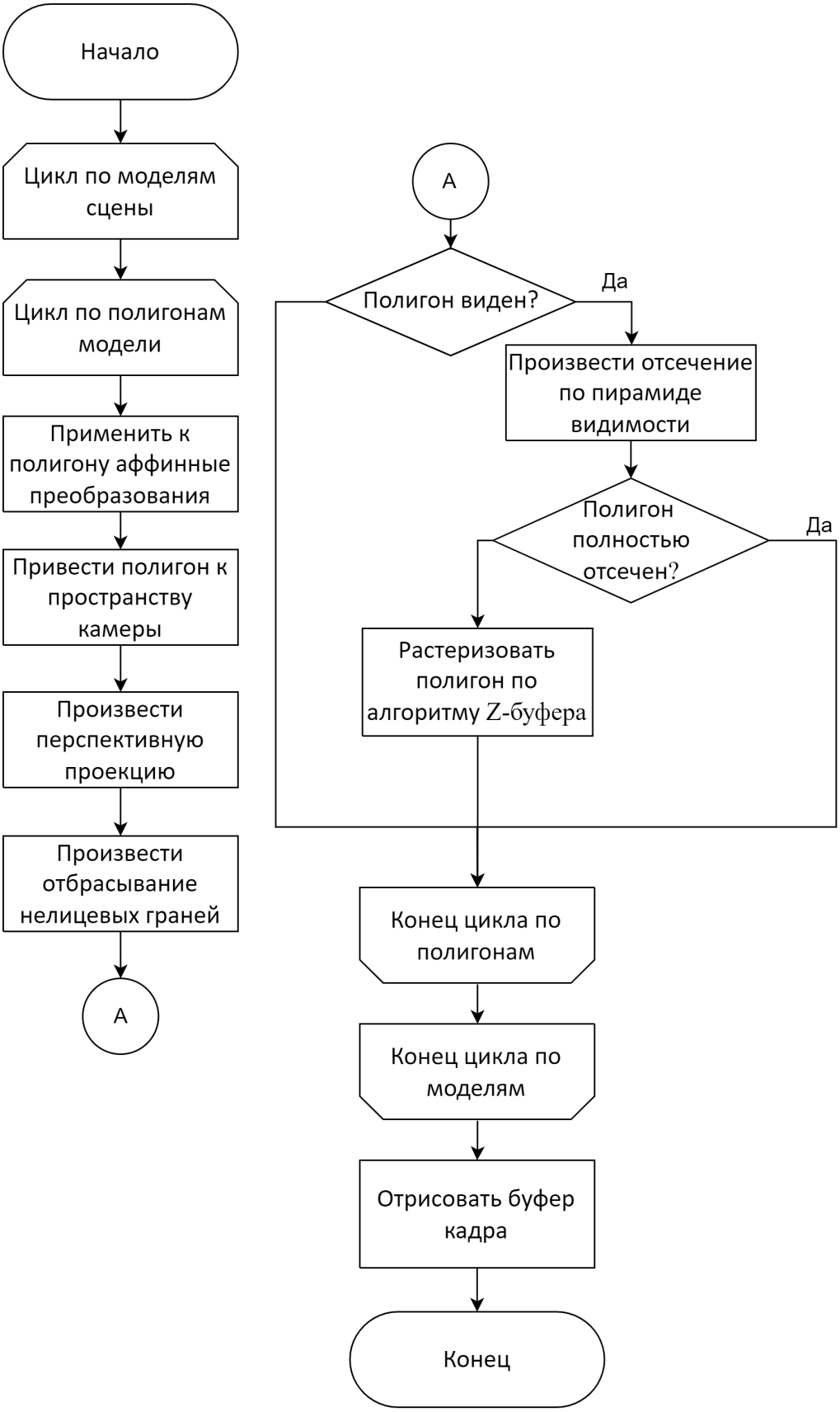

Рисунок 15 – Общая схема алгоритма синтеза изображения

2.4 Аффинные преобразования

В представленном алгоритме синтеза изображения первым этапом преобразования полигона пред его растеризацией является переход модели в мировое пространство. Такое действие осуществляется с помощью матриц аффинных преобразований [7]. В данном курсовом проекте над объектами возможно произвести следующие операции.

— Поворот вокруг координатных осей описывается углом α и осью вращения. Матрица поворота имеет вид:

– вокруг оси OX:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

– вокруг оси OY:

$$\begin{pmatrix} cos\alpha & 0 & -sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ sin\alpha & 0 & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

– вокруг оси OZ:

$$\begin{pmatrix} cos\alpha & sin\alpha & 0 & 0 \\ -sin\alpha & cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

— Перенос в трехмерном пространстве задается значения вдоль координатных осей OX, OY, OZ — dx, dy, dz соответственно. Матрица переноса

имеет вид:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{pmatrix} \tag{9}$$

2.5 Приведение к пространству камеры

Для перемещения по сцене используется камера, задаваемая точкой положения в пространстве, пирамидой видимости и собственной системой координат, которая состоит из трех ортогональных векторов.

Обозначим:

— P — положение камеры;

— D — вектор взгляда;

— U — вектор вверх;

— R — вектор вправо.

Для перехода в пространство камеры выполняется в два этапа, указанные далее.

1) Перенос полигона в отрицательную стороны от камеры на расстояние P с помощью матрицы переноса [8]:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -Px & -Py & -Pz & 1 \end{pmatrix} \tag{10}$$

2) Преобразование полигона к системе координат камеры при помощи

матрицы поворота [8]:

$$\begin{pmatrix} Rx & Ux & Dx & 0 \\ Ry & Uy & Dy & 0 \\ Rz & Uz & Dz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

Управление камеры производится с помощью изменения углом Эйлера. Обозначим α — угол поворота вокруг оси OУ (тангаж) и β — угол поворота вокруг оси OX (рыскание). Тогда координаты вектора направления камеры можно вычислить по следующим формулам:

$$D_x = cos(\alpha) \cdot cos(\beta) \tag{12}$$

$$D_y = sin(\alpha) \tag{13}$$

$$D_z = cos(\alpha) \cdot sin(\beta) \tag{14}$$

2.6 Перспективная проекция

После перехода в пространство камеры необходимо спроецировать полигон на картинную плоскость. В данном курсовом проекте используется перспективная проекция. Обозначим:

— $zoom_x$ — приближение по X;

— $zoom_y$ — приближение по Y;

— R — соотношение сторон экрана;

— F — расстояние от камеры до задней грани пирамиды видимости;

— N — расстояние от камеры до передней грани пирамиды видимости;

— γ — вертикальный угол обзора.

Увеличение объектов по координатам X и Y можно вычислить по формулам:

$$zoom_y = \frac{1}{tg(\gamma/2)} \tag{15}$$

$$zoom_x = \frac{zoom_y}{R} \tag{16}$$

Для перехода в пространство отсечения используется матрица перспективной проекции:

$$\begin{pmatrix} zoom_x & 0 & 0 & 0 \\ 0 & zoom_y & 0 & 0 \\ 0 & 0 & \frac{F+N}{F-N} & 1 \\ 0 & 0 & \frac{-2 \cdot F \cdot N}{F-N} & 0 \end{pmatrix} \tag{17}$$

2.7 Отбрасывание невидимых граней

С помощью отбрасывания нелицевых граней моделей при построении изображения можно существенно сократить временные затраты, так как не полигоны, невидимые по отношению к камере, растеризоваться не будут. Для определения видимости грани требуется использовать формулу:

$$(\overrightarrow{N}, \overrightarrow{V}) = \begin{cases} \geq 0, \text{ если грань невидима} \\ < 0, \text{ если грань видима} \end{cases}, \tag{18}$$

где $\overrightarrow{N}$ — вектор внешней нормали к грани модели, $\overrightarrow{V}$ — вектор от камеры до любой точки грани.

2.8 Отсечение по пирамиде видимости

Отсечение полигона по пирамиде видимости решает две задачи:

1) удаление полигонов, лежащих за камерой, но проецируемых на картинную плоскость в перевернутом виде;
2) сокращение времени на растеризацию полигонов за счет удаления их невидимых частей.

Для выполнения операции отсечения в данном курсовом проекте использовался алгоритм Сазерленда-Ходжмана [3]. Согласно рисунку **??** его идея состоит в том, чтобы последовательно отсекать полигон относительно каждой грани пирамиды видимости до тех пор, пока полигон не будет отсечен полностью либо пока все грани не будут пройдены.

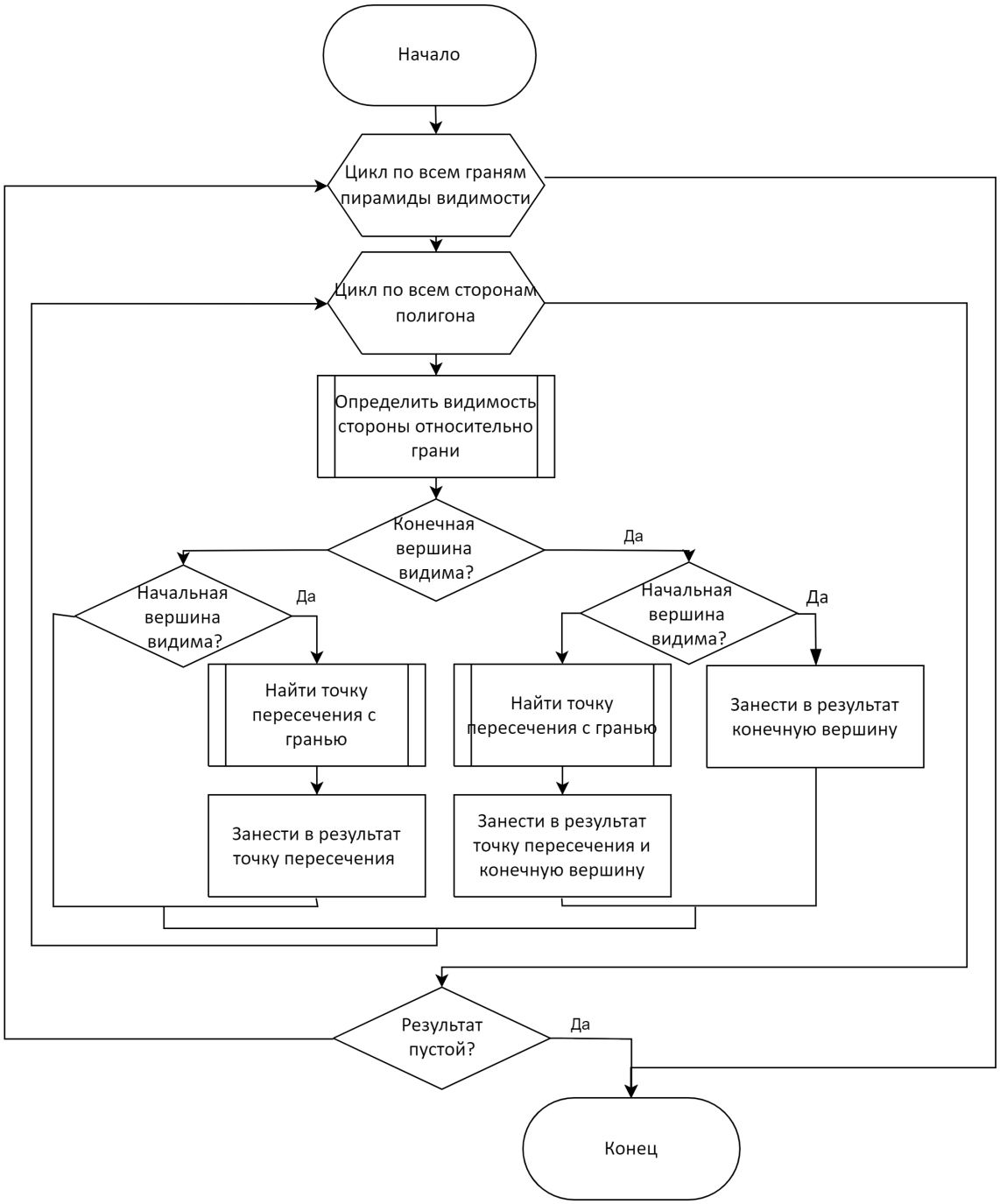

Рисунок 16 – Схема алгоритма Сазерленда-Ходжмана

2.9 Алгоритм Z-буфера

Для растеризации треугольного полигона сначала необходимо найти ограничивающий его прямоугольник, в котором это полигон содержится. Это делается для того, чтобы не тратить время на растеризацию пикселей, не являющихся частью полигона. Затем для каждого пиксела ограничивающего прямоугольника находятся его барицентрические координаты относительно вершин полигона.

Пусть A, B, C – вершины грани, P – пиксель внутри ограничивающего прямоугольника. Площадь треугольника можно найти по следующей формуле:

$$square = (A_y - C - y) \cdot (B_x - C_x) + (B_y - C_y) \cdot (C_x - A_x) \tag{19}$$

Тогда барицентрические координаты пиксела равны:

$$\alpha = \frac{(P_y - C_y) \cdot (B_x - C_x) + (B_y - C_y) \cdot (C_x - P_x)}{square} \tag{20}$$

$$\beta = \frac{(P_y - A_y) \cdot (C_x - A_x) + (C_y - A_y) \cdot (A_x - P_x)}{square} \tag{21}$$

$$\gamma = 1 - \ \alpha - \ \beta \tag{22}$$

В случае, если хоть одна из барицентрических координат отрицательна, то пиксель лежит вне полигона. Если пиксель лежит внутри треугольника, то найти значение его глубины можно по следующей формуле:

$$z = \frac{1}{\frac{\alpha}{A_z} + \frac{\beta}{B_z} + \frac{\gamma}{C_z}} \tag{23}$$

Затем производится сравнение значения глубины точки со значением глубины из Z-буфера. Если глубина пикселя меньше, значит он лежит ближе

к камере и должен быть растеризован. Происходит вычисление интенсивности пикселя, его значение заносится в буфер кадра, а в Z-буфер заносится значение глубины пиксела. Полная схема алгоритма Z-буфера представлена на рисунке 17.

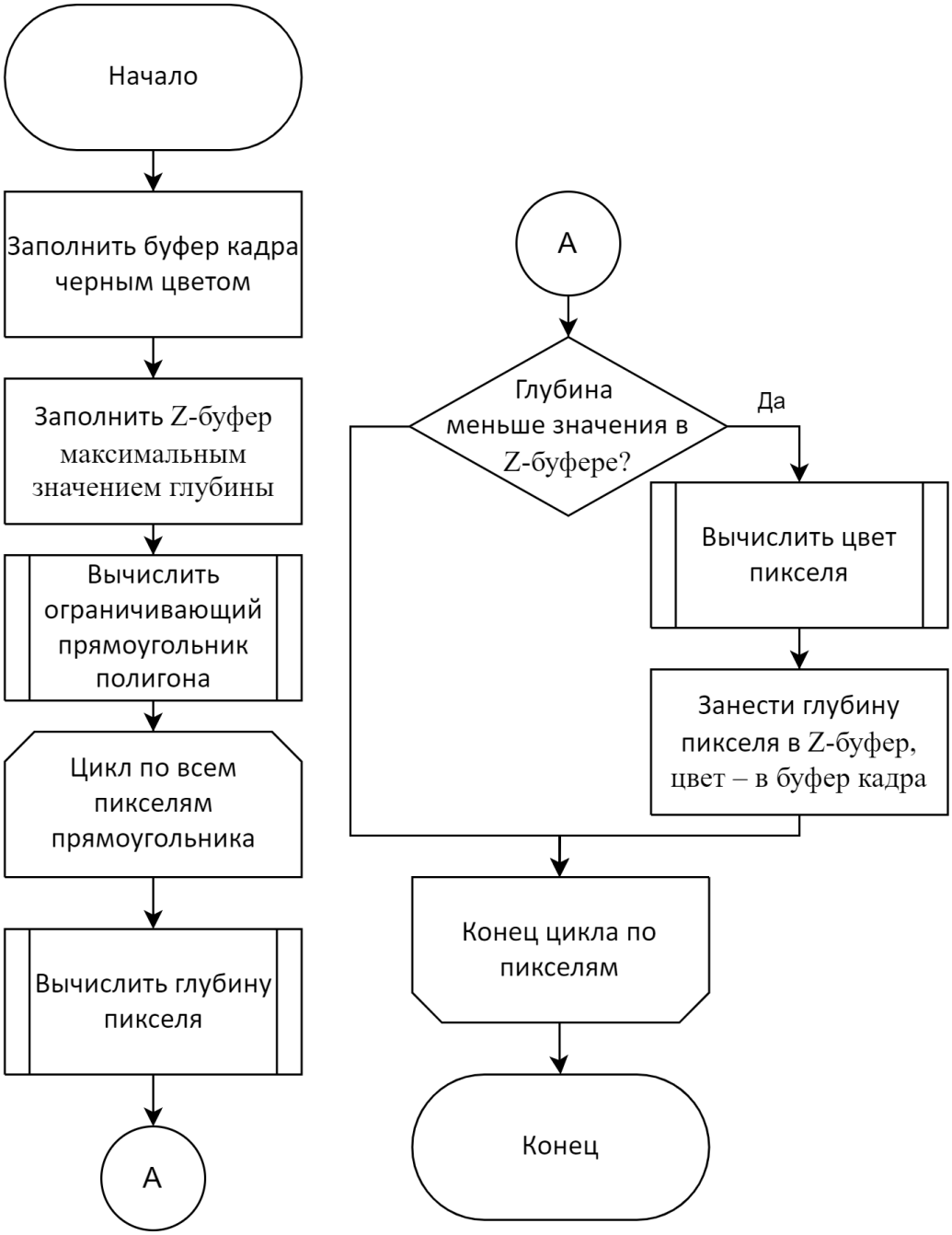

Рисунок 17 – Схема алгоритма Z-буфера

Вывод

Были описаны требования к программному обеспечению, структур данных, алгоритмы для построения сцены в пространстве изображения, изменения положения объекта в пространстве, построение камеры и ее проекций.

3 Технологическая часть

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

3.1 Средства реализации

Для написания данного курсового проекта был выбран язык C++ [9]. Выбор данного языка программирования обусловлен следующим образом:

— поддерживает объектно-ориентированную модель разработки, что позволяет структурировать программу и дает возможность эффективного написания качественного программного обеспечения;

— позволяет эффективно использовать ресурсы системы благодаря широкому набору функций и классов из стандартной библиотеки;

— обладает высокими показателями вычислительной производительности, а так как требуется быстродействие задач генерации реалистичных изображений, то язык C++ необходим.

В качестве среды разработки был использован Qt Creator [10]. Он обладает всем необходимым функционалом для написания, профилирования и отладки программ, а также создания графического пользовательского интерфейса. Данная среда поставляется с фреймворком Qt [11], который содержит в себе все необходимые средства, позволяющие работать непосредственно с пикселями изображения. Для упрощения сборки проекта программного обеспечения использовалась утилита qmake [12].

3.2 Сведения о модулях программы

3.3 Реализация алгоритмов

3.4 Интерфейс программного обеспечения

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

Технические характеристики устройства, на котором выполнялось тестирование.

— Процессор: Intel(R) Core(TM) i5-10300H CPU 2.50 ГГц [13].

— Количество ядре: 4 физических и 8 логических ядер.

— Оперативная память: 16 ГБайт.

— Операционная система: Windows 11 Pro 64-разрядная система [14].

При замерах времени ноутбук был включен в сеть электропитания и был нагружен только системными приложениями.

4.2 Постановка эксперимента

4.3 Результаты эксперимента

4.4 Вывод

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Горюнова А.М. Омельченко Т.В. Омельченко П.Н. Муслимов Д.А. Применение компьютерной графики для решения экономических и инженерных задач : учебное пособие. — Оренбург: ОГУ, 2018. — С. 153.

2 Юрьевич Аксенов Алексей. Модели и методы обработки и представления сложных пространственных объектов. — СПИИРАН, 2015. — С. 110.

3 Д. Роджерс. Алгоритмические основы машинной графики: Пер. с англ. — СПб: БХВ-Петербург, 1989. — С. 512.

4 Шикин Е. В. Боресков А. В. Компьютерная графика. Динамика, реалистические изображения. — Москва: ДИАЛОГ-МИФИ, 1996. — С. 288.

5 Метод прямой и обратной трассировки [Электронный ресурс]. — Режим доступа: `https://en.cppreference.com/w/c/language` (дата обращения: 14.09.2022).

6 Простые модели освещения [Электронный ресурс]. — Режим доступа: `https://grafika.me/node/344` (дата обращения: 24.09.2022).

7 Н. Порев В. Компьютерная графика. — СПб.: БХВ-Петербург, 2002. — С. 429.

8 Placing a Camera: the LookAt Function [Электронный ресурс]. — Режим доступа: `https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/lookat-function/framing-lookat-function.html` (дата обращения: 20.09.2022).

9 Документация по языку C++ [Электронный ресурс]. — Режим доступа:

https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-160 (дата обращения: 30.10.2022).

10 Qt Creator Manual [Электронный ресурс]. — Режим доступа: https://doc.qt.io/qtcreator (дата обращения: 30.10.2022).

11 All Qt Documentation [Электронный ресурс]. — Режим доступа: https://doc.qt.io/all-topics.html (дата обращения: 30.10.2022).

12 qmake Manual [Электронный ресурс]. — Режим доступа: https://doc.qt.io/qt-6/qmake-manual.html (дата обращения: 30.10.2022).

13 Intel [Электронный ресурс]. — Режим доступа: https://ark.intel.com/content/www/ru/ru/ark/products/201839/intel-core-i510300h-processor-8m-cache-up-to-4-50-ghz.html (дата обращения: 25.09.2022).

14 Windows 11 Pro 64-bit [Электронный ресурс]. — Режим доступа: https://www.microsoft.com/ru-ru/software-download/windows10 (дата обращения: 25.09.2022).

ПРИЛОЖЕНИЕ А

Презентация к научно-исследовательской работе

Презентация содержит 14 слайдов.