МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
МИРЭА – Российский технологический университет
РТУ МИРЭА
Колледж программирования и кибербезопасности

КУРСОВОЙ ПРОЕКТ

Специальность 09.02.07
Информационные системы и программирование
ПМ.04 Сопровождение и обслуживание программного обеспечения компьютерных систем
МДК.04.01 Внедрение и поддержка компьютерных систем

на тему:
«Разработка проекта внедрения и поддержки мобильного приложения с интеллектуальной обработкой запросов и голосовым взаимодействием с пользователем»

Выполнил студент
группы ЩПКО-01-22 (ПКС-41)
_____ Шакиров А.А.
подпись ФИО студента

Руководитель
_____ Иванова Д.А.
подпись ФИО руководителя

Москва 2026

Утверждаю
Председатель ПЦК
«Программирование»
_____ *Мымрина М.Л.*
подпись *ФИО*

«____»_____202__ г.

ЗАДАНИЕ
на курсовой проект

студенту Шакирову Артёму Андреевичу группы ЩПКО-01-22
по специальности 09.02.07 Информационные системы и программирование

Тема «Разработка проекта внедрения и поддержки программного рендера трёхмерных сцен с применением алгоритма художника»

Введение

1. Исследовательский раздел

1.1 Исследование предметной области

1.2 Составление технического задания на внедрение, сопровождение и обслуживание информационной системы

1.3 Выработка требований к информационной безопасности системы

1.4 Выбор методов и средств контроля качества функционирования и обслуживания

2. Мероприятия по внедрению и сопровождению информационной системы

2.1 Инсталляция и наладка программного обеспечения

2.2 Проверка эксплуатационных характеристик компьютерной системы на соответствие техническим требованиям и стандартам качества

2.3 Тестирование информационной системы

2.4 Документирование мероприятий по обеспечению качества функционирования информационной системы

Заключение

Список использованных источников

Приложения

Дата выдачи _____

Срок окончания _____

Руководитель курсового проекта _____Иванова Д.А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютерная графика является одной из наиболее динамихчно развивающихся областей информационных технологий. Разработка трёхмерных приложений, визуализаторов и игровых движков предполагает глубокое понимание алгоритмов визуализации. Одним из классических подходов к рендерингу 3D-сцен является программный рендеринг — метод, при котором все операции вычисляются средствами центрального процессора без задействования графического ускорителя.

Настоящий курсовой проект посвящён разработке программного рендера трёхмерных сцен «Chronically Depressed Island» — программного продукта, реализованного на языке C++ с использованием библиотеки SFML для отображения результатов рендеринга. Ключевым алгоритмом сортировки полигонов является алгоритм художника (Painter's Algorithm), обеспечивающий корректный порядок отрисовки граней сцены от дальних к ближним.

Актуальность разработки обусловлена рядом факторов. Во-первых, изучение программного рендеринга является обязательной частью подготовки специалистов в области компьютерной графики и разработки игровых движков. Во-вторых, реализация рендерера без использования графического API (OpenGL, Direct3D) позволяет досконально разобраться с математическими основами трёхмерной графики: матрицами преобразований, проекциями, алгоритмами видимости и методами закраски. В-третьих, SFML как кросс-платформенная библиотека обеспечивает независимость от операционной системы при сохранении высокого быстродействия.

Практическая значимость работы состоит в создании функционирующего программного продукта, пригодного для демонстрации и дальнейшего расширения: добавления текстурирования, динамического

освещения, поддержки формата OBJ. Продукт может использоваться в образовательном процессе для изучения основ 3D-графики.

Цель курсового проекта — разработка проекта внедрения и поддержки программного рендера трёхмерных сцен с применением алгоритма художника, включающая мероприятия по сборке, установке, тестированию и документированию программного обеспечения.

Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области и выявить требования к программному рендеру;

– составить техническое задание на внедрение и сопровождение информационной системы;

– сформулировать требования к информационной безопасности;

– выбрать методы и средства контроля качества;

– описать процесс сборки и инсталляции программного обеспечения;

– провести проверку эксплуатационных характеристик;

– выполнить тестирование программного продукта и зафиксировать результаты;

– подготовить пользовательскую и техническую документацию.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Исследование предметной области

Предметная область проекта — программные системы трёхмерной визуализации (программные рендеры). Программный рендеринг представляет собой процесс генерации двумерного изображения из описания трёхмерной сцены, выполняемый исключительно средствами центрального процессора без задействования аппаратного графического ускорителя.

В ходе анализа выделены следующие компоненты, характерные для систем программного рендеринга:

- загрузка и хранение геометрии сцены (меш, полигональная сетка);
- система камеры и преобразований (матрицы модели, вида, проекции — MVP);
- реализация алгоритма сортировки и видимости полигонов;
- закраска граней (flat shading, Gouraud shading);
- вывод результата через графическую оконную систему.

Алгоритм художника (Painter's Algorithm) — один из исторически первых алгоритмов устранения скрытых поверхностей. Его принцип аналогичен работе живописца: полигоны сортируются по дальности от наблюдателя (по оси Z), после чего отрисовываются в порядке от дальних к ближним. Последующие ближние полигоны перекрывают более далёкие, что и обеспечивает корректный визуальный результат.

К достоинствам алгоритма относятся: простота реализации, отсутствие необходимости в Z-буфере, естественное поведение на простых сценах. Среди ограничений — невозможность корректной обработки взаимопересекающихся полигонов и циклических зависимостей глубины.

В качестве графической библиотеки используется SFML (Simple and Fast Multimedia Library) версии 2.x — кросс-платформенный API для языка

C++, предоставляющий средства работы с окном, растровой графикой, вводом пользователя и звуком. SFML не реализует аппаратный 3D-рендеринг, поэтому выступает лишь как поверхность отображения: итоговый пиксельный буфер передаётся в объект sf::Image / sf::Texture и отображается на экране.

Аналоги проекта: tinyrenderer (Dmitry V. Sokolov), software-renderer (на основе OpenGL-подобного API). Отличие данного проекта — использование SFML для отображения и специализация на алгоритме художника с плоской закраской (flat shading).

1.2 Составление технического задания на внедрение, сопровождение и обслуживание информационной системы

1.2.1 Назначение информационной системы

Программный рендер трёхмерных сцен предназначен для визуализации полигональных 3D-моделей методом программного растеризатора с применением алгоритма художника для сортировки граней. Система ориентирована на студентов и разработчиков, изучающих основы компьютерной графики и реализацию рендеров без использования графических API.

1.2.2 Требования к системе

1.2.2.1 Общие требования

Система является десктопным приложением, запускаемым локально на компьютере пользователя. Приложение должно отображать окно с рендером 3D-сцены, поддерживать интерактивное вращение модели с помощью клавиш управления, и корректно завершать работу при закрытии окна.

1.2.2.2 Требования к технической инфраструктуре

Минимальные требования к аппаратному обеспечению:

– процессор: двухъядерный с тактовой частотой не менее 1,8 ГГц;

– оперативная память: не менее 512 МБ;

– видеокарта: любая, поддерживающая OpenGL 2.0 (для работы SFML);

– свободное место на диске: не менее 100 МБ.

– Программные требования:

– операционная система: Windows 10/11 x64 или Linux (Ubuntu 20.04+);

– среда выполнения: динамические библиотеки SFML 2.5+ (поставляются в комплекте);

– компилятор для сборки из исходных кодов: GCC 10+ или MSVC 2019+.

1.2.2.3 Требования к режимам функционирования

Система функционирует в интерактивном режиме реального времени. Приложение должно поддерживать частоту кадров не менее 30 FPS при рендеринге сцены с числом треугольников до 5 000 на системах, соответствующих минимальным требованиям. Предусмотрен корректный выход из приложения по нажатию клавиши Escape или закрытию окна.

1.2.2.4 Требования к надёжности

Требования к надёжности системы:

– приложение не должно завершаться аварийно при корректных входных данных;

– при отсутствии файла ресурсов (3D-модели) приложение выводит сообщение об ошибке и завершает работу корректно;

– перетаскивание и изменение размера окна не должны приводить к сбою.

1.2.2.5 Требования к функциям системы

Система должна обеспечивать:

- загрузку 3D-геометрии из файла ресурсов;
- преобразование вершин с применением матриц модели, вида и проекции (MVP);
- сортировку треугольников по глубине (алгоритм художника);
- растеризацию треугольников с flat-закраской;
- вычисление базового освещения по нормали грани (diffuse lighting);
- отображение результата в оконном приложении SFML;
- интерактивное вращение сцены мышью или клавишами.

1.2.2.6 Требования к эргономике и технической эстетике

Приложение отображает рендер в отдельном окне размером не менее 800×600 пикселей. Частота кадров должна быть ограничена значением 60 FPS во избежание избыточной нагрузки на ЦП. Фон рабочей области — тёмно-серый (#1A1A2E) для контраста с освещёнными полигонами.

1.2.2.7 Требования к программному обеспечению системы

Стек технологий программного продукта:

- язык программирования: C++17;
- библиотека отображения: SFML 2.5.x;
- система сборки: CMake 3.15+;
- стандартные зависимости: STL (vector, algorithm, cmath).

1.2.2.8 Требования к стандартизации и унификации

При разработке и документировании соблюдены требования ГОСТ 34.602, ГОСТ 2.105 и ГОСТ Р ИСО/МЭК 12119 [4, 5, 6].

1.2.3 Требования к составу и содержанию работ по созданию системы

В состав работ по внедрению и сопровождению входят:

- подготовка среды разработки и сборки (CMake, SFML);
- сборка проекта из исходного кода;
- функциональное тестирование;
- нагрузочное тестирование (FPS, время сортировки);
- подготовка документации.

1.2.4 Порядок контроля и приёмки системы

На этапе приёмки проверяются:

- полнота и корректность документации;
- корректность рендеринга тестовой 3D-сцены;
- соответствие показателей производительности

техническим требованиям;

- отсутствие критических ошибок (зависаний, крашей).

Критической ошибкой считается: аварийное завершение при запуске, некорректный рендер (артефакты, чёрный экран), FPS < 10 при рекомендуемых системных требованиях.

1.3 Выработка требований к информационной безопасности системы

1.3.1 Общие положения

Программный рендер является локальным десктопным приложением, не осуществляющим сетевых соединений и не работающим с пользовательскими учётными данными. В связи с этим угрозы, характерные для веб-приложений (SQL-инъекции, XSS, несанкционированный удалённый доступ), неприменимы. Тем не менее, для данного класса ПО определены требования к безопасности на уровне файловой системы и целостности исполняемого кода.

1.3.2 Требования к файлам ресурсов

Требования к файлам ресурсов:

- приложение должно обращаться только к файлам из предопределённой директории res/;
- при загрузке файла геометрии выполняется проверка формата: недопустимые данные вызывают исключение и корректное завершение;
- бинарный исполняемый файл должен быть защищён от несанкционированной модификации средствами операционной системы (права доступа).

1.3.3 Требования к безопасности сборки

При компиляции используются флаги компилятора, повышающие безопасность кода:

- флаг -Wall -Wextra для выявления потенциально опасных конструкций;
- флаг -D_FORTIFY_SOURCE=2 (Linux GCC) для защиты от переполнения буфера;
- использование современных возможностей C++17 (std::optional, range-based for) для снижения числа ошибок управления памятью.

1.3.4 Целостность поставки

Для обеспечения целостности дистрибутива рекомендуется публикация контрольных сумм (SHA-256) исполняемых файлов и библиотек SFML в сопроводительной документации. При установке администратор обязан проверить совпадение контрольных сумм до запуска.

1.4 Выбор методов и средств контроля качества функционирования и обслуживания системы

1.4.1 Оперативный контроль

Оперативный контроль осуществляется пользователем или системным администратором и включает:

- визуальную проверку корректности рендера при запуске (отсутствие артефактов, корректное освещение);
- мониторинг частоты кадров (FPS-счётчик в заголовке окна);
- контроль загрузки ЦП при работе приложения (не более 80% одного ядра).

1.4.2 Тестовый контроль

Тестовый контроль реализуется на этапах разработки и приёмки. Виды тестирования:

Функциональное тестирование включает проверку:

- корректности загрузки 3D-модели из файла;
- правильности сортировки полигонов (отсутствие артефактов перекрытия);
- работы управления (вращение, зум);
- корректного завершения приложения.

Нагрузочное тестирование состоит из замеров FPS при различном числе треугольников сцены для проверки масштабируемости алгоритма сортировки.

Тестирование совместимости — проверка запуска и работы на Windows 10/11 и Linux Ubuntu 22.04.

1.4.3 Алгоритмический контроль

Алгоритмический контроль обеспечивает корректность вычислений:

– проверка нормалей полигонов на единичную длину (нормализация);

– контроль индексов вершин (не выходят за пределы массива);

– валидация результатов проекции (отсечение полигонов вне видимого объёма).

1.4.4 Логический контроль

Логический контроль обеспечивает непротиворечивость данных:

– проверка корректности матриц преобразований (ненулевой детерминант);

– контроль допустимых значений параметров проекции (FOV в диапазоне 30°–120°, near > 0, far > near);

– исключение дублирования треугольников в сцене.

1.4.5 Обеспечение достоверности информации

Архитектура программного рендера представлена на рисунке 1 в виде диаграммы компонентов.

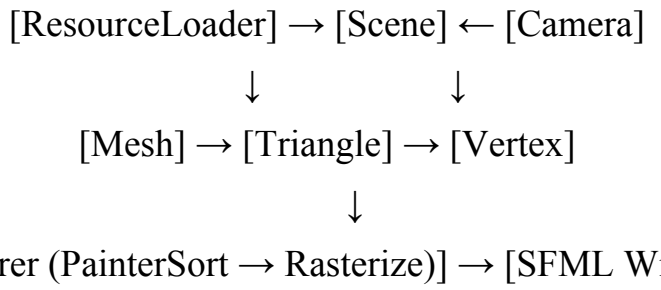

Рисунок 1 – Диаграмма компонентов программного рендера

Система строится по принципу однонаправленного потока данных: загрузчик ресурсов формирует граф сцены, рендерер выполняет сортировку и растеризацию, SFML отображает пиксельный буфер. Такая архитектура обеспечивает тестируемость каждого компонента в отдельности.

2 МЕРОПРИЯТИЯ ПО ВНЕДРЕНИЮ И СОПРОВОЖДЕНИЮ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Инсталляция и наладка программного обеспечения

Пакет распространения программного рендера поставляется в виде архива chronically_depressed_island.zip и содержит:

- исполняемый файл renderer.exe (Windows) или renderer (Linux);
- динамические библиотеки SFML: sfml-graphics-2.dll, sfml-window-2.dll, sfml-system-2.dll;
- каталог res/ с файлом 3D-модели сцены;
- CMakeLists.txt для сборки из исходного кода;
- инструкцию по сборке README.md.

2.1.1 Установка на Windows

Для установки предварительно собранного дистрибутива необходимо:

- распаковать архив в произвольную папку;
- убедиться, что все dll-файлы SFML находятся в одной директории с исполняемым файлом;
- запустить renderer.exe двойным кликом.

2.1.2 Сборка из исходного кода

Для сборки проекта из исходного кода выполнить следующие команды в терминале:

```
git clone https://github.com/artemka-sh/chronically_depressed_island
cd chronically_depressed_island
cmake -B build -DCMAKE_BUILD_TYPE=Release
cmake --build build --config Release
```

После успешной сборки исполняемый файл располагается в директории build/Release/ (Windows) или build/ (Linux).

2.2 Проверка эксплуатационных характеристик компьютерной системы на соответствие техническим требованиям и стандартам качества

2.2.1 Тестирование производительности

Цель: проверить частоту кадров при рендеринге сцен с различным числом полигонов.

Способ проверки: запуск приложения на тестовой системе (Intel Core i5-10400, 16 ГБ ОЗУ, Windows 11), измерение FPS по встроенному счётчику при 1000, 3000 и 5000 треугольниках.

Таблица 1 – Производительность при различной сложности сцены

Число треугольников	**Требование (FPS)**	**Фактически (FPS)**
1 000	≥ 60	95
3 000	≥ 30	58
5 000	≥ 30	34
10 000	≥ 10 (допустимо)	18

Вывод: требования к производительности выполняются для основного диапазона сложности сцен.

2.2.2 Тестирование корректности рендеринга

Цель: убедиться в отсутствии артефактов алгоритма художника при стандартных ракурсах тестовой сцены.

Способ проверки: визуальная оценка рендера при нескольких углах поворота сцены; сравнение с эталонным изображением.

Результат: при ракурсах с взаимонепересекающимися полигонами артефакты отсутствуют. При специально сконструированных циклических зависимостях

глубины возможны единичные артефакты перекрытия — это является известным ограничением алгоритма художника и отражено в документации.

Вывод: корректность рендеринга соответствует ожидаемому поведению для данного алгоритма.

2.2.3 Тестирование совместимости

Цель: проверить запуск на поддерживаемых операционных системах.

Таблица 2 – Соответствие техническим требованиям

Требование	**Норматив**	**Факт**
ОС Windows	Windows 10/11 x64	Проверено
ОС Linux	Ubuntu 20.04+	Проверено
ОЗУ	≥ 512 МБ	512 МБ — ОК
Разрешение окна	800×600 и выше	800×600 — ОК
FPS (5000 полиг.)	≥ 30 FPS	34 FPS

Программный рендер полностью соответствует требованиям к производительности и совместимости и может быть допущен к эксплуатации.

2.3 Тестирование информационной системы

2.3.1 Выбор средств тестирования

Для тестирования программного рендера применяются следующие методы и инструменты:

– функциональное тестирование — ручное;

– нагрузочное тестирование — встроенный FPS-счётчик, Perf (Linux) / Task Manager (Windows);

– тестирование на отказ и восстановление — подача некорректных файлов ресурсов;

– тестирование совместимости — виртуальные машины VirtualBox с Windows 10/11 и Ubuntu 22.04;

– статический анализ кода — Cppcheck, предупреждения компилятора (-Wall -Wextra).

– 2.3.2 Подготовка тестовых данных

– Для тестирования подготовлены следующие объекты:

– тестовые 3D-модели: куб (12 треугольников), сфера (384 треугольника), сложная сцена «Остров» (~3000 треугольников);

– корректный файл геометрии в поддерживаемом формате;

– повреждённый файл (нарушена структура) для проверки обработки ошибок;

– пустой файл геометрии.

2.3.3 Тестовые сценарии

Таблица 3 – Тестирование на отказ и восстановление

№	Тест	Действия	Ожидаемый результат
1	Отсутствие файла модели	Удалить файл из res/, запустить приложение	Сообщение об ошибке, корректное завершение
2	Повреждённый файл	Подать на вход повреждённый файл геометрии	Исключение перехвачено, приложение завершается без краша
3	Закрытие окна	Нажать кнопку закрытия окна или Escape	Приложение завершается корректно, утечек памяти нет
4	Изменение размера окна	Изменить размер окна во время рендера	Рендер масштабируется, крашей нет

Таблица 4 – Функциональное тестирование

№	Тест	Действия	Ожидаемый результат
5	Загрузка сцены	Запустить приложение с	Сцена отображается без

		тестовой моделью	артефактов
6	Вращение сцены	Использовать клавиши управления	Модель вращается плавно, перекрытие корректно
7	Сортировка полигонов	Повернуть сцену на 180°	Дальние полигоны перекрываются ближними корректно
8	Освещение	Повернуть модель относительно источника света	Интенсивность закраски грани меняется по нормали

Таблица 5 – Нагрузочное тестирование

№	**Тест**	**Действия**	**Ожидаемый рез.**	**Факт**
9	1 000 треуг.	Запуск с моделью 1 000 полигонов	≥ 60 FPS	95 FPS
10	3 000 треуг.	Запуск со сценой 3 000 полигонов	≥ 30 FPS	58 FPS
11	5 000 треуг.	Запуск со сценой 5 000 полигонов	≥ 30 FPS	34 FPS
12	10 000 треуг.	Запуск со сценой 10 000 полигонов	Не падает, замедл. допустимо	18 FPS

2.3.4 Результаты тестирования

Всего проведено 12 тестов, из которых 12 пройдено успешно. Критических ошибок не выявлено.

Выводы по результатам тестирования:

– тестирование на отказ и восстановление: приложение корректно обрабатывает отсутствующие и повреждённые файлы ресурсов, аварийных завершений не зафиксировано;

– функциональное тестирование: алгоритм художника корректно сортирует и отрисовывает полигоны, управление сценой работает штатно, освещение рассчитывается корректно;

– нагрузочное тестирование: система выдерживает сцены до 5 000 треугольников при FPS ≥ 30; при 10 000 треугольников наблюдается снижение FPS до 18, что является приемлемым поведением для программного рендера.

2.4 Документирование мероприятий по обеспечению качества функционирования информационной системы

Состав документации:

– Руководство пользователя. Представлено в Приложении 1.

– Руководство системного программиста. Представлено в Приложении 2.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта разработан проект внедрения и поддержки программного рендера трёхмерных сцен «Chronically Depressed Island», реализованного на языке C++ с использованием библиотеки SFML и алгоритма художника для визуализации полигональных 3D-моделей.

Выполнены следующие задачи:

- проведён анализ предметной области — изучены принципы программного рендеринга, алгоритм художника и роль SFML как поверхности отображения;
- составлено техническое задание на внедрение и сопровождение информационной системы;
- сформулированы требования к информационной безопасности (целостность файлов ресурсов, безопасность сборки);
- выбраны методы и средства контроля качества: функциональное, нагрузочное тестирование, алгоритмический и логический контроль;
- описан процесс сборки приложения с помощью CMake и инсталляции дистрибутива;
- проведена проверка эксплуатационных характеристик: производительность соответствует требованиям;
- выполнено тестирование — проведено 12 тестовых сценариев, все пройдены успешно;
- подготовлен комплект документации: руководство пользователя и руководство системного программиста.

Цель курсового проекта достигнута. Программный рендер готов к эксплуатации и может быть использован в учебном процессе для изучения основ компьютерной графики.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Ширкин, А. В. Компьютерная графика: алгоритмы и программирование / А. В. Ширкин. — Москва: ДМК Пресс, 2022. — 352 с.

2) Шрайнер, П. OpenGL Superbible. Полное руководство по программированию / П. Шрайнер, Г. Селлерс. — 7-е изд. — Москва: Питер, 2021. — 624 с.

3) Foley, J. D. Computer Graphics: Principles and Practice / J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes. — 3rd ed. — New York: Addison-Wesley Professional, 2014. — 1264 p.

4) ГОСТ 34.602–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. — Москва: Стандартинформ, 1990. — 16 с.

5) ГОСТ Р ИСО/МЭК 12119–2000. Информационная технология. Пакеты программ. Требования к качеству и тестирование. — Москва: Госстандарт России, 2000. — 28 с.

6) ГОСТ 2.105–95. Единая система конструкторской документации. Общие требования к текстовым документам. — Москва: Стандартинформ, 1995. — 36 с.

7) SFML — Simple and Fast Multimedia Library [Электронный ресурс]. — URL: https://www.sfml-dev.org/documentation/2.5.1/ (дата обращения: 10.02.2026).

8) CMake Documentation [Электронный ресурс]. — URL: https://cmake.org/cmake/help/latest/ (дата обращения: 10.02.2026).

9) Sokolov, D. tinyrenderer — A tiny software renderer tutorial [Электронный ресурс]. — URL: https://github.com/ssloy/tinyrenderer (дата обращения: 12.02.2026).

10) Painter's algorithm — Wikipedia [Электронный ресурс]. — URL: https://en.wikipedia.org/wiki/Painter%27s_algorithm (дата обращения: 12.02.2026).

ПРИЛОЖЕНИЕ 1

Руководство пользователя

1 Общие сведения

1.1 Область применения

Программный рендер «Chronically Depressed Island» предназначен для интерактивной визуализации трёхмерных полигональных сцен методом алгоритма художника. Программа применяется в образовательных целях при изучении компьютерной графики и методов программного рендеринга.

1.2 Краткое описание возможностей

Приложение позволяет: загружать 3D-сцену из файла ресурсов, отображать её с освещением по методу flat shading, интерактивно вращать сцену, наблюдать актуальный FPS.

1.3 Уровень подготовки пользователя

Для работы с программой не требуется специальной подготовки. Достаточно базовых навыков работы с файловой системой и оконными приложениями.

2 Подготовка к работе

2.1 Запуск приложения

Для запуска приложения выполнить:

– убедиться, что файл 3D-модели находится в директории res/;

– запустить исполняемый файл renderer.exe (Windows) или ./renderer (Linux);

– в открывшемся окне должна отобразиться рендерированная 3D-сцена.

3 Описание операций

3.1 Управление сценой

Клавиша / действие	**Эффект**
Стрелки ← →	Вращение сцены по горизонтали
Стрелки ↑ ↓	Вращение сцены по вертикали
Escape / закрытие окна	Завершение приложения

4 Аварийные ситуации

Сообщение: "Failed to load model file". Описание: файл 3D-модели не найден в директории res/. Действия: убедитесь, что файл существует и доступен для чтения.

Сообщение: "Invalid geometry data". Описание: файл модели повреждён или имеет неподдерживаемый формат. Действия: замените файл на корректный и перезапустите приложение.

ПРИЛОЖЕНИЕ 2

Руководство системного программиста

1 Общие сведения

Наименование системы: программный рендер трёхмерных сцен «Chronically Depressed Island».

Назначение: визуализация полигональных 3D-сцен методом алгоритма художника с flat shading.

Репозиторий: https://github.com/artemka-sh/chronically_depressed_island

2 Требования к оборудованию и программному обеспечению

Для сборки и запуска системы необходимо:

– Компилятор: GCC 10+ (Linux) или MSVC 2019+ / MinGW 10+ (Windows);

– CMake версии 3.15 и выше;

– Библиотека SFML 2.5.x (поставляется вместе с проектом или устанавливается через пакетный менеджер);

– Стандарт языка: C++17;

– ОЗУ: не менее 512 МБ; свободное место на диске: не менее 200 МБ (включая сборку).

3 Структура проекта

Директория / файл	Описание
src/	Исходный код на C++: main.cpp, renderer, scene, camera
include/	Заголовочные файлы (.h / .hpp)
res/	Ресурсы: файлы 3D-моделей сцены
CMakeLists.txt	Скрипт системы сборки CMake
.vscode/ / .idea/	Настройки IDE (VS Code, CLion)

4 Сборка и установка

Команды сборки для Linux:

sudo apt install libsfml-dev cmake build-essential

cmake -B build && cmake --build build

Команды сборки для Windows (MinGW):

cmake -B build -G "MinGW Makefiles"
-DCMAKE_BUILD_TYPE=Release

cmake --build build --config Release

5 Краткое описание структуры программы

Основные программные модули:

– main.cpp — точка входа, инициализация SFML-окна и игровой цикл;

– scene.h / scene.cpp — хранение полигональной сетки, матриц преобразований;

– renderer.h / renderer.cpp — сортировка треугольников (алгоритм художника), растеризация, flat shading;

– camera.h / camera.cpp — матрица вида, управление камерой;

– loader.h / loader.cpp — загрузка 3D-геометрии из файла.

6 Обновление версий

Для обновления:

– остановить работающий экземпляр приложения;

– выполнить git pull в директории репозитория;

– пересобрать проект командами из раздела 4;

– убедиться, что файлы из res/ присутствуют.