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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Студент Шпаковский Павел Александрович
фамилия, имя, отчество

Группа ИУ7-53Б

Тип практики технологическая

Название предприятия МГТУ им. Н.Э. Баумана

Студент _____ Шпаковский П. А.
подпись, дата *фамилия, и.о.*

Руководитель практики _____ Куров А. В.
подпись, дата *фамилия, и.о.*

Оценка _____

2023 г.

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)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____И.В.Рудаков
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине _____Компьютерная_графика_____

Студент группы _____ИУ7-53Б_____

_____Шпаковский Павел Александрович_____
(Фамилия, имя, отчество)

Тема курсовой работы Разработка программного обеспечения для построения трехмерных изображений с использованием микроконтроллера Raspberry Pi Pico_____

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
_____учебная_____
Источник тематики (кафедра, предприятие, НИР) _____кафедра_____

График выполнения работы: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание Разработать программное обеспечение с пользовательским интерфейсом, которое предоставляет возможность генерации трехмерных изображений с помощью микроконтроллера Raspberry Pi Pico. Интерфейс должен позволять пользователю задавать параметры моделей, их расположения на сцене и освещения, а также управлять положением камеры (вращение, перемещение, масштабирование). Необходимо реализовать возможность просмотра трехмерной сцены с помощью шести дисплеев, на каждом из которых отображается соответствующая проекция трехмерной модели на сцене.

Оформление курсовой работы:

Расчетно-пояснительная записка на _25-35__ листах формата А4.

Расчетно-пояснительная записка должна содержать постановку задачи, введение, аналитическую часть, конструкторскую часть, технологическую часть, эксперементально-исследовательский раздел, заключение, список литературы, приложения.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
На защиту проекта должна быть предоставлена презентация, состоящая из 10-15 слайдов.
На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура программного обеспечения, интерфейс, результаты проведенных исследований.

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсовой работы _____ Ю. В. Строганов
(Подпись, дата) (И.О.Фамилия)

Студент _____ П. А. Шпаковский
(Подпись, дата) (И.О.Фамилия)

Содержание

Введение *4*

1 Аналитическая часть *6*

1.1 Описание объектов визуализируемой сцены **6**

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей **6**

1.2.1 Алгоритм Робертса 7

1.2.2 Алгоритм, использующий Z-буфер 8

1.2.3 Алгоритм Варнока 9

1.2.4 Алгоритм, использующий список приоритетов 10

1.2.5 Алгоритм трассировки лучей 11

1.3 Анализ алгоритмов закраски **12**

1.3.1 Простая закраска 12

1.3.2 Закраска по Гуро 13

1.3.3 Закраска по Фонгу 14

1.4 Критерии выбора алгоритмов трехмерной визуализации **15**

1.5 Вывод из аналитической части **15**

Введение

Компьютерная графика — это область информационных технологий, которая занимается созданием, обработкой, анализом и визуализацией графических объектов и изображений с использованием компьютерных методов и алгоритмов. Она имеет широкий спектр применений, включая развлечения, образование, научные исследования, медицинскую диагностику, архитектурное проектирование, визуализацию данных и многое другое.

Основные задачи, которые решает компьютерная графика, включают:

1) *Создание и редактирование изображений*. Компьютерная графика позволяет создавать и редактировать графические объекты и изображения на компьютере. Это включает в себя рисование, моделирование форм и текстур, создание анимаций и спецэффектов.
2) *Визуализация данных*. В компьютерной графике данные могут быть представлены в визуальной форме, что делает их более понятными и удобными для анализа. Например, диаграммы, графики и трехмерные визуализации данных используются для отображения статистических и научных результатов.
3) *3D-моделирование и анимация*. Компьютерная графика позволяет создавать трехмерные модели объектов и сцен, а также анимировать их движение и поведение. Это широко используется в фильмах, видеоиграх, архитектурном проектировании и медицинской визуализации.

Общим для всех этих задач является использование математических алгоритмов, программного обеспечения и аппаратных средств для создания, модификации и визуализации графических данных с целью достижения определенных визуальных и функциональных результатов.

Для визуализации трехмерных сцен на конкретной аппаратной платформе часто требуется разрабатывать индивидуальные программные решения, учитывающие особенности данного обеспечения. Среди таких платформ можно выделить архитектуру ARM и, в частности, микроконтроллеры, оборудованные чипом RP2040, который был разработан компанией Raspberry Pi Ltd. Выбор платформы Raspberry Pi Pico обусловлен ее популярностью в промышленном применении и достаточной мощностью, необходимой для реализации алгоритмов компьютерной графики.

Цель данной работы — реализовать программное обеспечение для построения моделей трехмерных объектов, с использованием микроконтроллера Raspberry Pi Pico.

Чтобы достичь поставленной цели, требуется решить следующие задачи:

1) Описать структуру трехмерной сцены, включая объекты, из которых состоит сцена, и определить формат задания исходных данных.
2) Выбрать наиболее подходящий из существующих алгоритмов трехмерной графики, позволяющих синтезировать изображение трехмерной сцены.
3) Реализовать выбранные алгоритмы построения трехмерной сцены.
4) Исследовать возможности микроконтроллера Raspberry Pi Pico в области визуализации трехмерной графики.

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 Описание объектов визуализируемой сцены

Трехмерная сцена состоит из:

1) Объектов сцены, представляющих из себя многогранники, расположенные в пространстве сцены. Каждый многогранник задается списком полигонов, где полигон описывается множеством точек в трехмерном пространстве.
2) Материалов и свойств объектов сцены. Каждый объект может иметь определенные свойства, включая цвет, текстуры, прозрачность, отражение и блеск. Эти свойства могут быть формализованы с помощью числовых параметров, которые определяют, как объект взаимодействует со светом и как он будет визуализирован.
3) Источников света, которые могут быть формализованы как направленные вектора.
4) Камеры, через которую наблюдается данная сцена. Ее параметры могут включать положение, ориентацию, угол обзора и другие характеристики.

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей

В процессе создания реалистичных изображений ключевой задачей является обеспечение видимости объектов и их частей, учитывая перекрытия между ними. Другими словами, необходимо определить, какие элементы сцены должны быть видны, а какие скрыты от наблюдателя. Эта задача решается с помощью двух групп алгоритмов:

1) Алгоритмы, оперирующие в объектном пространстве. Эти алгоритмы связаны с мировой или физической системой координат. Они предназначены для определения видимости объектов, учитывая их пространственное расположение. Однако такие методы требуют значительных вычислительных ресурсов, которые зависят от сложности сцены и необходимой точности. Примерами таких алгоритмов могут быть алгоритм Робертса, алгоритмы на основе списков приоритетов и др.
2) Алгоритмы, оперирующие в пространстве изображения. Эти алгоритмы ориентированы на систему координат экрана или плоскости изображения, на которую проецируются объекты. Этот подход требует меньше вычислительных ресурсов по сравнению с первой группой, и его объем зависит от разрешения экрана и количества объектов на сцене. Примерами таких алгоритмов являются алгоритм Варнака, алгоритм Z-буфера и метод трассировки лучей.

1.2.1 Алгоритм Робертса

Алгоритм Робертса — это метод удаления невидимых линий и поверхностей в компьютерной графике, который действует в объектном пространстве. Его основной целью является определение, какие линии (рёбра) объектов на сцене будут видны из точки наблюдения, а какие будут скрыты за другими объектами.

Алгоритм Робертса делится на 3 этапа:

1) На первом этапе каждое тело анализируется индивидуально с целью удаления нелицевых плоскостей. Для каждого тела и каждой грани тела вычисляется уравнение плоскости. После чего проверяется знак уравнения плоскости и формируется матрица тела.
2) Удаление из каждого тела тех ребер, которые экранируются всеми остальными телами в сцене. Если задано только одно тело, то алгоритм завершается.

3) На третьем этапе вычисляются отрезки, которые образуют новые ребра при протыкании телами друг друга.

За счет того, что алгоритм Робертса работает в объектном пространстве, получается достичь высокую точность результирующего изображения. Однако в данном алгоритме используется много трудоемких математических вычислений и сложность растет как квадрат числа объектов на сцене.

1.2.2 Алгоритм, использующий Z-буфер

Алгоритм, использующий Z-буфер, представляет собой один из наиболее простых и широко применяемых методов для удаления невидимых поверхностей. Этот алгоритм оперирует в пространстве изображения и основывается на идее использования двух буферов: буфера кадра и Z-буфера.

1) Буфер кадра (Frame Buffer). Этот буфер представляет собой массив пикселей, в котором будут храниться атрибуты (например, цвет) каждого пикселя на экране.
2) Z-буфер (Depth Buffer). Это отдельный буфер глубины, который будет хранить глубину (Z-координату) каждого пикселя на экране. Исходно он заполняется максимально возможными значениями глубины.

Алгоритм работает следующим образом:

1) Перед началом работы буфер кадра заполняется информацией о фоновом изображении, а Z-буфер заполняется минимальными значениями глубины.
2) Для каждого объекта на сцене:
    a) Полигоны объекта преобразуются в растровую форму.
    b) Для каждого пикселя текущего полигона проверяется его Z-координата (глубина) в сравнении с соответствующим значением в Z-буфере.

c) Если Z-координата текущего пикселя меньше, чем значение в Z-буфере, то пиксель заносится в буфер кадра, а его Z-координата обновляется в Z-буфере.

3) На выходе получается изображение, в котором каждый пиксель находится в буфере кадра в соответствии с его видимостью и глубиной.

Преимущества и недостатки алгоритма:

Преимущества:

- Прост в реализации и позволяет достичь хороших результатов без сложной сортировки объектов.
- Эффективно обрабатывает сцены с большим количеством объектов.

Недостатки:

- Требует большого объема памяти для хранения двух буферов.
- Может иметь проблемы с прозрачностью и лестничным эффектом (ступенчатость на границах объектов).

1.2.3 Алгоритм Варнока

Алгоритм Варнока - это метод удаления невидимых поверхностей в компьютерной графике, который использует разбиение сцены на более мелкие части для определения видимости объектов.

Идея алгоритма Варнока заключается в разбиении экрана на рекурсивные прямоугольники, которые обозначают различные части сцены. Затем каждый прямоугольник проверяется на наличие видимых объектов. Если внутри прямоугольника объекты видимы, он разбивается на более мелкие прямоугольники, и процесс продолжается до достижения требуемого уровня детализации или пока не будут найдены видимые объекты.

Процесс алгоритма Варнока:

1) Разбиение экрана. Исходный экран разбивается на начальные прямоугольники. Эти прямоугольники могут представлять собой целый экран или другие крупные области.
2) Проверка видимости. Для каждого прямоугольника выполняется проверка видимости объектов, находящихся внутри него.

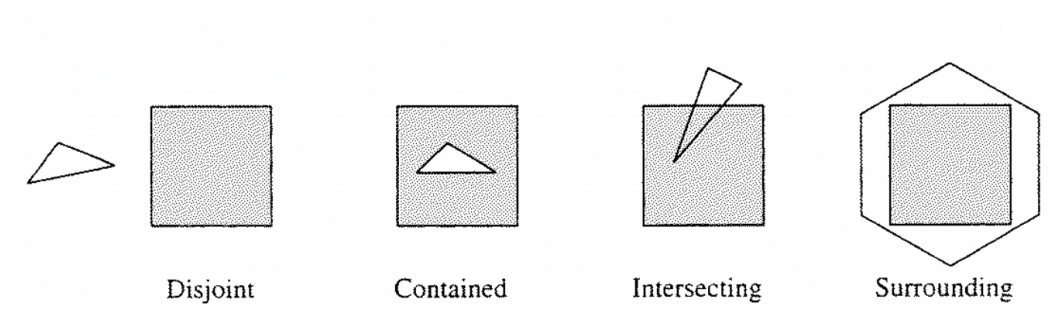

Рисунок 1.1 — Варианты взаимного расположения многоугольников

3) Принятие решения. В зависимости от результата проверки видимости объектов внутри прямоугольника, он может быть разбит на более мелкие подпрямоугольники или объединен с другими прямоугольниками. Этот процесс продолжается рекурсивно.
4) Заполнение буфера кадра. Для видимых прямоугольников или их частей выполняется заполнение буфера кадра атрибутами объектов.
5) Завершение. Процесс рекурсивного разбиения и проверки видимости продолжается до достижения требуемого уровня детализации или до тех пор, пока не будет достигнута наименьшая обрабатываемая единица.

Преимущества алгоритма Варнока включают возможность обработки сложных сцен, включая объекты с перекрывающимися поверхностями. Однако он может потребовать больших вычислительных ресурсов и времени, особенно при работе с детализированными сценами.

1.2.4 Алгоритм, использующий список приоритетов

Этот алгоритм базируется на упорядочивании объектов сцены в зависимости от их приоритета, который определяется их удаленностью от наблюдателя или

глубиной в сцене. Идея заключается в начальном рендеринге объектов, находящихся на большем расстоянии от наблюдателя, и последующем перекрытии их ближайшими объектами.

Сначала происходит сортировка объектов по глубине, что создает первоначальный список приоритетов. Затем этот список подвергается коррекции через выполнение тестов на экранирование для каждой пары многоугольников в списке. Эти тесты могут быть достаточно затратными по времени и сложности реализации. Суть тестов заключается в определении, перекрывает ли один многоугольник другой, и, если да, то какой из них ближе к наблюдателю.

Важно отметить, что идентификация случаев пересечения и циклического перекрытия многоугольников в этом алгоритме является сложной задачей. Вследствие этого алгоритм может сталкиваться с ограничениями в эффективности и точности при обработке таких сцен, где объекты перекрывают друг друга в сложных сценариях.

1.2.5 Алгоритм трассировки лучей

Алгоритм трассировки лучей – это метод удаления невидимых поверхностей в компьютерной графике, который симулирует путь световых лучей, падающих на объекты в сцене. Он создает фотореалистичные изображения, учитывая отражение, преломление и тени.

Предполагается, что наблюдатель находится на бесконечности положительной полуоси Z, и все световые лучи от наблюдателя параллельны этой оси. В ходе алгоритма обратной трассировки лучей, начиная от наблюдателя, генерируются лучи и находятся пересечения этих лучей с объектами сцены. Путем анализа этих пересечений определяется самая удаленная точка пересечения по оси Z, которая представляет видимую часть поверхности. Атрибуты этого объекта затем используются для определения характеристик пикселя, через который проходит этот световой луч.

Для создания реалистичных эффектов освещения алгоритм генерирует дополнительные лучи от точек пересечения к источникам света. Если на пути такого луча встречается непрозрачный объект, это означает, что данная точка находится в тени.

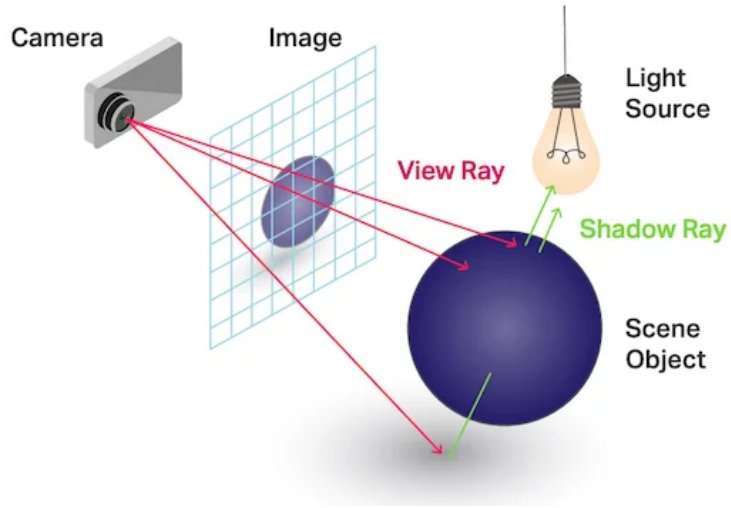

Рисунок 1.2 — Иллюстрация работы алгоритма трассировки лучей

Основными преимуществами данного алгоритма являются реалистичность получаемого изображения и возможность анализировать наложение света. Однако из-за использования большого количества лучей производительность алгоритм снижается, поскольку для каждого луча необходимо производить поиск пересечений с объектами сцены.

1.3 Анализ алгоритмов закраски

1.3.1 Простая закраска

Применяется закрашивание, в котором каждая грань получает один уровень интенсивности, вычисляемый согласно закону Ламберта. В результате такой закраски все плоские поверхности, включая те, что аппроксимируют фигуры, подвергаются однородному окрашиванию. В случае фигур вращения это может вызвать появление ложных границ между поверхностями.

Этот метод обладает высокой производительностью, однако все пиксели на грани получают одинаковую интенсивность, что придает сцене нереалистичный вид. Несмотря на это, метод очень прост в реализации и не требует значительных вычислительных ресурсов.

1.3.2 Закраска по Гуро

Закраска по методу Гуро основана на билинейной интерполяции интенсивностей на поверхности грани. Этот метод создает иллюзию плавной криволинейной поверхности, так как интенсивность вычисляется для каждой точки грани, что способствует мягкому переходу между цветами.

Закраска по методу Гуро хорошо сочетается с простой моделью освещения, основанной на диффузном отражении света. Она значительно улучшает качество изображения, придавая ему более реалистичный вид.

Однако у этого метода есть свои недостатки:

1) Возникает эффект полос Маха - артефакт, проявляющийся в виде полос вдоль граней изображения.
2) Из-за сглаживания интенсивностей может быть утрачена четкая граница между гранями многогранника, что в некоторых ситуациях может привести к неверному визуальному результату.

Процедура закрашивания граней методом Гуро состоит из четырех последовательных этапов:

1) Вычисление граничных нормалей. На этом этапе определяются нормали для каждой отдельной грани. Нормаль — это вектор, перпендикулярный к поверхности грани и указывающий направление в пространстве.
2) Определение вершинных нормалей. Затем определяются нормали в вершинах объекта. Нормаль в вершине вычисляется как среднее значение нормалей граней, смежных с данной вершиной. Это позволяет создать

плавные переходы между нормалями граней и учесть геометрическую структуру объекта.

3) Вычисление яркостей в вершинах. Исходя из нормалей в вершинах, определяются значения яркостей (интенсивностей) в этих вершинах, согласно выбранной модели отражения света. Это позволяет учесть, как свет падает на разные части объекта.
4) Закрашивание граней. Наконец, происходит закрашивание полигонов граней. Каждая грань получает цвет, который вычисляется путем линейной интерполяции значений интенсивностей в вершинах этой грани. Это создает плавные переходы цветов на поверхности и добавляет визуальную реалистичность объекту.

1.3.3 Закраска по Фонгу

Закраска по модели Фонга, по своей концепции, схожа с закраской по методу Гуро, но имеет отличие в том, как интерполируются значения. В методе Гуро производится интерполяция значений интенсивности для всех точек полигона, тогда как в методе Фонга интерполируются векторы нормалей граней, и на основе этих нормалей определяется интенсивность для каждой точки.

Этапы алгоритма закраски по Фонгу следующие:

1) Вычисление нормалей для каждой грани.
2) На основе нормалей граней определяются нормали в вершинах. В каждой точке грани интерполируется вектор нормали, учитывая соседние вершины.
3) С использованием направления векторов нормали определяется цвет точек грани в соответствии с выбранной моделью освещения.

Алгоритм закраски по модели Фонга требует больше вычислительных ресурсов по сравнению с предыдущими методами, но он позволяет достичь лучшей локальной аппроксимации кривизны поверхности. Это приводит к более

реалистичному изображению, так как метод учитывает изменения в направлениях нормалей на поверхности, что способствует созданию более точных эффектов освещения.

1.4 Критерии выбора алгоритмов трехмерной визуализации

При выборе алгоритмов компьютерной графики для визуализации трехмерной сцены следует учитывать особенности платформы, с помощью которой будет решаться задача. В данном случае необходимо рассмотреть возможности микроконтроллеров на чипе RP2040 и, в частности, платы Raspberry Pi Pico.

1) Объем оперативной памяти на микроконтроллере достаточно ограничен. Обычно он составляет не больше нескольких мегабайт. В случае модели Raspberry Pi Pico объем памяти 264Кб. Поэтому при выборе алгоритмов стоит учитывать размер используемых структур данных и затраченной памяти в целом.
2) Вычислительная мощность микроконтроллеров также невысокая, поэтому при выборе алгоритма стоит учитывать его простоту и эффективность работы. Немаловажной будет являться сложность работы алгоритма, по возможности, она должна быть линейной в зависимости от количества объектов сцены.
3) Объем встроенной постоянной памяти в микроконтроллерах небольшой и составляет не более 2 мегабайт. Исходя из этого размер исполняемого файла должен быть меньше, поэтому стоит выбирать алгоритмы, реализация которых более лаконичная и компактная.

Для поставленной задачи критерий, связанный с объемом оперативной памяти, будет приоритетным.

1.5 Вывод из аналитической части

Проанализировав алгоритмы удаления невидимых линий и поверхностей и различные алгоритмы закраски, а также рассмотрев критерии выбора наиболее

оптимального алгоритма для поставленной задачи визуализации трехмерной сцены с использованием микроконтроллера, можно сделать вывод, что алгоритм Варнока и алгоритм простой закраски подходят лучше всего. Алгоритм Варнока использует оптимальное количество памяти и требует меньшей вычислительной мощности, по сравнению с другими алгоритмами. Простая закраска наименее ресурсоемкая из всех алгоритмов закраски, при этом позволяет достичь поставленной цели. При выборе алгоритмов ключевую роль играли особенности аппаратной платформы.