МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**ПРИМЕНЕНИЕ МЕТОДОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ ДЛЯ ЗАДАЧ МЕДИЦИНСКОЙ ДИАГНОСТИКИ**

Работу выполнил Р. В. Реутов

(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики 3 курс

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Научный руководитель

КАК ТУТ ВЕРНО. Т.А. Приходько

(подпись, дата) (инициалы, фамилия)

Нормоконтролер

преп. А. А. Климец

(подпись, дата) (инициалы, фамилия)

Краснодар

2020

СОДЕРЖАНИЕ

Введение

[Введение 3](#_Toc58769219)

[1 Математические положения 5](#_Toc58769220)

[1.1 Базовые понятия линейной алгебры 5](#_Toc58769221)

[1.2 Матрицы преобразований и переходов 6](#_Toc58769222)

[1.3 Углы Эйлера 9](#_Toc58769223)

[2 Используемые алгоритмы и математические модели 12](#_Toc58769224)

[2.1 Модель освещения Фонга 12](#_Toc58769225)

[2.2 Карты нормалей и глубины 13](#_Toc58769226)

[2.3 Всенаправленные карты теней 14](#_Toc58769227)

[2.4 Эффекты постобработки 16](#_Toc58769228)

[3 Используемый инструментарий 17](#_Toc58769229)

[3.1 Описание OpenGL 3.3 17](#_Toc58769230)

[3.2 Связанные библиотеки 18](#_Toc58769231)

[3.3 Инструменты отладки 18](#_Toc58769232)

[4 Структура программного модуля 20](#_Toc58769233)

[4.1 Объектно-ориентированная оболочка над OpenGL 20](#_Toc58769234)

[4.2 Описание сцены и взаимодействия с ней 20](#_Toc58769235)

[Список использованных источников 24](#_Toc58769236)

[Приложение 25](#_Toc58769237)

# ВВЕДЕНИЕ

Компьютерная графика является областью деятельности, в которой компьютер используется для формирования, обработки и вывода изображений на экран. Компьютерная графика становится все более востребованной в современном мире, когда большая часть обрабатываемых данных требует визуализации, либо уже является изображением, требующим некоторой обработки. Для существенного ускорения всех алгоритмов, связанных с обработкой и созданием изображений используется GPU (Graphic Processor Unit), который позволяет обрабатывать каждый пиксель изображения отдельно.

На данный момент основные области, в которых требуется обработка компьютерной графики - системы автоматического проектирования (САПР), графические редакторы, компьютерные игры, спецэффекты для фильмов. Каждая из вышеперечисленных областей имеет свои требования к реалистичности изображения, производительности вовремя его создания. Актуальность задачи обосновывается тем, что названные области на данный момент очень востребованы и существует множество задач, которые требуют легких и простых программных решений.

Примером практической задачи, в которую можно решить посредствам трехмерной компьютерной графики, является задача рендера кадра мультфильма. Входными данными могут являться сцена, эффекты постобработки; модели, объекты и освещение на сцене, а результатом работы - отдельный кадр мультфильма. Данный программный модуль будет ценен тем, что является полностью открытым и имеющим документацию, сгенерированную Doxygen, что позволяет относительно легко и быстро использовать его для решения задач визуализации сцен.

В многопоточных программах, работающих с компьютерной графикой, зачастую используются отдельные однопоточные модули, отвечающие за рендер требуемой сцены. В данной курсовой работе будет представлен программный модуль, который визуализирует заданную сцену и предоставляет удобное API для более гибкой настройки модуля. Он работает на низком уровне, используя библиотеку OpenGL. Ценностью данного модуля является простота работы с ним, быстродействие и простота интеграции в приложение.

Курсовая работа состоит из четырех глав.

Первая глава работы содержит общие теоретические положения используемой в программном модуле линейной алгебры.

Вторая глава работы описывает используемые алгоритмы и подходы к решению поставленных задач, описывает используемые математические модели.

Третья глава содержит общие теоритические положения по используемым в программном модуле сторонним библиотекам.

Четвертая глава описывает непосредственно детали реализации программного модуля.

# 1 Математические положения

## Базовые понятия линейной алгебры

Так как GPU используется для ускорения вычислений, в основном связанных с матрицами и векторами, в модуле происходит огромное количество вычислений, связанных с ними.

Необходимо ввести следующие понятия и определения.

Будем называть матрицу квадратной, если , или же прямоугольной в противном случае.

Матрицу размерности называют вектор-столбцом, а размерности вектор-строкой.

Матрицу размерности , все элементы которой являются нулевыми называют нулевой матрицей и обозначают как или или . Данная матрица имеет ранг 0. Она является аддитивным нейтральным элементом множества матриц своего размера.

Матрицы и называют согласованными, если количество столбцов матрицы совпадает с количеством строк матрицы .

Для матриц определены алгебраические операции сложения, умножения матрицы на вектор, умножения матрицы на скаляр, умножения матрицы на матрицу. Необходимо рассмотреть их свойства для дальнейших рассуждений.

Некоторую алгебраическую операцию на множестве A называют коммутативной, если выполняется условие: . Операцию называют ассоциативной, если . Говорят, что две операции и называют дистрибутивными слева, если выполняется и дистрибутивными справа, если .

Операция сложения матриц A и B, представляющая собой вычисление новой матрицы C, элементы которой вычисляются по формуле , является ассоциативной и коммутативной операцией. Кроме того, для любой матрицы A существует противоположная матрица –A, такая что: .

Операция умножения матрицы A на число , которая заключается в построении новой матрицы C, элементы которой вычисляются по формуле , является ассоциативной и дистрибутивной слева и справа.

Операция умножения двух матриц и является операцией построения новой матрицы , элементы которой вычисляются следующим образом: . Умножение является ассоциативным, некоммутативным (в общем случае), дистрибутивным слева и справа. Операция ассоциативная и дистрибутивная относительно умножения на число .

## 1.2 Матрицы преобразований и переходов

В разрабатываемом программном модуле, в первую очередь, матрицы применяются для решения задачи вычисления вычислений новых координат объекта при его трансформации: перемещении, растягивании, повороте и других. Во всех последующих вычислениях предполагается, что они происходят в однородных двумерных или трехмерных системах координат, то есть таких, в которых определяемых с помощью точек объект, при умножении всех его точек на одно и то же ненулевое число не изменяется.

Так как используется однородная система координат, точка двумерного пространства с координатами представляется в виде вектора . Константный элемент вектора используется для использования вектора в вычислениях совместно с матрицами размера 3х3. В трехмерном случае ситуация аналогична. Точка с координатами представляется в виде вектора . Используются матрицы размерности 4х4.

Наиболее часто используемые матрицы трансформаций приведены в таблице 1.

Таблица 1 ­ Матрицы трансформации для двумерных и трехмерных однородных координат.

|  |  |  |
| --- | --- | --- |
|  | Однородные двумерные координаты | Однородные трехмерные координаты |
| Перемещение  ­ смещение по оси OX  ­ смещение по оси OY  ­ смещение по оси OZ |  |  |
| Масштабирование (растягивание)  ­ коэффициенты масштабирования по осям OX, OY, OZ соответственно. |  |  |
| Поворот  Угол является углом поворота в двумерной плоскости. |  | Относительно оси OX |
| Относительно оси OY |
| Относительно оси OZ |

Несколько матриц трансформации можно скомбинировать в одну, упрощая вычисления и оптимизируя их. Для ее вычисления необходимо перемножить все используемые матрицы трансформации по формуле 1.

(1)

Матрицей перехода называют матрицу, переводящую базис линейного пространства в иной базис . Рассматривается только случай трехмерного пространства. В программном модуле используются локальные (модельные) пространства, мировые пространства, пространства вида, пространства отсечения. Для преобразования точек объекта используются матрицы переходов между данными системами координат – матрица модели, матрица вида и матрица проекции.

В локальной системе координат точки объекта находятся изначально. Обычно, она начинается там же, где начинается и сам объект.

В мировой системе координат точки расположены относительно начала общей для всех объектов точки отчета. Точки в нее переводятся из локальных систем координат. Если бы точки изначально находились в мировой системе, произошло бы достаточно сложное и неудобное нагромождение. Преобразование происходит при помощи матрицы модели, вычисленной по формуле 1: . Матрица модели перемещает, масштабирует и вращает объект, располагая его в мировом пространстве.

Пространство вида (или же, как его еще называют, пространстве камеры или наблюдателя) – пространство, которое является результатом преобразований мировых координат в координаты, которые выглядят так, будто наблюдатель смотрит на объекты спереди. Таким образом – это пространство, видимое через видоискатель камеры. Для создания такой матрицы возможно использование самых разнообразных вычислений, которые в основном представляют собой совокупность сдвигов и вращений сцены.

Пространство отсечения – это пространство, все элементы которого находятся в диапазоне . Для преобразования координат пространства вида в координаты пространства отсечения используется матрица проекции . Матрица проекции определяет так называемый «объем просмотра», который представляет собой усеченную пирамиду. Все точки, попадающие в нее располагаются в вышеописанном диапазоне. Данное преобразование носит исключительно технический характер – оно переводит точки в нормализованную систему координат экрана (NDC). Матрица может быть представлена в двух формах: ортографической и перспективной.

В ортографической проекции усеченная пирамида представлена в виде параллелограмма. Происходит прямое преобразование попадающих точек в нормализованные координаты, без учета их глубины, то есть непосредственно отображает их на двумерную плоскость. Непосредственное отображение не всегда необходимо – зачастую, необходимо учитывать влияние перспективы на отображение точек.

Перспектива – это эффект, при котором объекты, расположенные дальше от точки наблюдения, выглядят меньше более близких. Данный эффект достигается при помощи применения матрицы перспективной проекции. В данном случае создается уже усеченная пирамида. В данном случае удаленные вершины кажутся намного более маленькими.

Все вышеописанные матрицы преобразований можно собрать в одну единую матрицу путем их перемножения. В дальнейшем ее можно использовать для нахождения координат точки в пространстве отсечения, исходя из ее локальных координат :

(2)

## 1.3 Углы Эйлера

Для описания поворотов камеры на сцене используются углы Эйлера. Они позволяют быстро и эффективно создать матрицу вида для отображения сцены.

Углы Эйлера – это совокупность трех углов, описывающая поворот тела в трехмерном евклидовом пространстве. Они описывают последовательность из трех поворотов системы координат, которые позволяют получить текущее ее положение из любого другого. Обозначим как начальное положение системы координат, а конечное как .

Будем называть пересечение осей xy и XY линией узлов N.

Тогда углом или углом процессии назовем угол между осью X и линией узлов. Углом или углом нутации угол между осями z и Z. Углом или углом собственного вращения угол между осью X и линией узлов.

Для описания последовательности из трех поворотов используются матрицы вращения. Матрица вращения вокруг оси Z на угол  вычисляется по формуле 3.

(3)

Матрица вращения вокруг оси X на угол вычисляется по формуле 4:

(4)

Матрица вращения вокруг оси Z на угол вычисляется по формуле 5.

(5)

Визуализация применения данных матриц поворотов и местоположения линии узлов в данной модели изображена на рисунке 1.

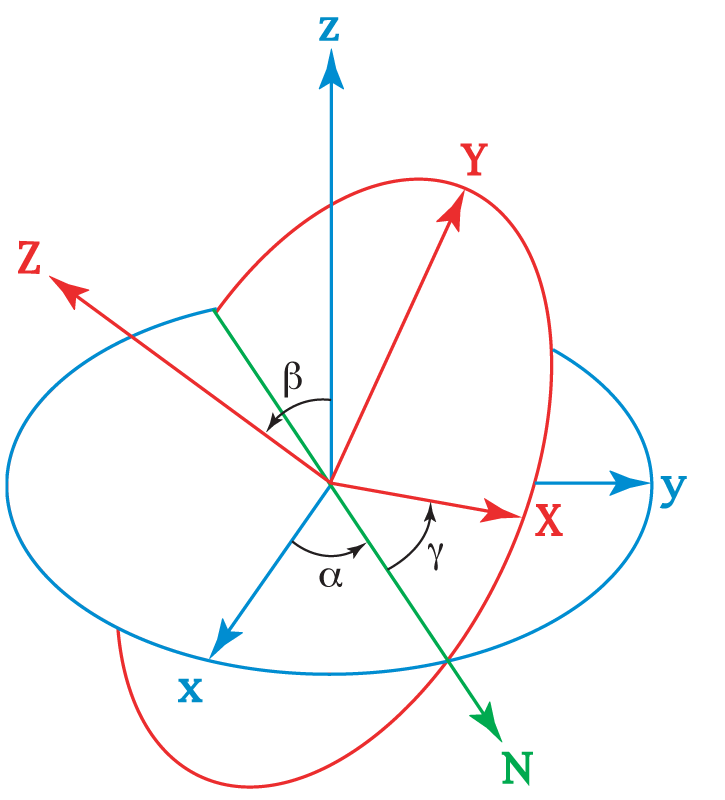


Рисунок 1 ­ Визуализация углов Эйлера

Для вычисления координат точек в требуемой системе координат используется матрица R, вычисляемая по формуле 6, являющаяся произведением матриц, вычисленных по формулам 3, 4 и 5.

(6)

Матрица, вычисленная по формуле 6 используется для вращения сцены и объектов на ней согласно рисунку 1.

# 2 Используемые алгоритмы и математические модели

## 2.1 Модель освещения Фонга

Для создания относительно реалистичной модели освещения в OpenGL используются модели освещения, основанные на физических законах. Одной из таких моделей является модель освещения Фонга (Phong lighting model), так же называемая моделью отражения Фонга (Phong reflection model) или моделью затенения Фонга. Она является относительно нетребовательной к ресурсам моделью освещения. Модель является локальной, то есть такой, которая учитывает только свойства конкретной точки модели и источников освещения, окружающих ее. Игнорируются эффекты рассеивания, отражения и линзирования от соседних объектов. Это позволяет использовать данную модель в нетребовательных к ресурсам системы приложениях реального времени. Иные же модели, учитывающие перечисленные эффекты, ведут к очень большому росту накладных расходов.

При расчете освещения используются три основные компоненты освещения: фоновая (ambient), рассеянная (diffuse) и глянцевых бликов (specular). Фоновая компонента представляет собой грубое приближение лучей света, рассеянных соседними компонентами. Рассеянная компонента имитирует рассеивание прямого излучения, достигшего требуемой точки. Глянцевая же его отражение. Для расчета интенсивности освещения точки используется следующая формула:

(7)

Где - коэффициенты фонового, диффузного и отраженного освещения соответственно для заданной точки;

- вектор нормали к поверхности заданной точки;

- вектор падающего луча на данную точку (направление на источник освещения);

- направление идеально отраженного от поверхности луча.

В разрабатываемом программном модуле коэффициенты формулы 7 задаются с помощью трех соответствующих им текстур. Формула 7 используется для просчета освещения на каждом из объектов.

## 2.2 Карты нормалей и глубины

Для улучшения качества картинки и реализации некоторых алгоритмов, необходимо иметь и вычислять данные, не связанные с отображаемым на экране пользователя изображением напрямую и используемые для повышения детализации или оптимизации расчетов.

Картой глубины является текстура, которая сохраняет в себе информацию об расстоянии до фрагментов объекты сцены относительно некоторой точки. Данная карта является фундаментальной в трехмерной графике и используется во время визуализации практически любой трехмерной сцены: благодаря ей дальние фрагменты сцены не перекрывают более близкие. Также она используется во время визуализации тумана, теней, контура объекта.

Карта нормалей используется при рельефном текстурировании. При помощи нее изменяется нормаль текущего фрагмента изображения, позволяя создавать углубления и неровности без использования дополнительных полигонов. Например, используя карту нормалей, можно сделать вмятину на каким-либо объекте, не используя дополнительных полигонов и ресурсов для их обработки. Использование данных карт позволяет сильно экономить ресурсы во время визуализации модели, повышать реалистичность сцены.

Карты нормалей связаны с картами высот – они позволяют дополнительно моделировать изменение высоты заданного пикселя. Таким образом повышая детализацию неровностей, заданных картами нормалей. Таким образом, карты нормалей не изменяют координат диффузных текстур, но изменяют сторону, с которой мы на нее смотрим, а карты высот наоборот – изменяют координаты текстуры, но не сторону, на которую мы смотрим. Их совместное использование позволяет моделировать тени максимально реалистично.

## 2.3 Всенаправленные карты теней

Тень является результатом отсутствия света, которое может появиться, если, например, между источником света и объектом находится непрозрачное препятствие. В отсутствии теней восприятие сцены осложняется: зрителю намного тяжелее увидеть ее глубину, осознать расположение объектов в пространстве. С тенями расположение объектов относительно друг друга станут намного более очевидными и, в некоторых случаях, заметным. Данный эффект демонстрируется на рисунке 2. С обоих сторон рисунка изображена одна и та же сцена с одного и того же ракурса. Однако, слева она изображена без использования теней, а справа они используются.

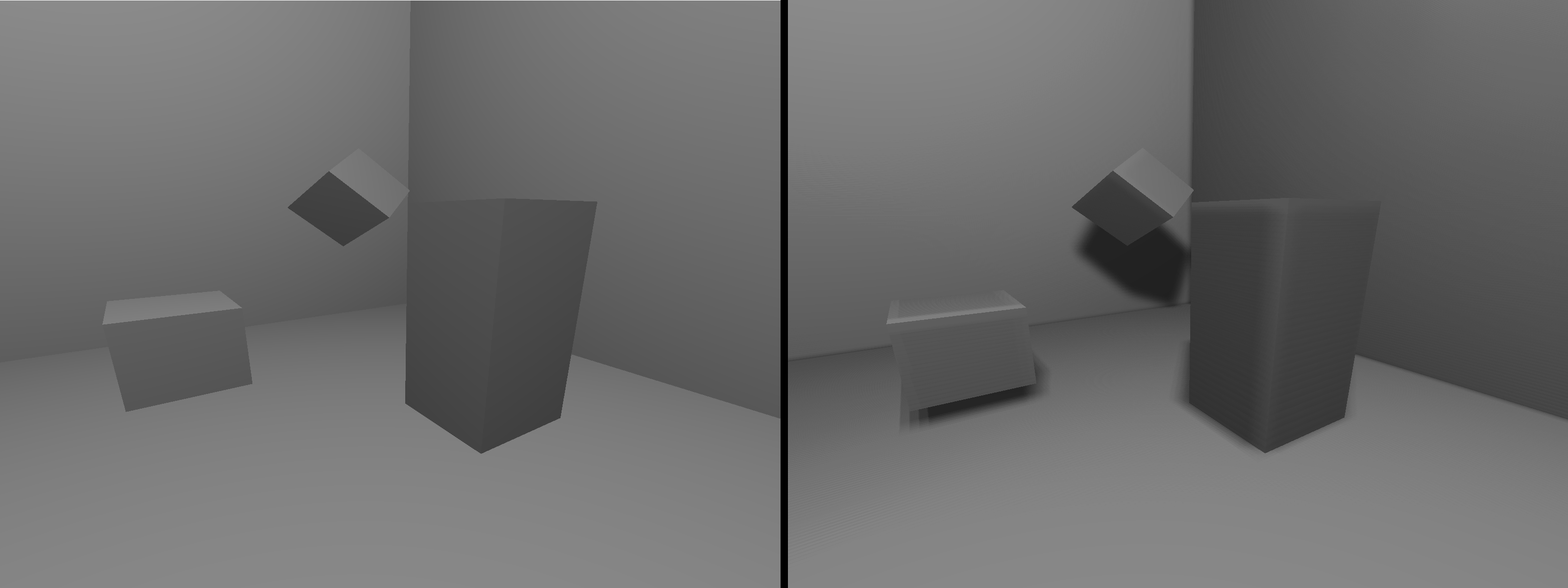


Рисунок 2 ­ Сравнение сцены с тенями и без

Реализация теней является достаточно сложным процессом, а их вычисление сильно нагружает систему. Сейчас не существует эффектного как по памяти, так и по быстродействию алгоритма точного вычисления теней в режиме реального времени, однако существует множество методов приближенного вычисления теней. Одними из таких методов являются методы теневых карт и методы всенаправленных теневых карт, базирующийся на нем.

Метод теневых карт является широко распространённым методом, который относительно прост в реализации и нетребователен к ресурсам системы. Идея заключается в том, что происходит визуализация сцены с точки зрения источника освещения в буфер глубины, который сохраняет удаленность элементов от источника освещения. После визуализации сцены, буфер сохранит в себе информацию о глубине ближайших к нему фрагментов сцены. Свет падает только на те элементы, чья глубина равняется сохраненной в буфере, в ином случае – свет не достигает фрагмента и не учитывается.

В данной работе алгоритм представлен на примере направленного источника света. Это источник освещения, расположенный столь далеко от визуализируемой сцены, что лучи, идущие от него, можно считать параллельными, а его удаление бесконечным. Например, им может быть или луна.

Метод всенаправленных теневых карт используется для обычных, не бесконечно удаленных, источников освещения – точечных, прожекторов, областей и других. Рассмотрим его на примере точечного источника освещения.

В отличии от обычной теневой карты, сцена визуализируется в кубическую текстуру глубины. То есть, происходит 6 визуализаций сцены – по одному для каждой стороны куба. Заметим, что для каждой визуализации сцены вычисляется матрица вида. После визуализации текстура сохраняет информацию о глубине ближайших к источнику освещения объектов, после чего происходят аналогичные предыдущим проверки и вычисления.

## 2.4 Эффекты постобработки

Для того, чтобы предать полученной картинке большую реалистичность или специальный эффект используются эффекты постобработки. Например, им может быть размытие или предания изображению какого-либо оттенка.

Для работы эффектов постобработки необходимы матрицы свертки.

Ядро или матрица свертки – это небольшой матричный массив данных, центрированный на данном пикселе изображения. Он умножает соседние пиксели на заданное значение, складывает их и записывает данные в текущий пиксель изображения. Существует огромное количество самых разнообразных ядер, которые позволяют достигнуть разнообразных эффектов.

Для создания эффекта размытия можно использовать матрицу, полученную из ядра 8:

 (8)

Матрица, получанная из ядра 8, используется для размытия изображения. Деление на 16 необходимо по причине того, что сумма элементов ядра равняется 16 и, чтобы изображение не стало ярче, необходимо разделить ядро на сумму его элементов.

Правильный выбор ядер дает возможность создать самые разнообразные эффекты, начиная от выделения границ отдельных объектов на сцене и заканчивая эффектами черно-белых изображений или серпии.

# 3 Используемый инструментарий

## 3.1 Описание OpenGL 3.3

OpenGL является платформонезависимой спецификацией программных интерфейсов, использующих двумерную и трехмерную компьютерную графику с ускорением на GPU. Областью ее применения являются САПР, виртуальная реальность, компьютерные игры, спецэффекты для кино.

Так как OpenGL является лишь спецификацией, определяющей и описывающей программные интерфейсы, реализация данных интерфейсов лежит на сторонних разработчиках. Обычно ими являются изготовители видеоадаптеров. После связывая используемого языка программирования и реализации интерфейса, можно использовать возможности библиотеки.

Для поддержания современных технологий, которые используются в новых моделях графических карт, используется механизм расширений. Для таких случаев предусмотрены константы и макросы, проверяющие наличие данного расширения в реализации библиотеки. Расширения, уже поддерживаемые многими компаниями, обычно входят в обновленные спецификации библиотеки.

Определим некоторые термины и понятия:

Мешем или полигональной сеткой является совокупность вершин и ребер, определяющих форму и грани некоторого трехмерного многогранного объекта. Гранями обычно являются треугольники, но, в некоторых случаях и реализациях, могут быть четырехугольники или иные многоугольники.

Текстурой будем называть растровое изображение, накладываемое на полигональную модель. Они используются для придания рисунка изображению, кодирования в них свойств отдельных его элементов.

Кадровым буфером будем называть область памяти, хранящую в себе некоторый визуализированный кадр экрана. Обычно, кадр сохраняется в структуру, которая содержит в себе изображение текстуры, которая будет выведена на экран.

## 3.2 Связанные библиотеки

Во время реализации программного модуля использовались некоторые сторонние библиотеки. Они необходимы для придания ему кроссплатформенности, облегчения некоторых вычислений.

GLEW (OpenGL Extension Wrangler Library) – является кроссплатформенной библиотекой, написанной на языке C, для нахождения и определения поддерживаемых расширений OpenGL на целевой платформе.

GLFW – кроссплатформенная библиотека, написанная на языке C, которая используется для создания и поддержания окна; обработки его событий. Стоит отметить тот момент, что данная библиотека используется исключительно для создания и управления окном, и она не поддерживает создания графических интерфейсов, отображения текста и иных более высокоуровневых действий.

GLM (Graphic Library Mathematic) – библиотека, используемая для проведения математических вычислений. Она позволяет вычислять матрицы трансформаций, вида, проекции; оперировать векторами и векторными вычислениями.

Assimp – библиотека, используемая для загрузки трехмерной сцены из файлов самого разнообразного формата.

SOIL – библиотека, используемая для загрузки растровых изображений из файлов самого разнообразного формата.

## 3.3 Инструменты отладки

Во время написания программы используется два подхода к отладке программы. Первый из них заключается в написании специальных шейдерных программ, которые способны визуализировать промежуточные результаты вычислений. Второй же предполагает использование сторонних программ для отладки приложения.

Стоит отметить, что отладка приложения, выполняемого на графическом процессоре является очень трудоемким процессом – сбор логов, просмотр переменных и некоторые другие привычные функции могут быть или невозможны, или сильно затруднены.

Программа GLSL lang позволяет произвести проверку исходного кода, проверяя синтаксис языка и выдавая сообщения-предупреждения (warning) на потенциально небезопасных участках кода.

Программа RenderDoc позволяет получать и анализировать отдельные кадры приложения. Можно просмотреть состояние конвейера OpenGL, используемые текстуры и выполняемые команды на видеопроцессоре.

Самым мощным из доступных инструментов является NVIDIA Nsight. Это официальный инструмент для профилирования и отладки шейдерных программ на видеоадаптерах от компании NVIDIA. Его использование позволяет легко определить распространенные ошибки программирования, провести анализ визуализации отдельного кадра, замерить производительность приложения вовремя визуализации.

# 4 Структура программного модуля

## 4.1 Объектно-ориентированная оболочка над OpenGL

Для облегчения использования библиотек, была написана Объектно-ориентированная оболочка над библиотекой OpenGL. Использование классов позволяет структурировать код, не смешивать процедурный подход языка C и объектно-ориентированный языка C++. Кроме того, повышается безопасность кода и уменьшаются шансы на утечку памяти – оперирование голыми номерами объектов намного более опасно, чем выделение ресурсов в конструкторе и их возврат в деструкторе.

В программе данный модуль представлен в пространстве имен Engine::Render::API. В нем содержатся классы-обертки для управления выделенными ресурсами и создания интерфейсов для текстур, мешей, кадровых буферов и иных объектов. На него опирается следующий элемент программного модуля, использующий данные элементы для построения новых, более сложных объектов.

## 4.2 Описание сцены и взаимодействия с ней

На данном уровне определяются классы для описания сцены, ее узлов, прикладных шейдерных программ и иных объектов. Присутствуют и интерфейсы для библиотек GLFW, Assimp и SOIL, что позволяет изменить библиотеку, используемую во время написания прикладного приложения. Пользователь модуля может легко его дополнять и изменять, наследуясь от описанных на данном уровне классов. Для хранения классов этого модуля используется пространство имен Engine::Render.

Для хранения сцены используются классы Scene и SceneNode. Scene представляет собой класс, хранящий все меши, источники освещения, текстуры и иные ресурсы сцены, а SceneNode – класс, сохраняющий в себе указатели на эти ресурсы в конкретном узле сцены. Во время работы пользователь может изменять и модифицировать узлы, удалять их и добавлять новые.

Для описания шейдерных программ используются классы, унаследованные от класса API::ShaderProgram. Они буфферизуют данные шейдеров для более быстрой и прямой обработки. Пользователю предлагается максимально удобный и комфортный интерфейс для реализации классов для оптимизации своих шейдерных программ. Для этого необходимо лишь унаследовать свой класс от класса API::ShaderProgram или от одного из классов стандартных шейдеров. На рисунке 1 представлена схема возможного наследования от данного класса – LightShadowShader используется для вычисления теневых карт; ModelShader – для описания любой программы, визуализирующей сцену; PosteffectShader – для любых эффектов постобработки; SkyboxShader – для визуализации скайбокса. Пользователь может легко определить свои классы для новых типов шайдерных программ. Диаграмма классов представлена на рисунке 3.

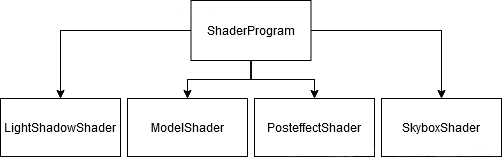


Рисунок 3­ ‒ Диаграмма классов-наследников API::ShaderProgram

Разработчик может допустить возможность загрузки пользовательских шейдерных программ, добавив возможность выбора файла, из которого произойдет их загрузка во время работы программы.

Отдельным достоинством модуля является то, что его относительно легко совместить с самыми разнообразными графическими интерфейсами пользователя – Windows API, GTK, QT и иные, поддерживающие встраивание областей, визуализируемых посредствам графических библиотек.

На рисунке 4 изображена схема композиций классов и хранимых ими данных, используемых во время визуализации сцены. Сцена представляет собой дерево, каждый узел которого может содержать в себе меш и освещение, и обязательно содержит в себе матрицу трансформации. Освещение и материал содержат все необходимые данные для реализации модели освещения Фонга. Меш – данные для своего представления в программе и связанные с ним буфера. Объекты освещения, мешей и материалов хранятся в классе Scene, а указатели на них в остальных классах.

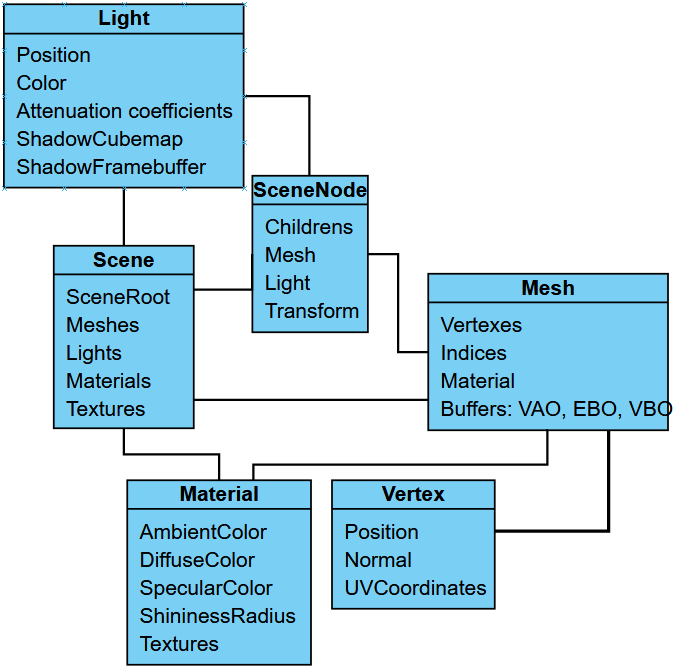


Рисунок 4 ‒­ Представление сцены в модуле

ЗАКЛЮЧЕНИЕ

Во время выполнения работы были получены навыки практической работы с графическим API OpenGL, языком программирования C++, знания общих принципов работы низкоуровневых графических библиотек, изучены алгоритмы работы с графикой.

На основе полученных знаний был разработан графический программный модуль, используемый для визуализации двумерных и трехмерных сцен.

В дальнейшем модуль планируется использован в реальном приложении, с использованием более точных и быстрых алгоритмов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Боресков Расширения OpenGL (+ CD-ROM) / Боресков. - М.: БХВ-Петербург, 2014. - 925 c.

2 Девис, Т. OpenGL. Руководство по программированию / Т. Девис, Д. Шрайнер, Дж. Нейдер, и др.. - М.: СПб: Питер, 2010. - 624 c.

3 Карпов, Б. C++: специальный справочник / Б. Карпов, Т. Баранова. - М.: СПб: Питер, 2016. - 480 c.

4 Рост OpenGL. Трехмерная графика и язык программирования шейдеров / Рост, Дж Рэнди. - М.: СПб: Питер, 2011. - 432 c.

5 Khronos Group. OpenGL® 4.5 Reference Pages [электронный ресурс] // The Khronos Group Inc. URL <https://www.khronos.org/registry/OpenGL-Refpages/gl4/> (дата обращения 10.10.2020)

# ПРИЛОЖЕНИЕ

Тут код. Но он, правда, сюда не влезет весь от слова совсем.