Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**КУРСОВАЯ РАБОТА**

**Моделирование решения Кубика Рубика 3x3**

по дисциплине «Структуры данных»

Выполнили

студенты гр. 5151003/30002 Д. А. Плотников

И. А. Штарев

Руководитель

асс. преподавателя И. Д. Панков

«07» июня 2024 г.

Санкт-Петербург

2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc168422588)

[1. ОПИСАНИЕ РЕШЕНИЯ 4](#_Toc168422589)

[1.1. ГРАФИЧЕСКАЯ ЧАСТЬ 5](#_Toc168422590)

[1.1.1. ГЛАВНОЕ ОКНО 6](#_Toc168422591)

[1.1.2. ПРЕДСТАВЛЕНИЕ КУБИКА И ЕГО ОТРИСОВКА 9](#_Toc168422592)

[1.1.3. ОКНО ВВОДА КУБИКА 12](#_Toc168422593)

[1.1.4. СИНХРОНИЗАЦИЯ КУБИКОВ 14](#_Toc168422594)

[1.2. АЛГОРИТМ РЕШЕНИЯ КУБИКА 15](#_Toc168422595)

# ВВЕДЕНИЕ

Кубик Рубика – культовая головоломка, покорившая умы миллионов людей по всему миру. С виду простая конструкция, состоящая из цветных квадратов, таит в себе невероятное количество комбинаций и алгоритмов, позволяющих достичь заветного результата – собрать все цвета на своих гранях.

Данная курсовая работа посвящена разработке алгоритма, способного решить кубик Рубика, состояние которого задаётся пользователем. В работе рассматриваются различные подходы к решению головоломки, анализируются их преимущества и недостатки.

В рамках исследования был разработан и реализован алгоритм, основанный на методе обхода в ширину.

Практическая значимость данной работы заключается в создании универсального решения для сборки кубика Рубика, доступного для использования пользователями с различным уровнем подготовки. Программа может быть использована как в развлекательных целях, так и в качестве инструмента для изучения алгоритмов решения головоломки.

# 1. ОПИСАНИЕ РЕШЕНИЯ

Приложение «Rubik Solver» используется для решения кубика Рубика с состоянием, заданным пользователем из файла или вручную (через интерфейс ввода состояния кубика).

После ввода состояния кубика пользователю предлагается нажать на кнопку «решить» для сохранения в файл последовательности поворотов, необходимых для решения заданного им кубика. Решение можно загрузить и из заранее подготовленного файла с последовательностью поворотом.

После задания состояния кубика и получения необходимой последовательности действий пользователю предлагается просмотреть решение пошагово, вызывая каждый новый шаг очередным нажатием кнопки «следующий шаг». Повороты кубика на экране сопровождаются выведением информации о соответствующем повороте.

После просмотра всех шагов пользователю сообщается, что решение завершено.

# 1.1. ГРАФИЧЕСКАЯ ЧАСТЬ

Для отрисовки графической части разработанного решения была использована библиотека OpenGL в связке со вспомогательными библиотеками GLFW и GLAD. Самый очевидный ход при отрисовке окон в OpenGL – одинарная буферизация. При ней все изменения в кадре сразу же отображаются в экранном буфере, что может привести к мерцанию экрана. Для разрешения этих ситуаций в решении использована двойная буферизация. При ней вместо одного буфера кадра используются два:

* Задний буфер (back buffer): скрытый от пользователя буфер, где происходит рисование объектов.
* Передний буфер (front buffer): буфер, который в данный момент отображается на экране.

Все операции рисования происходят в заднем буфере, невидимом для пользователя. Как только кадр полностью отрисован в заднем буфере, происходит мгновенный обмен (swap) между передним и задним буферами. Задний буфер становится видимым, а передний становится доступным для рисования следующего кадра. При двойной буферизации достигается целостность кадров: пользователь видит только полностью отрисованные кадры.

# 1.1.1. ГЛАВНОЕ ОКНО

Первое, что видит пользователь при запуске приложения – главное окно. Главное окно представлено на рисунке 1.

A screenshot of a game

Description automatically generated

Рисунок 1 – Главное окно разработанного приложения

Как можно пронаблюдать по рисунку 1, главное окно состоит из кубика и подписанных кнопок управления. Преставление кубика в памяти и процесс его отрисовки описаны в последующих главах.

Кнопка R отвечает за приведение кубика к изначальному (собранному) виду.

Цветные кнопки без надписей рядом с кнопкой R отвечают за повороты соответствующих сторон по и против часовой стрелки. Если нажать на одну из таких кнопок левой клавишей мыши, соответствующая сторона повернется по часовой стрелке, если правой – против часовой.

Кнопка O отвечает за открытие файла с шагами решения. Если у пользователя уже есть готовое решение для кубика, он может нажать на эту кнопку, далее откроется системный диалог выбора файла. После выбора файла он загружается в программу, становясь готовым для исполнения. Для реализации файлового диалога в Windows используется системная функция GetOpenFilename, в Linux используется встроенная в стандартный пакет Ubuntu утилита zenity. В файле с шагами каждое действие записывается в новой строке. Далее записывается название стороны в соответствии с общепринятой нотацией (F – Front, B – Back, U – Up, D – Down, L – Left, R – Right). Допускаются как строчные, так и прописные буквы, однако при решении куба в файле с шагами используются прописные. Далее опционально записывается требуемое число поворотов. Далее опционально записывается символ ‘'’ (одинарная кавычка), являющийся показателем того, что соответствующий поворот происходит против часовой стрелки. Пример файла с шагами представлен на рисунке 2.

A screenshot of a computer

Description automatically generated

Рисунок 2 – Пример файла с шагами

Кнопка N исполняет следующий шаг из открытого файла с шагами, выводя инструкцию к шагу наверху окна. В случае если файл не открыт, либо в случае если все шаги из открытого файла исполнены, выводится соответствующее сообщение.

Кнопка С открывает окно ввода кубика, который необходимо решить. Вызов этого окна переключает так называемый контекст, делая главное окно неактивным. Окно ввода кубика описано в последующих главах.

Кнопка S решает кубик и сразу предлагает пользователю сохранить решение в файл с шагами, а далее загружает этот файл в программу. После решения кубика пользователь может последовательно исполнить шаги решения, нажимая на кнопку N. Алгоритм решения кубика описан в последующих главах.

# 1.1.2. ПРЕДСТАВЛЕНИЕ КУБИКА И ЕГО ОТРИСОВКА

В программе используется два представления кубика: в виде одномерного массива и в виде двумерного. Такое решение было принято по причине того, что для отрисовки кубика с использованием OpenGL более оптимальным является линейное представление, в то время как решение с помощью метода обхода в ширину становится более оптимальным с точки зрения программного кода именно при использовании представления в виде матрицы. В обоих случаях кубик хранится в памяти в виде структуры, поля которой отражают цвета каждого квадрата на каждой грани. Порядок записи квадратов для каждой грани линейного представления изображен на рисунке \_. Индексы двумерных массивов, указывающие на цвета соответственных квадратов в матричном кубе, представлены на рисунке \_.

A colorful squares with numbers

Description automatically generated

Рисунок \_ – Индексы квадратов линейного куба

A colorful squares with numbers

Description automatically generated

Рисунок \_ – Индексы квадратов матричного куба

При запуске программы экземпляры этой структуры создаются в виде глобальных переменных. Хотя использование глобальных переменных и является небезопасной практикой, в данной случае решение об их использовании является оправданным, так как это обеспечивает удобный доступ к редактированию кубика из любого места программы.

Важно отметить, что в программе установлено чёткое соответствие между общепринятыми обозначениями сторон и их цветами:

U (Up) – Синий

L (Left) – Оранжевый

F (Front) – Белый

R (Right) – Красный

B (Back) – Желтый

D (Down) – Зеленый

В связи с этим все повороты, предлагаемые программой, следует совершать на кубике, ориентированном именно таким образом.

Пользователю не дается возможность выбрать цвета центральных квадратов при вводе начального состояния кубика с помощью мыши как раз по причине того, что соответствие сторон и их цветов строго зафиксировано.

Оба кубика хранятся в памяти программы, их состояния синхронизируются при каждом изменении состояния кубика. Синхронизация кубиков описана в последующих главах.

Отрисовка кубика происходит в изометрии: пользователю видны три смежных стороны, чего вполне достаточно для полноценного отображения любого поворота. За отрисовку кубика отвечают функции drawSide и drawSquare. Первая принимает на вход информацию о том, какую сторону следует отрисовать (левую, правую или верхнюю) и массив цветов, соответствующих этой стороне. Вторая вызывается первой девять раз, по каждому на каждую клетку. Эта функция принимает на вход координаты угла клетки и три компоненты цвета этой клетки.

Состояние кубика обновляется каждый кадр, а хранение его состояния в глобальной переменной позволяет делать повороты сторон из любой точки программы, моментально получая отрисованный результат.

# 1.1.3. ОКНО ВВОДА КУБИКА

На рисунке 3 представлен внешний вид окна ввода кубика.

A black and white sign with black letters

Description automatically generated

Рисунок 3 – Окно ввода кубика

Бирюзовым цветом отмечены квадраты, цвет которых еще не определен. Фиолетовым цветов отмечен квадрат, ввод цвета которого в данный момент ожидается от пользователя. Цвет выбирается с помощью одной из цветных кнопок, расположенных в правом нижнем углу окна ввода кубика. После указания цветов всех квадратов окно ввода кубика автоматически закрывается, а введенный пользователем кубик переносится на главное окно.

Пользователь может выбрать кубик из файла, нажав кнопку в F в левом нижнем углу окна ввода кубика. Нажатие кнопки создает системный диалог выбора файла аналогично диалогу открытия файла с шагами. Файл с состоянием кубика состоит из шести строк, порядок ввода цветов квадратов изображен на развёртке куба, представленной на рисунке 4 (индексация начинается с нуля). Сначала вводится синяя сторона, потом оранжевая, потом белая, потом красная, потом желтая, и, наконец, зеленая. Допускаются как строчные, так и прописные буквы. Пример того, как должно записываться состояние кубика в этом файла, представлен на рисунке 5.

A colorful squares with numbers

Description automatically generated

Рисунок 4 – Порядок ввода сторон, ожидаемый от пользователя

A screenshot of a computer

Description automatically generated

Рисунок 5 – Пример файла с состоянием кубика

В случае ввода из файла все квадраты, введенные пользователем вручную, игнорируются, а на главном окне появляется кубик, состояние которого отражено в файле.

# 1.1.4. СИНХРОНИЗАЦИЯ КУБИКОВ

Для синхронизации двух типов кубиков (линейного и матричного) были реализованы две функции, принимающие на вход кубик-назначение и кубик-источник.

Первая функция работает следующим образом: она построчно перебирает квадраты матричного куба, записывая в них цвета из соответствующих индексов линейного куба. Соответствие для каждой грани установлено с помощью массивов, заданных с упором на данные, изображенные на рисунке \_. Эта функция не затрагивает служебные поля матричного куба, так как в линейном имеется лишь информация, необходимая для отрисовки, в то время как служебные поля необходимы только для алгоритма решения куба.

Вторая функция работает обратным образом: она построчно перебирает квадраты матричного куба, записывая их в соответствующие индексы линейного куба, руководствуясь тем же соответствием.

A screen shot of a computer code

Description automatically generated

Рисунок \_ – Соответствие индексов квадратов на развертке куба

# 1.2. АЛГОРИТМ РЕШЕНИЯ КУБИКА

ЗАКОНЧИТЬ