Отчет по лабораторной работе «Разработка приложения для конвертации цветовых моделей CMYK-RGB-HSV»

Волковец Д.А.
3 курс, 3 группа

1 Введение

1.1 Цель работы

Разработать интерактивное приложение для преобразования цветов между тремя цветовыми моделями (CMYK, RGB, HSV) с возможностью интерактивного изменения параметров.

1.2 Задачи

- Изучить теоретические основы цветовых моделей
- Реализовать алгоритмы преобразования между моделями
- Создать графический интерфейс с элементами управления
- Обеспечить реальное время преобразования цветов

2 Теоретическая часть

2.1 Цветовые модели

2.1.1 RGB (Red, Green, Blue)

- **Тип:** Аддитивная модель
- **Применение:** Мониторы, дисплеи, телевизоры
- **Диапазон:** 0-255 для каждого канала
- **Принцип:** Сложение цветов для получения белого

2.1.2 HSV (Hue, Saturation, Value)

- **Тип:** Модель восприятия
- **Компоненты:**
  - **Hue (0-360°)** - тон цвета (положение в цветовом круге)
  - **Saturation (0-100%)** - насыщенность цвета

  - **Value (0-100%)** - яркость цвета

- **Преимущество:** Интуитивно понятна для пользователей

2.1.3 CMYK (Cyan, Magenta, Yellow, Key/Black)

- **Тип:** Субтрактивная модель
- **Применение:** Полиграфия, печать
- **Диапазон:** 0-100% для каждого компонента
- **Принцип:** Вычитание цветов из белого

2.2 Математические основы преобразований

2.2.1 RGB → HSV

$$
\begin{aligned}
V &= \max(R, G, B) \\
S &= \frac{\max - \min}{\max} \quad (\text{если } V \neq 0) \\
H &= 60^\circ \times \text{секторные вычисления}
\end{aligned}
$$

2.2.2 HSV → RGB

Используется шестисекторная модель с учетом яркости (Value)

2.2.3 RGB → CMYK

$$
\begin{aligned}
K &= 1 - \max(R, G, B) \\
C &= \frac{1 - R - K}{1 - K} \\
M &= \frac{1 - G - K}{1 - K} \\
Y &= \frac{1 - B - K}{1 - K}
\end{aligned}
$$

2.2.4 CMYK → RGB

$$
\begin{aligned}
R &= (1 - C) \times (1 - K) \\
G &= (1 - M) \times (1 - K) \\
B &= (1 - Y) \times (1 - K)
\end{aligned}
$$

3 Практическая реализация

3.1 Архитектура приложения

```
color_models/
 CMakeLists.txt
 src/
```

```
    main.cpp
    ColorConverter.cpp
 include/
     ColorConverter.h
```

3.2 Структуры данных

Листинг 1: Структура ColorModels

```
struct ColorModels {
    cv::Vec3b rgb;     // RGB                    (0-255)
    cv::Vec3f hsv;     // HSV                    (H:0-360, S,V:0-100)
    cv::Vec4f cmyk;    // CMYK                    (0-100%)
};
```

3.3 Ключевые алгоритмы

3.3.1 Преобразование RGB → HSV

Листинг 2: Функция rgbToHsv

```
cv::Vec3f ColorConverter::rgbToHsv(const cv::Vec3b& rgb) {
    float r = rgb[2] / 255.0f;
    float g = rgb[1] / 255.0f;
    float b = rgb[0] / 255.0f;

    float maxVal = std::max({r, g, b});
    float minVal = std::min({r, g, b});
    float delta = maxVal - minVal;

    float h = 0, s = 0, v = maxVal;

    if (delta > 0.0001f) {
        s = delta / maxVal;

        if (maxVal == r) {
            h = 60 * fmod(((g - b) / delta), 6.0f);
        } else if (maxVal == g) {
            h = 60 * (((b - r) / delta) + 2);
        } else {
            h = 60 * (((r - g) / delta) + 4);
        }

        if (h < 0) h += 360;
    }

    return cv::Vec3f(h, s * 100, v * 100);
}
```

3.3.2 Преобразование HSV → RGB

Листинг 3: Функция hsvToRgb

```
cv::Vec3b ColorConverter::hsvToRgb(const cv::Vec3f& hsv) {
    float h = hsv[0];
    float s = hsv[1] / 100.0f;
    float v = hsv[2] / 100.0f;

    if (s < 0.001f) {
        uchar gray = static_cast<uchar>(v * 255);
        return cv::Vec3b(gray, gray, gray);
    }

    float c = v * s;
    float x = c * (1 - fabs(fmod(h / 60.0f, 2) - 1));
    float m = v - c;

    float r, g, b;

    if (h >= 0 && h < 60) {
        r = c; g = x; b = 0;
    } else if (h >= 60 && h < 120) {
        r = x; g = c; b = 0;
    } else if (h >= 120 && h < 180) {
        r = 0; g = c; b = x;
    } else if (h >= 180 && h < 240) {
        r = 0; g = x; b = c;
    } else if (h >= 240 && h < 300) {
        r = x; g = 0; b = c;
    } else {
        r = c; g = 0; b = x;
    }

    return cv::Vec3b(
        static_cast<uchar>((b + m) * 255),
        static_cast<uchar>((g + m) * 255),
        static_cast<uchar>((r + m) * 255)
    );
}
```

3.4 Графический интерфейс

3.4.1 Элементы управления

- **12 трекбаров** (по 4 для каждой модели)
- **Визуализация цвета** (прямоугольник с текущим цветом)
- **Текстовое отображение** значений компонентов
- **Интерактивные подсказки**

3.4.2 Логика работы интерфейса

Листинг 4: Обработка изменений

```
//
if (lastChangedModel == "RGB") {
    currentColors = updateFromRgb(newRGB);
} else if (lastChangedModel == "HSV") {
    currentColors = updateFromHsv(newHSV);
} else if (lastChangedModel == "CMYK") {
    currentColors = updateFromCmyk(newCMYK);
}
//
```

4 Результаты работы

4.1 Функциональность приложения

1. **Реальное время преобразования** - изменения мгновенно отражаются во всех моделях
2. **Поддержка трех моделей** - RGB, HSV, CMYK
3. **Различные способы ввода:**
   - Точные значения через трекбары
   - Визуальная обратная связь
4. **Корректные преобразования** между всеми моделями

4.2 Примеры преобразований

4.2.1 Красный цвет

- RGB: (255, 0, 0)
- HSV: (0°, 100%, 100%)
- CMYK: (0%, 100%, 100%, 0%)

4.2.2 Зеленый цвет

- RGB: (0, 255, 0)
- HSV: (120°, 100%, 100%)
- CMYK: (100%, 0%, 100%, 0%)

4.2.3 Синий цвет

- RGB: (0, 0, 255)
- HSV: (240°, 100%, 100%)
- CMYK: (100%, 100%, 0%, 0%)

4.3 Особенности реализации

1. **Полная матрица преобразований** - 6 функций для всех парных преобразований
2. **Обработка граничных случаев** - черный, белый, серые цвета
3. **Предотвращение рекурсивных обновлений** - система отслеживания изменений
4. **Нормализация значений** - приведение к стандартным диапазонам

5 Выводы

- Разработано полнофункциональное приложение для конвертации цветов
- Реализованы корректные алгоритмы преобразования между моделями
- Создан интуитивно понятный графический интерфейс
- Обеспечена работа в реальном времени

Приложение А. Код CMakeLists.txt

Листинг 5: Файл CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)
project(ColorModelsApp)

set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

find_package(OpenCV REQUIRED)

include_directories(${OpenCV_INCLUDE_DIRS})
include_directories(include)

add_executable(ColorModelsApp
    src/main.cpp
    src/ColorConverter.cpp
)

target_link_libraries(ColorModelsApp ${OpenCV_LIBS})
```

Приложение Б. Инструкция по сборке

1. Установить зависимости: OpenCV, CMake
2. Создать директорию сборки:

   ```
   mkdir build && cd build
   ```

3. Сконфигурировать проект:

```
cmake ..
```

4. Собрать приложение:

```
make
```

5. Запустить программу:

```
./ColorModelsApp
```