K. K. Wagh Institute of Engineering Education and Research, Nashik.

Department of Computer Engineering

Class: S. Y. BTech **Div: B**

Course: Object Oriented Programming and Computer Graphics (OOPCG) **Semester: III**

Project Name: Color Converter Application

Team Members:

Name	**Roll No.:**
Vedant Purkar	52
Sanchita Rajurkar	53
Piyush Sanap	57
Snehal Sanap	58

1. Introduction

This project is a mini-project for the subject Object-Oriented Programming and Computer Graphics (OOPCG). It focuses on creating a color converter application using Python. The application allows users to convert colors between different color spaces, such as RGB, HSV, and CMY, providing a practical demonstration of graphical user interface programming in Python. The project leverages libraries such as Tkinter for GUI development and matplotlib for potential visualization capabilities.

2. Objectives

The primary objectives of this project are:

- To develop a user-friendly graphical application for color conversion.
- To implement color space conversion functions (RGB to HSV, HSV to RGB, RGB to CMY, and CMY to RGB).
- To demonstrate the use of Python libraries for GUI development (Tkinter) and data visualization (matplotlib).
- To practice object-oriented programming and understand its application in real-world projects.

3. System Requirements

- Software Requirements:
  - Python 3.x
  - Tkinter library (comes pre-installed with Python)
  - matplotlib library
- Hardware Requirements:
  - A computer with a minimum of 2 GB RAM
  - 200 MB of free disk space

4. Modules and Functionalities

1. Color Conversion Functions:
   - Converts between RGB, HSV, and CMY using the colorsys module.
2. Graphical User Interface (GUI):
   - Uses Tkinter sliders to adjust RGB values interactively.
   - Displays real-time color changes, providing immediate visual feedback.

3. Quiz Section:
    - A quiz module that tests the user’s knowledge of color theory.
    - Includes multiple-choice questions about color properties, conversion methods, and practical applications.
    - Displays the results at the end of the quiz to help users learn and improve.
4. Color Cube Visualization:
    - 3D representation of RGB values on a color cube.
    - Uses matplotlib for rendering a 3D plot that shows the distribution of colors within the RGB color space.
    - Allows users to rotate and zoom the color cube to explore different color representations.
5. Displaying Additional Information:
    - Shows information about color properties, including RGB values, HSV values, and CMY equivalents.
    - Provides educational content on color spaces and their applications.

5. Code Explanation

The main components of the code include:

1. Importing Necessary Libraries:

The code begins by importing essential libraries that serve different purposes:

- Tkinter: This is Python’s standard GUI (Graphical User Interface) library. Tkinter provides tools to create various UI elements like windows, buttons, sliders, and labels. Here, Tkinter is used to build the interface, allowing the user to adjust RGB values through sliders and view the corresponding color in real-time.
- colorsys: This library offers functions for converting between color spaces, specifically from RGB (Red, Green, Blue) to HSV (Hue, Saturation, Value) and vice versa. HSV is another popular color representation, and these conversions are valuable for displaying colors in different formats.
- matplotlib: This library is widely used for data visualization and plotting. In this code, it is used for handling color display and conversion visualization, though this depends on the specific functions utilized.

These libraries collectively support the GUI, color conversions, and real-time color updates.

2. Functions for Color Conversions:

To provide flexibility in color representation, the code includes functions that convert colors between different formats. Here’s a breakdown of the common color conversion functions:

- RGB to HSV and HSV to RGB:
    - RGB and HSV are both color models that represent colors differently. RGB uses Red, Green, and Blue components, while HSV uses Hue (color type), Saturation (color intensity), and Value (brightness).
    - The colorsys.rgb_to_hsv and colorsys.hsv_to_rgb functions facilitate these conversions, allowing users to see the color in the HSV model and better understand variations in hue, saturation, and brightness.
- RGB to CMY and CMY to RGB:
    - CMY (Cyan, Magenta, Yellow) is a subtractive color model commonly used in printing. In contrast, RGB is an additive model used for screens and light-based displays.
    - Conversions between RGB and CMY are done using custom functions:
        - RGB to CMY: Cyan, Magenta, and Yellow are derived by subtracting the RGB values from 1 (for normalized RGB values between 0 and 1).
        - CMY to RGB: RGB values are obtained by subtracting the CMY values from 1, essentially reversing the previous conversion.

These functions enable users to see how colors vary across different models, enriching the learning experience.

3. GUI Setup with Tkinter:

The GUI setup involves creating a window with Tkinter and adding interactive components:

- Sliders for RGB Values:
    - Three sliders are created, each representing one of the RGB components (Red, Green, and Blue). These sliders allow users to change the values of the colors dynamically.
    - Each slider has a range from 0 to 255, which aligns with the RGB color model’s standard range. Changing these values changes the color display, giving users an interactive understanding of how RGB values affect color.

- Real-Time Color Display:
    - A display area is set up within the window to show the resulting color based on the current slider values.
    - As the RGB values are adjusted, the display updates to reflect the new color, providing immediate visual feedback.
- Labels and Instructional Text:
    - Labels for each slider and the display area help users understand the purpose of each control. Additionally, text fields or labels may explain the current color values in different formats, such as RGB, HSV, and CMY.

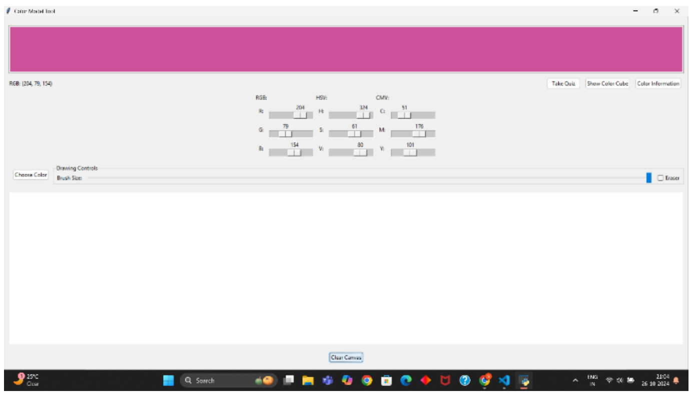
4. Event Handling Functions for Updating the Display Color:

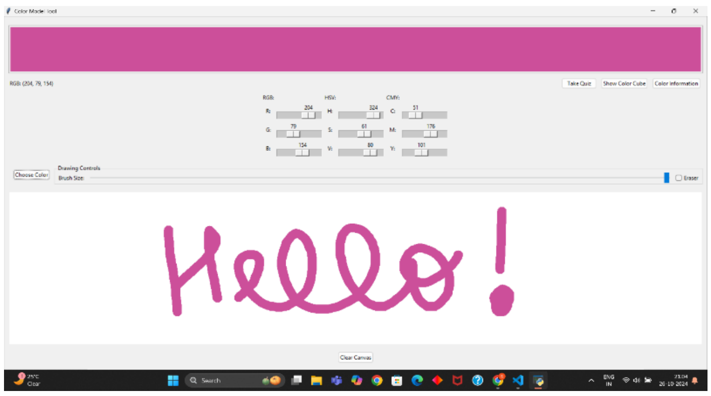
To ensure that the color display updates immediately as the sliders are moved, event-handling functions are implemented:

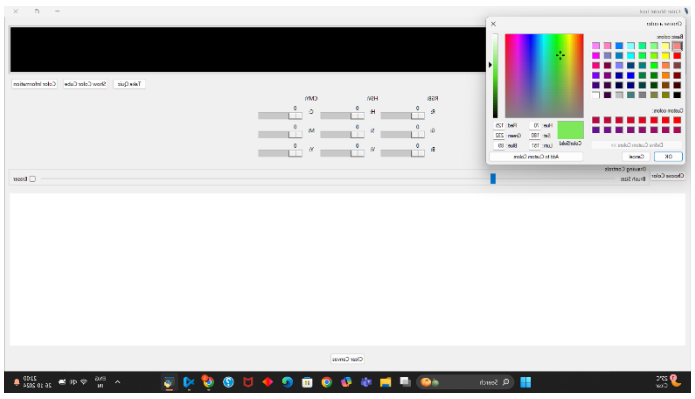
- Color Update Function:
    - This function is triggered whenever there’s a change in any of the RGB sliders.
    - It fetches the current values of the Red, Green, and Blue sliders and combines them to calculate the resulting RGB color.
    - The function then updates the color display area with the new color.
- Conversion Display Functions:
    - Additional functions may calculate and display the color’s HSV and CMY equivalents based on the RGB values.
    - These functions allow the code to show the color in multiple models simultaneously, aiding in understanding how colors differ across RGB, HSV, and CMY.

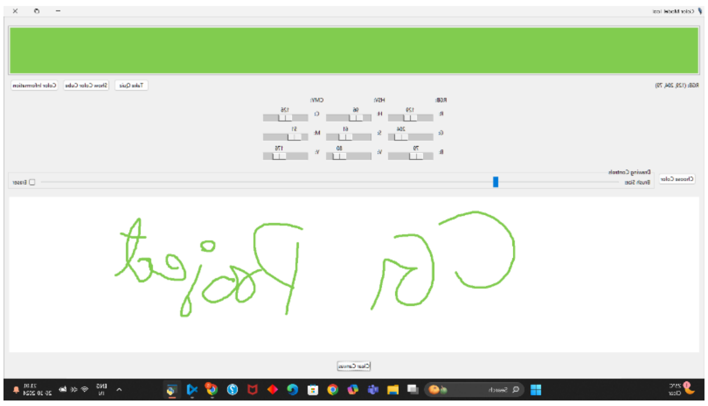
These event handlers make the application interactive, responding to user input in real time. When a user adjusts any of the RGB sliders, the corresponding color in the display area changes, and any displayed HSV or CMY values update as well.

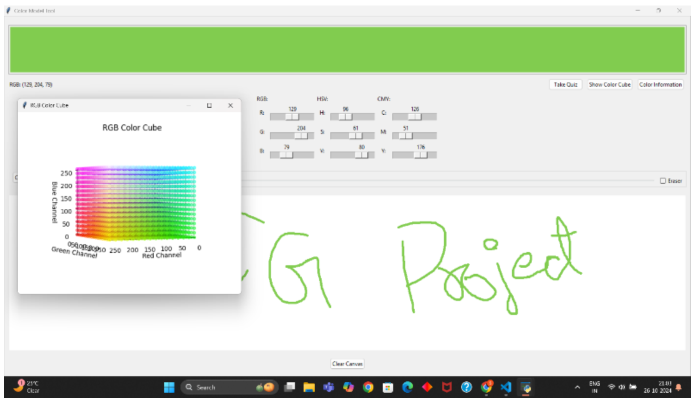
Screenshots

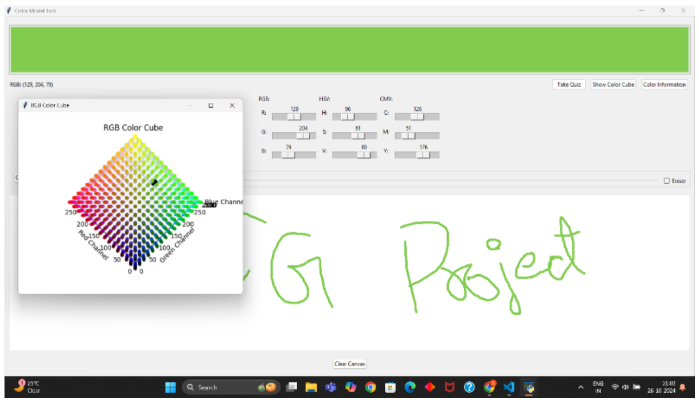


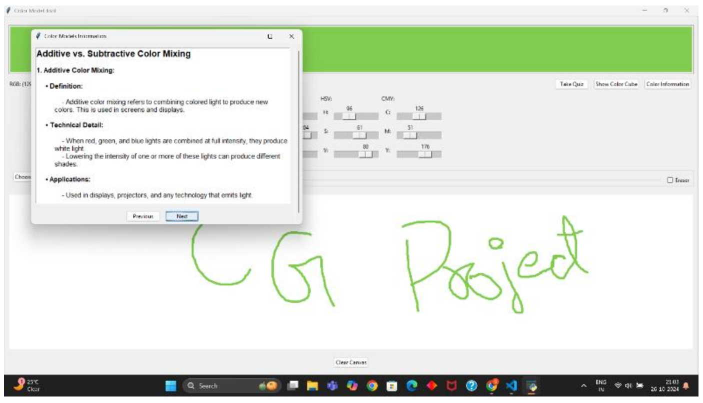












Color Models Information

Additive vs. Subtractive Color Mixing

1. Additive Color Mixing:

- **Definition:**

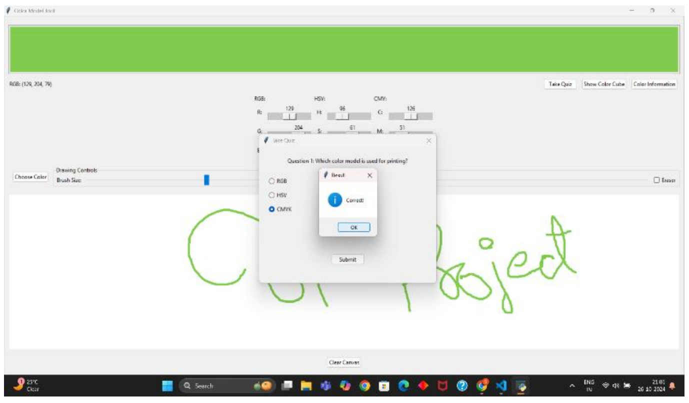
  - Additive color mixing refers to combining colored light to produce new colors. This is used in screens and displays.

- **Technical Detail:**

  - When red, green, and blue lights are combined at full intensity, they produce white light.
  - Lowering the intensity of one or more of these lights can produce different shades.

- **Applications:**

  - Used in displays, projectors, and any technology that emits light.



Question 1: Which color model is used for printing?

RGB HSV [✓ CMYK]

Correct!

6. Future Enhancements

Possible improvements to the project include:

- Adding support for more color spaces, such as HSL and LAB.
- Implementing additional features like color palette generation and color harmonies.
- Enhancing the GUI with more modern styling using frameworks like ttk or custom themes.
- Adding the ability to save and export color data in various formats.

7. Conclusion

This mini-project serves as a practical application of Object-Oriented Programming and Computer Graphics concepts. It demonstrates the development of a user-friendly graphical application that performs color conversion between different color spaces. The project highlights the use of Python libraries for GUI development and visual representation, providing a solid foundation for further exploration.