


Department of

Computer & Software Engineering

Fundamentals of Programming

Instructors:

Asst Prof. Jahan Zeb

Ayesha Khanum

Submitted by:

Musab Farooq(528553)

Rehan Saqib(514082)

Syed Hamza Hatib (531574)

Abdullah Baig(517059)

Jetandar Kumar(503508)

College of E&ME, NUST
Spring semester 2025

Introduction

The project is a simplified, full-screen drawing application inspired by Microsoft Paint, enhanced with an integrated music playback feature. Developed in C++ using the SFML (Simple and Fast Multimedia Library) for GUI and audio, and OpenCV for image manipulation, the application aims to offer an engaging multimedia experience while retaining simplicity and responsiveness.

Unlike standard paint tools, this application allows users to draw with basic shapes, customize colors and thickness, and play selected background tracks. It serves both as a creative tool and a programming exercise in GUI design, object-oriented structuring, multimedia integration, and event handling in C++.

Objectives

- Enable drawing of basic shapes with configurable thickness and color.
- Integrate a music system with multiple selectable tracks.
- Ensure minimal user input errors through exclusive tool selection.
- Maintain low computational overhead for compatibility with modest hardware.

Application Architecture

The codebase is organized modularly using custom header files:

- `shapes.h`: Contains shape-drawing routines like `draw_circle`, `draw_rectangle`, `draw_line` and `draw_arrow_line`.
- `screens.h`: Defines the `draw_line` class used for GUI screen and button management. It allows screen-specific button sets and rendering logic.
- `music.h` Manages background music, including song selection, pause/play functionality, and title displays.

The main logic resides in `main.cpp`, which initializes screens, buttons, and GUI elements, and handles event polling, input, and rendering in a full-screen SFML window.

Key Features

- **Shape Drawing**: Users can draw circles, rectangles, lines, and arrows using respective toolbar buttons.
- **Color & Thickness Customization**: Colors like Red, Yellow, Blue, Green, and Black can be chosen. A textbox interface allows setting thickness (0–9).
- **Exclusive Tool Use**: Only one tool can be activated at a time, avoiding accidental multi-inputs.
- **Canvas Management**:
  - Save the drawing as a PNG using SFML's `RenderTexture`.
  - Clear the canvas with a reset button.
- **Music Playback**: A separate screen allows users to play/pause five preloaded MP3 tracks:

- Idea 10
- Autumn Waltz
- Melancholic
- Magical Overture
- Gymnopedie No. 1

Each track has individual buttons for play and pause with visual labels rendered using SFML's `Text and Font`.

Implementation Summary

GUI Design

- The screen is divided into a toolbar, background area, and drawing canvas.
- Buttons are placed on the toolbar using `setbuttons()` from the `screens` class.
- Drawing area is managed using OpenCV `Mat`, converted to a SFML `Texture` and drawn via a `Sprite`.

GUI Design

- The main loop checks for mouse and keyboard input.
- Tools are activated based on button presses and shape drawing is triggered conditionally.
- Music screen is toggled by switching the `exist` state of `paint` and `musics` screen objects.

Saving Functionality

```
RenderTexture renderTexture;
renderTexture.create(updated_text.getSize().x, updated_text.getSize().y);
renderTexture.clear(Color::Transparent);
renderTexture.draw(sp);
renderTexture.display();
Image image = renderTexture.getTexture().copyToImage();
image.saveToFile("Output Image.png");
```

Music Playback

Handled via SFML's `Music` class. Song selection is dynamically linked to button states:

```
if (musics.buttonexist(3)) {
    bgmusic.openFromFile("Music/idea_10.mp3");
    bgmusic.play();
}
```

Limitations of Existing Systemsasks

- **Time Inefficiency**: Complicated interfaces slow down the user's workflow.
- **Lack of Engagement**: Absence of multimedia features reduces the immersive experience.
- **High Resource Usage**: Many applications require powerful hardware, unsuitable for students or budget users.
- **Input Confusion**: Concurrent tool selection can lead to accidental inputs and user frustration.

System Requirements

Hardware

- **Processor**: Intel Core i3 (5th Gen+)
- **RAM**: 4GB minimum
- **Disk**: 10GB free space
- **Input**: Mouse & Keyboard

Software

- **OS**: Windows 10 or above
- **Dependencies**: SFML, OpenCV
- **Fonts & Assets**: Arial.ttf, Textures, Music files

Results and Achievements

- Successfully implemented a lightweight drawing tool with clear UI structure.
- Integrated five-song playback system with dynamic button feedback.
- Enabled efficient drawing, color customization, thickness adjustment, and PNG saving.
- Designed a modular architecture to separate drawing, GUI management, and music logic.
-

Conclusion

This project exemplifies how core programming concepts in C++ can be applied to develop an interactive multimedia application. The modular codebase promotes maintainability and extensibility, making it suitable for future enhancements like additional shapes, file formats, or even animated interactions. By blending visual creativity with audio experience, this drawing application encourages a deeper user engagement and serves as an excellent educational tool for beginner programmers.