ABSTRACT

The Paint Application is a digital drawing tool that aims to provide users with a versatile platform for creating digital art, illustrations, and designs. It offers a range of drawing tools and features to cater users with varying skill levels.

INTRODUCTION

The Paint Application is a versatile digital drawing tool designed to cater to users creative needs. It provides a wide range of features, including drawing tools, colour options, save and load functionality, and more. This report provides an in-depth overview of the application's design, features, and implementation.

OBJECTIVES

The main objectives of the Paint Application are as follows:

- Provide a user-friendly and intuitive interface for drawing.

- Offer a variety of drawing tools and customization options.

- Enable users to save and load their drawings from files.

- Implement advanced features such as undo/redo and export options

Development Steps for a Paint application:

- Gather Requirements
- Design the System
- Implement the System
- Test the System
- Deploy the System

TECHNOLOGIES USED

- C Programming Language: The entire program is written in the C programming language.
- **stdio.h**: This header file is used for standard input and output operations. It provides functions like printf and scanf for console input and output.
- **stdlib.h**: The stdlib.h header file is used for various standard library functions, including memory allocation and file manipulation.
- **dos.h**: The dos.h header file contains functions for handling interrupts, producing sound, date and time functions.
- **graphics.h:** The graphics.h header file provides access to a simple graphics library that makes it possible to draw lines, rectangles, ovals etc on graphical window.

SYSTEM ARCHITECTURE

Main Function (main): The main function serves as the entry point of the program.

Draw Line (drawLine Function): Allows the user to enter the colour and coordinates. And then draws the lines in the given coordinates.

Draw Circle (drawCircle Function): Allows the user to enter the colour. Asks the user to enter the radius and cooordinates of the circle. And then draws the circle of the given radius and coordinates.

Draw Rectangle (drawRectangle): Allows the user to enter the colour. Asks the user to enter the cooordinates of the rectangle. And then draws the rectangle in the given coordinates.

DESIGN AND IMPLEMENTATION

Front-End Design:

• The front-end design is based on a command-line interface, where users can select options by entering numbers corresponding to their desired actions.

Back-End Design:

• The back-end logic is implemented in the C programming language.

• Each module has its functions for adding, displaying, modifying, and deleting records.

• File handling functions are used to perform operations on data files.

Database Design:

• Data for drawings is stored in separate file.

FEATURES AND FUNCTIONALITY

Canvas Area

The canvas is the primary drawing area where users create their artwork.

Toolbar

The toolbar contains icons for various drawing tools, colour selection, and line thickness adjustment.

Colour Palette

Users can choose colours from a palette to customize their drawings.

Thickness Slider

A slider allows users to adjust the thickness of drawing lines.

File Menu

The file menu provides options for saving and loading drawings.

Colour Selection

Users can choose colours from a palette or input specific colour values.

Line Thickness Adjustment

A slider or input field allows users to adjust the thickness of lines.

Save Drawing

Users can save their drawings to files in various formats (e.g., PNG, JPG).

Load Drawing

Allows users to load previously saved drawings.

File Formats

Decide on file formats for saving and loading drawings (e.g., JSON for storing shapes and properties).

Undo and Redo

Implement undo and redo functionality to correct mistakes or redo actions.

Keyboard Shortcuts

Consider adding keyboard shortcuts for common actions (e.g., Ctrl + Z for undo).

Zoom and Pan

Consider adding zoom and pan features for better navigation on the canvas.

User Guidance

Tooltips

Provide tooltips to guide users on how to use the tools effectively.

Help Section

Include a help section or documentation for more detailed instructions.

File Format

Define the data storage format for saving and loading drawings.

Exporting Drawings

Allow users to export their drawings in different formats, such as PNG, JPG, or SVG.

Supported Platforms

Consider making the application available on different platforms (Windows, macOS, web).

TESTING

1. Unit Testing:

- Unit testing involves testing individual functions and methods in isolation to ensure they perform as expected.
- For this Paint application, unit tests can be created for each function separately.
- For example, you can test the "drawline" function by providing various inputs and checking if the line is drawn.

2. Integration Testing:

- Integration testing verifies that different parts of the system work together seamlessly.
- In this system, integration testing could focus on interactions between functions. For instance, ensuring that drawing line and thickness of line work well together.

3. User Acceptance Testing (UAT)

- UAT involves testing the system from the end-user's perspective to ensure it meets their requirements and expectations.
- In the Paint application, UAT would involve real users using the system for its intended purposes.
- Users would perform tasks such as drawing line, circle and rectangle. The system should be intuitive and fulfill their needs.
- Feedback from users during UAT should be collected and incorporated to improve the system.

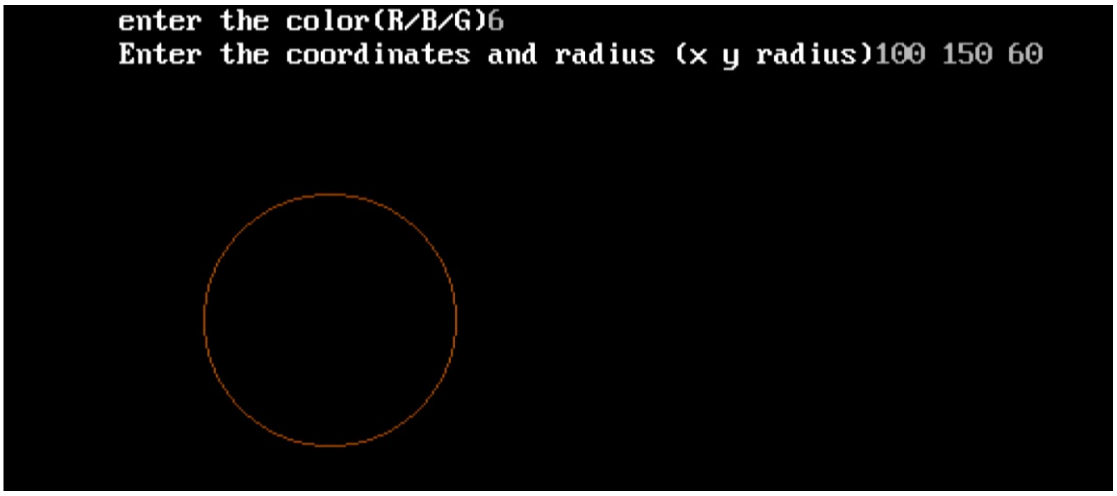
Testing is a critical phase in software development to ensure the system is robust, reliable, and meets user expectations. It helps identify and rectify issues early in the development process, resulting in a more dependable and user-friendly application.

CHALLENGES FACED

Following are some of the challenges, obstacles, and roadblocks encountered during the development of the Paint application project:
->**Data validation**: One of the challenges was ensuring that the data entered by the user was valid. This was done by using input validation techniques to check for invalid choices.
->**File handling:** Another challenge was handling files. This involved opening, reading, writing, and closing files.
->**Error handling**: Error handling was also a challenge. This involved handling errors that occurred during data validation, file handling, and other operation

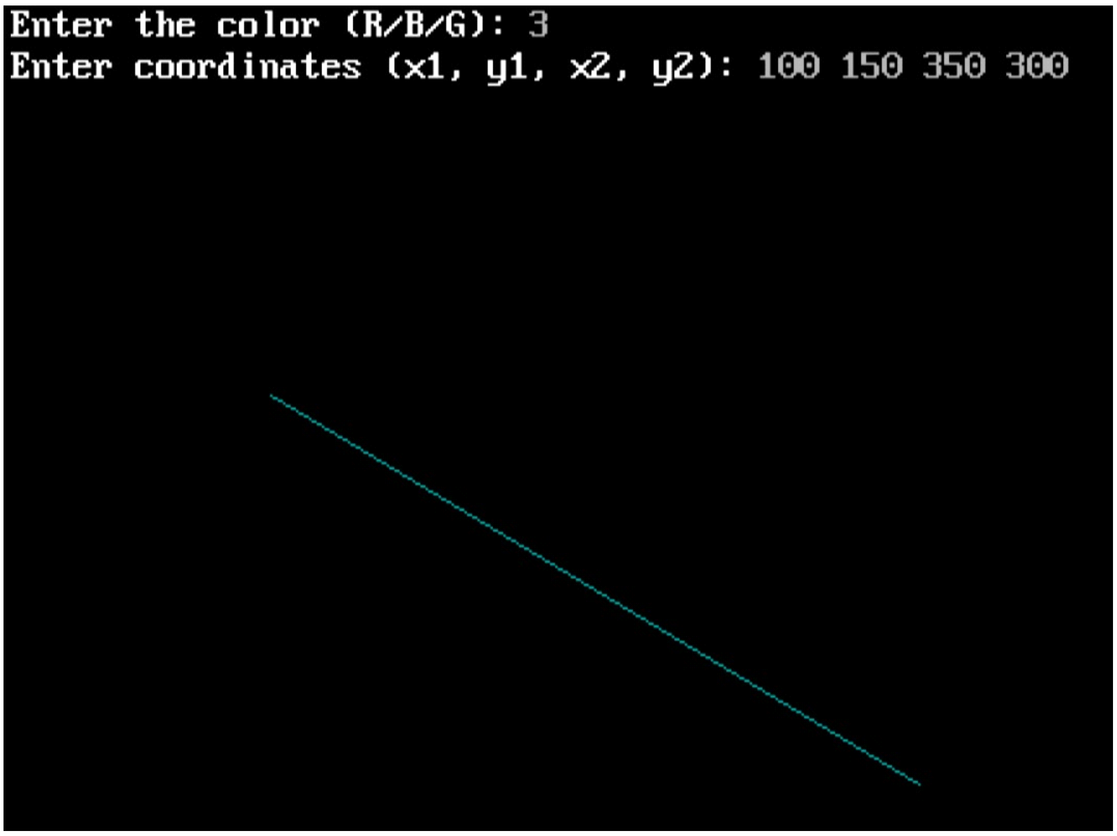
APPENDICES

DRAWING A CIRCLE



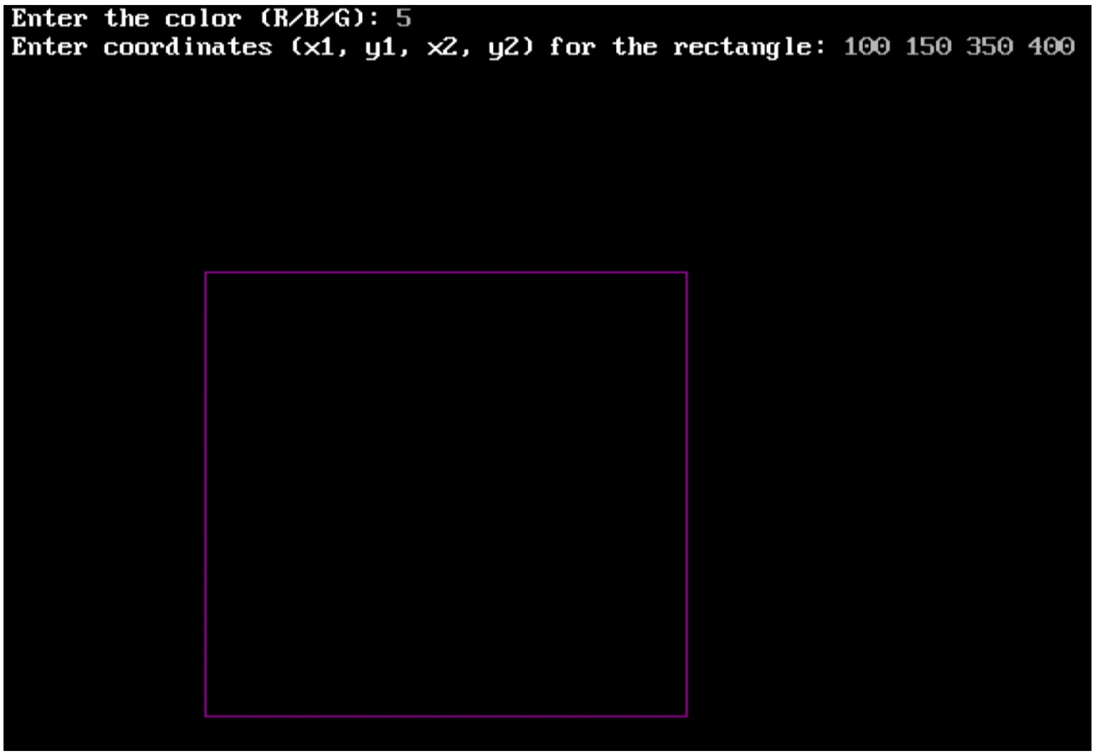
```
enter the color(R/B/G)6
Enter the coordinates and radius (x y radius)100 150 60
```

DRAWING A LINE



```
Enter the color (R/B/G): 3
Enter coordinates (x1, y1, x2, y2): 100 150 350 300
```

DRAWING A RECTANGLE



```
Enter the color (R/B/G): 5
Enter coordinates (x1, y1, x2, y2) for the rectangle: 100 150 350 400
```

CODE SNIPPET:

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <graphics.h>
#include <dos.h>

void drawCircle(int x1, int y1, int radius1)
{
    circle(x1, y1, radius1);
    getch();
}

void drawLine(int x1, int y1, int x2, int y2)
{
    line(x1, y1, x2, y2);
    getch();
}

void drawRectangle(int x, int y, int x3, int y3)
{
    rectangle(x, y, x3, y3);
    getch();
}
```

```c
void main()
{
    int x1 = 0, y1 = 0, x2 = 0, y2 = 0, x = 0, y = 0, x3 = 0, y3 = 0, radius =
0, choice = 0;
    int color = 0;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    printf("Enter the color (R/B/G): ");
    scanf("%d", &color);
    setcolor(color);
    printf("1. Circle\n2. Line\n3. Rectangle\nEnter your choice: ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
        printf("Enter the coordinates and radius (x y radius): ");
        scanf("%d %d %d", &x, &y, &radius);
        drawCircle(x, y, radius);
        printf("Circle drawn successfully\n");
        break;
    case 2:
        printf("Enter coordinates (x1,y1,x2,y2): ");
        scanf("%d %d %d %d", &x, &y, &x3, &y3);
        drawLine(x, y, x3, y3);
        printf("Line drawn successfully\n");
        break;
    case 3:
        printf("Enter coordinates (x1,y1,x2,y2): ");
        scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
        drawRectangle(x1, y1, x2, y2);
        printf("Rectangle drawn successfully\n");
        break;
    default:
        printf("Invalid Choice!!!");
    }
    getch();
}
```

CONCLUSION

The Paint application project was a resounding success, developed with a modular approach and continuous integration, ensuring correctness and easy maintenance.

This system empowers efficient paint application, allowing users to draw line, circle and rectangle. Its user-friendly interface, supported by clear instructions and a helpful guide, accommodates users of all technical levels.

Its accuracy, speed and versatility shines through its comprehensive features, making it a potent tool.

The Paint application is not only user-friendly but also efficient and reliable inventory application with precision. Its versatility makes it a powerful asset.

The system's future is bright, suitable for various drawing application.

In summary, the Paint application project exemplifies successful software development, offering user-friendliness, efficiency, and versatility for drawing application.

The Concludes outlines the design and features of the Paint Application, which aims to provide users with a versatile and user-friendly digital drawing tool. Building this application involves careful consideration of user interface design, drawing tools, functionality, and cross-platform compatibility to create a compelling user experience for artists and designers

REFERENCES

https://practice.geeksforgeeks.org
https://draw.io/