SPŠ Elektrotechnická Ječná

Informační technologie - Programování a digitální technologie

Praha 2, Ječná 517, 120 00 Nové Město

PaintApp

Jan Čaloun

Informační a komunikační technologie

2025

1. Goal of this work

The goal of this project is to design and develop a basic painting application that allows users to draw using tools such as a pencil, eraser, fill bucket, and more. The application should be intuitive, user-friendly, and easy to maintain or expand. It is intended to serve as a simple yet functional graphics editor, suitable for educational purposes or as a foundation for more complex projects.

2. Application Description

This is a basic drawing and image editing application designed for creating and modifying simple digital illustrations. The main goal is to provide users with an intuitive and lightweight tool for sketching, erasing, coloring, and making quick graphical edits without the complexity of professional software.

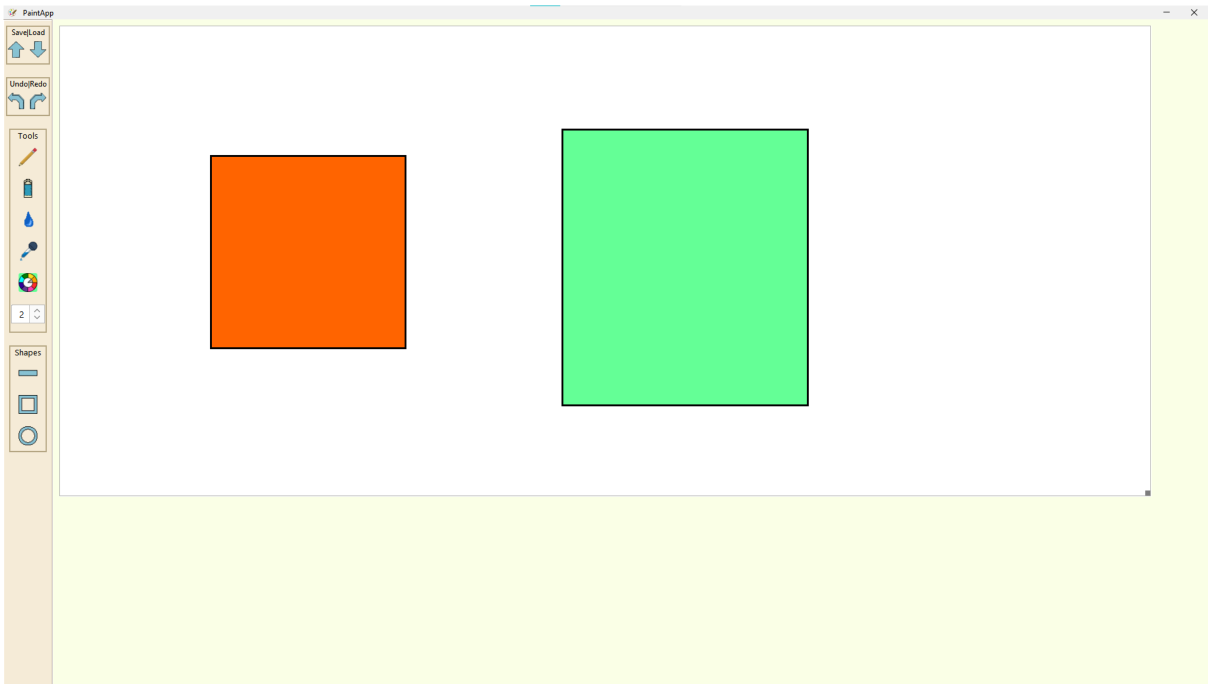
2.1. Flood fill algorithm

The application uses a flood fill algorithm to implement the "fill" tool . This algorithm starts from a clicked pixel and spreads outward, replacing all connected pixels of the same original color with the new color.
The implementation is based on an iterative approach using a LinkedList to avoid the stack overflow risks of recursion.

How it works:

1. The algorithm starts at the pixel clicked by the user.
2. If the pixel's color is different from the target (old) color, it stops immediately.
3. It uses a LinkedList to store points that need to be processed.
4. For each point, it checks if it's inside the canvas and has the old color.
5. If so, it changes the color and adds its four neighboring pixels (up, down, left, right) to the queue.
6. This continues until no more points need to be checked.
7. Finally, the canvas is repainted.

This approach is simple, avoids deep recursion, and works well for small to medium-sized fill operations.



2.2. Features

- **Pencil** – allows freehand drawing on the canvas by tracking mouse movements
- **Eraser** – removes parts of the drawing by making pixels transparent or restoring the background color
- **Fill Too**l – fills enclosed areas with the selected color using a flood fill algorithm
- **Thickness selector** – allows adjusting the thickness of lines and shapes within a range of 1 to 8 pixels for more precise control over drawing styles
- **Color Dropper** – lets the user choose colors from colors that are on canvas
- **Color palette** – lets the user choose colors for drawing and filling
- **Undo/Redo** – enables reversing or reapplying recent changes to correct mistakes or try different effects
- **Save/Load** – enables saving the current image to a file and loading it back later, with only one image stored at a time
- **Line** – draws straight lines between two points on the canvas.
- **Rectangle** – draws rectangular shapes by specifying two corner points
- **Ellipse** – draws elliptical shapes within a defined rectangular area
- **Canvas resizing** – allows user resize canvas at bottom right corner

3. System requirements

The application is developed in the Java programming language, specifically using Java SE Development Kit version 22.0.1 . To compile the code, OpenJDK 22.0.1 is required as the SDK. For the application to run properly, it is necessary to install the FlatLaf – Flat Look and Feel library, specifically version flatlaf-3.6-javadoc.jar. The rest of the application relies on standard Java libraries, making it possible to run the program in any environment that supports Java.

4. Basic structure

The program is written using object-oriented programming and is divided into several classes. These classes work together to ensure the functionality of the application. The application's launch and initialization are handled by the Main class, which starts the MainFrame. The MainFrame contains both the PaintCanvas and the ControlPanel.
The ControlPanel includes all the components used for drawing. The PaintCanvas serves as the canvas where all drawing actions take place. These actions are controlled by the xxxxState classes through methods such as mousePressed(), mouseDragged(), mouseReleased(), and mouseMoved(). These methods, in turn, call specific actions defined in the xxxxTool classes. The entire program is designed to be easily extendable and maintainable, making it simple to add new tools and functionalities in a clear and organized way.

5. Test Data

To manually test the application's core features, we can try several practical actions directly within the interface. We start by selecting the pencil tool, choosing different colors and line thicknesses, and drawing freely on the canvas to observe whether the lines appear correctly and respond smoothly to our input. By switching to the fill tool and clicking inside a closed shape, we can check whether the selected area fills completely without overflowing into adjacent regions.
Using the eyedropper tool, we may click on any part of an existing drawing to see if the active color updates to match the pixel we selected. Finally, we can draw basic shapes such as rectangles and lines, and verify whether they are displayed as expected, with correct size, position, color, and stroke width according to the currently set options.

6. User guide

The application offers a variety of drawing tools, all of which are displayed in the side panel. To use a specific tool, it is necessary to click on the corresponding button and then interact with the canvas using the mouse. The resulting artwork can be saved and later reloaded when the application is reopened. The size of the canvas can be adjusted according to the user's preferences

7. Conclusion

During the course of this project, I encountered a large number of errors, but I was able to resolve all of them. The biggest challenge was adhering to the line limit per class. However, this constraint led me to implement the State Behavior Pattern, which significantly improved the clarity of my code and made it easier to add new tools.
Although I initially intended to implement more tools, it ultimately proved unnecessary given the current scope of the project. Through this experience, I gained a better understanding of working with 2D graphics and using MouseListener().
Overall, I consider this project to be very beneficial and engaging.

8. Resources

Flood Fill algorithm. Online. Wikipedia. 2024. Dostupné z: https://en.wikipedia.org/wiki/Flood_fill. [cit. 2025-05-27].

FlatLaf library. Online. 2025. Dostupné z: https://search.maven.org/artifact/com.formdev/flatlaf/3.6/jar?eh=. [cit. 2025-05-27].