SeeSharper
Intuitive Drawing Application
Documentation

Baltariu Ionuț-Alexandru - 1305A
Beldiman Vladislav - 1350A
Nistor Paula-Alina - 1305B
Rusu Iulian - 1305A

”Gheorghe Asachi” Technical University of Iași
Faculty of Automatic Control and Computer Engineering

Software Requirements Specification

Contents

1 Introduction

1.1 Purpose

The point of this document is to give a detailed description of the requirements for the SeeSharper application. It will explain the purpose and features of the program, its interfaces and its core functionalities.

1.2 Scope

SeeSharper is a simple program that people can use for creating drawings and minimally editing images. The application must provide basic functionality for drawing and painting like different types of brushes, brush sizes and a color palette. SeeSharper should also provide shaped stencils and line tools, making the application useful for diagram development. After the drawing is ready, the user has the possibility to save the image in a specific format.

SeeSharper develops an efficiently easy to use interface. This application wants to be a simplified version of Microsoft Paint and Paint.Net.

1.3 Overview

The next chapter, the Overall Description section of this document, gives an overview of the functionality of the application. It describes expected user characteristics and general constraints. The third chapter, Specific Requirements section of this document, describes in technical terms the details of the functionality of the application and describes the graphical user interface.

2 Overall Description

2.1 Product Perspective

SeeSharper is a stand-alone drawing application with minimal image editing capabilities written in C# for the Windows Operating System. It aims to be an open source alternative to Microsoft Paint with a similar design and basic functionality unlike other open source products available such as GIMP.

SeeSharper will be implemented as a Windows Forms Application. The application will be developed with the Model-View-Presenter architecture, with a supervising controller. It will use a thread for the GUI and another one for the MVP components.

2.2 Product Functions

SeeSharper must provide the following major functions for its end users:

1. Image Loading - SeeSharper must be able to load user images so that they can be drawn on.
2. Drawing Saving - SeeSharper must have the functionality of saving a user drawing.
3. Customizable Brush - SeeSharper must provide a wide range of color selection, as well as different brush sizes.
4. Drawing Shapes - SeeSharper must provide easy to draw shapes with a customizable border and fill.
5. Drawing Text - SeeSharper should provide a way to add text to the drawing.

2.3 User Classes and Characteristics

SeeSharper is to be design for ease of use, especially for users that have not used a drawing application before. It, however, does not provide as many functionalities as a professional image editor in order to keep it from being overwhelming. Buttons and options should provide commonly used icons that convey their functionality instead of text or along with it whenever possible. Furthermore, menus should be easy to navigate without many layers of submenus. It is also designed for those who seek an open source application, but which is more accessible than its counterparts.

2.4 Design and Implementation Constraints

The main limiting constraints in this development cycle will be time as well as the technology constraint of developing everything for the .NET platform, which will be amplified by our inexperience with the C# language, and inexperience in software design and documentation.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

This section describes the graphical user interface (GUI) features and constraints.

1. The main window of the application should have a large canvas, initially with a white background.

2. A menu with options will be located in the top left corner (Load, Save, Help, etc.).
3. A menu that allows the user to select drawing tools.
4. When clicking the color menu, the user will see a new window where they will be able to select the desired RGB color. Optionally, other color spaces may be selectable.
5. Clicking on the Load/Save buttons will open a separate dialog window where the user will be able to open or save their drawing.
6. Clicking on the Help button will open a separate window with some documentation and/or tutorial.

3.1.2 Software Interfaces

The application must provide a way to communicate with the underlying Operating System's API to delegate the saving/loading of drawings as image files. There are no other software interface constraints as the application is self-contained.

3.2 Functional Requirements

This section exemplifies the actual functionalities of the application, with in-depth information about every client observable feature.

3.2.1 Color picking

The user must be able to select the color in which to draw or represent a preselected geometrical form. Colors from all the RGB spectrum must be present in the application.

3.2.2 Eraser tool

The user must be able to use an eraser tool in order to delete previously done work in the drawing canvas. The size of the eraser should be user modifiable (within predefined bounds), improving performance when trying to remove large portions of a drawing.

3.2.3 Brush tool(s)

In order to paint properly, a brush-like took must be present in the application, that, when selected, gives the user the opportunity to draw any shape or object of the previously selected color and brush-size.

The application has to contain various types of brushes, of different dimensions and drawing styles, in order to give a pleasing and versatile end-user experience.

3.2.4 Other advanced drawing tools

1. **Geometrical shape creator**

   The user should be able to select a developer-defined geometrical shape and generate it on the canvas within wished bounds. Shapes must be defined by a number of points, usually two. Ideally, a preview of the shape being drawn should be displayed and modified in real time as the user drags the mouse on the canvas. After clicking to pick the final point on the canvas, the user should have the option to change the color of the shape by selecting a different color in the menu. At this point, the shape should also allow dragging with the mouse to translate its position on the canvas. Upon clicking again, the shape will be finalized and not modifiable anymore. Should the Undo option be available, it will completely remove the shape.

2. **Shape filling at drawing time**

   The user might indicate if the shape that is to be drawn has to be filled with a color. If so, the fill color is a distinct one from the drawing color(border color) and can be selected from within the application.

3.2.5 Drawing saving/loading

The application must offer the possibility of saving the current drawing to an image (of a selected extension) that will be located on the storage disk of the computer.

Similarly to the previously mentioned functionality, the application must offer the possibility of loading an already existing image that the user wishes to be modified.

3.2.6 Undo/Redo

In order to provide a rollback feature when making mistakes, the application must offer the possibility of undoing an action at any given point. It is also wished that any undone action can be redone.

These actions might be done either by pressing the corresponding buttons in the graphical user interface or by using the well known shortcuts: **CTRL+Z** for undo, and **CTRL+Y** for redo.

The user might even undo a previously loaded image, as it is permitted by the application.

If multiple undo actions are commited, and then the user draws something on the canvas, he might not redo any action that existed before the additional drawing element has been added to the canvas.

3.2.7 Help window

When clicking the 'Help' button, a window with all of the application's features will be provided to the user in the form of a .chm document. In this way, the typical user might research all of the tools and features that *SeeSharper* is providing.

3.3 Performance Requirements

This section covers the requirements that concern the performance of the application in response to user interaction. It is aimed to describe general use case scenarios, as well as time performance constraints.

1. **Usage of the toolbar menu**
   The graphical interface must provide a menu that must be easily accessed at the top of the main application window in one click.

2. **Usage of drawing tools**
   All drawing tools must be easily selectable from a unified menu of drawing tools. The user should have the option to change the color, size and any other parameters of a selected tool easily.

3. **The drawing process**
   Drawing must feel smooth and be responsive to user input. The application should provide a way to undo/redo changes from history, without noticeably slowing down performance.

4. **File system interaction**
   All drawings must have the ability to be saved on the disk in a specific image format (`png, jpeg, bmp`). The application should provide a way to load images into the canvas and draw on them. The process of saving and loading should not take longer than doing so in other applications that interact with the file system.

5. **General timing constraints**
   Any user input should be processed without a noticeable delay. In the event that a more complex computation is required, the user must be notified and the graphical interface must remain responsive for the whole duration. Additionally, the interface may provide status messages to notify the user.

3.4 Design Constraints

This section describes the limitations imposed on the application by software or hardware characteristics of the environment.

1. **Disk space usage**
   The whole application must not occupy more than 20 MB of disk storage. Ideally, the application should fit into 10 MB.

2. **Memory usage**
   The application's RAM usage must not exceed 15 MB. Ideally, the application should not use more than 8 MB of RAM.

3.5 Attributes

This section describes different software system attributes, metrics and requirements for them.

1. **System reliability**
   Reliability refers to the system's capacity to correctly respond to user input. The application must correctly load, save and draw images 100% of the time.

2. **Maintainability**
   The application should be easily maintainable and extendable. The code should allow easy testing and should be open for future extensions and new features.

UML Diagrams

How to use the application

Drawing lines or various geometric shapes

In order to draw lines (straight or curved) or other shapes, the user has to select the shape before actually drawing on the canvas.

Keep in mind that in order to draw geometric shapes, the **click** mouse button has to be hold, then drag the figure to respect wished dimensions.

Figure 1: Tools and shapes menu

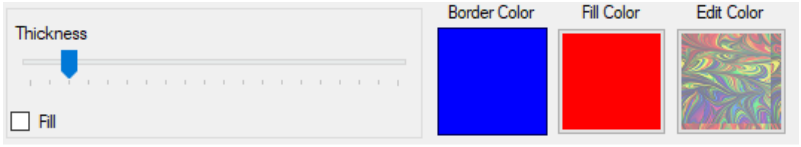


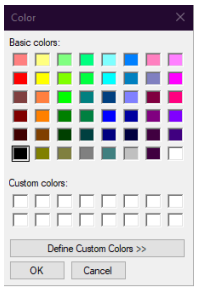
Figure 2: Tools and shapes menu

The user can also select whether the shape that is to be drawn will be filled by checking the ”Fill” box in **Figure 2** and might also change the border line thickness by dragging slider bar.

Changing drawing colors

As seen in **Figure 2**, the user can changed both the *fill* and *border* colors.

Firstly, the user has to select whether he wants to change the fill or the border colors, then, the **Edit Color** button has to be pressed. The following window will appear:



Here, any given color can be selected. However, if the user wants to select a custom color, the **Define Custom Colors »** button has to be pressed.

After that, one can give custom wanted colors by either entering RGB color codes or by selecting one from the window on the right.

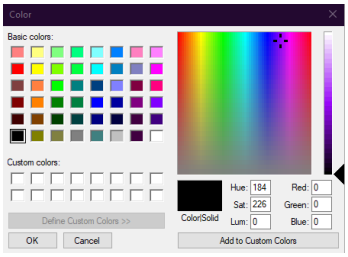


Figure 3: Custom color selection

Erasing

From **Figure 1** select the *Eraser tool* (the first square in line), change the thickness depending on the size of the area that is to be erased and, finally, erase by holding the click button on the canvas.

Using undo/redo functionality

After drawing a shape, if the user considers that a mistake has been made, he might undo the latest actions by clicking the **Undo** button **(Figure 4)**.

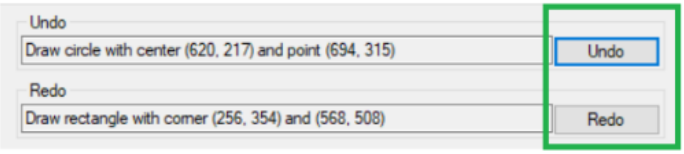


Figure 4: Undo/Redo buttons

If mistakes have been done while undoing, there is also the possibility to redo previously undone actions by pressing the **Redo** button. **(Figure 4)**

Saving/loading drawings/images

The application allows both loading and saving an image or a drawing to the file system as a PNG file. **(Figure 5)**

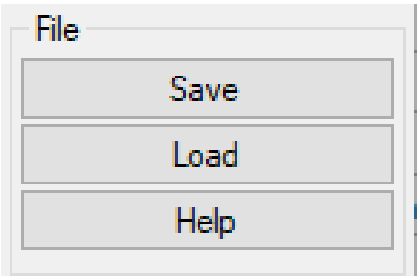


Figure 5: Menu for saving/loading

After clicking the **Save** button, the user has to navigate to the wished saving location and then give a valid name to the file, ending by clicking again the **Save** button in the new window.

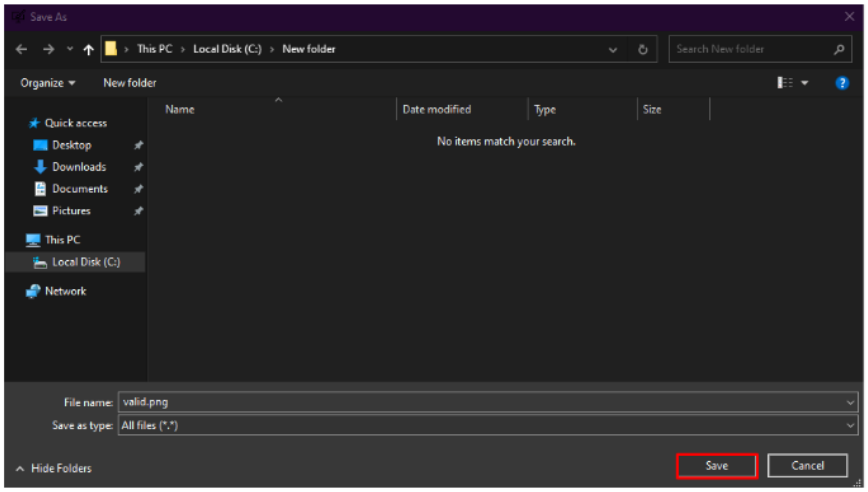


Figure 6: Save window

Same principles apply to the **Load** functionality.

Accesing help from within de application

Navigate to the menu at **Figure 5**, then click the **Help** button.

After that, use the menu found on the left side of the window to access information about wanted subjects.

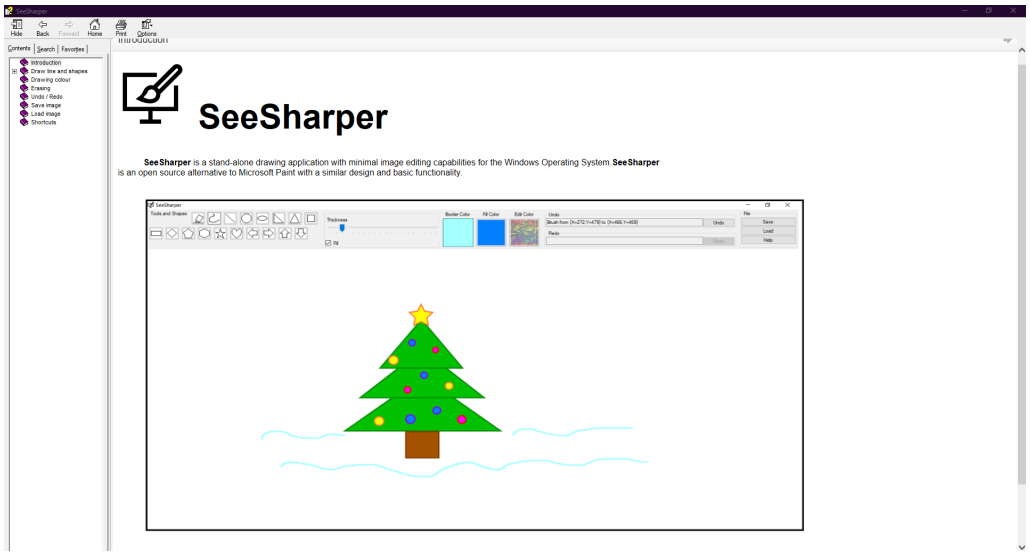


Figure 7: Help window

Shortcuts

Shortcut	Action
CTRL+S	Save
CTRL+L	Load
CTRL+H	Open Help
CTRL+Z	Undo
CTRL+Y	Redo

Screenshots of the application

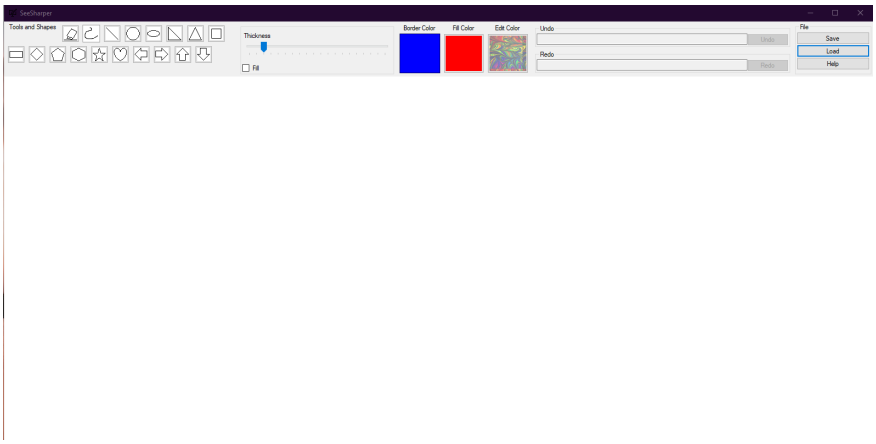


Figure 8: Application at start

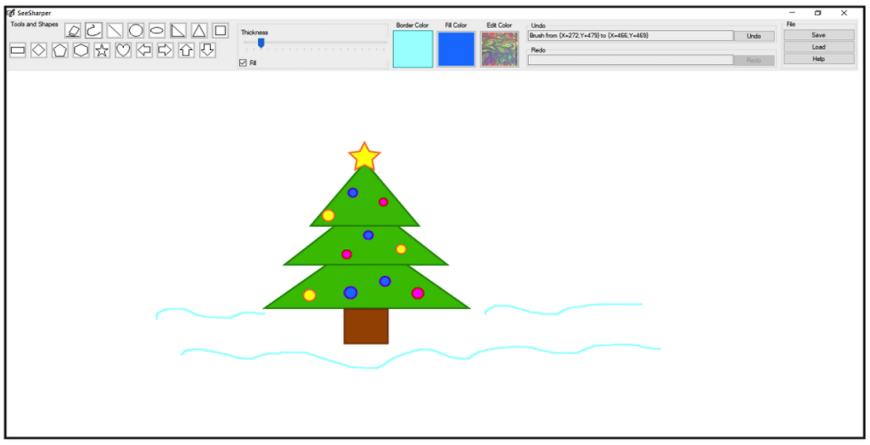


Figure 9: Application after drawing

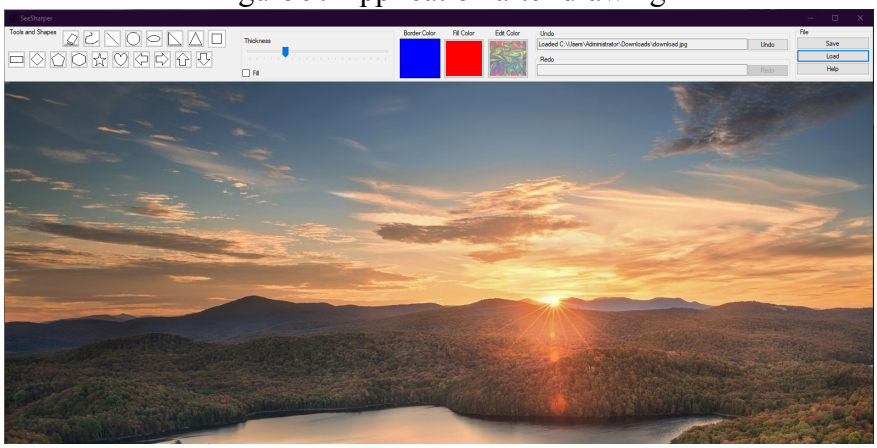


Figure 10: Application after loading an image
More screenshots can be seen at the previous section.