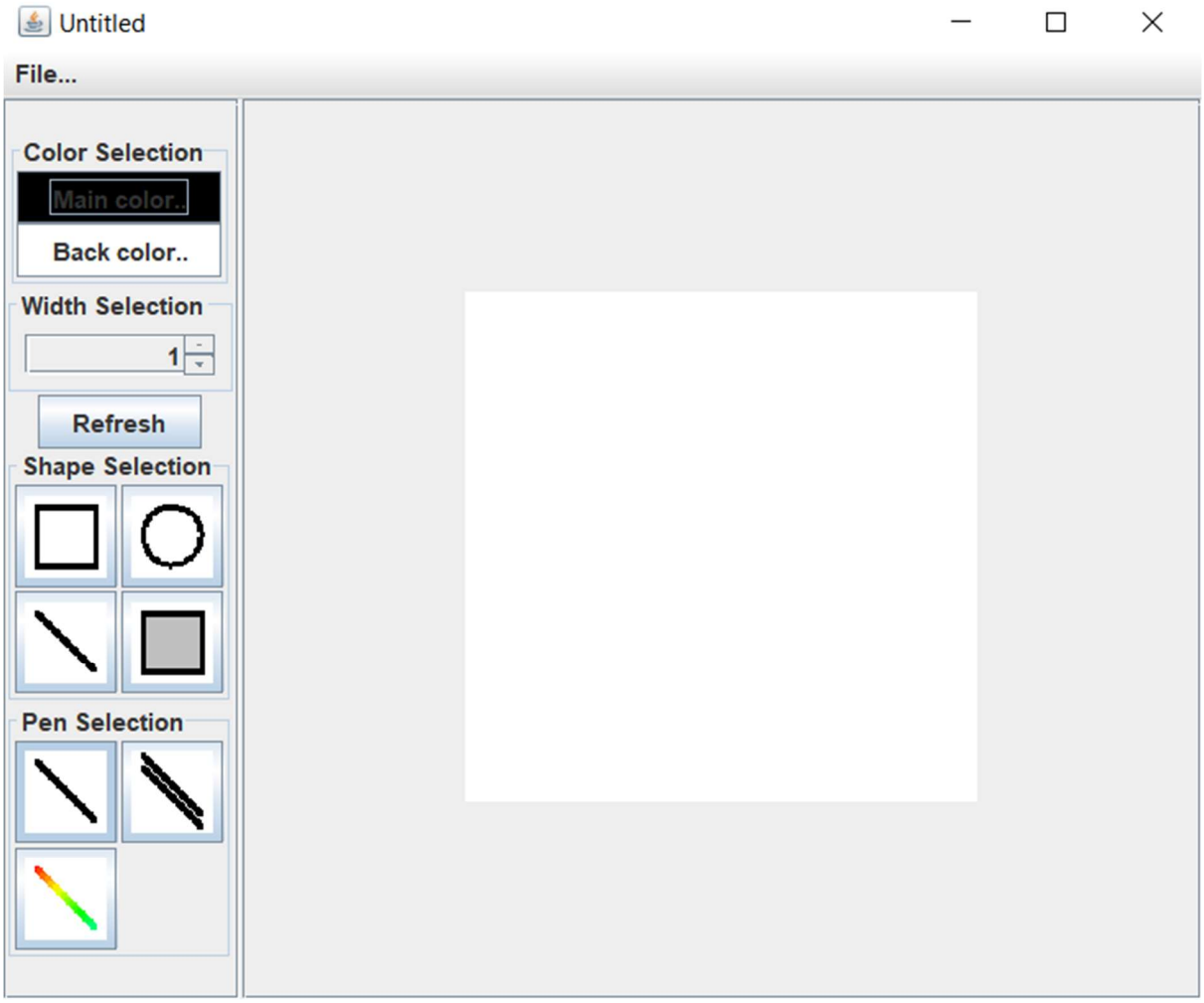
User Documentation

Basic function

Upon launch, the application displays this window:



The right side of the window is occupied by the **Drawing Panel**, which displays the current image. Upon launch, this will be an unsaved 250 by 250px white image.

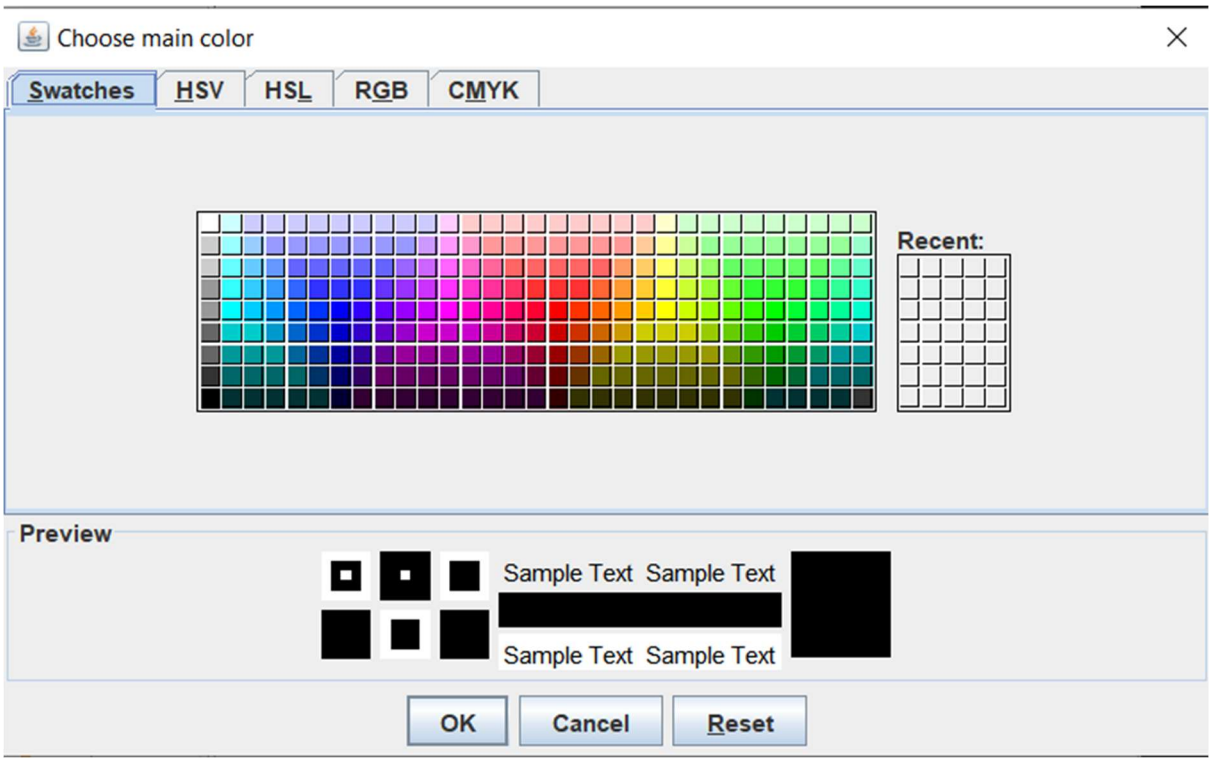
The image can be drawn on using the mouse. What the drawing will look like is determined by the **Settings** in the left panel.

The available settings are:

- **Main color** – determines the color of pen strokes and shape outlines (if applicable)
- **Back color** – determines the color of the background of new images and the fill color of filled shapes

- **Width selection** – determines the width of the pen stroke and shape outline (if applicable)
- **Refresh button** – refreshes the list of available shape and pen plugins (more info below)
- **Shape and Pen selections** – allows the user to select the drawing tool

The **color selection** buttons display a color selection dialog window with various ways to select a color:

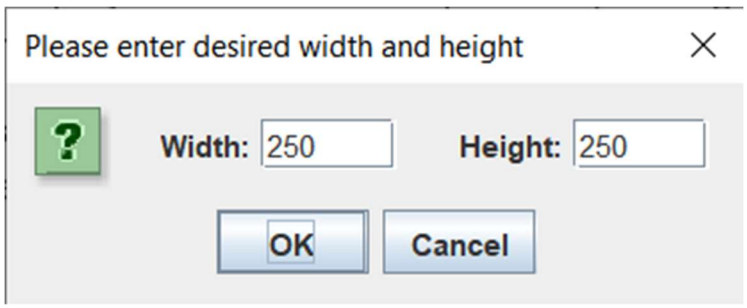


The **shape selection** displays three default shape tools (rectangle, ellipse and line) and one example plugin shape – the filled rectangle.

The **pen selection** displays two default pens (the basic pen and the double pen) and one example plugin pen – the rainbow pen.

The top **File...** menu provides basic file handing:

- Loading a new file of the specified dimension using the **New...** dialog



- Opening an existing image using a standard **Open...** dialog window

- Saving the image using a standard **Save...** dialog window (or saving to the previously selected save file).
- Saving the image to a specific location using a standard **Save as...** dialog window

Plugins

The application supports loading new pen and shape tools using plugins. To load a plugin, the .jar file with the implementation must be placed in the /Plugins directory. The plugin will be loaded if it is present upon startup, or upon clicking the **Refresh** button. Once loaded, the plugin will remain loaded until the app window is closed.

How to implement a plugin

A plugin is a .jar file with the plugin implementation. There are two types of plugins: basic Pen plugins and Shape plugins, which will be loaded into their appropriate selection menus in the app window.

The implementing classes must extend the Pen abstract class, resp. the ShapePen abstract class. In addition, the .jar file must be loadable for the Pen, resp. ShapePen service by the standard Java ServiceLoader, that is, it must have a META-INF/services/cz.cuni.mff.java.drawing.Pen (or .ShapePen) entry listing the implementing class(es). The classes also must have a non-parametric constructor.

Pen plugin implementations may override any of the Pen methods for handling mouse input directly. Additionally, they should override the reset() method, which should reset the pen to its default state and release any held resources. This method is called when the pen is deselected.

ShapePen plugins only need to override the getShape(int x1, int y1, int x2, int y2) method, which should return the desired Shape instance which is situated between the start point (x1, y1) the end point (x2, y2). If the shape should be filled, the fill filed must be set to true in the constructor. The actual drawing of the shape is handled by the ShapePen class itself.

Optionally, any plugin may override the getPreferredIcon() method to specify the icon to be displayed on the buttons. For the purposes of this app, the icon should be 40 by 40px. If the method is not overridden, then the app generates its own icon by simulating a mouse drag.

For more information, refer to the documentation of the Pen, ShapePen and PluginLoader classes, as well as the two provided example plugins, the RectFillPlugin and RainbowPlugin.