

## **A Java Application for Data Hiding in Image Files: User's Manual**

by Jessica Codr under guidance of Dr. Raj Jain

This manual briefly outlines the use of what I now call C-Hide, a simple Java application for hiding messages within various types of images, developed by Jessica Codr.

### **Requirements:**

To run this program and generate random images, you will need:

The java runtime environment for java 5.0 or later

The class files BitFlipper.class, BitFlipper\$1.class, BitFlipper\$2.class, BitFlipper\$3.class, BitFlipper\$4.class

OR

The java compiler and runtime environments for java 5.0 or later

The java file BitFlipper.java

OR

The .jar file BitFlipper.jar

To run this program embedding data within your own images, you will also need an image of the .jpg, .bmp, .png, or .gif image formats.

### **Installation and startup:**

From the .jar file (preferred method):

Double click on or otherwise launch BitFlipper.jar

From the java file using Eclipse:

Open a new Eclipse project and add a package called "imageMsg"

Copy BitFlipper.java into this package

Run BitFlipper.java from Eclipse

From the java file using the command line:

Copy the BitFlipper.java file to a directory from which you can compile Java code.

Navigate to this directory from the command line

Type "javac BitFlipper.java"

Type "java BitFlipper"

From the object file:

Place the BitFlipper.o file in a directory from which you can run Java.

Navigate to this directory from the command line.

Type "java BitFlipper"

### **Program Operation**

The following is a screen shot of the launched program window and a description of the different program options as they are numbered 1-8 on the screenshot:

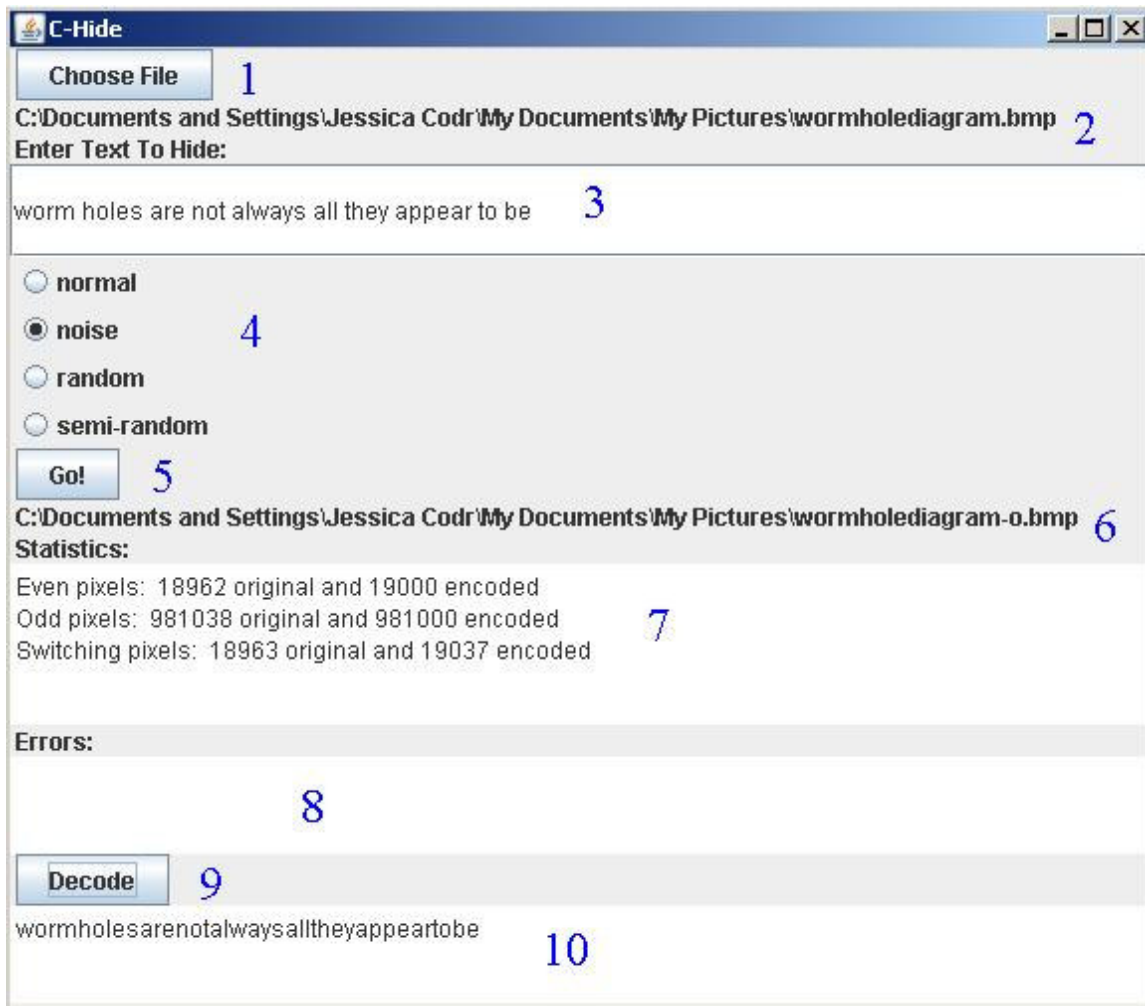


Figure 1: C-Hide UI

Each of the numbered sections of the UI are described as follows:

1. **Choose File Button.** Clicking this button brings up a dialog for selecting an image of the .jpg, .bmp, .png, or .gif file formats from the user's file directory. For the case of encoding a message within a pre-existing image file, this should be used to select the pre-existing image file. For the case of generating a random or semi-random image, this should be used to enter the name (and location) to which the random image should be saved.
2. **Selected File:** This displays the file selected from the Choose File Button.
3. **Message Text:** This is where the user types the text to be embedded within the image. Only letters (upper and lower case) and spaces are allowed. Any punctuation marks will cause the program to fail.
4. **Steganography Selection:** This set of radio buttons lets the user select what type of steganography they wish to use.

- a. normal: The message is embedded directly into the specified pre-existing image file.
- b. noise: Noise is added to the pre-existing image file and then message is embedded.
- c. random: A fully random image is generated and saved to the file specified by Selected File. The message will then be embedded in this file.
- d. semi-random: A random image in which neighboring pixels are of similar color is generated and saved to the file specified by Selected File. The message will then be embedded in this file.

In all cases, the originally supplied or generated file will be preserved and the stegoimage saved to the file Y-o.X where Y is the name of the original file and X is the file extension (either jpg, bmp, png, or gif).

5. Go Button: Press this button to run the steganography software and generate the random coverimage (if applicable) and the stegoimage. Note that this could take some time, especially when adding noise to a large image. If the Go button remains selected, the program is still working.

6. Output File: This lists where the result of performing steganography on the coverimage is saved. Thus, Selected File is the coverimage and Output File is the stegoimage.

7. Statistics: This text area displays information about how the pixel values of the cover image and stegoimage compare.

8. Errors: This text area displays information about errors that might occur while trying to run the software.

9. Decode Button: After the user has encoded a message by pressing the Go button and having the results output correctly, he can press the decode button to extract the message from the stegoimage and make certain it matches the intended message.

10. Decoded Message: Text area displaying the message hidden in the image which was found by decoding. Note that the message is entirely lower case letters (no spaces) since these are the only symbols or spaces hidden in the image.

To view your randomly generated images and/or stegoimages, simply navigate in your file system to where the Selected File and/or Output File text labels indicate they are located. You should observe that the cover and stegoimages appear visually the same.