Steganography – Version 2

The task is to create Sketch with two main functions: *encode()* and *decode()*.

encode(String in_filename, String message, String out_filename) will be given the name of a picture file (known to be 256 x 256 in size), plant the given secret *message* inside it, and save it on disk in *out_filename.* It will return a 0 on success and 1 on failure.

decode(String in_filename) will be given the name of a picture file (also 256 x 256) which has a secret message encoded within it, and will decode and return that message to the calling program which will print it on the console (the lower black area in your sketch window).

int encode()

1. Read the picture file into a PImage object. Display it on the screen.

2. Using *loadPixels()*, create the *pixels[]* array.

3. Using a for loop, examine and possibly modify each pixel in the *pixels[]* array in the following way: if the red value of a pixel is 255, modify that pixel so that its new red value is 254, but keep its green and blue values unchanged.

Digression: let's talk about Strings, characters and their numerical values…

Suppose the secret message is "ABC". Each character in this message has a numerical value (its "ASCII" value). It turns out that the ASCII value of "A" is 65, of "B" is 66 and "C" is 67. We can convert a character into its numerical value using the *int()* function. We can also convert an integer back into a character using the *char()* function.

Try this example:

```
char ch;
String msg = "ABC";
int fred;

ch = msg.charAt(0);  // this is "A"
println(ch,int(ch));

ch = msg.charAt(1);  // this is "B"
fred = int(ch);
println(fred, char(fred));
```

4. Now we know that there are no pixels in the pixels[] array with a red value of 255, but there are probably many with 254.

5. Using a for loop, go through the *pixels[]* array and find the first pixel with a red value of 254. Change that pixel in the following way:
a) Change the red value to 255
b) Keep the green value as it was
c) Change the blue value to the numeric value of the first character in the secret message.

6. Go on to find the next pixel in the array with a red value of 254, and place the next character of the secret message into it. Keep going until you run out of characters in the secret message.

7. If you run out of pixels with red=254, then you cannot encode the rest of the secret message. Give up and print an error message to the console saying: "Error: not enough original pixels with red=255. Encoding has failed. Try a different picture." Return 1 and terminate the program.

8.. If you have succeeded, *updatePixels*() and save the image into the *out_filename* file. Note: the *out_filename* string should have a suffix of ".png". Return 0.

It's a convention that for functions that can succeed or fail, returning 0 is sign of success, while returning a non-zero integer typically indicates which of several possible errors actually occurred.

String decode()

decode(String in_filename) should read an image file that has a secret message encoded into it, and return that secret message to the calling program which will print it to the console.

1. Read the picture file into a PImage object. Display it onto the screen.

2. *loadPixels*() to create the *pixels*[] array.

Digression: we can create longer strings by adding characters into the end of a string. Try this example…

```
int[] harry = {65,66,67};
String voldy = "";

for (int i = 0; i < harry.length; ++i) {
  voldy += char(harry[i]);
}

println(voldy);
```

3. Now go through the *pixels*[] array and find the ones with a red value of 255, and take the blue numerical value (which is a *float*), convert it into an *int*, convert that *int* into a *char*, and append it to the end of a String that will be the secret message. When you've run out of pixels with red = 255, you've composed the message. Return that message string to the main program.

4. The main program (possibly *setup*())s should print that message to the console.