1. Give a detailed explanation of how your algorithm works.
2. The beginning of embedding the message and retrieving the message

I create a hash map that consists of all characters that can be input into my algorithm the characters are between 'Space' and '\_'. I determine all possible prime numbers that fit in the picture. I determine an x value based on the next prime number. If the x value is greater than the product of the width and y value then I increment the y value. I get the color at the current x, y position.

1. For embedding the message.

I determine the next character from the message and turn the character into a binary string. I then retrieve the red value from the color. I turn the red value into a binary string. I make sure the binary string is 8 characters long. I separate the first 6 characters from the binary string. I separate the first two characters from the character binary string and attach the characters to the end of the first six characters of the red value binary string. I do the same process for green and blue. However for green I use the next two characters from the character binary string and for blue I use the last two characters from the character binary string. I parse the binary strings back into integers and use those values to make a new color and set that color as the new color for the current x, y position. Then I do the same for the next prime and next character until the message is completely embedded.

1. For retrieving the message.

After I get the color at the current x, y, position I get the red value and turn the red value into a binary string and make sure the binary string is 8 characters long. Then I get the last two values of the binary string. I do the same for green and blue. I concatenate the red green and blue ends together to form a 6 character long binary string. I search the hash map for a value that matches the binary string then return that key (character) once I've found it. If the value does not exist and there is no match, that character will be replaced with a space. I check for the escape sequence '/' and break if found. If not found, I attach the letter to the message. I return the message.

1. Prime Generator Logic

First I populate a list of consecutive numbers from 2 to the max that was sent into the constructor. Then I iterate from 2 to the square root of the max incrementing by one. Then I iterate from the current number squared to the max, incrementing by the current number. I check to see if the list of consecutive numbers contains that number, if not, continue looping. If so, find the index of the number then remove the number from the list using the index then continue looping.

1. Outline any help you received from classmates or other students

Steven reminded me how a swing timer in java works. For example, the action listener is an interface you need to implement in a class. You should probably make it an inner class, and every delay that you set in the timer constructor will call the overridden method you wrote.

1. Mathematically analyze the performance of your algorithm using big-Oh notation. Show that your algorithm performs at or better than O(width\*height).

T(√(width\*height) + [(width\*height)/10] + 1986)

√(w\*h) + [(w\*h)/10] + 1986 <= c\*w\*h, (w\*h) > 1

C = 1986, w\*h = 2

√2 + 2/10 + 1986 <= 1986 \* 2

1987.614 <= 3972

1. Empirically analyze the performance of your algorithm. Outline your approach (what made you decide these experiments were good? Etc.) and publish a table of your results, e.g.

|  |  |  |
| --- | --- | --- |
| Picture size | Message Size | Total Time |
| 31,277 bytes (png) | 10 | 0.527 sec |
| 31,277 bytes (png) | 1000 | 0.653 sec |
| 31,277 bytes (png) | 100000 | 2.36 sec |
| 22, 975 bytes (jpg) | 10 | 85.888 sec |
| 22, 975 bytes (jpg) | 1000 | 82.543 sec |
| 22, 975 bytes (jpg) | 100000 | 100.935 sec |
| 224, 968 bytes (png) | 10 | 85.576 sec |
| 224, 968 bytes (png) | 1000 | 85.493 sec |
| 224, 986 bytes (png) | 100000 | 107.569 sec |

I figured increasing the message size would increase the time but not by much, it’s increasing the image size that is the drastic part. I used different image sizes to prove the time would also change according to the size of the image. And I just now realized jpg are much more compressed than png. I retrieved the number of bytes from eclipse properties.