Universidad Nacional Autónoma de México

Facultad de Ciencias

Proyecto 02

17 de Diciembre del 2021

Autores: Castro Romero Ares Gael y Prado Escamilla Axel

Documentación de la Solución.

---

1. **Definición del problema.**

   El problema planteado consiste en entregar una imagen modificada que ,dicha modificación, sea imperceptible al ojo humano y se utilicen técnicas de programación para el revelado de la información oculta en dicha imagen. Dicho problema requiere utilizar los bits menos significativos de las imágenes para que dicho cambio sea imperceptible, en específico el último bit menos significativo.

---

2. **Análisis del problema.**

   La entrada del programa es un archivo .txt con una cadena de caracteres y una imagen, a la cual se le modificará para que quede escondido el texto del archivo y esa será su salida.

   El programa solución debe poder tener dos opciones, la opción en la que se esconde en la imagen el texto del archivo txt y la otra opción es la que se develerá el texto dentro de la imagen.

   El manejo de las imagen se hace mediante la clase BufferedImage que se importa desde java.awt. y así poder usar los métodos getRGB y setRGB que son de vital importancia enla resolución del problema.

   La salida de la primera parte del programa será la imagen modificada para después usarla en la segunda parte de dicho programa.

---

3. **Modelo y abstracción.**

   El paradigma de programación que será usado es el orientado a objetos y el lenguaje usado será Java.

   El algoritmo central para ocultar el texto de un archivo en una imagen del tipo **PNG**, cuya entrada es el archivo con el texto y la imagen, que debe seguir el programa es el siguiente:

```
    texto = leer(archivoTexto)
    buffer = leer(imagen)
    binario = convertirBinario(entrada)
    cont = 0
    i = 0
    lugarPixel = (0,i)
    while ( cont < longitud(binario)) {
        pixel = obtenerRGBA(buffer,lugarPixel)
        a = obtenerAlpha(pixel)
        a = reemplazarBit(a,binario[cont])
        r = obtenerRed(pixel)
        r = reemplazarBit(r,binario[cont+1])
        g = obtenerGreen(pixel)
        g = reemplazarGreen(g,binario[cont+2])
        b = obtenerBlue(pixel)
        b = reemplazarBit(b,binario[cont+3])
        pixel = nuevo Pixel(r,g,b,a)
        cambiarRGBA(buffer)
```

```
        cont += 4
        lugarPixel = (0,i++)
    }
```

El pseudocódigo hace lo siguiente:

a) Abre el archivo de texto para extraer el texto y guardarlo en una cadena.

b) Se abre un buffer para leer los dato de la imagen.

c) Convertimos todo el texto en una cadena de 0's y 1's.

d) Realizamos un bucle que se repite tantas veces como la longitud de la representación binaria del texto.

e) Dentro del bucle, obtenemos el píxel de la posición dada de la imagen.

f) del píxel sacamos sus componentes R, G, B y A.

g) A cada una de las componentes del píxel lo modificamos de tal forma que reemplazamos su bit menos significativo de cada uno por un bit de la representación binaria del texto. Se toman 4 bits de l representación binaria, cada uno de esos bits se almacena en una componente RGBA.

h) Cambiamos el píxel por otro con las nuevas componentes modificadas.

i) ajustamos en el buffer el nuevo píxel.

El método para recuperar el texto de una imagen recibe como entrada la imagen de donde se recuperará el texto y el tamaño del texto. El siguiente algoritmo permite recuperar el texto de la imagen:

```
    buffer = leer(imagen)
    l = longitudTexto
    for (i = 0; i<l; i++){
        lugarPixel = (0,i)
        pixel = obtenerRGBA(buffer,lugarPixel)
        a = obtenerAlpha(pixel)
        r = obtenerRed(pixel)
        g = obtenerGreen(pixel)
        b = obtenerBlue(pixel)
        String str = obtenerBit(a)+obtenerBit(r)+obtenerBit(g)+obtenerBit(b)
    }
    str = convertirTexto(str)
    return str
```

El pseudocódigo hace lo siguiente:

a) Toma como valores de entrada la longitud del texto y la imagen donde se esconde el texto

b) creamos un buffer para leer los píxeles de la imagen

c) abrimos un bucle para recorrer los píxeles de la imagen, se toman tantos pixeles como la longitud del texto

d) De cada píxel se sacan sus componentes RGBA.

e) De cada componente se saca el bit menos significativo y se concatenan.

f) La cadena binaria completa se transforma en texto formando así la cadena oculta en la imagen.

Para la implementación de los pseudocódigos, modelamos el problema con clases. Se implementaron las siguientes clases:

a) **Binarios**

Esta clase la incorporamos en el paquete *esteganografia.utilidades*.

Esta clase nos permite operar con cadenas binarias.

No contiene atributos.

Posee los siguientes métodos:

- *stringBinario:* Recibe como parámetros una cadena de texto y la transforma en una cadena binaria. Regresa la representación binaria del texto.

- *modificaCanal:* Método que reemplaza el dígito menos significativo de la representación binaria de un canal RGBA por otro dígito pasado como parámetro.

b) **Texto.**

Esta clase la incorporamos en el paquete *esteganografia.utilidades*.

Esta clase sirve para leer o escribir sobre un archivo de texto.

Esta clase posee los siguientes métodos:

- *obtenerContenido:* Método que regresa una cadena de texto con el contenido del archivo pasado como parámetro.
- *escribirContenido:* Método que escribe en un archivo una cadena de texto.

c) **EsteganografiaLSB.**

Esta clase la incorporamos en el paquete *esteganografia.lsb*.

Esta clase nos permite esconder un texto en una imagen o recuperar un texto de una imagen.

Los método que posee son:

- *recuperaBinarios:* Método que recupera de una imagen los dígitos binarios de una cadena. Los parámetros que recibe es la cantidad de píxeles a usar y la imagen de la cual se recuperará el texto.
- *recuperarCadena:* Recibe como parámetro la cantidad de píxeles a usar, el nombre de la imagen de donde se recuperara el texto y el nombre del archivo de salida. Este método transforma en texto la cadena binaria dada por recuperarBinarios.
- *copiarImage:* Este método copia una imagen en otra de forma que la copia pueda modificarse la transparencia. recibe como parámetro el buffer de la imagen original.
- *esconderCadena:* Método que esconde en una imagen una cadena de texto. Recibe como parámetros la cadena, un buffer de lectura de la imagen y el nombre de la imagen de salida.
- *esconderTexto:* Método que oculta el contenido de un archivo de texto en una imagen. Utiliza al método esconderCadena.

d) **Imagen.**

Esta clase contiene el método ***main*** que permite ejecutar el programa entero.

---

4. **Cambios Futuros**

Interfaz Gráfica

Una interfaz gráfica haría que todo sea más visible y tangible para el usuario, ya que podría ingresar el texto y después mostrar ambas imágenes en dos ventanas para que el usuario vea que el cambio es imperceptible al ojo humano y después en otra ventana el revelado del texto.

Hay limitaciones en el programa como que no se puede esconder un texto demasiado grande pues no se usa todo el contenido de la imagen.Esto es debido al tiempo de entrega, para el futuro se puede mejorar este problema.

---

5. **Costos**

Se decide cobrar por proyecto dado que por línea de código o incluso por hora realmente llegan a ser cantidades absurdas para ambas partes.

El costo del proyecto se basa principalmente con respecto a la fecha de entrega de éste. Si hay poco tiempo para desarrollarlo, el monto llega a duplicarse con respecto al precio que normalmente se cobrearía.

Ahora bien **¿Cuál es ese precio?** Cobramos un precio de aproxidamente de $3000 pesos mexicanos si es que no se requiere comprar una clave para solicitar a la API. Así, $\frac{1}{3}$ del precio son destinados a los recursos por parte de los integrantes del proyecto, quedando una ganacia de $\frac{1}{3}$ para cada quien. Así las ganancias son justas y se cubren gastos como comida, internet, luz, etc.
