Ps1b: Image Encryption Program

Assignment Description:

This assignment was for us to make an image encryption program using the LFSR program in the previous assignment. The program takes an input file in the form of a png due to png files being uncompressed. The program takes each pixel in the image and XORs the color values for the pixel with a randomly generated integer from the LFSR program. This means that the program will turn into visual snow when run through the program and will return to normal is reentered into the program.

Key Concepts and Algorithms:

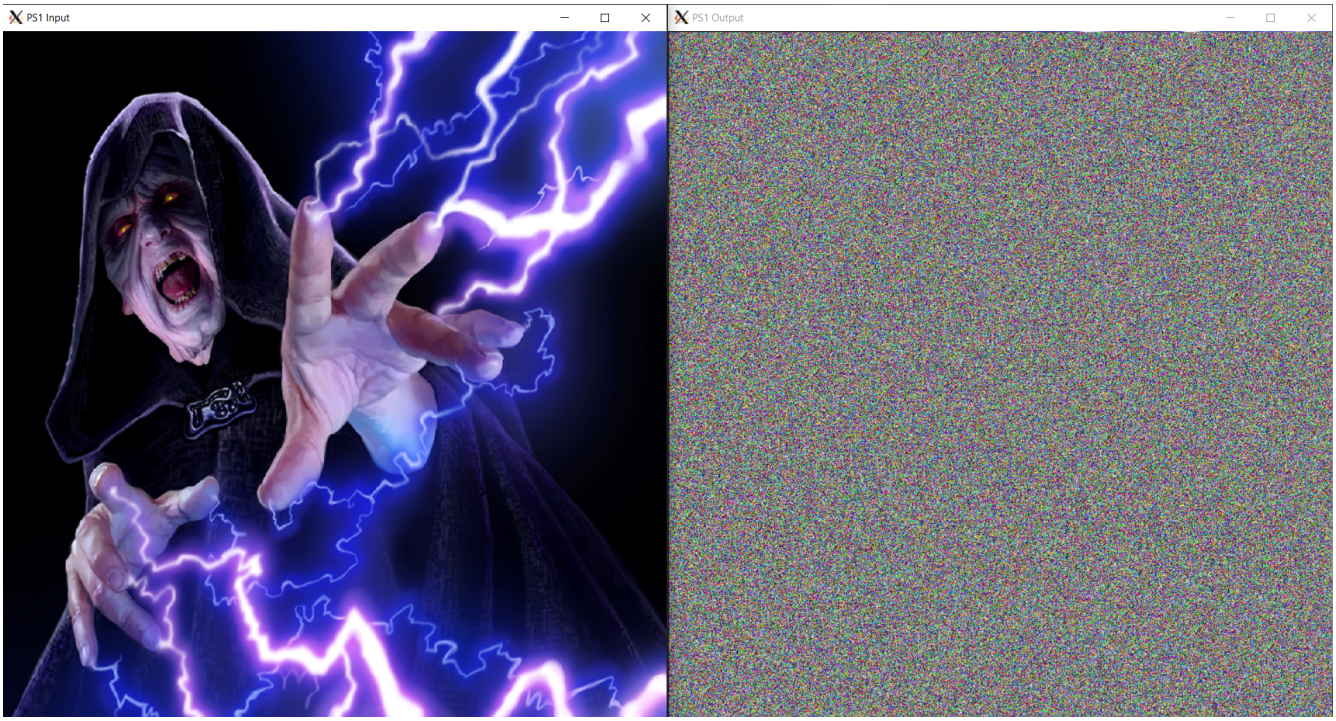
The key concepts in this project is the color values within a sprite object. Other concepts used in this assignment is the LFSR developed in Ps1a. For this assignment I took a photo of Emperor Palpatine from Star Wars. This picture would become unrecognizable when first run through the program but would return to normal once entered back into the program.

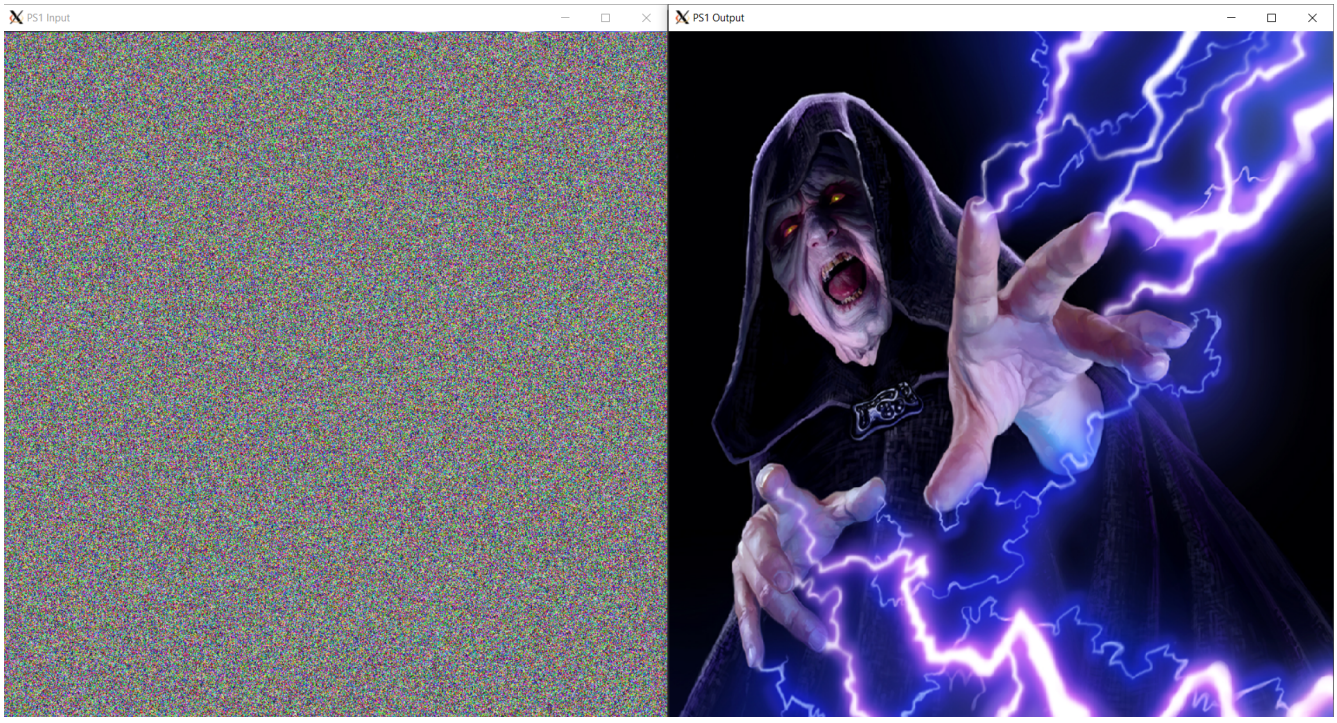
One issue I encountered while making this program was that the porgram would not function properly if the input file was not of the same resolution as the Palpatine.

What I learned in this assignment:

In this assignment I learned how color values work for SFML. I also learned in the course of this project the differences between png files and other photo file formates such as jpg files. This assignment taught me more about how SFML windows and sprites work and gave me more experience with graphical programs.

Ps1b: Screen shot





Ps1b Source Code: Makefile

```
CC = g++
CFLAGS =    -Wall -Werror -lsfml-graphics -lsfml-window -lsfml-system
DEPS =      FibLFSR.hpp

all: PhotoMagic.o FibLFSR.o PhotoMagic.cpp
    g++ FibLFSR.hpp PhotoMagic.cpp -o PhotoMagic FibLFSR.o -std=c++11 -Wall -Wer

FibLFSR: FibLFSR.cpp
    g++ FibLFSR.cpp -o FibLFSR.o -Wall -Werror

clean:
    rm PhotoMagic.o FibLFSR.o
```

Ps1b Source Code: PhotoMagic.cpp

```
/*
Name: John Simonson
Date: 2/10/2020
PS1b
*/
#include<SFML/Graphics.hpp>
#include <SFML/Graphics/Image.hpp>
#include"FibLFSR.hpp"
#include<unistd.h>
#include<string>
using namespace std;

int X = 959;
int Y = 832;

// transforms image using FibLFSR
void transform( sf::Image&, FibLFSR*);

int main(int argc, char* argv[]){
        string seed = argv[3];
        FibLFSR a(seed);
        sf::RenderWindow window1(sf::VideoMode(X, Y), "PS1 Input");

      sf::Image image1;
        if (!(image1.loadFromFile(argv[1])))
                std::cout << "Cannot load image";   //Load Image

```

```
28          sf::Texture texture1;
29          texture1.loadFromImage(image1);  //Load Texture from image
30          sf::Sprite Texture1;
31          Texture1.setTexture(texture1);
32
33          Texture1.getTexture()->copyToImage().saveToFile("output-file.png");
34
35          sf::RenderWindow window2(sf::VideoMode(X, Y), "PS1 Output");
36
37          sf::Image image2;
38              if (!(image2.loadFromFile(argv[2])))
39                  std::cout << "Cannot load image";   //Load Image
40
41              transform(image2, &a);
42
43              sf::Texture texture2;
44          texture2.loadFromImage(image2);  //Load Texture from image
45          sf::Sprite Texture2;
46          Texture2.setTexture(texture2);
47
48          Texture2.getTexture()->copyToImage().saveToFile("output-file.png");
49
50
51      while (window1.isOpen() && window2.isOpen()){
52          sf::Event event;
53          while (window1.pollEvent(event)) {
54              if (event.type == sf::Event::Closed)
55                  window1.close();
56          }
57          while (window2.pollEvent(event)) {
58              if (event.type == sf::Event::Closed)
59                  window2.close();
60          }
61          window1.clear();
62          window1.draw( Texture1);
63          window1.display();
64          window2.clear();
65          window2.draw(Texture2);
66          window2.display();
67      }
68
69
70      return 0;
71  }
72
73
```

```
74
75  void transform( sf::Image& image2, FibLFSR* a){
76      sf::Color buffer(0, 0, 0);
77
78          for(int i = X; i > 0; i--){
79              for(int j = Y; j > 0; j--){
80                  buffer = image2.getPixel(i, j);
81                  buffer.r = buffer.r ^ a->generate(8);
82                  buffer.g = buffer.g ^ a->generate(8);
83                  buffer.b = buffer.b ^ a->generate(8);
84                  image2.setPixel(i, j, buffer);
85              }
86          }
87
88          return;
89  }
```

Ps1b Source Code: FibLFSR.hpp

```
1  // John Simonson
2  // FibLFSR.hpp
3  // 2/3/20
4  #ifndef FIBLFSR_H
5  #define FIBLFSR_H
6  #endif
7  #include<iostream>
8  #include<string>
9  #include<cmath>
10 using namespace std;
11 class FibLFSR {
12 public:
13     FibLFSR(string seed);
14     int step();
15     int generate(int k);
16     friend ostream & operator <<(ostream& out, const FibLFSR c);
17 private:
18     string num;
19 };
```

Ps1b Source Code: FibLFSR.cpp

```
1  // John Simonson
```

```cpp
// FibLFSR.cpp
// 2/3/20
#include"FibLFSR.hpp"
using namespace std;

FibLFSR::FibLFSR(string seed){
    this->num = seed;
}

ostream & operator <<(ostream& out, const FibLFSR c){
        out << c.num;
        return out;
}

int FibLFSR::step(){
    int temp = this->num[0] ^ this->num[2];
    temp = temp ^ this->num[3];
    temp = temp ^ this->num[5];
    int i;
    for(i = 0; i <= 14; i++){
        this->num[i] = this->num[i+1];
    }
    this->num[15] = '0' + temp;
    return temp;
}

int FibLFSR::generate(int k){
    string output;
    for(int i = 0; i < k; i++){
    output += to_string(this->step());
    }
    int x = stoi(output, nullptr, 2);
    return x;
}
```