Samin Basir
COMP IV: Project Portfolio
Spring 2021

Contents:

Time to complete: ~6 Hours

PS0: Hello World with SFML

Description

This was the first assignment, and it introduced me to SFML. The first step of the assignment was to setup our building environment which led me to download Virtual Machine, which I ultimately used to complete all the assignments on Computing IV. We then had to run a demo code and familiarize ourselves with SFML. The goal of the assignment was to create a moving sprite.

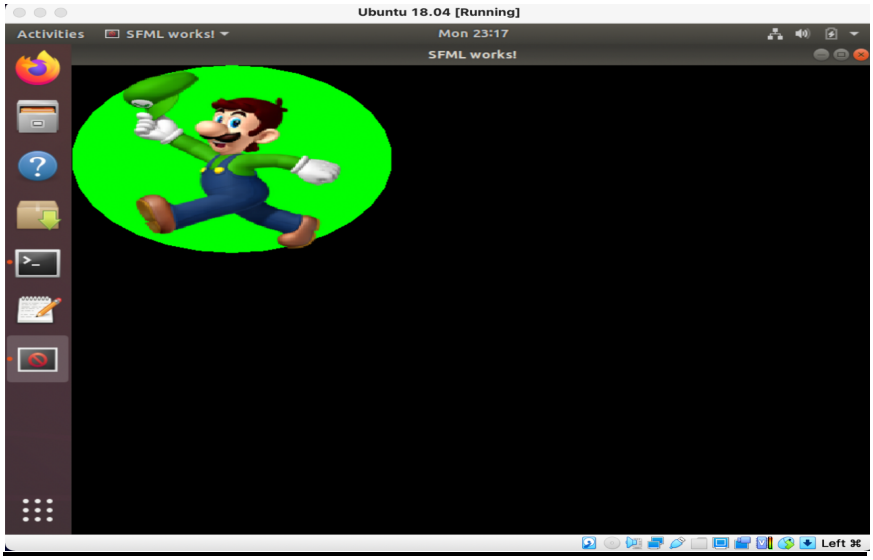
key Algorithms, Data Structures, or OO Designs

Being the first assignment, which was essentially an assignment to familiarize ourselves with SFML, there wasn't much use of Data Structures Although, the key of the assignment was to learn about SFML's window object and its render loop. The loop was running as long as the window was open.

What I have learned

Having never worked with SFML, this was a very informative assignment, and a great introductory assignment to Computing IV. I was able to learn about the SFML library and how to implement it. The takeaways from this assignment helped me in my future assignments for this course.

Evidence that the code ran



Source code

main.cpp:

```cpp
#include <SFML/Graphics.hpp>
#include <iostream>
int main() {
  // Create the window
  sf::RenderWindow window(sf::VideoMode(500, 500), "SFML works!");
  // Load a sprite
  sf::Texture texture;
  if (!texture.loadFromFile( "sprite.png"))
    return EXIT_FAILURE;
  sf::Sprite sprite(texture);
  // Create a circle
  sf::CircleShape shape(100.f);

  shape.setFillColor(sf::Color::Green);
  while (window.isOpen()) {
    sf::Event event;
    while (window.pollEvent(event)) {
      if (event.type == sf::Event::Closed)
  window.close();
    }
    window.clear();

    float offsetX = 0;
    float offsetY = 0;
    // Get current position of the sprite
    sf::Vector2f pos = sprite.getPosition();
    // Move the sprite image (Only if it won't fall off the screen)
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left) && pos.x != 0)
      offsetX = -1;
    else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)
    && pos.x != 400 - 198)
      offsetX = 1;
    else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up) && pos.y != 0)
      offsetY = -1;
    else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down) {
    && pos.y != 400 - 152)
      offsetY = 1;
    } else if (sf::Keyboard::isKeyPressed(sf::Keyboard::R)) {
      sprite.setPosition(0, 0);
      pos.x = pos.y = 0;
    } else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Escape)) {
      window.close();
    // Assign it to a new position
    sprite.setPosition(pos.x + offsetX, pos.y + offsetY);
    // Draw the images
    window.draw(shape);
    window.draw(sprite);
    window.display();
  }
  return 0;
}
```

PS1 part a and b: Linear Feedback Shift Register and Image Encoding

Description:

This was a two-part assignment where we had to create functional linear feedback shift register given a series of bits and an integer for the tap position. The second part of this assignment was to perform unit test it using the Boost unit test framework. Then use LFSR to generate pseudo random values to encode a png. Part a was the first assignment where we were required to submit a makefile, and this the first time I learned how to correctly create a makefile. Also in part a, Register bits were stored as a string, and string member functions were used to access bites. The easiest way for me was converting the individual extracted characters to integers by subtracting 48 in order to get the value of the integer being represented and storing the result in integer variables.

In part b of this assignment, we used the debugged FibLFSR class to encode and decode a png image, this was achieved by using the xor operator, and using the return value of the generate function. Once compiled the transform function displays the initial image and the decoded image where pixels are modified. When doing this each pixel are given new values, hence the modified version differs from the initial. Performing the same function not the modified image reverts the mage back to the initial image.

key Algorithms, Data Structures, or OO Designs

In part a, the use of strings was necessary for reading in user input to determine the bits which the LFSR will be created with. The LFSR requires an initial seed and tap value to run, and it is reversible, so running an image through the program with one set of parameters will output an image file that when passed as input to the program with the same parameters will output the original image.

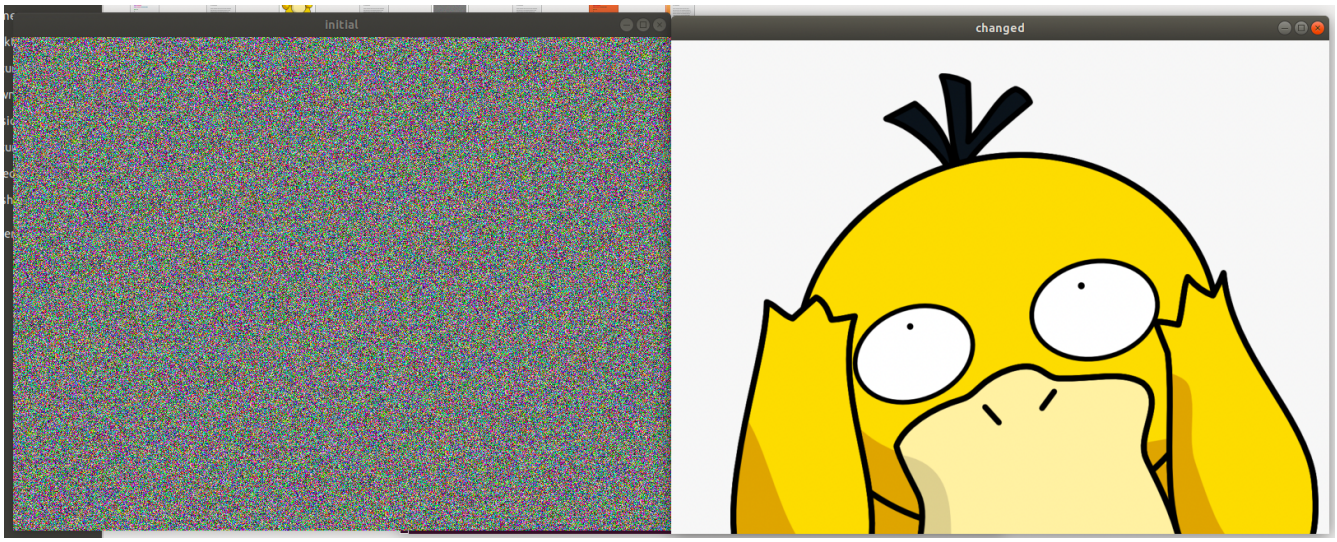
In Part b of this assignment, Strings and vectors were essential in order to create the linear feedback shift register, depending on the implementation. A string was used to store values into the linear

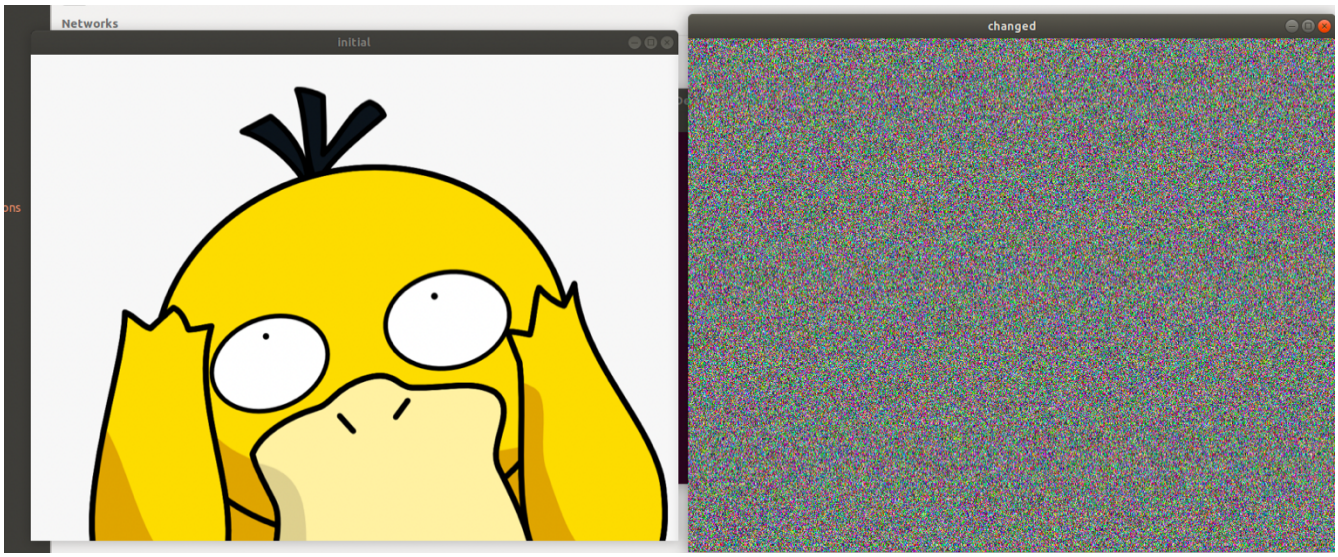
feedback shift register. The LSFR takes a seed and a key, and generates different bit strings that are used to generate a random RBG color and changes each pixel in the image to that color.

What I learned:

The First takeaway from this assignment for me was learning how to make a correct Makefile in part a of the assignment. Overall, I learned how to create a linear feedback shift register with characteristics based on user input. I also learned how to use this linear feedback shift register and its operations to both encrypt and decrypt an image successfully. This assignment taught me more about how SFML objects function on a fundamental level, I also learned about basic image encryption and gained some insight into how the png fille format works.

Evidence that the code ran





Source code

Makefile

```
CC = g++
CFLAGS = -c -g -Og -Wall -Werror -ansi -pedantic
SFML = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system
ALLOBJ = FibLFSR.o PhotoMagic.o
EXE = ps1a PhotoMagic

all: PhotoMagic

PhotoMagic: PhotoMagic.o
$(CC) -Wall PhotoMagic.o  -o PhotoMagic $(SFML)
ps1a: FibLFSR.o
$(CC) FibLFSR.cpp -c FibLFSR.o -o ps1a
clean:
\rm *.o *~ $(EXE) output-file.png
```

FibLFSR.cpp

```
//Samin Basir

#include "FibLFSR.hpp"
#include <iostream>
#include <string>
using namespace std;

FibLFSR::FibLFSR(string seed, int t){
  this->bits = seed;
  this->size = seed.length();
  this->tap = this->size -t-1;
}

int FibLFSR::step(){
```

```
15   //checking for valid index
16   if(this->tap > this->size || this->tap < 0)
17     {
18       return -1;
19     }
20
21   int x, y, z;
22   int i = 0;
23   x = this->bits.at(0) - 48; //furthest bit to the lst
24   y = this->bits.at(this->tap) - 48;
25   z = x ^ y;                 //getting the right most bit
26   while(i < this->size - 1)
27     {
28       this->bits.at(i) = this->bits.at(i+1);
29       i++;
30     }
31   this->bits.at(this->size - 1) = z + 48;  //changing to the result of the previous
32 operation
33
34   return z;
35 }
36
37 int FibLFSR::generate(int k){
38   //check for tap to be a valid index.
39   if(this->tap > this->size || this->tap < 0)
40
41     {
42       return -1;
43     }
44
45   int count = 0;
46   int i;
47   for(i = 0; i < k; i++)
48     {
49       count *= 2;
50       count += this->step();
51     }
52   return count;
53 }
54
55 ostream& operator << (ostream& out, FibLFSR lfsr){
56   int n, i;
57   for(i = lfsr.size - 1; i >= 0; i--)
58     {
59       n *= 2;
60       n += lfsr.bits.at(i) - 48;
61     }
62   out << "Current value of register is: " << lfsr.bits << ' ' << n << endl;
63   return out;
```

FibLFSR.hpp

```
1 //Samin Basir
2 #include <iostream>
```

```cpp
#include <string>
using namespace std;

class FibLFSR{
public:
  FibLFSR(string seed);

  FibLFSR();

  int setSeed(string newSeed);

  int step();

  int generate(int k);

  int bitstoint();

  friend ostream &operator <<(ostream& output, const FibLFSR see);

private:
  string LFSRseed;
};

FibLFSR::FibLFSR(string seed){
  LFSRseed = seed;
}

FibLFSR::FibLFSR(){
}

int FibLFSR::setSeed(string newSeed){
  LFSRseed = newSeed;
  return 0;
}

int FibLFSR::step(){
  char thirteen, twelve, ten, last;
  string newSeed = LFSRseed;

  thirteen = LFSRseed.at(2);
  twelve = LFSRseed.at(3);
  ten = LFSRseed.at(5);
  last = LFSRseed.front();

  if(last == thirteen){last = '0';}// checking for tap location
  else{last = '1';}
  if(last == twelve){last = '0';}
  else{last = '1';}
  if(last == ten){last = '0';}
  else{last = '1';}

  for(int i = 0; i < (LFSRseed.size() - 1); i++){
    if(i == (LFSRseed.size() - 2))
      newSeed.back() = last;
    newSeed.at(i) = LFSRseed.at(i + 1);
```

```
58   }
59
60   LFSRseed = newSeed;
61   return int(last - 48);
62 }
63
64 int FibLFSR::generate(int k){
65   int i, re;
66
67   for(i = 0; i < k; i++){ // running the k function
68     this->step();
69   }
70   re = this->bitstoint();
71   cout << LFSRseed << " " << re << endl;
72   return re;
73 }
74
75 int FibLFSR::bitstoint(){
76   int sum = 0, twoExponent = 1;
77
78   for(int i = 0; i < 8; i++){
79     sum += ((LFSRseed.at(LFSRseed.size() - 1 - i)) - 48) * twoExponent;
80     twoExponent *= 2;
81   }
82
83   return sum;
84 }
85 ostream &operator <<(ostream &output, const FibLFSR see){
86   output << see.LFSRseed;
87   return output;
88 }
```

PhotoMagic.cpp

```
 1 #include <iostream>
 2 #include <string>
 3 #include <cstdlib>
 4 #include <SFML/System.hpp>
 5 #include <SFML/Window.hpp>
 6 #include <SFML/Graphics.hpp>
 7 #include "FibLFSR.hpp"
 8 using namespace std;
 9
10 sf::Image transform(sf::Image image, FibLFSR *bitShift);
11
12 int main(int argc, char *argv[]){
13   int x, y;
14   FibLFSR colorCounter;
15   sf::Image image1, image2;
16
17   if(!image1.loadFromFile(argv[1]))
18     return -1;
19   if(!image2.loadFromFile(argv[1]))
20     return -1;
```

```
21
22   colorCounter.setSeed(argv[argc - 1]);
23
24   image2 = transform(image2, &colorCounter); // transforming the image
25
26   sf::Texture texture1, texture2;
27   texture1.loadFromImage(image1);
28   texture2.loadFromImage(image2);
29
30   sf::Sprite sprite1, sprite2;
31   sprite1.setTexture(texture1);
32   sprite2.setTexture(texture2);
33
34   sf::RenderWindow window1(sf::VideoMode(800, 600), "initial");
35   sf::RenderWindow window2(sf::VideoMode(800, 600), "changed");
36
37   while(window1.isOpen() && window2.isOpen()){
38     sf::Event event;
39     while(window1.pollEvent(event))
40       {
41         if(event.type == sf::Event::Closed)
42            window1.close();
43       }
44     while(window2.pollEvent(event))
45       {
46         if(event.type == sf::Event::Closed)
47            window2.close();
48       }
49
50     window1.clear();// loading the sprite
51     window1.draw(sprite1);
52     window1.display();
53
54     window2.clear();
55     window2.draw(sprite2);
56     window2.display();
57   }
58
59   if(!image2.saveToFile(argv[2]))
60     return -1;
61
62   return 0;
63 }
64
65 sf::Image transform(sf::Image image, FibLFSR *bitShift){
66       sf::Color p;
67       unsigned int x, y;
68       sf::Vector2u v1;
69       v1 = image.getSize();
70
71
72     for(x = 0; x < v1.x; x++){
73         for(y = 0; y < v1.y; y++){
74         p = image.getPixel(x, y);
75
```

```
76          p.r = p.r ^ bitShift->generate(8);
77          p.g = p.g ^ bitShift->generate(8);
78          p.b = p.b ^ bitShift->generate(8);
79          image.setPixel(x, y, p);     // setting the pixel
80          }
81      }
82        return image;
83 }
```

PS2 part a and b: N-Body Simulation

Description:

Part a of this program we were asked to display a static universe using SFML, the instruction provided a file containing the information for the planets that are to be displayed. This program used a vector that contained all of the planets and the sun of the inner part of the solar system and loaded a space background. When it is run, the program displays a window containing the particles.

For part b of this assignment, we were asked to animate the solar system we created on our last assignment. We had to calculate the force of gravity exerted on each body by each other body in a solar system and simulate them in motion. This celestial simulation program reads the initial data and performs calculations based on physics knowledge. I was also able to add sound and time text.

key Algorithms, Data Structures, or OO Designs

For part a, the number of planets n is read from the file, size of the planets and read and stored ex:

```
 "getline(cin, input);

 universeSize = (atof(input.c_str()));", and getline(cin, input);   n =
(atoi(input.c_str())); used to read in the planets.

   for(int i = 0; i < n; i++) {

    getline(cin, input);

    istringstream iss(input);

    iss >> universe[i]; }
```

this was used to read in member variable to corresponding bodies. temporary Body was used as a template for all bodies pushed onto the vector. After creating a vector of bodies and using the temporary body n times, then begin reading information from stdin line by line. After the data is stored input string it is then converted back in to inputstringstream. Once all the is inputted, begin drawing each line:

```
void Body::draw(sf::RenderTarget& target, sf::RenderStates states) const {
  sf::Sprite sprite_temp = sprite;
  double ratio = (windowSize / 2) / universeSize;
  double rxpos = xpos * ratio + (windowSize / 2);
  double rypos = ypos * ratio + (windowSize / 2);
```

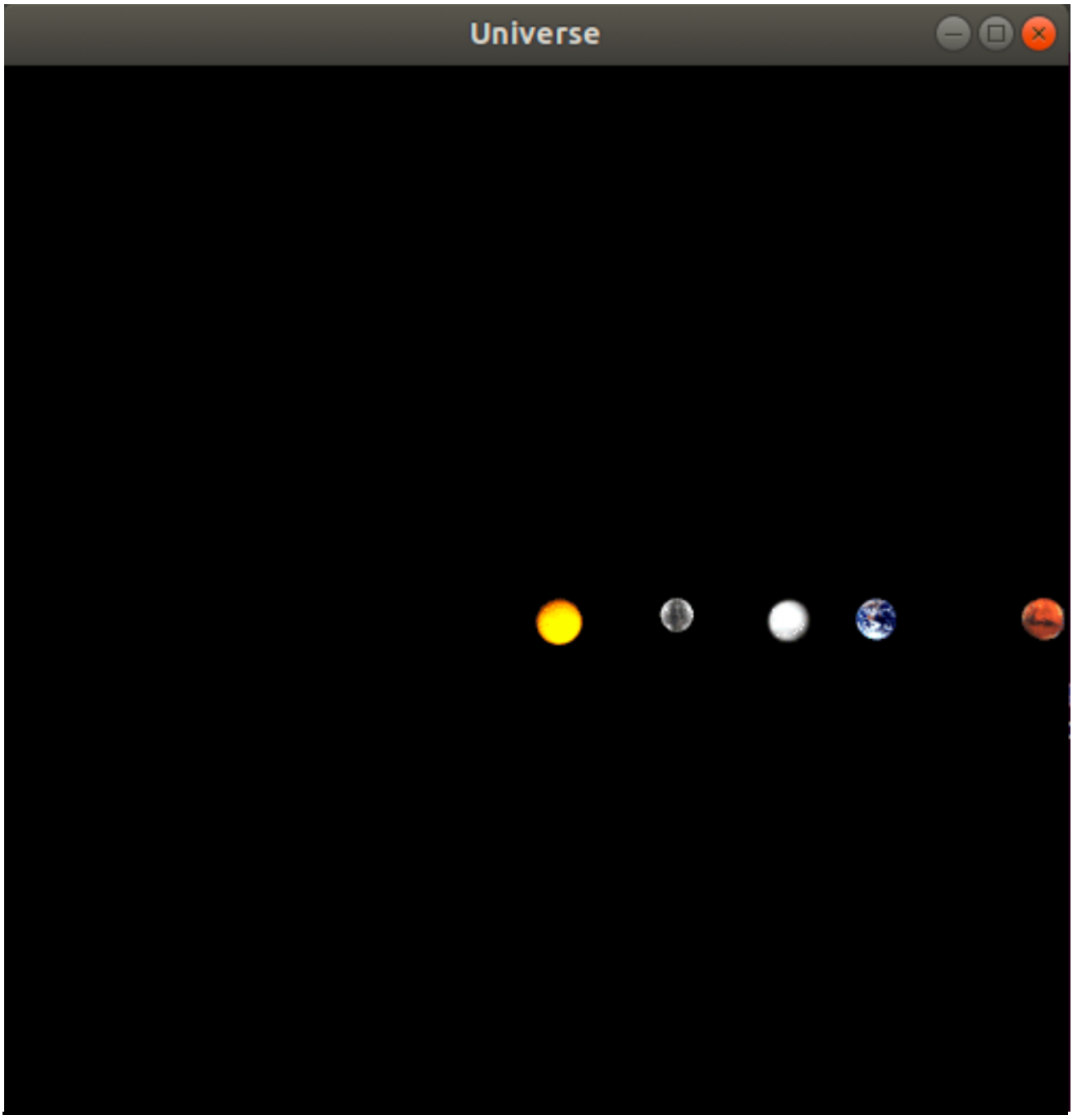
The draw operator simply takes a ratio of the windowSize and universeSize and multiplies it to the planet size. It then adds half the windowSize to the result to shift the window origin to the center instead of the top-left.

What I learned:

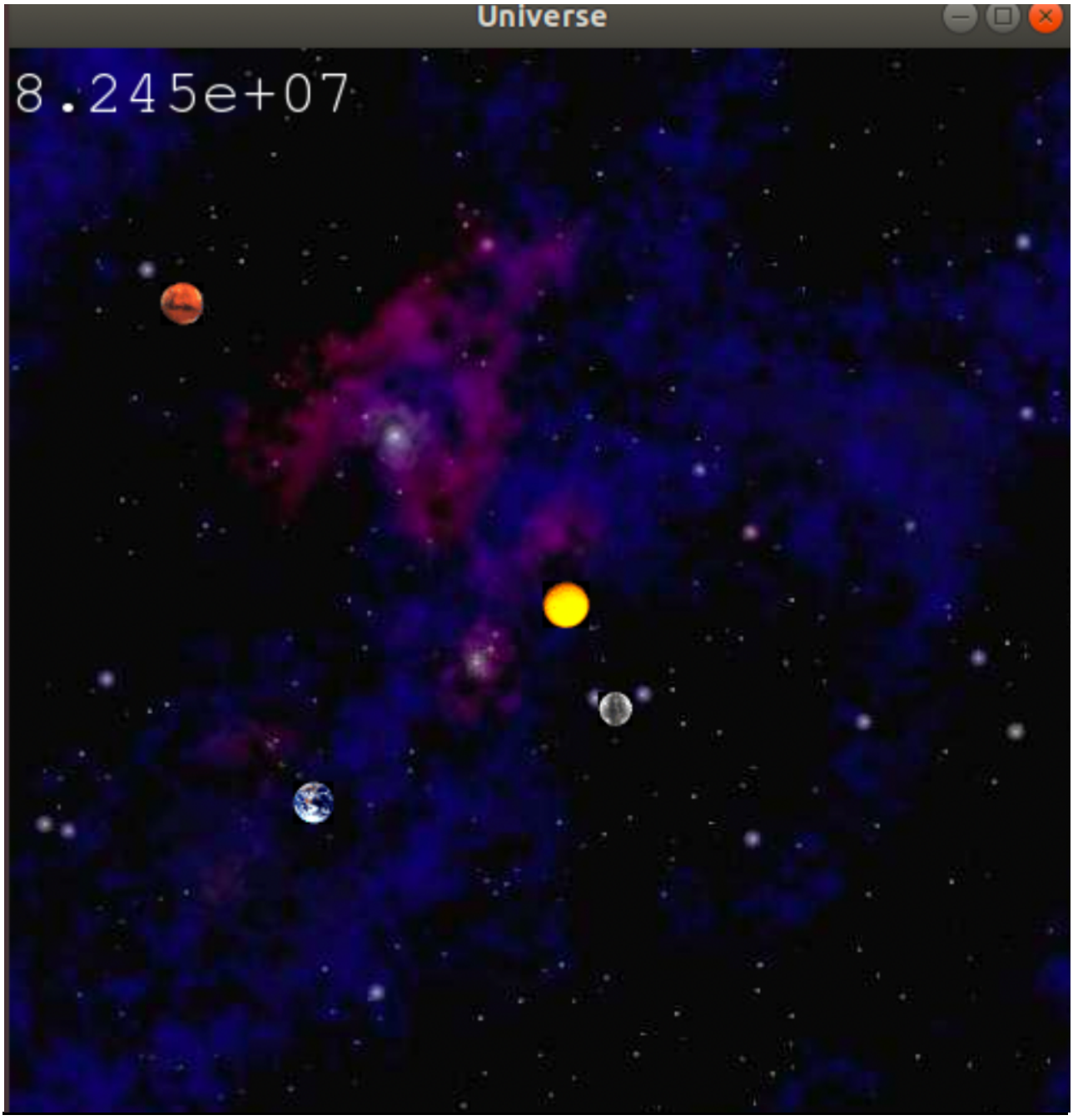
Doing this assignment I learned about smart pointers and convenience of using smartpointers. I learned using smartpointers you can avoid having memoryleaks. I also learned how to create basic physics simulations. I was also able to learn about how to implement sounds using SFML.

Evidence that the code ran

From part a:



From part b:



PS3: DNA Sequence Alignment

Description:

In this assignment we created an optimal sequence alignment between two strings, we essentially measured the similarity of two genetic sequences by their edit distance. This was done by creating a two-dimensional matrix based on the lengths of the input strings. The rightmost column is filled with values from 0 to 2M, where M is the length of the first-string input to the program. matrix. After filling the matrix with values, the edit distance is computed by traversing the matrix. Valgrind was also used in this assignment to analyze time complexity.

Key Algorithms, Data Structures, or OO concepts

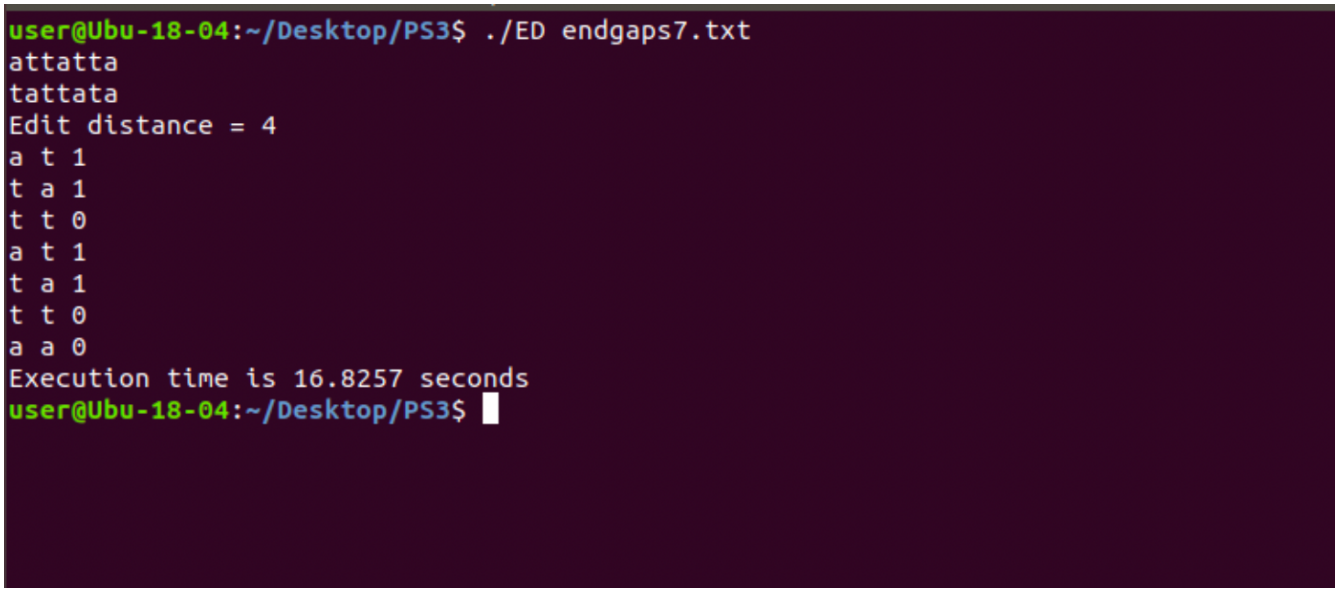
I instead used Hirschbergís algorithm as I found it better implementing it this way then using recursion. Hirschberg's algorithm is a generally applicable algorithm for optimal sequence alignment. The advantage of using this algorithm is that it breaks up a problem and creates sub problems then in turn uses the answer from the Subproblems to answer the initial problem which avoids repeatedly computing the same quantity. One con is that this method can take up a lot of memory. This assignment required the use of strings and vectors. The two DNA sequences were read in from the input file and stored as strings by the program. Two vectors of integers are necessary for storing the matrix and computing the appropriate edit distance. The given algorithm for computing edit distance was also very important for this assignment.

What I learned:

I learned how to use the Hirschbergís algorithm, something I had never heard of before. I also learned how Valgrind analyzes the time complexity and memory usage of a program. Doing this assignment I learned about the utility of these kind of programs. I learned how to optimize code in various ways

when speed at which the program runs is a factor.

Evidence that the code ran:



Source Code

Makefile

```
1 CC= g++
2 CFLAGS= -g -O -Wall -Werror -std=c++11 -pedantic
3 SFLAGS= -lsfml-system
4 all: ED
5 ED: main.o ED.o
6 $(CC) main.o ED.o -o ED $(SFLAGS)
7 main.o: main.cpp ED.cpp ED.hpp
8 $(CC) $(CFLAGS) -c main.cpp
9 ED.o: ED.cpp ED.hpp
10 $(CC) $(CFLAGS) -c ED.cpp
11 clean:
12 rm ED main.o ED.o
```

ED.cpp

```
1 #include <stdlib.h>
2 #include <string>
3 #include <vector>
4 #include <iostream>
5 #include "ED.hpp"
6
7 Edistance::Edistance(std::string s1, std::string s2) {
8     s1L = (s1.length() + 1);
```

```
 9     s2L = (s2.length() + 1);
10     str1 = s1;
11     str2 = s2;
12     int penalty = 0;
13
14
15     matrix = new int*[s1L];
16     for (int i = 0; i < s1L; i++) {
17         matrix[i] = new int[s2L];
18     }
19
20
21     for (int i = 0; i < s1L; i++) {
22         for (int j = 0; j < s2L; j++) {
23             matrix[i][j] = 0;
24         }
25     }
26     for (int j = s2L-1; j >= 0; j--) {
27         matrix[s1L - 1][j] = penalty;
28         penalty += 2;
29     }
30     penalty = 0;
31     for (int i = s1L-1; i >= 0; i--) {
32         matrix[i][s2L - 1] = penalty;
33         penalty += 2;
34     }
35 }
36
37 Edistance::~Edistance() {
38     for (int i = 0; i < s1L; i++) {
39         delete[] matrix[i];
40     }
41     delete[] matrix;
42 }
43
44 int Edistance::penalty(char a, char b) {
45     if (a == b)
46         return 0;
47     else
48         return 1;
49 }
50
51 int Edistance::my_min(int a, int b, int c) {
52     int min = a;
53
54     if (b < min)
55         min = b;
56     if (c < min)
57         min = c;
58
59     return min;
60 }
61
62 int Edistance::OptDistance() {
63     int pen;
```

```
64
65      for (int i = s1L - 2; i >= 0; i--) {
66          for (int j = s2L - 2; j >= 0; j--) {
67              pen = penalty(str1[i], str2[j]);
68              matrix[i][j] = my_min(matrix[i][j + 1] + 2,
69                  matrix[i + 1][j] + 2, matrix[i + 1][j + 1] + pen);
70          }
71      }
72
73      return matrix[0][0];
74  }
75
76  std::string Edistance::Alignment() {
77      std::string combined;
78      int match, i = 0, j = 0, asciichar;
79
80
81      while (i < s1L - 1 || j < s2L - 1) {
82          match = penalty(str1[i], str2[j]);
83
84          if ((i + 1) != s1L && matrix[i+1][j] == (matrix[i][j] - 2)) {
85
86              combined += str1[i];
87              combined += ' ';
88              combined += '-';
89              combined += ' ';
90              combined += 50;
91              combined += '\n';
92              i++;
93          } else if (i + 1 != s1L && j + 1 != s2L && matrix[i+1][j+1] ==
94                                                    matrix[i][j] - match) {
95
96              combined += str1[i];
97              combined += ' ';
98              combined += str2[j];
99              combined += ' ';
100             if (match == 0) {
101                asciichar = 48;
102             } else {
103                 asciichar = 49;
104               }
105             combined += asciichar;
106             combined += '\n';
107             i++;
108             j++;
109         } else if (matrix[i][j+1] == (matrix[i][j] - 2)) {
110
111             combined += '-';
112             combined += ' ';
113             combined += str2[j];
114             combined += ' ';
115             combined += 50;
116             combined += '\n';
117             j++;
118         }
```

```
119     }
120
121     return combined;
122 }
```

ED.hpp

```
 1 #include <SFML/System.hpp>
 2 #include <SFML/Graphics.hpp>
 3 #include <string>
 4 #include <vector>
 5
 6
 7 class Edistance {
 8  public:
 9     Edistance(std::string s1, std::string s2);
10     ~Edistance();
11     static int penalty(char a, char b);
12
13
14     static int my_min(int a, int b, int c);
15
16     int OptDistance();
17
18
19     std::string Alignment();
20
21
22  private:
23     int **matrix;
24     int s1L, s2L;
25     std::string str1, str2;
26 };
```

main.cpp

```
 1 #include <SFML/System.hpp>
 2 #include <iostream>
 3 #include <string>
 4 #include "ED.hpp"
 5
 6 int main(int argc, char* argv[]) {
 7     if (argc < 1) {
 8         std::cout << "ED < [exmample.txt]" << std::endl;
 9         return -1;
10     }
11     sf::Clock clock;
12     sf::Time t;
13
14     int distance;
15
16     std::string s1, s2;
17     std::cin >> s1 >> s2;
```

```
18
19
20      Edistance ED(s1, s2);
21      distance = ED.OptDistance();
22      std::cout << "Edit distance = " << distance << std::endl;
23      std::string aligned = ED.Alignment();
24      std::cout << aligned;
25
26
27      t = clock.getElapsedTime();
28      std::cout << "Execution time is " << t.asSeconds() << " seconds \n";
29      return 0;
30  }
```

PS4 part a and b: Synthesizing a Plucked String Sound

Description:

On part a of this assignment, we were asked to write a program to simulate plucking a guitar string using the Karplus-Strong algorithm. This algorithm played a seminal role in the emergence of physically modeled sound synthesis. The circular queue was implemented and The assignment also consisted of using the BOOST unit testing framework to test the functionality of the member functions. On part b of this assignment, Implement the Karplus-Strong guitar string simulation, and generate a stream of string samples for audio playback under keyboard control.

On part b of this assignment, we were asked to build a fully functional simulation of a Guitar which plays 37 notes of a chromatic scale. Each note corresponds to a specific key on the keyboard. When a key is pressed, the correct note is played. This is accomplished through the use of the Karplus-Strong algorithm. This is done by making parallel vectors of sixteen bit integers. After the Guitar String is plucked and ticked, the sixteen bit integer that represents frequency is then loaded into a sound buffer, and the buffer is then loaded into a sound.

Key Data Structures, Algorithms and OO Design Patterns:

In part a, The constructor for the CicularBuffer essential as its needed to initialize the class, the constructor initializes he capacity integer and throws an error if the capacity is less than one.

In part b, I created the StringSound(double frequency) where if the argument is smaller than or equal to zero it throws an invalid argument and then in void StringSound::tic() function where if the size is smaller than or equal to 1 then it throws an exception. I also implemented lamda function e.g:

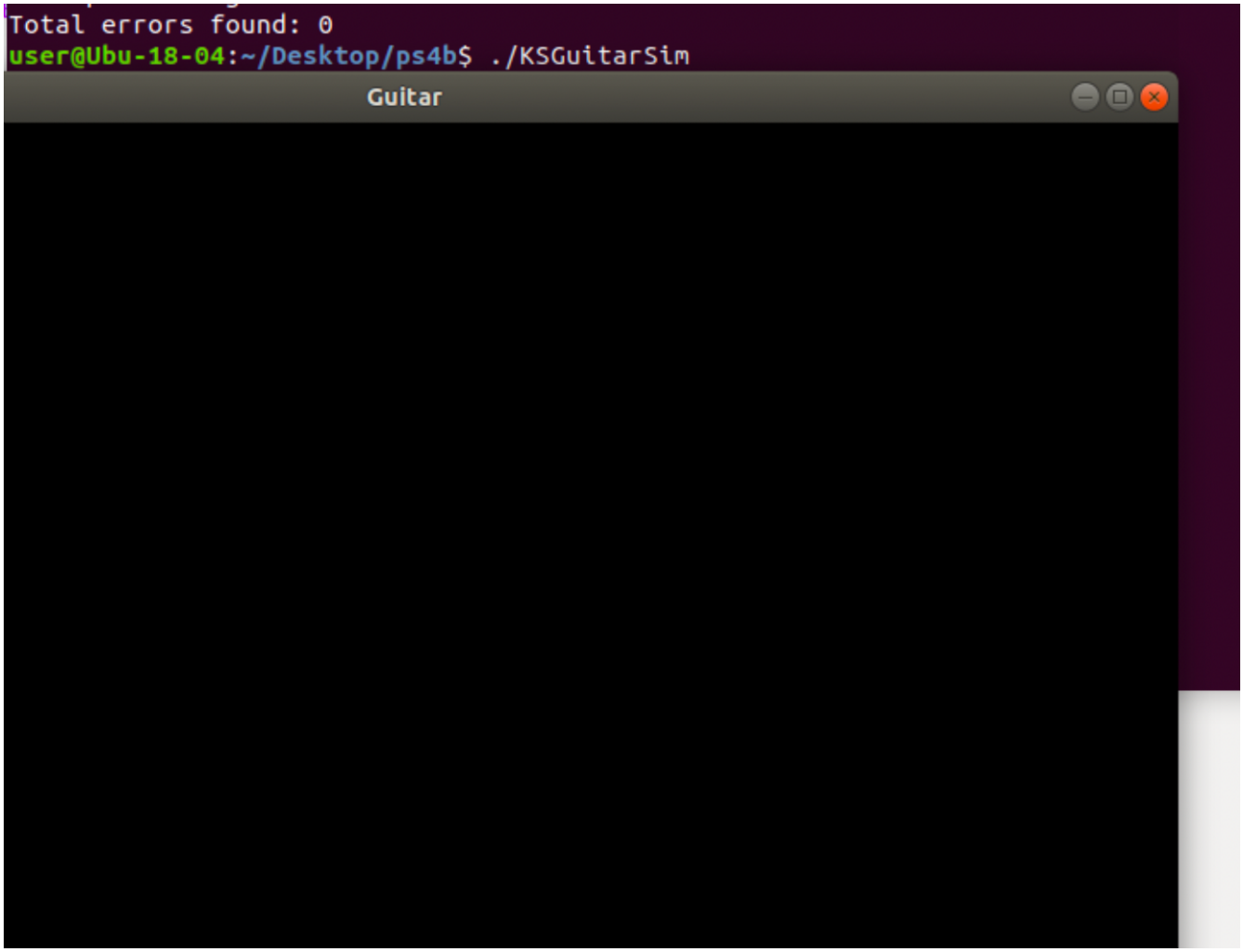
```
void CircularBuffer::enqueue(int16_t x) {
  if ((signed)buffer.size() == capacity)
    throw std::runtime_error("can't enqueue ");
  auto push = [&](int a) { // assigning the lambda expression to an auto
variable
    buffer.push_back(a);
  };
  push(x);
}
```

I also created another executable named extracredit where I changed the frequency so that it makes a Noise sound.

What I Learned:

This was the first assignment where I learned about cpplint, it's an open source lint-like tool developed by Google, designed to ensure that C++ code conforms to Google's coding style guides. I also learned about Boost testing which, allows for users to implement test cases which check whether or not an exception is thrown.

Evidence that the code ran:



Source code:

Makefile

```
CC=g++
CFLAGS=-g -O -Wall -Werror -std=c++11 -pedantic
SFLAGS=-lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
all: KSGuitarSim extracredit
KSGuitarSim: KSGuitarSim.o StringSound.o CircularBuffer.o
$(CC) KSGuitarSim.o StringSound.o CircularBuffer.o -o KSGuitarSim $(SFLAGS)
extracredit: extracredit.o StringSound.o CircularBuffer.o
$(CC) extracredit.o StringSound.o CircularBuffer.o -o extracredit $(SFLAGS)
KSGuitarSim.o: KSGuitarSim.cpp StringSound.h
$(CC) $(CFLAGS) -c KSGuitarSim.cpp
extracredit.o: extracredit.cpp StringSound.h
$(CC) $(CFLAGS) -c extracredit.cpp
StringSound.o: StringSound.cpp StringSound.h
$(CC) $(CFLAGS) -c StringSound.cpp
```

```
15 CircularBuffer.o: CircularBuffer.cpp CircularBuffer.hpp
16 $(CC) $(CFLAGS) -c CircularBuffer.cpp
17 clean:
18 rm KSGuitarSim extracredit KSGuitarSim.o extracredit.o StringSound.o CircularBuffer.o
```

CircularBuffer.cpp

```
 1 #include "CircularBuffer.hpp"
 2 CircularBuffer::CircularBuffer(int capacity) {
 3   std::string error = "The capacity must be greater than zero.";
 4   if (capacity < 1)
 5     throw std::invalid_argument(error);
 6   this->capacity = capacity;
 7 }
 8
 9 int CircularBuffer::size() {
10   return buffer.size();
11 }
12
13 bool CircularBuffer::isEmpty() {
14   if (buffer.size() == 0) {
15     return true;
16   } else {
17     return false;
18   }
19 }
20
21 bool CircularBuffer::isFull() {
22   if ((signed)buffer.size() == capacity)
23     return true;
24   else
25     return false;
26 }
27
28 void CircularBuffer::enqueue(int16_t x) {
29   if ((signed)buffer.size() == capacity)
30     throw std::runtime_error("can't enqueue to a full ring.");
31   auto push = [&](int a) {
32     buffer.push_back(a);
33   };
34   push(x);
35 }
36
37 int16_t CircularBuffer::dequeue() {
38   if (isEmpty())
39     throw std::runtime_error("can't dequeue to an empty ring.");
40   int16_t element = buffer.front();
41   auto erase = [&]() {
42     buffer.erase(buffer.begin());
43   };
44   erase();
45   return element;
46 }
```

```
47
48 int16_t CircularBuffer::peek() {
49   if (isEmpty())
50     throw std::runtime_error("can't peek to an empty.");
51   return buffer.front();
52 }
```

CircularBuffer.hpp

```
 1 #ifndef CircularBuffer_HPP
 2 #define CircularBuffer_HPP
 3
 4 #include <stdint.h>
 5 #include <iostream>
 6 #include <string>
 7 #include <sstream>
 8 #include <exception>
 9 #include <stdexcept>
10 #include <vector>
11
12 class CircularBuffer {
13
14  public:
15   explicit CircularBuffer(int capacity);
16   int size();
17   bool isEmpty();
18   bool isFull();
19   void enqueue(int16_t x);
20   int16_t dequeue();
21   int16_t peek();
22
23  private:
24   std::vector<int16_t> buffer;
25   int capacity;
26 };
27
28 #endif
```

KSGuitarSim.cpp

```
 1 #include <SFML/Graphics.hpp>
 2 #include <SFML/System.hpp>
 3 #include <SFML/Audio.hpp>
 4 #include <SFML/Window.hpp>
 5
 6 #include <math.h>
 7 #include <limits.h>
 8
 9 #include <iostream>
10 #include <string>
11 #include <exception>
12 #include <stdexcept>
13 #include <vector>
14
```

```cpp
#include "CircularBuffer.hpp"
#include "StringSound.h"

using namespace std;
using namespace sf;

#define CONCERT_A 220.0
#define SAMPLES_PER_SEC 44100

vector<Int16> makeSamplesFromString(StringSound gs) {
  vector<Int16> samples;
  gs.pluck();
  int duration=8;
  int i;
  for (i= 0; i<SAMPLES_PER_SEC*duration; i++) {
    gs.tic();
    samples.push_back(gs.sample());
  }
  return samples;
}
int main() {
  RenderWindow window(VideoMode(900, 900), "Guitar");
  Event event;
  double freq;
  string keys = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
  vector<vector<Int16>> samples(37);
  vector<SoundBuffer> buff(37);
  vector<Sound> sound(37);
  for (int i=0; i<37; i++) {
    freq=CONCERT_A*pow(2, ((i-24)/12.0));
    samples[i]=makeSamplesFromString(StringSound(freq));
    buff[i].loadFromSamples(&samples[i][0], SAMPLES_PER_SEC*8, 2, SAMPLES_PER_SEC);
    sound[i].setBuffer(buff[i]);
  }
  while(window.isOpen()) {
    while (window.pollEvent(event)) {
      if(event.type==Event::Closed) {
        window.close();
      }
        if(event.type==Event::TextEntered) {
          char note=(char)event.text.unicode;
            for (int i=0; i<37; i++) {
              if (keys[i]==note) {
                sound[i].play();
              }
            }
        }
      }
    window.clear();
    window.display();
  }
  return 0;
}
```

StringSound.cpp

```cpp
#include "StringSound.h"

StringSound::StringSound(double frequency) {

  if(frequency<=0)
    throw std::invalid_argument("Frequency has to greater than 0.");
  buffer=new CircularBuffer(ceil(SAMPLING_RATE/frequency));
  size=ceil(SAMPLING_RATE/frequency);
  for (int i=0; i<size; i++) {
    buffer->enqueue((int16_t)0);
  }
}

StringSound::StringSound(std::vector<sf::Int16> init) {
  buffer=new CircularBuffer(init.size());
  size=init.size();
  for (int i=0; i<size; i++) {buffer->enqueue((int16_t)init[i]);}
}

StringSound::~StringSound() {
  delete buffer;
}

void StringSound::pluck() {
  for(int i=0; i<size; i++) {buffer->dequeue();}
  for(int i=0; i<size; i++) {
    buffer->enqueue((sf::Int16)(rand()&0xffff));
  }
}

void StringSound::tic() {
  if(size<=1)
    throw std::runtime_error("Size = too small");
  int16_t variable=buffer->dequeue();
  buffer->enqueue((sf::Int16)((variable+buffer->peek())/2)*ENERGY_DECAY_FACTOR);
  Time++;
}

sf::Int16 StringSound::sample() {return (sf::Int16)buffer->peek();}
int StringSound::time() {return Time;}
```

StringSound.h

```cpp
#ifndef STRINGSOUND_H
#define STRINGSOUND_H
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Audio.hpp>
#include <iostream>
#include <cmath>
#include <string>
#include <vector>
```

```
10 #include <math.h>
11 #include "CircularBuffer.hpp"
12
13 const int SAMPLING_RATE = 44100;
14 const double ENERGY_DECAY_FACTOR = 0.996;
15
16 class StringSound {
17
18  public:
19   explicit StringSound(double frequency);
20   explicit StringSound(std::vector<sf::Int16> init);
21   ~StringSound();
22   void pluck();
23   void tic();
24   sf::Int16 sample();
25   int time();
26
27  private:
28   CircularBuffer* buffer;
29   int size, Time=0;
30 };
31
32 #endif
```

extracredit.cpp

```
 1 #include "CircularBuffer.hpp"
 2 #include "StringSound.h"
 3
 4 using namespace std;
 5 using namespace sf;
 6
 7 #define CONCERT_A 440.0
 8 #define SAMPLES_PER_SEC 44100
 9
10 vector<Int16> makeSamplesFromString(StringSound gs) {
11   vector<Int16> samples;
12   gs.pluck();
13   int duration=8;
14   int i;
15   for (i= 0; i<SAMPLES_PER_SEC*duration; i++) {
16     gs.tic();
17     samples.push_back(gs.sample());
18   }
19   return samples;
20
21 }
22
23 int main() {
24   RenderWindow window(VideoMode(900, 900), "Guitar");
25   Event event;
26   double freq;
27   string keys = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
28   vector<vector<Int16>> samples(37);
```

```
29   vector<SoundBuffer> buff(37);
30   vector<Sound> sound(37);
31
32   for (int i=0; i<37; i++) {
33     freq=CONCERT_A*pow(2, ((i-64)/12.0));
34     samples[i]=makeSamplesFromString(StringSound(freq));
35     buff[i].loadFromSamples(&samples[i][0], SAMPLES_PER_SEC*8, 2, SAMPLES_PER_SEC);
36     sound[i].setBuffer(buff[i]);
37   }
38
39   while(window.isOpen()) {
40     while (window.pollEvent(event)) {
41       if(event.type==Event::Closed) {
42         window.close();
43       }
44         if(event.type==Event::TextEntered) {
45           char note=(char)event.text.unicode;
46             for (int i=0; i<37; i++) {
47               if (keys[i]==note) {
48                 sound[i].play();
49               }
50             }
51         }
52       }
53     window.clear();
54     window.display();
55   }
56   return 0;
57 }
```

PS5: Recursive Graphics

Description:

In this assignment we were asked to write a program TFractal.cpp with a recursive function fTree(), and a main() program that calls the recursive function. The program takes two command-line arguments L and N: L length of the side of the base equilateral triangle (double) N the depth of the recursion (int). The main algorithm was recursion, where each set of three triangles was supposed to call itself while the depth was not met.

Key Algorithms, Data Structures, or OO concepts:

The Triangle.cpp defines the function in the class, in the first constructor the midpoint formula was

implemented in order set points for the triangle. And in the second constructor the midpoint formula was used to Set the filled triangle points. The depth was also decremented and checked to see if it is greater than 0. The program takes the command line arguments of L length of the side of the base equilateral triangle (double) and N the depth of the recursion (int). Also color was added. I also made it so that the triangle spins when the mouse scrolls.

For example, in this function:

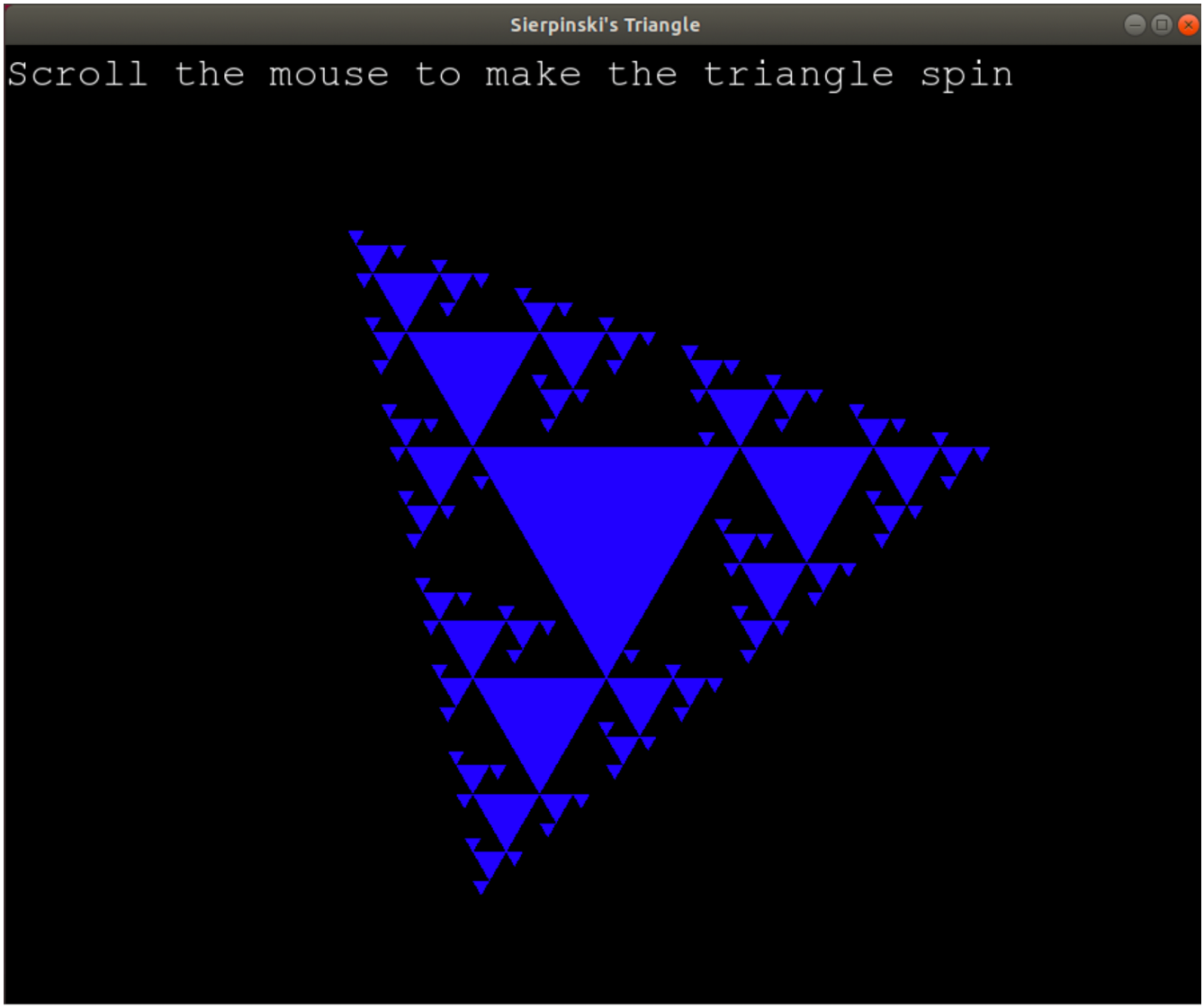
```
void Triangle::draw(sf::RenderTarget& target, sf::RenderStates states)
const {

  target.draw(triangle, states);

}void Triangle::setFillColorBlue() {

  triangle.setFillColor(sf::Color::Blue);
```

To display the results the points of the outside triangle was used first then the points for the upside-down triangle was set. Basically, multiple triangles within one another were drawn. When constructed the Triangle class would create a large triangle and then draw using sprites to the window, smaller level triangles until the depth of recursion have been met.

What I Learned:

I have learned how to use the drawable class. Specifically how to use the SFML framework to draw the Sierpinski triangle, more specifically how to recursively create shapes in using the SFML library. I also learned a fair amount about Fractals. I was also able to implement a function which moves the triangle as the mouse is being scrolled, to do this I referred back to what I learned in the very first assignment.

Evidence that the code ran:



Source code:

Makefile

```
1 CC = g++
2 CFLAGS = -Wall -Werror -ansi -pedantic
3 LFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
4
5 all: Triangle
6
7 Triangle: TFractal.o Triangle.o
8 $(CC) TFractal.o Triangle.o -o Triangle $(LFLAGS)
9
10 TFractal.o: TFractal.cpp Triangle.hpp
11 $(CC) -c TFractal.cpp $(CFLAGS)
```

```
12
13 Triangle.o: Triangle.cpp Triangle.hpp
14 $(CC) -c Triangle.cpp $(CFLAGS)
15
16 clean:
17 rm Triangle
18 rm TFractal.o
19 rm Triangle.o
```

TFractal.cpp

```
 1 // copyright Samin
 2
 3 #include"Triangle.h"
 4 #include<iostream>
 5 #include<vector>
 6 #include<string>
 7 #include<iomanip>
 8
 9 int main(int argc, char* argv[]) {
10   if (argc != 3) {
11       std::cerr << "Invalid command line argument" << std::endl;
12       return -1;
13     }
14   std::string lengthInput = argv[1];
15   std::string NInput = argv[2];
16   float length = std::stof(lengthInput);
17   int N = std::stoi(NInput);
18   float centerY = 0.0f;
19   float centerX = 0.0f;
20   if (N <= 0) {
21       std::cerr << "Input must be positive" << std::endl;
22       return -1;
23     }
24   sf::RenderWindow window(sf::VideoMode(900, 900), "Sierpinski's Triangle");
25   std::vector<Triangle> triangles;
26
27   sf:: Font letters;
28   if (!letters.loadFromFile("cour.ttf"))
29     return -1;
30
31   sf::Text dispText;
32   dispText.setFont(letters);
33   dispText.setFillColor(sf::Color::White);
34   dispText.setCharacterSize(30);
35   dispText.setString("Scroll the mouse to make the triangle spin ");
36   sf::Vector2u size = window.getSize();
37   centerX = size.x / 2;
38   centerY = size.y / 2;
39
40   Triangle triangle(length, centerX, centerY);
41
42   fTree(length, N, triangles, triangle, centerX, centerY);
```

```
  while (window.isOpen()) {
    sf::Event event;
    while (window.pollEvent(event)) {
      if (event.type == sf::Event::Closed)
      window.close();

      if (event.type == sf::Event::KeyPressed) {
              if (event.key.code == sf::Keyboard::Escape) {
                 window.close();
               }
            }
       if (event.type == sf::Event::MouseWheelMoved) {
              if (event.mouseWheel.x) {
            for (std::size_t i = 0; i < triangles.size(); i++) {
                      triangles.at(i).setRotation();
               }
            }
            }
    }
    window.clear();
    window.draw(dispText);

    for (std::size_t i = 0; i < triangles.size(); i++) {
         window.draw(triangles.at(i));
      }
    window.display();
  }
  return 0;
}
void fTree(float length, int N,
 std::vector<Triangle>& triangles, Triangle triangle,
 float centerX, float centerY) {
  float height = (length * sqrt(3)/2);
  float aX, aY, bX, bY, cX, cY;
  if (N == 1) {
      triangles.push_back(triangle);
    } else {
      aX = (centerX - (length/2));
      aY = (centerY - ((2 * height) / 3));
      Triangle triangle1(length/2, aX, aY);
      triangles.push_back(triangle1);
      fTree(length/2, N - 1, triangles, triangle, aX, aY);

      bX = (centerX + ((3.0/4.0) * length));
      bY = (centerY - ((1 * height) / 6));
      Triangle triangle2(length/2, bX, bY);
      triangles.push_back(triangle2);
      fTree(length/2, N - 1, triangles, triangle, bX, bY);

      cX = (centerX - ((1.0/4.0) * length));
      cY = (centerY + ((5 * height) / 6));
      Triangle triangle3(length/2, cX, cY);
      triangles.push_back(triangle3);
      fTree(length/2, N - 1, triangles, triangle, cX, cY);
    }
```

```
98 }
```

Triangle.cpp

```
 1 // Copyright Samin
 2
 3 #include "Triangle.h"
 4
 5 Triangle::Triangle(float length, float centerX, float centerY) {
 6   height = (length * sqrt(3)/2);
 7   triangle.setPointCount(3);
 8
 9   triangle.setPoint(0, sf::Vector2f((centerX - (length/2)),
10   (centerY - (height/3))));
11   triangle.setPoint(1, sf::Vector2f((centerX + (length/2)),
12   (centerY - (height/3))));
13   triangle.setPoint(2, sf::Vector2f((centerX),
14   (centerY + (((2 * height)/3)))));
15
16   triangle.setFillColor(sf::Color::Blue);
17 }
18 void Triangle::draw(sf::RenderTarget& target, sf::RenderStates states) const {
19   target.draw(triangle, states);
20 }
21 void Triangle::setFillColorBlue() {
22   triangle.setFillColor(sf::Color::Blue);
23 }
24 void Triangle::setRotation() {
25   triangle.setOrigin(450, 450);
26   triangle.setPosition(450, 450);
27   triangle.rotate(5.f);
28 }
```

Triangle.h

```
 1 // Copyright Samin
 2 #ifndef _HOME_USER_DESKTOP_PSX_PS5_TRIANGLE_H_
 3 #define _HOME_USER_DESKTOP_PSX_PS5_TRIANGLE_H_
 4 #include <iostream>
 5 #include<cmath>
 6 #include<vector>
 7 #include <SFML/System.hpp>
 8 #include <SFML/Window.hpp>
 9 #include <SFML/Graphics.hpp>
10 #include <SFML/Graphics/Export.hpp>
11 #include <SFML/Graphics/RenderStates.hpp>
12
13 class Triangle : public sf::Drawable, public sf::Transformable {
14  public:
15   Triangle() {}
16   Triangle(float length, float centerX, float centerY);
17   void setFillColorBlue();
18   void setRotation();
19
```

```
20 private:
21   float height;
22   sf::ConvexShape triangle;
23   virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const;
24 };
25
26 void fTree(float length, int N, std::vector<Triangle> &triangles,
27 Triangle triangle, float centerX, float centerY);
28
29 #endif  // _HOME_USER_DESKTOP_PSX_PS5_TRIANGLE_H_
```

PS6: Kronos Time Clock

Description:

In this assignment we had to write a program that reads through a log file from a Kronos Time Clock, parse the log file correctly, and output a text file report chronologically using expression parsing and the Boost library. I used Boost date and time functions to compute elapsed time between the server boot and boot completion. Two regular expressions were used to search for matches between an expected start message and expected end message. The program searches for error messages and outputs whether the boot was a success or failure by writing the information to the file.

Key Data Structures, Algorithms and OO Design Patterns:

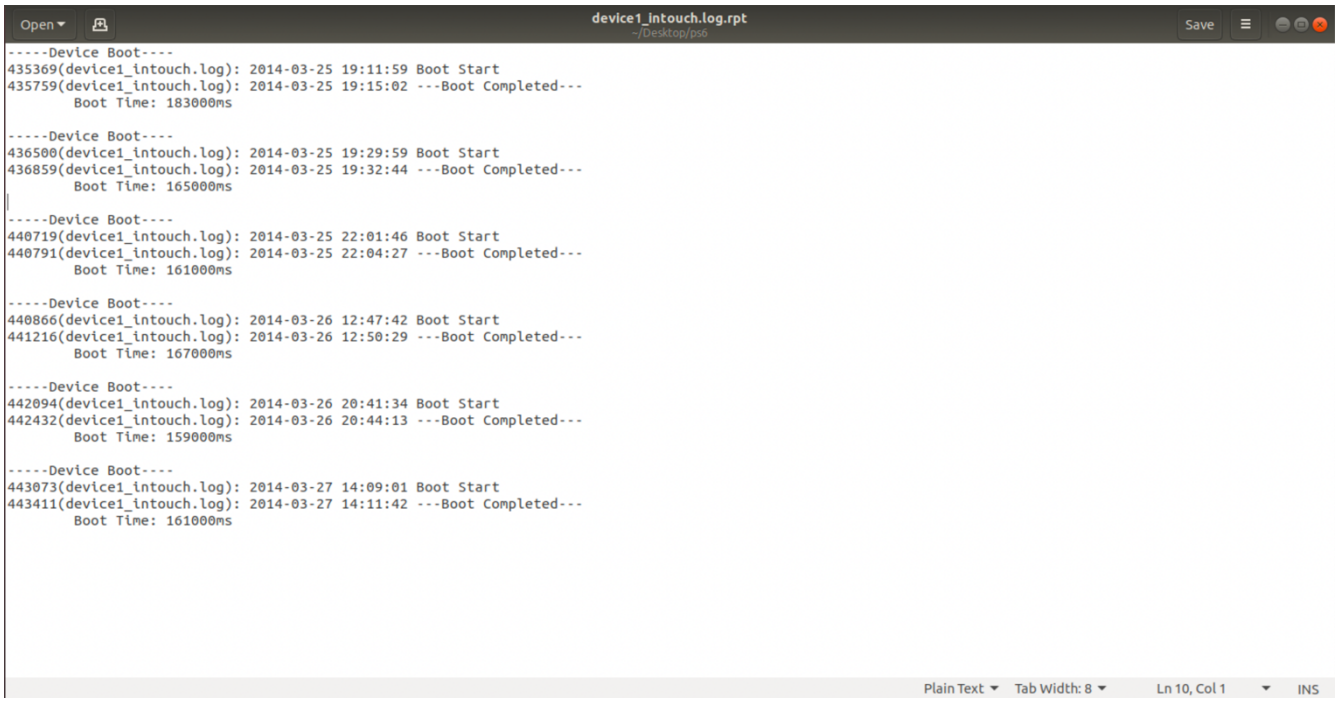
The two regexes were created to search for the strings signaling a device has begun to boot up and the string signaling a device has finished booting. Then the entire log was read, attempting to parse it with multiple conditions in the form of if statements. the commad line argument was used to help assessing which files need to read and write to. variables were created from the date class to store the date. vectors were used to store hours minutes and seconds. More specifically the vector's element was used to put the line information in the other two vectors, and used the string containing the service complete line to grab the elapsed time and output.

What I Learned:

This assignment introduced me to regular expressions, which were vital to this assignment because of the need to search for specific occurrences of strings in the log file. and large text file parsing. I also learned what regexes are.

Evidence that the code ran:

When running the command "./ps6 device1_intouch.log" I get the report:



Source code:

Makefile

```
C=g++ -g -Wall --std=c++98 -Werror
E=.cpp
O=main.o
P=ps6
BOOST= -lboost_regex -lboost_date_time
all: $(P)
$(P):$(O)
$(C) -o $(P) $(O) $(BOOST)

$(E).o:
```

```
11 $(C) -c $<  -o $@
12
13 clean:
14 rm $(O) $(P)
```

main.cpp

```
 1 // copyright Samin Basir
 2
 3 #include <boost/regex.hpp>
 4
 5 #include <iostream>
 6 #include <string>
 7 #include <cstdlib>
 8 #include <fstream>
 9 #include <vector>
10
11 #include "boost/date_time/gregorian/gregorian.hpp"
12 #include "boost/date_time/posix_time/posix_time.hpp"
13
14 using std::cout;
15 using std::endl;
16 using std::string;
17 using std::ifstream;
18 using std::ofstream;
19
20 using boost::regex;
21
22 using boost::gregorian::date;
23 using boost::gregorian::from_simple_string;
24
25 using boost::posix_time::ptime;
26 using boost::posix_time::time_duration;
27
28
29 template <typename T>
30 int to_int(const T& sm) {
31   return atoi(sm.str().c_str());
32 }
33
34 int main(int argc, char* argv[]) {
35   if (argc != 2) {
36     cout << "Command Line argument invlaid" << endl;
37     return 0;
38   }
39
40
41   ifstream inputFile(argv[1], ifstream::in);  // if device can't open
42   if (!inputFile.is_open()) {
43     cout << "Unable to open file \"" << argv[1] << "\"" << endl;
44     return 0;
45   }
46
47   string outputFileName(string(argv[1]) + ".rpt");
```

```
48  ofstream outputFile;
49  outputFile.open(outputFileName.c_str());
50
51
52  string s_date("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2}) ");
53  string s_time("([0-9]{2}):([0-9]{2}):([0-9]{2})");
54  string s_boot("(.*log.c.166.*)");
55  string s_done("(.*oejs.AbstractConnector:Started SelectChannelConnector.*)");
56
57
58  regex r_boot(s_date + s_time + s_boot);  // regex expression
59  regex r_done(s_date + s_time + s_done);
60
61
62  boost::smatch m;
63
64
65  string line;
66  int line_number = 1;
67  bool booting = false;
68  ptime t1, t2;
69
70  while (getline(inputFile, line)) {
71  if (regex_match(line, m, r_boot)) {
72      if (booting)
73        outputFile << "----Boot Incomplete---- \n" << endl;
74
75      date d(from_simple_string(m[0]));
76      ptime temp(d, time_duration(to_int(m[4]), to_int(m[5]), to_int(m[6])));
77      t1 = temp;
78
79      outputFile << "-----Device Boot----" << endl;
80      outputFile << line_number << "(" << argv[1] << "): ";
81      outputFile << m[1] << "-" << m[2] << "-" << m[3] << " ";
82      outputFile << m[4] << ":" << m[5] << ":" << m[6] << " ";
83      outputFile << "Boot Start" << endl;
84      booting = true;
85
86    } else if (regex_match(line, m, r_done)) {
87      if (booting) {
88        date d(from_simple_string(m[0]));
89        ptime temp(d, time_duration(to_int(m[4]), to_int(m[5]), to_int(m[6])));
90        t2 = temp;
91
92        time_duration td = t2 - t1;
93
94        outputFile << line_number << "(" << argv[1] << "): ";
95        outputFile << m[1] << "-" << m[2] << "-" << m[3] << " ";
96        outputFile << m[4] << ":" << m[5] << ":" << m[6] << " ";
97        outputFile << "---Boot Completed---" << endl;
98
99        outputFile << "\t" << "Boot Time: ";
100       outputFile << td.total_milliseconds() << "ms \n" << endl;
101
102       booting = false;
```

```
103       } else {
104         outputFile << "---Unexpected Boot----*\n" << endl;
105       }
106     }
107 
108     line_number++;
109   }
110 
111   return 0;
112 }
```