Name: Srikanth Reddy Gogulamudi
COMP IV: Project Portfolio
Fall 2021

Contents:

Time to Complete: 7 hours

PS0: Hello World

The Assignment

Hello World was our first Computing IV assignment. The main goal of this assignment was to setup our Linux build environment and to test out the SFML audio/graphics library. This included getting Linux running – either through a Virtualbox image or natively, and running some SFML example code to test out SFML. We then had to extend the demo code to make it do something interesting. I was already familiar with Linux at this point, so it really didn't take very long to setup my environment – a few sudo apt-get install commands and I had SFML ready to test out.

The keys are used to result the following changes for the image

W – Move Image Upward Direction

S - Move Image Downward

D – Move Image Left Direction

A – Move Image Right Direction

Q – Terminate the Program

P – Pause the Music

Z – Play the Music

Key Concepts

I did not utilize any algorithms to develop the program because it was my first assignment. Mostly simply fundamental programming concepts like variables, objects, and a few while / for loops to keep the window open and move the sprites around. After all, the purpose of this assignment was to experiment with SFML's many classes, such as Images, Sprites, Text, Texture, and Keyboard. You must load a picture into a Texture, and a sprite must set a Texture to be used.

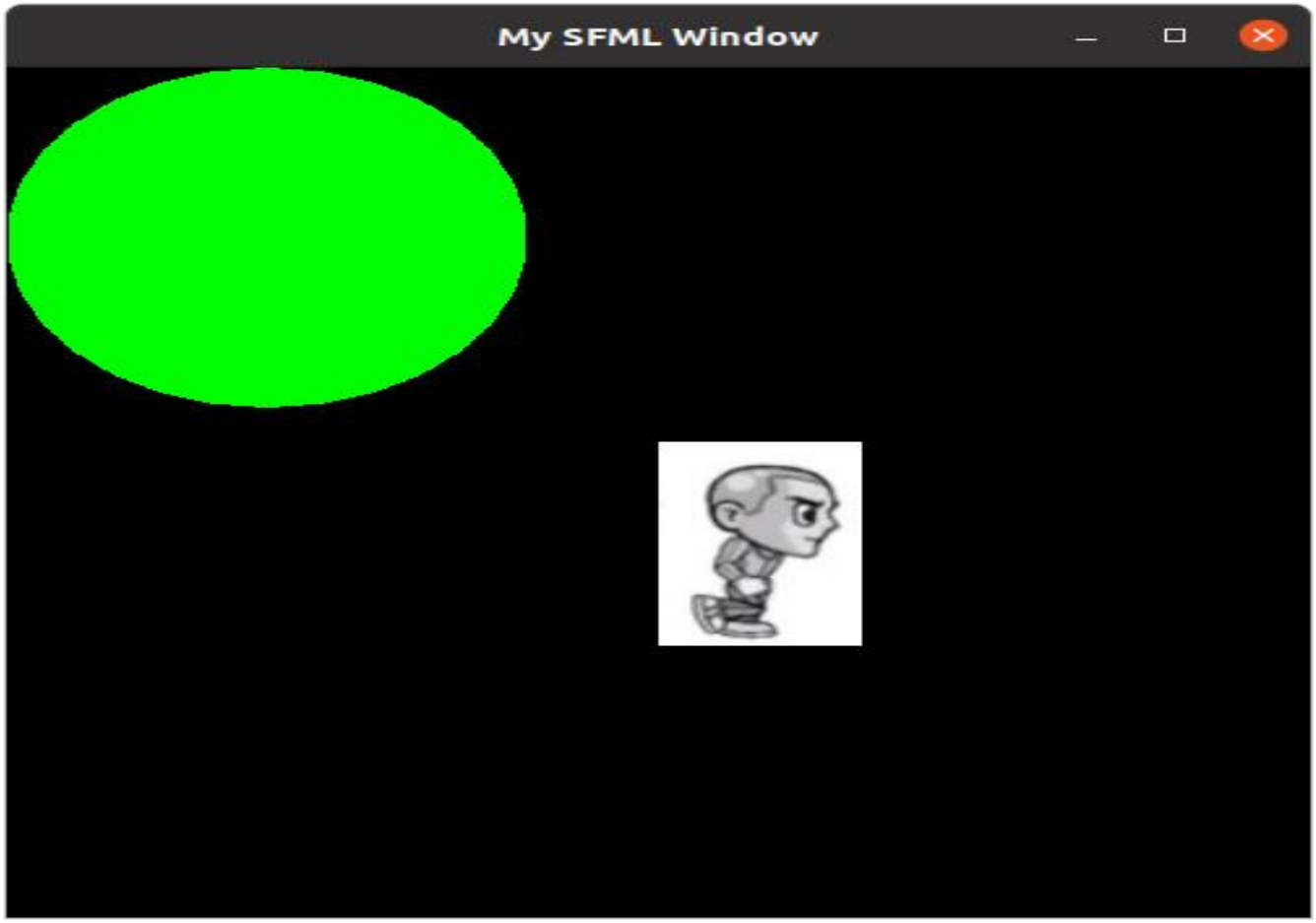
For this project, I constructed a software that displays a moving image on an SFML window using SFML's sprite class. I added some simple controls to move it about — arrow keys for up/down/left/right, plus and minus keys to raise and reduce the size of the sprite, respectively. I also played around with several other SFML features, such as getting text to appear on the screen and making a second image move about in a predetermined way.

What I Learned

This homework taught me a few things about SFML. I used in this assignment were a couple of if statements enclosed within while loops, which controlled the opening and closing of the different SFML windows (i.e. if (event.type == sf::Event::Closed) { window.close(); }). A second important algorithm I used in this was an if statement to control the movement of the object. A series of if statements is used to determine whether the left, right, up, or down keys are pressed and if they are, the shape will move (e.g. if (sf::Keyboard::isKeyPressed()) { shape.move() }). I learnt how to utilize SFML at a fundamental level, how to display graphics

in an SFML window, and even how to control sprites using SFML's Keyboard library, to name a few things. I learned very nothing about Linux or how to set up a Linux build environment.

Screenshots



Source code for PS0 Hello World

```
#include<SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
int main()
{
  int direction=1,shiftx=70;

  sf::RenderWindow window(sf::VideoMode(500,500),"My SFML
Window",sf::Style::Close|sf::Style::Default|sf::Style::Resize);

  sf::Clock clock;
```

```
11  sf::Event event;
12  sf::Texture texture;
13
14 ///Users/srikanthreddygogulamudi/Desktop/SFML/
15  if(!texture.loadFromFile("sprite.jpeg"))
16  return EXIT_FAILURE;
17
18
19
20  sf::Sprite sprite2(texture);
21
22  sf::IntRect rectSourceSprite(0,0,78,120);
23  sf::Sprite sprite(texture, rectSourceSprite);
24  sprite.setOrigin(-250,-220);
25
26  sf::CircleShape shape(100.f);
27    shape.setFillColor(sf::Color::Green);
28
29  sf::Music music;
30  ///Users/srikanthreddygogulamudi/Desktop/SFML/
31    if (!music.openFromFile("1.ogg"))
32        return EXIT_FAILURE;
33        music.play();
34
35  while(window.isOpen()){
36    while(window.pollEvent(event))
37        if (event.type == sf::Event::Closed)
38              window.close();
39
40              if(sf::Keyboard::isKeyPressed(sf::Keyboard::Q))
41              {
42                 exit(0);
43              }
44              if(sf::Keyboard::isKeyPressed(sf::Keyboard::W))
45              {
46                sprite.move(0,-0.1);
47              }
48              if(sf::Keyboard::isKeyPressed(sf::Keyboard::S))
49              {
50                 sprite.move(0,0.1);
51              }
52              if(sf::Keyboard::isKeyPressed(sf::Keyboard::D))
53              {
54                 sprite.move(0.1,0);
55              }
56              if(sf::Keyboard::isKeyPressed(sf::Keyboard::A))
57              {
58                 sprite.move(-0.1,0);
59              }
60              if(sf::Keyboard::isKeyPressed(sf::Keyboard::P))
61              {
62                 music.stop();
```

```
            }
            if(sf::Keyboard::isKeyPressed(sf::Keyboard::R))
            {
               music.play();
            }

        if (clock.getElapsedTime().asSeconds() > 1.0f){
        if (rectSourceSprite.left == 50)
          rectSourceSprite.left = 0;
        else
        {
          rectSourceSprite.left += 70;
          sprite.setTextureRect(rectSourceSprite);
          clock.restart();
          if(rectSourceSprite.left>400)
            rectSourceSprite.left = 0;
        }
    }

    window.clear();
    window.draw(shape);
    window.draw(sprite);
    window.display();

  }
}
```

PS1a: Linear Feedback Shift Register and Unit Testing

The Assignment

Princeton's Linear Feedback Shift Register was required for this project. After shifting all bits left one position, this form of register XORs the leftmost bit and the seed bit to fill the vacant space on the far right side. Our major objectives were to create the shift register in a class called "LFSR" and to use the Boost test framework to implement multiple unit tests. In addition, the Boost test framework was used to check that the newly built LFSR class functioned correctly and produced the desired output. The LFSR works by taking a linear function of the previous state as input, performing discrete step operations to shift the bits one position to the left, and replacing the vacated bit with the XOR of the previously shifted off bit and the bit previously at the provided tap location in the register. To use this method to encode an image, we must first set the state of the register to the encryption key. We can use the LFSR to generate 8-bit integers repeatedly and overwrite the original pixel value with an 8-bit integer formed by bitwise XORing the original pixel value with the generated integer value because each pixel in an image is represented by an 8-bit value. As a result, the final product will be a garbled and encrypted image, which may be used to repeat the process of ungarbling and decrypting the original image.

Key Concepts

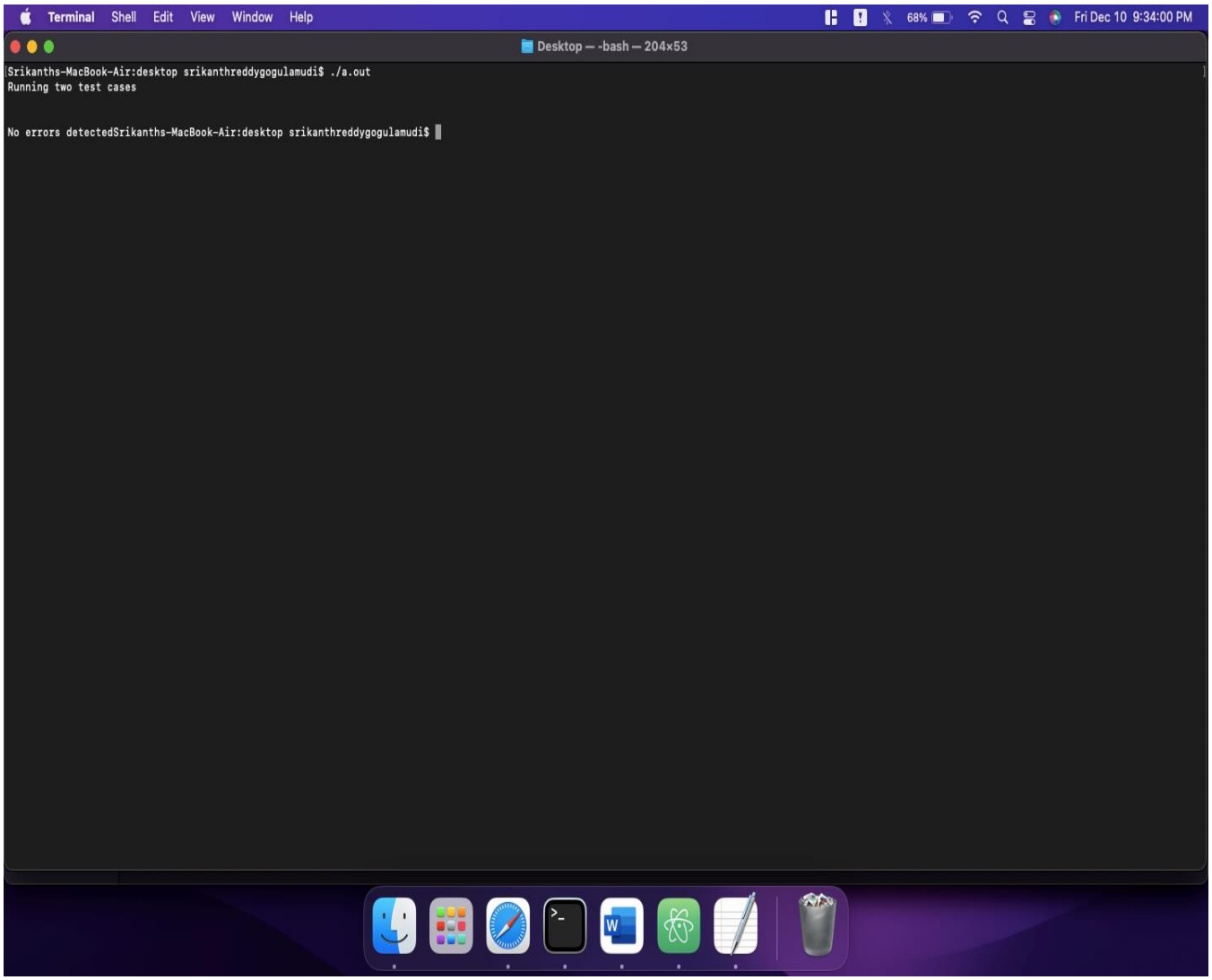
Important Concepts We utilized the Boost test framework to test our LFSR class, in which the shift register is represented as a C++ string in my code. I also utilized an integer to store the tap location, which allowed me to build the step and produce functions using C++'s string and ostringstream classes. Shifting left was performed by attaching the result of XORing the leftmost bit with the tap position to the string representing the register in an ostringstream object. The Boost test framework was used to test our LFSR class, with the step / generate methods being tested against edge situations, common scenarios, and even some of Princeton's test cases utilizing Boost's auto rest case methods.

What I Learned

This assignment taught me a lot about testing in C++. I had never really thought much about testing my code using unit tests – in the past, I've done a combination of compiling, making sure the program runs, and then manually testing different aspects to see if it looks "OK". It was the exclusive or implementation algorithm that was essential to this program. I used two variables, one representing the 8th bit and one representing the 10th bit. In the case of equal bits, the rightmost integer will be 0, but if the bits are different, the rightmost integer will be 1. That establishes the control for the LFSR. The one that calculates the value for the generate() function is another important algorithm. To accomplish this, I would use the vector to observe whether or not the value at that particular bit is 1 or 0. Using this information, I would create a

grand total corresponding to the bit's placement (i.e. if (vector[10] == 1) then [total + 1]) (note:different variables are defined in source code.)

SCREENSHOT :



Source Code for PS1a Linear Feedback Shift Register

Makefile

```
# Makefile for ps1a
# Flags to save on typing all this out
CC= g++
CFLAGS= -Wall -Werror -ansi -pedantic
Boost= -lboost_unit_test_framework

# Make both projects
all: main.out ps1a

# Boost unit tests
ps1a: test.o FibLFSR.o
	$(CC) test.o FibLFSR.o -o ps1a $(Boost)

test.o: test.cpp FibLFSR.hpp
	$(CC) -c test.cpp FibLFSR.hpp $(CFLAGS)

#Main tester
main.out: main.o FibLFSR.o
	$(CC) main.o FibLFSR.o -o main.out

main.o: main.cpp FibLFSR.hpp
	$(CC) -c main.cpp $(CFLAGS)

FibLFSR.o: FibLFSR.cpp FibLFSR.hpp
	$(CC) -c FibLFSR.cpp $(CFLAGS)

# Cleanup
clean:
	rm *.o
	rm *.out
	rm ps1a
```

FibLFSR.cpp

```
/*
NAME OF THE STUDENT: SRIKANTH REDDY GOGULAMUDI
NAME OF THE COURSE : Computing IV (COMP 2040)
PS0 Assignment
INSTRUCTOR: Dr. YELENA RYKALOVA
DUE DATE OF THE ASSIGNMENT: 20 SEPTEMBER 2021 12:00 AM
*/
#include <iostream>
#include <string>
#include <sstream>
#include "FibLFSR.hpp"

FibLFSR::FibLFSR(std::string seed){   bits = seed;}
int FibLFSR::step()
{

        int r= bits[0]^bits[2];
        r= r^bits[3];
        r= r^bits[5];
        std::string::size_type i;
        std::ostringstream ostring;
        for(i = 0; (unsigned)i < bits.length()-1;i++)
        {
                ostring << bits[i+1];
        }
        ostring << r;
        bits = ostring.str();
        return r;
}
std::ostream& operator<< (std::ostream &out, FibLFSR &cFibLFSR)
{
        out << cFibLFSR.bits;
        return out;
}
int FibLFSR::generate(int k)
{
        int x = 0;
        for(int i = 0;i<k;i++)
        {
                x=(x*2)+step();
        }
        return x;
}
```

LFSR.hpp

```
#ifndef FibLFSR_HPP
#define FibLFSR_HPP
#include <iostream>
class FibLFSR
{
         public:
                FibLFSR(std::string seed);
                int step();
                int generate(int k);
        friend std::ostream& operator<< (std::ostream &out, FibLFSR &cFibLFSR);
        private:
                std::string bits;
}
#endif
```

test.cpp

```
#include <iostream>
#include <sstream>
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include "FibLFSR.hpp"
#include <string>

BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps)
{
        FibLFSR test("1011011000110110");
        BOOST_REQUIRE(test.step() == 0);
        BOOST_REQUIRE(test.step() == 0);
        BOOST_REQUIRE(test.step() == 0);
        BOOST_REQUIRE(test.step() == 1);
        BOOST_REQUIRE(test.step() == 1);
        BOOST_REQUIRE(test.step() == 0);
        BOOST_REQUIRE(test.step() == 0);
          BOOST_REQUIRE(test.step() == 1);
        FibLFSR test2("1011011000110110");
        BOOST_REQUIRE(test2.generate(9) == 51);
}

BOOST_AUTO_TEST_CASE(TEST2)
{
        FibLFSR test("01101000010");
        BOOST_REQUIRE(test.step() == 1);
        BOOST_REQUIRE(test.step() == 0);
        BOOST_REQUIRE(test.step() == 0);
        BOOST_REQUIRE(test.step() == 0);
        FibLFSR test2("01101000010");
        BOOST_REQUIRE(test2.generate(5) == 16);
}

BOOST_AUTO_TEST_CASE(TEST3)
{
        FibLFSR test("01101000010");
        BOOST_REQUIRE(test.step() == 1);
        BOOST_REQUIRE(test.step() == 0);
        BOOST_REQUIRE(test.step() == 0);
        FibLFSR test2("01101000010");
        BOOST_REQUIRE(test2.generate(4) == 8);
}
```

PS1b: Image Encoding

The Assignment

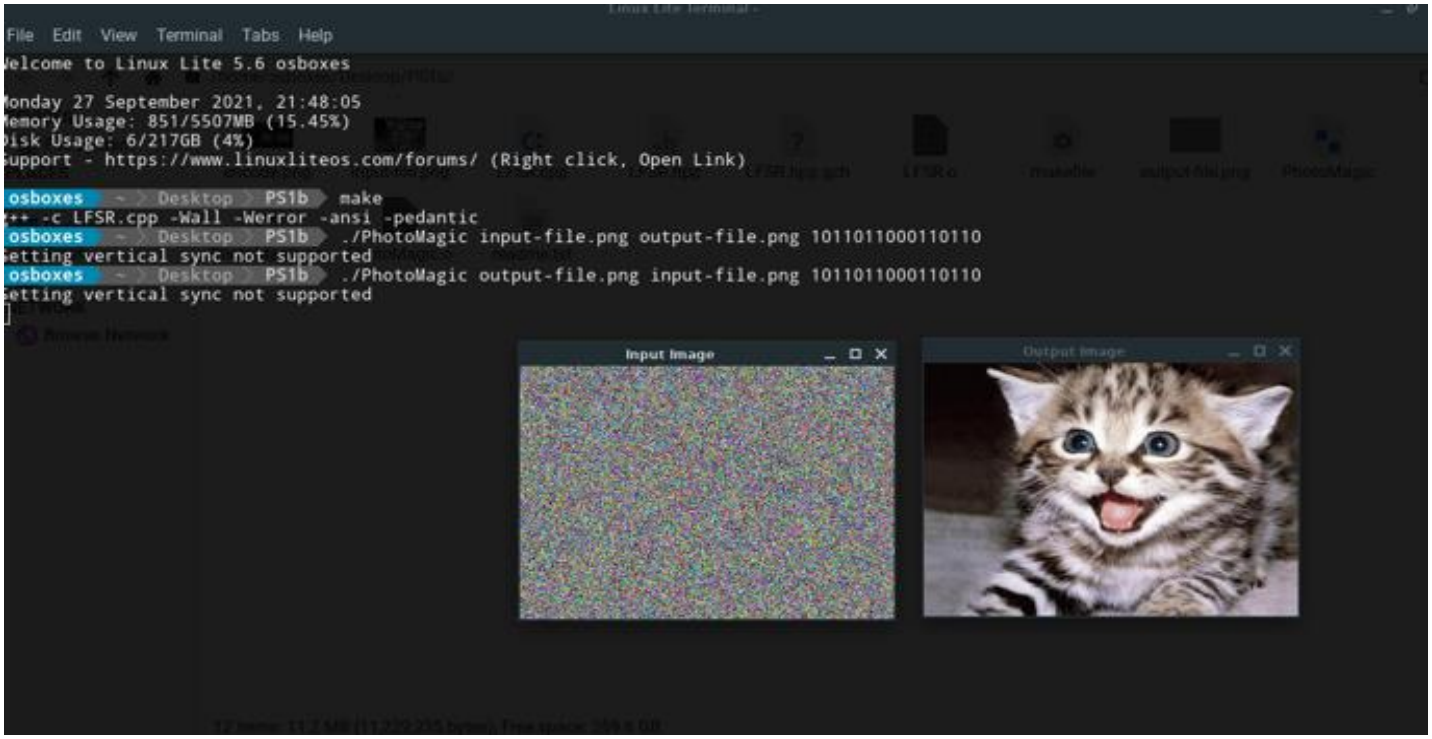
This assignment is a continuation of the previous one (PS1a). We were charged with constructing a program that reads a photo from the command line and then outputs the same image, but encoded, using the LFSR class we constructed in PS2a (encypted). The image was encoded using XOR utilizing the LFSR class, which left shifted all of the bits in the image. We also had to save the encypted image to a file and show it in an SFML window.

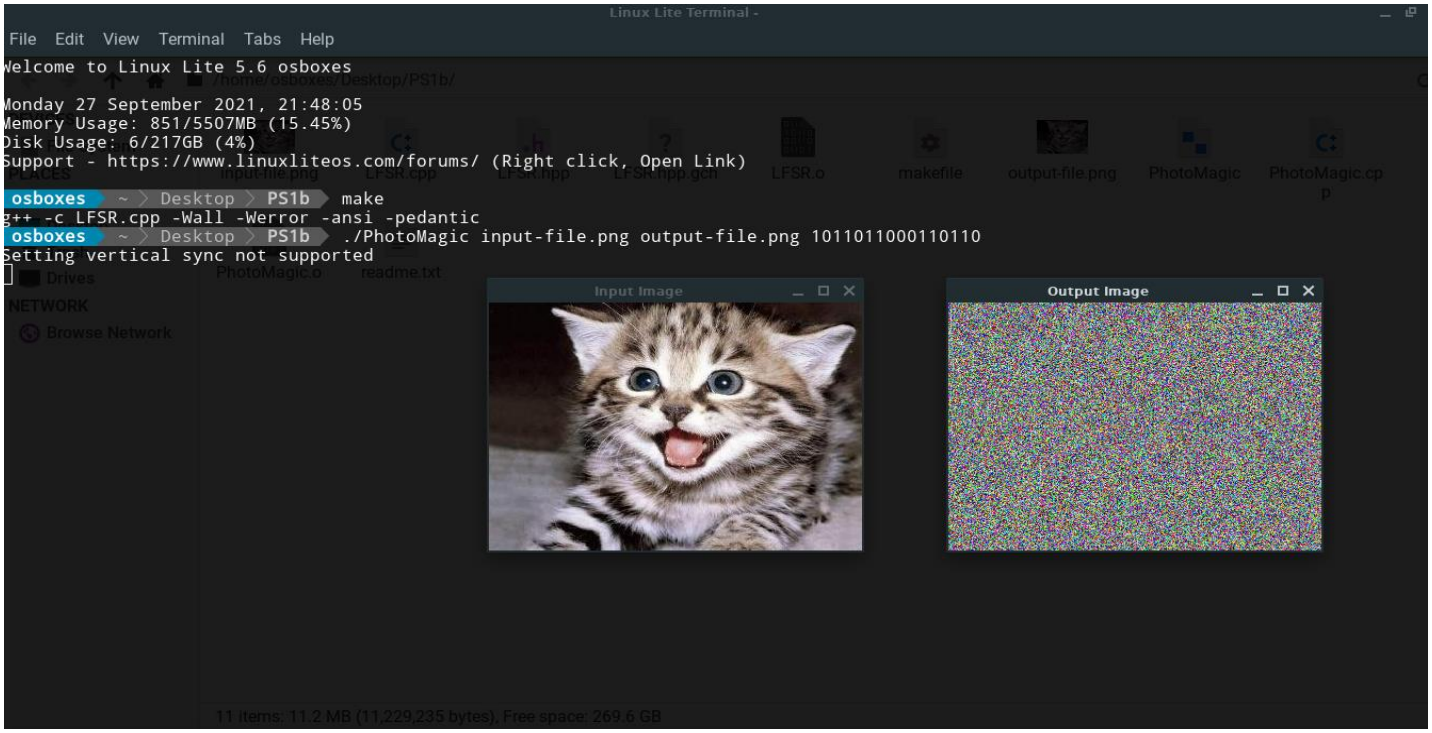
Key Concepts

The main thing that this assignment used was the LFSR class from the previous homework, PS1a. The LFSR class that we built uses a shift register to store bits and has two methods, step and generate, that we used to left shift all the bits. We also used several SFML objects, such as textures, images and sprites to read in the file, encode the file and output the final encoded image to both the screen and disk For this assignment the most central aspect in my opinion was being able to go into each pixel in the provided image and then changing it into a seemingly random color so that the image appeared hidden after the encryption process. The way this was handled in the code was through a function called transform, to which I passed as argument the input image and a previously constructed LFSR. The function had two nested loops so that it was possible to iterate through every single pixel in the image, then get the the amount of red, green and blue in the pixel. After storing these, the bit would then be XORed with a generated 8 bit int from the LFSR, creating a new color that was then assigned as the new color of the pixel. . The image class was the main way we encoded the image – we were able to get both the red, green and blue pixel using .getPixel().

What I Learned

this project was a lot of fun to work on because it made use of previously constructed materials. It was kind of cool to create something for one homework project and then reuse it for another. However, because it uses the LFSR class, I didn't gain any new stuff from this assignment. The encoding element of the assignment, on the other hand, was maybe the most beneficial. It was a lot of fun messing around with pixels and XORing them to produce an encoded image, then showing the encoded image. However, because that part of the assignment also employs XOR, I didn't learn anything new. I largely explored new uses for the LFSR class, which was a fun learning experience in and of itself.**Screenshots**


File Edit View Terminal Tabs Help
Welcome to Linux Lite 5.6 osboxes
Monday 27 September 2021, 21:48:05
Memory Usage: 851/5507MB (15.45%)
Disk Usage: 6/217GB (4%)
Support - https://www.linuxliteos.com/forums/ (Right click, Open Link)
osboxes ~ Desktop PS1b make
g++ -c LFSR.cpp -Wall -Werror -ansi -pedantic
osboxes ~ Desktop PS1b ./PhotoMagic input-file.png output-file.png 1011011000110110
Setting vertical sync not supported
osboxes ~ Desktop PS1b ./PhotoMagic output-file.png input-file.png 1011011000110110
Setting vertical sync not supported
Input Image
Output Image


Linux Lite Terminal -
File Edit View Terminal Tabs Help
Welcome to Linux Lite 5.6 osboxes
Monday 27 September 2021, 21:48:05
Memory Usage: 851/5507MB (15.45%)
Disk Usage: 6/217GB (4%)
Support - https://www.linuxliteos.com/forums/ (Right click, Open Link)
osboxes ~ Desktop PS1b make
g++ -c LFSR.cpp -Wall -Werror -ansi -pedantic
osboxes ~ Desktop PS1b ./PhotoMagic input-file.png output-file.png 1011011000110110
Setting vertical sync not supported
Input Image
Output Image
11 items: 11.2 MB (11,229,235 bytes), Free space: 269.6 GB

Source Code for PS1b Image Encoding

Makefile

```
# ps1b makefile
CC= g++
CFLAGS= -Wall -Werror -ansi -pedantic
SFMLFLAGS= -lsfml-graphics -lsfml-window -lsfml-system


all:    PhotoMagic


PhotoMagic:     PhotoMagic.o LFSR.o
        $(CC) PhotoMagic.o LFSR.o -o PhotoMagic $(SFMLFLAGS)


PhotoMagic.o:  PhotoMagic.cpp LFSR.hpp
        $(CC) -c PhotoMagic.cpp LFSR.hpp $(CFLAGS)

LFSR.o:         LFSR.cpp LFSR.hpp
        $(CC) -c LFSR.cpp $(CFLAGS)


clean:
        rm *.o
        rm PhotoMagic
```

Photomagic.cpp (main)

```
/*NAME OF THE STUDENT: GOGULAMUDI SRIKANTH REDDY
  STUDENT ID         : 01988167
  NAME OF THE COURSE : COMPUTING IV
  NAME OF THE PROFESSOR: DR.YELENA RYKALOVA
  SUBJECT CODE         : COMP.2040
    PS1b  ASSINGMENT
*/
#include <iostream>
#include <string>
#include <sstream>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "LFSR.hpp"

int main(int argc, char* argv[])
{
  if(argc != 4)
  {
   std::cout << "Usage: $ ./PhotoMagic [input file] [output file] [seed] \n";
    return -1;
  }

  std::string input_filename(argv[1]);
  std::string output_filename(argv[2]);
  std::string seed(argv[3]);

 int tap = 8;

  LFSR randomizer(seed, tap);
  sf::Image input_image;
  if (!input_image.loadFromFile(input_filename))
    return -1;

  sf::Image output_image;
  if (!output_image.loadFromFile(output_filename))
    return -1;

  // p is a pixel
  sf::Color p;
    sf::Color p1;

  // Setup the two windows
  sf::Vector2u size = input_image.getSize();
  for(int x= 0; x < (signed)size.x; x++){
    for(int y = 0; y < (signed)size.y; y++)
    {
      p = input_image.getPixel(x, y);
      p.r = (255-p.r)^ randomizer.generate(tap);
      p.g = (255-p.g)^ randomizer.generate(tap);
```

```
      p.b = (255-p.b)^ randomizer.generate(tap);
      output_image.setPixel(x, y, p);
    }}

  sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "Input Image");
  sf::RenderWindow window2(sf::VideoMode(size.x, size.y), "Output Image");


  sf::Texture input_texture, output_texture;
  input_texture.loadFromImage(input_image);
  output_texture.loadFromImage(output_image);


  sf::Sprite input_sprite, output_sprite;
  input_sprite.setTexture(input_texture);
  output_sprite.setTexture(output_texture);

  while (window1.isOpen() && window2.isOpen()) {
  sf::Event event;
  while (window1.pollEvent(event)) {
  if (event.type == sf::Event::Closed)
  window1.close();
  }
  while (window2.pollEvent(event)) {
  if (event.type == sf::Event::Closed)
  window2.close();
  }
  window1.clear();
  window1.draw(input_sprite);
  window1.display();
  window2.clear();
  window2.draw(output_sprite);
  window2.display();
  }

  if (!output_image.saveToFile(output_filename))
      exit(0);
  return 0;
}
```

LFSR.hpp

```
#ifndef LFSR_HPP
#define LFSR_HPP

#include <iostream>

class LFSR {
public:
  LFSR(std::string seed, int t);     // Constructor
  int step();                        // simulates one step
  int generate(int k);               // simulates k steps

  // Overloaded << operator
  friend std::ostream& operator<< (std::ostream &out, LFSR &cLFSR);

private:
  std::string bits;                  // holds the LFSR
  int tap;
};

#endif
```

LFSR.cpp

```
#include <iostream>
#include <string>
#include <sstream>
#include "LFSR.hpp"

LFSR::LFSR(std::string seed, int t)
{
  bits = seed;
  tap = t;
}

int LFSR::generate(int k)
{
  int g = 0;
  for(int i = 0; i < k; i++)
  {
    g = (g * 2) + step();
  }
  return g;
}


int LFSR::step()
{
  int tap_pos = bits.length() - tap - 1;
  int result = bits[0] ^ bits[tap_pos];
  std::string::size_type i;
  std::ostringstream ostring;
  for(i = 0; (unsigned)i < bits.length() - 1; i++)
    ostring << bits[i + 1];
  ostring << result;
  bits = ostring.str();
  return result;
}


std::ostream& operator<< (std::ostream &out, LFSR &cLFSR)
{
  out << cLFSR.bits;
  return out;
}
```

PS2a: N-Body Simulation: static universe

The Assignment

We used Princeton's N-Body Simulation problem for this project. Its goal is to create a realistic simulation of the universe on a 2D plane using Newton's equations of gravity. We had to use Newton's laws of physics to replicate the movement of heavenly bodies in a 2D plane. The motion of the celestial bodies would be animated in the simulation developed over time, with each frame indicating a movement over a set period. After the celestial body data is supplied, a.txt file detailing the end-state of the N-Body system generated will be output.The total simulation time and the time step were retrieved from the command line, and a static universe was presented on the screen. PS3b was used to implement the finished, moving universe. This section of the assignment mostly focused on reading a file from standard I/O and using the data from that file to populate sprites (which displayed the numerous planets) in the correct location in an SFML window.

Key Concepts

We employed a few essential C++ / Linux concepts in our project. The first was reading a file into standard I/O using the command line operator. A central data structure for my implementation of the universe was the vector. I chose the vector as the structure to contain my particles instead of an array because it was easier to work with the objects on a vector. I did not need to specify the size of the vector or the index of the last particle in order to add it. Since the memory in vector is dynamically allocated all I had to do was continue adding particles into the universe until they were all in there. Another important aspect of this assignment is that I had to use inheritance, one of the most important concepts of Object Oriented programming. By using inheritance I was able to take advantage of the drawable class in the SFML library to make my own objects drawable. I only used cin to read the contents of the file inside the main program; someone could type all of the planet's data in manually if they wanted to.To read in data easily, we also overloaded the >> operator – this way, we were able to just type:

```
cin >> c_body;
```

While not required by the assignment, I also overloaded the << operator to provide an easy way to test the program – just one line will output all the data inside the bodys object to standard I/O, like so:

```
cout << c_body;
```

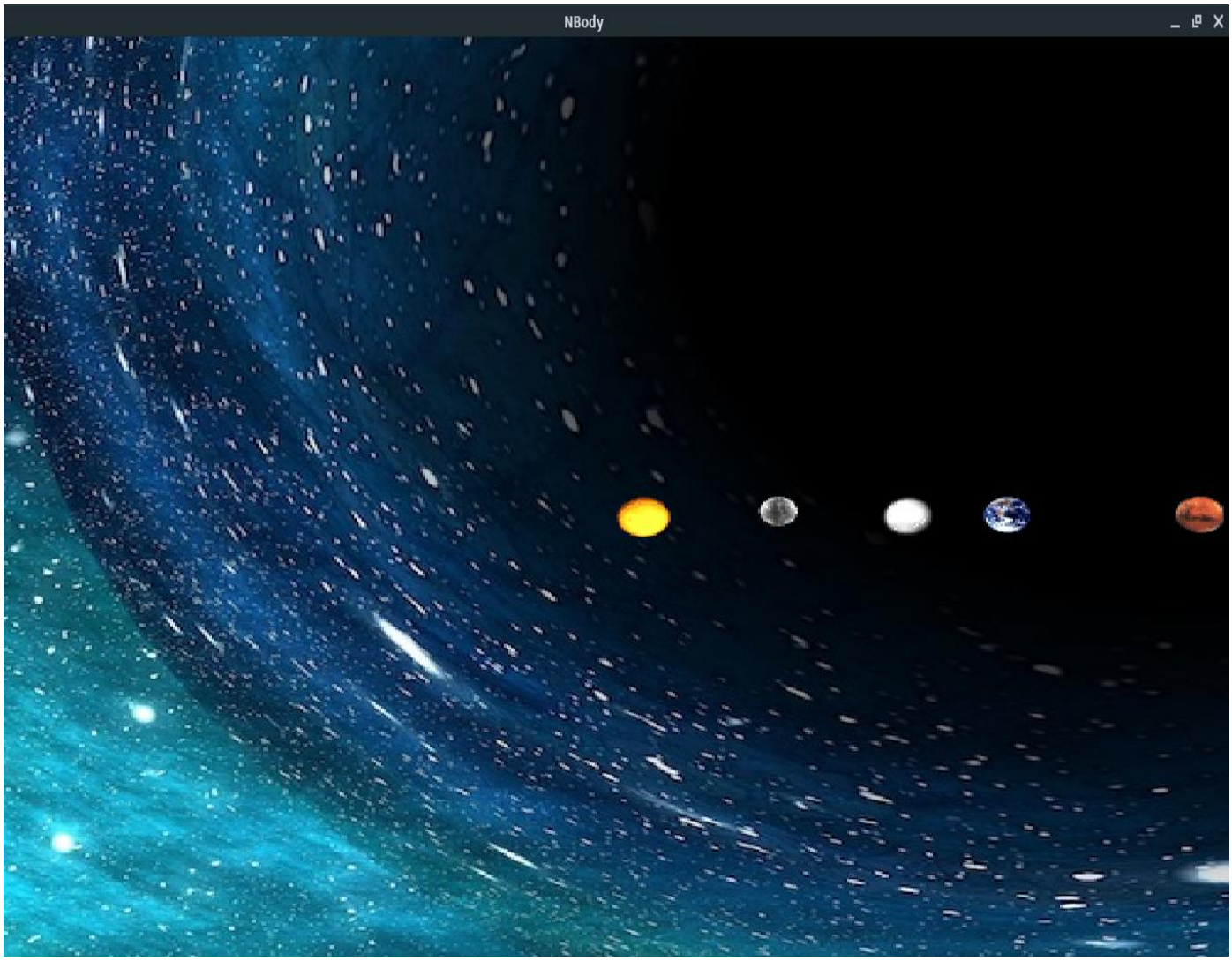
One other thing I had to do was create a method to convert X / Y positions of the planets to SFML coordinates. I was able to do this by realizing that the SFML window system sets (0, 0) as the top left corner. To convert the coordinates to the SFML system, I added half the height or side of the given window to the coordinate.

What I Learned

Another thing we had to do was implement the draw method in our bodys class – something I hadn't done in a long time, so it was fun to experiment with the draw function to get it to work. I didn't learn anything from overloading the >> operator because I'd done it a few times before and was already familiar with the process. I had even overloaded the operator for debugging purposes, so I was fairly comfortable with it. Similarly, we've worked with SFML's textures, pictures, and sprite objects previously, so displaying the planets was simple. Perhaps the most important thing I learnt was how to convert plantary units to SFML units - this took some time to perfect, as the planets would sometimes not show up, and other times would show up in the wrong place.

Screenshots

The Universe



Source Code for PS2a

Makefile

```
# Makefile for ps3a
# Flags to save on typing all this out
CC= g++
CFLAGS= -Wall -Werror -ansi -pedantic
SFMLFLAGS= -lsfml-graphics -lsfml-window -lsfml-system

# Make ps2b
all:    NBody

# Universe executable
NBody:  main.o Universe.o
        $(CC) main.o Universe.o -o NBody $(SFMLFLAGS)

# object files
main.o: main.cpp Universe.hpp
        $(CC) -c main.cpp Universe.hpp $(CFLAGS)

Universe.o:    Universe.cpp Universe.hpp
        $(CC) -c Universe.cpp Universe.hpp $(CFLAGS)

# Cleanup
clean:
        rm *.o
        rm *.gch
        rm NBody
```

main.cpp

```
/*NAME OF THE STUDENT    : GOGULAMUDI SRIKANTH REDDY
  NAME OF THE COURSE    : COMPUTING IV
  SUBJECT CODE          : COMP.2040
  NAME OF THE PROFESSOR : YELENA RYKOLOVA
  ASSIGNMENT            ; PS2a
*/



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

int main(int argc, char* argv[])
{
  string num_planets;
  string radius;
  cin >> num_planets;
  cin >> radius;
  int number_planets = atoi(num_planets.c_str());
  float universe_radius = atof(radius.c_str());

  cout << "Num of planets: " << number_planets << endl;
  cout << "Radius: " << universe_radius << std::endl << endl;
  vector<Universe> Universe_vector;
  for(int i = 0; i < number_planets; i++)
  {

    Universe* tmp = new Universe();

    cin >> *tmp;

    tmp->set_radius(universe_radius);
    tmp->set_position();

    Universe_vector.push_back(*tmp);

    cout << *tmp;
  }

  sf::RenderWindow window(sf::VideoMode(window_side, window_height), "NBody");

  window.setFramerateLimit(1);

  sf::Image bg_image;

  if (!bg_image.loadFromFile("stars.jpg"))
  {
    return -1;
  }
```

```
51
52   sf::Texture bg_texture;
53   bg_texture.loadFromImage(bg_image);
54
55   sf::Sprite bg_sprite;
56   bg_sprite.setTexture(bg_texture);
57
58   bg_sprite.setPosition(sf::Vector2f(-900, -900));
59
60   while (window.isOpen())
61   {
62
63     sf::Event event;
64     while(window.pollEvent(event))
65       if (event.type == sf::Event::Closed)
66         window.close();
67
68     window.clear();
69     window.draw(bg_sprite);
70
71     vector<Universe>::iterator it;
72
73     for(it = Universe_vector.begin(); it != Universe_vector.end(); it++)
74       window.draw(*it);
75
76     window.display();
77   }
78
79   return 0;
80 }
```

Universe.hpp

```
/*NAME OF THE STUDENT    : GOGULAMUDI SRIKANTH REDDY
  NAME OF THE COURSE    : COMPUTING IV
  SUBJECT CODE          : COMP.2040
  NAME OF THE PROFESSOR : YELENA RYKOLOVA
  ASSIGNMENT            ; PS2a
*/

#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

const int window_height = 500;
const int window_side = 500;

class Universe: public sf::Drawable
{
public:


  Universe();
  Universe(double pos_x, double pos_y, double vel_x, double vel_y,
       double obj_mass, double radius, std::string file_name);

  void set_radius(float radius);
  void set_position();

  friend std::istream& operator>> (std::istream &input, Universe
&CelestialB0dy);

  friend std::ostream& operator<< (std::ostream &output, Universe
&CelestialB0dy);

private:

  void virtual draw(sf::RenderTarget& target, sf::RenderStates states) const;

  double _pos_x, _pos_y;
  double _vel_x, _vel_y;
  double _mass;
  double _radius;
  std::string _filename;

  sf::Image _image;
  sf::Sprite _sprite;
  sf::Texture _texture;
};
```

Universe.cpp

```
/*NAME OF THE STUDENT    : GOGULAMUDI SRIKANTH REDDY
  NAME OF THE COURSE    : COMPUTING IV
  SUBJECT CODE          : COMP.2040
  NAME OF THE PROFESSOR : YELENA RYKOLOVA
  ASSIGNMENT            ; PS2a
*/

#include "Universe.hpp"
Universe::Universe()
{

  return;
}

Universe::Universe(double pos_x, double pos_y, double vel_x, double vel_y,
           double obj_mass, double radius, std::string file_name)
{

  _pos_x = pos_x;
  _pos_y = pos_y;
  _vel_x = vel_x;
  _vel_y = vel_y;
  _mass = obj_mass;
  _filename = file_name;

  if (!_image.loadFromFile(file_name))
    return;


  _texture.loadFromImage(_image);


  _sprite.setTexture(_texture);


  _sprite.setPosition(sf::Vector2f(_pos_x, _pos_y));
}



void Universe::set_radius(float radius)
{
  _radius = radius;
  return;
}


// Sets the planets position
void Universe::set_position()
{

```

```
  _pos_x = ( (_pos_x / _radius) * (window_side / 2) ) + (window_side / 2);
  _pos_y = ( (_pos_y / _radius) * (window_height / 2) ) + (window_height / 2);


  _sprite.setPosition(sf::Vector2f(_pos_x, _pos_y));
}



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

  target.draw(_sprite);
}



std::istream& operator>> (std::istream &input, Universe &CelestialB0dy)
{
  input >> CelestialB0dy._pos_x >> CelestialB0dy._pos_y;
  input >> CelestialB0dy._vel_y >> CelestialB0dy._vel_y;
  input >> CelestialB0dy._mass  >> CelestialB0dy._filename;

  if (!CelestialB0dy._image.loadFromFile(CelestialB0dy._filename))
    return input;

  CelestialB0dy._texture.loadFromImage(CelestialB0dy._image);
  CelestialB0dy._sprite.setTexture(CelestialB0dy._texture);
  CelestialB0dy._sprite.setPosition(sf::Vector2f(CelestialB0dy._pos_x,
CelestialB0dy._pos_y));

  return input;
}

std::ostream& operator<< (std::ostream &output, Universe &CelestialB0dy)
{

  output << "Filename: " << CelestialB0dy._filename << std::endl;
  output << "Pos (x): " << CelestialB0dy._pos_x << std::endl;
  output << "Pos (y): " << CelestialB0dy._pos_y << std::endl;
  output << "Vel (x): " << CelestialB0dy._vel_x << std::endl;
  output << "Vel (y): " << CelestialB0dy._vel_y << std::endl;
  output << "Mass: " << CelestialB0dy._mass << std::endl << std::endl;

  return output;
}
```

PS2b: N-Body Simulation:
Using Newton's laws of physics, animate the universe

The Assignment

We had to use Newton's laws of physics to replicate the movement of heavenly bodies in a 2D plane. The motion of the celestial bodies would be animated in the simulation developed over time, with each frame indicating a movement over a set period. After the celestial body data is supplied, a.txt file detailing the end-state of the N-Body system generated will be output.

Key Concepts

The main concepts for this assignment deal with Physics. They include:

- Newton’s law of universal gravitation
- The principle of superposition
- Newton’s second law of motion

We implemented these concepts in PS2b by using a few formulas, such as:

F = (G * M1 * M2) / R^2
R = square root (R2)
R2 = (Δx)^2 + (Δy)^2
Δx = x2 - x1
Δy = y2 - y1

Using these formulas, we were able to simulate the movement of the planets throughout the universe. This was a tricky part of the assignment, as getting the formulas right is the key to making the universe work correctly.

What I Learned

In this assignment, I mostly learned a little physics and how to use different equations in a program. It was difficult to get the equations accurate since, as I discovered when programming this project, one incorrect or slightly off equation might send all of the planets into pandemonium. You'll get a great simulation of the cosmos once you've implemented them appropriately. I learned how to play music using SFML's audio library in addition to building the physics aspect. Even though I performed this as part of an extra credit assignment, it was still really cool to hear the theme song from 2001: A Space Odyssey playing while the planets rotated around the Sun.

Screenshots

Source Code for PS2b

Makefile

```
CC = g++
CFLAGS = -c -g -Og -Wall -Werror -ansi -pedantic
LIBS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio

all: main.o Body.o
        $(CC) main.o Body.o -o NBody $(LIBS)

main.o: main.cpp
        $(CC) $(CFLAGS) -o $@ $< -std=c++14

Body.o: Body.cpp Body.hpp
        $(CC) $(CFLAGS) -o $@ $< -std=c++14

clean:
        rm *.o
```

main.cpp

```
/*NAME OF THE STUDENT    : GOGULAMUDI SRIKANTH REDDY
  NAME OF THE COURSE    : COMPUTING IV
  SUBJECT CODE          : COMP.2040
  NAME OF THE PROFESSOR : YELENA RYKOLOVA
  ASSIGNMENT            ; PS2b
*/

#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>

#include <iostream>
#include <fstream>

#include "Body.hpp"

#include <string>
#include <vector>
#include <cmath>

using namespace std;
using namespace sf;

int main(int argc, char *argv[])
{
        const double G = 6.67e-11;
        const double TOTAL_TIME = stoi(argv[1]);
        const double stepTime = stoi(argv[2]);
        cout << G << TOTAL_TIME << stepTime << endl;

        int nBody;
        float radius;
        float xPos;
        float yPos;
        float xVel;
        float yVel;
        float mass;
        string fileName;

        //taking input
        cin >> nBody;
        cin >> radius;

        vector<unique_ptr<Body>> vectorbody;
        for (int i = 0; i < nBody; ++i)
        {
                cin >> xPos >> yPos >> xVel >> yVel >> mass >> fileName;
                unique_ptr<Body> tempBody = make_unique<Body>(radius,
sf::Vector2u(640, 480), xPos, yPos, xVel, yVel, mass, fileName);
                vectorbody.push_back(move(tempBody));
```

```
        }

//Adding Music File
         sf::Music music;
         if(!music.openFromFile("1.ogg"))
          return 1;
   music.play();

        int elapsedTime = 0;

        //RenderWindow
        RenderWindow window(VideoMode(640, 480), "ps2b");
        while (window.isOpen() && elapsedTime < TOTAL_TIME)
        {
                for (Event event; window.pollEvent(event);)
                {
                        if(event.type == Event::Closed)
                                window.close();
                        if(sf::Keyboard::isKeyPressed(sf::Keyboard::P))
                          music.stop();
                        if(sf::Keyboard::isKeyPressed(sf::Keyboard::R))
                          music.play();
                }

                window.setFramerateLimit(60);
                window.clear(sf::Color::Black);

//calculating forces
                for (int i = 0; i < nBody; ++i)
                {
                        double xF = 0;
                        double yF = 0;

                        for (int j = 0; j < nBody; ++j)
                        {
                                if (i != j)
                                {
                                        double dX = vectorbody.at(j)->getXpos() -
vectorbody.at(i)->getXpos();
                                        double dY = vectorbody.at(j)->getYpos() -
vectorbody.at(i)->getYpos();
                                        double dist = sqrt((dX * dX) + (dY *
dY));
                                        double netF = (G * vectorbody.at(i)-
>getMass() * vectorbody.at(j)->getMass()) / (dist * dist);
                                        xF += netF * (dX / dist);
                                        yF += netF * (dY / dist);
                                }
                        }


                        double xAccel = xF / vectorbody.at(i)->getMass();
```

```
                        double yAccel = yF / vectorbody.at(i)->getMass();
                        double tempXvel = vectorbody.at(i)->getXvel() +
(stepTime * xAccel);
                        double tempYvel = vectorbody.at(i)->getYvel() -
(stepTime * yAccel);
                        vectorbody.at(i)->setXvel(tempXvel);
                        vectorbody.at(i)->setYvel(tempYvel);
                }

                for (int i = 0; i < nBody; ++i)
                {
                        vectorbody.at(i)->step(stepTime);
                        window.draw(vectorbody.at(i)->getPlanetSprite());
                }

                window.display();
                elapsedTime++;

                cout << "\n************* STEP NUMBER " << elapsedTime << "
***************************" << endl;
                cout << nBody << endl;
                cout << radius << endl;
                for (int i = 0; i < nBody; ++i)
                {
                        vectorbody.at(i)->print();
                }
        }
        cout << "\nElapsed Time: " << elapsedTime << endl;
        cout << "Total Time: " << TOTAL_TIME << endl;
}
```

body.hpp

```
/*NAME OF THE STUDENT    : GOGULAMUDI SRIKANTH REDDY
  NAME OF THE COURSE    : COMPUTING IV
  SUBJECT CODE          : COMP.2040
  NAME OF THE PROFESSOR : YELENA RYKOLOVA
  ASSIGNMENT            ; PS2b
*/
#include <string>
#ifndef BODY_H
#define BODY_H
#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Audio.hpp>

class Body : public sf::Drawable
{
public:
        Body();
        Body(float rad, sf::Vector2u winSize, float xp, float yp, float xv, float
yv, float m, std::string s);

        void setXpos(float xp);
        void setYpos(float yp);
        void setXvel(float xv);
        void setYvel(float yv);
        void setMass(float m);
        void setRadius(float r);
        void setWinSize(sf::Vector2u ws);
        void setFileName(std::string s);

        float getXpos();
        float getYpos();
        float getXvel();
        float getYvel();
        float getMass();
        float getRadius();
        sf::Vector2u getWinSize();
        sf::Sprite getPlanetSprite();
        std::string getFileName();

        void step(const double stepT);
        void print();

private:
        float xPos;
        float yPos;
        float xVel;
        float yVel;
        float mass;
        float radius;
```

```
51        sf::Vector2u winSize;
52        sf::Texture planetTexture;
53        sf::Sprite planetSprite;
54        std::string fileName;
55        sf::Music music;
56        virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
57 const;
58
59 };

   #endif
```

body.cpp

```
/*NAME OF THE STUDENT    : GOGULAMUDI SRIKANTH REDDY
  NAME OF THE COURSE    : COMPUTING IV
  SUBJECT CODE          : COMP.2040
  NAME OF THE PROFESSOR : YELENA RYKOLOVA
  ASSIGNMENT            ; PS2b
*/
#include "Body.hpp"
#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <math.h>
#include <iostream>
#include <SFML/Audio.hpp>

using namespace std;

Body::Body()
{
        xPos = 0, yPos = 0, xVel = 0, yVel = 0, mass = 0;
        fileName = "";
}

Body::Body(float rad, sf::Vector2u winSize, float xp, float yp, float xv, float
yv, float m, std::string s)
{
        xPos = xp, yPos = yp;
        xVel = xv, yVel = yv;
        mass = m;
        radius = rad;
        fileName = s;

        //image
        sf::Image image;
        image.loadFromFile(fileName);
        planetTexture.loadFromImage(image);
        planetSprite.setTexture(planetTexture);

        //sprite
        planetSprite.setOrigin(planetTexture.getSize().x / 2,
planetTexture.getSize().y / 2);

        //origin
        float xWinRadius = winSize.x / 2;
        float yWinRadius = winSize.y / 2;
        float xScalar = (xWinRadius) / rad;
        float yScalar = (yWinRadius) / rad;
        float xOrigin = xPos * xScalar + xWinRadius;
        float yOrigin = yPos * yScalar + yWinRadius;
        planetSprite.setPosition(xOrigin, yOrigin);
}

```

```
void Body::setXpos(float xp) { xPos = xp; }

void Body::setYpos(float yp) { yPos = yp; }

void Body::setXvel(float xv) { xVel = xv; }

void Body::setYvel(float yv) { yVel = yv; }

void Body::setMass(float m) { mass = m; }

void Body::setRadius(float r) { radius = r; }

void Body::setWinSize(sf::Vector2u ws) { winSize = ws; }

void Body::setFileName(std::string s) { fileName = s; }

float Body::getXpos() { return xPos; }

float Body::getYpos() { return yPos; }

float Body::getXvel() { return xVel; }

float Body::getYvel() { return yVel; }

float Body::getMass() { return mass; }

float Body::getRadius() { return radius; }

sf::Vector2u Body::getWinSize() { return winSize; }

sf::Sprite Body::getPlanetSprite() { return planetSprite; }

std::string Body::getFileName() { return std::string(); }

void Body::step(const double stepT)
{
        xPos += stepT * xVel;
        yPos -= stepT * yVel;

        double xWinRadius = 640 / 2;
        double yWinRadius = 480 / 2;
        double xScalar = (xWinRadius) / radius;
        double yScalar = (yWinRadius) / radius;

        double scaledXpos = (xPos * xScalar) + xWinRadius;
        double scaledYpos = (yPos * yScalar) + yWinRadius;

        planetSprite.setPosition(scaledXpos, scaledYpos);
}

void Body::print()
{
```

```
        cout << xPos << " " << yPos << " " << xVel << " " << yVel << " " << mass << " \t" << fileName << endl;
}

void Body::draw(sf::RenderTarget &target, sf::RenderStates states) const
{
        target.draw(Body::planetSprite, states);
}
```

PS3: Recursive Graphic

The Assignment

We were given the challenge of implementing the Sierpinski triangle assignment from Princeton for our second assignment. The major goal of the assignment was to use recursion to construct a complex-looking triangle that only required a few lines of code. The main program was to accept an integer, N, and use it to adjust the recursion's depth. After that, our program would draw one triangle at depth 1, four triangles at depth 2, and so on, recursively drawing triangles within triangles. The second half of the project was to make our own recursively image, which was distinct from the Sierpinski triangle design but still used recursion to make a stunning image. I was able to make the Sierpinski triangle design as well as my own circle within a circle (color changing) design.

Key Concepts

Key to this program was the idea of recursion. I had to figure out a way to implement the triangle drawing within each other recursively, and do so without using a ton of resources. I found the best way to do this was using pointers to other triangles – the first main triangle has three Sierpinski pointers, and those also have pointers to three more triangles and so on until the max depth is reached. I recreated a little diagram in my README file that illustrates this quite well:

```
                                    Blank triangle
        /    |        \
      ptr   ptr       ptr
    /  | \ / | \     / | \
   p   p p p p p p    p  p  p
  /|\/|\/|\|\|\|\|\  /|\/|\/|\
Etc – all the way to max depth.
```

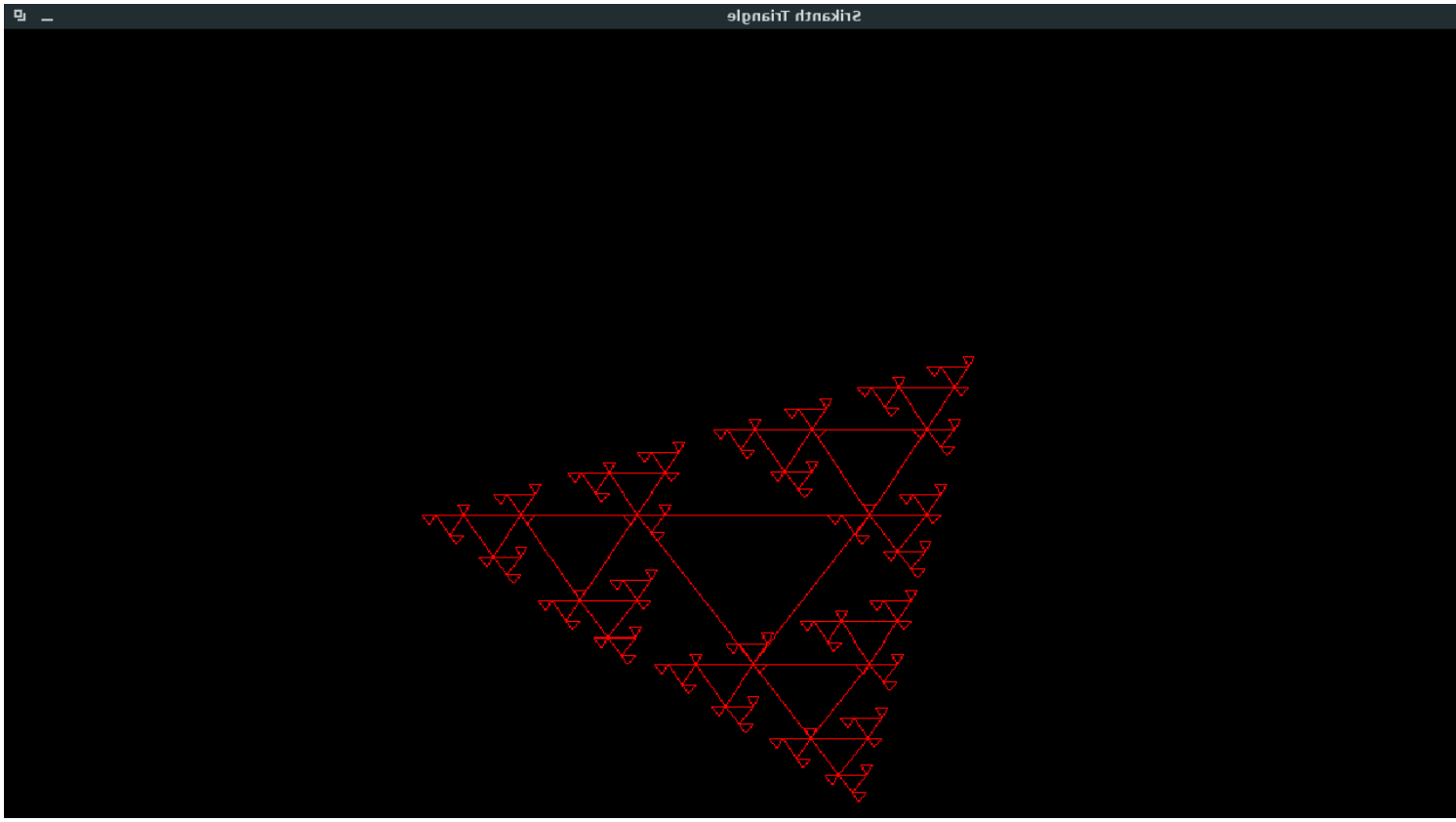
By using pointers, I was able to recursively draw the triangles out in the Sierpinski::Draw method – I

This ends up calling draw on each triangle, which in turn called draw on all of their triangles, until the recursion depth is reached.

What I Learned

I learned a little bit about recursion – I had used it before in previous classes, but never quite like this, where I was able to use pointers to other objects in order to recursively go back and draw them out. Implementing my own design after learning how to draw an object recursively was also much easier, as I already knew how I could set up the code, I just had to think about what kind of image I wanted to draw and how to make it look interesting. In the end, it was a fun assignment that taught me about recursion in a different way than I was used to.

Screenshots



Source Code for PS3 Recursive Graphic

Makefile

```
# Makefile for ps3
# Flags
compiler= g++
cppFlags= -Wall -Werror -pedantic
SFMLFlags= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio


all: TFractal

# body executable
TFractal:      TFractal.o Triangle.o
        $(compiler) TFractal.o Triangle.o -o TFractal $(SFMLFlags)

# object files
TFractal.o:    TFractal.cpp Triangle.h Triangle.h
        $(compiler) -c TFractal.cpp Triangle.h $(cppFlags)

# Cleanup
clean:
        rm *.o
        rm TFractal
        rm *.gch
```

TFractal.cpp

```
/*Name of the Student : Gogulamudi Srikanth Reddy
  Student ID          : 01988167
  Name of the Professor : DR. Yelena Rykolova
  Subject               : Computing IV
  assignment            : PS3*/


#include "Triangle.h"
#include <vector>
#include <iostream>
#include <random>

#define WIN_H 1000
#define WIN_W 1000
#define PYT 0.867

void fTree(Triangle traingle, int recursion, int length, sf::RenderWindow
*window) {
    if(recursion <= 0)
        return;

    traingle.bottom = new Triangle(sf::Vector2f(traingle.a3.x - length,
traingle.a3.y), traingle.a3, sf::Vector2f(traingle.a3.x - length/2,
traingle.a3.y + (length * PYT)));
    traingle.left = new Triangle(sf::Vector2f(traingle.a1.x - length/2,
traingle.a1.y - length), sf::Vector2f(traingle.a1.x + length/2, traingle.a1.y -
length), traingle.a1);
    traingle.right = new Triangle(traingle.a2, sf::Vector2f(traingle.a2.x +
length, traingle.a2.y), sf::Vector2f(traingle.a2.x + length/2, traingle.a2.y +
length));

    window -> draw(*traingle.bottom);
    window -> draw(*traingle.left);
    window -> draw(*traingle.right);

    recursion--;

    fTree(*traingle.left, recursion, length/2, window);
    fTree(*traingle.right, recursion, length/2, window);
    fTree(*traingle.bottom, recursion, length/2, window);
}

int main(int argc, char* argv[]) {

    sf::RenderWindow window(sf::VideoMode(WIN_W, WIN_H), "Srikanth Triangle");
    window.setFramerateLimit(60);

    int length = atoi(argv[1]);
    int recursion = atoi(argv[2]);

    float x = WIN_W/2 - length/2;
```

```
    float y = WIN_H/2 - length/2;


    Triangle initialTriangle(sf::Vector2f(x, y), sf::Vector2f(x + length, y),
sf::Vector2f(x + length/2, y + (length * PYT)));

    window.draw(initialTriangle);
    fTree(initialTriangle, recursion, length/2, &window);

    std::cout << "Done rendering." << std::endl;
    sf::Vector2u windowSize = window.getSize();
    sf::Texture texture;
    texture.create(windowSize.x, windowSize.y);
    texture.update(window);
    sf::Image capture = texture.copyToImage();
    sf::Texture fraxture;
    fraxture.loadFromImage(capture);
    sf::Sprite fracrite;
    fracrite.setTexture(fraxture);

    int frames = 0;
    bool rotate = false;

    fracrite.setColor(sf::Color::Red);
    while (window.isOpen()){
        sf::Event event;
        while (window.pollEvent(event)){
            if (event.type == sf::Event::Closed)
                window.close();
        }
        window.clear();
        frames=frames+1;
        if (frames % 50 == 0)
            rotate = !rotate;
        window.draw(fracrite);
        window.display();
    }
}
```

Triangle.hpp

```
/*Name of the Student : Gogulamudi Srikanth Reddy
  Student ID          : 01988167
  Name of the Professor : DR. Yelena Rykolova
  Subject               : Computing IV
  assignment            : PS3*/


#include "Triangle.h"
#include <vector>
#include <iostream>
#include <random>

#define WIN_H 1000
#define WIN_W 1000
#define PYT 0.867

void fTree(Triangle traingle, int recursion, int length, sf::RenderWindow
*window) {
    if(recursion <= 0)
        return;

    traingle.bottom = new Triangle(sf::Vector2f(traingle.a3.x - length,
traingle.a3.y), traingle.a3, sf::Vector2f(traingle.a3.x - length/2,
traingle.a3.y + (length * PYT)));
    traingle.left = new Triangle(sf::Vector2f(traingle.a1.x - length/2,
traingle.a1.y - length), sf::Vector2f(traingle.a1.x + length/2, traingle.a1.y -
length), traingle.a1);
    traingle.right = new Triangle(traingle.a2, sf::Vector2f(traingle.a2.x +
length, traingle.a2.y), sf::Vector2f(traingle.a2.x + length/2, traingle.a2.y +
length));

    window -> draw(*traingle.bottom);
    window -> draw(*traingle.left);
    window -> draw(*traingle.right);

    recursion--;

    fTree(*traingle.left, recursion, length/2, window);
    fTree(*traingle.right, recursion, length/2, window);
    fTree(*traingle.bottom, recursion, length/2, window);
}

int main(int argc, char* argv[]) {

    sf::RenderWindow window(sf::VideoMode(WIN_W, WIN_H), "Srikanth Triangle");
    window.setFramerateLimit(60);

    int length = atoi(argv[1]);
    int recursion = atoi(argv[2]);

```

```
    float x = WIN_W/2 - length/2;
    float y = WIN_H/2 - length/2;


    Triangle initialTriangle(sf::Vector2f(x, y), sf::Vector2f(x + length, y),
sf::Vector2f(x + length/2, y + (length * PYT)));

    window.draw(initialTriangle);
    fTree(initialTriangle, recursion, length/2, &window);

    std::cout << "Done rendering." << std::endl;
    sf::Vector2u windowSize = window.getSize();
    sf::Texture texture;
    texture.create(windowSize.x, windowSize.y);
    texture.update(window);
    sf::Image capture = texture.copyToImage();
    sf::Texture fraxture;
    fraxture.loadFromImage(capture);
    sf::Sprite fracrite;
    fracrite.setTexture(fraxture);

    int frames = 0;
    bool rotate = false;

    fracrite.setColor(sf::Color::Red);
    while (window.isOpen()){
        sf::Event event;
        while (window.pollEvent(event)){
            if (event.type == sf::Event::Closed)
                window.close();
        }
        window.clear();
        frames=frames+1;
        if (frames % 50 == 0)
            rotate = !rotate;
        window.draw(fracrite);
        window.display();
    }
}
```

Triangle.cpp

```
/*Name of the Student   : Gogulamudi Srikanth Reddy
  Student ID            : 01988167
  Name of the Professor : DR. Yelena Rykolova
  Subject               : Computing IV
  assignment            : PS3*/


#include "Triangle.h"
Triangle::Triangle(sf::Vector2f a, sf::Vector2f b, sf::Vector2f c) {
    a1 = a;
    a2 = b;
    a3 = c;
    bottom = NULL;
    left = NULL;
    right = NULL;
}

void Triangle::draw(sf::RenderTarget &target, sf::RenderStates states) const {
    sf::Vertex edge[] = {
        sf::Vertex(a1),
        sf::Vertex(a2)
    };

    target.draw(edge, 2, sf::Lines);

    edge[0] = sf::Vertex(a2);
    edge[1] = sf::Vertex(a3);
    target.draw(edge, 2, sf::Lines);

    edge[0] = sf::Vertex(a1);
    edge[1] = sf::Vertex(a3);
    target.draw(edge, 2, sf::Lines);
}
```

mainart.cpp

```
#include <SFML/Graphics.hpp>
#include <cstdlib>
#include <math.h>
#include <iostream>
int main(int argc, char* argv[])
{
int recursionDepth = std::atoi(argv[1]);
int size = std::atoi(argv[2]);
if(argc != 3)
    {
    std::cout << "\nEXITING PROGRAM\n\tEnter: <file> <recursion depth
    integer> <screen size greater than 100>\n" << std::endl;
    return EXIT_FAILURE;
    }
    if(size < 100)
    {
    std::cout << "\nEXITING PROGRAM\n\tRecommended Minimum Size:
    100\n\tPlease run again with a bigger screen size.\n" << std::endl;
    return EXIT_FAILURE;
    }
    sf::RenderWindow window(sf::VideoMode(size, size), "Original Fractal
    Art");
    int radius = size/2;
    sf::Vector2f position(size/2, size/2);
    float outline = radius/40;
    while(window.isOpen())
    {
    sf::Event event;
    Original circle(recursionDepth, radius, position, outline);
    while(window.pollEvent(event))
    {
    if(event.type == sf::Event::Closed)
    window.close();
    }
    window.clear(sf::Color(204,230,255,255));
    window.draw(circle);
    window.display();
    }
    return 0;
    }
```

PS4a: Synthesizing a Plucked String Sound

The Assignment

I created a RingBuffer for this project, replete with unit tests and exceptions The goal of this project was to create a program that could replicate the sound of a guitar string. We've created a Ring Buffer (called CircularBuffer) to accomplish this, which is a fixed-size queue that can be filled with random data. Iterating the process deletes the value at the top of the queue and enqueues a new value equal to the average of the next item in line and the deleted value, multiplied by a decay factor. The decay factor is used to mimic the "decay" of a plucked string. The circular buffer class is fully, perfectly, and entirely operational, thanks to Boost's unit testing framework. The simulation has 37 keys, and it may be played with the keys on a computer keyboard.. The main purpose of the project was to implement the RingBuffer, which operates by wrapping around like a circle array to store values — in our instance, 16 bit integers. The RingBuffer was additionally checked for problems using Boost unit tests, and the RingBuffer was designed to throw specific exceptions for specific errors. If you try to build a RingBuffer with a capacity of 0 or less, you'll get a std::invalid argument, and if you try to enqueue a full RingBuffer or dequeue or peek at an empty RingBuffer, you'll get a std::runtime error.

Key Concepts

The main concepts for this assignment focused on implementing the RingBuffer using exceptions, which is something we had not really discussed in class before this assignment. Exceptions were utilized often in this project, especially in the context of the Boost unit test framework. Every internal function that didn't only fetch data or return a boolean value had to check the function call's legitimacy based on the parameters passed in or the state of the object at the time it was called. When an error was raised, a short text was displayed, describing where it occurred and why it occurred, making the code more compact and easier to change. Lambda expressions were also utilized to effectively empty the available memory when the window was closed, particularly to construct the samples for each of the 37 notes on the keyboard. Try / Catch blocks were used to test for invalid actions. Another key idea was using Google’s cpplint python script to check our code for consistency. This was introduced in the optional PSX assignment, but it was the first time we began to use cpplint in class. Boost was also used to check the RingBuffer, and ensure that the buffer threw the correct exceptions when it was supposed to, and no exceptions were thrown for valid actions.

What I Learned

Exceptions and cpplint were what I really took away from this assignment. Having only used exceptions a few times in Computing III, I was somewhat rusty at them. It did not take too long to figure them out though, and having used them in this class, I was able to apply exceptions to another class – Android Development, which uses Java and Java, like C++, uses exceptions a fair amount. Cpplint was also an interesting tool to use. Mainly in the class

absolutely hate it, but I found it to be useful at times. It forced me to change some bad coding habbits, such as using really long lines (80+ characters in length). At the same time though, it forced me to program Google's way, which is annoying if you've already got a programming style that you like (which I did somewhat have). For example, I used to like putting brackets on a separate line, but after using cpplint I've become used to putting them on the same line as the if / loop statement. This was hard to get used to, but now that I am used to it I've pretty much stuck with that style for other programming assignments that I do outside of class.

Screenshots

```
PS4a_SrikanthReddy_Gogulamudi — -zsh — 192×60
srikanthreddygogulamudi@Srikanths-MacBook-Air PS4a_SrikanthReddy_Gogulamudi % make
g++ -c test.cpp CircularBuffer.hpp -g -Wall -Werror -std=c++0x -pedantic
g++ -c CircularBuffer.cpp CircularBuffer.hpp -g -Wall -Werror -std=c++0x -pedantic
g++ test.o CircularBuffer.o -o CircularBuffer -lboost_unit_test_framework
g++ -c main.cpp CircularBuffer.hpp -g -Wall -Werror -std=c++0x -pedantic
g++ main.o CircularBuffer.o -o main.out
srikanthreddygogulamudi@Srikanths-MacBook-Air PS4a_SrikanthReddy_Gogulamudi % python cpplint.py test.cpp
Done processing test.cpp
Total errors found: 0
srikanthreddygogulamudi@Srikanths-MacBook-Air PS4a_SrikanthReddy_Gogulamudi % ./main
zsh: no such file or directory: ./main
srikanthreddygogulamudi@Srikanths-MacBook-Air PS4a_SrikanthReddy_Gogulamudi % ./main.out
Test main.
Peek: 1
Deq 1: 1
Deq 2: 2
        First           : 2
        Last            : 3
        Capacity        : 100
        Size            : 1
        Vector size     : 100
        Vector capacity : 100
        Buffer          : 3  0     0     3     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
        0     0     0     0     0     0     0
        First           : 1
        Last            : 1
        Capacity        : 3
        Size            : 3
        Vector size     : 3
        Vector capacity : 3
        Buffer          : 2  3     4     4     2     3
srikanthreddygogulamudi@Srikanths-MacBook-Air PS4a_SrikanthReddy_Gogulamudi % ./CircularBuffer
Running 6 test cases...

*** No errors detected
srikanthreddygogulamudi@Srikanths-MacBook-Air PS4a_SrikanthReddy_Gogulamudi %
```

Source Code for PS4a

Makefile

```
CC= g++
FLAGS= -g -Wall -Werror -std=c++0x -pedantic
Boost= -lboost_unit_test_framework
all:  PS4a main.out
PS4a: test.o CircularBuffer.o
	$(CC) test.o CircularBuffer.o -o PS4a $(Boost)
main.out: main.o CircularBuffer.o
	$(CC) main.o CircularBuffer.o -o main.out
CircularBuffer.o: CircularBuffer.cpp CircularBuffer.h
```

```
10:             $(CC) -c CircularBuffer.cpp CircularBuffer.h $(CFLAGS)
11:       test.o: test.cpp CircularBuffer.h
12:             $(CC) -c test.cpp CircularBuffer.h $(CFLAGS)
13:       main.o:     main.cpp CircularBuffer.h
14:             $(CC) -c main.cpp CircularBuffer.h $(CFLAGS)
15:       clean:
16:             rm *.o
17:             rm *.gch
18:             rm PS4a
19:             rm *.out
```

main.cpp

```
/*
 * Copyright vennela
 * All rights reserved.

 *
 */
#include "RingBuffer.hpp"

int main() {
  std::cout << "Test main.\n";

  RingBuffer test(100);
  test.enqueue(1);
  test.enqueue(2);
  test.enqueue(3);
  std::cout << "Peek: " << test.peek() << "\n";

  std::cout << "Deq 1: " << test.dequeue() << "\n";
  std::cout << "Deq 2: " << test.dequeue() << "\n";


  test.output();

  // Test looping back around
  RingBuffer test2(3);

  test2.enqueue(1);
  test2.enqueue(2);
  test2.enqueue(3);

  test2.dequeue();
  test2.dequeue();
  test2.dequeue();

  test2.enqueue(1);
  test2.enqueue(2);
  test2.enqueue(3);
  test2.dequeue();
  test2.enqueue(4);

  test2.output();

  return 0;
}

```

RingBuffer.hpp

```
/*
 * Copyright vennela
 * All rights reserved.

 *
 */
#include <stdint.h>
#include <iostream>
#include <string>
#include <sstream>
#include <exception>
#include <stdexcept>
#include <vector>

class RingBuffer {
 public:
  // API functions

  // Empty ring buffer, with given max capacity.
  explicit RingBuffer(int capacity);
  int size();                 // return # of items in the buffer.
  bool isEmpty();             // is size == 0?
  bool isFull();              // is size == capacity?
  void enqueue(int16_t x);    // add item x to the end.
  int16_t dequeue();          // delete and return item from the front
  int16_t peek();             // return (don't delete) item from the front.

  // Other functions
  void output();

 private:
  std::vector<int16_t> _buffer;
  int _first;
  int _last;
  int _capacity;
  int _size;
};

```

RingBuffer.cpp

```
/*
 * Copyright vennela
 * All rights reserved.

 *
 */
#include "RingBuffer.hpp"

// Create an empty ring buffer, with given max capacity.
RingBuffer::RingBuffer(int capacity) {
  if (capacity < 1) {
    throw
    std::invalid_argument("RB constructor: capacity must be greater than zero");
  }

  _last = 0;
  _first = 0;
  _size = 0;
  _capacity = capacity;
  _buffer.resize(capacity);

  return;
}


// Return # of items in the buffer.
int RingBuffer::size() {
  return _size;
}


// Is size == 0?
bool RingBuffer::isEmpty() {
  // Determine if the RingBuffer is empty.
  if (_size == 0) {
    return true;
  } else {
    return false;
  }
}


// Is size == capacity?
bool RingBuffer::isFull() {
  // Determine if size equals capacity.
  if (_size == _capacity) {
    return true;
  } else {
    return false;
  }
}


// Add item x to the end.
```

```
void RingBuffer::enqueue(int16_t x) {
  // See if the buffer is full
  if (isFull()) {
    throw
      std::runtime_error("enqueue: can't enqueue to a full ring");
  }

  // Check to see if we need to loop last back around to 0.
  if (_last >= _capacity) {
    _last = 0;
  }

  // If we don't throw any exceptions, then continue on!
  _buffer.at(_last) = x;

  // Increase counter variables.
  _last++;
  _size++;
}


// Delete and return item from the front
int16_t RingBuffer::dequeue() {
  if (isEmpty()) {
    throw
      std::runtime_error("dequeue: can't dequeue to an empty ring");
  }

  // Remove from the front.
  int16_t first = _buffer.at(_first);
  _buffer.at(_first) = 0;

  // Decrease counter variables.
  _first++;
  _size--;

  // Check to see if we need to loop first back around to 0.
  if (_first >= _capacity) {
    _first = 0;
  }

  return first;
}


// Return (don't delete) item from the front.
int16_t RingBuffer::peek() {
  // This is an easy function - return the first buffer position.
  if (isEmpty()) {
    throw
      std::runtime_error("peek: can't peek an empty ring");
  }

  return _buffer.at(_first);
}
```

```
// Dumps the variables to stdout
void RingBuffer::output() {
  std::cout << "   First: " << _first << "\n";
  std::cout << "    Last: " << _last << "\n";
  std::cout << "Capacity: " << _capacity << "\n";
  std::cout << "    Size: " << _size << "\n";
  std::cout << "Vector size: " << _buffer.size() << "\n";
  std::cout << "Vector capacity: " << _buffer.capacity() << "\n";
  std::cout << "Buffer (no blanks): \n";

  int x = 0;
  int y = _first;

  while (x < _size) {
    // Make the loop go back to 0 to continue printing.
    if (y >= _capacity) {
      y = 0;
    }

    std::cout << _buffer[y] << " ";
    y++;
    x++;
  }

  std::cout << "\nDump the entire buffer (including blanks): \n";

  for (int x = 0; x < _capacity; x++) {
    std::cout << _buffer[x] << " ";
  }

  std::cout << "\n\n";
}

```

test.cpp

```
/*
 * Copyright vennela
 * All rights reserved.

 *
 */
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

#include "RingBuffer.hpp"

// Tests various aspects of the constructor.
BOOST_AUTO_TEST_CASE(Constructor) {
  // Normal constructor - shouldn't fail.
  BOOST_REQUIRE_NO_THROW(RingBuffer(100));

  // These should fail.
  BOOST_REQUIRE_THROW(RingBuffer(0), std::exception);
  BOOST_REQUIRE_THROW(RingBuffer(0), std::invalid_argument);
  BOOST_REQUIRE_THROW(RingBuffer(-1), std::invalid_argument);
}


// Checks the size() method
BOOST_AUTO_TEST_CASE(Size) {
  RingBuffer test(1);

  // This should be size 0.
  BOOST_REQUIRE(test.size() == 0);

  test.enqueue(5);

  // This should be size 1.
  BOOST_REQUIRE(test.size() == 1);

  test.dequeue();
  BOOST_REQUIRE(test.size() == 0);
}


// Checks the isEmpty() method
BOOST_AUTO_TEST_CASE(isEmpty) {
  // This should be true
  RingBuffer test(5);
  BOOST_REQUIRE(test.isEmpty() == true);

  // This should be false
  RingBuffer test2(5);
  test2.enqueue(5);
  BOOST_REQUIRE(test2.isEmpty() == false);
}


```

```
55    // Checks the isFull() method
56    BOOST_AUTO_TEST_CASE(isFull) {
57      RingBuffer test(5);
58      BOOST_REQUIRE(test.isFull() == false);
59
60      RingBuffer test2(1);
61      test2.enqueue(5);
62      BOOST_REQUIRE(test2.isFull() == true);
63    }
64
65
66    // Test enqueue
67    BOOST_AUTO_TEST_CASE(Enqueue) {
68      // These test basic enqueuing
69      RingBuffer test(5);
70
71      BOOST_REQUIRE_NO_THROW(test.enqueue(1));
72      BOOST_REQUIRE_NO_THROW(test.enqueue(2));
73      BOOST_REQUIRE_NO_THROW(test.enqueue(3));
74      BOOST_REQUIRE_NO_THROW(test.enqueue(4));
75      BOOST_REQUIRE_NO_THROW(test.enqueue(5));
76      BOOST_REQUIRE_THROW(test.enqueue(6), std::runtime_error);
77    }
78
79
80    // Test dequeue
81    BOOST_AUTO_TEST_CASE(Dequeue) {
82      RingBuffer test(5);
83
84      test.enqueue(0);
85      test.enqueue(1);
86      test.enqueue(2);
87
88      BOOST_REQUIRE(test.dequeue() == 0);
89      BOOST_REQUIRE(test.dequeue() == 1);
90      BOOST_REQUIRE(test.dequeue() == 2);
91      BOOST_REQUIRE_THROW(test.dequeue(), std::runtime_error);
92    }
```

PS4b: StringSound implementation and SFML audio output

The Assignment

The RingBuffer from PS4a was utilized in the second portion of PS4 to generate a Guitar model by implementing the Karplus-Strong algorithm to replicate the plucking of a guitar string. We were given the responsibility of implementing a few methods to replicate guitar playing, such as plucking, ticking, sampling, and so on. Finally, we developed the primary software, GuitarHero, respond to keyboard presses by generating various notes for each key.

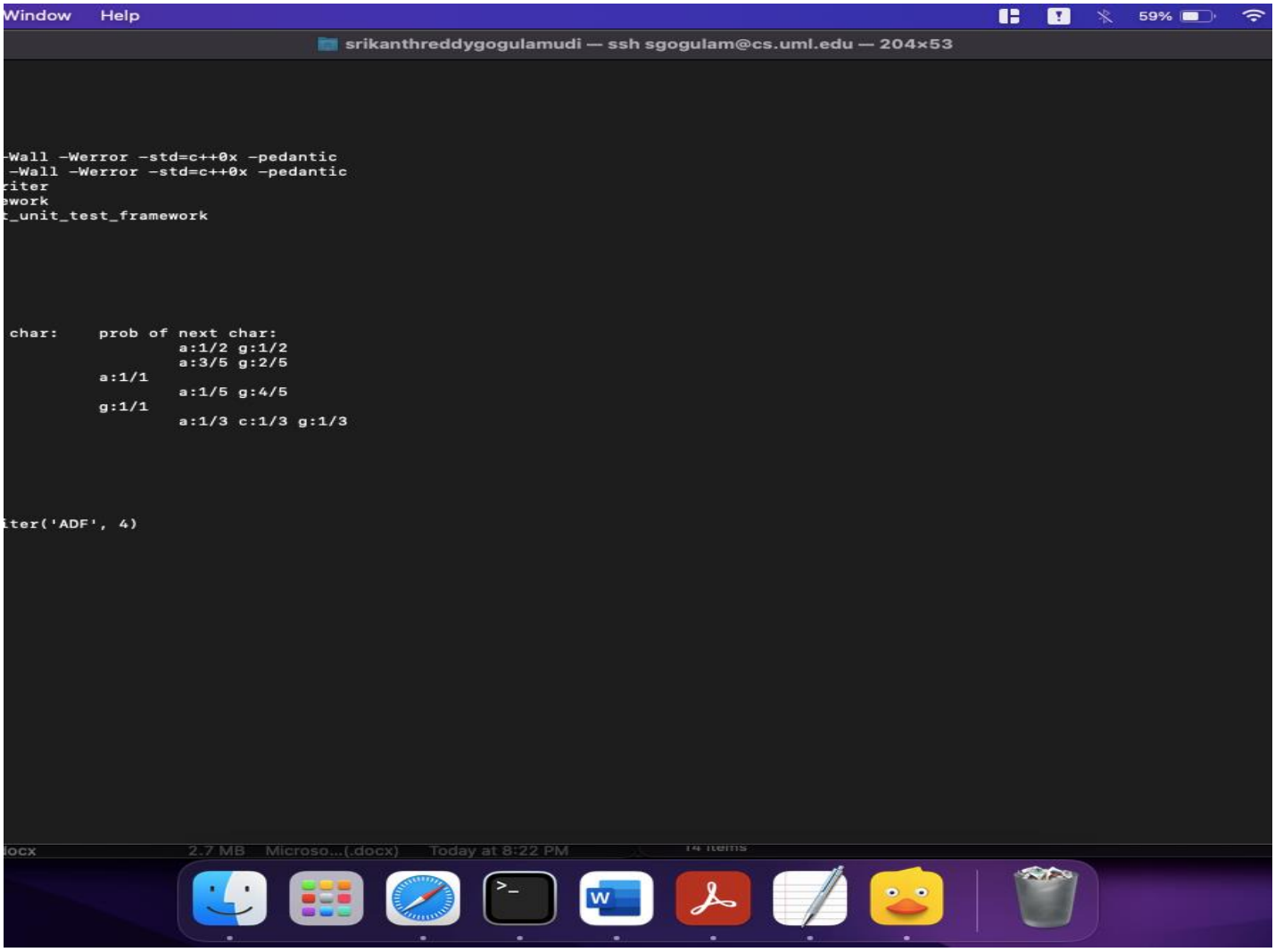
Key Concepts

The main algorithm was used in this assignment was the Karplus-Strong algorithm, which was used to simulate the plucking of a guitar. The Karplus-Strong algorithm works by modeling frequencies, and it takes the first two values, averages them and then multiplies the result by the energy decay factor, which in our case was .1996. This, along with the RingBuffer, allowed us to model sound (to some degree) and made it seem like a guitar string was being plucked.

What I Learned

It was fascinating to learn about the Karplus-Strong algorithm, and it gave me insight into how to model sound in a software. The Karplus-Strong update was similarly difficult to get right, and I had a lot of segfaults when I first created the program with pointers. The segfaults disappeared after I converted to a member initialization list, and the program was able to function somewhat again. I also had trouble getting the sound to play, but I'm not sure how I corrected it – I think messing around with the SFML keyboard settings helped. Speaking of which, SFML's Keyboard library is quite handy for manipulating a piano, guitar, or other instrument — I was thinking about how it could be used in simple 2D games, which, combined with the planet stuff from the PS2, could make a pretty nice space invaders-style game.

Screenshot


Window Help
59%
srikanthreddygogulamudi — ssh sgogulam@cs.uml.edu — 204×53
-Wall -Werror -std=c++0x -pedantic
-Wall -Werror -std=c++0x -pedantic
iter
work
_unit_test_framework
char: prob of next char:
a:1/2 g:1/2
a:3/5 g:2/5
a:1/1
a:1/5 g:4/5
g:1/1
a:1/3 c:1/3 g:1/3
iter('ADF', 4)
2.7 MB Microso...(.docx) Today at 8:22 PM

Source Code for PS4b

Makefile

```
CFLAGS= -g -Wall -Werror -std=c++0x -pedantic
SFLAGS= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio

all:    KSGuitarSim

KSGuitarSim:   KSGuitarSim.o StringSound.o CircularBuffer.o
        g++ KSGuitarSim.o StringSound.o CircularBuffer.o -o KSGuitarSim $(SFLAGS)

KSGuitarSim.o: KSGuitarSim.cpp StringSound.hpp
        g++ -c KSGuitarSim.cpp StringSound.hpp $(CFLAGS)

StringSound.o: StringSound.cpp StringSound.hpp
        g++ -c StringSound.cpp StringSound.hpp $(CFLAGS)

CircularBuffer.o: CircularBuffer.cpp CircularBuffer.hpp
        g++ -c CircularBuffer.cpp CircularBuffer.hpp $(CFLAGS)

clean:
        rm *.o
        rm *.gch
        rm KSGuitarSim
```

StringSound.cpp (main)

```
/*
 * Copyright 2020 Srikanth Reddy Gogulamudi
 * All rights reserved.
 * MIT Licensed - see http://opensource.org/licenses/MIT for details.
 *
 */

 /*Name of the Student : Srikanth Reddy Gogulamudi
   Student ID : 01988167
   Assignment : PS4b
   Name of the professor : Dr. Yelena Rykolova
   Student Email : srikanthreddy_gogulamudi@student.uml.edu
   */
#include "StringSound.hpp"
#include <vector>

StringSound::StringSound(double frequency) : _buff(ceil(SAMPLING_RATE /
frequency)) {
  _N = ceil(SAMPLING_RATE / frequency);
  for (int i = 0; i < _N; i++)
    _buff.enqueue((int16_t)0);
  _tic = 0;
}

StringSound::StringSound(std::vector<sf::Int16> init) : _buff(init.size()) {
  _N = init.size();
  try {
    if (_N == 0) {
    std::runtime_error("StringSound: can't dequeue to an empty ring");
  }
  }
  catch(std::exception e) {
    std::cout << "StringSound:Initial Size Error\n";
  }
  std::vector<sf::Int16>::iterator it;
  for (it = init.begin(); it < init.end(); it++)
    _buff.enqueue((int16_t)*it);
  _tic = 0;
}

void StringSound::pluck() {
  for (int i = 0; i < _N; i++)
    _buff.dequeue();
  for (int i = 0; i < _N; i++)
    _buff.enqueue((sf::Int16)(rand() & 0xffff));
  return;
}

void StringSound::tic() {
  int16_t first = _buff.dequeue();
```

```
  int16_t second = _buff.peek();
  int16_t avg = (first + second) / 2;
  int16_t karplus = avg * ENERGY_DECAY_FACTOR;

  _buff.enqueue((sf::Int16)karplus);
  _tic++;
  return;
}

sf::Int16 StringSound::sample() {
  sf::Int16 sample = (sf::Int16)_buff.peek();
  return sample;
}

int StringSound::time() {
  return _tic;
}
```

StringSound.hpp

```
/*
 * Copyright 2020 Srikanth Reddy Gogulamudi
 * All rights reserved.
 * MIT Licensed - see http://opensource.org/licenses/MIT for details.
 *
 */


  /*Name of the Student : Srikanth Reddy Gogulamudi
    Student ID : 01988167
    Assignment : PS4b
    Name of the professor : Dr. Yelena Rykolova
    Student Email : srikanthreddy_gogulamudi@student.uml.edu
    */
#ifndef STRINGSOUND_HPP
#define STRINGSOUND_HPP

#include <SFML/Audio.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <cmath>
#include <iostream>
#include <string>
#include <vector>
#include "CircularBuffer.hpp"

const int SAMPLING_RATE = 44100;
const double ENERGY_DECAY_FACTOR = 0.996;

class StringSound {
 public:
  explicit StringSound(double frequency);
  explicit StringSound(std::vector<sf::Int16> init);
  void pluck();
  void tic();
  sf::Int16 sample();
  int time();
 private:
  CircularBuffer _buff;
  int _N;
  int _tic;
};
#endif
```

CircularBuffer.cpp

```
/*
 * Copyright 2020 Srikanth Reddy Gogulamudi
 * All rights reserved.
 * MIT Licensed - see http://opensource.org/licenses/MIT for details.
 *
 */


  /*Name of the Student : Srikanth Reddy Gogulamudi
    Student ID : 01988167
    Assignment : PS4b
    Name of the professor : Dr. Yelena Rykolova
    Student Email : srikanthreddy_gogulamudi@student.uml.edu
    */
#include "CircularBuffer.hpp"

CircularBuffer::CircularBuffer(int capacity) {
  if (capacity < 1) {
    throw
    std::invalid_argument("RB constructor: capacity must be greater than
zero");
  }

  _last = 0;
  _first = 0;
  _size = 0;
  _capacity = capacity;
  _buffer.resize(capacity);

  return;
}


int CircularBuffer::size() {
  int _size1 = _size;
  auto lambdasize = [_size1]()->int{
    return _size1;
  };
  return lambdasize();
}


bool CircularBuffer::isEmpty() {
  int _size1 = _size;
  auto lambdasize = [_size1]()->bool{
    if (_size1 == 0)
      return true;
    else
      return false;
    };
```

```
  return lambdasize();
}



bool CircularBuffer::isFull() {
  int _size1 = _size;
  int _capacity1 = _capacity;
  auto lambdasize = [=]()->bool{
    if (_size1 == _capacity1)
      return true;
    else
      return false;
    };
    return lambdasize();
}


void CircularBuffer::enqueue(int16_t x) {
  if (isFull()) {
    throw
      std::runtime_error("enqueue: can't enqueue to a full ring");
  }


  if (_last >= _capacity) {
    _last = 0;
  }

  _buffer.at(_last) = x;

  _last++;
  _size++;
}



int16_t CircularBuffer::dequeue() {
  if (isEmpty()) {
    throw
      std::runtime_error("dequeue: can't dequeue to an empty ring");
  }


  int16_t first = _buffer.at(_first);
  _buffer.at(_first) = 0;


  _first++;
  _size--;


```

```
  if (_first >= _capacity) {
    _first = 0;
  }

  return first;
}



int16_t CircularBuffer::peek() {
  if (isEmpty()) {
    throw
      std::runtime_error("peek: can't peek an empty ring");
  }

  return _buffer.at(_first);
}



void CircularBuffer::output() {
  std::cout << "   First:             " << _first << "\n";
  std::cout << "    Last:             " << _last << "\n";
  std::cout << "Capacity:             " << _capacity << "\n";
  std::cout << "    Size:             " << _size << "\n";
  std::cout << "Vector size:          " << _buffer.size() << "\n";
  std::cout << "Vector capacity:      " << _buffer.capacity() << "\n";
  std::cout << "Buffer: \n            ";

  int x = 0;
  int y = _first;

  while (x < _size) {
    if (y >= _capacity) {
      y = 0;
    }

    std::cout << _buffer[y] << " ";
    y++;
    x++;
  }

  std::cout << "\nDump the entire buffer (including blanks): \n";

  for (int x = 0; x < _capacity; x++) {
    std::cout << _buffer[x] << " ";
  }

  std::cout << "\n\n";
}
```

CircularBuffer.hpp

```
/*
 * Copyright 2020 Srikanth Reddy Gogulamudi
 * All rights reserved.
 * MIT Licensed - see http://opensource.org/licenses/MIT for details.
 *
 */


  /*Name of the Student : Srikanth Reddy Gogulamudi
    Student ID : 01988167
    Assignment : PS4b
    Name of the professor : Dr. Yelena Rykolova
    Student Email : srikanthreddy_gogulamudi@student.uml.edu
    */
#ifndef CircularBuffer_HPP
#define CircularBuffer_HPP

#include <stdint.h>
#include <iostream>
#include <string>
#include <sstream>
#include <exception>
#include <stdexcept>
#include <vector>
#include <algorithm>

class CircularBuffer {
 public:
  explicit CircularBuffer(int capacity);
  int size();
  bool isEmpty();
  bool isFull();
  void enqueue(int16_t x);
  int16_t dequeue();
  int16_t peek();
  void output();

 private:
  std::vector<int16_t> _buffer;
  int _first;
  int _last;
  int _capacity;
  int _size;
};
#endif
```

RingBuffer.cpp

```
/*
 * Copyright vennela
 * All rights reserved.
 *
 *
 */
#include "RingBuffer.hpp"

// Create an empty ring buffer, with given max capacity.
RingBuffer::RingBuffer(int capacity) {
  if (capacity < 1) {
    throw
    std::invalid_argument("RB constructor: capacity must be greater than zero");
  }

  _last = 0;
  _first = 0;
  _size = 0;
  _capacity = capacity;
  _buffer.resize(capacity);

  return;
}


// Return # of items in the buffer.
int RingBuffer::size() {
  return _size;
}


// Is size == 0?
bool RingBuffer::isEmpty() {
  // Determine if the RingBuffer is empty.
  if (_size == 0) {
    return true;
  } else {
    return false;
  }
}


// Is size == capacity?
bool RingBuffer::isFull() {
  // Determine if size equals capacity.
  if (_size == _capacity) {
    return true;
  } else {
    return false;
  }
}


// Add item x to the end.
```

```
void RingBuffer::enqueue(int16_t x) {
  // See if the buffer is full
  if (isFull()) {
    throw
      std::runtime_error("enqueue: can't enqueue to a full ring");
  }

  // Check to see if we need to loop last back around to 0.
  if (_last >= _capacity) {
    _last = 0;
  }

  // If we don't throw any exceptions, then continue on!
  _buffer.at(_last) = x;

  // Increase counter variables.
  _last++;
  _size++;
}


// Delete and return item from the front
int16_t RingBuffer::dequeue() {
  if (isEmpty()) {
    throw
      std::runtime_error("dequeue: can't dequeue to an empty ring");
  }

  // Remove from the front.
  int16_t first = _buffer.at(_first);
  _buffer.at(_first) = 0;

  // Decrease counter variables.
  _first++;
  _size--;

  // Check to see if we need to loop first back around to 0.
  if (_first >= _capacity) {
    _first = 0;
  }

  return first;
}


// Return (don't delete) item from the front.
int16_t RingBuffer::peek() {
  // This is an easy function - return the first buffer position.
  if (isEmpty()) {
    throw
      std::runtime_error("peek: can't peek an empty ring");
  }

  return _buffer.at(_first);
}
```

```
// Dumps the variables to stdout
void RingBuffer::output() {
  std::cout << "   First: " << _first << "\n";
  std::cout << "    Last: " << _last << "\n";
  std::cout << "Capacity: " << _capacity << "\n";
  std::cout << "    Size: " << _size << "\n";
  std::cout << "Vector size: " << _buffer.size() << "\n";
  std::cout << "Vector capacity: " << _buffer.capacity() << "\n";
  std::cout << "Buffer (no blanks): \n";

  int x = 0;
  int y = _first;

  while (x < _size) {
    // Make the loop go back to 0 to continue printing.
    if (y >= _capacity) {
      y = 0;
    }

    std::cout << _buffer[y] << " ";
    y++;
    x++;
  }

  std::cout << "\nDump the entire buffer (including blanks): \n";

  for (int x = 0; x < _capacity; x++) {
    std::cout << _buffer[x] << " ";
  }

  std::cout << "\n\n";
}

```

PS5: DNA Sequence Alignment

The Assignment

For PS4, we implemented a program to find the optimal alignment of two strings. Princeton calls it the alignment of two DNA strings. A key idea for this program was also to use dynamic programming to make calculating the edit distance efficient.

Key Concepts

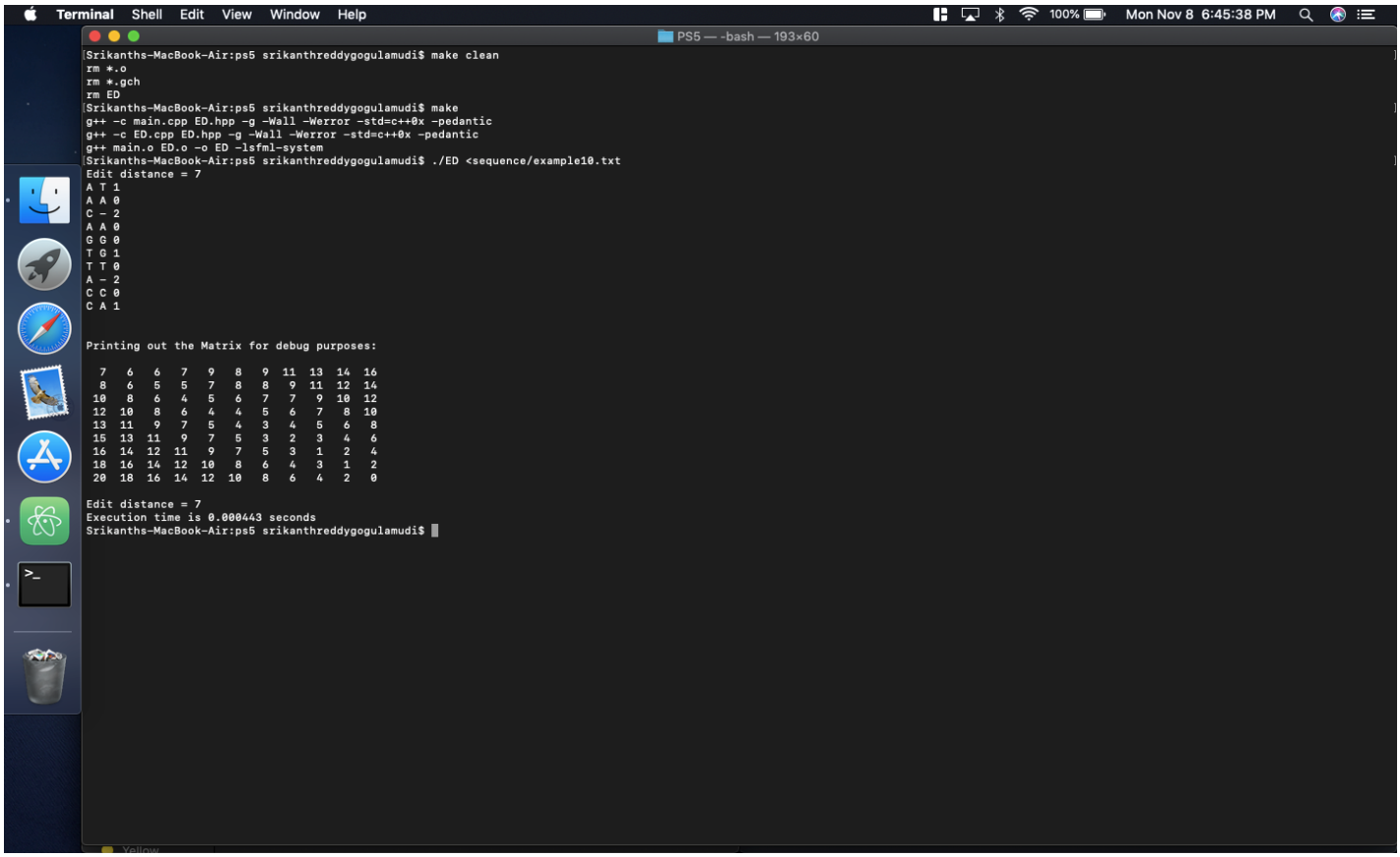
The main concept that was introduced for this program is known as the Needleman-Wunsch method, which is a way of using dynamic programming to calculate subproblems, and then use those subproblems to find the main solution. In the case of this program, we used an NxM matrix to do so. This works by first calculating the easy edit distances – and then using those solutions to find the next round of edit distances, until you've arrived at the solution in the [0][0] cell of the matrix.

We also were able to recover the path that our algorithm took by retracing our steps through the matrix. We did this by using a few rules: First, The optimal alignment matches $x[i]$ up with $y[j]$. In this case, we must have $opt[i][j] = opt[i+1][j+1]$ if $x[i]$ equals $y[j]$, or $opt[i][j] = opt[i+1][j+1] + 1$ otherwise.
Second, The optimal alignment matches $x[i]$ up with a gap. In this case, we must have $opt[i][j] = opt[i+1][j] + 2$.
Thirdly, The optimal alignment matches $y[j]$ up with a gap. In this case, we must have $opt[i][j] = opt[i][j+1] + 2$.
Using these three rules, we will able to go from the top left most cell of the matrix ([0][0]), where we originally found the final edit distance, and then trace our steps back to the bottom right most cell ([N][M]).

What I Learned

I learned a few things from this assignment. First, that vectors are pretty inefficient compared to c arrays – seeing the results of other classmates' implementations on iSENSE showed me that. I also got to play around with valgrind, which I've used in the past. I found a nice way of visualizing valgrind's results as well using massif visualizer. See the screenshot from it in the screenshots section. Also, using the Needleman-Wunsch method to calculate subproblems was pretty interesting – that was not something I had really considered before, so it has given me an insight into other methods of programming.

Screenshots



Source Code for PS5

Makefile

```
CFLAGS= -g -Wall -Werror -std=c++0x -pedantic
SFLAGS= -lsfml-system -lsfml-graphics

all:    EDistanceistance

EDistanceistance:      main.o EDistance.o
        g++ main.o EDistance.o -o EDistance $(SFLAGS)

main.o: main.cpp EDistance.hpp
        g++ -c main.cpp EDistance.hpp $(CFLAGS)

EDistance.o:   EDistance.cpp EDistance.hpp
        g++ -c EDistance.cpp EDistance.hpp $(CFLAGS)

clean:
        rm *.o
        rm *.gch
        rm EDistance
```

main.cpp

```
/*Name of the student   : SRIKANTH REDDY GOGULAMUDI
  Nmae of the course    : Computing IV
  Name of the professor : Dr. Yelena Rylolova
  Assignment            : PS5
  */

#include "EDistance.hpp"
using namespace std;
int main(int argc, const char* argv[])
{
  sf::Time t;
  sf::Clock clock;

  string input1, input2;

  cin >> input1 >> input2;

  ED ed_test(input1, input2);

  int dis = ed_test.Optdis();
  string alignment = ed_test.Alignment();

  cout << "Edit dis = " << dis << "\n";
  cout << alignment;

  ed_test.printval();
  t = clock.getElapsedTime();

  cout << "\nExecution time is " << t.asSeconds() << " seconds \n";
  auto a=[=]()->int
  {
    if(dis>=0)
    cout<<"\t";
    return dis;
  };
  cout<<"\nEdit dis="<<a()<<"\n";
}
```

EDistance.hpp

```
/*Name of the student   : SRIKANTH REDDY GOGULAMUDI
  Nmae of the course    : Computing IV
  Name of the professor : Dr. Yelena Rylolova
  Assignment            : PS5
  */
#ifndef ED_HPP
#define ED_HPP

#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdexcept>
#include <vector>
#include <SFML/System.hpp>
#include <algorithm>

using namespace std;

class ED
{
  public:
    ED();
    ED(string string_one, string string_two);
    int penalty(char a, char b);
    int min(int a, int b, int c);
    int Optdis();
    string Alignment();
    void printval();

  private:
    string _string_one, _string_two;
    vector< vector<int> > _matrix;
};

#endif
```

EDistance.cpp

```
/*Name of the student   : SRIKANTH REDDY GOGULAMUDI
  Nmae of the course    : Computing IV
  Name of the professor : Dr. Yelena Rylolova
  Assignment            : PS5
  */
#include "EDistance.hpp"
using namespace std;

ED::ED(string string_one, string string_two)
{
  _string_one = string_one;
  _string_two = string_two;
}


int ED::penalty(char a, char b)
{
  if(a == b)
    return 0;
  else if(a != b)
    return 1;

  return -1;
}



int ED::min(int a, int b, int c)
{
  if(a < b && a < c)
    return a;
  else if(b < a && b < c)
    return b;
  else if(c < a && c < b)
    return c;

  return a;
}

int ED::Optdis()
{

  int i, j;
  int N = _string_one.length();
  int M = _string_two.length();


  for(i = 0; i <= M; i++)
  {
    std::vector<int> tmp;
```

```
    _matrix.push_back(tmp);
    for(j = 0; j <= N; j++)
      _matrix.at(i).push_back(0);
  }


  for(i = 0; i <= M; i++)
    _matrix[i][N] = 2 * (M - i);


  for(j = 0; j <= N; j++)
    _matrix[M][j] = 2 * (N - j);


  for(i = M - 1; i >= 0; i--)
    for(j = N - 1; j >= 0; j--)
    {
      int opt1 = _matrix[i+1][j+1] + penalty(_string_one[j], _string_two[i]);
      int opt2 = _matrix[i+1][j] + 2;
      int opt3 = _matrix[i][j+1] + 2;
      _matrix[i][j] = min(opt1, opt2, opt3);
    }

  return _matrix[0][0];
}
void ED::printval()
{
  cout << "\n\nMatrix: \n\n";
  vector< std::vector<int> >::iterator a;
  vector<int>::iterator b;

  for(a = _matrix.begin(); a != _matrix.end(); a++)
    for(b = (*a).begin(); b != (*a).end(); b++)
      cout << right << setw(3) << *b << " ";
    cout << "\n";
}

string ED::Alignment()
{

  ostringstream return_string;

  int M = _string_two.length();
  int N = _string_one.length();
  int i = 0, j = 0;
  int pen, opt1, opt2, opt3;
  string ret_str;

  while(i < M || j < N)
  {
    try{
      pen =  penalty(_string_one[j], _string_two[i]);
```

```
      opt1 = _matrix.at(i+1).at(j+1) + pen;
    }
    catch(const out_of_range& error)
    {
      opt1 = -1;
    }
    try{
      opt2 = _matrix.at(i+1).at(j) + 2;
    }catch(const out_of_range& error)
    {
      opt2 = -1;
    }
    try{
      opt3 = _matrix.at(i).at(j+1) + 2;
    }catch(const out_of_range& error)
    {
      opt3 = -1;
    }
    if(_matrix[i][j] == opt1)
    {
      return_string << _string_one[j] << " " <<  _string_two[i] << " "  << pen
<< "\n";
      i++;
      j++;
    }
    else if(_matrix[i][j] == opt2)
    {
      return_string << "- " << _string_two[i] << " 2\n";
      i++;
    }
    else if(_matrix[i][j] == opt3)
    {
      return_string << _string_one[j] << " -" << " 2\n";
      j++;
    }
  }

  ret_str = return_string.str();
  return ret_str;
}
```

PS6: Random Writer

The Assignment

This assignment involved using regular expressions to parse files of various Kronos InTouch time clock logs to analyze them, verifying the device's boot up timing, and noting whether or not these startups were fully successful or not. The purpose of this is so information can be gathered on what was occurring with the device at the time of the bootup failures, in order to eventually solve the problems occurring with the InTouch device. In order to help do this. PS6 asks us to scan the complete log files given and create a text file report chronologically describing each time the device was restarted, noting if it failed or succeeded in completely doing so, and giving the elapsed time for the sequence if the bootup was successful.

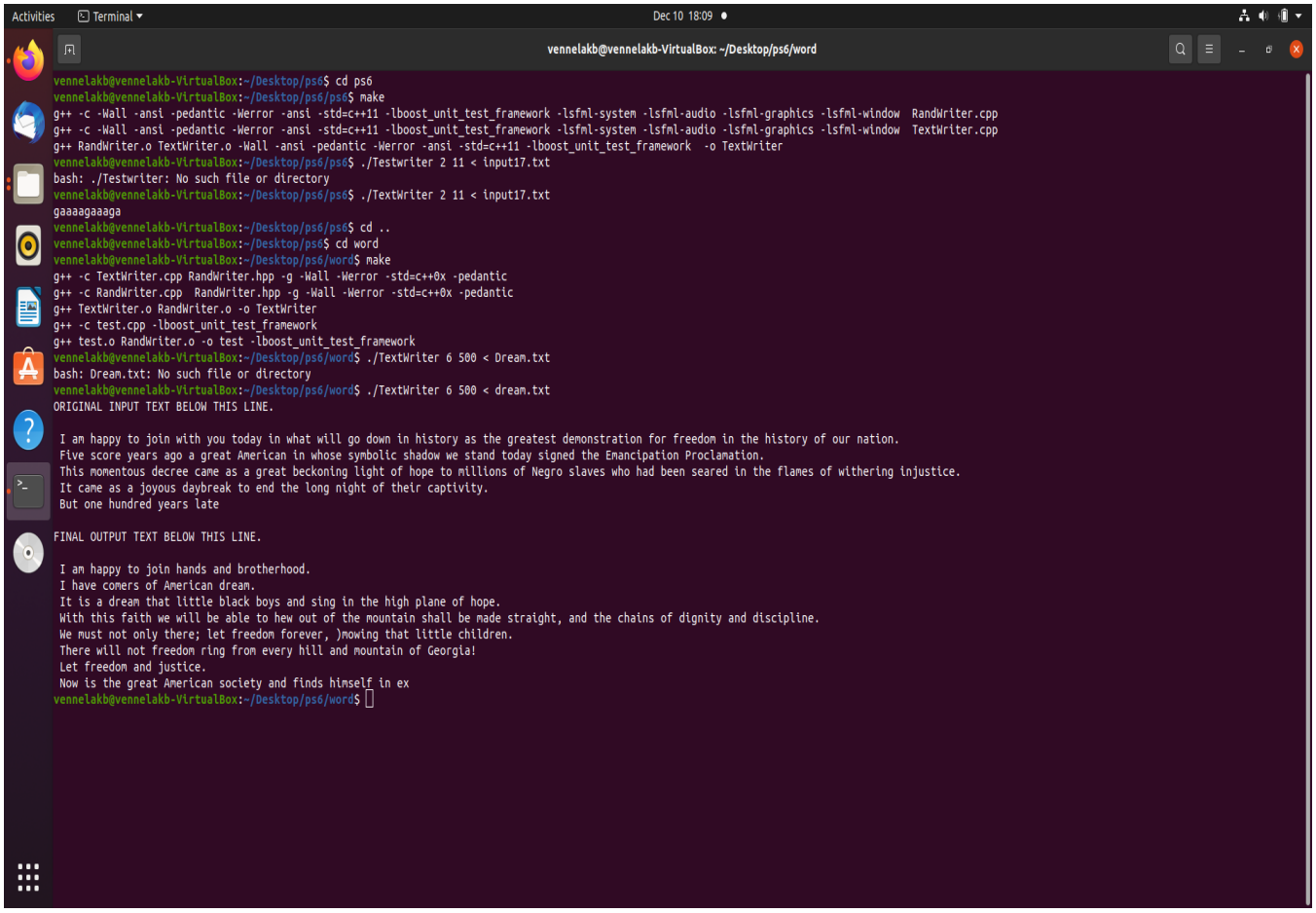
Key Concepts

The Boost regex package is required for the completion of this project since it is utilized to locate certain pieces of data among the specified device log files, which can be tens of thousands of lines long. In order to discover the exact lines of text that include the relevant information for the log report, four regular expressions were employed in particular: one for the date, time, boot sequence string indicating a startup is occurring, and a regex including all three. Furthermore, the Boost date and time functions were implemented to assist in the computation of the elapsed time of successful startup sequences.

What I Learned

This project provided an excellent introduction to regular expressions in the C++ language, and it made me feel much more at ease with the Boost regex library, as well as other regex libraries in general. The goal of the project was to improve my ability to output complete files while also providing a simple approach to correctly parse a file. Finally, the Boost date and time methods improved the efficiency of my code when it came to computing the elapsed time based on the string inputs from the log file.

Screenshots



```
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6$ cd ps6
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/ps6$ make
g++ -c -Wall -ansi -pedantic -Werror -ansi -std=c++11 -lboost_unit_test_framework -lsfml-system -lsfml-audio -lsfml-graphics -lsfml-window  RandWriter.cpp
g++ -c -Wall -ansi -pedantic -Werror -ansi -std=c++11 -lboost_unit_test_framework -lsfml-system -lsfml-audio -lsfml-graphics -lsfml-window  TextWriter.cpp
g++ RandWriter.o TextWriter.o -Wall -ansi -pedantic -Werror -ansi -std=c++11 -lboost_unit_test_framework  -o TextWriter
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/ps6$ ./Testwriter 2 11 < input17.txt
bash: ./Testwriter: No such file or directory
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/ps6$ ./TextWriter 2 11 < input17.txt
gaaaagaaaga
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/ps6$ cd ..
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6$ cd word
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/word$ make
g++ -c TextWriter.cpp RandWriter.hpp -g -Wall -Werror -std=c++0x -pedantic
g++ -c RandWriter.cpp  RandWriter.hpp -g -Wall -Werror -std=c++0x -pedantic
g++ TextWriter.o RandWriter.o -o TextWriter
g++ -c test.cpp -lboost_unit_test_framework
g++ test.o RandWriter.o -o test -lboost_unit_test_framework
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/word$ ./TextWriter 6 500 < Dream.txt
bash: Dream.txt: No such file or directory
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/word$ ./TextWriter 6 500 < dream.txt
ORIGINAL INPUT TEXT BELOW THIS LINE.

 I am happy to join with you today in what will go down in history as the greatest demonstration for freedom in the history of our nation.
 Five score years ago a great American in whose symbolic shadow we stand today signed the Emancipation Proclamation.
 This momentous decree came as a great beckoning light of hope to millions of Negro slaves who had been seared in the flames of withering injustice.
 It came as a joyous daybreak to end the long night of their captivity.
 But one hundred years late

FINAL OUTPUT TEXT BELOW THIS LINE.

 I am happy to join hands and brotherhood.
 I have comers of American dream.
 It is a dream that little black boys and sing in the high plane of hope.
 With this faith we will be able to hew out of the mountain shall be made straight, and the chains of dignity and discipline.
 We must not only there; let freedom forever, )mowing that little children.
 There will not freedom ring from every hill and mountain of Georgla!
 Let freedom and justice.
 Now is the great American society and finds himself in ex
vennelakb@vennelakb-VirtualBox:~/Desktop/ps6/word$
```

Source Code for PS6

Makefile

```
# Makefile for ps6
# Flags to save on typing all this out
CC = g++
CFLAGS = -g -Wall -Werror -std=c++0x -pedantic
SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
Boost = -lboost_unit_test_framework

# Make ps5a & a dummy tester
all:    TextWriter test

# PS6 executable
TextWriter:    TextWriter.o RandWriter.o
        $(CC) TextWriter.o RandWriter.o -o TextWriter

test:   test.o RandWriter.o
        $(CC) test.o RandWriter.o -o test $(Boost)

# Object files
TextWriter.o:TextWriter.cpp RandWriter.h
        $(CC) -c TextWriter.cpp RandWriter.h $(CFLAGS)

RandWriter.o:RandWriter.cpp RandWriter.h
        $(CC) -c RandWriter.cpp  RandWriter.h $(CFLAGS)

test.o:test.cpp
        $(CC) -c test.cpp $(Boost)

# Cleanup object files
clean:
        rm *.o
        rm *.gch
        rm TextWriter
        rm test
```

RandWriter.cpp:

```
// Copyright 2021 Srikanth
#include "RandWriter.h"
#include <utility>
#include <map>
#include <string>

RandWriter::RandWriter(std::string text, int k) {
    _text = text;
    _k = k;

```

```
    if (_text.length() < static_cast<unsigned int>(_k)) {
        throw std::invalid_argument("RandWriter(string text, int k): order k"
        " must be less than or equal to text length.");
    }

    unsigned int pos = 0;
    for (unsigned int i = 0; i < _text.length(); i++) {
        std::string kgram;
        std::map<char, int> _ftable;

        for (unsigned int j = i; j < i + _k; j++) {
            if (j >= _text.length()) {
                pos = j - _text.length();
            } else {
                pos = j;
            }
            kgram += _text.at(pos);
        }

        pos++;
        if (pos >= _text.length()) { pos -= _text.length(); }
        _ftable.insert(std::make_pair(_text.at(pos), 0));

        if (_mtable.count(kgram) == 0) {
            _mtable.insert(std::make_pair(kgram, _ftable));
        }

        _mtable[kgram][_text.at(pos)]++;
    }
}

int RandWriter::order_k() const { return _k; }
std::string RandWriter::getText() const { return _text; }
std::map<std::string, std::map<char, int>> RandWriter::getMTable() const {
    return _mtable;
}

int RandWriter::freq(std::string kgram) const {
    if (kgram.length() < static_cast<unsigned int>(_k)) {
        throw std::runtime_error("freq(string kgram): kgram must be of"
        " length greater than or equal to order k.");
    }
    int count = 0;
    for (unsigned int i = 0; i < _text.length(); i++) {
        unsigned int pos = 0;
        std::string kg;
        for (unsigned int j = i; j < i + _k; j++) {
            if (j >= _text.length()) {
                pos = j - _text.length();
            } else {
                pos = j;
            }
```

```
            kg += _text.at(pos);
        }
        if (kgram == kg) { count++; }
    }
    return count;
}

int RandWriter::freq(std::string kgram, char c) const {
    if (kgram.length() < static_cast<unsigned int>(_k)) {
        throw std::runtime_error("freq(string kgram, char c): kgram must be"
        " of length greater than or equal to order k.");
    }
    return _mtable.at(kgram).at(c);
}

char RandWriter::kRand(std::string kgram) const {
    if (kgram.length() < static_cast<unsigned int>(_k)) {
        throw std::runtime_error("kRand(string kgram): kgram must be of"
        " length greater than or equal to order k.");
    }
    if (_mtable.count(kgram) == 0) {
        throw std::runtime_error("kRand(string kgram): kgram does not"
        " exist.");
    }
    std::string alphabet;
    for (auto const &var1 : _mtable) {
        if (var1.first == kgram) {
            for (auto const &var2 : var1.second) {
                alphabet += var2.first;
            }
        }
    }
    std::random_device device;
    std::mt19937 mt_rand(device());
    std::uniform_int_distribution<int> distribution(0, alphabet.length()
    - 1);

    return alphabet[distribution(mt_rand)];
}

std::string RandWriter::generate(std::string kgram, int L) const {
    if (kgram.length() < static_cast<unsigned int>(_k)) {
        throw std::runtime_error("generate(string kgram, int L): kgram must"
        " be of length greater than or equal to order k.");
    }
    std::string generated = kgram;
    for (int i = _k; i < L; i++) {
        generated += kRand(generated.substr(i - _k, _k));
    }
    return generated;
}
```

```
std::ostream& operator<<(std::ostream& out, const RandWriter& RandWriter) {
    out << "Markov Model\tOrder: " << RandWriter._k << std::endl;
    out << "kgram:\tfrequency:\tfrqncy of next char:\tprob of next char:" <<
    std::endl;

    for (auto const &var1 : RandWriter._mtable) {
        out << var1.first << "\t";
        out << RandWriter.freq(var1.first) << "\t\t";
        for (auto const &var2 : var1.second) {
            out << var2.first << ":" << var2.second << " ";
        }
        out << "\t\t\t";
        for (auto const &var2 : var1.second) {
            out << var2.first << ":" << var2.second << "/" <<
            RandWriter.freq(var1.first) << " ";
        }
        out << std::endl;
    }
    return out;
}
```

RandWriter.h:

```
// Copyright 2021 Srikanth
#ifndef RandWriter_H  //NOLINT
#define RandWriter_H  //NOLINT

#include <iostream>
#include <string>
#include <map>
#include <exception>
#include <utility>
#include <random>

class RandWriter {
 public:
    RandWriter(std::string text, int k);

    // return the order k
    int order_k() const;

    // return the input text
    std::string getText() const;

    std::map<std::string, std::map<char, int>> getMTable() const;

    int freq(std::string kgram) const;
```

```

    int freq(std::string kgram, char c) const;

    char kRand(std::string kgram) const;

    std::string generate(std::string kgram, int L) const;

    friend std::ostream& operator<<(std::ostream& out, const RandWriter&
    RandWriter);

 private:
    int _k;             // order of Markov Model
    std::string _text;  // text to analyze

    std::map<std::string, std::map<char, int>> _mtable;
};
#endif   //NOLINT
```

Test.cpp:

```
// Copyright 2021 Srikanth
#include "RandWriter.h"
#include <string>

#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(base_test) {
    std::cout << " Test Case 1 " <<
    std::endl;

    int k = 2;
    std::string str = "gagggagaggcgagaaa";
    RandWriter RandWriter(str, k);
    std::cout << "Printing out Markov Table for string:\n" <<
    str << std::endl << std::endl;
    std::cout << RandWriter << std::endl;
    std::cout << "Testing order_k and freq functions" << std::endl;
    BOOST_REQUIRE(RandWriter.order_k() == k);
    BOOST_REQUIRE(RandWriter.freq("gg") == 3);
    BOOST_REQUIRE(RandWriter.freq("ga", 'g') == 4);

    std::cout << "Testing kRand function" << std::endl;
    char rand = RandWriter.kRand("aa");
```

```
    BOOST_REQUIRE(rand == 'a' || rand == 'g');

    std::cout << "Testing generate function" << std::endl << std::endl;
    BOOST_REQUIRE(RandWriter.generate("ga", 10).length() == 10);
}

BOOST_AUTO_TEST_CASE(exception_test) {
    std::cout << " Test Case 2 " <<
    std::endl;
    std::cout << "Testing construction exception: RandWriter('ADF', 4)" <<
    std::endl;

    BOOST_REQUIRE_THROW(RandWriter("ADF", 4), std::invalid_argument);

    std::cout << "Testing function exceptions" << std::endl;
    RandWriter testMM("abc", 3);
    BOOST_REQUIRE_THROW(testMM.freq("a"), std::runtime_error);
    BOOST_REQUIRE_THROW(testMM.freq("ab", 'b'), std::runtime_error);
    BOOST_REQUIRE_THROW(testMM.kRand("g"), std::runtime_error);
}
```

TestWriter.cpp:

```
// Copyright 2021 Srikanth
#include "RandWriter.h"
#include <fstream>

int main(int argc, char *argv[]) {
    if (argc != 3) {
        std::cerr << "Usage: ./TextWriter k L < input.txt" << std::endl;
        exit(-1);
    }
    int k = std::atoi(argv[1]);
    int L = std::atoi(argv[2]);

    int count = 0;
    int length = 0;
    std::string input;
    std::string output;

    // read input line by line and generate output
    while (std::getline(std::cin, input) && count < L) {
        if (input.length() > static_cast<unsigned int>(k)) {
            try {
```

```
                RandWriter RandWriter(input, k);
                if (static_cast<int>(input.length()) > L) {
                    length = L;
                } else if (static_cast<int>(input.length()) + count > L) {
                    length = L - count;
                } else {
                    length = input.length();
                }
                output = RandWriter.generate(input.substr(0, k), length);
                count += output.length();
                std::cout << output << std::endl;
            }
            catch (std::invalid_argument const& err) {
                std::cerr << err.what() << std::endl;
                exit(-1);
            }
            catch (std::runtime_error const& err) {
                std::cerr << err.what() << std::endl;
                exit(-1);
            }
        }
    }

    return 0;
}
```

PS7: Kronos Time Clock

Assignment Description

This assignment involved using regular expressions to parse files of various Kronos InTouch time clock logs to analyze them, verifying the device's boot up timing, and nothing whether these startups were fully successful or not. The purpose of this is so, information can be gathered on what was occurring with the device at the time of the bootup failures, in order to eventually solve the problems occurring with the InTouch device. In order to help do this. PS6 asks us to scan the complete log files given and create a text file report chronologically describing each time the device was restarted, noting if it failed or succeeded in completely doing so, and giving the elapsed time for the sequence if the bootup was successful.

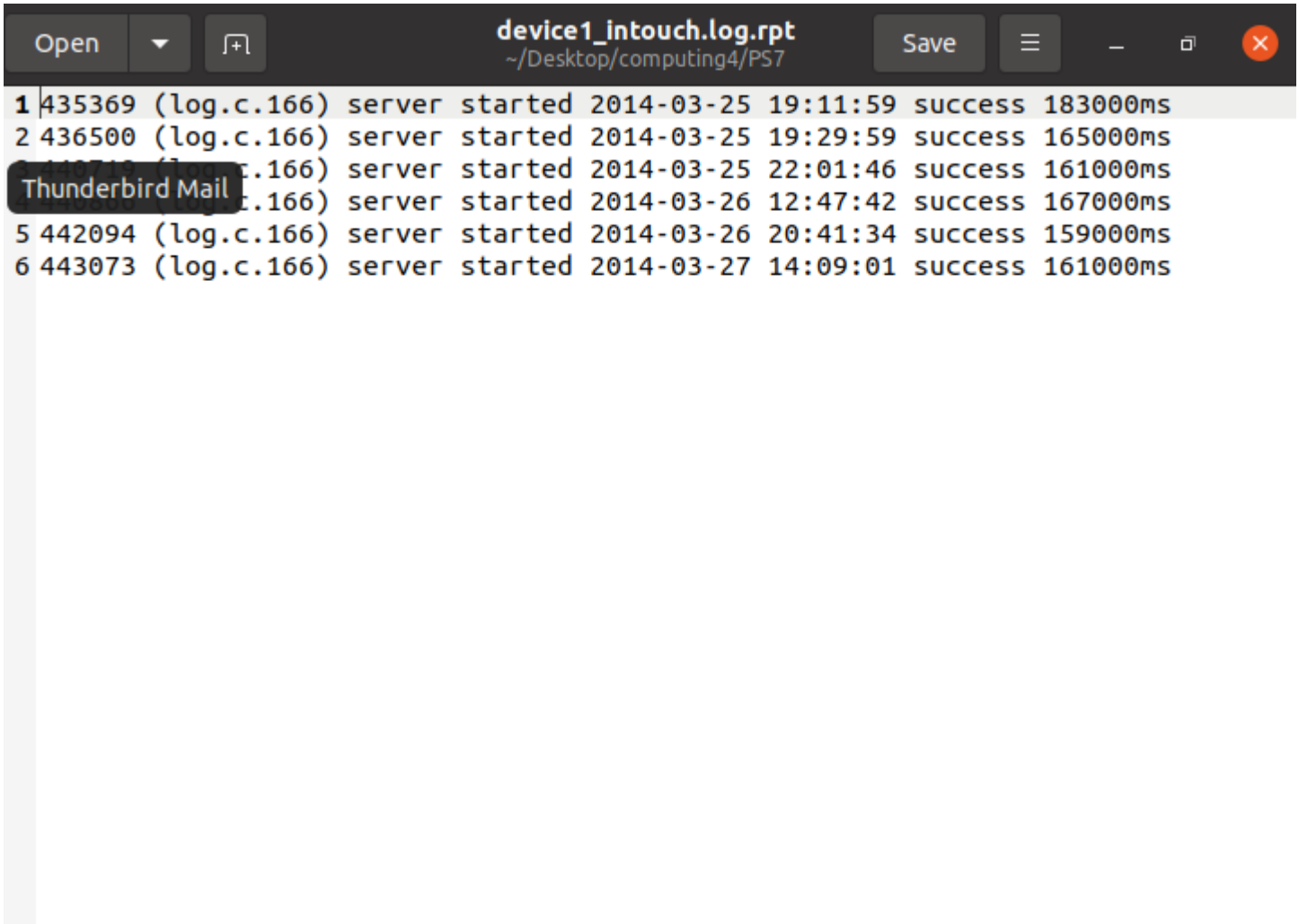
Key Concepts

The usage of the Boost regex library is essential to the completion of this assignment, as it is used in order to find the specific pieces of data located within the given device log files, which are often tens of thousands of lines long. Four regular expressions were used in particular in order to find the specific lines of text that contain the needed information for the log report: one for the date, time, one for the boot sequence string indicating a startup is occurring, and a regex containing all three together. On top of this, the Boost date and time functions were implemented in order to serve as a helping hand to computing the elapsed time of the successful startup sequences.

What I Learned

I learned a ton about using regular expressions in this program – enough, that I feel like I could probably do some other parsing of files in the future if a job or future class requires it. I also can see a lot of practical uses for a program such as this – an intern (probably me in the future) could easily create parsing program to verify successful / failure code, or to double check that certain devices are working properly. It was also pretty handy to learn about date and time. Having used Boost's date and time libraries, I feel pretty good about using other libraries in the future that involve date and time. It has given me a solid introduction to working with dates – perhaps in the future I may need to do something that involves calculation dates and time.

Screenshots




```
435369 (log.c.166) server started 2014-03-25 19:11:59 success 183000ms
436500 (log.c.166) server started 2014-03-25 19:29:59 success 165000ms
.166) server started 2014-03-25 22:01:46 success 161000ms
.166) server started 2014-03-26 12:47:42 success 167000ms
442094 (log.c.166) server started 2014-03-26 20:41:34 success 159000ms
443073 (log.c.166) server started 2014-03-27 14:09:01 success 161000ms
```

Source Code for PS7

Makefile

```
# Makefile for PS7
# Flags to save on typing all this out
CC = g++
CFLAGS = -g -Wall -Werror -std=c++0x -pedantic
SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
Boost = -lboost_regex -lboost_date_time

#
all:    ps7

# PS7A executable
ps7a:   ps7.o
    $(CC) ps7.o -o ps7 $(Boost)

# Object files
ps7a.o: ps7.cpp
    $(CC) -c ps7.cpp $(CFLAGS)

# Cleanup object files
clean:
    rm *.o
    rm ps7

```

main.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <boost/regex.hpp>
#include "boost/date_time/posix_time/posix_time.hpp"


using std::cout;
using std::cin;
using std::endl;
using std::string;
using boost::regex;
using boost::smatch;
using boost::regex_error;
using boost::gregorian::date;
using boost::gregorian::from_simple_string;
using boost::gregorian::date_period;
using boost::gregorian::date_duration;
using boost::posix_time::ptime;
using boost::posix_time::time_duration;


int main(int argc, char **args)
{
    if (argc != 2)
    {
        cout << "usage: ./ps7 [logfile]" << endl;
        exit(1);
    }
    string s, rs;
    regex e1;
    regex e2;
    bool flag = false;
    ptime t1, t2;
    string filename(args[1]);
    std::ifstream infile(filename);
    std::ofstream outfile(filename + ".rpt");
    if (!infile || !outfile)
    {
        cout << "open file error" << endl;
        exit(1);
    }
    try
    {
        e1 = regex(R"((.*): (\(log.c.166\) server started.*))");
        e2 = regex("(.*)\\.\\d*:INFO:oejs.AbstractConnector:Started "
                   "SelectChannelConnector@0.0.0.0:9080.*");
    }
    catch (regex_error &exc)
    {
```

```
        cout << "Regex constructor failed with code " << exc.code() << endl;
        exit(1);
    }
    int line_number = 1;
    string str;
    while (getline(infile, s))
    {
        if (regex_match(s, e1))
        {
            smatch sm;
            regex_match(s, sm, e1);
            if (flag)
            {
                outfile << "failure" << endl;
            }
            flag = true;
            t1 = ptime(boost::posix_time::time_from_string(sm[1]));
            str = sm[2];
            outfile << line_number << " (log.c.166) server started "
                    << sm[1] << " ";
        }
        if (regex_match(s, e2))
        {
            smatch sm;
            regex_match(s, sm, e2);
            t2 = ptime(boost::posix_time::time_from_string(sm[1]));
            outfile << "success " << (t2 - t1).total_milliseconds()
                    << "ms" << endl;
            flag = false;
        }
        line_number++;
    }
    auto lamdaexp=[&]()->bool{
      bool temp=flag;
      return temp;
    };
    bool lamdatemp=lamdaexp();
    if (lamdatemp)
    {
        outfile << "failure" << endl;
    }
    infile.close();
    outfile.close();
    return 0;
    }
```