Computing IV Sec 011: Project Portfolio

Aniketh Rai

Summer 2022

Contents

TIME TO COMPLETE PORTFOLIO : 12 Hours

1 PS0: Hello SFML

1.1 Discussion:

The first project of CompIV 22 is **Hello World with SFML**. In this assignment, At first we set up our environment by installing the SFML library. We also check for the newest version of the C++. Thereby, We test the given code and check whether it is working or not. At the last part of the assignment we include and image and use the sprite property of the SFML to move the image using the Keys of the Keyboard. The output is in **Figure: 2**

1.2 Key algorithms, Data structures and OO Designs used in this Assignment:

This assignment did not require any of the key algorithm, Data structures and OO Designs, as the project it self is basic project with the code provided and we just had to add SFML sprite and few Keyboard events to complete the project. So, There is no use of the Algorithms and Data Structures.

1.3 Images used:

Figure 1: Sprite Image

1.4 What I accomplished :

I accomplished to move the Sprite image using Keyboard keys as well as with the Mouse click.

1.5 What I already knew :

I was aware of how to move the image using keyboard keys and also mouse pointer events as I already learnt Javascript. So, It was much easy to do. I know SFML is different than JavaScript but the concept of it is similar to me.

1.6 What I learned :

I learned to use SFML for the first time and also how to add events and manipulate them in the SFML field. Overall, It was fun to do this assignment, as everything for me in this assignment was new an amazing.

1.7 Challenges :

Setting up the SFML environment was challenging to me. Also learning much about the SFML functions.

1.8 Acknowledgements:

- `https://www.sfml-dev.org/tutorials/2.5/`
- `https://youtu.be/axIgxBQVBg0`

1.9 Codebase

Makefile:

This Makefile was provided in portal.

```
CC = g++
CFLAGS = --std=c++14 -Wall -Werror -pedantic
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework

all: sfml-app

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

sfml-app: main.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

clean:
    rm *.o sfml-app
```

main.cpp:

The main file where the code runs and provides the valid output as shown in figure 2.

```
#include <SFML/Graphics.hpp>

int main()
{
    sf::RenderWindow rai_window(sf::VideoMode(700, 700), "HELLO WORLD");
    sf::CircleShape rai_circle(50.0f);
    rai_circle.setFillColor(sf::Color::Magenta);
    sf::Texture rai_texture;
    rai_texture.loadFromFile("sprite.png");

    sf::Sprite rai_sprite(rai_texture);

    rai_sprite.setPosition(100.0f,0);









    while (rai_window.isOpen())
    {
        sf::Event event;
        while (rai_window.pollEvent(event))
        {

            if (event.type == sf::Event::Closed)
                rai_window.close();
             if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::BackSpace))
             rai_sprite.setPosition(100.0f,0);


             if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::A))
             {
               rai_sprite.move(-10.0f,0);

        }
            if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::D))
             {
```

```
38                 rai_sprite.move(10.0f,0);
39
40         }
41             if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::W))
42              {
43                rai_sprite.move(0,-10.0f);
44
45         }
46             if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::S))
47              {
48                rai_sprite.move(0,10.0f);
49
50               }
51               if(sf::Mouse::isButtonPressed(sf::Mouse::Left))
52               { sf::Vector2i mousePos=sf::Mouse::getPosition(rai_window);
53                 rai_sprite.setPosition((float)mousePos.x,(float)mousePos.y);
54               }
55
56
57
58
59         rai_window.clear();
60         rai_window.draw(rai_circle);
61         rai_window.draw(rai_sprite);
62         rai_window.display();
63
64     }
65     }
66     return 0;
67
68 }
```

1.10 Output:

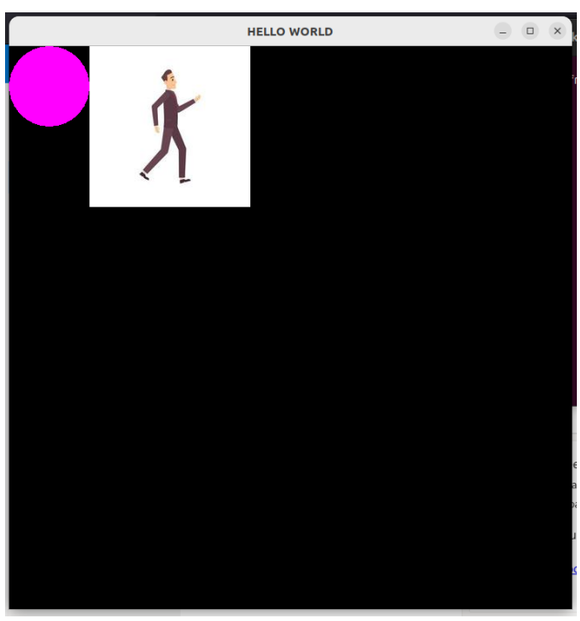

Figure 2: Output of PS0 Assignment

2 PS1a: Linear Feedback Shift Register

2.1 Discussion:

The ps1a assignment is an implementation of **LFSR(Linear Feedback shift Register) which is the Fibonacci LFSR**. This assignment is used for the ps1b i.e PhotoMagic. In this project, there are two main functions i.e, step() and generate(), The step() funtion is used for left shifting the one bit of the given seed, along the lsb is the result of the tap positions. These tap positions use the XOR operations and later it gives the lsb result. The generate() generates the states according to the given k inputs.
XOR Truth Table:

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

2.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I have used string for the seed. Rather than using the xor operator I used the != operator as it works same and I have tried using it. I used simple string functions also.
The Tap position algorithm is as follows:

```
1  int _TAPbitvalue = funXOR(rgs[0], rgs[2]);
2 _TAPbitvalue = funXOR(_TAPbitvalue, funGetBit(rgs[3]));
3 _TAPbitvalue = funXOR(_TAPbitvalue, funGetBit(rgs[5]));
4
```

2.3 What I accomplished :

I accomplished the full work of the LFSR and also both the important functions step() as well as generate() works completely fine. I learnt how to lint the program using the **cpplint.**

2.4 What I already knew :

I knew the concept of XOR and also I am well aware of how to shift the bits and also the concept of the MSB and LSB, as I have already taken Digital Logic design and Computer Architecture back in India.

2.5 What I learned :

I learned that how this LFSR is going to be used for the Image encoding and decoding of the Image in further part of the PS1 code. Where we utilize this part of assignment.

2.6 Challenges :

To generate the LSB bit from the tap positions and re-attaching it to the seed was tough and also writing the new test case was challenging for me as I was new to the concept of the Boost Library.

2.7 Codebase

Makefile:

This Makefile is created by the referrence of the Version2 Makefile from the notes.

```
#Copyright 2022 Aniketh Rai
boost = boost_unit_test_framework
all: ps1a lint
lint:
    cpplint --filter=-runtime/references,-build/c++11 --root=. *.h *.cpp
ps1a: FibLFSR.o test.o
    g++ FibLFSR.o test.o -o ps1a -l $(boost)

FibLSFR.o: FibLSFR.cpp FibLSFR.h
    gcc -c FibLSFR.cpp

test.o: FibLFSR.h FibLFSR.cpp test.cpp
    g++ -c FibLFSR.cpp test.cpp -l $(boost)

clean:
    rm *.o
    rm ps1a
```

FibLFSR.cpp:

This file contains the important methods such as step() and generate().

```
// Copyright 2022 Aniketh Rai
#include "FibLFSR.h"
#include <math.h>
#include <string>
int FibLFSR::funGetBit(char a) {
if (a == '1') return 1;


else if (a == '0') return 0;


else
return -1;
}
std::ostream& operator<< (std::ostream& out, const FibLFSR& fibLFSR) {
    out << fibLFSR.rgs;
    return out;
}
FibLFSR::FibLFSR(std::string seed) {
    rgs = seed;
}
int FibLFSR::funXOR(int a, int b) {
    return a != b;
}
int FibLFSR::step() {
std::string _newRGS = rgs.substr(1); // NOLINT
int _TAPbitvalue = funXOR(rgs[0], rgs[2]);
_TAPbitvalue = funXOR(_TAPbitvalue, funGetBit(rgs[3]));
_TAPbitvalue = funXOR(_TAPbitvalue, funGetBit(rgs[5]));
FibLFSR::rgs = _newRGS;
FibLFSR::rgs += std::to_string(_TAPbitvalue);
return _TAPbitvalue;
}

int FibLFSR::generate(int k) {
```

```
int _result = 0;
for(int i = 0; i < k; i++) {
    int x = step();
    std::cout << rgs << " " << x << std::endl;
    _result = (_result * 2) + x;
}
return _result;
}
```

FibLFSR.h:

this file is the header file for the above provided file "FibLFSR.cpp" . This file contains the initialization of the functions, libraries and variables.

```
// Copyright 2022 Aniketh Rai
#ifndef FIBLFSR_H_
#define FIBLFSR_H_
#include <iostream>
#include <string>
class FibLFSR {
 public:
FibLFSR(std::string seed);
int step();
int generate(int k);
friend std::ostream& operator<<
(std::ostream& out, const FibLFSR& fibLFSR);
 private:
std::string rgs; // NOLINT
int funGetBit(char a);
int funXOR(int a, int b);
};
#endif  // FIBLFSR_H_
```

test.cpp:

This file is the test file which utilizes the <Boost Library>
The test cases are as follows:

- **At first case it was simple 16 bit seed (given)**
- **At second case it is random 20 bit seed**
- **At third case it is random 16 bit seed**

```
// Copyright 2022 Aniketh Rai
#include <iostream>
#include <string>
#include "FibLFSR.h"
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 1);
  BOOST_REQUIRE(l.step() == 1);
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 0);
  BOOST_REQUIRE(l.step() == 1);

  FibLFSR l2("1011011000110110");
```

```
21   BOOST_REQUIRE(l2.generate(9) == 51);
22 }
23 BOOST_AUTO_TEST_CASE(TwentyBitsThreeTaps) {
24   FibLFSR l("10101001101010011010");
25   BOOST_REQUIRE(l.step() == 0);
26   BOOST_REQUIRE(l.step() == 1);
27   BOOST_REQUIRE(l.step() == 1);
28   BOOST_REQUIRE(l.step() == 1);
29
30   FibLFSR l2("10101001101010011010");
31   BOOST_REQUIRE(l2.generate(5) == 14);
32 }
33
34 BOOST_AUTO_TEST_CASE(SixteenBitsThreeTaps2) {
35   FibLFSR l("1011011000110111");
36   BOOST_REQUIRE(l.step() == 0);
37   BOOST_REQUIRE(l.step() == 0);
38   BOOST_REQUIRE(l.step() == 0);
39   BOOST_REQUIRE(l.step() == 1);
40   BOOST_REQUIRE(l.step() == 1);
41   BOOST_REQUIRE(l.step() == 0);
42
43   FibLFSR l2("1011011000110111");
44   BOOST_REQUIRE(l2.generate(5) == 3);
45 }
```

3 PS1b: PhotoMagic

3.1 Discussion:

The ps1b utilizes the ps1a i.e **LFSR: LINEAR FEEDBACK SHIFT REGISTER** as a supporting for the encoding and decoding of the image given as input.This project takes the input image i.e, Cat.png and Encrypts using the LFSR Randomizer and encrypts each and every pixel of the image and saves to the cat-out.png file. For Encryption I used Common Example i.e;

./PhotoMagic cat.png cat-out.png 0011001100000 8

Figure:4

If we want to decrypt the encrypted image we write it as follows.

./PhotoMagic cat-out.png cat.png 0011001100000 8

Figure: 6

3.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I used The Vector STL in the LFSR as it is much easier to use and I felt flexible in it. Also Improvised the sample code to the Final code to get the perfect output. I used Image and its funtions and letter sent to Texture then to Sprite to draw on the window I used Two windows and two sprites for showing the difference between the normal cat image and the encoded and decoded image. The following code is used for the encoding and decoding:

```
1  void transform(sf::Image& kitty_image, FibLFSR* fibo){
2      sf::Vector2u size = kitty_image.getSize();
3      sf::Color color, newColor;
4
5      for(int i = 0; i < size.x; i++){
6          for(int j = 0; j < size.y; j++){
7              color = kitty_image.getPixel(i, j);
8
9              color.r = color.r xor fibo->generate(8);
10             color.g = color.g xor fibo->generate(8);
11             color.b = color.b xor fibo->generate(8);
12
13             kitty_image.setPixel(i, j, color);
14         }
15     }
16 }
17
```

3.3 Images used:

Figure 3: Cat Image

3.4 What I accomplished :

I accomplished the Encoding and Decoding of the Image using the C++ language in an efficient way. I have achieved using multiple windows, which was an amazing thing to do.

3.5 What I already knew :

I knew how to take the input of the file and write to an output file. I already knew about how to create sprite using images and texture. I also knew how to display the windows and also to create the Makefile for the project.

3.6 What I learned :

I learned how to utilize two windows for the different output. I understood the concept of encoding and decoding using the pixels of the image and using the LFSR Randomizer. I also got to know about the mathematical calculations for the pixels.

3.7 Challenges :

To find out the image pixels and the also to identify how to tranform them to the normal to encoded and then decode the image.

3.8 Acknowledgements :

- `https://www.sfml-dev.org/tutorials/2.5/`

3.9 Codebase

Makefile:

This Makefile is contains no lint but it includes the flags as well as it is extension of the ps1a Makefile.

```
Compiler = g++
flags = -Wall -Werror -std=c++14 -pedantic

all: ps1_partone PhotoMagic

PhotoMagic: PhotoMagic.o FibLFSR.o
    $(Compiler) $(flags) -o PhotoMagic PhotoMagic.o FibLFSR.o -lsfml-
   graphics -lsfml-window -lsfml-system

ps1_partone: test.o FibLFSR.o
    $(Compiler) $(flags) -o ps1a test.o FibLFSR.o -
   lboost_unit_test_framework

photomagic.o: PhotoMagic.cpp PhotoMagic.h
    $(Compiler) $(flags) -c PhotoMagic.cpp

test.o: test.cpp FibLFSR.hpp
    $(Compiler) $(flags) -c test.cpp

FibLFSR.o: FibLFSR.cpp FibLFSR.hpp
    $(Compiler) $(flags) -c FibLFSR.cpp

clean:
    rm -f *.o ps1_partone PhotoMagic
```

PhotoMagic.cpp:

This file is the main file where the reading and writing also the encoding and decoding of the image takes place. This file gives the output in two windows. Input window and output window of the file.

```

#include "PhotoMagic.h"
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

int main(int argc, char*argv[]) {
    std::string pw = argv[3];
    std::string seed;
    std::string sF = "";

    if(pw.size() < 16 && pw.size() > 0){
        int i = 0;
        std::string seed = pw;
        while(seed.size() != 16){
            seed.push_back(pw[i]);
            i++;
            i = i % pw.size();
        }
    }

    if(pw.size() > 16){
        for(int i = 0; i < 16; i++){
            seed.push_back(pw[i]);
        }
    }
```

```
27
28      if(pw.size() == 16){
29          seed = pw;
30      }
31
32
33      for(int i = 0; i < 16; i++){
34          sF += std::to_string((unsigned int)(seed[i])%2);
35      }
36      sf::Image kitty_image;
37
38      if (!kitty_image.loadFromFile("cat.png"))
39          return -1;
40      sf::Texture texture;
41      texture.loadFromImage(kitty_image);
42
43      sf::Sprite Kitty1;
44      Kitty1.setTexture(texture);
45
46      FibLFSR fibo(sF);
47
48      transform(kitty_image, &fibo);
49
50
51      if (!kitty_image.saveToFile("cat-out.png"))
52          return -1;
53      sf::Image kitty_img2;
54      if(!kitty_img2.loadFromFile("cat-out.png"))
55           return -1;
56      sf::Texture tex1;
57      tex1.loadFromImage(kitty_img2);
58      sf::Sprite Kitty2;
59      Kitty2.setTexture(tex1);
60
61
62
63      sf::Color p;
64
65
66      for (int x = 0; x<350; x++) {
67          for (int y = 0; y< 250; y++) {
68              p = kitty_image.getPixel(x, y);
69              p.r = 255 - p.r;
70              p.g = 255 - p.g;
71              p.b = 255 - p.b;
72              kitty_image.setPixel(x, y, p);
73          }
74      }
75
76      sf::Vector2u size_of_img = kitty_image.getSize();
77      sf::RenderWindow window_one(sf::VideoMode(size_of_img.x, size_of_img.y),
        "Kitty Cat");
78      sf::RenderWindow window_two(sf::VideoMode(size_of_img.x, size_of_img.y),
        "No hacker can hack :D");
79
80
81
82
83      while (window_one.isOpen() && window_two.isOpen())
```

```
    {
        sf::Event event1,event2;
        while (window_one.pollEvent(event1))
        {
            if (event1.type == sf::Event::Closed)
                window_one.close();
        }
        while (window_two.pollEvent(event2))
        {
            if (event2.type == sf::Event::Closed)
                window_two.close();
        }

        window_one.clear();
        window_one.draw(Kitty1);
        window_one.display();

        window_two.clear();
        window_two.draw(Kitty2);
        window_two.display();
    }




    return 0;
}
void transform(sf::Image& kitty_image, FibLFSR* fibo){
    sf::Vector2u size = kitty_image.getSize();
    sf::Color color, newColor;

    for(int i = 0; i < size.x; i++){
        for(int j = 0; j < size.y; j++){
            color = kitty_image.getPixel(i, j);

            color.r = color.r xor fibo->generate(8);
            color.g = color.g xor fibo->generate(8);
            color.b = color.b xor fibo->generate(8);

            kitty_image.setPixel(i, j, color);
        }
    }
}
```

PhotoMagic.h:

This file contains the initializations and header files for the PhotoMagic.cpp

```
#include <SFML/Graphics.hpp>
#include <string>
#include <iostream>
#include "FibLFSR.hpp"
void transform(sf::Image& kitty_image, FibLFSR* fibo);
```

test.cpp:

Given test file for the ps1a Assignment.

```
// Copyright 2022
// By Dr. Rykalova
// Editted by Dr. Daly
// test.cpp for PS1a
// updated 5/12/2022

#include <iostream>
#include <string>

#include "FibLFSR.hpp"

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

BOOST_AUTO_TEST_CASE(testStepInstr1) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
}

BOOST_AUTO_TEST_CASE(testStepInstr2) {
  FibLFSR l2("1011011000110110");
  BOOST_REQUIRE_EQUAL(l2.generate(9), 51);
}
```

3.10 Output:

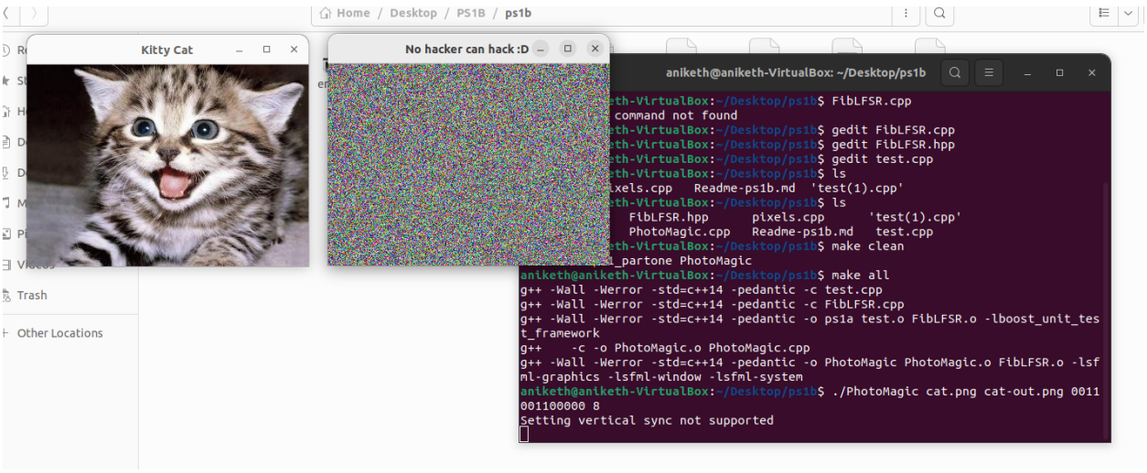

Figure 4: Encoded Image

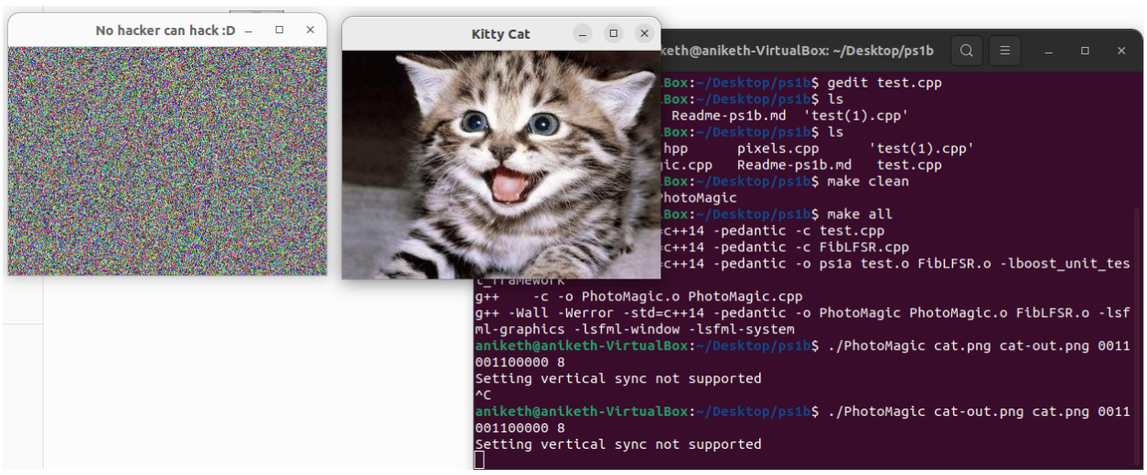

Figure 5: Decode Image

4 PS2: Traingle Fractal

4.1 Discussion:

This Triangle Fractal is a variation of the Sierpinski triangle. The Polish mathematician Waclaw Sierpinski described the pattern in 1915, but it has appeared in Italian art since the 13th century.

In this Assignment, We have to create a Triangle using only Recursion and display on the window by the help of the SFML. We have to utilize the SFML Drawable class for the drawing of the Triangles. These traingles are similar to Sierpinski triangle but not the same. In this ps2, The triangles are created for the 3 sides of each and every triangle until unless the depth of the triangle becomes zero. The program should take two command line inputs, L and N in that order:

- L The length of the side of the base equilateral triangle (double)
- N The depth of the recursion (int)

The command Line is as Follows:
./TFractal 200 5

4.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I added Escape key to close the window as It was convenient. I made use of VertexArray to draw individual triangles themselves. The Triangle is completely draw until the depth value becomes zero. I added colors to the triangle. I Used cmath for the math functions.I added background to the triangle as you can see in the output **Figure: 7**
Changes I did:

- I made use of VertexArray for individual triangle.
- Implemented different Mathematic formulas.
- Added Background to the Triangle.
- Added Cpplint to target file.
- Used randr() rather than rand()

Recursive method used in ftree:

```
1 fTree(tris, len / 2, depth - 1, _xcen_a, ycenter_a);
2 fTree(tris, len / 2, depth - 1, _xcen_b, ycenter_b);
3 fTree(tris, len / 2, depth - 1, _xcen_c, ycenter_c);
4
```

Random Color Code:

```
1     int v1 = rand_r(&range) % 255;
2 int v2 = rand_r(&range) % 255;
3 int v3 = rand_r(&range) % 255;
4 triangle[0].position = sf::Vector2f(xvertex_a , yvertex_a);
5 triangle[0].color = sf::Color(v1 , v2 , v3);
6 triangle[1].position = sf::Vector2f(xvertex_b , yvertex_b);
7 triangle[1].color = sf::Color(v1 , v2 , v3);
8 triangle[2].position = sf::Vector2f(xvertex_c , yvertex_c);
9 triangle[2].color = sf::Color(v1 , v2 , v3);
10 triangle[3].position = triangle[0].position;
11 triangle[3].color = sf::Color(v1 , v2 , v3);
12
```

4.3 Images used:

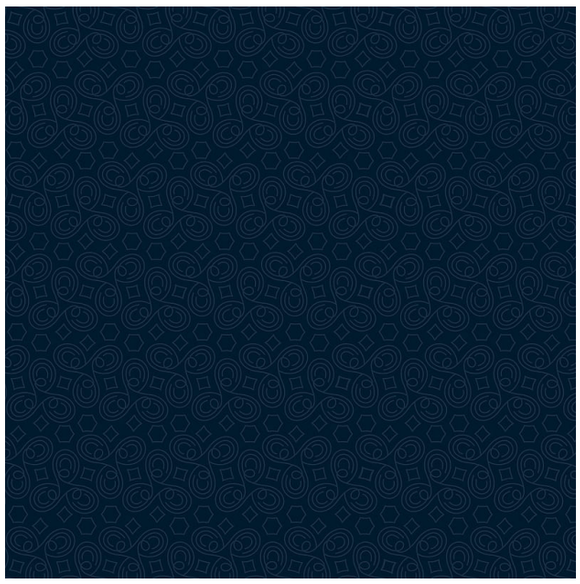

Figure 6: Background for the Triangle

4.4 What I accomplished :

I have accomplished the creation of the Triangle Fractal using the Recursive methods. It was really an Amazing and effective way to create patterns using recursion. I have added random colors for it to gain extra credit.

4.5 What I already knew :

I already knew how to create a window and few events. I also knew how to add background to the window and set the window size. I also knew how to take command line arguments.

4.6 What I learned :

I learned how to implement the sf::Drawable class in much efficient manner. I also got to know how to utilize the mathematic functions and recursive methods to draw amazing patterns.

4.7 Challenges :

To find out the exact math formula and align the triangle to the center was the challenging part.

4.8 Codebase

Makefile:

This Makefile contains the linting too.

```
CC = g++
CFLAGS = -std=c++17 -Wall -Werror -pedantic
LFLAGS = -lsfml-system -lsfml-window -lsfml-graphics
all: TFractal lint
lint:
    cpplint --filter=-runtime/references,-build/c++11 --root=. *.h *.cpp
TFractal: Triangle.o TFractal.o
    $(CC) -o $@ Triangle.o TFractal.o $(LFLAGS)

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

Triangle.o: Triangle.cpp
    $(CC) $(CFLAGS) -c Triangle.cpp -o Triangle.o

clean:
    rm -f TFractal *.o
```

TFractal.cpp

This file is important as the Recursion takes place as well as the window is drawn. It calls the Triangle.cpp for creating the Triangles until unless the depth becomes zero.

```
/*Copyright 2022 Aniketh Rai*/
#include <iostream>
#include <string>
#include <vector>
#include "Triangle.h"
#include <cmath>
#define WIDTH 800
#define HEIGHT 800

void fTree(std::vector<Triangle> *tris , double len ,
            int depth , double _xcen , double _ycen) {
if (depth < 0)
return;
tris->push_back(Triangle(_xcen, _ycen, len));
double height = (std::sqrt(3) / 2.0) * len;
double _xcen_a = _xcen - (len / 2.0);
double ycenter_a = _ycen - (2.0/3.0) * height;
double _xcen_b = _xcen + (3.0/4.0) * len;
double ycenter_b = _ycen - (1.0/6.0) * height;
double _xcen_c = _xcen - (1.0/4.0) * len;
double ycenter_c = _ycen + (5.0/6.0) * height;
fTree(tris, len / 2, depth - 1, _xcen_a, ycenter_a);
fTree(tris, len / 2, depth - 1, _xcen_b, ycenter_b);
fTree(tris, len / 2, depth - 1, _xcen_c, ycenter_c);}
int main(int argcc, char* argv[]) {
double len = std::stod(argv[1]);
int depth = std::stoi(argv[2]);
std::vector<Triangle> tris;
double centerx = 400;
double centery = 400;
tris.clear();
fTree(&tris, len, depth, centerx, centery);
sf::RenderWindow window(sf::VideoMode(WIDTH , HEIGHT), "TFractal");
sf::Image img;
```

```
img.loadFromFile("backg.jpg");
sf::Texture t;
t.loadFromImage(img);
sf::Sprite s;
s.setTexture(t);
while (window.isOpen()) {
sf::Event event;
while (window.pollEvent(event)) {
if (event.type == sf::Event::Closed)
window.close();
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Escape))
window.close();}
window.clear();
window.draw(s);
for (Triangle &x : tris)
window.draw(x);
window.display();}
return 0;
}
```

Triangle.h

It is a header file where the initialization of libraries as well as variables and methods takes place.

```
/*Copyright 2022 Aniketh Rai*/
#ifndef TRIANGLE_H_
#define TRIANGLE_H_
#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
class Triangle : public sf::Drawable {
 public:
Triangle(double xc, double yc, double len);
 private:
virtual void draw(sf::RenderTarget& target , sf::RenderStates states) const
   {
            target.draw(triangle , states);}
sf::VertexArray triangle;
};
#endif  // TRIANGLE_H_
```

Triangle.cpp

This file utilizes the SFML library to create the Triangles.

```
/*Copyright 2022 Aniketh Rai*/
#include "Triangle.h"
#include <stdlib.h>
#include <cmath>
static unsigned int range = 54321;
Triangle::Triangle(double xc, double yc,
                double len):triangle(sf::LineStrip, 4) {
double height = std::sqrt(3.0) / 2.0 * len;
double xvertex_a = xc - (len / 2.0);
double yvertex_a = yc - (height / 3.0);
double xvertex_b = xc + (len / 2.0);
double yvertex_b = yc - (height / 3.0);
double xvertex_c = xc;
double yvertex_c = yc + (2.0/3.0) * height;
int v1 = rand_r(&range) % 255;
int v2 = rand_r(&range) % 255;
int v3 = rand_r(&range) % 255;
triangle[0].position = sf::Vector2f(xvertex_a , yvertex_a);
triangle[0].color = sf::Color(v1 , v2 , v3);
triangle[1].position = sf::Vector2f(xvertex_b , yvertex_b);
triangle[1].color = sf::Color(v1 , v2 , v3);
triangle[2].position = sf::Vector2f(xvertex_c , yvertex_c);
triangle[2].color = sf::Color(v1 , v2 , v3);
triangle[3].position = triangle[0].position;
triangle[3].color = sf::Color(v1 , v2 , v3);
}
```

4.9 Output:

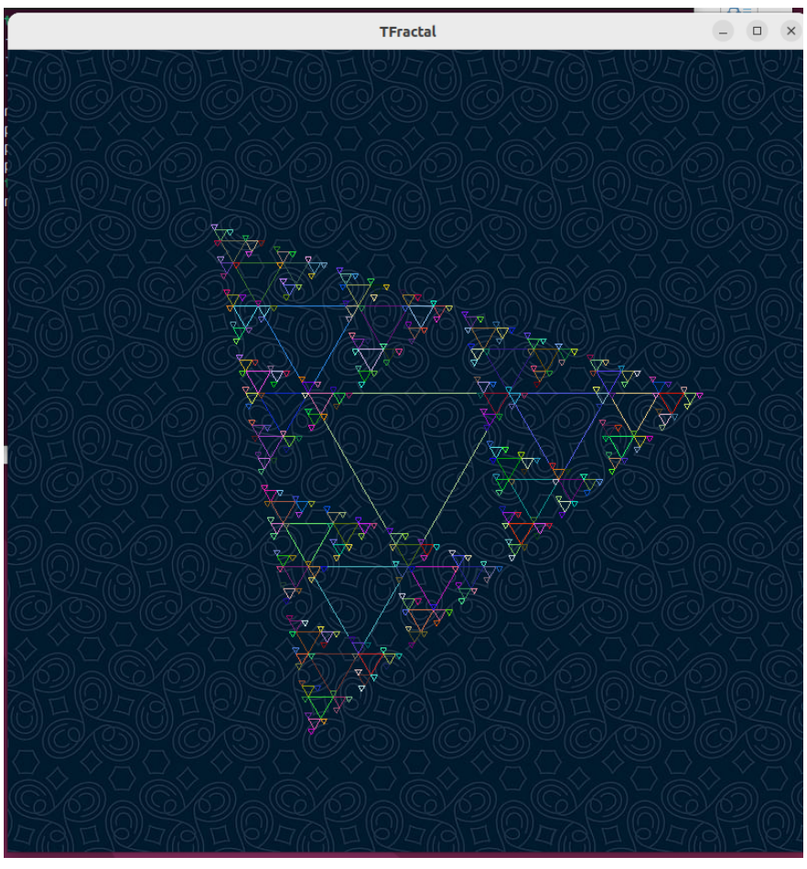

Figure 7: Triangle TFractal

5 PS3a: N-Body Simulation (Static)

5.1 Discussion:

We have been through our school education where we learned about solar systems. We wondered how these planets are implemented as an animation to display on the screen. Well to answer these thoughts, such assignments are created to know how to create animation. At the first part we will create a static animation of the solar system and in the second part we will simulate the static bodies by using physics.

The **PS3a** asks us to load and display static images of the planets. We are using two classes Universe and CelestialBody classes. The class CelestialBody is used to show any planet rendered in the universe, example: Our planet Earth. Universe class has a vector of CelestialBody and has a function to create the entire universe by accessing the vector.

We use the provided planets.txt file for this assignment: The values are as follows:
Inputs: 5
Radius: 2.50e+11

x-axis	y-axis	velocity of x-axis	velocity of y-axis	Mass	filename
1.4960e+1	0.0000e+00	0.0000e+00	2.9800e+04	5.9740e+24	earth.gif
2.2790e+11	0.0000e+00	0.0000e+00	2.4100e+04	6.4190e+2	mars.gif
5.7900e+10	0.0000e+00	0.0000e+00	4.7900e+04	3.3020e+23	mercury.gif
0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	1.9890e+30	sun.gif
1.0820e+11	0.0000e+00	0.0000e+00	3.5000e+04	4.8690e+24	venus.gif

5.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I used istream to read the data from planets.txt and ostream as output.Displays the planets in an SFML window. Reads input of file using < operator. I did not use any smart pointers. Just used General Pointers. Simply added (space) key to exit from the window. used Address-operator for managing the addresses of variables and also the objects. I have not used any smart pointers.

istream code:

```
1       std::istream& operator>> (std::istream &data, CelestialBody &body) {
2
3 data >> body.x >> body.y >> body.vx >> body.vy >> body.mass >> body.filename
    ;
4
5 if (!body.image.loadFromFile(body.filename))
6     return data;
7 body.calculatePosition();
8 body.texture.loadFromImage(body.image);
9 body.sprite.setTexture(body.texture);
10 body.sprite.setPosition(body.rx, body.ry);
11 return data;
12
13 }
14
```

ostream code:

```
1       std::ostream& operator<< (std::ostream &outstream, CelestialBody &
    body){
2 outstream << "CelestialBody :  x:" << body.x << " y:";
3 outstream << body.y << " velx:" << body.vx << " vely:";
4 outstream << body.vy << " mass:" << body.mass << "";
5 outstream << " ~~x:" << body.rx << "  --- " << body.ry << "~~" << " " <<
    body.filename << std::endl;
6 return outstream;
```

7

5.3 Images used:

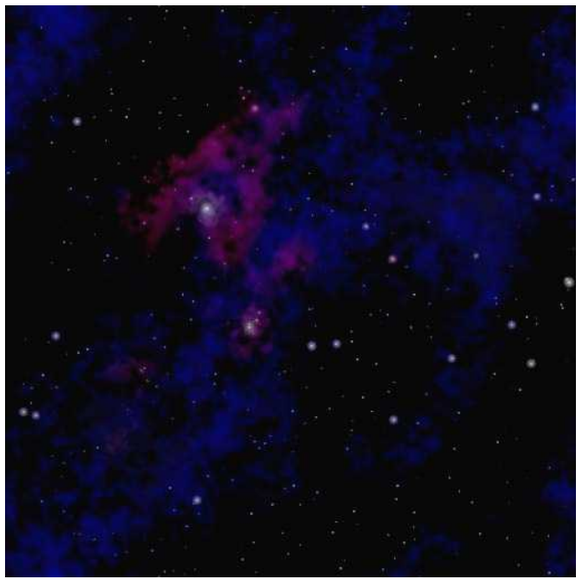

Figure 8: Background Image

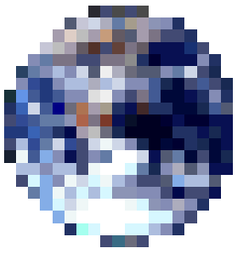

Figure 9: Background Image

Figure 10: Background Image

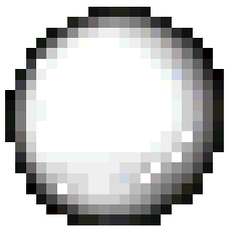

Figure 11: Background Image

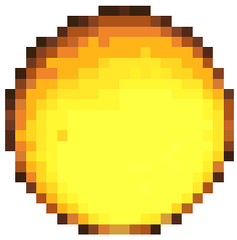

Figure 12: Background Image

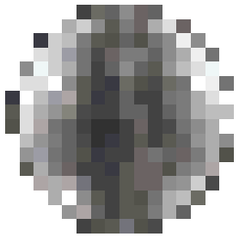

Figure 13: Background Image

5.4 What I accomplished :

I accomplished creating a solar system by the given input data. I created my first animated solar system. I am amazed that how physics can be implemented in the code.

5.5 What I already knew :

I knew how to add the background and use of the draw. I also knew how to take input from the file and display. I knew how to create header files and implement them into the cpp files.

5.6 What I learned :

I learnt how to use physics in creating the solar model. I understood much use of the istream and ostream in this assignment. I understood the calculations required for the placement of the CelestialBodies. I was able to learn how to use unitary methods and also vectors in a good level. Honestly, I was weak in operator overloading but by doing this project I am able to grip on it.

5.7 Challenges :

I was unable to use smart pointers and few algorithm classes, so I kind of felt a difficulty in that.

5.8 Acknowledgments :

Links :

- `https://www.sfml-dev.org/tutorials/2.5/graphics-sprite.php`
- `https://icarus.cs.weber.edu/~dab/cs1410/textbook/11.Operators/io_overload.html`
- `https://www.cplusplus.com/reference/vector/vector/`
- `https://www.sfml-dev.org/documentation/2.5.1/classsf_1_1Drawable.php`
- `https://stackoverflow.com/questions/34458791/making-custom-types-drawable-with-sfml`

5.9 Codebase

Makefile

This Makefile has no Linting as the program does not have any lints.

```
Compiler= g++
Flags= -Wall -Werror --std=c++14 -ansi -pedantic
lsfml_flags= -lsfml-graphics -lsfml-window -lsfml-system


all: NBody

NBody:  main.o CelestialBody.o Universe.o
    $(Compiler) main.o Universe.o CelestialBody.o -o NBody $(lsfml_flags)


main.o: main.cpp CelestialBody.hpp Universe.hpp
    $(Compiler) -c main.cpp CelestialBody.hpp Universe.hpp $(Flags)

CelestialBody.o: CelestialBody.cpp CelestialBody.hpp
    $(Compiler) -c CelestialBody.cpp CelestialBody.hpp $(Flags)

Universe.o: Universe.cpp Universe.hpp
    $(Compiler) -c Universe.cpp Universe.hpp $(Flags)

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

main.cpp

The main file is important as it is base for creating a window and displaying the CelestialBodies on it.

```
#include <iostream>
#include "CelestialBody.hpp"
#include "Universe.hpp"

int height = 512, width = 512;

int main() {
sf::RenderWindow window(sf::VideoMode(width, height), "ps3a");
window.setFramerateLimit(60);
sf::RectangleShape rect(sf::Vector2f(20, 20));
rect.setPosition(100, 100);
std::string num, rad;
std::cin >> num;
std::cin >> rad;
sf::Image img;
img.loadFromFile("starfield.jpg");
sf::Texture tex;
tex.loadFromImage(img);
sf::Sprite s;
s.setTexture(tex);
Universe universe(atoi(num.c_str()), atof(rad.c_str()));
std::cout << "There are " << num << " planets..." << std::endl;
for(int i = 0; i < universe.numberOfPlanets; i++) {
    CelestialBody* body = new CelestialBody();
    body -> screenh = height;
    body -> screenw = width;
    body -> setUniverseRadius(universe.getRadius());
    std::cin >> *body;universe.bodies.push_back(*body);
```

```
    std::cout << *body;
    }
while (window.isOpen()){
    sf::Event event;
    while (window.pollEvent(event)){
        if (event.type == sf::Event::Closed)
            window.close();
        if(sf::Keyboard::isKeyPressed(sf::Keyboard::Space))
            window.close();
}
    window.clear();
    window.draw(s);
    std::vector<CelestialBody>::iterator itr;
for(itr = universe.bodies.begin(); itr != universe.bodies.end(); itr++)
      window.draw(*itr);

    window.display();
    }
    return 0;
}
```

CelestialBody.hpp

The CelestialBody.hpp contains the initializations of istream,ostream and variable and methods for the creation of the Nbodies.

```
#ifndef CELESTIALBODY_H
#define CELESTIALBODY_H

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

class CelestialBody: public sf::Drawable {
public:
CelestialBody();
CelestialBody(double x, double y, double vx, double vy,
double mass, double r, std::string filename);
int screenw, screenh;
float univRadius;
void setRadius(double new_rad);
void setUniverseRadius(float);
friend std::istream& operator>> (std::istream &input, CelestialBody &cBody);
friend std::ostream& operator<< (std::ostream &output, CelestialBody &cBody)
    ;
private:
void draw (sf::RenderTarget &target, sf::RenderStates states) const;
void calculatePosition(void);
double x, y, rx, ry, vx, vy, mass, radius;
std::string filename;
sf::Image image;
sf::Sprite sprite;
sf::Texture texture;
};


#endif
```

CelestialBody.cpp

This is the file where the Celestial body data are taken by the istream and provide an accurate calculation for the position of them as well as provide the data of each CelestialBody on the terminal.

```
1  #include "CelestialBody.hpp"
2  #include <iostream>
3
4  CelestialBody::CelestialBody(double x, double y, double vx, double vy,
5  double mass, double r, std::string filename) {
6  CelestialBody::x = x;
7  CelestialBody::y = y;
8  CelestialBody::vx = vx;
9  CelestialBody::vy = vy;
10 CelestialBody::filename = filename;
11 CelestialBody::mass = mass;
12 CelestialBody::radius = r;
13 }
14
15 CelestialBody::CelestialBody() {
16     return;
17 }
18
19 std::istream& operator>> (std::istream &data, CelestialBody &body) {
20
21 data >> body.x >> body.y >> body.vx >> body.vy >> body.mass >> body.filename
       ;
22
23 if (!body.image.loadFromFile(body.filename))
24     return data;
25 body.calculatePosition();
26 body.texture.loadFromImage(body.image);
27 body.sprite.setTexture(body.texture);
28 body.sprite.setPosition(body.rx, body.ry);
29 return data;
30
31 }
32
33
34 void CelestialBody::draw (sf::RenderTarget &target, sf::RenderStates states)
       const {
35
36     target.draw(sprite);
37
38 }
39
40 void CelestialBody::setUniverseRadius(float urad) {
41     univRadius = urad;
42 }
43
44 std::ostream& operator<< (std::ostream &outstream, CelestialBody &body){
45 outstream << "CelestialBody :  x:" << body.x << " y:";
46 outstream << body.y << " velx:" << body.vx << " vely:";
47 outstream << body.vy << " mass:" << body.mass << "";
48 outstream << " ~~x:" << body.rx << "  --- " << body.ry << "~~" << " " <<
      body.filename << std::endl;
49 return outstream;
50 }
51
52 void CelestialBody::calculatePosition() {
```

```
    rx = ((x/univRadius) * (screenw / 2)) + (screenw/2);
    ry = ((y/univRadius) * (screenh / 2)) + (screenh/2);

}
```

Universe.hpp

This header file contains the declarations of few important variables such as numberOFplanets,radius and also methods.

```
#include "CelestialBody.hpp"

class Universe {
public:
int numberOfPlanets;
void setRadius(float radius);
void setNumPlanets(int n);
float getRadius();
int getNumPlanets();
std::vector<CelestialBody> getBodies();
void addBody(CelestialBody* body);
Universe(int, float);
std::vector<CelestialBody> bodies;
private:
float radius;
};
```

Universe.cpp

This file sets radius and adds Body to window

```
#include "Universe.hpp"
float Universe::getRadius() {
    return radius;
}
std::vector<CelestialBody> Universe::getBodies() {
    return bodies;
}
void Universe::setRadius(float radius) {
    Universe::radius = radius;
}
void Universe::addBody(CelestialBody *body) {
    bodies.push_back(*body);
}
Universe::Universe(int n, float r){
    numberOfPlanets = n;
    radius = r;
}
```

5.10 Output:

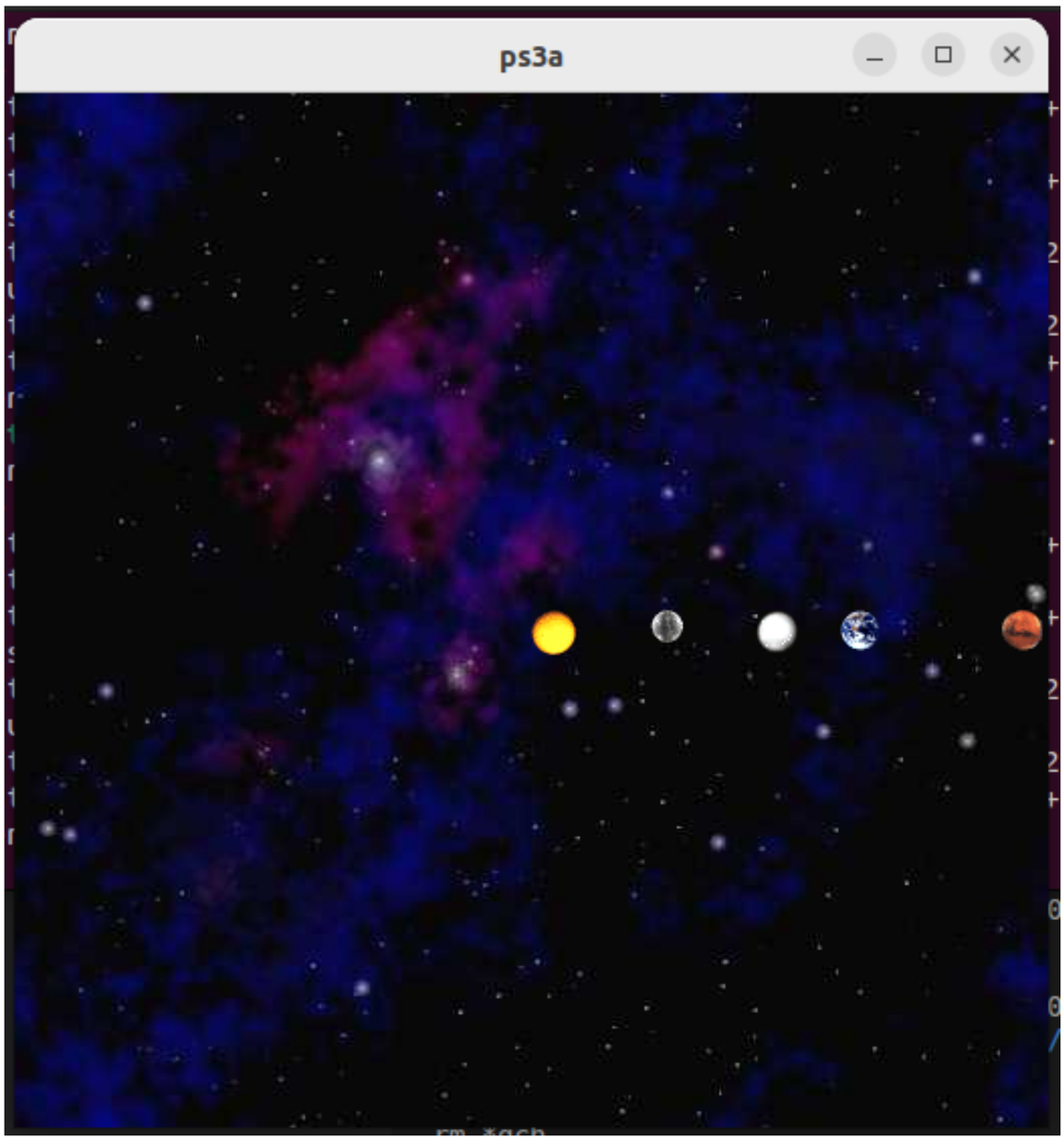

Figure 14: NBody Simulation (static)

6 PS3b: N-Body Simulation

6.1 Discussion:

In this project we are simulating the created solar system in **ps3a** by using physics in it. Basic physics such as net force, Acceleration , Pair-wise forces and their formulas. According to the given simulating time and time step the Planets Revolve around the sun.
By default the detail code.
./NBody 157788000.0 25000.0 < planets.txt

6.2 Key algorithms, Data structures and OO Designs used in this Assignment:

The smart pointers that I have implemented in this project and improved the previous ps3a code. I have studied the notes of class as well as few physics site to actaully get the correct physics formula to run. I used smart pointers to mange the memory locations of variables as well as objects in CelestialBody and Universe classes. I also used the classic vector for the memory management in the project. There was much utilizations of the Physic laws for creating a simulation of this project.

6.3 Images used:

The images are already shown in PS3a Sections, To browse them you can click on the following *Figure Numbers:*

- **The Background is Figure 8**
- **The Earth is Figure 9**
- **The Mars is Figure 10**
- **The Venus is Figure 11**
- **The Sun is Figure 12**
- **The Mercury is Figure 13**

6.4 What I accomplished :

I accomplished creating a full animated solar system using C++ SFML library. I have also used audio for the window for (extra point).

6.5 What I already knew :

I already knew how to read data from the given input file. I was well aware of the SFML library to draw bodies and put the background to it.

6.6 What I learned :

I learnt how physics can be the part of the computer field. It enlightened me in using different the formulas of physics in different aspects of the functions in this assignment. By doing this assignment's extra credit, I got to know how to create different solar system as well how to implement audio into the window of the SFMl.

6.7 Challenges :

At starting of the simulation i have a glitch in my sound, I Guess that the convertion of mp3 to wav was not good enough, or may be other reason. Linting became challenging for me in this assingment.

6.8 Codebase

Makefile

This Makefile has no Linting as the program does not have any lints.

```
Compiler=g++
LSFML=-lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system
FLAGS=-g -Wall -ansi -pedantic -std=c++14
Ofile=main.o Universe.o CelestialBody.o
lint: cpplint ps2b
all: $(Ofile)
    $(Compiler) $(FLAGS) -o NBody $(Ofile) $(LSFML)
 main.o: main.cpp
    $(Compiler) $(FLAGS) -c main.cpp
 Universe.o: Universe.cpp Universe.hpp
    $(Compiler) $(FLAGS) -c Universe.cpp
 CelestialBody.o: CelestialBody.cpp CelestialBody.hpp
    $(Compiler) $(FLAGS) -c CelestialBody.cpp
 clean:
    rm *.o
    rm NBody
```

main.cpp

The main file is important as it is base for creating a window and displaying the CelestialBodies on it.In addition, in this part-b I have added more functions to make a simulation of these CelestialBodies. Moreover, I added audio and Time Elapsed to the SFML window.

```
#include <iostream>
#include <fstream>
#include "Universe.hpp"
#include "CelestialBody.hpp"
#include <string>
#include <exception>
#include <vector>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#define widthwin 512
#define fpswin 60
#define heightwin 512
using namespace std;
 int main(int argc, char* argv[])
{
 if(argc < 2)
 {
   cout << "wrong syntax use the code provided in readme" << endl;
   return 1;
 }
float mtime, ctime;
float time_change;
ctime = 0;
mtime = stod(argv[1]);
time_change = stod(argv[2]);
sf::Vector2f centerUniverse{widthwin / 2, heightwin / 2};
Universe solarSystem(centerUniverse);
cin >> solarSystem;
sf::RenderWindow window(sf::VideoMode(widthwin, heightwin), "ps3b");
sf::Texture spaceTextures;
sf::Font sFont;
sf::SoundBuffer buffer;
window.setFramerateLimit(fpswin);
if(!spaceTextures.loadFromFile("starfield.jpg")) {
```

```
throw FileNotFoundException();
cout << "No background image selected" << endl;
        }
sf::Sprite spaceBackground(spaceTextures);
spaceBackground.setScale(static_cast<float>(widthwin) / spaceTextures.
   getSize().x ,static_cast<float>(heightwin) / spaceTextures.getSize().y);
 if(!sFont.loadFromFile("font.ttf"))
{throw FileNotFoundException();       }
if(!buffer.loadFromFile("muj.wav"))
{throw FileNotFoundException();}
sf::Sound sound(buffer);
sf::Text timeElapsed{"Time Elapsed: " + to_string(ctime), sFont};
timeElapsed.setPosition(0,0);
timeElapsed.setCharacterSize(20);
timeElapsed.setOutlineColor(sf::Color::White);
while(window.isOpen())
{    sf::Event event;
while(window.pollEvent(event))
{sound.play();
if (event.type == sf::Event::Closed ||
sf::Keyboard::isKeyPressed(sf::Keyboard::Escape))
{ofstream result;
result.open("output.txt");
result << solarSystem;
result.close();
window.close();
 }
}
if (ctime < mtime) {
window.clear();
 window.draw(spaceBackground);
for ( const auto &p : solarSystem.getBodies() )
 window.draw(*p);
window.draw(timeElapsed);
window.display();
solarSystem.step(time_change);
ctime += time_change;
timeElapsed.setString("Time Elapsed: " + to_string(ctime) + "Seconds" ); }
else
window.setFramerateLimit(60);
}
 return 0;
 }
```

CelestialBody.hpp

The CelestialBody.hpp contains the initializations of istream,ostream and variable and methods for the creation of the Nbodies. In this part-b, I have also added File eXceptions and more functions as to achieve the simulation of the planets.

```
 #ifndef CELESTIAL_BODY_HPP_
 #define CELESTIAL_BODY_HPP_
 #include <iostream>
 #include <string>
 #include <exception>
 #include <iomanip>
 #include <SFML/Graphics.hpp>
 using namespace std;
 struct FileNotFoundException : public exception {
        const char * what() const noexcept {
        return "Can't find file!";
```

```
        }
};
 class CelestialBody : public sf::Drawable {
 public:
        CelestialBody() {}
        CelestialBody(sf::Vector2f iPosition, sf::Vector2f iVelocity,float
   iMass, std::string iImageRef);
        inline float getMass() { return mass; }
        inline sf::Vector2f getPosition() { return position; }
        inline void setPosition(sf::Vector2f nPosition) { position =
   nPosition; }
        inline sf::Vector2f getVelocity() { return velocity; }
        inline void setVelocity(sf::Vector2f nVelocity) { velocity =
   nVelocity; }
        static void createUniverse(sf::Vector2f iCenterUniverse, float
   iRadiusUniverse);
        void spriteUpdate();
        ~CelestialBody() { delete sprite_textures; };
        friend ostream& operator<<(ostream &out, const CelestialBody& cb);
 private:
        sf::Vector2f position;
        sf::Vector2f velocity;
        float mass;
        string imageRef;
        sf::Sprite sprite;
        sf::Texture* sprite_textures;
        virtual void draw(sf::RenderTarget& rendTarget,
        sf::RenderStates rendStates) const;
        static sf::Vector2f centerUniverse;
        static float radiusUniverse;
 };
 #endif
```

CelestialBody.cpp

This is the file where the Celestial body data are taken by the istream and provide an accurate calculation for the position of them as well as provide the data of each CelestialBody on the terminal.

```
 #include "CelestialBody.hpp"
 using namespace std;
 sf::Vector2f CelestialBody::centerUniverse{0,0};
 float CelestialBody::radiusUniverse = 0;
 CelestialBody::CelestialBody(sf::Vector2f iPosition, sf::Vector2f iVelocity
   , float iMass, string iImageRef) {
mass = iMass;
position = iPosition;
velocity = iVelocity;
imageRef = iImageRef;sprite_textures = new sf::Texture;
if(!sprite_textures->loadFromFile(imageRef)) {
throw FileNotFoundException();
}
sprite = sf::Sprite(*sprite_textures);
sprite.setOrigin(sprite_textures->getSize().x / 2,
sprite_textures->getSize().y / 2);
spriteUpdate();
 }
 void CelestialBody::createUniverse(sf::Vector2f iCenterUniverse,
 float iRadiusUniverse) {
centerUniverse = iCenterUniverse;
radiusUniverse = iRadiusUniverse;
 }
```

```
23  void CelestialBody::spriteUpdate() {
24 sf::Vector2f sprite_position{position.x / radiusUniverse *centerUniverse.x +
       centerUniverse.x,position.y / radiusUniverse * centerUniverse.y +
     centerUniverse.y};
25 sprite.setPosition(sprite_position);
26  }
27  void CelestialBody::draw(sf::RenderTarget& rendTarget, sf::RenderStates
     rendStates) const {
28 rendTarget.draw(sprite, rendStates);
29  }
30  ostream& operator<<(ostream &out, const CelestialBody& cb) {\
31 out.setf(ios_base::scientific);
32 out << setprecision(4) << left;
33 out << setw(12) << cb.position.x << setw(12) << cb.position.y << setw(12)<<
     cb.velocity.x << setw(12) << cb.velocity.y << setw(12)<< cb.mass <<
     right << setw(12) << cb.imageRef;
34 out.unsetf(ios_base::scientific);
35          return out;
36  }
```

Universe.hpp

This header file contains the declarations of few important variables such as numberOFplanets,radius and also methods. In part-b, we added istream and ostream to it.

```
1  #ifndef UNIVERSE_HPP_
2  #define UNIVERSE_HPP_
3  #include <iostream>
4  #include <cmath>
5  #include "CelestialBody.hpp"
6  #include <SFML/Graphics.hpp>
7  #include <string>
8  #include <vector>
9  using namespace std;
10 class Universe {
11 public:
12          Universe() {}
13          Universe(sf::Vector2f iCenter) : center(iCenter) {}
14          inline const vector<unique_ptr<CelestialBody>>&getBodies() const {
     return celBodies; }
15          friend istream& operator>>(istream& in, Universe& u);
16          friend ostream& operator<<(ostream& out, const Universe& u);
17          void step(float seconds);
18 private:
19          sf::Vector2f center;
20          float radius;
21          vector<unique_ptr<CelestialBody>> celBodies;
22 };
23 #endif
```

Universe.cpp

The universe file in part-b has also played a major role in case of calculating scaleForce, Netforce and organizing the Bodies while they are revolving along the orbit provided.

```
1  #include "Universe.hpp"
2  using namespace std;
3  std::istream& operator>>(std::istream& in, Universe& u)
4  {
5          int numBodies;
6          in >> numBodies >> u.radius;
7          CelestialBody::createUniverse(u.center, u.radius);
```

```
8         for(int i = 0; i < numBodies; i++) {
9                 float xPosition, yPosition, xVelocity, yVelocity, mass;
10                string imageRef;
11                in >> xPosition >> yPosition >> xVelocity >> yVelocity >>
    mass >>imageRef;
12                u.celBodies.push_back(
13                make_unique<CelestialBody>(sf::Vector2f(xPosition,
    yPosition),
14                        sf::Vector2f(xVelocity, yVelocity),mass, imageRef))
    ;
15        }
16        return in;
17 }
18 ostream& operator<<(ostream& out, const Universe& u)
19 {
20        out << u.celBodies.size() << endl;
21        out << u.radius << endl;
22        for(const auto &b : u.celBodies) out << (*b) << endl;
23        return out;
24 }
25 void Universe::step(float seconds)
26 {
27        auto getNetForce = [&](size_t planetIndex) -> sf::Vector2f
28        {
29                sf::Vector2f netForce;
30                for(size_t i = 0; i < celBodies.size(); i++)
31                {
32                        if(i != planetIndex)
33                        {
34                                sf::Vector2f position_change = celBodies[i
    ]->getPosition() - celBodies[planetIndex]->getPosition();
35                                float planetDistance = hypot(
    position_change.x, position_change.y);
36                                float scaleForce =(6.67430e-11 * celBodies[
    planetIndex]->getMass() * celBodies[i]->getMass()) /pow(planetDistance,
    2);
37                                sf::Vector2f force_xy =
38                                {
39                               scaleForce * (position_change.x /
    planetDistance),
40                               scaleForce * (position_change.y /
    planetDistance)
41                                };
42                                netForce += force_xy;
43                        }
44                }
45                return netForce;
46        };
47        vector<sf::Vector2f> rPosition, rVelocity;
48        for(size_t i = 0; i < celBodies.size(); i++)
49        {
50                sf::Vector2f netForce = getNetForce(i);
51                sf::Vector2f planetAccel =
52                {
53                        netForce.x / celBodies[i]->getMass(),
54                        netForce.y / celBodies[i]->getMass()
55                };
56                sf::Vector2f velocity =
57                {
```

```
58                          celBodies[i]->getVelocity().x + seconds *
    planetAccel.x,
59                           celBodies[i]->getVelocity().y + seconds *
    planetAccel.y
60                  };
61                  sf::Vector2f position =
62                  {
63                          celBodies[i]->getPosition().x + seconds * velocity.
    x,
64                          celBodies[i]->getPosition().y + seconds * velocity.
    y
65                  };
66                  rVelocity.push_back(velocity);
67                  rPosition.push_back(position);
68          }
69          for(size_t i = 0; i < celBodies.size(); i++) {
70                  celBodies[i]->setVelocity(rVelocity[i]);
71                  celBodies[i]->setPosition(rPosition[i]);
72                  celBodies[i]->spriteUpdate();
73          }
74  }
```

6.9 Output:

To see the static Output got to Figure: 14

6.10 Acknowledgements:

- https://docs.microsoft.com/en-us/cpp/cpp/smart-pointers-modern-cpp?view=msvc-170
- https://www.geeksforgeeks.org/vector-in-cpp-stl/
- https://study.com/learn/lesson/net-force-formula-examples-how-find.html
- https://www.sfml-dev.org/tutorials/2.5/audio-sounds.php

7 PS4a: CircularBuffer

7.1 Discussion:

The ps4a assignment creates a circular buffer which will be used for the follow up assignment ps4b. In this assignment, we create basically a circular queue, where the head and tail will be connected and follows the same mechanism as queue i.e, FIFO (First In First Out) Mechanism.

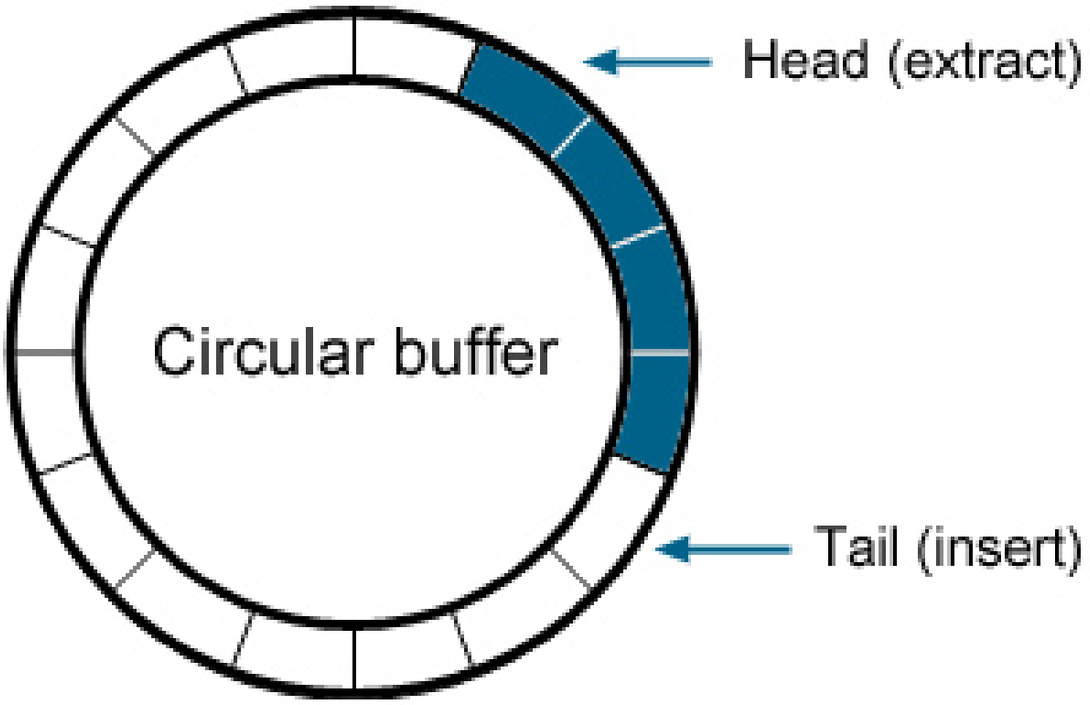

Figure 15: Circular Buffer

7.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I make use of Vectors in standard manner rather than the functions as I found it amusing and easy to understand theoratically, Where I used basic rules such as initial head, tail. I used the default format of the pdf to create the template version and also added few functions such as print etc. I used exceptions to inform the provided arised exceptions.

7.3 What I accomplished :

I have created a CircularBuffer using the vector. It works properly and gives the output too in the main function. I have used the templates so therefore there is no need for .cpp file.

7.4 What I already knew :

I knew how to implement template. I was aware of the CircularBuffer concept as I did a Circular buffer program without using templates in my Data Structure Class.

7.5 What I learned :

I learnt how to implement vector and the template in an efficient way.

7.6 Challenges :

To implement template throughout the code as well as implementing in the test file was quite a challenge for me.

7.7 Codebase

Makefile:

This Makefile has linting included.

```
CC= g++
CPP_FLAGS= -Wall -Werror -pedantic --std=c++14 -g
lint: ps4a
    cpplint --filter=-runtime/references,-build/c++11 --root=. *.h *.cpp
all: ps4a main.o
ps4a: test.o
    $(CC) test.o -o ps4a -lboost_unit_test_framework
test.o: test.cpp CircularBuffer.h
    $(CC) -c test.cpp CircularBuffer.h $(CPP_FLAGS)
main.o: main.cpp CircularBuffer.h
    $(CC) -o a main.cpp CircularBuffer.h  $(CPP_FLAGS)
clean:
    rm *.o
    rm *.gch
    rm ps4a
    rm a
```

CircularBuffer.h:

This file is the important file as it contains all the methods for the CircularBuffer to run. It does not have a cpp file as I have utilized the template.

```
/*Copyright 2022 Aniketh Rai*/
#ifndef CIRCULARBUFFER_H_
#define CIRCULARBUFFER_H_
#include <stdlib.h>
#include <stdint.h>
#include <vector>
#include <exception>
#include <stdexcept>
#include <iostream>


template <class T>
class CircularBuffer {
 public:
explicit CircularBuffer(size_t capacity) {
    if (capacity < 1) {            // checking for condn where size must be
    > 0
  throw
    std::invalid_argument("CircularBuffer constructor: capacity > 0");}
Max_size = capacity;
buffer.resize(Max_size);
head = 0;
tail = 0;
len = 0;
}
void prettyPrint() {
int f = 0;
int back = head;
std::cout << "Buffer: ";
while (f < len) {
if (back >= Max_size) {back = 0;}
std::cout << buffer[back] << " ";
back++;
f++;}
std::cout << std::endl;}
bool isEmpty() {
```

```
auto lamda = [](int len){ return len == 0;  };
return lamda(len);
}
bool isFull() {
return (len == Max_size) ? true : false; }  // use of ternary operator
void enqueue(T item) {
if (isFull()) {
throw
std::runtime_error("enqueue: can't enqueue to a full ring");
}
if (tail >= Max_size) {
tail = 0;}
buffer.at(tail) = item;
tail++;
len++;
}
inline T dequeue() {
    if (isEmpty()) {
throw
  std::runtime_error("dequeue: can't dequeue an empty ring");
  }
int first = buffer.at(head);
buffer.at(head) = 0;
head++;
len--;
if (head >= Max_size) {
head = 0;}
return first;
}
inline T peek() {
    if (isEmpty()) {
  throw
  std::runtime_error("cant peek");}
return buffer.at(head);
}
size_t size() {
    return len;
}

 private:
std::vector<int> buffer;
int len;
int Max_size;
int head;
int tail;
};
#endif  // CIRCULARBUFFER_H_
```

main.cpp:

This file is just for the output of the sample input and methods of the Circular-Buffer

```
/*Copyright 2022 Aniketh Rai*/
#include "CircularBuffer.h"
int main() {
CircularBuffer<int> obj(10);
std::cout << "Entering 10 elements" << std::endl;
std::cout << "Entering 89 ->" << std::endl;
obj.enqueue(89);
std::cout << "Entering 99 ->" << std::endl;
obj.enqueue(99);
```

```
std::cout << "Entering 29 ->" << std::endl;
obj.enqueue(29);
std::cout << "Entering 19 ->" << std::endl;
obj.enqueue(19);
std::cout << "Entering 59 ->" << std::endl;
obj.enqueue(59);
std::cout << "Entering 42 ->" << std::endl;
obj.enqueue(42);
std::cout << "Entering 15 ->" << std::endl;
obj.enqueue(15);
std::cout << "Entering 12 ->" << std::endl;
obj.enqueue(12);
std::cout << "Entering 434 ->" << std::endl;
obj.enqueue(434);
std::cout << "Entering 55 ->" << std::endl;
obj.enqueue(55);
// printing the buffer
obj.prettyPrint();
// deleting 5 ele and printing then deleting 2
for (int i=0 ; i < 5 ; i++) {
    int f = obj.dequeue();
    std::cout << f << "is deleted" << std::endl;
}
// printing the buffer
obj.prettyPrint();
// deleting 2 ele
for (int i=0 ; i < 2 ; i++) {
    int f = obj.dequeue();
    std::cout << f << "is deleted" << std::endl;;
}
// using peek here
int x = obj.peek();
std::cout << "peek element is " << x;
// checking for isEmpty
bool t = obj.isEmpty();
std::cout << "isEmpty() = " << t << std::endl;
// checking for isFull
bool t2 = obj.isFull();
std::cout << "isFull() = " << t2 << std::endl;
}
```

test.cpp:

The test file tests for the Exceptions as well as all the functions of the CircularBuffer as you can see in the output section.

```
/*Copyright 2022 Aniketh Rai*/
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include "CircularBuffer.h"

BOOST_AUTO_TEST_CASE(constructor) {
BOOST_REQUIRE_THROW(CircularBuffer<size_t>(0), std::exception);
BOOST_REQUIRE_THROW(CircularBuffer<size_t>(0), std::invalid_argument);
BOOST_REQUIRE_NO_THROW(CircularBuffer<size_t>(20));
}
BOOST_AUTO_TEST_CASE(size) {
CircularBuffer<int> testBuffer(10);
BOOST_REQUIRE(testBuffer.size() == 0);
testBuffer.enqueue(5);
testBuffer.enqueue(5);
BOOST_REQUIRE(testBuffer.size() == 2);
testBuffer.dequeue();
BOOST_REQUIRE(testBuffer.size() == 1);
testBuffer.dequeue();
BOOST_REQUIRE(testBuffer.size() == 0);
testBuffer.enqueue(5);
testBuffer.dequeue();
BOOST_REQUIRE(testBuffer.size() == 0);
}
BOOST_AUTO_TEST_CASE(isEmpty) {
CircularBuffer<int> testBuffer(5);
BOOST_REQUIRE(testBuffer.isEmpty() == true);
testBuffer.enqueue(5);
BOOST_REQUIRE(testBuffer.isEmpty() == false);
}
BOOST_AUTO_TEST_CASE(isFull) {
CircularBuffer<int> testBuffer(1);
BOOST_REQUIRE(testBuffer.isFull() == false);
testBuffer.enqueue(5);
BOOST_REQUIRE(testBuffer.isFull() == true);
}
BOOST_AUTO_TEST_CASE(Enqueue) {
CircularBuffer<int> testBuffer(1);
BOOST_REQUIRE_NO_THROW(testBuffer.enqueue(1));
BOOST_REQUIRE(testBuffer.dequeue() == 1);
testBuffer.enqueue(1);
BOOST_REQUIRE_THROW(testBuffer.enqueue(1), std::runtime_error);
}
BOOST_AUTO_TEST_CASE(Dequeue) {
CircularBuffer<int> testBuffer(5);
testBuffer.enqueue(56);
testBuffer.enqueue(32);
testBuffer.enqueue(23);
testBuffer.enqueue(89);
testBuffer.enqueue(99);
BOOST_REQUIRE(testBuffer.dequeue() == 56);
BOOST_REQUIRE(testBuffer.dequeue() == 32);
BOOST_REQUIRE(testBuffer.dequeue() == 23);
BOOST_REQUIRE(testBuffer.dequeue() == 89);
BOOST_REQUIRE(testBuffer.dequeue() == 99);
```

```
57 BOOST_REQUIRE_THROW(testBuffer.dequeue(), std::runtime_error);
58 }
59 BOOST_AUTO_TEST_CASE(peek) {
60 CircularBuffer<int> testBuffer(1);
61 BOOST_REQUIRE_THROW(testBuffer.peek(), std::runtime_error);
62 testBuffer.enqueue(1);
63 BOOST_REQUIRE(testBuffer.peek() == 1);
64 }
```

7.8 Output :

```
aniketh@aniketh-VirtualBox:~/Desktop/ps4a$ ./ps4a
Running 7 test cases...

*** No errors detected
aniketh@aniketh-VirtualBox:~/Desktop/ps4a$ ./a
Entering 10 elements
Entering 89 ->
Entering 99 ->
Entering 29 ->
Entering 19 ->
Entering 59 ->
Entering 42 ->
Entering 15 ->
Entering 12 ->
Entering 434 ->
Entering 55 ->
Buffer: 89 99 29 19 59 42 15 12 434 55
89is deleted
99is deleted
29is deleted
19is deleted
59is deleted
Buffer: 42 15 12 434 55
42is deleted
15is deleted
peek element is 12isEmpty() = 0
isFull() = 0
aniketh@aniketh-VirtualBox:~/Desktop/ps4a$
```

Figure 16: PS4a Output in Terminal

8 PS4b: StringSound

8.1 Discussion:

This Project produces sound using the SFML library <Audio>, Where we produces notes by using our keys. We utilize the ps4a CircularBuffer in this project. Basic motto of this project is to create a simulation of guitar plucking, but as I wanted to do extra credit, I made a Piano simulation note, It is not accurate but close. Main thing in this project is using the Karplus-Strong algorithm.

8.2 Key algorithms, Data structures and OO Designs used in this Assignment:

The Vector is used for the samples and also creating a sound buffer and holding the sounds bpm. Basic switch method is used for taking the inputs of keys and sending them to calculate the frequency and return with a audio. There are much use of new functions in creating this assignment.

8.3 Images used:

Figure 17: Jazz Background

8.4 What I accomplished :

I accomplished producing piano sounds by altering the frequency. It is an amazing project to create our own instrument using C++.

8.5 What I already knew :

I knew how to implement few math functions as well as basics of the SFML.

8.6 What I learned :

I learnt how the frequency can be use to create different instruments. I understood how to take the sample inputs from keyboard and produce sound.

8.7 Challenges :

Finding out the right frequency and implementing of the samples by reading from the keys was difficult challenge to me.

8.8 Codebase

Makefile

This Makefile contains the lint and is extension of the PS4a Makefile.

```
compiler= g++
flags= -g -std=c++11
SFMLFlags= -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio

all:    lint KSGuitarSim main.o

lint:
    cpplint --filter=-runtime/references,-build/c++11 --root=. *.h *.cpp

KSGuitarSim: CircularBuffer.h StringSound.o KSGuitarSim.o
    $(compiler) KSGuitarSim.o CircularBuffer.h StringSound.o -o KSGuitarSim
   $(SFMLFlags)

KSGuitarSim.o: KSGuitarSim.cpp
    $(compiler) -c KSGuitarSim.cpp $(flags)

StringSound.o: StringSound.h StringSound.cpp
    $(compiler) -c StringSound.h StringSound.cpp $(flags)

test.o: test.cpp CircularBuffer.h
    $(compiler) -c test.cpp CircularBuffer.h $(flags)

main.o: CircularBuffer.h main.cpp
    $(compiler) $(flags) -o a CircularBuffer.h main.cpp

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

KSGuitarSim.cpp

This is the main file where the Samples are taken and window is drawn as well as where takes inputs from keyboard and produces sound.

```
// Copyright 2022 Aniketh Rai
#include <math.h>
#include <limits.h>

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include <vector>

#include "StringSound.h"

#define CONCERT_A 440.0
#define SAMPLES_PER_SEC 44100

std::vector<sf::Int16> makeSamples(StringSound *gs) {
  std::vector<sf::Int16> samples;
  gs -> pluck();
  int duration = 8;  // seconds
  int i;
  for (i= 0; i < SAMPLES_PER_SEC * duration; i++) {
    gs -> tic();
```

```
23      samples.push_back(gs -> sample());
24    }
25
26    return samples;
27  }
28
29  int main() {
30    sf::RenderWindow window(sf::VideoMode(400 , 400),
31      "PS4B");
32    window.setFramerateLimit(60);
33
34    sf::Image img;
35    img.loadFromFile("backg.jpg");
36    sf::Texture t;
37    t.loadFromImage(img);
38    sf::Sprite s;
39    s.setTexture(t);
40
41    sf::Event event;
42    std::vector<sf::Int16> sample;
43    double freq = CONCERT_A;
44
45    std::string keyboardString = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
46    std::vector<sf::Sound> sounds(123);
47    std::vector<sf::SoundBuffer> buffers(keyboardString.size());
48
49    for (int i = 0; i < keyboardString.size(); i++) {
50      sounds[static_cast<int>(keyboardString[i])] = sf::Sound();
51      freq = CONCERT_A * pow(2, (i-24)/12.0);
52      StringSound gs = StringSound(freq);
53      sample = makeSamples(&gs);
54      if (!buffers[i].loadFromSamples(&sample[0],
55          sample.size(), 2, SAMPLES_PER_SEC))
56          throw std::runtime_error(
57            "sf::SoundBuffer: failed to l ad from samples.");
58      sounds[static_cast<int>(keyboardString[i])].setBuffer(buffers[i]);
59    }
60
61    while (window.isOpen()) {
62      while (window.pollEvent(event)) {
63        switch (event.type) {
64        case sf::Event::Closed:
65          window.close();
66          break;
67
68        case sf::Event::TextEntered:
69          sounds[event.text.unicode].play();
70
71          window.clear();
72          window.draw(s);
73          window.display();
74        }
75      }
76    }
77    return 0;
78  }
```

StringSound.h

The initialization of the StringSound methods and variables.

```
// Copyright 2022 Aniketh Rai
#ifndef STRINGSOUND_H_
#define STRINGSOUND_H_
#include <vector>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#include "CircularBuffer.h"


class StringSound{
 public:
    explicit StringSound(double frequency);
    explicit StringSound(std::vector<sf::Int16> init);
    StringSound(const StringSound &obj) {}
    ~StringSound();
    void pluck();
    void tic();
    sf::Int16 sample();
    int time();

 private:
    CircularBuffer<int> *buffer{};
    int _time;
    int blen;
};
#endif  // STRINGSOUND_H_
```

StringSound.cpp

The frequency is blent in and produces the sound required according to the frequency.

```
// Copyright 2022 Aniketh Rai
#include "StringSound.h"
#include <math.h>
#include <random>

#define SAMPLING_RATE 44100  // sample rate provided in class
#define DECAY_FACTOR 0.996
#define M_PI 3.14159265358979323846  // Math PI value

StringSound::StringSound(double frequency) {
blen = ceil(SAMPLING_RATE/frequency);
buffer = new CircularBuffer<int>(blen);
for (int i = 0; i < blen; i++)
buffer -> enqueue(0);
_time = 0;
}

void StringSound::pluck() {
buffer -> CircularBuffer::Empty();
for (int i = 1; i <= blen; i++) {
try {
buffer->enqueue(10000*sin(2 * M_PI * i/blen));
} catch(const char* msg) {
std::cout << msg << std::endl;
}
```

```
}
}
void StringSound::tic() {
int16_t val = 0;
try {
val = buffer -> dequeue();
} catch(const char* msg) {
std::cout << msg << std::endl;
}
int16_t ns = (val + buffer -> peek())/ 2 * DECAY_FACTOR;
buffer -> enqueue(int16_t(ns));
_time++;
}

int StringSound::time() {
return _time;
}
sf::Int16 StringSound::sample() {
try {
return buffer -> peek();
} catch(const char* msg) {
std::cout << msg << std::endl;
}
return 0;
}

StringSound::~StringSound() {}
```

8.9 Output:

Figure 18: PS4b Output Window

9 PS6: Random Writer

9.1 Discussion:

The Markov Model predicts the next characters in a sequence of k length words called kgrams. The RandWriter class takes a string and order k as input and maps each kgram in the string to each character following that kgram and it's frequency. This class can now generate strings of any given length, using the kgrams that have been seen before. The string's characters are based on the probabilities of the k-length strings that were stored in the hashmap.

I used the given structure in this assignment:

```
 class RandWriter {
public:

RandWriter(string text, int k);
int orderK() const; // Order k of Markov model
// Number of occurences of kgram in text
// Throw an exception if kgram is not length k
int freq(string kgram) const;
// Number of times that character c follows kgram
// if order=0, return num of times that char c appears
// (throw an exception if kgram is not of length k)
int freq(string kgram, char c) const;
// Random character following given kgram
// (throw an exception if kgram is not of length k)
// (throw an exception if no such kgram)
char kRand(string kgram);

string generate(string kgram, int L);
}


```

9.2 Key algorithms, Data structures and OO Designs used in this Assignment:

RandWriter's constructor takes a string and and the order of the kgrams as input. It initializes and constructs the hashmap. The kRand function generates a string of length n, using the hashmap that was generated. The function just iterates through a kgram's character map (the nested, inner map) and appends them in a string. Then it randomly returns a char from this string. The freq function returns the frequency of a kgram by searching/traversing the string. The second overloaded function returns the frequency of a character, which is simple and just looks it up in the hashmap; > return ktable.at(kgram).at(c); The generate function generates a new string by just calling the rand function, until the desired output length is reached.

9.3 What I accomplished :

I have accomplished to implement Markov Model. I have created a working Random Writer.

9.4 What I learned :

I have learnt about the Markov model and how it works. Also learnt about the Hash Map.

9.5 Challenges :

Implementing the Markov Model was Challenging for me.

9.6 Codebase

Makefile :

This Makefile contains the lint.

```
compiler = g++
flags_c = -g -Wall -Werror -std=c++0x -pedantic
flags_S = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
lboost = -lboost_unit_test_framework

all:    TextWriter test lint

lint:
     cpplint --filter=-runtime/references,-build/c++14 --root=. *.h *.cpp

TextWriter: TextWriter.o RandWriter.o
    $(compiler) TextWriter.o RandWriter.o -o TextWriter

test:   test.o RandWriter.o
    $(compiler) test.o RandWriter.o -o test $(lboost)

TextWriter.o:TextWriter.cpp RandWriter.h
    $(compiler) -c TextWriter.cpp RandWriter.h $(flags_c)

RandWriter.o:RandWriter.cpp RandWriter.h
    $(compiler) -c RandWriter.cpp  RandWriter.h $(flags_c)

test.o:test.cpp
    $(compiler) -c test.cpp $(lboost)

clean:
    rm *.o
    rm *.gch
    rm TextWriter
    rm test
```

TextWriter.cpp :

This file is important as it takes command line arguments and also gives you the output.

```
// Copyright 2022 Aniketh Rai
#include <string>
#include "RandWriter.h"
int main(int argc, const char *argv[]) {
  if (argc != 3) {
    throw "bad Input. Usage: ./TextWriter (int k) (int T) < input.txt";
    return 0;
  }
std::string str_k(argv[1]);
std::string str_t(argv[2]);

int k = std::stoi(str_k);
int l = std::stoi(str_t);

std::string input = ""; // NOLINT
std::string output = ""; // NOLINT

int i = 0, len = 0;

while (std::getline(std::cin, input) && i < l) {
    if (input.length() > static_cast<unsigned int>(k)) {
        try {
```

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

    return 0;
}
```

RandWriter.h :

The initialization of variables and methods.

```
// Copyright 2022 Aniketh Rai
#ifndef RANDWRITER_H_
#define RANDWRITER_H_
#include <algorithm>
#include <iostream>
#include <map>
#include <string>
#include <stdexcept>

class RandWriter {
 public:
  // constructor
  RandWriter(std::string text, int k);
  // API
  int order_k();
  int freq(std::string kgram);
  int freq(std::string kgram, char c);
  char k_Rand(std::string kgram);
  std::string generate(std::string kgram, int T);
  // overload <<
  friend std::ostream &operator<<(std::ostream &out, RandWriter &mm);
 private:
  int k;
  std::string text;
  std::map<std::string, std::map<char, int>> ktable;
  std::string alpha;
};
#endif  // RANDWRITER_H_
```

RandWriter.cpp :

The important functions for the Markov Model are implemented in this file.

```
// Copyright 2022 Aniketh Rai
#include "RandWriter.h"
#include <stdlib.h>
#include <algorithm>
#include <map>
#include <string>
#include <stdexcept>
#include <vector>
#include <utility>

RandWriter::RandWriter(std::string text, int k) {
  RandWriter::k = k;
  std::map<std::string, int>::iterator it;
  RandWriter::text = text;

  if (text.length() < (unsigned)k) {
    throw std::runtime_error("Error.RandWriter.RandWriter -"
      " length of text should be > than k");
  }

  std::map<char, int> duplicates;
  for (int i = 0; i < static_cast<int>(text.length()); i++) {
    char tmp = text.at(i);

    if (!duplicates.count(tmp)) {
      alpha.push_back(tmp);
      duplicates[tmp] = 1;
    }
  }
  std::sort(alpha.begin(), alpha.end());
  int pos = 0;
    for (int i = 0; i < static_cast<int>(text.length()); i++) {
        std::string kgram;
        std::map<char, int> ftable;
        for (int j = i; j < i + k; j++) {
            pos = j >= static_cast<int>(text.length()) ? \
              j - static_cast<int>(text.length()) : j;
            kgram += text.at(pos);
        }
        pos++;
        if (pos >= static_cast<int>(text.length())) \
          pos -= static_cast<int>(text.length());
        ftable.insert(std::make_pair(text.at(pos), 0));

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

        ktable[kgram][text.at(pos)]++;
    }
}

int RandWriter::order_k() {
  return [=]() -> int {
    return k;
  }();
}
```

```
58
59 int RandWriter::freq(std::string k_gram) {
60   if (k_gram.length() != (unsigned)k) {
61     throw std::runtime_error("Error.RandWriter.freq - kgram not of length k.
     ");
62   }
63
64   int count = 0;
65     for (int i = 0; i < static_cast<int>(text.length()); i++) {
66       int pos = 0;
67         std::string kg;
68         for (int j = i; j < i + k; j++) {
69             pos = j >= static_cast<int>(text.length()) ? j - text.length() :
     j;
70             kg += text.at(pos);
71         }
72         if (k_gram == kg) count++;
73     }
74     return count;
75 }
76
77 int RandWriter::freq(std::string k_gram, char c) {
78   if (k_gram.length() != (unsigned)k) {
79     throw std::runtime_error("Error.RandWriter.freq - kgram not of length k.
     ");
80   }
81   try {
82     return ktable.at(k_gram).at(c);
83   } catch (std::out_of_range &e) {
84     return 0;
85   }
86 }
87
88 char RandWriter::k_Rand(std::string k_gram) {
89   if (k_gram.length() != (unsigned)k) {
90     throw std::runtime_error("Error.RandWriter.k_Rand -"
91       " kgram not of length k.");
92   }
93
94   std::map<std::string, std::map<char, int>>::iterator it;
95   it = ktable.find(k_gram);
96   std::string outx = "";
97   for (auto const &var2 : it -> second) {
98     outx += var2.first;
99   }
100  unsigned int seed = 100;
101  int r = rand_r(&seed) % outx.length();
102  return outx.at(r);
103 }
104
105 std::string RandWriter::generate(std::string k_gram, int L) {
106   if (k_gram.length() != (unsigned)k) {
107     throw std::runtime_error("Error.RandWriter.generate -"
108     " kgram not of length k.");
109   }
110   for (int i = k; i < L; i++) k_gram += k_Rand(k_gram.substr(i - k, k));
111   return k_gram;
112 }
113
```

```
114 std::ostream &operator<<(std::ostream &out, RandWriter &randWriter ) {
115   out << "Original Text: " << randWriter.text << std::endl;
116   out << "Markov Order: " << randWriter.k << std::endl;
117   out << "Frequency Map: " << std::endl;
118
119   std::map<std::string, std::map<char, int>>::iterator it;
120
121   for (it = randWriter.ktable.begin(); it != randWriter.ktable.end(); it++)
       {
122     out << it->first << ":" << "\n";
123     for (auto const &var2 : it -> second)
124       out << "\t" << var2.first << ": " << var2.second << "\n";
125   }
126
127   return out;
128 }
```

test.cpp :

The test file tests for the given input as well as runtimeerror exception

```
1  // Copyright 2022 Aniketh Rai
2
3  #include "RandWriter.h"
4
5  #define BOOST_TEST_DYN_LINK
6  #define BOOST_TEST_MODULE Main
7  #include <boost/test/unit_test.hpp>
8
9  BOOST_AUTO_TEST_CASE(base_test) {
10     int k = 2;
11     std::string str = "gagggagaggcgagaaa";
12     RandWriter rWriter(str, k);
13
14     std::cout << "Testing with: " << std::endl;
15     std::cout << rWriter << std::endl;
16
17     std::string gen = rWriter.generate("cg", 10);
18     BOOST_REQUIRE(gen.length() == 10);
19
20     BOOST_REQUIRE(rWriter.freq("aa") == 2);
21     BOOST_REQUIRE(rWriter.freq("ga", 'g') == 4);
22     BOOST_REQUIRE(rWriter.freq("ag", 'a') == 3);
23     BOOST_REQUIRE(rWriter.freq("ag", 'c') == 0);
24
25     // a is the only character that comes after cg
26     char rand = rWriter.k_Rand("cg");
27     BOOST_REQUIRE(rand == 'a');
28
29     BOOST_REQUIRE(rWriter.order_k() == k);
30 }
31
32 BOOST_AUTO_TEST_CASE(exception_test) {
33     RandWriter rWriter("pqrs", 4);
34     BOOST_REQUIRE_THROW(rWriter.freq("x"), std::runtime_error);
35     BOOST_REQUIRE_THROW(rWriter.freq("bb", 'b'), std::runtime_error);
36     BOOST_REQUIRE_THROW(rWriter.k_Rand("y"), std::runtime_error);
37 }
```

10 PS7: Kronos Log Parsing

10.1 Description:

We analyze the Kronos Intouch time clock log by using regular expressions to parse the file. Also we verify device boot up timing. here we take the given device[1-6]*underscore*intouch.log files and give a resultant file which contains the time, date, status and line number of the boot to the .rpt file.

10.2 Key algorithms, Data structures and OO Designs used in this Assignment:

I used the lambda expression for getting the time from first 19 characters i.e substring(0,19). I prefer using lambda expression than ordinary method as its more efficient. Just utilized the three functions from the regex library to finish my project.

10.3 Explanation of the code:

Firstly, I initialize regular expressions to the starting and ending log entries so it can be used to compare the regex to find. if there is a match in every line of the file(.log). I create an -outputfile where the data of the result is stored as filename.log.rpt .There is a use of bool exp to keep tracking of the incomplete booting. I utilize the regex-search function to search a match for the starting regular expression(log.c.166). if there is no incomplete booting, I write the date,time,line number and status of the boot to our -outputfile. The else if condn is used to search the line for finding the match for the ending regular expression. if that is found, The calculation of the duration of the time is begun by using posix. Again I write the line-number, date , time and duration it took for the completion of sequence.

10.4 What I accomplished :

I accomplished creating Kronos log Parsing and approach towards regular expression in C++.

10.5 What I learned :

I learnt implementing of the regex library <regex> More efficiently.
I have implemented following regex functions into my code.

- boost::regex startMessage("(
  (log.c.166
) server started)");
- boost::regex endMessage("(oejs.AbstractConnector:Started SelectChannelConnector)");
- regex-search(s, startMessage)
- regex-search(s, endMessage)

10.6 Challenges :

Implementing a proper format was challenging to me to the output file.

10.7 Codebase

Makefile:
This Makefile has lint.

```
1 # Copyright 2022 Aniketh Rai
2 CC = g++
3 CFLAGS = -std=c++17 -Wall -Werror -pedantic
4 LFLAGS = -lboost_regex -lboost_date_time
```

```
5
6  all: ps7 lint
7  lint:
8      cpplint --filter=-runtime/references,-build/c++11 --root=. *.cpp
9
10 ps7: main.o
11     $(CC) -o $@ main.o $(LFLAGS)
12
13 main.o: main.cpp
14     $(CC) $(CFLAGS) -c main.cpp -o main.o
15
16 clean:
17     rm -f ps7 *.o
```

main.cpp:

The main file contains all the regex library as well as time library and functions to achieve the *Kronos Log Parsing*

```
1  // Copyright 2022 Aniketh Rai
2  #include <iostream>
3  #include <fstream>
4  #include <string>
5  #include <boost/date_time/posix_time/ptime.hpp>
6  #include <boost/date_time/posix_time/time_parsers.hpp>
7  #include "boost/date_time/posix_time/posix_time.hpp"
8  #include <boost/regex.hpp>
9  using boost::posix_time::time_duration;
10 int main(int argc, char *argv[]) {
11 int n = 1;
12
13 std::string s, _inputfile; // NOLINT
14 boost::regex startMessage("( \\(log.c.166\\) server started)");
15 boost::regex endMessage(
16     "(oejs.AbstractConnector:Started SelectChannelConnector)");
17 _inputfile = argv[1];
18
19 std::ifstream myFile(_inputfile);
20 std::ofstream output_file;
21 std::string _ofilename = _inputfile + ".rpt"; // NOLINT
22
23 bool stillBooting = false;
24 boost::posix_time::ptime start_time, end_time;
25 if (!myFile) {
26 std::cout << "file cannot be opened\n";
27 exit(-1);
28 }
29 output_file.open(_ofilename);
30 if (!output_file.is_open()) {
31 std::string oerror = "Creation of file " + _ofilename + " not possible"; //
     NOLINT
32 exit(1);
33 }
34 auto getTime = [](std::string time_str) {
35 return time_str.substr(0, 19);};
36 if (output_file.is_open()) {
37         output_file << "The Device Boot Report\n";
38         output_file << "InTouch log file: " << argv[1] << "\n\n";}
39 while (getline(myFile, s)) {
40 if (regex_search(s, startMessage)) {
41     start_time = boost::posix_time::time_from_string(getTime(s));
42     if (stillBooting) {
```

```
        output_file << "~~~~ Incomplete Booting ~~~~ \n\n";
        stillBooting = false;
        }
    stillBooting = true;
    output_file << "***** Device Boot *****\n";
    output_file << n << "(" << argv[1] << "): "
            << start_time << " Boot Start\n";
} else if (regex_search(s, endMessage)) {
    stillBooting = false;
    end_time = boost::posix_time::time_from_string(getTime(s));
    time_duration duration = end_time - start_time;
    output_file << n << "(" << argv[1] << "): "
                << end_time << " Boot Completed\n\tBoot Time: "
                << duration.total_milliseconds() << "ms\n\n";
    }
n++;}
if (stillBooting) {
    output_file << "!!!! Incomplete Booting !!!! \n\n";
}
return 0;
}
```

10.8 Output:

```
The Device Boot Report
InTouch log file: device1_intouch.log

***** Device Boot *****
435369(device1_intouch.log): 2014-Mar-25 19:11:59 Boot Start
435759(device1_intouch.log): 2014-Mar-25 19:15:02 Boot Completed
        Boot Time: 183000ms

***** Device Boot *****
436500(device1_intouch.log): 2014-Mar-25 19:29:59 Boot Start
436859(device1_intouch.log): 2014-Mar-25 19:32:44 Boot Completed
        Boot Time: 165000ms

***** Device Boot *****
440719(device1_intouch.log): 2014-Mar-25 22:01:46 Boot Start
440791(device1_intouch.log): 2014-Mar-25 22:04:27 Boot Completed
        Boot Time: 161000ms

***** Device Boot *****
440866(device1_intouch.log): 2014-Mar-26 12:47:42 Boot Start
441216(device1_intouch.log): 2014-Mar-26 12:50:29 Boot Completed
        Boot Time: 167000ms

***** Device Boot *****
442094(device1_intouch.log): 2014-Mar-26 20:41:34 Boot Start
442432(device1_intouch.log): 2014-Mar-26 20:44:13 Boot Completed
        Boot Time: 159000ms

***** Device Boot *****
443073(device1_intouch.log): 2014-Mar-27 14:09:01 Boot Start
443411(device1_intouch.log): 2014-Mar-27 14:11:42 Boot Completed
        Boot Time: 161000ms
```

Figure 19: device1 output

```
The Device Boot Report
InTouch log file: device5_intouch.log

***** Device Boot *****
31063(device5_intouch.log): 2014-Jan-26 09:55:07 Boot Start
31176(device5_intouch.log): 2014-Jan-26 09:58:04 Boot Completed
        Boot Time: 177000ms

***** Device Boot *****
31274(device5_intouch.log): 2014-Jan-26 12:15:18 Boot Start
~~~~ Incomplete Booting ~~~~

***** Device Boot *****
31293(device5_intouch.log): 2014-Jan-26 14:02:39 Boot Start
31401(device5_intouch.log): 2014-Jan-26 14:05:24 Boot Completed
        Boot Time: 165000ms

***** Device Boot *****
32623(device5_intouch.log): 2014-Jan-27 12:27:55 Boot Start
~~~~ Incomplete Booting ~~~~

***** Device Boot *****
32641(device5_intouch.log): 2014-Jan-27 12:30:23 Boot Start
~~~~ Incomplete Booting ~~~~

***** Device Boot *****
32656(device5_intouch.log): 2014-Jan-27 12:32:51 Boot Start
~~~~ Incomplete Booting ~~~~

***** Device Boot *****
32674(device5_intouch.log): 2014-Jan-27 12:35:19 Boot Start
~~~~ Incomplete Booting ~~~~

***** Device Boot *****
32693(device5_intouch.log): 2014-Jan-27 14:02:38 Boot Start
32801(device5_intouch.log): 2014-Jan-27 14:05:21 Boot Completed
        Boot Time: 163000ms

***** Device Boot *****
33709(device5_intouch.log): 2014-Jan-28 12:44:17 Boot Start
~~~~ Incomplete Booting ~~~~

***** Device Boot *****
33725(device5_intouch.log): 2014-Jan-28 14:02:33 Boot Start
33833(device5_intouch.log): 2014-Jan-28 14:05:15 Boot Completed
        Boot Time: 162000ms

***** Device Boot *****
34594(device5_intouch.log): 2014-Jan-29 12:43:07 Boot Start
~~~~ Incomplete Booting ~~~~

***** Device Boot *****
34613(device5_intouch.log): 2014-Jan-29 14:02:35 Boot Start
```

Figure 20: device5 output

```
The Device Boot Report
InTouch log file: device6_intouch.log

***** Device Boot *****
2(device6_intouch.log): 2014-Apr-03 20:27:48 Boot Start
161(device6_intouch.log): 2014-Apr-03 20:31:01 Boot Completed
        Boot Time: 193000ms

***** Device Boot *****
82079(device6_intouch.log): 2014-Apr-09 14:51:15 Boot Start
82303(device6_intouch.log): 2014-Apr-09 14:54:39 Boot Completed
        Boot Time: 204000ms

***** Device Boot *****
85398(device6_intouch.log): 2014-Apr-10 18:13:13 Boot Start
85564(device6_intouch.log): 2014-Apr-10 18:16:37 Boot Completed
        Boot Time: 204000ms

***** Device Boot *****
85957(device6_intouch.log): 2014-Apr-10 19:11:05 Boot Start
86123(device6_intouch.log): 2014-Apr-10 19:14:24 Boot Completed
        Boot Time: 199000ms

***** Device Boot *****
86127(device6_intouch.log): 2014-Apr-10 19:18:36 Boot Start
86293(device6_intouch.log): 2014-Apr-10 19:21:56 Boot Completed
        Boot Time: 200000ms

***** Device Boot *****
86568(device6_intouch.log): 2014-Apr-10 19:32:16 Boot Start
86732(device6_intouch.log): 2014-Apr-10 19:35:36 Boot Completed
        Boot Time: 200000ms

***** Device Boot *****
86750(device6_intouch.log): 2014-Apr-10 20:06:27 Boot Start
86821(device6_intouch.log): 2014-Apr-10 20:09:07 Boot Completed
        Boot Time: 160000ms

***** Device Boot *****
86939(device6_intouch.log): 2014-Apr-11 00:15:56 Boot Start
87111(device6_intouch.log): 2014-Apr-11 00:18:49 Boot Completed
        Boot Time: 173000ms

***** Device Boot *****
87116(device6_intouch.log): 2014-Apr-11 13:28:25 Boot Start
87286(device6_intouch.log): 2014-Apr-11 13:31:12 Boot Completed
        Boot Time: 167000ms

***** Device Boot *****
87836(device6_intouch.log): 2014-Apr-11 13:58:02 Boot Start
88009(device6_intouch.log): 2014-Apr-11 14:00:49 Boot Completed
        Boot Time: 167000ms

```

Figure 21: device6 output