Computing IV Sec 011: Project Portfolio

Jeongjae Han

Summer 2022

Contents

Time to Complete Portfolio: 6 hours

1 PS0: Hello World with SFML

1.1 Discussion

This project was to setup a Linux build environment and use SFML library. I used Virtual Box with my desk top. I had to use SFML graphics library to put sprite and interact with it. I made the program to move the picture up and down, left to right, and rotate by using the code.

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::W)) spritePosition.y -= yVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::S)) spritePosition.y += yVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::A)) spritePosition.x -= xVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::D)) spritePosition.x += xVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Q)) sprite.setScale(-1,1);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::E)) sprite.setScale(1,1);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Space)) sprite.rotate(10.f);
```

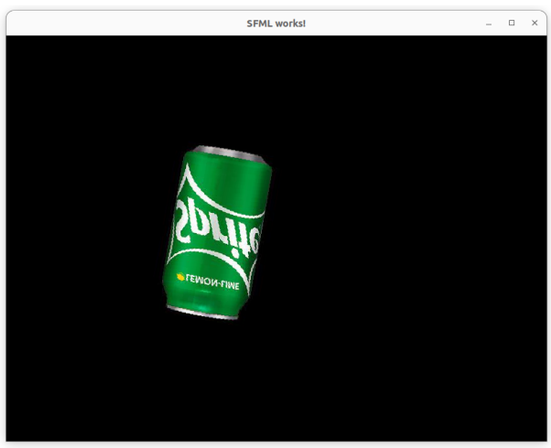


Figure 1: PS0 Result with the code implementation

1.2 Places to get help

I got most of the help from YouTube videos. It helped me to understand how SFML library works and which code to use in order to draw sprite on window and use effects.

1.3 What I accomplished

I accomplished to put the picture and move around with the code above, rotate, and flip the sprite over with two different keys.

1.4 What I learned

I learned how to make a program on window. I always was curious how to actually interact with a window and show some graphics other than showing the result output on the console box. It was cool to use SFML libraries and flag codes to put everything on the window which makes it easier for us.

1.5 Mistakes

I got two points off on this project because I miss the part of ps0 prompt saying I have to keep green circle that is provided us. For the other point off, I did not use relative path to

bring pictures to the program as prompt asked me to because I tried using it, but it was not calling the sprite correctly, so I ended up using absolute file path.

1.6 Codebase

Makefile

```
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

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

using namespace std;

int main()
{


    sf::RenderWindow window(sf::VideoMode(800, 600), "SFML works!");
    window.setFramerateLimit(60);


    sf::Texture texture;
    if(!texture.loadFromFile("/home/j/Desktop/comp4/ps0/sprite.png")){
        cout << "Could not load sprite texture";
        return 0;
    }
    sf::Sprite sprite;
    sprite.setTexture(texture);

    sf::Vector2f spritePosition(0,0);
    sprite.setOrigin(sf::Vector2f(88, 125));




    sprite.setPosition(spritePosition);

    float xVelocity = 10;
    float yVelocity = 5;


    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();

        }

        if (sf::Keyboard::isKeyPressed(sf::Keyboard::W)) spritePosition.y -= yVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::S)) spritePosition.y += yVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::A)) spritePosition.x -= xVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::D)) spritePosition.x += xVelocity;
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Q)) sprite.setScale(-1,1);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::E)) sprite.setScale(1,1);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Space)) sprite.rotate(10.f);

        // Setting the limit

        if (spritePosition.x < 0 + 88 ) spritePosition.x = 800 - 88;
        if (spritePosition.x > 800 - 88) spritePosition.x = 0 + 88;
        if (spritePosition.y < 0 + 125 ) spritePosition.y = 600-125;
        if (spritePosition.y > 600 - 125) spritePosition.y = 0 + 125;

        //physics

        sprite.setPosition(spritePosition);

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

    }

    return 0;
}
```

2 PS1a: Linear Feedback Shift Register

2.1 Discussion

Linear Feedback Shift Register is a register that takes a linear function of a previous state as an input. Most commonly, this function is a Boolean exclusive-or (XOR). LFSR performs discrete step operators. Therefore, when we keep shifting the bitstring, we do not get string with 0 at the end. However, for this project we had to XOR 4 bits from the seed and put it at the new spot. For this project, code step function and generate function.

For step, the function had to shift left once and return the new bit that I XORed. For generate I had to pass an integer value 'k' and run step function k times. Then, update the seed and returns a integer that is step's return value = new value + value * 2.

Also, I had to test the code by using boost unit test framework library.

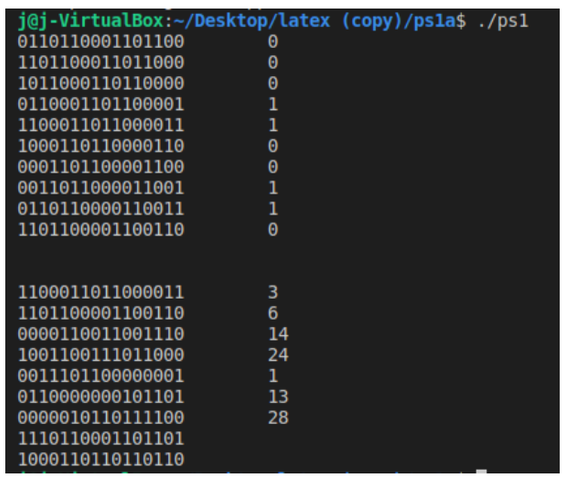


Figure 2: The result of main.cpp

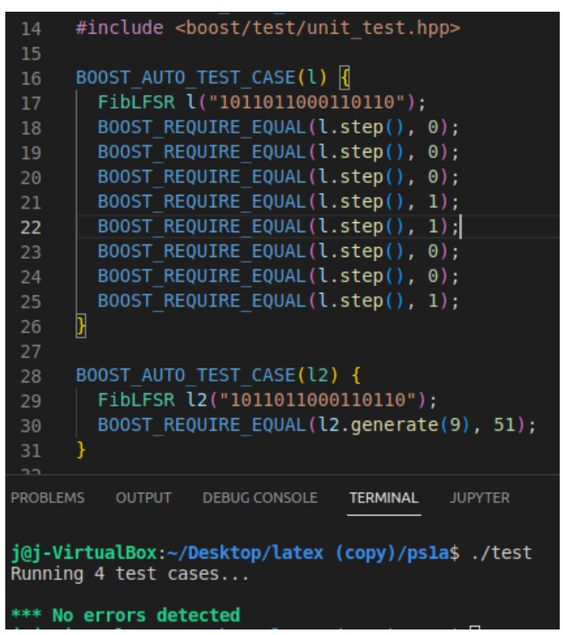


Figure 3: The result of test.cpp

2.2 What I accomplished

I accomplished to XOR by adding the right index of the string and and gate 0x0001 because I searched on the internet, for XOR, I can power the value or add the values, so I used addition because the logic seemed easier.

```
output = (aSeed.at(0) + aSeed.at(2) + aSeed.at(3) + aSeed.at(5)) & 0x00001;
```

2.3 What I already knew

I knew how to use string's .at(index) function to XOR the right index, so it was not hard to shift the bitstring.

2.4 What I learned

I learned how to code for boost unit test framework library, and it was useful to know because if I have to test it by coding one by one, it will not be as efficient as test framework library.

2.5 Mistakes

I got two points off because I had to make two tests by myself, but I did not know that. However, I made it for PSXa and got my points back.

2.6 Codebase

Makefile

```
CC = g++
CFLAGS = --std=c++14 -Wall -Werror -pedantic
DEPS = FibLFSR.hpp
LIBS = -lboost_unit_test_framework

all: ps1 test lint

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

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

ps1: main.o FibLFSR.o
    $(CC) $(CFLAGS) -o $@ $^

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

test: test.o FibLFSR.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIBS)

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

clean:
    rm *.o ps1 test
```

main.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/8/2022]
#include <iostream>
#include <string>
#include "FibLFSR.hpp"

int main() {
    FibLFSR flfsr("1011011000110110");

    int output = 0;
    for (int i = 0; i < 10; i++) {
        output = flfsr.step();
        std::cout << flfsr << '\t' << output << std::endl;
```

```
    }

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

    FibLFSR flfsr1("1011011000110110");
    for (int i = 0; i < 7; i++) {
        output = flfsr1.generate(5);
        std::cout << flfsr1 << '\t' << output << std::endl;
    }

    FibLFSR l3("0111011000110110");
    l3.step();
    std::cout << l3.getSeed() << std::endl;

    FibLFSR l4("0111011000110110");
    l4.generate(6);
    std::cout << l4.getSeed() << std::endl;

    return 0;
}
```

FibLFSR.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#include "FibLFSR.hpp"

std::string FibLFSR::getSeed() const {
    return iSeed;
}

void FibLFSR::setSeed(std::string seed) {
    iSeed = seed;
}

int FibLFSR::step() {
    std::string aSeed = getSeed();
    int output;
    std::string out;

    // addition works as xor
    output = (aSeed.at(0) + aSeed.at(2) + aSeed.at(3) + aSeed.at(5)) & 0
   x00001;
    aSeed.erase(aSeed.begin());

    // changing the output to string
    if (output == 1) out = "1";
    else if (output == 0) out = "0";
    aSeed = aSeed + out;
    setSeed(aSeed);

    return output;
}


int FibLFSR::generate(int k) {
        int result = 0;
        for (int i = 0; i < k; i++) {
            result *= 2;
            result += step();
        }
        return result;
```

```
    }

std::ostream& operator<<(std::ostream& out, const FibLFSR& lfsr) {
    out << lfsr.getSeed();
    return out;
}

FibLFSR::~FibLFSR() {
    iSeed.clear();
}
```

FibLFSR.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#pragma once
#include <iostream>
#include <string>

class FibLFSR {
 public:
    // Constructor to create LFSR with the given initial seed
    explicit FibLFSR(std::string seed) : iSeed(seed) {}
    std::string getSeed() const;
    void setSeed(std::string seed);
    // Simulate one step and return the new bit as 0 or 1
    int step();
    // Simulate k steps and return a k-bit integer
    int generate(int k);

    ~FibLFSR();

 private:
    // Any fields that you need
    std::string iSeed;
};

std::ostream& operator<<(std::ostream& out, const FibLFSR& lfsr);
```

test.cpp

```
// 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(1) {
  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(l2) {
  FibLFSR l2("1011011000110110");
  BOOST_REQUIRE_EQUAL(l2.generate(9), 51);
}

BOOST_AUTO_TEST_CASE(l3) {
  FibLFSR l3("0111011000110110");
  BOOST_REQUIRE_EQUAL(l3.step(), 1);
}

BOOST_AUTO_TEST_CASE(l4) {
  FibLFSR l4("0111011000110110");
  BOOST_REQUIRE_EQUAL(l4.generate(6), 54);
}
```

3 PS1b: PhotoMagic

3.1 Discussion

This project uses PS1a's codes and use the codes to encrypt and decrypt a picture. The program calls a pixel and takes the bitstring of the color of the pixel and uses generate function to change the value of the color and stores it. Then moves to other pixel and do the same thing until the last pixel, so the program encrypts the picture. Then if you run the encrypted picture, it will use the same technique to decrypt the picture and you will be able to see the original image.

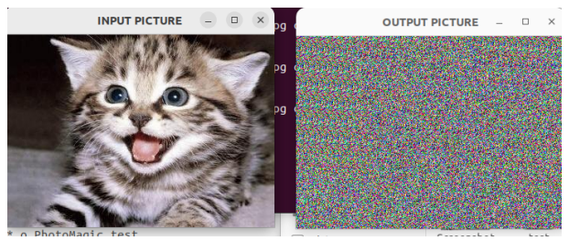


Figure 4: After encrypting an image

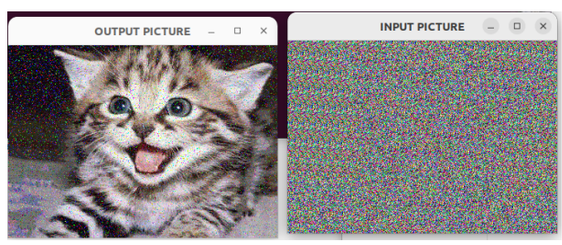


Figure 5: After decrypting an image

3.2 Places to get help

I got help from stackoverflow, cplusplus.com, ps0, and youtube.com

3.3 What I accomplished

My program ran as it should. Since the size of the picture was 200 x 200 I made two for-loops one for the side and one for the up and down, so the program goes over all the pixels.

```
for (int x = 0; x<200; x++) {
    for (int y = 0; y< 200; y++) {
        p = image.getPixel(x, y);
        p.r = 255 - p.r;
        p.g = 255 - p.g;
        p.b = 255 - p.b;
        image.setPixel(x, y, p);
    }
}
```

3.4 What I learned

I learned that every pixel stores 3 values for color: red, green, and blue which are the three colors for the light, and I can interact with it by assigning the values for each of them. Also, I did not expect that left shifting the value with a key will encrypt and also decrypt image.

3.5 Extra Credit

I got 2 extra credits because I made the binary bit string by adding the characters of the string up and modular 2 and divide by 2 the integer. I implemented it to the header file then, I can test if the program the codes are running properly.

3.6 Codebase

Makefile

```
CC = g++
DEP = FibLFSR.hpp
CFLAGS = -Wall -Werror -pedantic --std=c++14
LIB = -lsfml-graphics -lsfml-window -lsfml-system

all: PhotoMagic test

PhotoMagic.o: PhotoMagic.cpp $(DEP)
    $(CC) $(CFLAGS) -o $@ -c $<

FibLFSR.o: FibLFSR.cpp $(DEP)
    $(CC) $(CFLAGS) -o $@ -c $<

PhotoMagic: PhotoMagic.o FibLFSR.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

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

test: test.o FibLFSR.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB) -lboost_unit_test_framework

clean:
    rm *.o PhotoMagic test
```

pixels.cpp

```
// pixels.cpp:
// using SFML to load a file, manipulate its pixels, write it to disk


// g++ -o pixels pixels.cpp -lsfml-graphics -lsfml-window -lsfml-system

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

int main()
{
    sf::Image image;
    if (!image.loadFromFile("cat.jpg"))
        return -1;

    // p is a pixelimage.getPixel(x, y);
    sf::Color p;

    // create photographic negative image of upper-left 200 px square
    for (int x = 0; x<200; x++) {
        for (int y = 0; y< 200; y++) {
            p = image.getPixel(x, y);
            p.r = 255 - p.r;
            p.g = 255 - p.g;
            p.b = 255 - p.b;
            image.setPixel(x, y, p);
        }
    }

    sf::Vector2u size = image.getSize();
```

```
    sf::RenderWindow window(sf::VideoMode(size.x, size.y), "Meow");

    sf::Texture texture;
    texture.loadFromImage(image);

    sf::Sprite sprite;
    sprite.setTexture(texture);

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

        window.clear(sf::Color::White);
        window.draw(sprite);
        window.display();
    }

    // fredm: saving a PNG segfaults for me, though it does properly
    //   write the file
    if (!image.saveToFile("cat-out.bmp"))
        return -1;

    return 0;
}
```

FibLFSR.cpp

```
#include "FibLFSR.hpp"

FibLFSR::FibLFSR(std::string seed){
    string sentence = seed;
    // alphabet password to binary
    int alpha= 0;
    for(int i = 0; i < (signed)sentence.length(); i++)
    {
        if(isalpha(sentence.at(i))||sentence.at(i)>=2){
        alpha++;
        break;
        }
    }
    if(alpha > 0){

        int number= 0;
        string binary;
        for(int i=0; i<(signed)sentence.length(); i++){
            number += sentence.at(i);
        }

        while(number!=0){
            binary = ((number%2 == 0 ? "0" : "1" ) + binary);
            number/=2;
        }
        sentence = binary;
    }
    while(sentence.length() < 16){
        sentence = "0" + sentence;
```

```
    }
    iSeed = sentence;

}
int FibLFSR::step() {
    string aSeed = getSeed();
    int output;
    string out;

    //addition works as xor
    output = (aSeed.at(0) + aSeed.at(2) + aSeed.at(3) + aSeed.at(5)) & 0x00001;
    aSeed.erase(aSeed.begin());

    //changing the output to string
    if (output == 1) out = "1";
    else if (output == 0) out = "0";
    aSeed = aSeed + out;
    setSeed(aSeed);

    return output;
}


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

ostream& operator<<(std::ostream& out, const FibLFSR& lfsr) {
    out << lfsr.getSeed();
    return out;
}


FibLFSR::~FibLFSR() {
    iSeed.clear();
}
```

FibLFSR.hpp

```
#pragma once
#include <iostream>
#include <string>

using namespace std;

class FibLFSR {
public:
    // Constructor to create LFSR with the given initial seed
    FibLFSR(std::string seed);
    string getSeed() const { return iSeed; }
    void setSeed(string seed) { iSeed = seed; }
    // Simulate one step and return the new bit as 0 or 1
    int step();
    // Simulate k steps and return a k-bit integer
```

```
    int generate(int k);

    ~FibLFSR();

private:
    // Any fields that you need
    string iSeed;
};

ostream& operator<<(std::ostream& out, const FibLFSR& lfsr);
```

PhotoMagic.cpp

```

// g++ -o pixels pixels.cpp -lsfml-graphics -lsfml-window -lsfml-system
#include <iostream>
#include <string>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "FibLFSR.hpp"

using namespace std;

void transform(sf::Image& input, FibLFSR* encrypt);

int main(int argc, char* argv[])
{
    //start
    if(argc !=4){
        cout << "Usage: ./PhotoMagic [input_file_name.type] [output_file_name.type] [seed(binary)]" << endl;
        return -1;
    }

    string iPic = argv[1];
    string oPic = argv[2];
    FibLFSR key(argv[3]);

    //Input pic
    sf::Image inPic;
    if (!inPic.loadFromFile(iPic)) return -1;

    //Output pic
    sf::Image outPic;
    if (!outPic.loadFromFile(iPic)) return -1;

    transform(outPic, &key);

    // window size set up
    sf::Vector2u size = inPic.getSize();
    sf::RenderWindow inPic_window(sf::VideoMode(size.x, size.y), "INPUT PICTURE");
    sf::RenderWindow outPic_window(sf::VideoMode(size.x, size.y), "OUTPUT PICTURE");

    //bring the image
    sf::Texture inPic_text, outPic_text;
    inPic_text.loadFromImage(inPic);
    outPic_text.loadFromImage(outPic);

```

```
    sf::Sprite inPic_spr, outPic_spr;

    inPic_spr.setTexture(inPic_text);
    outPic_spr.setTexture(outPic_text);

    while (inPic_window.isOpen() && outPic_window.isOpen())
    {
        sf::Event event;
        while (inPic_window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                inPic_window.close();
        }

        while(outPic_window.pollEvent(event)){
            if(event.type == sf::Event::Closed)
                outPic_window.close();
        }
        inPic_window.clear();
        inPic_window.draw(inPic_spr);
        inPic_window.display();
        outPic_window.clear();
        outPic_window.draw(outPic_spr);
        outPic_window.display();
    }

    // fredm: saving a PNG segfaults for me, though it does properly
    //   write the file
    if (!outPic.saveToFile(oPic))
        return -1;

    return 0;
}

void transform(sf::Image& input, FibLFSR* encrypt){
    int x = 0, y = 0;

    sf::Vector2u size = input.getSize();

    sf::Color p;

    for (x = 0; x< (signed)size.x; x++) {
        for (y = 0; y< (signed)size.y; y++) {
            p = input.getPixel(x, y);
            p.r = p.r ^ encrypt->generate(8);
            p.g = p.g ^ encrypt->generate(8);
            p.b = p.b ^ encrypt->generate(8);
            input.setPixel(x, y, p);
        }
    }
}
```

test.cpp

```
// 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(l) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  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_AUTO_TEST_CASE(l2) {
  FibLFSR l2("1011011000110110");
  BOOST_REQUIRE_EQUAL(l2.generate(9), 236);
}

BOOST_AUTO_TEST_CASE(l3){
  FibLFSR l3("0111011000110110");
  BOOST_REQUIRE_EQUAL(l3.step(),0);
}

BOOST_AUTO_TEST_CASE(l4){
  FibLFSR l4("0111011000110110");
  BOOST_REQUIRE_EQUAL(l4.generate(6), 29);
}

BOOST_AUTO_TEST_CASE(b1){
  FibLFSR b1("jason");
  BOOST_REQUIRE_EQUAL(b1.getSeed(), "0000001000011011");
}

BOOST_AUTO_TEST_CASE(b2){
  FibLFSR b2("1234");
  BOOST_REQUIRE_EQUAL(b2.getSeed(), "0000000011001010");
}
```

4 PS2: Triangle Fractal

4.1 Discussion

For this program, I had to write a program that draws Sierpinski triangle. The input must be depth and length of the triangles. Depth is for the number of smaller trinagles and length is for the length for the first triangle. There should be a big triangle and smaller triangles at its vertices. The positions of small triangles are fixed, so I just had to make a function that draws /2 length of the edges and assign their edges' positions.

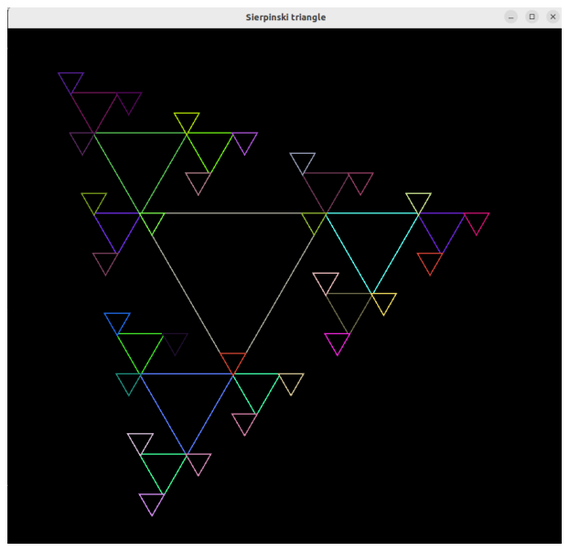


Figure 6: Triangle Fractal with depth of 4 and size of 300

4.2 Places to get help

I got help from Samin Patel who helped me to understand what the program is asking for, sfml-dev.org helped me to understand how sf::drawable works. For coding, c++.com, stackoverflow, and YouTube helped me.

4.3 Challenges

I used sf::drawable as a parent inheritance since the project prompt told me to. I tried to make it as a tree only with the class by making other triangles as the members, but it was not working well, so I used vector since c++ has the library for it.

4.4 Mistakes

I got two points off because I have one issue with cpplint. I am not sure which code I have to run it with. Whenever I run cpplint code that is provided, it says, "FATAL ERROR: No files were specified." However, I tried to run it with the one on the lectrue note which is "'cpplint *.cpp, *.hpp, it shows many error messages and for some of them which I do not understand what it is asking for.. Also, ironically, it does not check cpp files, so I have to run the code just for cpp again. Since the prompt provided the code, I put that code in my Makefile.

For other point, I did not use command-line parameters instead I used stdin.

4.5 Extra Credit

The triangles have different colors. I assigned random color code for each triangle's rgb.

4.6 Codebase

Makefile

```makefile
CC = g++
CFLAGS = -Wall -Werror -pedantic --std=c++14
LIB = -lsfml-graphics -lsfml-window -lsfml-system
DEP = TFractal.hpp

all: TFractal lint

TFractal.o: TFractal.cpp $(DEP)
    $(CC) $(CFLAGS) -o $@ -c $<

TFractal: TFractal.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

lint:
    cpplint --filter=-runtime/references,-build/header_guard,-build/c++11 --extensions=cpp,hpp

clean:
    rm *.o TFractal
```

TFractal.cpp

```cpp
#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
#include <cmath>
#include <vector>
#include "TFractal.hpp"

using namespace std;

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

    int color = rand() % 255;

    double length ;
    int depth ;

    cout << "Input format: depth of the Sierpinski triangle(enter)\t length of the triangle (enter)\n";
    cin >> depth;
    cin >> length;

    vector<Triangle> triTree;

    //put the first triangle to the vector
    triTree.push_back(Triangle(depth, length, length*5/7, length, color));

    //iterate the function to create more triangles
    fTree(triTree.front(), triTree);

    sf::RenderWindow window(sf:: VideoMode(length*3,length*3), "Sierpinski triangle");
    window.setFramerateLimit(60);

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

```
        }

        window.clear();
        //iterate the vector to draw
        vector<Triangle>::iterator p;
        for(p = triTree.begin(); p!= triTree.end(); ++p){
            window.draw(*p);
        }
        window.display();
    }

    return 0;
}
```

TFractal.hpp

```
#pragma once
#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
#include <cmath>
#include <vector>

using namespace std;

class Triangle: public sf::Drawable{
public:
    Triangle(int d, double l, double x, double y, int color);

    int getDepth() const;
    double getLength() const;

    // coordinates of the center
    double getX() const;
    double getY() const;

    ~Triangle();

private:
    void draw(sf::RenderTarget& target, sf::RenderStates states) const {
        target.draw(triangle, states);
    }
    int depth;
    double length;
    double pos_x;
    double pos_y;
    sf::ConvexShape triangle;
};

void fTree(const Triangle &parent, std::vector<Triangle> &v);

Triangle::Triangle(int d, double l, double x, double y, int color): depth(d)
    , length(l), pos_x(x), pos_y(y) {

    double h = l*sqrt(3)/2; //height

    triangle.setPointCount(3);

    triangle.setPoint(0, sf::Vector2f(0,0)); // x = 0, y = 0;
    triangle.setPoint(1, sf::Vector2f(l, 0)); // x = l, y = 0;
```

```
    triangle.setPoint(2, sf::Vector2f(l/2, h)); // x = l/2, y = -h

    triangle.setFillColor(sf::Color::Black);
    triangle.setOutlineThickness(2.f);
    triangle.setOutlineColor(sf::Color(color, rand() % 255, rand() % 255)); //random color

    triangle.setPosition(pos_x,pos_y);
}

int Triangle::getDepth() const{
    return depth;
}
double Triangle::getLength() const{
    return length;
}

double Triangle::getX() const{
    return pos_x;
}
double Triangle::getY() const{
    return pos_y;
}

Triangle::~Triangle(){
    length = 0;
    depth = 0;
}

void fTree(const Triangle &parent, vector<Triangle> &v){
    if(parent.getDepth() == 0) return;

    int color = rand() % 255;

    Triangle Left(parent.getDepth()-1.0, parent.getLength()/2.0,\
        parent.getX() - parent.getLength()/4.0, parent.getY() - parent.getLength()*sqrt(3.0)/4.0, color);

    Triangle Right(parent.getDepth()-1.0, parent.getLength()/2.0,\
        parent.getX() + parent.getLength(), parent.getY(), color);

    Triangle Bottom(parent.getDepth()-1.0, parent.getLength()/2.0,\
        parent.getX(), parent.getY() + parent.getLength()*sqrt(3.0)/2.0, color);

    v.push_back(Left);
    v.push_back(Right);
    v.push_back(Bottom);

    fTree(Left, v);
    fTree(Right, v);
    fTree(Bottom, v);

}
```

5 PS3a: N-Body Simulation (Static)

5.1 Discussion

This program reads the text file which stored the informations of numbers, names, sizes, postions, and speed of planets, and takes those information in it and register them into the program and run. This program uses SFML library, to put the planet pictures on the window.

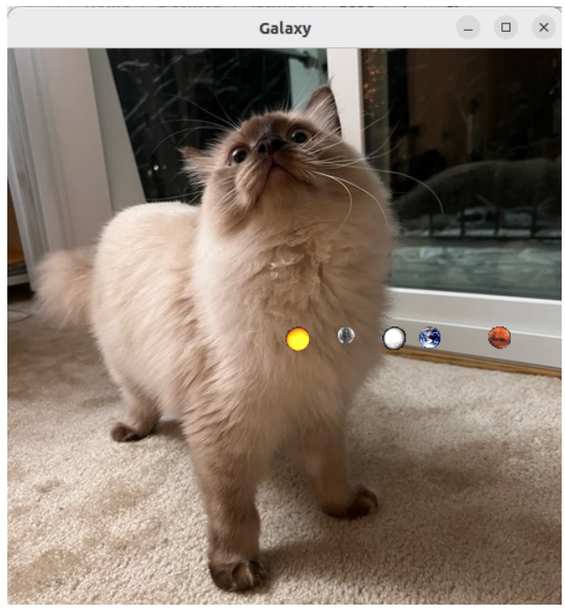


Figure 7: The program shows the planets

5.2 Places to get help

I got help from c++.com, stackoverflow, and Izzy from Dr.Daly's discord group.

5.3 What I accomplished

I used vector because I used vector for the previous program, so it was easier than other containers to declare the planets. I drew the planets with iterator since I used vector, so it was easy to draw. .gch files get produced when I compile, but I put the code in Makefile that it cleans it.

5.4 What I learned

I learned how to read the file and store it in to the class member by using these codes.

```
std::istream& operator>> (std::istream &in, universe &u) {
    std::string planetNum, radius;

    in >> planetNum;
    in >> radius;

    u.planetNum = atoi(planetNum.c_str());
    u.universeR = atof(radius.c_str());

    std::cout << "Number of planets: " << u.planetNum << std::endl;
    std::cout << "Radius: " << u.universeR << std::endl;

    for (int i = 0 ; i < u.planetNum; i++) {
        CelestialBody* cb = new CelestialBody();
        in >> *cb;
        cb->setRadius(u.universeR);
```

```
        cb->setPos();
        u.cbVec.push_back(*cb);
        std::cout << *cb;
    }

    return in;
}
```

5.5 Challenges

when I compile the codes, it shows error with .hpp files. It said it has problem with pragma once, but if I compile it again, it compiles fine. I got a feedback from my professor that I was trying to compile .hpp file.

Also, I tried to put vector variable for planets in the private member, but I could not because I did not know how to push back the vector as private member.

5.6 Mistakes

I got 3 points off because yaxis is inverted, Universe not Drawable, and vector variable was in public field.

5.7 Extra Credit

I got one extra credit because I put the background picture.

5.8 Codebase

Makefile

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

all: NBody lint

main.o: main.cpp universe.hpp
    $(CC) $(CFLAGS) -o $@ -c $<

universe.o: universe.cpp universe.hpp CelestialBody.hpp
    $(CC) $(CFLAGS) -c $^

CelestialBody.o: CelestialBody.cpp CelestialBody.hpp
    $(CC) $(CLFAGS) -c $<

NBody: main.o universe.o CelestialBody.o
    $(CC) $(CLFAGS) -o $@ $^ $(LIB)

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

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

main.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]

#include "universe.hpp" //NOLINT

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

```
    universe* uni = new universe();

    std::cin >> *uni;

    sf::RenderWindow window(sf::VideoMode(windowWidth, windowHeight), "
   Galaxy");
    window.setFramerateLimit(60);

    sf::Texture laylaTex;
    laylaTex.loadFromFile("Layla.jpg");
    if (!laylaTex.loadFromFile("Layla.jpg")) {  // Background image
     std::cout << "Cannot load Layla.jpg" << std::endl;
        exit(1);
    }

    sf::Sprite laylaSp;
    laylaSp.setTexture(laylaTex);
    laylaSp.setPosition(sf::Vector2f(0, 0));

    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed) {
                window.close();
            }
        }
        window.clear();
        window.draw(laylaSp);  // Draws Layla in the backgroud
        std::vector<CelestialBody>::iterator p;
        for (p = uni->cbVec.begin(); p != uni->cbVec.end(); p++) {
            window.draw(*p);
        }
        window.display();
    }

    return 0;
}
```

universe.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#pragma once
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include "CelestialBody.hpp"  //NOLINT
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

class universe {
 public:
    universe();

    friend std::istream& operator>> (std::istream &in, universe &u);

    std::vector<CelestialBody> cbVec;
    ~universe();
 private:
    int planetNum;
```

```
    double universeR;
};
```

universe.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#include <vector>
#include "universe.hpp"  //NOLINT

universe::universe() {
    return;
}

std::istream& operator>> (std::istream &in, universe &u) {
    std::string planetNum, radius;

    in >> planetNum;
    in >> radius;

    u.planetNum = atoi(planetNum.c_str());
    u.universeR = atof(radius.c_str());

    std::cout << "Number of planets: " << u.planetNum << std::endl;
    std::cout << "Radius: " << u.universeR << std::endl;

    for (int i = 0 ; i < u.planetNum; i++) {
        CelestialBody* cb = new CelestialBody();
        in >> *cb;
        cb->setRadius(u.universeR);
        cb->setPos();
        u.cbVec.push_back(*cb);
        std::cout << *cb;
    }

    return in;
}

universe::~universe() {
    planetNum = 0;
    universeR = 0;
}
```

CelestialBody.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#pragma once
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

const int windowHeight = 500;
const int windowWidth = 500;

class CelestialBody: public sf::Drawable {
 public:
    CelestialBody();
    CelestialBody(double posiX,  double posiY, double veloX, \
        double veloY, double m, double r, std::string file);
```

```

    void setRadius(float radius);
    void setPos();

    friend std::istream& operator>> (std::istream &in, CelestialBody &c);
    friend std::ostream& operator<< (std::ostream &out, CelestialBody &c);

    ~CelestialBody();

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

    double posX;
    double posY;
    double velX;
    double velY;
    double mass;
    double radius;
    std::string name;

    sf::Image cbIm;
    sf::Sprite cbSp;
    sf::Texture cbTex;
};
```

CelestialBody.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#include "CelestialBody.hpp" //NOLINT

CelestialBody::CelestialBody() {}

CelestialBody::CelestialBody(double posiX, double posiY, double VeloX, \
    double veloY, double m, double r, std::string file) {
    posX = posiX;
    posY = posiY;
    velX = VeloX;
    velY = veloY;
    mass = m;

    if (!cbIm.loadFromFile(file)) {
        std::cout << "Cannot load the file" << std::endl;
        exit(1);
    }

    cbTex.loadFromImage(cbIm);
    cbSp.setTexture(cbTex);
    cbSp.setPosition(sf::Vector2f(posX, posY));
}

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

void CelestialBody :: setPos() {
    double x = (windowWidth / 2.5) * (posX / radius);
    double y = (windowWidth / 2.5) * (posY / radius);

    cbSp.setPosition(x + (windowWidth / 2), y + (windowHeight / 2));
}
```

```

void CelestialBody::draw(sf::RenderTarget& target, \
        sf::RenderStates states) const {
    target.draw(cbSp);
}

std::istream& operator>> (std:: istream &in, CelestialBody &c) {
    in >> c.posX;
    in >> c.posY;
    in >> c.velX;
    in >> c.velY;
    in >> c.mass;
    in >> c.name;

    if (!c.cbIm.loadFromFile(c.name)) {
        std::cout << "Cannot load the file for Celestial Body" << std::endl;
        return in;
    }

    c.cbTex.loadFromImage(c.cbIm);
    c.cbSp.setTexture(c.cbTex);
    c.cbSp.setPosition(sf::Vector2f(c.posX, c.posY));

    return in;
}

std::ostream& operator<< (std::ostream &out, CelestialBody &c) {
    out << "Name of the Planet: " << c.name << std::endl;
    out << "Position x: " << c.posX << std::endl;
    out << "Position y: " << c.posY << std::endl;
    out << "Velocity x: " << c.velX << std::endl;
    out << "Velocity y: " << c.velY << std::endl;
    out << "Mass : " << c.mass << std::endl;
    return out;
}

CelestialBody::~CelestialBody() {
    posX = 0;
    posY = 0;
    velX = 0;
    velY = 0;
    mass = 0;
    name.clear();
}
```

6 PS3b: N-Body Simulation

6.1 Discussion

This projects read a text file and register it to the classes and bring the files it needs and display on the window. Also, it calculates the numbers from the text and move the planets. I used leapfrog method to get acceleration, velocity and position for each planets. This was not hard because for accelaration I just have to devide force by mass and for velocity and position I have to multiply time.

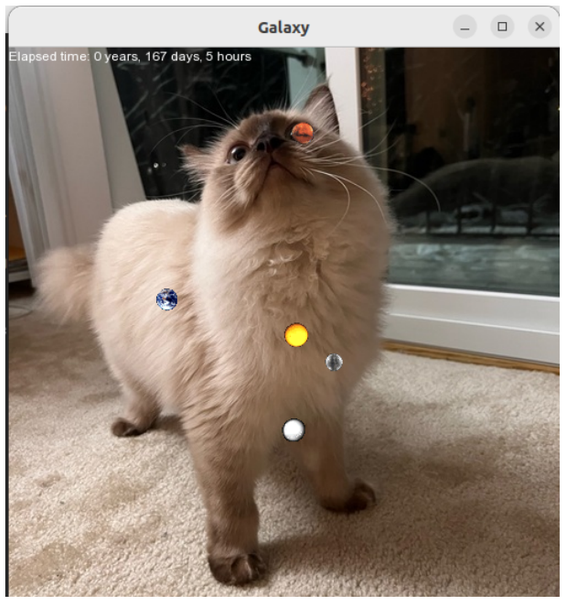


Figure 8: The program shows the planets in different positions

6.2 Places to get help

I got help from: SFML website, stackoverflow, c++.com, lecture slide: physics, smart-ptr

6.3 Challenges

I tried to use the unique pointer, but I could not make it work because push back did not work, so I ended up using shared pointer since push back worked like a normal vector. At first, in order to get force, sun is fixed argument and the planets have to be other arugments, then it moved the planets to the right. Therefore, I tried with all the planets and it worked properly. I made a lot of mutator and accessor function since member variables are private.

6.4 Mistakes

I got 2.5 points off and 10 percent. Since the program does not stop when it should, Newton's third law is not applied to my calculation, and vector for planets are not in private field. Also, about the 10 percent off, I turned the project in late for a day.

6.5 Extra Credit

I got 0.5 extra credits because I put the clock on the left top corner with units.

6.6 Codebase

Makefile

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

all: NBody lint
```

```

main.o: main.cpp universe.hpp
    $(CC) $(CFLAGS) -o $@ -c $<

universe.o: universe.cpp universe.hpp CelestialBody.hpp
    $(CC) $(CFLAGS) -c $<

CelestialBody.o: CelestialBody.cpp CelestialBody.hpp
    $(CC) $(CLFAGS) -c $<

NBody: main.o universe.o CelestialBody.o
    $(CC) $(CLFAGS) -o $@ $^ $(LIB)

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

clean:
    rm *.o NBody
```

main.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]

#include <string>
#include <sstream>
#include <SFML/Audio.hpp>
#include "universe.hpp" //NOLINT

template <typename T> std::string tString(const T &a) {
    std::ostringstream os;
    int year = a / 31536000;
    int afterY = a - (year * 31536000);
    int day = afterY / 86400;
    int hour = (a - day * 86400) / 3600;
    os << year << " years, " << day << " days, " << hour;
    os << " hours" << std::endl;
    return os.str();
}

int main(int argc, char* argv[]) {
        if (argc != 3) {
        std::cout << "Usage: ./NBody [double/ big t] " << std::endl;
        std::cout << "[double/ triangle t] < planets.txt " << std::endl;
        return 1;
    }

    std::string bigT(argv[1]);
    std::string triT(argv[2]);
    double simTime = 0;
    double bT = std::atoi(bigT.c_str());
    double tT = std::atoi(triT.c_str());

    universe* uni = new universe();

    std::cin >> *uni;

    sf::RenderWindow window(sf::VideoMode(windowWidth, windowHeight), "
   Galaxy");
    window.setFramerateLimit(120);

```

```
    sf::Font timeF;
    timeF.loadFromFile("arial.ttf");

    sf::Text timeT;
    timeT.setFont(timeF);
    timeT.setCharacterSize(12);
/*
    sf::Music music;
    if (!music.openFromFile("2001.ogg")) {
        std::cout << "cannot load the music" << std::endl;
        exit(1);
    }

    music.play();
    music.setLoop(true);
*/
    sf::Texture laylaTex;
    laylaTex.loadFromFile("Layla.jpg");
    if (!laylaTex.loadFromFile("Layla.jpg")) {  // Background image
     std::cout << "Cannot load Layla.jpg" << std::endl;
        exit(1);
    }

    sf::Sprite laylaSp;
    laylaSp.setTexture(laylaTex);
    laylaSp.setPosition(sf::Vector2f(0, 0));

    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed) {
                window.close();
            }
        }
        window.clear();
        window.draw(laylaSp);  // Draws Layla in the backgroud

        timeT.setString("Elapsed time: " + tString(simTime));
        window.draw(timeT);

        double forceX, forceY;

        std::vector<std::shared_ptr<CelestialBody>>::iterator a, b;
        for ( a = uni->cbVec.begin() ; a != uni->cbVec.end() ; a++ ) {
            forceX = 0;
            forceY = 0;
                for ( b = uni->cbVec.begin() ; b != uni->cbVec.end() ; b++ )
    {
                    if (a != b) {
                        forceX += findForX(*(*a), *(*b));
                        forceY += findForY(*(*a), *(*b));
                    }
                    (*a)->setFor(forceX, forceY);
                }
        }

        std::vector<std::shared_ptr<CelestialBody>>::iterator it;
        for ( it = uni->cbVec.begin() ; it != uni->cbVec.end() ; it++ ) {
            window.draw(*(*it));
```

```
        }
        uni->step(tT);

        window.display();

        simTime += tT;
        if (simTime == bT) break;  // end
    }

    std::cout << std::endl << std::endl;
    std::cout << "Final State: \n";
    std::vector<std::shared_ptr<CelestialBody>>::iterator it;
    for (it = uni->cbVec.begin(); it != uni->cbVec.end(); it++) {
        std::cout << *(*it) << std::endl;
    }

    return 0;
}
```

universe.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#pragma once
#include <memory>
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include "CelestialBody.hpp"  //NOLINT
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

class universe:public sf::Drawable {
 public:
    universe();

    friend std::istream& operator>> (std::istream &in, universe &u);
    std::vector<std::shared_ptr<CelestialBody>> cbVec;
    CelestialBody getPIndex(int index);
    void step(double seconds);

    int getPN() const;

    ~universe();
 private:
    int planetNum;
    double universeR;
    void virtual draw(sf::RenderTarget& target, \
      sf::RenderStates states) const {}
};
```

universe.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#include <vector>
#include <memory>
#include "universe.hpp"  //NOLINT

universe::universe() {
    return;
}
```

```

std::istream& operator>> (std::istream &in, universe &u) {
    std::string planetNum, radius;

    in >> planetNum;
    in >> radius;

    u.planetNum = atoi(planetNum.c_str());
    u.universeR = atof(radius.c_str());

    std::cout << "Number of planets: " << u.planetNum << std::endl;
    std::cout << "Radius: " << u.universeR << std::endl;

    for (int i = 0 ; i < u.planetNum; i++) {
        std::shared_ptr<CelestialBody> cb(new CelestialBody(u.universeR));
        in >> *cb;
        // cb->setRadius(cb->getPX() / u.universeR);
        cb->setPos();
        u.cbVec.push_back(cb);
        std::cout << *cb;
    }

    return in;
}

int universe::getPN() const {
    return planetNum;
}

CelestialBody universe::getPIndex(int index) {
    return *(cbVec[index]);
}


void universe::step(double seconds) {
    std::vector<std::shared_ptr<CelestialBody>>::iterator it, p;
    for (it = cbVec.begin(); it != cbVec.end(); it++) {
        (*it)->setAccX((*it)->getFX() / (*it)->getM());
        (*it)->setAccY((*it)->getFY() / (*it)->getM());

        (*it)->setVelX((*it)->getVX() + ((*it)->getAX() * seconds));
        (*it)->setVelY((*it)->getVY() - ((*it)->getAY() * seconds));

        (*it)->setPosX((*it)->getPX() + ((*it)->getVX() * seconds));
        (*it)->setPosY((*it)->getPY() - ((*it)->getVY() * seconds));

        (*it)->setPos();
    }
}


universe::~universe() {
    planetNum = 0;
    universeR = 0;
}
```

CelestialBody.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#pragma once
#include <iostream>
```

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

const int windowHeight = 500;
const int windowWidth = 500;
const double gravity = 6.67e-11;

class CelestialBody: public sf::Drawable {
 public:
    CelestialBody();
    CelestialBody(double posiX,  double posiY, double veloX, \
        double veloY, double m, double r, std::string file);
    CelestialBody(double r); //NOLINT
    void setRadius(float radius);
    void setPos();
    double getFX() const;
    double getFY() const;
    double getM() const;
    double getVX() const;
    double getVY() const;
    double getPX() const;
    double getPY() const;
    double getAX() const;
    double getAY() const;
    double getR() const;
    std::string getN() const;
    friend std::istream& operator>> (std::istream &in, CelestialBody &c);
    friend std::ostream& operator<< (std::ostream &out, CelestialBody &c);

    friend double findForX(CelestialBody &sun, CelestialBody &p);
    friend double findForY(CelestialBody &sun, CelestialBody &p);
    void setFor(double fX, double fY);
    void setAccX(double a);
    void setAccY(double a);
    void setVelX(double a);
    void setVelY(double a);
    void setPosX(double a);
    void setPosY(double a);

    ~CelestialBody();

 private:
    void virtual draw(sf::RenderTarget& target, sf::RenderStates states)
   const;
    double posX;
    double posY;
    double velX;
    double velY;
    double mass;
    double radius;
    std::string name;
    double accX;
    double accY;
    double forX;
```

```
    double forY;

    sf::Image cbIm;
    sf::Sprite cbSp;
    sf::Texture cbTex;
};
```

CelestialBody.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [6/2/2022]
#include "CelestialBody.hpp" //NOLINT

CelestialBody::CelestialBody() {}

CelestialBody::CelestialBody(double posiX, double posiY, double VeloX, \
    double veloY, double m, double r, std::string file) {
    posX = posiX;
    posY = posiY;
    velX = VeloX;
    velY = veloY;
    mass = m;

    if (!cbIm.loadFromFile(file)) {
        std::cout << "Cannot load the file" << std::endl;
        exit(1);
    }

    cbTex.loadFromImage(cbIm);
    cbSp.setTexture(cbTex);
    cbSp.setPosition(sf::Vector2f(posX, posY));
}

CelestialBody::CelestialBody(double r) {
    radius = r;
}

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

void CelestialBody::setPos() {
    double x = (windowWidth / 2.5) * (posX / radius);
    double y = (windowWidth / 2.5) * (posY / radius);

    cbSp.setPosition(x + (windowWidth / 2), y + (windowHeight / 2));
}

double CelestialBody::getFX() const {
    return forX;
}

double CelestialBody::getFY() const {
    return forY;
}

double CelestialBody::getM() const {
    return mass;
}

double CelestialBody::getVX() const {
    return velX;
```

```
}

double CelestialBody::getVY() const {
    return velY;
}

double CelestialBody::getPX() const {
    return posX;
}

double CelestialBody::getPY() const {
    return posY;
}

double CelestialBody::getAX() const {
    return accX;
}

double CelestialBody::getAY() const {
    return accY;
}

double CelestialBody::getR() const {
    return radius;
}

std::string CelestialBody::getN() const {
    return name;
}

void CelestialBody::draw(sf::RenderTarget& target, \
        sf::RenderStates states) const {
    target.draw(cbSp);
}

std::istream& operator>> (std:: istream &in, CelestialBody &c) {
    in >> c.posX;
    in >> c.posY;
    in >> c.velX;
    in >> c.velY;
    in >> c.mass;
    in >> c.name;

    if (!c.cbIm.loadFromFile(c.name)) {
        std::cout << "Cannot load the file for Celestial Body" << std::endl;
        return in;
    }

    c.cbTex.loadFromImage(c.cbIm);
    c.cbSp.setTexture(c.cbTex);
    c.cbSp.setPosition(sf::Vector2f(c.posX, c.posY));

    c.setAccX(0.f);
    c.setAccY(0.f);

    c.setFor(0.f, 0.f);

    return in;
}
```

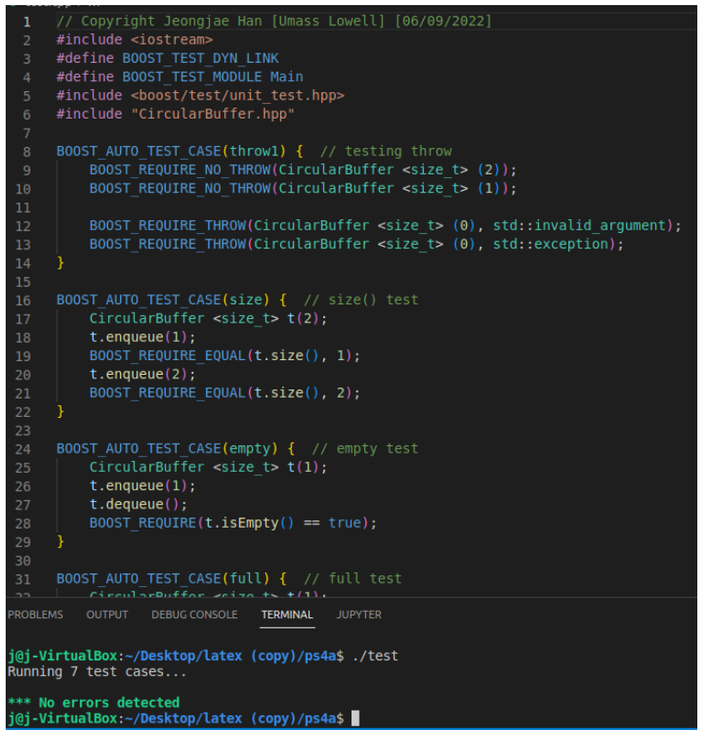
```

std::ostream& operator<< (std::ostream &out, CelestialBody &c) {
    out << "Name of the Planet: " << c.name << std::endl;
    out << "Position x: " << c.posX << std::endl;
    out << "Position y: " << c.posY << std::endl;
    out << "Velocity x: " << c.velX << std::endl;
    out << "Velocity y: " << c.velY << std::endl;
    out << "Mass : " << c.mass << std::endl;
    out << "Accelaration X: " << c.accX << std::endl;
    out << "Accelaration Y: " << c.accY << std::endl;
    out << "Force X: " << c.forX << std::endl;
    out << "Force Y: " << c.forY << std::endl;
    out << "Radius: " << c.radius << std::endl << std::endl;
    return out;
}

double findForX(CelestialBody &sun, CelestialBody &p) {
    double x, y, r2, root, force, forX, radius;
    /*if (p.posY == 0) {
        radius = p.posX - sun.posX;
        force = gravity * sun.mass * p.mass / pow(radius, 2);

    }*/

    x = p.posX - sun.posX;
    y = p.posY - sun.posY;

    r2 = pow(x, 2) + pow(y, 2);
    root = sqrt(r2);

    // F = (g * m1 * m2) / R^2;
    force = (gravity * sun.mass * p.mass) / r2;
    forX = force * (x / root);

    return forX;
}

double findForY(CelestialBody &sun, CelestialBody &p) {
    double x, y, r2, root, force, forY, radius;

    x = p.posX - sun.posX;
    y = p.posY - sun.posY;

    r2 = pow(x, 2) + pow(y, 2);
    root = sqrt(r2);
    // F = (g * m1 * m2) / r^2
    force = (gravity * sun.mass * p.mass) / r2;
    forY = force * (y / root);

    return forY;
}

void CelestialBody::setFor(double fX, double fY) {
    forX = fX;
    forY = fY;
}

void CelestialBody::setAccX(double a) {
    accX = a;
```

```
}

void CelestialBody::setAccY(double a) {
    accY = a;
}

void CelestialBody::setVelX(double a) {
    velX = a;
}

void CelestialBody::setVelY(double a) {
    velY = a;
}

void CelestialBody::setPosX(double a) {
    posX = a;
}

void CelestialBody::setPosY(double a) {
    posY = a;
}

CelestialBody::~CelestialBody() {
    posX = 0;
    posY = 0;
    velX = 0;
    velY = 0;
    mass = 0;
    accX = 0;
    accY = 0;
    forX = 0;
    forY = 0;
    name.clear();
}
```

7 PS4a: CircularBuffer

7.1 Discussion

This project is to prepare a header file for ps4b. we have to store a data into the container and use it for later. I used vector to store the datas because I am used to it. I used this for 2 other projects. For member variables I have 5 because we need vector to store, beg to find out the first vector, end for the last, and si for size of the vector and cap for capacity of the vector.



```
// Copyright Jeongjae Han [Umass Lowell] [06/09/2022]
#include <iostream>
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include "CircularBuffer.hpp"

BOOST_AUTO_TEST_CASE(throw1) {  // testing throw
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (2));
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (1));

    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::invalid_argument);
    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::exception);
}

BOOST_AUTO_TEST_CASE(size) {  // size() test
    CircularBuffer <size_t> t(2);
    t.enqueue(1);
    BOOST_REQUIRE_EQUAL(t.size(), 1);
    t.enqueue(2);
    BOOST_REQUIRE_EQUAL(t.size(), 2);
}

BOOST_AUTO_TEST_CASE(empty) {  // empty test
    CircularBuffer <size_t> t(1);
    t.enqueue(1);
    t.dequeue();
    BOOST_REQUIRE(t.isEmpty() == true);
}

BOOST_AUTO_TEST_CASE(full) {  // full test
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

j@j-VirtualBox:~/Desktop/latex (copy)/ps4a$ ./test
Running 7 test cases...

*** No errors detected
j@j-VirtualBox:~/Desktop/latex (copy)/ps4a$
```

Figure 9: Testing CircularBuffer.hpp

I made tests for all the functions I have for headerfile : Constructor, size(), isEmpty(), isFull(), peek(), enqueue(T item), and dequeue(). For exceptions I used invalid-argument and run-time error and the code throws the exceptions properly.

For the complexity for my CircularBuffer, it is O(1), since I have the member variable for begin and end, so it can be used as index.

7.2 Places to get help

I got help from: Lecture slide: template, stackoverflow, youtube.

7.3 What I learned

I also learned how to use the vector as private member variable. I was having hard time putting vector container as private member variable for previous projects.

For the construct argument, due to the type sizeT, negative value cannot be passed, so I tried to figure out how to throw an exception, but could not find out anywhere.

7.4 Challenges

I got used to using template more and for the constructor with this project. I have to declare argument's original datatype to pass it properly. It took a while to find it out.

7.5 Mistakes

I got one point off because my program has incorrect behavior when interleaving enqueue and dequeue operation.

7.6 Codebase

Makefile

```
CC = g++
CFLAGS = --std=c++14 -Wall -Werror -pedantic
DEPS = CircularBuffer.hpp
LIBS = -lboost_unit_test_framework

all: test lint

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

test: test.o
    $(CC) $(CFLAGS) -o $@ $< $(LIBS)

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

clean:
    rm *.o test
```

CircularBuffer.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [06/09/2022]
#pragma once
#include <iostream>
#include <string>
#include <sstream>
#include <exception>
#include <stdexcept>
#include <vector>

template <class T>
class CircularBuffer {
 public:
    explicit CircularBuffer(size_t capacity) {
        // Create an empty ring buffer with given max capacity
        if ( capacity < 1 ) {
            throw std::invalid_argument\
            ("Capacity must be bigger than 0.");
        } else {
            beg = 0;
            end = 0;
            cap = capacity;
            si = 0;
            cbVec.resize(capacity);
        }
    }

    size_t size() const {  // The number of items currently in the buffer
        return si;
    }

    bool isEmpty() const {  // Is the buffer is empty?
        if (si == 0 ) return true;
        return false;
    }

    bool isFull() const {  // Is the buffer full?
```

```
        if (si == cap) return true;
        return false;
    }

    void enqueue(T item) {  // Add item to the end
        if (isFull()) throw std::runtime_error("The buffer is full");

        if (end >= cap) end = 0;
        cbVec.at(si) = item;
        si++;
        end++;
    }

    T dequeue() {  // Delete and return item from the front
        if (isEmpty()) throw std::runtime_error\
            ("The buffer is already empty: dequeue()");

        T item = cbVec.at(beg);
        cbVec.at(beg) = 0;
        beg++;
        si--;
        if (beg >= cap) beg = 0;

        return item;
    }

    T peek() const {  // Return (but do not delete) item from the front
        if (isEmpty()) throw std::runtime_error\
            ("The buffer is empty: no peek()");
        return cbVec.at(beg);
    }

    ~CircularBuffer() {
        cbVec.clear();
        beg = 0;
        end = 0;
        cap = 0;
        si = 0;
    }

 private:
    std::vector<T> cbVec;
    int beg;  // first index
    int end;  // last index
    int cap;
    int si;
};
```

test.cpp

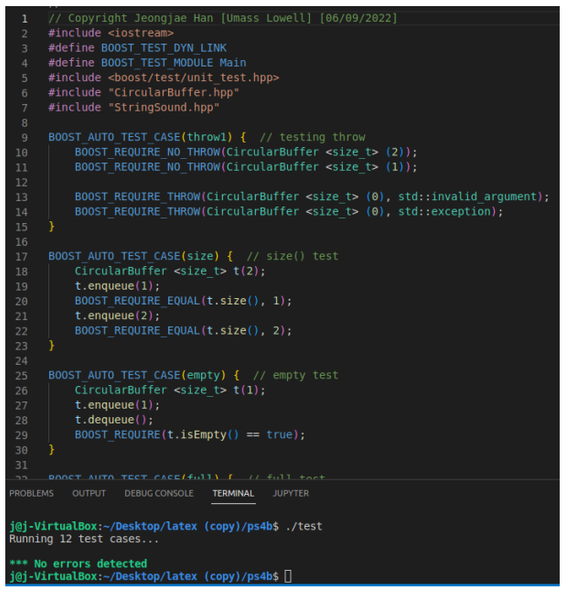
```
// Copyright Jeongjae Han [Umass Lowell] [06/09/2022]
#include <iostream>
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include "CircularBuffer.hpp"

BOOST_AUTO_TEST_CASE(throw1) {  // testing throw
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (2));
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (1));

```

```
    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::invalid_argument);
    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::exception);
}

BOOST_AUTO_TEST_CASE(size) {  // size() test
    CircularBuffer <size_t> t(2);
    t.enqueue(1);
    BOOST_REQUIRE_EQUAL(t.size(), 1);
    t.enqueue(2);
    BOOST_REQUIRE_EQUAL(t.size(), 2);
}

BOOST_AUTO_TEST_CASE(empty) {  // empty test
    CircularBuffer <size_t> t(1);
    t.enqueue(1);
    t.dequeue();
    BOOST_REQUIRE(t.isEmpty() == true);
}

BOOST_AUTO_TEST_CASE(full) {  // full test
    CircularBuffer <size_t> t(1);
    BOOST_REQUIRE(t.isFull() == false);
    t.enqueue(1);
    BOOST_REQUIRE(t.isFull() == true);
}

BOOST_AUTO_TEST_CASE(enqueue) {  // checking enqueue function errors
    CircularBuffer <size_t> t(1);
    BOOST_REQUIRE_NO_THROW(t.enqueue(1));
    BOOST_REQUIRE_THROW(t.enqueue(1), std::runtime_error);
}

BOOST_AUTO_TEST_CASE(dequeue) {  // dequeue() test
    CircularBuffer <size_t> t(1);
    t.enqueue(1);
    BOOST_REQUIRE(t.dequeue() == 1);
    BOOST_REQUIRE_THROW(t.dequeue(), std::runtime_error);
}

BOOST_AUTO_TEST_CASE(peek) {
    CircularBuffer <size_t> t(3);
    t.enqueue(4);
    t.enqueue(1);
    BOOST_REQUIRE_EQUAL(t.peek(), 4);
}
```

8 PS4b: StringSound

8.1 Discussion

This project uses the Karplus-Strong algorithm to simulate the plucking of a guitar, CircularBuffer.hpp that I made for ps4a to store the keys and notes. Therefore, you keyboard become a piano.

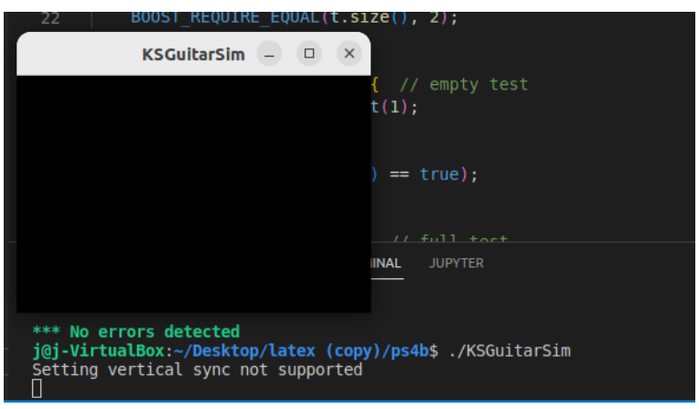


```
// Copyright Jeongjae Han [Umass Lowell] [06/09/2022]
#include <iostream>
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include "CircularBuffer.hpp"
#include "StringSound.hpp"

BOOST_AUTO_TEST_CASE(throw1) {  // testing throw
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (2));
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (1));

    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::invalid_argument);
    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::exception);
}

BOOST_AUTO_TEST_CASE(size) {  // size() test
    CircularBuffer <size_t> t(2);
    t.enqueue(1);
    BOOST_REQUIRE_EQUAL(t.size(), 1);
    t.enqueue(2);
    BOOST_REQUIRE_EQUAL(t.size(), 2);
}

BOOST_AUTO_TEST_CASE(empty) {  // empty test
    CircularBuffer <size_t> t(1);
    t.enqueue(1);
    t.dequeue();
    BOOST_REQUIRE(t.isEmpty() == true);
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

j@j-VirtualBox:~/Desktop/latex (copy)/ps4b$ ./test
Running 12 test cases...

*** No errors detected
j@j-VirtualBox:~/Desktop/latex (copy)/ps4b$
```

Figure 10: Testing the hpp, and cpp of this project

I wrote the tests for all the public functions: Constructors, pluck, time, tic, and sample. I constructor to throw invalid-argument exception when the user tries to put invalid-argument. Other fuctions passed the test.



```
KSGuitarSim
*** No errors detected
j@j-VirtualBox:~/Desktop/latex (copy)/ps4b$ ./KSGuitarSim
Setting vertical sync not supported
```

Figure 11: The window of KSGuitarSim it runs as it should

Cannot put audio on this file, so I cannot prove that the program makes right sound, but I want to show that when you run it it will show black window.

8.2 Places to get help

Lecture slide to understand the calculation and physics. cppreference

8.3 Challenges

Instead of switch, I made the program finds and recognizes the input and finds the right note. For the lambda expression, I tried the template on the lecture slide, but it was not working. Therefore, I googled it and cppreference showed me different ways to use it and found a syntax that works for this.

8.4 Mistakes

I got 0.5 points off because I used lambda for Karplus-Strong algorithm. In order to get the return value of tick, I needed to use the average of two items that are stored in CircularBuffer and multiply by decay rate. I used the lambda to get average of two items. I did not use any algorithm functions because it was easier to code by myself. However, I had to call lambda by passing it as an argument, but the way I used was calling lambda directly.

8.5 Codebase

Makefile

```
CC = g++
CFLAGS = --std=c++14 -Wall -Werror -pedantic
SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
DEPS = CircularBuffer.hpp StringSound.hpp
LIBS = -lboost_unit_test_framework


all: KSGuitarSim test lint

StringSound.o: StringSound.cpp
    $(CC) $(CFLAGS) -o $@ -c $<

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

KSGuitarSim: KSGuitarSim.o StringSound.o
    $(CC) -o $@ $^ $(SFLAGS)

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

test: test.o StringSound.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIBS)

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

clean:
    rm *.o test KSGuitarSim
```

KSGuitarSim.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [06/12/2022]
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>
#include <SFML/Window.hpp>

#include "StringSound.hpp"

#define CONCERT_A 220.0
#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.tick();
        samples.push_back(gs.sample());
    }
    return samples;
}

int main() {
    sf::RenderWindow window(sf::VideoMode(300, 200), "KSGuitarSim");
    sf::Event event;
    std::string keys = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";

    double freq;

    std::vector<sf::Int16> samples[37];
    sf::Sound sound1[37];
    sf::SoundBuffer buf1[37];

    for (int i = 0; i < 37 ; i++) {
        freq = 440 * pow(2.0, (i-24) / 12.0);
        StringSound gs(freq);
        samples[i] = makeSamples(gs);
        // std::cout << samples[i].size() << std::endl;
        buf1[i].loadFromSamples(&samples[i][0], samples[i].size(), 2, 44100);
        sound1[i].setBuffer(buf1[i]);
    }

    while (window.isOpen()) {
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed) {
                window.close();
            } else if (event.type == sf::Event::TextEntered) {
                int i = keys.find(event.text.unicode);
                if (i >= 0 && i <= 36) {
                    sound1[i].play();
                }
            }
        }
        window.clear();
        window.display();
    }
    return 0;
}
```

CircularBuffer.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [06/09/2022]
#pragma once
#include <iostream>
#include <string>
#include <sstream>
#include <exception>
#include <stdexcept>
#include <vector>

template <class T>
class CircularBuffer {
 public:
    explicit CircularBuffer(size_t capacity) {
        // Create an empty ring buffer with given max capacity
```

```
        if ( capacity < 1 ) {
            throw std::invalid_argument\
            ("Capacity must be bigger than 0.");
        } else {
            beg = 0;
            end = 0;
            cap = capacity;
            si = 0;
            cbVec.resize(capacity);
        }
    }

    size_t size() const {  // The number of items currently in the buffer
        return si;
    }

    bool isEmpty() const {  // Is the buffer is empty?
        if (si == 0 ) return true;
        return false;
    }

    bool isFull() const {  // Is the buffer full?
        if (si == cap) return true;
        return false;
    }

    void enqueue(size_t item) {  // Add item to the end
        if (isFull()) throw std::runtime_error("The buffer is full");

        if (end >= cap) end = 0;
        cbVec.at(end) = item;
        si++;
        end++;
    }

    T dequeue() {  // Delete and return item from the front
        if (isEmpty()) throw std::runtime_error\
            ("The buffer is already empty: dequeue()");

        T item = cbVec.at(beg);
        cbVec.at(beg) = 0;
        beg++;
        si--;
        if (beg >= cap) beg = 0;

        return item;
    }

    T peek() const {  // Return (but do not delete) item from the front
        if (isEmpty()) throw std::runtime_error\
            ("The buffer is empty: no peek()");
        return cbVec.at(beg);
    }

    ~CircularBuffer() {
        cbVec.clear();
        beg = 0;
        end = 0;
        cap = 0;
```

```
        si = 0;
    }

 private:
    std::vector<T> cbVec;
    int beg;  // first index
    int end;  // last index
    int cap;
    int si;
};
```

StringSound.hpp

```
// Copyright Jeongjae Han [Umass Lowell] [06/11/2022]
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
#include <random>
#include <SFML/Audio.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
#include "CircularBuffer.hpp"

const int rate = 44100;
const double decay = 0.996;

class StringSound {
 public:
    explicit StringSound(double frequency);
      // Create a guitar string sound of the given
      // frequency using a sampling rate of 44,100
    explicit StringSound(std::vector<sf::Int16> init);
      // Create a guitar string with size and initial values given by the vector
    StringSound(const StringSound& obj) = delete;  // No copy constructor
    void pluck();
      // Pluck the guitar string by replacing the buffer with random values,
      // representing the white noise
    void tick();  // Advance the simulation one time step
    sf::Int16 sample() const;  // Return the current sample
    size_t time() const;  // Return the number of times tick was called so far
    ~StringSound();

 private:
    CircularBuffer<size_t> cb;
    int t;
};
```

StringSound.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [06/11/2022]
#include "StringSound.hpp"

StringSound::StringSound(double frequency): \
        cb((ceil(rate / frequency))) {
    // cb = new CircularBuffer(static_cast<size_t> (ceil(44100 / frequency)));
    if (frequency <= 0) throw std::invalid_argument\
```

```
            ("Frequency must be higher than 0.");

    t = 0;
}

StringSound::StringSound(std::vector<sf::Int16> init): \
        cb(static_cast<size_t> (init.size())) {
    std::vector<sf::Int16>::iterator it;
    for (it = init.begin(); it < init.end(); it++) {
        cb.enqueue((int16_t)*it);
    }

    t = 0;
}

void StringSound::pluck() {
    while (!cb.isEmpty()) {
        cb.dequeue();
    }

    while (!cb.isFull()) {
        std::random_device rd;
        std::mt19937 rng(rd());
        std::uniform_int_distribution<int> \
        distribution(-32768, 32767);
        sf::Int16 ranDom = distribution(rng);
        cb.enqueue(ranDom &0xffff);
    }
}

void StringSound::tick() {
    int16_t first = cb.dequeue();
    int16_t second = cb.peek();
    int16_t karplus = decay * [&] {return (first + second) / 2;}();

    cb.enqueue((sf::Int16)karplus);
    t++;
}

sf::Int16 StringSound::sample() const {
    sf::Int16 sample = (sf::Int16)cb.peek();

    return sample;
}

size_t StringSound::time() const {
    return t;
}

StringSound::~StringSound() {
    // cb.~CircularBuffer();
    t = 0;
}
```

test.cpp

```
// Copyright Jeongjae Han [Umass Lowell] [06/09/2022]
#include <iostream>
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
```

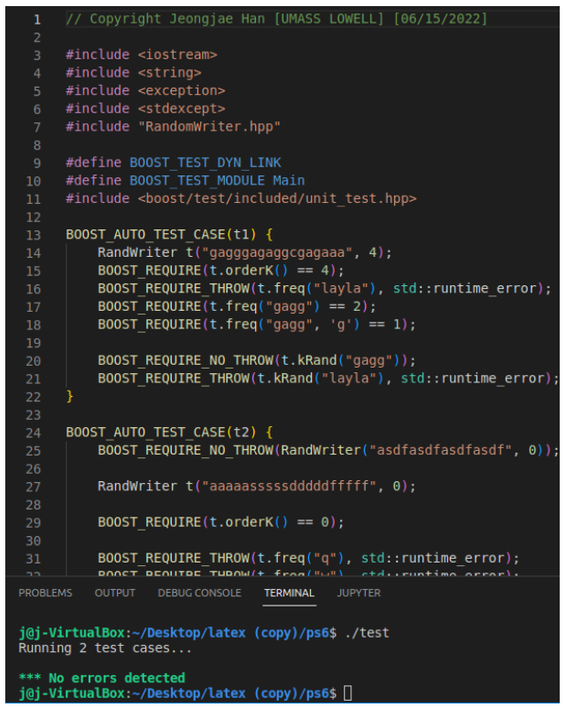
```
#include "CircularBuffer.hpp"
#include "StringSound.hpp"

BOOST_AUTO_TEST_CASE(throw1) {  // testing throw
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (2));
    BOOST_REQUIRE_NO_THROW(CircularBuffer <size_t> (1));

    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::invalid_argument);
    BOOST_REQUIRE_THROW(CircularBuffer <size_t> (0), std::exception);
}

BOOST_AUTO_TEST_CASE(size) {  // size() test
    CircularBuffer <size_t> t(2);
    t.enqueue(1);
    BOOST_REQUIRE_EQUAL(t.size(), 1);
    t.enqueue(2);
    BOOST_REQUIRE_EQUAL(t.size(), 2);
}

BOOST_AUTO_TEST_CASE(empty) {  // empty test
    CircularBuffer <size_t> t(1);
    t.enqueue(1);
    t.dequeue();
    BOOST_REQUIRE(t.isEmpty() == true);
}

BOOST_AUTO_TEST_CASE(full) {  // full test
    CircularBuffer <size_t> t(1);
    BOOST_REQUIRE(t.isFull() == false);
    t.enqueue(1);
    BOOST_REQUIRE(t.isFull() == true);
}

BOOST_AUTO_TEST_CASE(enqueue) {  // checking enqueue function errors
    CircularBuffer <size_t> t(1);
    BOOST_REQUIRE_NO_THROW(t.enqueue(1));
    BOOST_REQUIRE_THROW(t.enqueue(1), std::runtime_error);
}

BOOST_AUTO_TEST_CASE(dequeue) {  // dequeue() test
    CircularBuffer <size_t> t(1);
    t.enqueue(1);
    BOOST_REQUIRE(t.dequeue() == 1);
    BOOST_REQUIRE_THROW(t.dequeue(), std::runtime_error);
}

BOOST_AUTO_TEST_CASE(peek) {
    CircularBuffer <size_t> t(3);
    t.enqueue(4);
    t.enqueue(1);
    BOOST_REQUIRE_EQUAL(t.peek(), 4);
}

BOOST_AUTO_TEST_CASE(const1) {  // check constructor
    BOOST_REQUIRE_NO_THROW(StringSound(2));
    BOOST_REQUIRE_NO_THROW(StringSound \
        t(std::vector<sf::Int16> init = {1, 2, 3}));
    BOOST_REQUIRE_THROW(StringSound(0), std::invalid_argument);
}
```

```

BOOST_AUTO_TEST_CASE(time1) {  // time check
    StringSound t(3.f);
    BOOST_REQUIRE_EQUAL(t.time(), 0);
}

BOOST_AUTO_TEST_CASE(sample1) {
    std::vector<sf::Int16> init = {1, 2, 3};
    StringSound t(init);
    BOOST_REQUIRE_EQUAL(t.sample(), 1);
}

BOOST_AUTO_TEST_CASE(pluck1) {  // pluck
    std::vector<sf::Int16> init = {1, 2, 3};
    StringSound t(init);
    t.pluck();
    BOOST_REQUIRE_NE(t.sample(), 1);
}

BOOST_AUTO_TEST_CASE(tic) {  // tic()
    std::vector<sf::Int16> init = {1, 2, 3};
    StringSound t(init);
    t.tick();
    BOOST_REQUIRE_EQUAL(t.time(), 1);
}
```

9 PS6: Random Writer

9.1 Discussion

This program takes a string and two integers. one of the integer decides how many characters the map is going to store from the string. Then, interacts with the map to randomly produce a character or a string.



```
// Copyright Jeongjae Han [UMASS LOWELL] [06/15/2022]

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include "RandomWriter.hpp"

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

BOOST_AUTO_TEST_CASE(t1) {
    RandWriter t("gagggagaggcgagaaa", 4);
    BOOST_REQUIRE(t.orderK() == 4);
    BOOST_REQUIRE_THROW(t.freq("layla"), std::runtime_error);
    BOOST_REQUIRE(t.freq("gagg") == 2);
    BOOST_REQUIRE(t.freq("gagg", 'g') == 1);

    BOOST_REQUIRE_NO_THROW(t.kRand("gagg"));
    BOOST_REQUIRE_THROW(t.kRand("layla"), std::runtime_error);
}

BOOST_AUTO_TEST_CASE(t2) {
    BOOST_REQUIRE_NO_THROW(RandWriter("asdfasdfasdfasdf", 0));

    RandWriter t("aaaaassssssddddddfffff", 0);

    BOOST_REQUIRE(t.orderK() == 0);

    BOOST_REQUIRE_THROW(t.freq("q"), std::runtime_error);
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

j@j-VirtualBox:~/Desktop/latex (copy)/ps6$ ./test
Running 2 test cases...

*** No errors detected
j@j-VirtualBox:~/Desktop/latex (copy)/ps6$
```

Figure 12: Testing ps6 codes

My test has no error. I made two tests. My test.cpp file tests all the public functions, and exception, but generate function because I am not sure how to make a test for a generate which generates a random string that I cannot even predict the result.

```
// Copyright Jeongjae Han [UMASS LOWELL] [06/15/2022]

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include "RandomWriter.hpp"

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

BOOST_AUTO_TEST_CASE(t1) {
    RandWriter t("gagggagaggcgagaaa", 4);
    BOOST_REQUIRE(t.orderK() == 4);
    BOOST_REQUIRE_THROW(t.freq("layla"), std::runtime_error);
    BOOST_REQUIRE(t.freq("gagg") == 2);
    BOOST_REQUIRE(t.freq("gagg", 'g') == 1);

    BOOST_REQUIRE_NO_THROW(t.kRand("gagg"));
    BOOST_REQUIRE_THROW(t.kRand("layla"), std::runtime_error);
}

BOOST_AUTO_TEST_CASE(t2) {
    BOOST_REQUIRE_NO_THROW(RandWriter("asdfasdfasdfasdf", 0));

    RandWriter t("aaaaassssssddddddfffff", 0);

    BOOST_REQUIRE(t.orderK() == 0);

    BOOST_REQUIRE_THROW(t.freq("q"), std::runtime_error);
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

j@j-VirtualBox:~/Desktop/latex (copy)/ps6$ ./TextWriter
Format ./TextWriter [number for K] [number for L]
j@j-VirtualBox:~/Desktop/latex (copy)/ps6$ ./TextWriter 2 11 <input17.tx
gaggcgaggcg
j@j-VirtualBox:~/Desktop/latex (copy)/ps6$
```

Figure 13: Testing ps6 codes

I was not sure about input17.txt, so I created input17.txt and it has example string that the prompt provided.

I tried to use size-t for index for for-loops because we covered it in during the class.

Also, for iterator, I used auto p instead of ::iterator p and for-each loop because the map was declared as private member of the class, so the compiler was asking for operator functions.

9.2 Places to get help

I got help from Youtube, Comp3 assignments, stackoverflow, c++reference, how to loop through a map.pdf.

9.3 What I learned

I learned how I should interact with a map. For comp3, there was an assignment that I have to use map, and it helped me to get to know about map, but this project helped me to get used to it. I also referred the assignment for this project.

9.4 Mistakes

I got three points off. I could not use lambda expression as parameter for this project, I tried to use it in the kRand. I made a random function by using lambda expression and declaring a variable as auto, but I am not sure if this is working as a parameter. My kRand did not generate all expected characters. Also, my output was not reasonable.

9.5 Codebase

Makefile

```
CC = g++
CFLAGS = --std=c++14 -Wall -Werror -pedantic
SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
DEPS = RandomWriter.hpp
LIBS = -lboost_unit_test_framework


all: TextWriter test lint

RandomWriter.o: RandomWriter.cpp
    $(CC) $(CFLAGS) -o $@ -c $<

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

TextWriter: TextWriter.o RandomWriter.o
    $(CC) -o $@ $^

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

test: test.o RandomWriter.o
    $(CC) $(CFLAGS) -o $@ $^ $(LIBS)

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

clean:
    rm *.o test TextWriter
```

TextWriter.cpp

```
// Copyright Jeongjae Han [UMASS LOWELL] [06/15/2022]

```

```
#include <iostream>
#include <string>
#include "RandomWriter.hpp"

int main(int argc, const char* argv[]) {
    if (argc != 3) {
        std::cout << "Format ./TextWriter [number for K] [number for L] \n";
        exit(1);
    }

    int k = std::stoi(argv[1]);
    int l = std::stoi(argv[2]);

    std::string input;
    std::string text;

    while (std::getline(std::cin, text)) {
        input.append(text);
        input.append(1, ' ');
    }

    RandWriter rw(input, k);
    // std::cout << "Number of 'ga': " << rw.freq("ga") << std::endl;
    // std::cout << "Number of 'gc': " << rw.freq("gc") << std::endl;
    // std::cout << "Number of 'ab': " << rw.freq("ab") << std::endl;
    // std::cout << "Random character produced: " << rw.kRand("ga") << std::endl;
    std::cout << rw.generate(input.substr(0, k), l) << std::endl;
}
```

RandomWriter.hpp

```
// Copyright Jeongjae Han [UMASS LOWELL] [06/15/2022]
#pragma once

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

class RandWriter {
 public:
    // Create a Markov model of order k from given text
    // Assume that text has length at least k.
    RandWriter(std::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(std::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(std::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(std::string kgram);
    // Generate a string of length L characters by simulating a trajectory
    // through the corresponding Markov chain.  The first k characters of
```

```
    // the newly generated string should be the argument kgram.
    // Throw an excpetion if kgram is not of length k.
    // Assume that L is at least k
    std::string getA() const;
    std::string generate(std::string kgram, int L);
    friend std::ostream& operator<<(std::ostream& out, const RandWriter &r);

    auto begin() const;
    auto end() const;
    ~RandWriter();

 private:
    int order;
    std::string alphabet;
    std::map <std::string, int> kMap;
};
// Overload the stream insertion operator << and display the internal state
// of the Markov model.  Print out the order, alphabet, and the frequencies
// of the k-grams and k+1-grams
```

RandomWriter.cpp

```
// Copyright Jeongjae Han [UMASS LOWELL] [06/15/2022]
#include "RandomWriter.hpp"

RandWriter::RandWriter(std::string text, int n): order(n) {
    // Create a Markov model of order k from given text
    // Assume that text has length at least k.

    std::string str = text;

    for ( int i = 0; i < order; i++ ) {
        str.push_back(text[i]);
    }

    char temp;
    bool taken = false;

    for (size_t i = 0; i < text.length(); i++) {
        temp = text.at(i);
        taken = false;
        for (size_t j = 0; j < alphabet.length(); j++) {
            if (alphabet.at(j) == temp) taken = true;
        }
        if (!taken) alphabet.push_back(temp);
    }

    std::string tempStr;

    for (int i = order; i <= order + 1; i++) {
        for (size_t j = 0; j < text.length(); j++) {
            tempStr.clear();
            tempStr = str.substr(j, i);

            kMap.insert(std::pair<std::string, int>(tempStr, 0));
        }
    }

    int count = 0;

    for (int i = order; i <= order + 1; i++) {
```

```
        for (size_t j = 0; j < text.length(); j++) {
            tempStr.clear();
            tempStr = str.substr(j, i);

            auto p = kMap.find(tempStr);
            count = p->second;
            count++;

            kMap[tempStr] = count;
        }
    }
}

int RandWriter::orderK() const {
     return order;
}

int RandWriter::freq(std::string kgram) const {
    if (kgram.size() == static_cast<size_t>(order)) {
        auto p = kMap.find(kgram);

        if (p == kMap.end()) return 0;

        return p->second;
    } else {
        throw std::runtime_error\
            ("for frequency, provided string is not right size.");
    }
}

int RandWriter::freq(std::string kgram, char c) const {
    if (kgram.size() == static_cast<size_t>(order)) {
        kgram.push_back(c);
        auto p = kMap.find(kgram);
        if (p == kMap.end()) {
            return 0;
        } else {
            return p->second;
        }
    } else {
        throw std::runtime_error\
            ("for frequency2, provided string is not right");
    }
}

char RandWriter::kRand(std::string kgram) {
    if (kgram.length() != static_cast<size_t>(order))\
        throw std::runtime_error\
        ("for krand your kgram is wrong");

    srand((int)time(NULL));  // NOLINT

    int kgramF = freq(kgram);
    int ranVal = rand() % kgramF; //NOLINT
    double test = 0;
    auto creatRan = [=] () \
        {return static_cast<double>(ranVal) / kgramF; };
    double ranNum = creatRan();
    double lVal = 0;
```

```

    for (size_t i = 0; i < alphabet.length(); i++) {
        test = static_cast<double> \
            (freq(kgram, alphabet[i])) / kgramF;

        if ((ranNum < (test + lVal)) && test != 0) {
            return alphabet[i];
        }
        lVal += test;
    }
    return 0;
}

std::string RandWriter::generate(std::string kgram, int L) {
    if (kgram.length() != static_cast<size_t>(order)) throw std::
   runtime_error\
        ("generate string kgram is wrong");

    std::string finStr = "";
    char retChar;

    finStr += "" + kgram;

    for ( int i = 0 ; i < (L - order); i++ ) {
        retChar = kRand(finStr.substr(i, order));
        finStr.push_back(retChar);
    }
    return finStr;
}

std::string RandWriter::getA() const {
    return alphabet;
}

auto RandWriter::begin() const {
    return kMap.begin();
}

auto RandWriter::end() const {
    return kMap.end();
}

std::ostream& operator<< (std::ostream &out, RandWriter &rw) {
    out << "Order: " << rw.orderK() << std::endl;
    out << "Alphabet: " << rw.getA() << std::endl;
    out << "Map: ";

    for (auto p = rw.begin(); p != rw.end(); p++) {
        out << p->first << "\t" << p->second << std::endl;
    }
    return out;
}

RandWriter::~RandWriter() {
    order = 0;
    alphabet.clear();
}
```

test.cpp

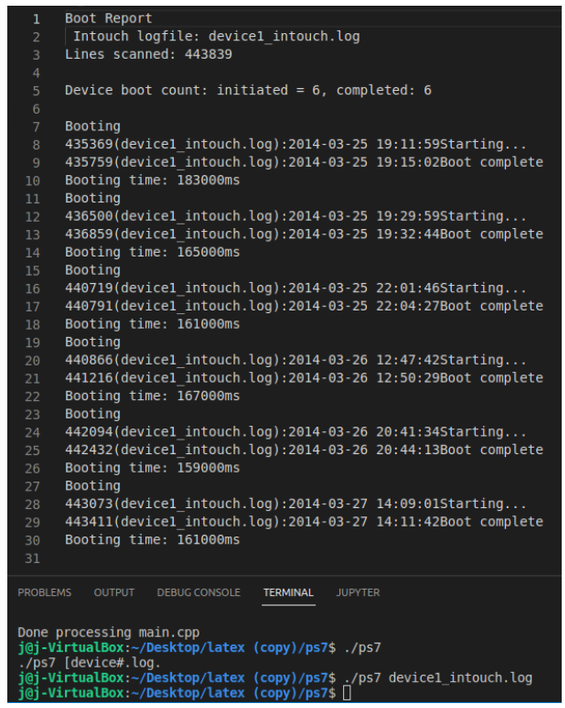
```
// Copyright Jeongjae Han [UMASS LOWELL] [06/15/2022]
```

```

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include "RandomWriter.hpp"

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

BOOST_AUTO_TEST_CASE(t1) {
    RandWriter t("gagggagaggcgagaaa", 4);
    BOOST_REQUIRE(t.orderK() == 4);
    BOOST_REQUIRE_THROW(t.freq("layla"), std::runtime_error);
    BOOST_REQUIRE(t.freq("gagg") == 2);
    BOOST_REQUIRE(t.freq("gagg", 'g') == 1);

    BOOST_REQUIRE_NO_THROW(t.kRand("gagg"));
    BOOST_REQUIRE_THROW(t.kRand("layla"), std::runtime_error);
}

BOOST_AUTO_TEST_CASE(t2) {
    BOOST_REQUIRE_NO_THROW(RandWriter("asdfasdfasdfasdf", 0));

    RandWriter t("aaaaassssssdddddfffff", 0);

    BOOST_REQUIRE(t.orderK() == 0);

    BOOST_REQUIRE_THROW(t.freq("q"), std::runtime_error);
    BOOST_REQUIRE_THROW(t.freq("w"), std::runtime_error);

    BOOST_REQUIRE(t.freq("") == 20);
    BOOST_REQUIRE(t.freq("", 'a') == 5);
    BOOST_REQUIRE(t.freq("", 'e') == 0);
}
```

10 PS7: Kronos Log Parsing

10.1 Discussion

This project calls a log file from a Kronos InTouch time clock by using regex library expressions and makes a .rpt file to leave new logs with the time by using boost date time libraries.



```
Boot Report
 Intouch logfile: device1_intouch.log
Lines scanned: 443839

Device boot count: initiated = 6, completed: 6

Booting
435369(device1_intouch.log):2014-03-25 19:11:59Starting...
435759(device1_intouch.log):2014-03-25 19:15:02Boot complete
Booting time: 183000ms
Booting
436500(device1_intouch.log):2014-03-25 19:29:59Starting...
436859(device1_intouch.log):2014-03-25 19:32:44Boot complete
Booting time: 165000ms
Booting
440719(device1_intouch.log):2014-03-25 22:01:46Starting...
440791(device1_intouch.log):2014-03-25 22:04:27Boot complete
Booting time: 161000ms
Booting
440866(device1_intouch.log):2014-03-26 12:47:42Starting...
441216(device1_intouch.log):2014-03-26 12:50:29Boot complete
Booting time: 167000ms
Booting
442094(device1_intouch.log):2014-03-26 20:41:34Starting...
442432(device1_intouch.log):2014-03-26 20:44:13Boot complete
Booting time: 159000ms
Booting
443073(device1_intouch.log):2014-03-27 14:09:01Starting...
443411(device1_intouch.log):2014-03-27 14:11:42Boot complete
Booting time: 161000ms

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

Done processing main.cpp
j@j-VirtualBox:~/Desktop/latex (copy)/ps7$ ./ps7
./ps7 [device#.log.
j@j-VirtualBox:~/Desktop/latex (copy)/ps7$ ./ps7 device1_intouch.log
j@j-VirtualBox:~/Desktop/latex (copy)/ps7$
```

Figure 14: Output .rpt file

I declared using since for the date functions, there is a lot of things follow with it and stackoverflow suggested me to use it. For regex, it is not long as date, so I did not use using.

I used regular expressions to find matching string from the log file. boost::regex-match or boost::regex-search were used to find a match against the regular expressions previously created.

10.2 Places to get help

I got help from Regex lecture slide, stackoverflow, c++.com

10.3 Challenges

The string for regex was a struggle due to the typo I spend too much time finding out my mistake.

10.4 Mistakes

I got two points off because my formatting had problem, and did not describe regexes in readme.

10.5 Extra Credit

I got 0.5 extra credit because my program had header but not mentioned in readme.

10.6 Codebase

Makefile

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

```
RFLAGS = -lboost_regex -lboost_date_time
DEPS = RandomWriter.hpp
LIBS = -lboost_unit_test_framework


all: ps7 lint

main.o: main.cpp
    $(CC) $(CFLAGS) -o $@ -c $<

ps7: main.o
    $(CC) -o $@ $^ $(RFLAGS)

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

clean:
    rm *.o ps7 *.rpt
```

main.cpp

```
// Copyright Jeongjae Han   [UMASS LOWELL] [06/19/2022]
#include <iostream>
#include <string>
#include <fstream>
#include <boost/regex.hpp>
#include "boost/date_time/gregorian/gregorian.hpp"
#include "boost/date_time/posix_time/posix_time.hpp"

using boost::gregorian::date;
using boost::gregorian::from_simple_string;
using boost::gregorian::date_duration;
using boost::gregorian::date_period;
using boost::posix_time::time_duration;
using boost::posix_time::ptime;
using std::string;

int main(int argc, const char* argv[]) {
    if (argc != 2) {
        std::cout << "./ps7 [device#.log. " << std::endl;
        exit(1);
    }

    int success = 0;
    int numBoot = 0;
    int counter1 = 1;

    string input(argv[1]);
    string output = input + ".rpt";
    string bDate = "", eDate = "", cDate = "", report = "", boot = "";

    int h = 0, m = 0, s = 0;

    ptime start, end;

    date f_d, s_d;

    time_duration timeDiff;

    boost::regex startReg("([0-9]+)-([0-9]+)-([0-9]+) ([0-9]+):([0-9]+)
```

```
:([0-9]+): \\(log.c.166\\) server started.*"); //NOLINT

    boost::regex endReg("([0-9]+)-([0-9]+)-([0-9]+) ([0-9]+):([0-9]+)
:([0-9]+).([0-9]+):INFO:oejs.AbstractConnector:Started
SelectChannelConnector@0.0.0.0:9080.*"); //NOLINT

    boost::smatch smat;

    string str;
    std::ifstream file(input.c_str());

    bool fBoot = false;

    if (file.is_open()) {
        while (getline(file, str)) {
            bDate.clear();
            eDate.clear();

            if (boost::regex_search(str, smat, startReg)) {
                cDate = smat[1] + "-" + smat[2] + "-" + smat[3];
                bDate = cDate + " " + smat[4] + ":" + smat[5] + ":" + smat
[6];

                f_d = date(from_simple_string(cDate));

                h = std::stoi(smat[4]);
                m = std::stoi(smat[5]);
                s = std::stoi(smat[6]);

                start = ptime(f_d, time_duration(h, m, s));

                if (fBoot == true) {
                    boot += "Booting Failed \n";
                }

                boot += "Booting\n" + std::to_string(counter1) + "(" +\
                    input + "):" + bDate + "Starting...\n";
                numBoot++;
                fBoot = true;
                }
            if (boost::regex_match(str, smat, endReg)) {
                cDate = smat[1] + "-" + smat[2] + "-" + smat[3];
                eDate = cDate + " " + smat[4] + ":" + smat[5] + ":" + smat
[6];

                s_d = date(from_simple_string(cDate));

                h = std::stoi(smat[4]);
                m = std::stoi(smat[5]);
                s = std::stoi(smat[6]);

                end = ptime(s_d, time_duration(h, m, s));

                boot += std::to_string(counter1) + "(" +\
                    input + "):" + eDate + "Boot complete\n";

                auto bootEqu = [&] () {return (end-start);};
                timeDiff = bootEqu();

```

```
                boot += "Booting time: ";
                boot += std::to_string(timeDiff.total_milliseconds()) + "ms
    \n";

                success++;
                fBoot = false;
            }
            counter1++;
        }
        file.close();
    }

    report += "Boot Report\n Intouch logfile: " + input + "\n" +\
         "Lines scanned: " + std::to_string(counter1) + "\n\n";

    report += "Device boot count: initiated = " + std::to_string(numBoot) +\
        ", completed: " + std::to_string(success) + "\n\n";

    report += boot;

    std::ofstream out(output.c_str());
    out << report;
    out.close();

    return 0;
}
```