ALEX MAZZARESE – C++ PORTFOLIO
12/8/19

TABLE OF CONTENTS

PS0: “Hello World” With SFML

1) Discussion

This assignment’s purpose was to develop a core understanding of how the SFML library works. We were supposed to implement a simple CircleShape object and display it to the screen. I also had to implement a Texture object and load our own image into it and display this using a Sprite object. Finally, I also implemented code so that the user can move the first CircleShape object. I did this by using the isKeyPressed() function as well as the shape.move() function.

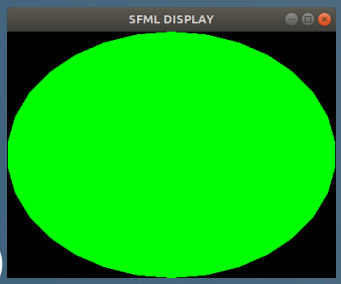
2) Key Algorithms & Data Structures

One of the main algorithms that I used in this assignment was a couple of if statements within while loops to control the opening and closing of the different SFML windows (i.e. if (event.type == sf::Event::Closed) { window.close(); }). The other important algorithm I used in this was the if statements to control the CircleShape object’s movement. I used a series of if statements to control whether the left, right, up or down keys are pressed, and if they are the shape will move (i.e. if (sf::Keyboard::isKeyPressed()) { shape.move() }).

3) What I Learned

This assignment helped me to understand the basic functionality of how a simple SFML window can be used to display different types of SFML objects. Another important concept that this taught me is that different SFML objects can also be manipulated to be moved using the SFML function sf::Keyboard::isKeyPressed() function. I thought this concept was especially interesting because this opens up the door for different ideas such as a video game with moving sprites among other things.

4) Example Output

A) (CircleShape object) 

B) (Sprite object) 

5) Errors/Unfinished Sections

There were no errors or parts of the code that were not finished. This implementation should function fully as intended.

6) Source Code

main.cpp

```
1  //WRITTEN BY: ALEX MAZZARESE
2  //CLASS: COMP2400 – Section 201
3
4  #include <SFML/Graphics.hpp>
5
6  int main()
7  {
8      sf::RenderWindow window(sf::VideoMode(200, 200), "SFML works!");
9      sf::CircleShape shape(100.f);
10     shape.setFillColor(sf::Color::Green);
11
12     sf::RenderWindow window2(sf::VideoMode(200,200), "SFML DISPLAY 2");
13     sf::Texture texture;
14     if(!texture.loadFromFile("sprite.png"))
15     return EXIT_FAILURE;
16     sf::Sprite sprite(texture);
17
18     sf::RenderWindow window3(sf::VideoMode(200,200), "SFML DISPLAY 3");
19     sf::CircleShape shape2(100.f);
20     shape2.setFillColor(sf::Color::Blue);
21
22     sf::Vector2u size = window2.getSize();
23     unsigned int x = size.x;
24     unsigned int y = size.y;
25
26     while (window.isOpen())
27     {
28         sf::Event event;
29         while (window.pollEvent(event))
30         {
31             if (event.type == sf::Event::Closed)
32                 window.close();
33         }
34
35     sf::Event event2;
36     while (window2.pollEvent(event2))
37     {
38         if (event2.type == sf::Event::Closed)
39             window2.close();
40     }
41
42     sf::Event event3;
```

```
    while (window3.pollEvent(event3))
    {
        if (event3.type == sf::Event::Closed)
            window3.close();
    }

        window.clear();
        window.draw(shape);
    //changes the size of the window
    window.setSize(sf::Vector2u(400,300));
    //changes the display title for the window
    window.setTitle("SFML DISPLAY");
        window.display();
    window.setFramerateLimit(60);

    window2.clear();
    window2.draw(sprite);
    //changes the size of the window
    window2.setSize(sf::Vector2u(300,400));
    //changes the display title for the window
    window2.setTitle("DALI");
    window2.display();
    window2.setFramerateLimit(60);
    //moves the sprite
    sprite.move(sf::Vector2f(1,1));

    window3.clear();
    window3.draw(shape2);
    window3.setSize(sf::Vector2u(400,300));
    window3.setTitle("SFML DISPLAY 3");
    window3.display();
    window3.setFramerateLimit(60);

if(sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
    shape.move(sf::Vector2f(-5,0));
else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Right))
    shape.move(sf::Vector2f(5,0));
else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Up))
    shape.move(sf::Vector2f(0,-5));
else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Down))
    shape.move(sf::Vector2f(0,5));


}

    return 0;

}
```

Makefile

```
CC = g++
CFLAGS = -c -g -Og -Wall -Werror -ansi -pedantic
OBJ = main.o
DEPS =
LIBS = -lsfml-graphics -lsfml-window -lsfml-system
EXE = SFML-app

all: $(OBJ)
	$(CC) $(OBJ) -o $(EXE) $(LIBS)

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

clean:
	rm $(OBJ) $(EXE)
```

PS1a: Linear Feedback Shift Register (Part A)

1) Discussion

   For this assignment we were supposed to experiment with the manipulation of a Linear Feedback Shift Register. What this is supposed to do is take a sequence of bits and use an exclusive or on the bits 8 and 10 to determine what the initial seed will be for the next bit sequence. To do this, I implemented a vector to control the series of bits. I also implemented the exclusive or part within the step() function so that the bit sequence can shift properly with each call of step(), this also prints out the corresponding rightmost integer for each bit sequence. Furthermore, I implemented a total variable for the generate() function which will give us a value for each corresponding bit sequence.

2) Key Algorithms & Data Structures

   The main algorithm that was essential to this program was the exclusive or implementation. For this I used two variables, one that represented the 8^{th} bit and one that represented the 10^{th} bit. If the two bits were equal, the resulting rightmost integer is 0, but if they are different then the rightmost integer is equal to 1. This sets up the control for the LFSR. Another important algorithm used is the one that calculates the value for the generate() function. To compute this I simply would use the vector to observe whether or not the value at that specific bit was 1 or 0. Then, based on this, I would create a grand total corresponding to the bit's placement (i.e. if (vector[10] == 1) { total += 1 }) (note: different variables used in source code).

3) What I Learned

   This project was essential in helping me grasp the understanding of how a bitwise shift works. It also helped me understand how the exclusive or can be useful in determining values for different types of algorithms. Finally, this program overall helped me to understand how shift registers work and how they can be used in a useful way.

4) Example Output

```
Testing the constructor and extraction
operator...
Testing using LFSR lfsr('00001011001', 8)...
Current: 00001011001 8
Constructor & Extraction Operator SUCCESS


Testing the step() function...
Calling step() 6 times...
Testing using LFSR lfsrA('100001101101', 8)...
00010110010 0
00101100100 0
01011001001 1
10110010010 0
01100100100 0
11001001001 1
step() function SUCCESS!


Testing the generate() function...
Testing using LFSR lfsrB('00110100100', 8)...
00100111011 27
11101101110 14
10111010110 22
01011010100 20
01010001100 12
00110000101 5
00010111100 28
11110010010 18
01001001110 14
00111011011 27
01101110101 21
generate() function SUCCESS
```

5) Errors/Unfinished Sections

No errors, the program functions as intended. There were also no unfinished sections of the code.

6) Source Code

LFSR.cpp

```cpp
#include <iostream>
#include <string>
#include <vector>
#include "LFSR.hpp"

LFSR::LFSR(std::string seed, int t)
{
    _t = t;
    for(int i = 0; i < 11; i++)
    {
        v1.push_back(seed[i] - 48);
    }
}

int LFSR::step()
{
    int sig;
    int xr1 = v1[0];
    int xr2 = v1[2];
    if(xr1 == xr2)
    {
        sig = 0;
        v1[11] = 0;
    }
    if(xr1 != xr2)
    {
        sig = 1;
        v1[11] = 1;
    }

    for(int i = 0; i < v1.size(); i++)
    {
        v1[i] = v1[i+1];
    }

    for(int i = 0; i < v1.size(); i++)
    {
        std::cout << v1[i];
    }

    std::cout << " " << sig << std::endl;

```

```
43      return 0;
44  }
45
46  int LFSR::generate(int k)
47  {
48  for(int d = 0; d < v1.size(); d++)
49  {
50      for(int a = 0; a < k; a++)
51      {
52          for(int b = 0; b < v1.size(); b++)
53          {
54              v1[b] = v1[b+1];
55              int x3 = v1[0];
56              int x4 = v1[2];
57              if(v1[0] == v1[2])
58              {
59                  v1[11] = 0;
60              }
61              else
62              {
63                  v1[11] = 1;
64              }
65          }
66      }
67
68      for(int c = 0; c < v1.size(); c++)
69      {
70          std::cout << v1[c];
71      }
72
73      int sum = 0;
74      if(v1[10] == 1)
75      {
76          sum += 1;
77      }
78      if(v1[9] == 1)
79      {
80          sum += 2;
81      }
82      if(v1[8] == 1)
83      {
84          sum += 4;
```

```
85      }
86      if(v1[7] == 1)
87      {
88          sum += 8;
89      }
90      if(v1[6] == 1)
91      {
92          sum += 16;
93      }
94
95      std::cout << " " << sum << std::endl;
96  }
97  }
98
99  std::ostream& operator<<(std::ostream &out, const LFSR& lfsr)
100 {
101     out << "Current: ";
102     for(int i = 0; i < lfsr.v1.size(); i++)
103     {
104         out << lfsr.v1[i];
105     }
106     out << " " << lfsr._t << std::endl;
107     return out;
108 }
109
110 int main(int argc, char* argv[])
111 {
112     std::cout << "Testing the constructor and extraction operator..." << std::endl;
113     LFSR lfsr("00001011001", 8);
114     std::cout << "Testing using LFSR lfsr('00001011001', 8)..." << std::endl;
115     std::cout << lfsr;
116     std::cout << "Constructor & Extraction Operator SUCCESS" << std::endl;
117     std::cout << "\n" << std::endl;
118
119     LFSR lfsrA("10000110110", 8);
120     std::cout << "Testing the step() function..." << std::endl;
121     std::cout << "Calling step() 6 times..." << std::endl;
122     std::cout << "Testing using LFSR lfsrA('100001101101', 8)..." << std::endl;
123     lfsr.step();
124     lfsr.step();
125     lfsr.step();
126     lfsr.step();
127     lfsr.step();
128     lfsr.step();
129     std::cout << "step() function SUCCESS!" << std::endl;
130     std::cout << "\n" << std::endl;
131
132     LFSR lfsrB("00110100100", 8);
133     std::cout << "Testing the generate() function..." << std::endl;
134     std::cout << "Testing using LFSR lfsrB('00110100100', 8)..." << std::endl;
135     lfsr.generate(5);
136     std::cout << "generate() function SUCCESS" << std::endl;
137
138     return 0;
139 }
```

LFSR.hpp

```
#include <iostream>
#include <vector>
#include <string>

class LFSR
{
    public:
        LFSR(std::string seed, int t);
        int step();
        int generate(int k);
        void stringConvert();
        friend std::ostream& operator<<(std::ostream&, const LFSR&);
    private:
        std::vector<int> v1;
        int _t;
};
```

Makefile

```
CC = g++
CFLAGS = -std=c++11 -c -g -0g -Wall -Werror -ansi -pedantic
OBJ = LFSR.o
DEPS =
LIBS =
EXE = LFSR-app

all: $(OBJ)
    $(CC) $(OBJ) -o $(EXE) $(LIBS)

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

clean:
    rm $(OBJ) $(EXE)
```

PS1b: Linear Feedback Shift Register (Part B)

1) Discussion

For this assignment, we were supposed to use the Linear Feedback Shift Register that we created in the previous assignment, and create an image that is first displayed as normal and then displayed with inverted colors. To complete this, I added a new file entitled PhotoMagic.cpp whose purpose was to load the pictures from the file and print both a normal version and an inverted version. I successfully implemented the pictures to display both versions of the image, but the only thing I did not implement was that this version of the code can't accept command line arguments or save the encrypted files.

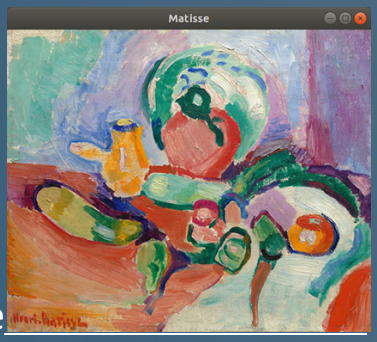
2) Key Algorithms/Data Structures

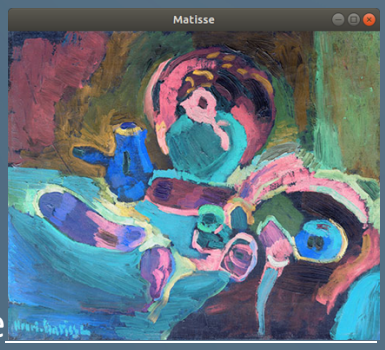
In this part of the assignment, we had already completed the Linear Feedback Shift Register, so now all we had to do was implement a way for the pictures to display in both a normal and inverted method. To do this, I simply adjusted colors within a Color variable and then applied these adjusted inverted colors to the Image variable.

3) What I Learned

From this assignment, I learned that the Feedback Shift Register can be used for much more than just bitwise operations. Furthermore, I found that manipulation of images in SFML can be done through a variety of different methods, in this instance we used the LFSR. This part of the assignment overall taught me how to further manipulate the different SFML variables and functions.

4) Sample Output

A) Normal Image 

B) Inverted Image 

5) Errors/Unfinished Sections

Despite the fact that the program compiles and functions, there were some parts that did not meet the specs in the project description. This code doesn't accept the command line arguments and it also doesn't save the encrypted image.

6) Source Code

PhotoMagic.cpp

```
#include <iostream>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "LFSR.hpp"

int main(int argc, char* argv[])
{
    sf::Image i1;
    if(!i1.loadFromFile("pic1.jpg"))
        return EXIT_FAILURE;
    sf::Image i2;
    if(!i2.loadFromFile("pic2.jpg"))
        return EXIT_FAILURE;
    sf::Image i3;
    if(!i3.loadFromFile("pic1copy.jpg"))
        return EXIT_FAILURE;
    sf::Image i4;
    if(!i4.loadFromFile("pic2copy.jpg"))
        return EXIT_FAILURE;

    sf::Color p1;
        for(int a = 0; a < 280; a++)
        {
                for(int b = 0; b < 218; b++)
                {
                        p1 = i1.getPixel(a,b);
                        p1.r = 255 - p1.r;
                        p1.g = 255 - p1.g;
                        p1.b = 255 - p1.b;
                        i1.setPixel(a,b,p1);
                }
        }

        sf::Color p2;
        for(int c = 0; c < 483; c++)
        {
                for(int d = 0; d < 400; d++)
                {
                        p2 = i2.getPixel(c,d);
                        p2.r = 255 - p2.r;
                        p2.g = 255 - p2.g;
```

```
                        p2.b = 255 - p2.b;
                        i2.setPixel(c,d,p2);
                }
        }

    sf::Vector2u v1 = i1.getSize();
        sf::RenderWindow w1(sf::VideoMode(v1.x, v1.y), "Monet");
        sf::Vector2u v2 = i2.getSize();
        sf::RenderWindow w2(sf::VideoMode(v2.x, v2.y), "Matisse");
    sf::Vector2u v3 = i3.getSize();
        sf::RenderWindow w3(sf::VideoMode(v3.x, v3.y), "Monet");
        sf::Vector2u v4 = i4.getSize();
        sf::RenderWindow w4(sf::VideoMode(v4.x, v4.y), "Matisse");

    sf::Texture t1;
        t1.loadFromImage(i1);
        sf::Texture t2;
        t2.loadFromImage(i2);
        sf::Texture t3;
        t3.loadFromImage(i3);
        sf::Texture t4;
        t4.loadFromImage(i4);

    sf::Sprite s1(t1);
        sf::Sprite s2(t2);
        sf::Sprite s3(t3);
        sf::Sprite s4(t4);

    while(w1.isOpen() && w2.isOpen() && w3.isOpen() && w4.isOpen())
    {
         sf::Event event;
            while (w1.pollEvent(event))
            {
            if (event.type == sf::Event::Closed)
                    w1.close();
            }

           sf::Event event2;
           while (w2.pollEvent(event2))
           {
               if (event2.type == sf::Event::Closed)
                        w2.close();
```

```
85                  }
86
87                  sf::Event event3;
88                  while(w3.pollEvent(event3))
89                  {
90                          if(event3.type = sf::Event::Closed)
91                                  w3.close();
92                  }
93
94                  sf::Event event4;
95                  while(w4.pollEvent(event4))
96                  {
97                          if(event4.type = sf::Event::Closed)
98                                  w4.close();
99                  }
100
101         w1.clear(sf::Color::White);
102         w1.draw(s1);
103         w1.display();
104         w1.setFramerateLimit(60);
105         w2.clear(sf::Color::White);
106         w2.draw(s2);
107         w2.display();
108         w2.setFramerateLimit(60);
109                 w3.clear(sf::Color::White);
110                 w3.draw(s3);
111                 w3.display();
112                 w3.setFramerateLimit(60);
113                 w4.clear(sf::Color::White);
114                 w4.draw(s4);
115                 w4.display();
116                 w4.setFramerateLimit(60);
117
118     }
119
120     return 0;
121 }
```

ALL OTHER FILES IN PS1b (LFSR.cpp, LFSR.hpp, Makefile) ARE THE SAME AS SHOWN IN PS1a]

PS2: Recursive Graphics (Pythagoras Tree)

1) Discussion

For this project, we were supposed to design a Pythagoras Tree using the SFML library to aid us in displaying this graphically. For implementation, I followed the standard output method for the main() function, just simply allowed for a screen to be displayed and close it afterwards. As for the pTree() function, I had to program a way for the system to display a Convex Shape, and iterate it through the tree function. To do this, I used help from the SFML tutorials page to help get me started using the ConvexShape.hpp folder. After I knew how to make the shape itself, I simply set a pattern that changes for each recursive call of the pTree() function. To do this I created a ratio by witch it expands, as well as changing x and y vertices so that the shape can form a full pattern of rectangles.

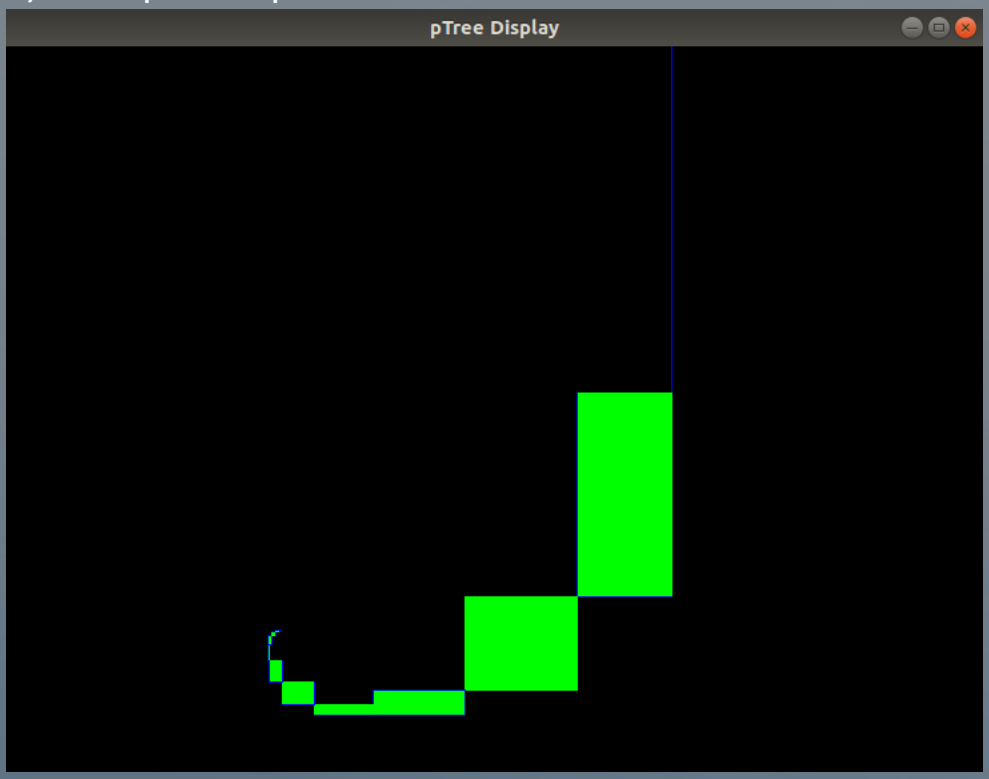
2) Key Algorithms/Data Structures

One of the most important algorithms that I used in this project was the recursive use of the pTree() function. We had to use this because in order to get a repeating pattern printed you need a decay value that changes with each iteration. To do this I picked a rate by which to decrease, as well as a depth factor. Then, to get a decreasingly smaller pattern printed, I simply used this number in the recursive call for pTree() so that with each iteration the size of the printed object would grow smaller.

3) What I Learned

This was definitely one of the more in-depth projects with using the SFML libraries. But It helped me to understand the full capabilities of what can be produced using the SFML display and how you can adjust this using recursive methods or other functions. It also taught me what the SFML displays will and will not accept, when troubleshooting for this assignment I realized there were some methods that would result in a blank SFML screen, so now I know how to code around these errors for similar future SFML based assignments.

4) Example Output



5) Errors/Unfinished Sections

For this assignment, I got the program to function, but It doesn't accept command line arguments.

6) Source Code

pTree.cpp

```
1  #include <SFML/Graphics.hpp>
2  #include <SFML/System.hpp>
3  #include <SFML/Window.hpp>
4  #include <iostream>
5  #include <cmath>
6  #include <ctime>
7  #include <math.h>
8  #include "pTree.hpp"
9
10 float baseSQR = 90;
11 float x = 500;
12 float y = 0;
13 int depth = 400;
14
15 sf::RenderWindow w(sf::VideoMode(600,600), "pTree Display");
16
```

```
17 void pTree(float x, float y, int baseSQR, int depth)
18 {
19         if(depth < 1)
20         {
21             return;
22         }
23         else
24         {
25          float pi = 3.1415926;
26          float rate = 0.65;
27          float base1 = cos(baseSQR * pi / 180);
28          float base2 = sin(baseSQR * pi / 180);
29          float x1 = x + depth * rate * base1;
30                  float y1 = y + depth * rate * base2;
31                  sf::ConvexShape shape;
32          shape.setPointCount(4);
33          shape.setPoint(0, sf::Vector2f(x,y));
34          shape.setPoint(1, sf::Vector2f(x,y1));
35          shape.setPoint(2, sf::Vector2f(x1,y1));
36          shape.setPoint(3, sf::Vector2f(x1,y));
37          shape.setOutlineColor(sf::Color::Blue);
38          shape.setOutlineThickness(0.5);
39          shape.setFillColor(sf::Color::Green);
40                  w.draw(shape);
41          float baseTmp1 = baseSQR + 25;
42                  pTree(x1, y1, baseTmp1, depth*rate);
43     }
44 }
45
```

```
46 int main(int argc, char **argv)
47 {
48     while(w.isOpen())
49     {
50         sf::Event event;
51         while(w.pollEvent(event))
52         {
53             if(event.type == sf::Event::Closed)
54             {
55                 w.close();
56             }
57         }
58         w.clear();
59         pTree(x, y, baseSQR, depth);
60         w.display();
61         w.setFramerateLimit(60);
62     }
63     return 0;
64 }
```

pTree.hpp

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

class PTree : public sf::Drawable, sf::Transformable
{
    public:
    PTree();
    ~PTree();
    void pTree(float x, float y, float baseSQR, int depth);
};
```

Makefile

```
CC = g++
CFLAGS = -c -g -Og -Wall -Werror -ansi -pedantic
OBJ = pTree.o
DEPS =
LIBS = -lsfml-graphics -lsfml-window -lsfml-system
EXE = pTree

all: $(OBJ)
    $(CC) $(OBJ) -o $(EXE) $(LIBS)

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

clean:
    rm $(OBJ) $(EXE)
```

PS3a: N-Body Simulation (Part A)

1) Discussion

This assignment was intended to teach us how to implement multiple sprite objects onto the same display screen. Specifically, we were supposed to create a virtual representation of the solar system (the sun and the first four planets). To make this program function correctly, we created a class that contains variables that correspond to each of the different values listed in the "planets.txt" file. Once we had this, we simply read in the data from the txt file and applied each number to a different pre-made variable.

2) Key Algorithms/Data Structures

The key algorithms involved in making this app work are as follows: we had to create a loop that reads in the data about the planets from the file. Once this was working, we simply had to pass the data through our class and apply each trait to its corresponding variable in memory.

3) What I Learned

Through this assignment, I learned how to create multiple objects for display use within the SFML display window. It also taught me how to take these display objects and apply parameters within a class to them. Furthermore, I now understand better how to implement a functioning display that could support any number of sprites on the SFML display window.

4) Sample Output

5) Errors/Unfinished Sections

The code compiles fully and functions as intended. However, the source code does not match the specifications fully because there are no smart pointers in this implementation.

6) Source Code

Body.cpp

```
#include <string>
#include <iostream>
using namespace std;
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <math.h>
#include <vector>
#include <fstream>

const int window_y = 800;
const int window_x = 800;

class body : public sf:: Drawable
{
public:
    body()
    {
     return;
    }

    body(double xin, double yin, double xvel, double yvel, double mas,double radius, string img)
    {
     x=xin;
     y=yin;
     mass=mas;
     velx=xvel;
     vely=yvel;
     mass=mas;
     filename=img;
     if(!image.loadFromFile(filename))
     {
         return;
     }
```

```
      texture.loadFromImage(image);
      obj.setTexture(texture);
      obj.setPosition(sf::Vector2f(x,y));
     }
     void set_radius(double rad)
     {
      radius=rad;
     }
     void new_position()
     {
      double px=((x/radius)*(window_x/2))+(window_x/2);
      double py=((y/radius)*(window_y/2))+(window_y/2);
      obj.setPosition(sf::Vector2f(px,py));
     }
     friend istream& operator>>(istream &input, body &nbody)
     {
      input >> nbody.x>> nbody.y;
      input >> nbody.velx >> nbody.vely;
      input >> nbody.mass >>nbody.filename;
      if(!nbody.image.loadFromFile(nbody.filename))
      {
      return input;
      }
      nbody.texture.loadFromImage(nbody.image);
      nbody.obj.setTexture(nbody.texture);
      nbody.obj.setPosition(sf::Vector2f(nbody.x,nbody.y));


      return input;
     }


private:
     virtual void draw(sf::RenderTarget& window, sf::RenderStates)const
     {
      window.draw(obj);
     }
     double x;
     double y;
     double mass;
     double velx;
     double vely;

     string filename;
     double radius;
     sf::Image image;
     sf::Texture texture;
     sf::Sprite obj;
};
```

main.cpp

```
#include "Body.cpp"

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

    string n_planets, rad;
    string::size_type size;

    cin>>n_planets;
    cin>>rad;
    int nplanets = stoi(n_planets,&size);
    float radius=stod(rad, &size);

    vector<body>planets;
    vector<body>::iterator it;
    for(int i=0; i < nplanets;i++)
    {
     body* planet=new body();
     cin>>*planet;
     planet->set_radius(radius);
     planet->new_position();
     planets.push_back(*planet);
    }

    sf::RenderWindow window(sf::VideoMode(window_x, window_y),
    "The Solarrrr");

    while (window.isOpen())
    {
     sf::Event event;
     while(window.pollEvent(event))
     {
         if (event.type == sf::Event::Closed)
         {
             window.close();
         }
     }
     window.clear();
     for(it = planets.begin(); it != planets.end(); it++)
     {
         window.draw(*it);
     }

     window.display();

    }
    return 0;
}
```

Makefile

```
CC= g++
CFLAGS= -Wall -Werror -std=c++0x -pedantic
SFMLFLAGS= -lsfml-graphics -lsfml-window -lsfml-system
OBJ = Body.o
EXE = Nbody
all: NBody

NBody: main.o Body.o
    $(CC) main.o Body.o -o NBody $(SFMLFLAGS)

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

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

clean:
    rm *.o
    rm NBody
```

PS3b: N-Body Simulation (Part B)

1) Discussion

For this part of the ps3 assignment we had to implement force elements (x,y), acceleration elements (x,y), and a step() function to our already assembled planetary representation of the universe. What these additions are supposed to do is simulate real life movement of the different planetary bodies using force and acceleration calculations and display these using SFML. For this code project I got the project to successfully move the planetary bodies with the correct force, acceleration and velocity implementations. However, I was unable to get the smart pointers to function correctly. I attempted to create a vector of smart pointers to Body objects, but it didn't work with my previous code from assignment ps3a.

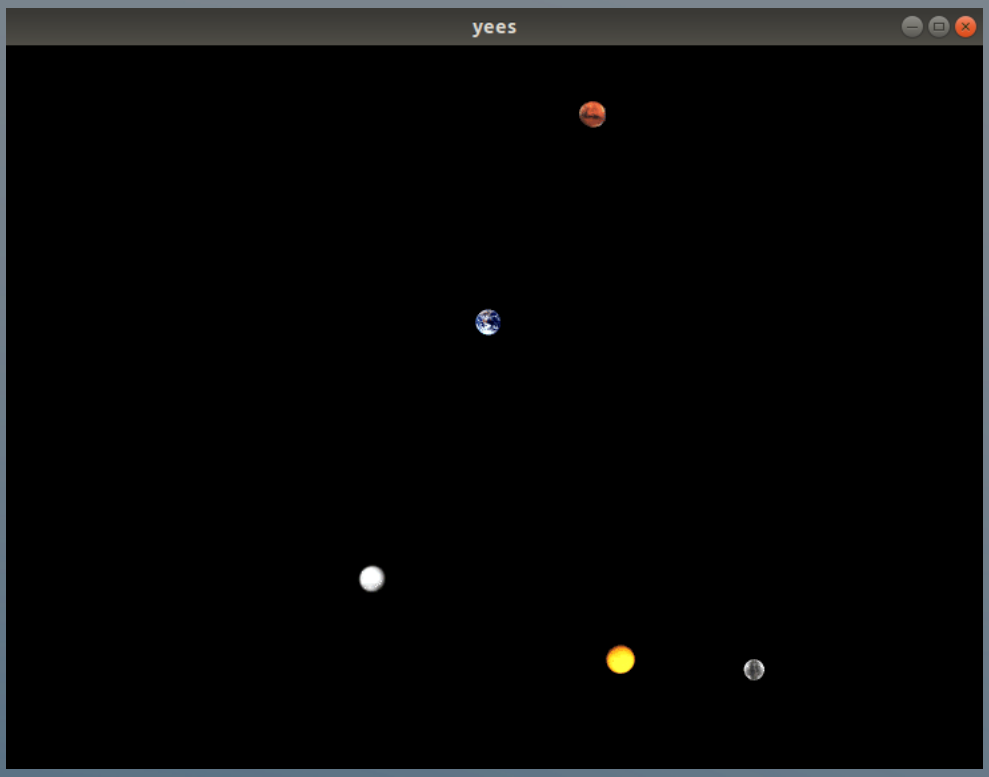
2) Key Algorithms/Data Structures

One essential part of the algorithms for this assignment include the force and acceleration calculations. The acceleration calculations are based on force(x or y direction) / mass. The force calculations are based on (Gravity)*((mass1*mass2)/radius). Another essential design element to this code is the iteration through the number of planets using a for loop(). The code implements a loop that ends after (# of planets) is reached, and this loop is used to create and move the planets in the main() function.

3) What I Learned

For this part of the assignment, we had to focus more on calculations as opposed to display problems. From this, I learned how to take an algorithm and apply it to an SFML display. The algorithm we used for this assignment was the calculation for pairwise force. In this implementation, we calculated the different force values in both the x and y directions, after this we applied a final step() function to get the next position for the planetary bodies. Overall, this assignment taught me how to take real life physics and calculations and apply them to a valid SFML display.

4) Sample Output



5) Errors/Unfinished Code

The program compiles and functions as intended. However, it is different from the specifications in the sense that it doesn't print the state of the universe at the end of the simulation, and it also does not use smart pointers.

6) Source Code

Body.cpp (update)

```
#include <string>
#include <iostream>
using namespace std;
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <math.h>
#include <vector>
#include <fstream>

const int window_y =900;
const int window_x = 900;
const double gravity= 6.67*pow(10,-11);
```

```
14 class body : public sf:: Drawable
15 {
16 public:
17     body()
18     {
19      return;
20     }
21
22     body(double xin, double yin, double xvel, double yvel, double mas, string img)
23     {
24      x=xin;
25      y=yin;
26      velx=xvel;
27      vely=yvel;
28      mass=mas;
29      filename=img;
30
31      if(!image.loadFromFile(filename))
32      {
33          return;
34      }
35      texture.loadFromImage(image);
36      obj.setTexture(texture);
37      obj.setPosition(sf::Vector2f(x,y));
38     }
39
40     void set_radius(double rad)
41     {
42      radius=rad;
43     }
44
45     void new_position()
46     {
47      double py=((x/radius)*(window_x/2))+(window_x/2);
48      double px=((y/radius)*(window_y/2))+(window_y/2);
49      obj.setPosition(sf::Vector2f(px,py));
50     }
51
52     friend istream& operator>>(istream &input, body &nbody)
53     {
54
55      input >> nbody.x;
56      input >> nbody.y;
57      input >> nbody.velx;
58      input >> nbody.vely;
59      input >> nbody.mass;
60      input >> nbody.filename;
61      if(!nbody.image.loadFromFile(nbody.filename))
62      {
63          return input;
64      }
65
```

```
66        nbody.texture.loadFromImage(nbody.image);
67        nbody.obj.setTexture(nbody.texture);
68        nbody.obj.setPosition(sf::Vector2f(nbody.x,nbody.y));
69
70        nbody.forx=0;
71        nbody.fory=0;
72        nbody.accx=0;
73        nbody.accy=0;
74
75        return input;
76
77      }
78
79      friend double forcex(body &body1, body &body2)
80      {
81        double dx = body2.x-body1.x;
82        double dy = body2.y-body1.y;
83        double r1=pow(dx,2)+pow(dy,2);
84        double r=sqrt(r1);
85        double force = (gravity)*((body1.mass*body2.mass)/r1);
86        return force*(dx/r);
87      }
88
89      friend double forcey(body &body1, body &body2)
90      {
91        double dx = body2.x-body1.x;
92        double dy = body2.y-body1.y;
93        double r1=pow(dx,2)+pow(dy,2);
94        double r=sqrt(r1);
95        double force = (gravity)*((body1.mass*body2.mass)/r1);
96        return force*(dy/r);
97      }
98
99      void set_force(double fx, double fy)
100     {
101       forx=fx;
102       fory=fy;
103     }
104
105     void step(double t)
106     {
107       accx=forx/mass;
108       accy=fory/mass;
109       velx=(accx*t)+velx;
110       vely=(accy*t)+vely;
111       x=(velx*t)+x;
112       y=(vely*t)+y;
113       new_position();
114     }
115
```

```
116 private:
117     virtual void draw(sf::RenderTarget& window, sf::RenderStates)const
118     {
119      window.draw(obj);
120     }
121     double x;
122     double y;
123     double mass;
124     double velx;
125     double vely;
126     double radius;
127     double fory;
128     double accy;
129     double forx;
130     double accx;
131     string filename;
132     sf::Image image;
133     sf::Texture texture;
134     sf::Sprite obj;
135 };
```

main.cpp (update)

```
1  #include "Body.cpp"
2  int main(int argc, char* argv[])
3  {
4      string n_planets, rad;
5      string secs(argv[1]);
6      string stime(argv[2]);
7      string::size_type sz;
8
9      double simulationtime = 0;
10     double sim= stod(secs, &sz);
11     double time= stod(stime, &sz);
12
13     cin>>n_planets;
14     cin>>rad;
15
16     int nplanets = stoi(n_planets,&sz);
17     float radius=stod(rad, &sz);
18     vector<body>planets;
19
20     for(int i=0; i < nplanets;i++)
21     {
22      body* planet=new body();
23      cin>>*planet;
24      planet->set_radius(radius);
25      planet->new_position();
26      planets.push_back(*planet);
27     }
28
```

```
    sf::RenderWindow window(sf::VideoMode(window_x, window_y),
    "yees");
    window.setFramerateLimit(60);
    vector<body>::iterator it;
    vector<body>::iterator x,y;

    while (window.isOpen())
    {
     sf::Event event;
     while(window.pollEvent(event))
     {
         if (event.type == sf::Event::Closed)
         {
             window.close();
         }
     }
     window.clear();
     x=planets.begin();
     double fx, fy;
     for(int i = 0; i<nplanets; i++)
     {
         y=planets.begin();
         fx=0;
         fy=0;
     for(int j=0; j<nplanets; j++)
     {
         if(i!=j)
         {
             fx+=forcex(*x,*y);
             fy+=forcey(*x,*y);
         }

         y++;
     }
     x->set_force(fx,fy);
     x++;
     }
     for(it = planets.begin(); it != planets.end(); it++)
     {
         window.draw(*it);
         it->step(time);
         it->new_position();
     }
     window.display();
     simulationtime+=time;
     if(simulationtime==sim)
     {
         break;
     }
    }
    return 0;
}
```

MAKEFILE IS THE SAME AS IN PS3a)

PS4: DNA Sequence Alignment

1) Discussion

For this assignment we were required to find the alignment of two strands of DNA. The main purpose was to find the "edit distance" between the two DNA strands. To do this, we had to line up the two strands using strings and either insert a gap, align two characters that do not match or align two characters that do match. For each of those characteristics we assigned the values 2, 1 and 0 respectively. Finally, once you complete all of this, you will be left with a total value generated by the values described earlier, this is your edit distance.

2) Key Algorithms/Data Structures

The most important algorithm in this particular assignment is the equation used to find the DNA alignment. To find the alignment, we followed the Needleman-Wunsch algorithm. This method incorporates a N x M matrix to control the DNA sequence comparison itself. Through this matrix the alignment comes from the shortest edit distance required to traverse the path used to form the matrix. It does this by comparing the values at [i+1][j+1], [i+1][j], [i][j+1].

3) What I Learned

From this assignment, I think the most important thing I learned is that you can manipulate data (such as a DNA strand) using different algorithms to control how the system processes that data. The other important thing that this project taught me was how to work with a N x M matrix, which has a different control flow than your standalone matrix.

4) Sample Output

```
Edit distance = 7
A T 1
A A 0
C - 2
A A 0
G G 0
T G 1
T T 0
A - 2
C C 0
C A 1

Execution time is 0.001406 seconds
```

5) Errors/Unfinished Code

Everything in the code functions as intended, the only thing missing from the original spec is that the program does not show the largest N time calculation.

6) Source Code

ED.cpp

```
1  #include "ED.hpp"
2
3  ED::ED()
4  {
5
6  }
7  ED::ED(std::string string_one, std::string string_two)
8  {
9    _string_one = string_one;
10   _string_two = string_two;
11 }
12
13 ED::~ED()
14 {
15
16 }
17
18 int ED::penalty(char a, char b)
19 {
20   if(a == b)
21   {
22     return 0;
23   }
24
25   else if(a != b)
26   {
27     return 1;
28   }
29
30   return -1;
31 }
32
33 int ED::min(int a, int b, int c)
34 {
35   if(a < b && a < c)
36   {
37     return a;
38   }
39
```

```
  else if(b < a && b < c)
  {
    return b;
  }

  else if(c < a && c < b)
  {
    return c;
  }

  return a;
}
int ED::OptDistance()
{
  int i, j;
  int N = _string_one.length();
  int M = _string_two.length();

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

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

  for(i = 0; i <= M; i++)
  {

    _matrix[i][N] = 2 * (M - i);
  }

  for(j = 0; j <= N; j++)
  {

    _matrix[M][j] = 2 * (N - j);
  }

  for(i = M - 1; i >= 0; i--)
  {
    for(j = N - 1; j >= 0; j--)
    {

      int opt1 = _matrix[i+1][j+1] + penalty(_string_one[j], _string_two[i]);
      int opt2 = _matrix[i+1][j] + 2;
      int opt3 = _matrix[i][j+1] + 2;

      _matrix[i][j] = min(opt1, opt2, opt3);
    }
  }

  return _matrix[0][0];
}

```

```
97  std::string ED::Alignment()
98  {
99    std::ostringstream return_string;
100
101   int M = _string_two.length();
102   int N = _string_one.length();
103   int i = 0, j = 0;
104   int pen, opt1, opt2, opt3;
105   std::string ret_str;
106
107   while(i < M || j < N)
108   {
109
110     try{
111       pen =  penalty(_string_one[j], _string_two[i]);
112       opt1 = _matrix.at(i+1).at(j+1) + pen;
113     }
114     catch(const std::out_of_range& error)
115     {
116       opt1 = -1;
117     }
118     try{
119       opt2 = _matrix.at(i+1).at(j) + 2;
120     }catch(const std::out_of_range& error)
121     {
122       opt2 = -1;
123     }
124     try{
125       opt3 = _matrix.at(i).at(j+1) + 2;
126     }catch(const std::out_of_range& error)
127     {
128       opt3 = -1;
129     }
130     if(_matrix[i][j] == opt1)
131     {
132       return_string << _string_one[j] << " " <<  _string_two[i] << " "  << pen << "\n";
133       i++;
134       j++;
135     }
136     else if(_matrix[i][j] == opt2)
137     {
138       return_string << "- " << _string_two[i] << " 2\n";
139       i++;
140     }
141     else if(_matrix[i][j] == opt3)
142     {
143       return_string << _string_one[j] << " -" << " 2\n";
144       j++;
145     }
146   }
147   ret_str = return_string.str();
148   return ret_str;
149 }
```

main.cpp

```cpp
#include "ED.hpp"

int main(int argc, const char* argv[])
{
  sf::Clock clock;
  sf::Time t;
  std::string string1, string2;
  std::cin >> string1 >> string2;
  ED ed_test(string1, string2);
  int distance = ed_test.OptDistance();
  std::string alignment = ed_test.Alignment();
  std::cout << "Edit distance = " << distance << "\n";
  std::cout << alignment;
  t = clock.getElapsedTime();
  std::cout << "\nExecution time is " << t.asSeconds() << " seconds \n";

  return 0;
}
```

ED.hpp

```cpp
#ifndef ED_HPP
#define ED_HPP
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdexcept>
#include <vector>
#include <SFML/System.hpp>

class ED
{
  public:
    ED();
    ED(std::string string_one, std::string string_two);
    ~ED();
    int penalty(char a, char b);
    int min(int a, int b, int c);
    int OptDistance();
    std::string Alignment();
    void PrintMatrix();
  private:
    std::string _string_one, _string_two;
    std::vector< std::vector<int> > _matrix;
};

#endif
```

Makefile

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

all:    ED
ED: main.o ED.o
    $(CC) main.o ED.o -o ED $(SFLAGS)
main.o: main.cpp ED.hpp
    $(CC) -c main.cpp ED.hpp $(CFLAGS)

ED.o:   ED.cpp ED.hpp
    $(CC) -c ED.cpp ED.hpp $(CFLAGS)
clean:
    rm *.o
    rm ED
```

PS5a: Guitar Hero-Ring Buffer Implementation With Unit Tests

---

1) Discussion

For this assignment we were simply supposed to implement a queue that is meant to handle sound frequencies in a later assignment. I accomplished a successful queue that can handle adding and removing data values from a buffer. This implementation also features multiple "safety nets" as to avoid any sort of errors while using the program. Another feature that I implemented involved simple integer addition and subtraction. For example, to adjust the size of the queue, I implemented commands such as: sizeVal--, or sizeVal++.

2) Key Algorithms/Data Structures

The key algorithm for this assignment was based around the way the queue is organized. I used a simple algorithm where upon addition of a new element you increment the size of the queue and add the value into the correct location. Similarly, if you want to delete an item, it will delete an item and adjust the size of the queue accordingly.

3) What I Learned

In this part of the Guitar Hero assignment, I learned how to establish a Ring Buffer that can be used for a variety of things. A ring buffer simply is like a queue where you have a list of values and you use two key values within the sequence to calculate the rightmost value for the next iteration. This is useful for creating a decreasing or increasing array of values, for this assignment in particular we created one that can take decreasing values to mimic the Karplus-Strong algorithm.

4) Sample Output

test.cpp

```
Running 2 test cases…

*** No errors detected
```

ps5a.cpp

```
TEST #1:
============================
enqueued: 5, 7, 2, 3, capacity: 10
Dequeue: 5
Peek: 7
Capacity: 10
Size: 3
Front: 7
Back: 3

TEST #2:
============================
enqueued: 5, 4, 3, 2, 1, capacity: 5
Dequeue: 5
Dequeue: 4
Dequeue: 3
Peek: 2
Capacity: 5
Size: 2
Front: 2
Back: 1
```

5) Errors/Unfinished Code

For this part of the assignment everything compiled and ran as intended.

6) Source Code

RingBuffer.cpp

```cpp
/* Copyright [2019] <Alex Mazzarese> */
#include "RingBuffer.hpp"
RingBuffer::RingBuffer() {}
RingBuffer::~RingBuffer() {}
RingBuffer::RingBuffer(int capacity) {
if(capacity <= 0) {throw std::invalid_argument("ERROR!");}
queue.resize(capacity);
capacityVal = capacity;
sizeVal = 0;
frontVal = 0;
backVal = 0;
return;}
int RingBuffer::size() {
return sizeVal;}
bool RingBuffer::isEmpty() {
if(sizeVal == 0) return true;
else
return false;}
bool RingBuffer::isFull() {
if(sizeVal == capacityVal) return true;
else
return false;}
void RingBuffer::enqueue(int16_t x) {
if(isFull()) {throw std::runtime_error("ERROR!");}
if(backVal >= capacityVal) backVal = 0;
queue.at(backVal) = x;
backVal++;
sizeVal++;}
int16_t RingBuffer::dequeue() {
if(isEmpty()) {throw std::runtime_error("ERROR!");}
queue.at(frontVal) = 0;
frontVal++;
sizeVal--;
if(frontVal >= capacityVal) frontVal = 0;
return frontVal;}
int16_t RingBuffer::peek() {
if(isEmpty()) {throw std::runtime_error("ERROR!");}
return queue.at(frontVal);}
```

RingBuffer.hpp

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

class RingBuffer{
public:
    RingBuffer();
    ~RingBuffer();
    RingBuffer(int capacity);
    int size();
    bool isEmpty();
    bool isFull();
    void enqueue(int16_t x);
    int16_t dequeue();
    int16_t peek();
    std::vector<int16_t> queue;
    int capacityVal;
    int sizeVal;
    int frontVal;
    int backVal;
};
```

main.cpp

```
/* Copyright [2019] <Alex Mazzarese> */
#include "RingBuffer.hpp"
int main() {
RingBuffer buffer1(10);
buffer1.enqueue(5);
buffer1.enqueue(7);
buffer1.enqueue(2);
buffer1.enqueue(3);
std::cout << "TEST #1:" << std::endl;
std::cout << "==================================" << std::endl;
std::cout << "enqueued: 5, 7, 2, 3, capacity: 10" << std::endl;
std::cout << "Dequeue: " << buffer1.queue.at(0) << std::endl;
buffer1.dequeue();
std::cout << "Peek: " << buffer1.peek() << std::endl;
std::cout << "Capacity: " << buffer1.capacityVal << std::endl;
std::cout << "Size: " << buffer1.sizeVal << std::endl;
std::cout << "Front: " << buffer1.queue.at(1) << std::endl;
std::cout << "Back: " << buffer1.queue.at(3) << std::endl;
```

```
RingBuffer buffer2(5);
buffer2.enqueue(5);
buffer2.enqueue(4);
buffer2.enqueue(3);
buffer2.enqueue(2);
buffer2.enqueue(1);
std::cout << "\nTEST #2:" << std::endl;
std::cout << "================================" << std::endl;
std::cout << "enqueued: 5, 4, 3, 2, 1, capacity: 5" << std::endl;
std::cout << "Dequeue: " << buffer2.queue.at(0) << std::endl;
buffer2.dequeue();
std::cout << "Dequeue: " << buffer2.queue.at(1) << std::endl;
buffer2.dequeue();
std::cout << "Dequeue: " << buffer2.queue.at(2) << std::endl;
buffer2.dequeue();
std::cout << "Peek: " << buffer2.peek() << std::endl;
std::cout << "Capacity: " << buffer2.capacityVal << std::endl;
std::cout << "Size: " << buffer2.sizeVal << std::endl;
std::cout << "Front: " << buffer2.queue.at(3) << std::endl;
std::cout << "Back: " << buffer2.queue.at(4) << std::endl;
return 0;}
```

test.cpp

```
/* Copyright [2019] <Alex Mazzarese> */
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include <stdint.h>
#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include "RingBuffer.hpp"
BOOST_AUTO_TEST_CASE(Constructor) {
// Functional Constructor
BOOST_REQUIRE_NO_THROW(RingBuffer(10));
// Failed Constructor
BOOST_REQUIRE_THROW(RingBuffer(0), std::exception);
BOOST_REQUIRE_THROW(RingBuffer(0), std::invalid_argument);
}
BOOST_AUTO_TEST_CASE(enqueue_dequeue) {
RingBuffer queue1(100);
queue1.enqueue(2);
BOOST_REQUIRE_NO_THROW(queue1.dequeue());
RingBuffer queue2(1);
queue2.enqueue(1);
queue2.dequeue();
BOOST_REQUIRE_THROW(queue2.dequeue(), std::runtime_error);}
```

Makefile

```
CC = g++
CFLAGS = -g -Wall -Werror -std=c++0x -pedantic
BOOST = -lboost_unit_test_framework

all: test ps5a

test: test.o RingBuffer.o
    $(CC) test.o RingBuffer.o -o test $(BOOST)

ps5a: main.o RingBuffer.o
    $(CC) main.o RingBuffer.o -o ps5a

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

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

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

clean:
    rm *.o
    rm ps5a
    rm test
```

PS5b: Guitar Hero-GuitarString Implementation And SFML Audio Output

---

1) Discussion

For this part of the assignment we were supposed to use our previously designed RingBuffer class to implement another class entitled GuitarString. This GuitarString class is meant to control the sample and the way that the frequency is manipulated for later use. After designing this, I went on to implement the GuitarHero.cpp file which acts as a main() of sorts, this supplies the actual SFML sound variables along with other data that corresponds to the way the notes should be played.

2) Key Algorithms/Data Structures

One of the main algorithms that is necessary for this assignment is the calculation for the tuning of the note. This comes from the frequency 440hz, and I implemented control of this within my GuitarHero.cpp file. Another key data structure that helped the functionality of this program was the RingBuffer being passed through the GuitarString.cpp file, which allowed us to use the algorithms from the ps5a assignment as well as the new ones.

3) What I Learned

Through this assignment, I learned how SFML can be used to create sounds and samples. Furthermore, we learned how things like a Ring Buffer can be used to manipulate these sounds and create an entire keyboard of samples that can be played similar to how an actual instrument plays.

4) Sample Output

For this part of the assignment in particular there was nothing displayed on the SFML display. This is because when the code is run it creates an instrument out of the keyboard, and using the keys “q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' “ the user can type and the SFML window will play a corresponding note. However, there is no actual output onto the SFML display window.

5) Errors/Unfinished Code

The code was fully functional and compiled, however the notes being played are much higher than they're supposed to be.

6) Source Code

GuitarString.cpp

```
#include "GuitarString.hpp"
#include <vector>

GuitarString::GuitarString(double frequency):
    rB(ceil(44100/frequency))
{
    val = (ceil(44100/frequency));
    for(int i = 0; i < val; i++)
    {
        rB.enqueue((int16_t)0);
    }
    ticCount = 0;
}

GuitarString::GuitarString(std::vector<sf::Int16> init):
    rB(init.size())
{
    val = init.size();
    std::vector<sf::Int16>::iterator i;
    for(i = init.begin(); i < init.end(); i++)
    {
        rB.enqueue((int16_t)*i);
    }
    ticCount = 0;
}

void GuitarString::pluck()
{
    for(int i = 0; i < val; i++)
    {
        rB.dequeue();
    }
    for(int j = 0; j < val; j++)
    {
        rB.enqueue((sf::Int16)(rand() & 0xffff));
    }
    return;
}

void GuitarString::tic()
{
    int16_t val1 = rB.dequeue();
```

```
    int16_t val2 = rB.peek();
    int16_t val3 = (val1 + val2)/2;
    int16_t val4 = val3 * 0.996;
    rB.enqueue((sf::Int16)val4);
    ticCount++;
    return;
}

sf::Int16 GuitarString::sample()
{
    sf::Int16 val1 = (sf::Int16)rB.peek();
    return val1;
}

int GuitarString::time()
{
    return ticCount;
}
```

GuitarString.hpp

```
#ifndef GUITARSTRING_HPP
#define GUITARSTRING_HPP

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

class GuitarString{
public:
    GuitarString(double frequency);
    GuitarString(std::vector<sf::Int16> init);
    void pluck();
    void tic();
    sf::Int16 sample();
    int time();
    RingBuffer rB;
    double frequency;
    std::vector<sf::Int16> init;
    int ticCount;
    int val;
};

#endif
```

GuitarHero.cpp

```cpp
#include <SFML/Audio.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <math.h>
#include <limits.h>
#include <iostream>
#include <string>
#include <vector>
#include <stdexcept>
#include <exception>
#include "RingBuffer.hpp"
#include "GuitarString.hpp"

const int size = 37;

std::vector<sf::Int16> makeSamplesFromString(GuitarString gs)
{
    std::vector<sf::Int16> samples;
    gs.pluck();
    int duration = 8;
    int i;
    for(i = 0; i < (44100*duration); i++)
    {
        gs.tic();
        samples.push_back(gs.sample());
    }
    return samples;
}

int main()
{
    sf::RenderWindow window(sf::VideoMode(600, 600), "SFML Works!");
    sf::Event event;
    double f1;
    std::vector<sf::Int16> sample;
    std::vector<std::vector<sf::Int16>> samples(size);
    std::vector<sf::SoundBuffer> buffers(size);
    std::vector<sf::Sound> sounds(size);
    std::string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/'";
    for(int i = 0; i < (signed)keyboard.size(); i++)
    {
        f1 = 440.0 * pow(2, ((i-24)/12.0));
        GuitarString tmp = GuitarString(f1);
        sample = makeSamplesFromString(tmp);
        samples[i] = sample;
        if(!buffers[i].loadFromSamples(&samples[i][0], samples[i].size(), 2, 44100)){ throw
            std::runtime_error("sf::SoundBuffer: failed to load samples!"); }
        sounds[i].setBuffer(buffers[i]);
    }
```

```
    while(window.isOpen())
    {
        while(window.pollEvent(event))
        {
            if(event.type == sf::Event::TextEntered)
            {
                if(event.text.unicode < 128)
                {
                    char key = static_cast<char>(event.text.unicode);
                    for(int i = 0; i < (signed)keyboard.size(); i++)
                    {
                        if(keyboard[i] == key)
                        {
                            std::cout << "Keyboard key is: " << keyboard[i] << "\n";
                            sounds[i].play();
                            break;
                        }
                    }
                }
            }
        }
        window.clear();
        window.display();
    }
    return 0;
}
```

Makefile

```
CC = g++
CFLAGS = -g -Wall -Werror -std=c++1z -pedantic
SFLAGS = -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio
BOOST = -lboost_unit_test_framework

all: GuitarHero

GuitarHero: GuitarHero.o GuitarString.o RingBuffer.o
    $(CC) GuitarHero.o GuitarString.o RingBuffer.o -o GuitarHero $(SFLAGS)

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

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

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

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

PS6: Airport Simulation

1) Discussion

This assignment was meant to test our knowledge of concurrency as well as the use of lambda expressions. This program did this by allowing us to test on a semi-functional air traffic control simulator. I accomplished some use of the mutex and lambda expressions, however my program still does not run for 15 minutes. I attempted to implement the sync for this program, however I was slightly unsuccessful in creating a working implementation.

2) Key Algorithms/Data Structures

I tried to make use of the lambda expressions as well as the manipulation of the chrono::system_clock. However, i only did it this way because my version with only mutex and condition_variable was not functional and could not compile. But some of the key features I did implements were mutex variables, locks and lock guards.

3) What I Learned

From this assignment I got a better understanding of how mutex variables should be used in order to attain the wanted goal. I also learned more about how you can use locks and lock guards to prevent certain data from being accessed at certain times.

4) Sample Output

```
Airplane #7 is acquiring any needed runway(s) for landing on Runway 15L

Checking airport status for requested Runway 15L...
Number of simultaneous landing requests == 0, max == 0
Number of planes landing on runway 4L == 0
Number of planes landing on runway 4R == 0
Number of planes landing on runway 9 == 0
Number of planes landing on runway 14 == 0
Number of planes landing on runway 15L == 1
Number of planes landing on runway 15R == 0
Status check complete, no rule violations (yay!)
Airplane #7 is taxiing on Runway 15L for 9 milliseconds
Airplane #6 is acquiring any needed runway(s) for landing on Runway 15R
```

The rest of the program should run similar to this, but mine does not run for more than 15 minutes.

5) Errors/Unfinished Code

For this assignment I struggled with my use of the mutex variables as well as the condition variables. My code is not fully finished because it does not run for 15 minutes. It also does not contain the use of a lambda expression which was another requirement in the spec.

6) Source Code

AirportServer.cpp

```
#include <iostream>
#include <thread>
#include <condition_variable>

#include "AirportServer.h"


/**
*  Called by an Airplane when it wishes to land on a runway
*/
void AirportServer::reserveRunway(int airplaneNum, AirportRunways::RunwayNumber runway)
{
    // Acquire runway(s)
    {  // Begin critical region

//      unique_lock<mutex> runwaysLock(runwaysMutex);

        {
            lock_guard<mutex> lk(AirportRunways::checkMutex);

            cout << "Airplane #" << airplaneNum << " is acquiring any needed runway(s) for landing
                on Runway "
                 << AirportRunways::runwayName(runway) << endl;
        }

        /**
         *  ***** Add your synchronization here! *****
         */

        // Check status of the airport for any rule violations
        AirportRunways::checkAirportStatus(runway);

//      runwaysLock.unlock();

    } // End critical region

    // obtain a seed from the system clock:
    unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
    std::default_random_engine generator(seed);

```

```
40      // Taxi for a random number of milliseconds
41      std::uniform_int_distribution<int> taxiTimeDistribution(1, MAX_TAXI_TIME);
42      int taxiTime = taxiTimeDistribution(generator);
43
44      {
45          lock_guard<mutex> lk(AirportRunways::checkMutex);
46
47          cout << "Airplane #" << airplaneNum << " is taxiing on Runway " <<
                AirportRunways::runwayName(runway)
48               << " for " << taxiTime << " milliseconds\n";
49      }
50
51      std::this_thread::sleep_for(std::chrono::milliseconds(taxiTime));
52
53  } // end AirportServer::reserveRunway()
54
55
56   /**
57    *  Called by an Airplane when it is finished landing
58    */
59  void AirportServer::releaseRunway(int airplaneNum, AirportRunways::RunwayNumber runway)
60  {
61      // Release the landing runway and any other needed runways
62      { // Begin critical region
63
64  //      unique_lock<mutex> runwaysLock(runwaysMutex);
65
66          {
67              lock_guard<mutex> lk(AirportRunways::checkMutex);
68
69              cout << "Airplane #" << airplaneNum << " is releasing any needed runway(s) after landing
                    on Runway "
70                   << AirportRunways::runwayName(runway) << endl;
71          }
72
73          unique_lock<mutex> testLock(lock1);
74          {
75              lock_guard<mutex> lk2(AirportRunways::checkMutex);
76              inUse.wait(testLock, [this]{ return AirportRunways::getNumLandingRequests();});
77          }
78          testLock.unlock();
79
80          if(AirportRunways::runwayName(runway) == "4L" || AirportRunways::runwayName(runway) == "4R")
81          {
82              std::this_thread::sleep_for(std::chrono::milliseconds(1000));
83          }
84
85          if(AirportRunways::runwayName(runway) == "15R" || AirportRunways::runwayName(runway) ==
                "15L")
86          {
87              std::this_thread::sleep_for(std::chrono::milliseconds(1000));
88          }
89
90          if(AirportRunways::runwayName(runway) == "9")
91          {
92              std::this_thread::sleep_for(std::chrono::milliseconds(1000));
93          }
```

```
94
95          // Update the status of the airport to indicate that the landing is complete
96          AirportRunways::finishedWithRunway(runway);
97
98  //      runwaysLock.unlock();
99
100     } // End critical region
101
102     // obtain a seed from the system clock:
103     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
104     std::default_random_engine generator(seed);
105
106     // Wait for a random number of milliseconds before requesting the next landing for this Airplane
107     std::uniform_int_distribution<int> waitTimeDistribution(1, MAX_WAIT_TIME);
108     int waitTime = waitTimeDistribution(generator);
109
110     {
111         lock_guard<mutex> lk(AirportRunways::checkMutex);
112
113         cout << "Airplane #" << airplaneNum << " is waiting for " << waitTime << " milliseconds
                before landing again\n";
114     }
115
116     std::this_thread::sleep_for(std::chrono::milliseconds(waitTime));
117
118 } // end AirportServer::releaseRunway()
```

This was the only file provided that needed alteration for this assignment, the rest of the files were already given to s as starter code.

PS7: Kronos Time Clock-Introduction To Regular Expression Parsing

---

1) Discussion

For solving this problem I immediately started my testing of the different regex functions and provided cope examples. After deciding to use the regex_match()function, I simply found the constraints as described in the PS7 assignment .pdf and built a large if/else statement to determine whether this process was successful or if the boot failed. Following all of this, I simply added a method to print out the required statements as well as the successful boots as well as their corresponding data. Finally, I also implemented an if else statement that spans most of the code which is meant to detect if the file is actually read or not.

2) Key Algorithms/Data Structures

For this assignment, the main focus for the algorithms were around the regex variables and expressions. The regular expressions that I used in this assignment are as follows:

#1

```
boost::regex start("([0-9]+)-([0-9]+)-([0-9]+) "
"([0-9]+):([0-9]+):([0-9]+): " "(\\log.c.166)\\ server success.*");
boost::regex finish("([0-9]+)-([0-9]+)-([0-9]+) "
"([0-9]+):([0-9]+):([0-9]+).([0-9]+):INFO:"
"oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*");
```

these regex variables are used to establish a variable that we can use later touse within a regex_match() function. this is used to help us determine whether the boot is successful or not

#2

```
if(regex_match(line, s1, start)) {
```

this is the first instance of my use of the regex_match() function, for this first one, we will use our 's1' smatch variable to determine whether the regex variable 'start', which contains my parameters, matches anything in a 'line' (contains data read in from the file) of our device#_intouch.log. in this case its to supposed to report if the process fails to read anything.

#3
else if (regex_match(line, s1, finish)) {

this is the second instance of the regex_match() function. this function is to determine whether or not the process finds a boot and it will print out the corresponding boot and data if it does find a match in the file.

3) What I Learned

I learned how the use of regular expressions in a program can greatly increase the speed of certain tasks. For instance, in this assignment we intended to find whenever a boot was located, and the exact time that it happened. This taught me a real life example of how code would be used in a work environment. The regular expressions helped with this assignment in particular because it allowed us to find certain parts of a large file very quickly and with great ease.

4) Sample Output

```
(\log.c.166)\ server success.*
oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*
435759 device1_intouch.log: 2014-03-25 19:15:02 Boot Complete
 Boot Time: -1511828489 ms
436859 device1_intouch.log: 2014-03-25 19:32:44 Boot Complete
 Boot Time: -1511828489 ms
440791 device1_intouch.log: 2014-03-25 22:04:27 Boot Complete
 Boot Time: -1511828489 ms
441216 device1_intouch.log: 2014-03-26 12:50:29 Boot Complete
 Boot Time: -1511828489 ms
442432 device1_intouch.log: 2014-03-26 20:44:13 Boot Complete
 Boot Time: -1511828489 ms
443411 device1_intouch.log: 2014-03-27 14:11:42 Boot Complete
 Boot Time: -1511828489 ms
```

5) Errors/Unfinished Code

For the most part this code functioned exactly as I intended. However, according to the problem statement, my code is missing a print out for when a boot fails.

6) Source Code

ps7.cpp

```
1  // Copyright [2019] <Alex Mazzarese>
2  #include <iostream>
3  #include <fstream>
4  #include <string>
5  #include <exception>
6  #include <boost/date_time/posix_time/posix_time.hpp>
7  #include <boost/date_time/gregorian/gregorian.hpp>
8  #include <boost/regex.hpp>
9  using boost::gregorian::days;
10 using boost::gregorian::months;
11 using boost::gregorian::years;
12 using boost::gregorian::date;
13 using boost::gregorian::date_period;
14 using boost::gregorian::date_duration;
15 using boost::gregorian::from_simple_string;
16 using boost::posix_time::seconds;
17 using boost::posix_time::minutes;
18 using boost::posix_time::hours;
19 using boost::posix_time::ptime;
20 using boost::posix_time::time_duration;
21 int main(int argc, char* argv[]) {
22 std::ifstream inFile(argv[1]);
23 std::string line;
24 std::string filename = argv[1];
25 std::ofstream outFile(filename + ".rpt", std::ofstream::out);
26 date d1;
27 ptime p1;
28 outFile << "(\\log.c.166)\\ server success.*\n";
29 outFile << "oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*\n";
30 if(argc != 2) {
31 throw std::runtime_error("ERROR");}
32 else {
33 bool isValid = false;
34 int loc = 1;
35 boost::regex start("([0-9]+)-([0-9]+)-([0-9]+) "
36 "([0-9]+):([0-9]+):([0-9]+): " "(\\log.c.166)\\ server success.*");
37 boost::regex finish("([0-9]+)-([0-9]+)-([0-9]+) "
38 "([0-9]+):([0-9]+):([0-9]+).([0-9]+):INFO:"
39 "oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*");
40 if(inFile.is_open()) {
41 while(getline(inFile, line)) {
42 boost::smatch s1;
43 if(regex_match(line, s1, start)) {
44 date tmpd(stoi(s1[1]), stoi(s1[2]), stoi(s1[3]));
45 d1 = (tmpd);
46 ptime tmpp(d1, time_duration(stoi(s1[4]), stoi(s1[5]), stoi(s1[6])));
47 p1 = tmpp;
```

```
if(isValid) {
outFile << "INCOMPLETE BOOT\n";
isValid = false;}
outFile << "DEVICE BOOT\n" << loc << " " << argv[1] << ": "
<< s1[1] << "-" << s1[2] << "-" << s1[3] << " " << s1[4]
<< ":" << s1[5] << ":" << s1[6] << " " <<"Boot Start\n";
isValid = true;}
else if (regex_match(line, s1, finish)) {
date tmpd2(stoi(s1[1]), stoi(s1[2]), stoi(s1[3]));
ptime tmpp2(tmpd2, time_duration(stoi(s1[4]), stoi(s1[5]), stoi(s1[6])));
outFile << loc << " " << argv[1] << ": " << s1[1] << "-" << s1[2] << "-"
<< s1[3] << " " << s1[4] << ":" << s1[5] << ":" << s1[6] << " Boot Complete\n";
time_duration ttotal = tmpp2 - p1;
int finalTime = ttotal.total_milliseconds();
outFile << " " << "Boot Time: " << finalTime << " ms\n";
isValid = false;}
loc++;}
outFile.close();
inFile.close();}
else {
throw std::runtime_error("ERROR");}
}
return 0;
}
```

Makefile

```
CC = g++
CFLAGS = -g -Wall -Werror -std=c++0x -pedantic
BOOST = -lboost_regex -lboost_date_time

all = ps7

ps7: ps7.o
    $(CC) ps7.o -o ps7 $(BOOST)

ps7.o: ps7.cpp
    $(CC) -c ps7.cpp $(CFLAGS)

clean:
    rm *.o
    rm ps7
```