

Computing IV Sec 202: Project Portfolio

Camden Andersson

Spring 2024

Contents

1	PS0: Hello SFML	2
2	PS1: LFSR	5
3	PS2: Pythagoras Tree	12
4	PS3: Sokoban	16
5	PS4: NBody	31
6	PS5: DNA Sequence Alignment	42
7	PS6: Random Writer	48
8	PS7: Kronos Log Parsing	55

1 PS0: Hello SFML

1.1 What I accomplished

The assignment aimed to familiarize me with the setup and use of a build environment for software development with the Simple and Fast Multimedia Library (SFML). The primary objective was to configure a development environment using GCC and Make, and to craft a simple SFML application that demonstrated the correct setup by displaying a basic graphical window. This window initially featured a simple green circle to ensure everything was functioning correctly.

In my solution, I built upon the basic requirements by adding interactive and dynamic elements to the application. I introduced a sprite to the SFML window, then I programmed this sprite to respond to user inputs, allowing it to move across the window based on keystroke commands. This enhancement not only tested my ability to manipulate graphical objects within SFML but also provided a practical demonstration of handling user interactions within a graphical application. The implementation involved using SFML's graphics and window management features to control the sprite's position dynamically and to update the display in real-time. This extension of the demo code allowed me to explore more of SFML's capabilities, particularly in terms of animation and input handling.

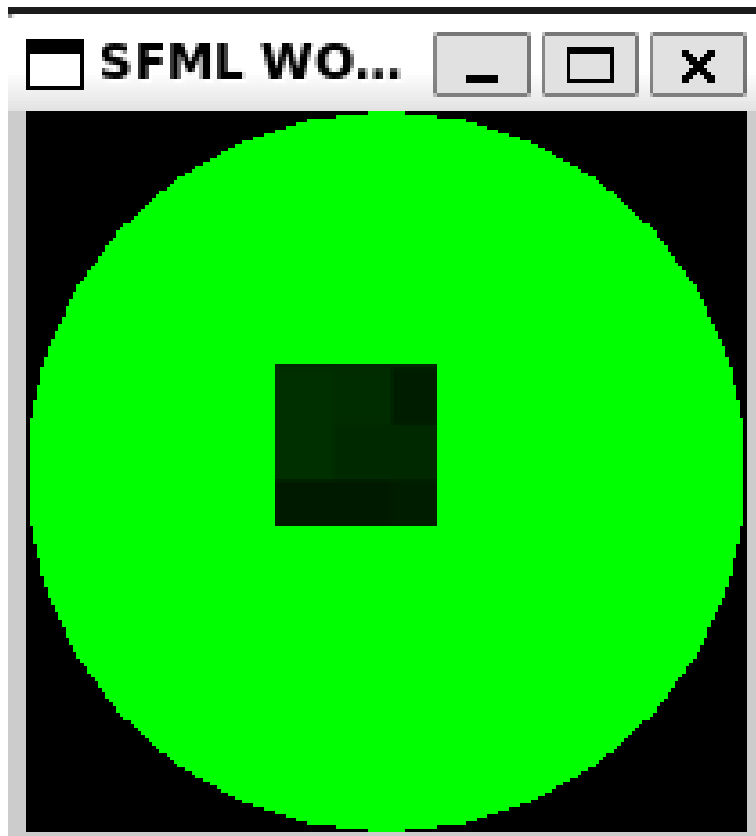


Figure 1: Window produced showcasing the sprite and circle

1.2 What I already knew

C++ itself was the only knowledge I had going in, as the project itself was about learning the very basics of SFML, which up to the point of taking the class I was completely unfamiliar with. Additionally, this was the first time since Computing II we had to use Makefiles, which was a large re-adjustment.

1.3 What I learned

I learned the very basics of SFML, such as creating a window loop and taking input from the keyboard. I also re-learned how to use and set up Makefiles for future projects, and what is expected of us in our code format from the use of CPPLINT.

1.4 Challenges

This was the first project, so naturally it was no challenge and was completed in minimal time.

1.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS =
6 # Your compiled .o files
7 OBJECTS =
8 # The name of your program
9 PROGRAM = sfml-app
10
11 .PHONY: all clean lint
12
13
14 all: $(PROGRAM)
15
16 # Wildcard recipe to make .o files from corresponding .cpp file
17 %.o: %.cpp $(DEPS)
18     $(CC) $(CFLAGS) -c $<
19
20 $(PROGRAM): main.o $(OBJECTS)
21     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
22
23 clean:
24     rm *.o $(PROGRAM)
25
26 lint:
27     cpplint *.cpp *.hpp
```

```
1 // Copyright Camden Andersson 2023
2
3 #include <SFML/Graphics.hpp>
4
5 int main() {
6     sf::RenderWindow window(sf::VideoMode(200, 200), "SFML WORKS!");
7     sf::CircleShape shape(100.f);
8     shape.setFillColor(sf::Color::Green);
9
10    sf::Texture texture;
11
12    if (!texture.loadFromFile("sprite.png")) {
13        std::exit(1);
14    }
15
16    sf::Sprite wood;
17    wood.setTexture(texture);
18    wood.setTextureRect(sf::IntRect(15, 15, 45, 45));
19
20    while (window.isOpen()) {
21        sf::Event event;
22        while (window.pollEvent(event)) {
23            if (event.type == sf::Event::Closed)
24                window.close();
```

```
25     }
26
27     window.clear();
28     window.draw(shape);
29     window.draw(wood);
30     window.display();
31 }
32
33
34 return 0;
35 }
```

2 PS1: LFSR

2.1 What I accomplished

In PS1a: LFSR/PhotoMagic, you're tasked with developing a 16-bit Fibonacci LFSR simulation within the PhotoMagic namespace, focusing on bit-level operations in C++. This involves creating the FibLFSR class, which uses XOR operations across three "tap" positions to generate pseudo-random bit sequences and functions to step through the LFSR operations. The assignment requires implementing and testing these functionalities using the Boost test framework, culminating in a static library PhotoMagic.a that includes the LFSR implementation. The project also tests your skills in build process management using Makefiles and code validation through testing.

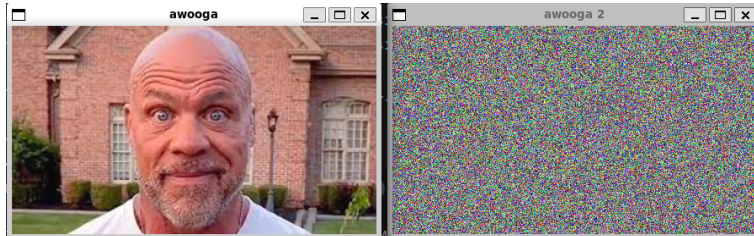


Figure 2: Windows produced from the encryption

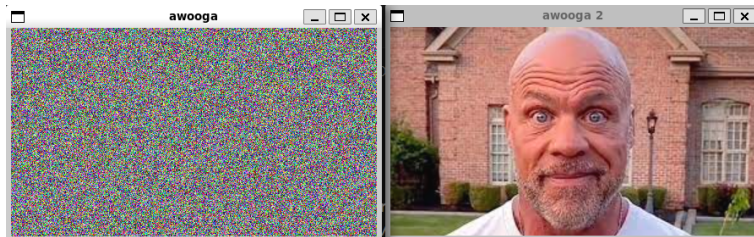


Figure 3: Windows produced from the decryption

My solution focused on the efficient and effective implementation of a 16-bit Fibonacci Linear Feedback Shift Register (LFSR) encapsulated within the FibLFSR class in the PhotoMagic namespace. Opting to represent the LFSR with a single integer, my approach made use of bitwise operations to manipulate bits directly, which is both computationally efficient and straightforward. This integer-based representation enabled simple implementation of essential operations such as shifts and XORs, which are vital for the LFSR's functionality.

The core functionality of my FibLFSR class includes a step function that performs one iteration of the LFSR sequence. In this function, specific bits at predefined "tap" positions are XORed together to generate a new bit, which is then used to update the LFSR's state, mimicking the feedback process found in Fibonacci LFSRs. Additionally, a generate function was implemented to produce a sequence of bits, useful for tasks requiring multiple bits, such as encrypting an image by altering pixel values. To facilitate debugging and visualization, I also overloaded the stream insertion operator to display the LFSR's current state.

This project not only demonstrated my capability to handle low-level bit manipulations in C++ but also reinforced my skills in using development tools like Makefiles for building processes and the Boost testing framework for validating the functionality of my code. This comprehensive approach provided a robust foundation for applying the LFSR to practical encryption tasks, showcased by encrypting images where the LFSR sequence influenced the RGB values of pixels, effectively altering the visual content.

2.2 What I already knew

I already knew the bitshift operations we were using (OR, XOR, AND), etc. These are vital to use in the step() function. The basics of SFML that were learned last project were put to use immediately in this project, as you need to display two windows showing the input and output of encrypting your image with your LFSR.

2.3 What I learned

I learned what a FibLFSR is, how to implement one, and what it can be used for. Additionally, I learned how to use the boost library to test your implementation to check my codes validity. This is a key skill that was learned during this assignment, showing its importance later on.

2.4 Challenges

Learning what a FibLFSR is. It took me a while to fully understand the concept of what it was, but after understanding it, the implementation wasn't so difficult. Learning and testing using the boost library was also somewhat challenging as well, as you need to think carefully about how to implement your tests to get them to make any sense. I would like to look more into the boost library at its use, while seeming daunting initially, showed the importance of testing your solution.

2.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = FibLFSR.hpp PhotoMagic.hpp
6 # Your compiled .o files
7 OBJECTS = FibLFSR.o PhotoMagic.o
8 # The name of your program
9 PROGRAM = PhotoMagic
10
11 SLIB = PhotoMagic.a
12
13 .PHONY: all clean lint
14
15 # Wildcard recipe to make .o files from corresponding .cpp file
16 #%.o: %.cpp $(DEPS)
17 # $(CC) $(CFLAGS) -c $<
18
19 all: $(PROGRAM) PhotoMagic.a test
20
21 main.o: main.cpp FibLFSR.hpp PhotoMagic.hpp
22     $(CC) $(CFLAGS) -c $< -o $@
23
24 PhotoMagic.o: PhotoMagic.cpp PhotoMagic.hpp
25     $(CC) $(CFLAGS) -c $< -o $@
26
27 FibLFSR.o: FibLFSR.cpp FibLFSR.hpp
28     $(CC) $(CFLAGS) -c $< -o $@
29
30 PhotoMagic: PhotoMagic.o main.o FibLFSR.o
31     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
32
33 test: test.o FibLFSR.o
34     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
35
36 test.o: test.cpp FibLFSR.hpp
37     $(CC) $(CFLAGS) -c $< -o $@
38
39 PhotoMagic.a: PhotoMagic.o FibLFSR.o
40     ar rcs PhotoMagic.a FibLFSR.o PhotoMagic.o
41
```

```
42
43
44 clean:
45     rm *.o *.a $(PROGRAM) test *.gch
46
47 lint:
48     cpplint *.cpp *.hpp
```

```
1  // Copyright 2024 Camden Andersson
2  // pixels.cpp:
3  // using SFML to load a file, manipulate its pixels, write it to disk
4
5
6
7  // g++ -o pixels pixels.cpp -lsfml-graphics -lsfml-window
8  #include <iostream>
9  #include <SFML/System.hpp>
10 #include <SFML/Window.hpp>
11 #include <SFML/Graphics.hpp>
12 #include "PhotoMagic.hpp"
13 int main(int argc, char* argv[]) {
14     sf::Image image;
15     if (!image.loadFromFile(argv[1]))
16         return -1;
17
18     std::cout << "loaded image" << std::endl;
19
20     sf::Vector2u size = image.getSize();
21     sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "awooga");
22     sf::RenderWindow window2(sf::VideoMode(size.x, size.y), "awooga 2");
23
24     std::cout << "windows made" << std::endl;
25
26     sf::Texture texture;
27     texture.loadFromImage(image);
28
29     sf::Sprite sprite;
30     sprite.setTexture(texture);
31
32     // transform image
33     std::string seedSTR = argv[3];
34     PhotoMagic::FibLFSR seed(seedSTR);
35     PhotoMagic::FibLFSR* ptr = &seed;
36     PhotoMagic::transform(image, ptr);
37     std::cout << "seed loaded into transform" << std::endl;
38
39     sf::Texture texture2;
40     texture2.loadFromImage(image);
41
42     sf::Sprite sprite2;
43     sprite2.setTexture(texture2);
44
45     std::cout << "texture loaded and sprites created, drawing window:" <<
std::endl;
46
47     while (window1.isOpen() && window2.isOpen()) {
48         sf::Event event;
49         while (window1.pollEvent(event)) {
50             if (event.type == sf::Event::Closed)
51                 window1.close();
```

```

52     }
53     while (window2.pollEvent(event)) {
54         if (event.type == sf::Event::Closed)
55             window2.close();
56     }
57
58     window1.clear();
59     window1.draw(sprite);
60     window1.display();
61
62     window2.clear();
63     window2.draw(sprite2);
64     window2.display();
65 }
66
67
68 // fredm: saving a PNG segfaults for me, though it does properly
69 // write the file
70 if (!image.saveToFile(argv[2]))
71     return -1;
72
73 return 0;
74 }

```

```

1 // Copyright 2023 Camden Andersson
2 #ifndef PHOTOMAGIC_HPP
3 #define PHOTOMAGIC_HPP
4
5 #include "FibLFSR.hpp"
6
7 using PhotoMagic::FibLFSR;
8
9 namespace PhotoMagic {
10 // Transforms image using FibLFSR
11 void transform(sf::Image&, FibLFSR*);
12 // Display an encrypted copy of the picture, using the LFSR to do the
   encryption
13 }
14 #endif

```

```

1 // Copyright 2023 Camden Andersson
2 #include "PhotoMagic.hpp"
3
4 namespace PhotoMagic {
5 // Transforms image using FibLFSR
6 // take the iamge as an input, along with the fib lsfr.
7 // go pixel by pixel stepping once to transform each pixel
8
9 void transform(sf::Image& image, FibLFSR* lfsr) {
10     int vert, hori;
11     sf::Color pixel;
12
13     sf::Vector2u imageSize = image.getSize();
14
15     vert = imageSize.x;
16     hori = imageSize.y;
17     // look at pdf for explanation on what to do
18     for (int i = 0; i < hori; i++) {
19         for (int j = 0; j < vert; j++) {
20             pixel = image.getPixel(j, i);

```



```

21     pixel.r = pixel.r ^ lfsr->generate(8);
22     pixel.g = pixel.g ^ lfsr->generate(8);
23     pixel.b = pixel.b ^ lfsr->generate(8);
24     image.setPixel(j, i, pixel);
25 }
26 }
27 }
28 // Display an encrypted copy of the picture, using the LFSR to do the
    encryption
29 } // namespace PhotoMagic

```

```

1 // Copyright 2024 Camden Andersson
2 #ifndef FibLFSR_HPP
3 #define FibLFSR_HPP
4
5 #include <iostream>
6 #include <string>
7 #include <sstream>
8 #include <SFML/System.hpp>
9 #include <SFML/Window.hpp>
10 #include <SFML/Graphics.hpp>
11
12 namespace PhotoMagic {
13 class FibLFSR {
14 public:
15     // Constructor to create LFSR with the given initial seed
16     explicit FibLFSR(std::string seed);
17     // Simulate one step and return the new bit as 0 or 1
18     int step();
19     // Simulate k steps and return a k-bit integer
20     int generate(int k);
21     // friend output operator
22     friend std::ostream& operator<<(std::ostream& out, const FibLFSR&
    lfsr);
23     // Transforms image using FibLFSR
24     void transform(sf::Image&, FibLFSR*);
25     // Display an encrypted copy of the picture, using the LFSR to do
    the encryption
26 private:
27     // Any fields that you need
28     // stored lfsr as an int
29     unsigned int storedLFSR = 0;
30 };
31 } // namespace PhotoMagic
32 #endif

```

```

1 // Copyright 2024 Camden Andersson
2 #include "FibLFSR.hpp"
3
4
5
6 /*
7 Take the seed as an input of 16 bits
8Xor bits 15, 13, 12 and 10 , output to bit 1
9*/
10
11 namespace PhotoMagic {
12 FibLFSR::FibLFSR(std::string seed) {
13     storedLFSR = std::stoi(seed, 0, 2);
14 }

```

```

15     // for testing
16     // std::cout << storedLFSR;
17 }
18
19 int FibLFSR::step() {
20     int newBit = ((((((storedLFSR >> 15) & 1) ^ ((storedLFSR >> 13) & 1)))
21     ^ ((storedLFSR >> 12) & 1)) ^ ((storedLFSR >> 10) & 1));
22     storedLFSR = storedLFSR << 1;
23     storedLFSR = storedLFSR | newBit;
24
25     return newBit;
26 }
27
28 int FibLFSR::generate(int k) {
29     int result = 0;
30
31     for (int i = 0; i < k; ++i) {
32         result = (result << 1) | step();
33     }
34     return result;
35 }
36
37 std::ostream& operator<<(std::ostream& out, const PhotoMagic::FibLFSR& lfsr)
38 {
39     for (int i = 15; i >= 0; --i) {
40         out << (((lfsr.storedLFSR) >> i) & 1);
41     }
42     return out;
43 }
44 } // namespace PhotoMagic

```

```

1 // Copyright 2022
2 // By Dr. Rykalova
3 // Editted by Dr. Daly
4 // test.cpp for PS1a
5 // updated 1/8/2024
6
7 #include <iostream>
8 #include <string>
9 #include <sstream>
10 #include "FibLFSR.hpp"
11
12
13 #define BOOST_TEST_DYN_LINK
14 #define BOOST_TEST_MODULE Main
15 #include <boost/test/unit_test.hpp>
16
17 using PhotoMagic::FibLFSR;
18
19 BOOST_AUTO_TEST_CASE(testStepInstr) {
20     FibLFSR l("1011011000110110");
21     BOOST_REQUIRE_EQUAL(l.step(), 0);
22     BOOST_REQUIRE_EQUAL(l.step(), 0);
23     BOOST_REQUIRE_EQUAL(l.step(), 0);
24     BOOST_REQUIRE_EQUAL(l.step(), 1);
25     BOOST_REQUIRE_EQUAL(l.step(), 1);
26     BOOST_REQUIRE_EQUAL(l.step(), 0);
27     BOOST_REQUIRE_EQUAL(l.step(), 0);
28     BOOST_REQUIRE_EQUAL(l.step(), 1);

```

```

29 }
30
31 BOOST_AUTO_TEST_CASE(testGenerateInstr) {
32     FibLFSR l("1011011000110110");
33     BOOST_REQUIRE_EQUAL(l.generate(9), 51);
34 }
35
36 // new test cases
37
38 BOOST_AUTO_TEST_CASE(testOutput) {
39     FibLFSR l("1011011000110110");
40     BOOST_REQUIRE_EQUAL(l.generate(9), 51);
41     std::cout << l << std::endl;
42 }
43
44
45 BOOST_AUTO_TEST_CASE(testGenStep) {
46     FibLFSR l("1011011000110110");
47     BOOST_REQUIRE_EQUAL(l.generate(6), 6);
48     BOOST_REQUIRE_EQUAL(l.step(), 0);
49     BOOST_REQUIRE_EQUAL(l.step(), 1);
50 }
51
52 BOOST_AUTO_TEST_CASE(testGenSStream) {
53     FibLFSR l("1011011000110110");
54     std::stringstream ss;
55     l.generate(8);
56     ss << l;
57     BOOST_REQUIRE_EQUAL(ss.str(), "0011011000011001");
58 }
59
60 BOOST_AUTO_TEST_CASE(testDifferentSeed) {
61     FibLFSR l("1011010000000001");
62     BOOST_REQUIRE_EQUAL(l.generate(6), 17);
63 }
64
65 BOOST_AUTO_TEST_CASE(testListStep) {
66     FibLFSR l("1011011000110110");
67     BOOST_REQUIRE_EQUAL(l.step(), 0);
68     BOOST_REQUIRE_EQUAL(l.step(), 0);
69     BOOST_REQUIRE_EQUAL(l.step(), 0);
70     BOOST_REQUIRE_EQUAL(l.step(), 1);
71     BOOST_REQUIRE_EQUAL(l.step(), 1);
72     BOOST_REQUIRE_EQUAL(l.step(), 0);
73     BOOST_REQUIRE_EQUAL(l.step(), 0);
74     BOOST_REQUIRE_EQUAL(l.step(), 1);
75     BOOST_REQUIRE_EQUAL(l.step(), 1);
76     BOOST_REQUIRE_EQUAL(l.step(), 0);
77     BOOST_REQUIRE_EQUAL(l.step(), 0);
78 }

```

3 PS2: Pythagoras Tree

3.1 What I accomplished

In PS2: Recursive Graphics (Pythagoras Tree), the assignment tasked me with creating a program that plots a Pythagoras tree, a fractal constructed from squares forming right triangles, illustrating the Pythagorean theorem. This mathematical model, named after Pythagoras, was invented by Albert E Bosman in 1942, and the assignment required implementing this using the SFML (Simple and Fast Multimedia Library). My program needed to take two command-line arguments: the length of one side of the base square (L) and the depth of the recursion (N).

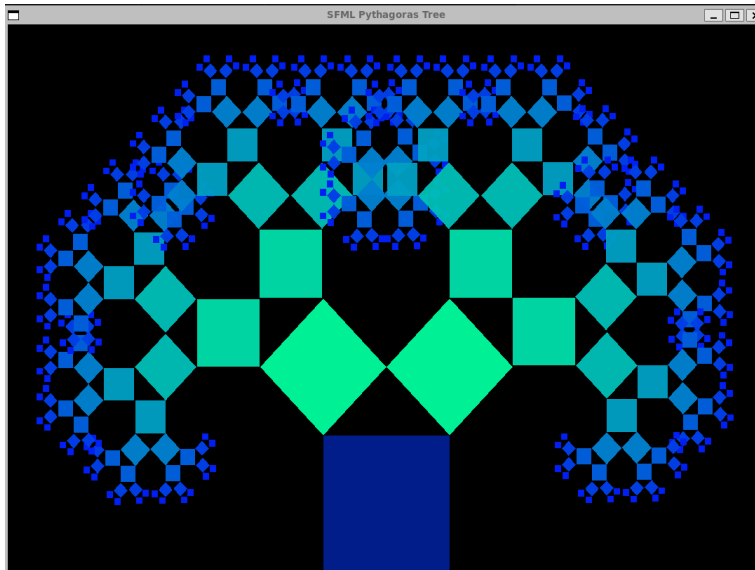


Figure 4: Window produced from running the program

In my solution, I focused on developing a recursive function to draw this fractal tree. I utilized the `sf::Drawable` class for drawing each part of the tree and handled the recursive drawing by calling a helper function from the main recursive function `pTree()`. For each recursion level, two smaller squares were drawn at right angles to each other, adhering to the mathematical principles outlined for the tree construction. I managed the geometry by calculating the side lengths of the child squares using trigonometric functions, specifically using cosine and sine based on a 45-degree angle.

I employed SFML's `sf::RectangleShape` (instead of `sf::ConvexShape`) to render squares and handled transformations such as rotation and scaling based on the depth of recursion to maintain the fractal's structure. Moreover, I ensured that the drawing's orientation and scaling were accurate by setting the shapes' origins appropriately, which was critical for the correct application of rotation transformations. The SFML window was sized dynamically based on the input parameter L to prevent the fractal from spilling over the window boundaries, optimizing the space usage according to the tree size. For extra credit, I incorporated multiple colors to differentiate between levels of the tree and allowed modifications of the inner triangle's angle to explore different fractal patterns. This extension provided a rich, visually engaging representation of the Pythagoras tree, showcasing my ability to integrate mathematical concepts with graphic programming in C++.

3.2 What I already knew

Before starting this assignment, I was already familiar with the mathematical concepts of sine and cosine functions, as it's a crucial concept in geometry. This foundational knowledge was crucial for handling the rotations of squares in the project. Additionally, my prior experience with SFML from previous projects proved invaluable, as it directly influenced how graphical objects were manipulated and drawn in the application.

3.3 What I learned

Through this project, I deepened my understanding of applying the `cos()` and `sin()` functions to achieve rotation effects in graphical programming. Moreover, I learned how to use

recursion effectively to construct a Pythagoras Tree, which introduced me to the concept of plane fractals—a term and mathematical model I had not encountered before.

3.4 Challenges

The most significant challenge I faced was grasping the complexities of rotational transformations for each square in the fractal structure. To overcome this, I found it extremely helpful to sketch out the transformations (as directed in class), which clarified the relationships and positioning of the squares. This visualization was instrumental in adjusting my `newLeft` and `newRight` variables accurately, ensuring the fractal's correct creation.

3.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = PTree.hpp
6 # Your compiled .o files
7 OBJECTS = PTree.o
8 # The name of your program
9 PROGRAM = PTree
10
11 .PHONY: all clean lint
12
13 all: $(PROGRAM)
14
15 all: $(PROGRAM)
16
17 # Wildcard recipe to make .o files from corresponding .cpp file
18 %.o: %.cpp $(DEPS)
19     $(CC) $(CFLAGS) -c $<
20
21 $(PROGRAM): main.o $(OBJECTS)
22     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
23
24 clean:
25     rm *.o $(PROGRAM)
26
27 lint:
28     cpplint *.cpp *.hpp
```

```
1 // Copyright 2023 Camden Andersson
2 #include <string>
3 #include "PTree.hpp"
4
5 int main(int argc, char* argv[]) {
6     int sideLength = std::atoi(argv[1]);
7     int recursionAmt = std::atoi(argv[2]);
8
9     sf::RenderWindow window(sf::VideoMode((sideLength * 6),
10 (sideLength * 4)), "SFML Pythagoras Tree");
11
12     while (window.isOpen()) {
13         sf::Event event;
14         while (window.pollEvent(event)) {
15             if (event.type == sf::Event::Closed)
16                 window.close();
```

```

17     }
18     PTree pTree;
19
20     window.clear();
21     pTree.pTree(window, (sideLength * 2.5), (sideLength * 3), sideLength
, 0, recursionAmt);
22
23     window.display();
24 }
25
26
27
28     return 0;
29 }

```

```

1 // Copyright 2023 Camden Andersson
2
3 #include <iostream>
4 #include <cmath>
5 #include <SFML/Graphics.hpp>
6
7 class PTree: public sf::RectangleShape {
8 public:
9     void pTree(sf::RenderTarget& window, int x, int y,
10     int L, double angle, int depth);
11 };

```

```

1 // Copyright 2023 Camden Andersson
2 #include "PTree.hpp"
3
4 const double ANGLE = M_PI / 4.0; // 45 degrees in radians
5
6 void PTree::pTree(sf::RenderTarget& window, int x, int y,
7 int L, double angle, int depth) {
8     if (depth <= 0) {
9         return;
10    }
11
12    sf::RectangleShape rect(sf::Vector2f(L, L));
13    // rect.setFillColor(sf::Color(0,0,0,255));
14    rect.setPosition(x, y);
15    // rect.setSize(sf::Vector2f(L,L));
16    rect.setRotation(angle * (180 / M_PI));
17    // rect.setOutlineColor(sf::Color::White);
18    // rect.setOutlineThickness(1);
19    rect.setFillColor(sf::Color
20    { 0xff9ff * (static_cast<sf::Uint32>(depth) * 2)});
21    window.draw(rect);
22
23    // Look at the 4 points of the previous rect you drew
24    sf::Vector2f vTL = rect.getTransform().transformPoint(rect.getPoint(0));
25    sf::Vector2f vTR = rect.getTransform().transformPoint(rect.getPoint(1));
26    // sf::Vector2f vBR = rect.getTransform().transformPoint(rect.getPoint
27    (2));
28    // sf::Vector2f vBL = rect.getTransform().transformPoint(rect.getPoint
29    (3));
30    // int newRightX = vTL.x - (L / sqrt(2)) * cos(angle + (M_PI / 4));
31    // int newRightY = vTL.y - (L / sqrt(2)) * sin(angle + (M_PI / 4));

```

```
31 // pTree(window, newRightX, newRightY, L / sqrt(2), (angle - (M_PI / 4)
32 ), depth - 1);
33
34 int newLeftX = vTL.x - (L / sqrt(2)) * cos(angle + (M_PI / 4));
35 int newLeftY = vTL.y - (L / sqrt(2)) * sin(angle + (M_PI / 4));
36
37 int newRightX = vTR.x + (L) * sin(angle);
38 int newRightY = vTR.y - (L) * cos(angle);
39
40 pTree(window, newLeftX, newLeftY, L / sqrt(2),
41 (angle - (M_PI / 4)), depth - 1);
42
43 pTree(window, newRightX, newRightY, L / sqrt(2),
44 (angle + (M_PI / 4)), depth - 1);
45 }
```

4 PS3: Sokoban

4.1 What I accomplished

In PS3b: Sokoban, the assignment was to enhance a basic Sokoban game by integrating gameplay mechanics that allow a player to move and interact within a predefined grid environment, using keyboard inputs (WASD or arrow keys). The gameplay involved moving the player character around the grid to push boxes into designated storage areas, with walls and other boxes potentially blocking movement. The game is won when all boxes are successfully placed in their respective storage areas. The task also included implementing functions to handle the game's winning condition, player movement, and game resetting.



Figure 5: Window produced from running the program

My solution for PS3b focused on expanding the Sokoban game's interactivity and playability. I implemented the `movePlayer` method in the `Sokoban` class, which accepted a direction input and moved the player accordingly, considering the presence of walls and the ability to push boxes into open spaces. This required careful handling of grid-based movement and collision detection to ensure the game mechanics worked as intended.

I also implemented the `isWon` method to check if all boxes were correctly placed in their storage locations, effectively determining the end of the game. This is also where my lambda function was located. To manage game resets, I incorporated functionality to revert the game to its initial state whenever the player pressed 'R', or to go back one move once you press 'U', which was particularly useful during development and testing for quickly retrying game scenarios.

Handling the player's movement involved calculating the potential new position based on the current direction and checking if the space was free, blocked by a wall, or had a box that could be pushed. If a box was in the player's path and could be moved (i.e., the space beyond the box was free), I updated the positions of both the player and the box. Otherwise, the move was blocked.

I also implemented robust collision detection, which helped to manage the various game states effectively. This included ensuring that the game accurately recognized win conditions and prevented illegal moves, such as pushing multiple boxes or moving into walls. Additionally, I had to carefully design the user interface to provide clear feedback on game actions, such as movement success or failure, and game reset or win scenarios. This was complemented by visual cues and messages to enhance the player's experience and engagement with the game.

4.2 What I already knew

Prior to starting Sokoban, I was already well-acquainted with the Sokoban game concept, having played similar games before. This familiarity significantly eased the implementation process, as I had a clear understanding of the game mechanics and objectives. Additionally, I was versed in handling data structures in both column and row major order, choosing to use column major order for this project due to its ease of use in the context of the game's grid-based logic.

4.3 What I learned

Throughout the development of Sokoban, I gained valuable insights into the complexities of game management and code maintainability. One of the key lessons was the importance of writing clean and concise code. Managing the intricacies of game logic alongside the graphical representation of the game underscored how crucial it is to maintain a well-organized codebase. This experience deepened my understanding of the interplay between game logic and its visual representation, shedding light on how these two aspects are integrated in real-time applications.

4.4 Challenges

The primary challenge in this project was managing and navigating my own code as I integrated more features. The complexity increased significantly with the addition of numerous nested if-else statements, which made the codebase cumbersome to modify and maintain. Although the project itself presented manageable difficulties, the real test was ensuring that the expanding code remained functional and accessible for future adjustments. This experience highlighted the necessity of adopting strategies to keep the codebase manageable despite its growing complexity.

4.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = Sokoban.hpp
6 # Your compiled .o files
7 OBJECTS = Sokoban.o
8 # The name of your program
9 PROGRAM = Sokoban
10
11 .PHONY: all clean lint
12
13 all: $(PROGRAM) test Sokoban.a
14
15 # Wildcard recipe to make .o files from corresponding .cpp file
16 %.o: %.cpp $(DEPS)
17     $(CC) $(CFLAGS) -c $<
18
19 $(PROGRAM): main.o $(OBJECTS)
20     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
21
22 Sokoban.a: Sokoban.o
23     ar rcs Sokoban.a Sokoban.o
24
25 test: test.o Sokoban.o
26     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
27
28 test.o: test.cpp Sokoban.hpp
```

```

29 $(CC) $(CFLAGS) -c $< -o $@
30
31 clean:
32     rm *.o $(PROGRAM) *.a
33
34 lint:
35     cpplint *.cpp *.hpp

```

```

1  // Copyright 2024 Camden Andersson
2
3  #include <iostream>
4  #include <SFML/Graphics.hpp>
5  #include <SFML/Window.hpp>
6  #include <SFML/Audio/Sound.hpp>
7  #include <SFML/Audio/SoundBuffer.hpp>
8  #include "Sokoban.hpp"
9
10 int main(int argc, char* argv[]) {
11     // Create the sokoban object
12     SB::Sokoban sokoban;
13     sokoban.setLevelName(argv[1]);
14     std::ifstream file;
15     file.open(argv[1]);
16     file >> sokoban;
17
18     file.close();
19     sf::SoundBuffer buffer;
20     sf::Sound sound;
21     buffer.loadFromFile("sound.wav");
22     sound.setBuffer(buffer);
23
24     // Play the game (draw the level for now)
25     sf::RenderWindow window(sf::VideoMode(64 * sokoban.width(),
26     64 * sokoban.height()), "Sokoban Game");
27     bool soundPlayed = false; // Variable to track if the sound has been
    played
28     while (window.isOpen()) {
29         sf::Event event;
30         while (window.pollEvent(event)) {
31             if (event.type == sf::Event::Closed)
32                 window.close();
33
34             window.clear();
35
36             SB::Direction playerDirection;
37             if (event.type == sf::Event::KeyReleased) {
38                 sf::Keyboard::Key key = event.key.code;
39                 std::cout << "Key released: " << event.key.code << std::endl
    ;
40
41                 if (sokoban.isWon()) {
42                     if (key == sf::Keyboard::R)
43                         sokoban.resetLevel();
44                 } else {
45                     switch (key) {
46                         case sf::Keyboard::W:
47                         case sf::Keyboard::Up:
48                             playerDirection = SB::Up;
49                             sokoban.movePlayer(playerDirection);
50                             break;
51                         case sf::Keyboard::A:

```

```

51         case sf::Keyboard::Left:
52             playerDirection = SB::Left;
53             sokoban.movePlayer(playerDirection);
54             break;
55         case sf::Keyboard::S:
56         case sf::Keyboard::Down:
57             playerDirection = SB::Down;
58             sokoban.movePlayer(playerDirection);
59             break;
60         case sf::Keyboard::D:
61         case sf::Keyboard::Right:
62             playerDirection = SB::Right;
63             sokoban.movePlayer(playerDirection);
64             break;
65         case sf::Keyboard::R:
66             sokoban.resetLevel();
67             break;
68         case sf::Keyboard::U:
69             sokoban.resetLevelBack();
70         default:
71             break;
72     }
73 }
74 }
75 window.draw(sokoban);
76
77 if (sokoban.isWon()) {
78     if (!soundPlayed) {
79         sound.play();
80         soundPlayed = true; // Mark the sound as played
81     }
82     // Set up the font
83     sf::Font font;
84     if (!font.loadFromFile("arial.ttf")) {
85         return 1; // Exit if the font file is not found
86     }
87     // Create a text object
88     sf::Text text("You Won!", font, 50);
89     text.setFillColor(sf::Color::Green);
90     // Center the text
91     sf::FloatRect textRect = text.getLocalBounds();
92     text.setOrigin(textRect.left + textRect.width / 2.0f,
93         textRect.top + textRect.height / 2.0f);
94     text.setPosition(sf::Vector2f(window.getSize().x / 2.0f,
95         window.getSize().y / 2.0f));
96
97     // Draw the text
98     window.draw(text);
99 }
100 // display window
101 window.display();
102 }
103 }
104 }

```

```

1 // Copyright 2024 Camden Andersson
2 #include <fstream>
3 #include <string>
4 #include <iostream>
5 #include <vector>

```

```

6  #include <algorithm>
7  #include <SFML/Graphics.hpp>
8  namespace SB {
9  enum Direction {Up, Down, Left, Right};
10 class Sokoban : public sf::Drawable {
11     public:
12         Sokoban();
13         // void fillTextures();
14         void setInitialPlayerLoc();
15         int width(void) const;
16         int height(void) const;
17         sf::Vector2i playerLoc() const;
18         void movePlayer(enum Direction _direction);
19         bool isWon() const;
20         void setPlayerX(int x);
21         void setPlayerY(int y);
22         void setLevelName(std::string _levelName);
23         std::string getLevelName(void);
24         char getGridData(int x, int y);
25         char getGridOriginalData(int x, int y);
26         void setGridData(int x, int y, char c);
27         void resetLevel();
28         void fillOriginalGrid();
29         bool isBoxOrStoredBox(int x, int y);
30         bool isWall(int x, int y);
31         bool isOrigSpaceStorage(int x, int y);
32         void fillPreviousGrid();
33         void resetLevelBack();
34         friend std::ifstream& operator>>(std::ifstream& is, SB::Sokoban& grid);
35
36     protected:
37         virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
38             const;
39
40     private:
41         std::vector<std::vector<char>> gridData; // The 2d grid
42         std::vector<std::vector<char>> gridDataOriginal; // unmodified grid to
43             allow for a reset
44         std::vector<std::vector<char>> gridDataPrevious; // The 2d grid before
45             we move
46         std::string levelName;
47         int internalWidth;
48         int internalHeight;
49         Direction lastDirection;
50         int playerX;
51         int playerY;
52         sf::Texture texturePlayer05;
53         sf::Texture texturePlayer08;
54         sf::Texture texturePlayer17;
55         sf::Texture texturePlayer20;
56         sf::Texture textureBlock06;
57         sf::Texture textureCrate03;
58         sf::Texture textureEnviroment03;
59         sf::Texture textureGround01;
60         sf::Texture textureGround04;
61 };
62 } // namespace SB

```

```

1  // Copyright 2024 Camden Andersson
2  #include "Sokoban.hpp"

```

```

3
4 namespace SB {
5 // make it take in an ifstream and output an ifstream
6 std::ifstream& operator>>(std::ifstream& is, SB::Sokoban& grid) {
7     // is.open(grid.levelName);
8     if (!is.is_open()) {
9         std::cerr << "Error opening file." << std::endl;
10        return is;
11    }
12
13    // Read the width and height
14    is >> grid.internalWidth >> grid.internalHeight;
15
16    // Resize the grid
17    grid.gridData.resize(grid.internalWidth, std::vector<char>(grid.
internalHeight));
18
19    // Read the grid data
20    for (int i = 0; i < grid.internalWidth; ++i) {
21        for (int j = 0; j < grid.internalHeight; ++j) {
22            is >> grid.gridData[i][j];
23
24            if (grid.gridData[i][j] == '@') {
25                grid.playerX = j;
26                grid.playerY = i;
27            }
28        }
29    }
30
31    // Load in all the textures and set initial player loc
32    // grid.fillTextures(); // now you can use them in draw()
33    grid.fillOriginalGrid();
34    // grid.setInitialPlayerLoc();
35    return is;
36 }
37
38
39 void Sokoban::setLevelName(std::string _levelName) {
40     levelName = _levelName;
41 }
42 // returns height due to how columns are read
43 int Sokoban::width(void) const {
44     return internalHeight;
45 }
46 // returns width due to how columns are read
47 int Sokoban::height(void) const {
48     return internalWidth;
49 }
50
51 sf::Vector2i Sokoban::playerLoc() const {
52     // returns player position (x,y)
53     return sf::Vector2i(playerX, playerY);
54 }
55
56 bool Sokoban::isWall(int x, int y) {
57     if ((getGridData(x, y)) == '#')
58         return true;
59     else
60         return false;

```

```

61 }
62
63
64 bool Sokoban::isBoxOrStoredBox(int x, int y) {
65     if ((getGridData(x, y) == 'A') || (getGridData(x, y) == '1'))
66         return true;
67     else
68         return false;
69 }
70
71 bool Sokoban::isOrigSpaceStorage(int x, int y) {
72     if (getGridOriginalData(x, y) == 'a')
73         return true;
74     else
75         return false;
76 }
77
78 void Sokoban::fillPreviousGrid() {
79     // Initialize and copy gridData to another vector
80     gridDataPrevious.resize(gridData.size());
81     for (size_t i = 0; i < gridData.size(); ++i) {
82         gridDataPrevious[i].resize(gridData[i].size());
83         for (size_t j = 0; j < gridData[i].size(); ++j) {
84             gridDataPrevious[i][j] = gridData[i][j];
85         }
86     }
87 }
88
89
90 void Sokoban::resetLevelBack() {
91     // Put the original grid data back to one previous step
92     for (int i = 0; i < internalWidth; ++i) {
93         for (int j = 0; j < internalHeight; ++j) {
94             gridData[i][j] = gridDataPrevious[i][j];
95
96             if (gridData[i][j] == '@') {
97                 playerX = j;
98                 playerY = i;
99             }
100         }
101     }
102 }
103
104
105 void Sokoban::movePlayer(enum Direction _direction) {
106     std::cout << "Player Moved: " << std::endl;
107
108     // This is where the player is currently at
109     sf::Vector2i currLoc = playerLoc();
110
111     // See if the move is possible
112     bool bMoveOk = true;
113     if ((_direction == Right) && (currLoc.x == internalHeight - 1))
114         bMoveOk = false;
115     else if ((_direction == Left) && (currLoc.x == 0))
116         bMoveOk = false;
117     else if ((_direction == Up) && (currLoc.y == 0))
118         bMoveOk = false;
119     else if ((_direction == Down) && (currLoc.y == internalWidth-1))

```

```

120         bMoveOk = false;
121     if (!bMoveOk) {
122         std::cout << "Player move not possible." << std::endl;
123         return;
124     }
125
126     // TO DO: make sure a move is possible ie a player can't go off the
board
127     int nextX = 0;
128     int nextY = 0;
129     int nextNextX = 0;
130     int nextNextY = 0;
131
132     // Before you move, store the current state so you can undo one
step with the Y key.
133     fillPreviousGrid();
134     // Make a move
135     switch (_direction) {
136     case Up:
137     case Down:
138
139         if (_direction == Up) {
140             nextY = currLoc.y - 1;
141             nextNextY = currLoc.y - 2;
142         } else {
143             nextY = currLoc.y + 1;
144             nextNextY = currLoc.y + 2;
145         }
146         // If the nextY is not a wall or nothing, then move
147         if (!isWall(currLoc.x, nextY)) {
148             // If the next spot up is a box or stored box
149             if (isBoxOrStoredBox(currLoc.x, nextY)) {
150                 // See if the next next spot is wall or nothing
151                 if (!isWall(currLoc.x, nextNextY)) {
152                     // adding check for if a box is after the box you
are pushing
153                     if (!isBoxOrStoredBox(currLoc.x, nextNextY)) {
154                         // Move the player
155                         setPlayerY(nextY);
156                         setGridData(currLoc.x, nextY, '@');
157
158                         // See if the box has moved into a storage spot
159                         if (getGridData(currLoc.x, nextNextY) == 'a') {
160                             // Move the box
161                             setGridData(currLoc.x, nextNextY, '1');
162
163                             // Replace the players old location with a
'.' or 'a'
164                             if (isOrigSpaceStorage(currLoc.x, currLoc.y)
)
165                                 setGridData(currLoc.x, currLoc.y, 'a');
166                             else
167                                 setGridData(currLoc.x, currLoc.y, '.');
168                         } else {
169                             if (!isWall(currLoc.x, nextNextY)) {
170                                 setGridData(currLoc.x, nextNextY, 'A');
171                                 setGridData(currLoc.x, currLoc.y, '.');
172                             }
173                         }

```



```

174         }
175     }
176     } else { // It's not a box, just move
177         // Set the new y location
178         setPlayerY(nextY);
179         setGridData(currLoc.x, nextY, '@');
180
181         // Replace the players old location with a .
182         if (isOrigSpaceStorage(currLoc.x, currLoc.y))
183             setGridData(currLoc.x, currLoc.y, 'a');
184         else
185             setGridData(currLoc.x, currLoc.y, '.');
186     }
187 }
188 break;
189
190 case Left:
191 case Right:
192
193     if (_direction == Left) {
194         nextX = currLoc.x - 1;
195         nextNextX = currLoc.x - 2;;
196     } else {
197         nextX = currLoc.x + 1;
198         nextNextX = currLoc.x + 2;
199     }
200     if (!isWall(nextX, currLoc.y)) {
201         // If the next spot left is a box or stored box
202         if (isBoxOrStoredBox(nextX, currLoc.y)) {
203             // See if the next next spot is wall
204             if (!isWall(nextNextX, currLoc.y)) {
205                 if (!isBoxOrStoredBox(nextNextX, currLoc.y)) {
206                     // Move the player
207                     setPlayerX(nextX);
208                     setGridData(nextX, currLoc.y, '@');
209
210                     // See if the box has moved into a storage spot
211                     if (getGridData(nextNextX, currLoc.y) == 'a') {
212                         // Move the box
213                         setGridData(nextNextX, currLoc.y, '1');
214
215                         // Replace the players old location with a
216                         // '.' or 'a'
217                         if (isOrigSpaceStorage(currLoc.x, currLoc.y))
218                             setGridData(currLoc.x, currLoc.y, 'a');
219                         else
220                             setGridData(currLoc.x, currLoc.y, '.');
221                     } else {
222                         if (!isWall(nextNextX, currLoc.y)) {
223                             setGridData(nextNextX, currLoc.y, 'A');
224
225                             // Replace the players old location with a
226                             // '.' or 'a'
227                             if (isOrigSpaceStorage(currLoc.x, currLoc.y))
228                                 setGridData(currLoc.x, currLoc.y, 'a');
229                             else
230                                 setGridData(currLoc.x, currLoc.y, '.');

```



```

229         }
230     }
231 }
232 }
233     } else { // It's not a box, just move
234         // Set the new y location
235         setPlayerX(nextX);
236         setGridData(nextX, currLoc.y, '@');
237
238         // Replace the players old location with a '.' or 'a'
239         if (isOrigSpaceStorage(currLoc.x, currLoc.y))
240             setGridData(currLoc.x, currLoc.y, 'a');
241         else
242             setGridData(currLoc.x, currLoc.y, '.');
243     }
244 }
245     break;
246
247
248     default:
249         break;
250 }
251     lastDirection = _direction;
252     std::cout << "Player is at: " << playerX << ", " << playerY << std::
endl;
253 }
254
255 // uses algorithm and lambda functions
256 bool Sokoban::isWon() const {
257     // When there are no more unstored boxes 'A', you won
258     bool bIsWon = true;
259     for (int i = 0; i < internalWidth; ++i) {
260         for (int j = 0; j < internalHeight; ++j) {
261             if (gridData[i][j] == 'a')
262                 bIsWon = false;
263         }
264     }
265     if (bIsWon) {
266         return true;
267     } else {
268         auto it = std::find_if(gridData.begin(), gridData.end(), [](const
auto& row) {
269             return std::find(row.begin(), row.end(), 'A') != row.end();
270         });
271         return it == gridData.end();
272     }
273 }
274
275
276 std::string Sokoban::getLevelName(void) {
277     return levelName;
278 }
279
280 void Sokoban::setPlayerX(int x) {
281     if (x >= 0)
282         playerX = x;
283 }
284
285 void Sokoban::setPlayerY(int y) {
286     if (y >= 0)

```

```

286     playerY = y;
287 }
288
289 void Sokoban::resetLevel() {
290     // Put the original grid data back into the grid data
291     // that will be drawn
292     for (int i = 0; i < internalWidth; ++i) {
293         for (int j = 0; j < internalHeight; ++j) {
294             gridData[i][j] = gridDataOriginal[i][j];
295
296             if (gridData[i][j] == '@') {
297                 playerX = j;
298                 playerY = i;
299             }
300         }
301     }
302 }
303
304 void Sokoban::fillOriginalGrid() {
305     // Initialize and copy gridData to another vector
306     gridDataOriginal.resize(gridData.size());
307     for (size_t i = 0; i < gridData.size(); ++i) {
308         gridDataOriginal[i].resize(gridData[i].size());
309         for (size_t j = 0; j < gridData[i].size(); ++j) {
310             gridDataOriginal[i][j] = gridData[i][j];
311         }
312     }
313 }
314
315 Sokoban::Sokoban() {
316     // make sure all textures are valid
317     texturePlayer05.loadFromFile("player_05.png");
318     texturePlayer08.loadFromFile("player_08.png");
319     texturePlayer17.loadFromFile("player_17.png");
320     texturePlayer20.loadFromFile("player_20.png");
321     textureBlock06.loadFromFile("block_06.png");
322     textureCrate03.loadFromFile("crate_03.png");
323     textureEnviroment03.loadFromFile("environment_03.png");
324     textureGround01.loadFromFile("ground_01.png");
325     textureGround04.loadFromFile("ground_04.png");
326 }
327
328 // moved to constructor instead
329 // void Sokoban::fillTextures() {
330 // }
331
332 char Sokoban::getGridData(int x, int y) {
333     return gridData[y][x];
334 }
335
336 char Sokoban::getGridOriginalData(int x, int y) {
337     return gridDataOriginal[y][x];
338 }
339
340 void Sokoban::setGridData(int x, int y, char c) {
341     gridData[y][x] = c;
342 }
343
344 void Sokoban::setInitialPlayerLoc() {

```

```

345     char cCurr;
346     for (int i = 0; i < internalWidth; ++i) {
347         for (int j = 0; j < internalHeight; ++j) {
348             cCurr = gridData[i][j];
349             if (cCurr == '@') {
350                 playerX = j;
351                 playerY = i;
352             }
353         }
354     }
355 }
356
357 void Sokoban::draw(sf::RenderTarget& target, sf::RenderStates states) const
358 {
359     sf::Sprite tPlayer;
360     sf::Sprite tCrate;
361     char cCurr;
362
363     for (int i = 0; i < internalWidth; ++i) {
364         for (int j = 0; j < internalHeight; ++j) {
365             cCurr = gridData[i][j];
366             sf::Sprite sprite;
367             switch (cCurr) {
368                 case '#':
369                     sprite.setTexture(textureBlock06);
370                     sprite.setPosition(j * 64, i*64);
371                     target.draw(sprite, states);
372                     break;
373                 case '.':
374                     sprite.setTexture(textureGround01);
375                     sprite.setPosition(j * 64, i*64);
376                     target.draw(sprite, states);
377                     break;
378                 case '@':
379                     sprite.setTexture(textureGround01);
380                     sprite.setPosition(j * 64, i * 64);
381                     target.draw(sprite, states);
382                     switch (lastDirection) {
383                         case Up:
384                             tPlayer.setTexture(texturePlayer08);
385                             break;
386                         case Right:
387                             tPlayer.setTexture(texturePlayer17);
388                             break;
389                         case Left:
390                             tPlayer.setTexture(texturePlayer20);
391                             break;
392                         default: // Down
393                             tPlayer.setTexture(texturePlayer05);
394                     }
395                     tPlayer.setPosition(j * 64, i * 64);
396                     target.draw(tPlayer, states);
397                     break;
398                 case 'A':
399                     sprite.setTexture(textureGround01);
400                     sprite.setPosition(j * 64, i*64);
401                     target.draw(sprite, states);
402
403                     tCrate.setTexture(textureCrate03);

```

```

403         tCrate.setPosition(j * 64, i*64);
404         target.draw(tCrate, states);
405         break;
406     case 'a':
407         sprite.setTexture(textureGround04);
408         sprite.setPosition(j * 64, i*64);
409         target.draw(sprite, states);
410         break;
411     case '1':
412         sprite.setTexture(textureGround01);
413         sprite.setPosition(j * 64, i*64);
414         target.draw(sprite, states);
415
416         tCrate.setTexture(textureCrate03);
417         tCrate.setPosition(j * 64, i*64);
418         target.draw(tCrate, states);
419     }
420 }
421 }
422 }
423 } // namespace SB

```

```

1  // Copyright 2024 Camden Andersson
2  #include <iostream>
3  #include <fstream>
4  #include "Sokoban.hpp"
5
6  #define BOOST_TEST_DYN_LINK
7  #define BOOST_TEST_MODULE Main
8  #include <boost/test/unit_test.hpp>
9
10 using SB::Sokoban;
11
12 BOOST_AUTO_TEST_CASE(testInstantWin) {
13     std::ifstream file;
14     file.open("level1.lvl");
15     SB::Sokoban sb;
16     file >> sb;
17     BOOST_REQUIRE_EQUAL(false, sb.isWon());
18 }
19
20 BOOST_AUTO_TEST_CASE(testCantMove) {
21     std::ifstream file;
22     file.open("level1.lvl");
23     SB::Sokoban sb;
24     file >> sb;
25     sb.movePlayer(SB::Direction::Right);
26     int startX = sb.playerLoc().x;
27     int startY = sb.playerLoc().y;
28     sb.movePlayer(SB::Direction::Up);
29     int endX = sb.playerLoc().x;
30     int endY = sb.playerLoc().y;
31     BOOST_REQUIRE_EQUAL(startX, endX);
32     BOOST_REQUIRE_EQUAL(startY, endY);
33 }
34
35 BOOST_AUTO_TEST_CASE(testBoxWall) {
36     std::ifstream file;
37     file.open("level3.lvl");
38     SB::Sokoban sb;

```

```

39  file >> sb;
40  sb.movePlayer(SB::Direction::Right);
41  sb.movePlayer(SB::Direction::Right);
42  sb.movePlayer(SB::Direction::Right);
43  sb.movePlayer(SB::Direction::Right);
44  int startX = sb.playerLoc().x;
45  int startY = sb.playerLoc().y;
46  sb.movePlayer(SB::Direction::Right);
47  int endX = sb.playerLoc().x;
48  int endY = sb.playerLoc().y;
49  BOOST_REQUIRE_EQUAL(startX, endX);
50  BOOST_REQUIRE_EQUAL(startY, endY);
51 }
52
53 BOOST_AUTO_TEST_CASE(testBoxBox) {
54     std::ifstream file;
55     file.open("level2.lvl");
56     SB::Sokoban sb;
57     file >> sb;
58     int startX = sb.playerLoc().x;
59     int startY = sb.playerLoc().y;
60     sb.movePlayer(SB::Direction::Up);
61     int endX = sb.playerLoc().x;
62     int endY = sb.playerLoc().y;
63     BOOST_REQUIRE_EQUAL(startX, endX);
64     BOOST_REQUIRE_EQUAL(startY, endY);
65 }
66
67 BOOST_AUTO_TEST_CASE(testMoveOffScreen) {
68     std::ifstream file;
69     file.open("level4.lvl");
70     SB::Sokoban sb;
71     file >> sb;
72     sb.movePlayer(SB::Direction::Down);
73     sb.movePlayer(SB::Direction::Down);
74     sb.movePlayer(SB::Direction::Down);
75     sb.movePlayer(SB::Direction::Right);
76     sb.movePlayer(SB::Direction::Right);
77     sb.movePlayer(SB::Direction::Right);
78     sb.movePlayer(SB::Direction::Right);
79     sb.movePlayer(SB::Direction::Right);
80     sb.movePlayer(SB::Direction::Right);
81     sb.movePlayer(SB::Direction::Right);
82     int startX = sb.playerLoc().x;
83     int startY = sb.playerLoc().y;
84     sb.movePlayer(SB::Direction::Right);
85     int endX = sb.playerLoc().x;
86     int endY = sb.playerLoc().y;
87     BOOST_REQUIRE_EQUAL(startX, endX);
88     BOOST_REQUIRE_EQUAL(startY, endY);
89 }
90
91 BOOST_AUTO_TEST_CASE(testMoreBoxes) {
92     std::ifstream file;
93     file.open("level5.lvl");
94     SB::Sokoban sb;
95     file >> sb;
96     sb.movePlayer(SB::Direction::Up);
97     sb.movePlayer(SB::Direction::Up);

```

```

98     sb.movePlayer(SB::Direction::Up);
99     sb.movePlayer(SB::Direction::Up);
100    sb.movePlayer(SB::Direction::Right);
101    sb.movePlayer(SB::Direction::Right);
102    sb.movePlayer(SB::Direction::Right);
103    sb.movePlayer(SB::Direction::Right);
104    sb.movePlayer(SB::Direction::Down);
105    sb.movePlayer(SB::Direction::Right);
106    sb.movePlayer(SB::Direction::Up);
107    BOOST_REQUIRE_EQUAL(true, sb.isWon());
108 }
109
110 BOOST_AUTO_TEST_CASE(testMoreStorage) {
111     std::ifstream file;
112     file.open("level6.lvl");
113     SB::Sokoban sb;
114     file >> sb;
115     sb.movePlayer(SB::Direction::Right);
116     sb.movePlayer(SB::Direction::Right);
117     sb.movePlayer(SB::Direction::Right);
118     sb.movePlayer(SB::Direction::Up);
119     sb.movePlayer(SB::Direction::Right);
120     sb.movePlayer(SB::Direction::Up);
121     sb.movePlayer(SB::Direction::Right);
122     sb.movePlayer(SB::Direction::Down);
123     sb.movePlayer(SB::Direction::Up);
124     sb.movePlayer(SB::Direction::Up);
125     sb.movePlayer(SB::Direction::Up);
126     sb.movePlayer(SB::Direction::Left);
127     sb.movePlayer(SB::Direction::Left);
128     sb.movePlayer(SB::Direction::Left);
129     sb.movePlayer(SB::Direction::Left);
130     sb.movePlayer(SB::Direction::Down);
131     sb.movePlayer(SB::Direction::Left);
132     sb.movePlayer(SB::Direction::Up);
133     BOOST_REQUIRE_EQUAL(true, sb.isWon());
134 }

```


5 PS4: NBody

5.1 What I accomplished

For PS4: N-Body Simulation, the assignment involved developing a physics-based simulation to model the gravitational interactions among celestial bodies within a universe. This task extended the `CelestialBody` class to handle dynamic physics calculations, including updating velocities based on gravitational forces. A crucial component was the implementation of a `step` method in the `Universe` class, which progresses the simulation based on time increments and updates the positions and velocities of all celestial bodies accordingly.

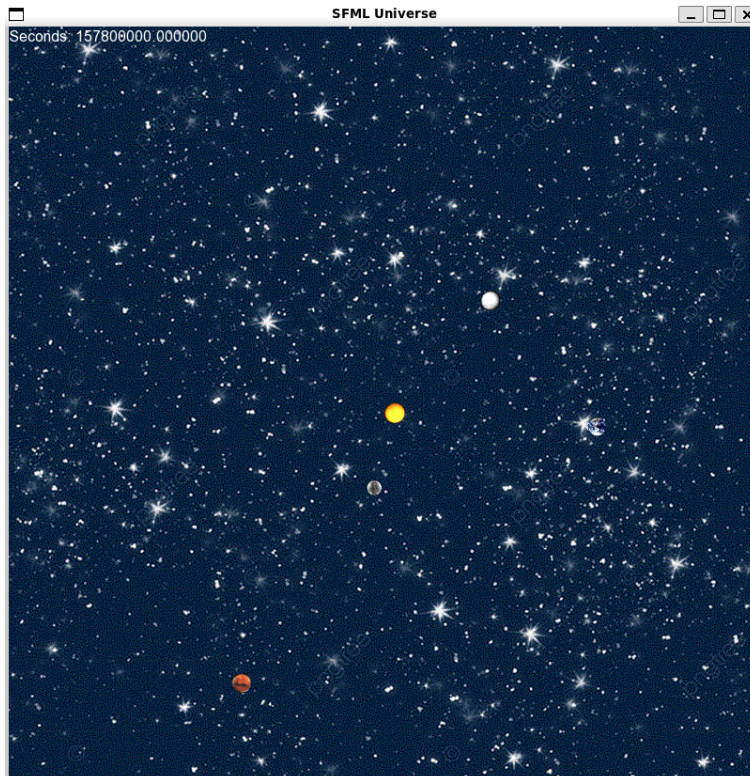


Figure 6: Window produced from running the program

My solution for PS4b was centered on accurately simulating the orbital mechanics of celestial bodies using Newton's laws of motion and gravitation. I extended the `CelestialBody` class to include methods for calculating gravitational forces exerted on each body, which involved computing pairwise forces and their components in both x and y directions. These calculations were based on the mass of the bodies and their distances from each other, employing the gravitational constant to determine the magnitude of the force.

In the `Universe` class, I implemented the `step` method to advance the simulation for a specified time step. This method calculated the net forces on each body, determined their accelerations, and updated their velocities and positions accordingly. The simulation employed a leapfrog integration scheme to ensure numerical stability and accuracy over time.

To manage memory efficiently and ensure robustness, I used smart pointers for handling the lifetimes of `CelestialBody` objects. This approach prevented memory leaks and ensured that objects were properly cleaned up, which was critical given the dynamic nature of the simulation where bodies could be added or removed based on various simulation conditions.

For extra credit, I designed my own simulation scenario, which featured a celestial body in an unusual orbit around the sun, demonstrating the flexibility and accuracy of the physics engine in handling non-standard orbital dynamics. Additionally, I integrated a time counter into the simulation, which displayed the elapsed time in seconds within the universe, enhancing the interactivity and user engagement by providing a real-time update of the simulation's progress.

One of the technical challenges was ensuring the accuracy of the simulation, particularly in maintaining the precision of calculations over long periods. This required careful implementation of the physics equations and attention to the cumulative errors that can occur in numerical simulations. The project not only deepened my understanding of astrophysical simulations but also enhanced my skills in managing complex object-oriented programming structures and memory management in a high-performance computing context.

5.2 What I already knew

Before starting on PS4, I already possessed a solid foundation in kinematics, thanks to my previous physics courses. This background was instrumental in formulating the simulation's dynamics, as it allowed me to effectively apply principles of motion and force to the celestial bodies in the virtual universe.

5.3 What I learned

During this project, I significantly deepened my understanding of how physics calculations integrate with graphical representations, building on the concepts explored in previous projects like PS3. This project highlighted the crucial link between the backend physics calculations and the frontend graphical display, which is fundamental in real-world applications such as games or simulation software. It provided me with a practical context for applying theoretical knowledge in a way that visually demonstrates complex concepts.

5.4 Challenges

The most significant challenge in this project was implementing the `step()` function, which involved coordinating the physics calculations with data input from various files to update the state of the universe accurately. While the integration of these elements was complex, the project was somewhat more straightforward compared to previous ones, such as PS3, because it involved fewer conditional logic blocks. The primary difficulty lay in ensuring that the physics behaved as expected across different scenarios and inputs. Additionally, I did not pass the `testLeapFrog` in the auto grader, as the decimal was not precise enough. I'm not entirely sure why that is, as I used doubles for all my internal calculations as requested, so it shouldn't have been an issue.

5.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = Universe.hpp CelestialBody.hpp
6 # Your compiled .o files
7 OBJECTS = Universe.o CelestialBody.o
8 # The name of your program
9 PROGRAM = NBody
10
11 .PHONY: all clean lint
12
13 all: $(PROGRAM) test NBody.a
14
15 # Wildcard recipe to make .o files from corresponding .cpp file
16 %.o: %.cpp $(DEPS)
17     $(CC) $(CFLAGS) -c $<
18
19 $(PROGRAM): main.o $(OBJECTS)
20     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
21
22 NBody.a: Universe.o CelestialBody.o
23     ar rcs NBody.a Universe.o CelestialBody.o
24
25 test: test.o Universe.o CelestialBody.o
26     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
27
28 test.o: test.cpp Universe.hpp CelestialBody.hpp
29     $(CC) $(CFLAGS) -c $< -o $@
```



```

30
31 clean:
32     rm *.o $(PROGRAM) *.a test
33
34 lint:
35     cpplint *.cpp *.hpp

```

```

1  // Copyright 2024 Camden Andersson
2
3  #include <iostream>
4  #include <SFML/Graphics.hpp>
5  #include <SFML/Window.hpp>
6  #include <SFML/Audio/Sound.hpp>
7  #include <SFML/Audio/SoundBuffer.hpp>
8
9  #include "CelestialBody.hpp"
10 #include "Universe.hpp"
11
12
13 int main(int argc, char* argv[]) {
14     // Load the file
15     // std::ifstream file("kevin.txt");
16     // if (!file.is_open()) {
17     //     std::cerr << "Failed to open file" << std::endl;
18     //     return 1;
19     // }
20
21     // Load an instance of the Universe class
22     NB::Universe universe;
23     std::cin >> universe;
24     // std::cout << universe;
25     sf::Font textCounterFont;
26     if (!textCounterFont.loadFromFile("arial.ttf")) {
27         std::cout << "No arial.ttf file found." << std::endl;
28     }
29     sf::Text textForCounter;
30     textForCounter.setFont(textCounterFont);
31     textForCounter.setCharacterSize(16);
32     sf::Clock clock;
33
34
35
36     // Set the screen size and load the Universe->CelestialBody textures
37     int screenWidth = 800;
38     int screenHeight = 800;
39     universe.setScreenDimensions(screenWidth, screenHeight);
40     universe.setMaxDimension();
41     universe.loadTextures();
42
43     // Setup the times
44     double deltaT = std::stod(argv[2]);
45     double T = std::stod(argv[1]);
46     // double deltaT = 25000.00;
47     // double T = 157788000.0;
48
49     // NOTE: 1 day = 86400 sec
50     // 1 year = 3153600 sec
51     // double deltaT = 86400;
52     // double T = 3153600
53

```

```

54 // Initialize the current time
55 double dCurrT = 0.0 + deltaT;
56 std::string seconds = "Seconds: ";
57
58 sf::SoundBuffer buffer;
59 sf::Sound sound;
60 buffer.loadFromFile("backgroundmusic.wav");
61 sound.setBuffer(buffer);
62 sound.play();
63 sound.setLoop(true);
64
65 // Create the window itself
66 sf::RenderWindow window(sf::VideoMode(screenWidth, screenHeight), "SFML
Universe");
67 while (window.isOpen()) {
68     sf::Event event;
69     while (window.pollEvent(event)) {
70         if (event.type == sf::Event::Closed) {
71             std::cout << universe;
72             sound.stop();
73             window.close();
74         }
75     }
76     if (dCurrT < T) {
77         // Do the physics to compute the forces and positions
78         universe.step(deltaT);
79         dCurrT = dCurrT + deltaT;
80         // Display the results
81         // temporary disabling for submission std::cout << "time: " <<
dCurrT << std::endl;
82     }
83     textForCounter.setString(seconds + std::to_string(dCurrT));
84     window.clear();
85     window.draw(universe);
86     window.draw(textForCounter);
87     window.display();
88 }
89
90 return 0;
91 }

```

```

1 // Copyright 2024 Camden Andersson
2 #pragma once
3
4 #include "CelestialBody.hpp"
5 namespace NB {
6 class Universe : public sf::Drawable {
7     private:
8         std::vector<std::shared_ptr<CelestialBody>> bodies;
9         int numParticles;
10        double _radius;
11        sf::Texture backgroundTexture;
12        int screenWidth;
13        int screenHeight;
14    protected:
15        virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
const; // override;
16    public:
17        Universe();
18        explicit Universe(const std::string filename);

```

```

19     void loadTextures();
20     void setScreenDimensions(int width, int height);
21     void setMaxDimension();
22     void step(double deltaT);
23     friend std::istream& operator>>(std::istream& is, Universe& universe);
24     friend std::ostream& operator<<(std::ostream& os, const Universe&
universe);
25     CelestialBody& operator[](size_t index);
26     double radius() const;
27     int numPlanets() const;
28 };
29 } // namespace NB

```

```

1 // Copyright 2024 Camden Andersson
2
3 #include "Universe.hpp"
4 namespace NB {
5 // Base Constructor
6 Universe::Universe() {
7     numParticles = 0;
8     _radius = 0.0;
9 }
10 // is this
11 Universe::Universe(const std::string filename) {
12     std::ifstream file(filename);
13     // same as the input operator
14     file >> numParticles >> _radius;
15     // bodies.clear(); // Clear any existing bodies
16     // bodies.reserve(numParticles); // Reserve space for the bodies
17     // bodies.resize(numParticles);
18     for (int i = 0; i < numParticles; ++i) {
19         auto body = std::make_shared<CelestialBody>();
20         file >> *body; // Use the CelestialBody operator to read the body
data
21         bodies.push_back(body);
22     }
23 }
24
25 void Universe::setScreenDimensions(int width, int height) {
26     screenWidth = width;
27     screenHeight = height;
28
29     // Now set that dimension in all the CelestialBodies
30     for (size_t i = 0; i < bodies.size(); i++) {
31         bodies[i]->setScreenDimensions(width, height);
32     }
33 }
34
35 void Universe::setMaxDimension() {
36     double dMax = _radius;
37     // Now set that dimension
38     for (size_t i = 0; i < bodies.size(); i++) {
39         bodies[i]->setMaxDimension(dMax);
40     }
41 }
42
43 void Universe::loadTextures() {
44     // EC: load the background texture
45     if (!backgroundTexture.loadFromFile("background.gif")) {
46         // Error handling if the file fails to load

```

```

47     std::cerr << "Failed to load texture: background.gif " << std::endl;
48 }
49 backgroundTexture.setSmooth(true); // Enable smooth resizing
50 // Load the CelestialBody textures
51 for (size_t i = 0; i < bodies.size(); i++) {
52     bodies[i]->loadTextures();
53 }
54 }
55
56 void Universe::draw(sf::RenderTarget& target, sf::RenderStates states) const
57 {
58     // Resize the texture and draw the background
59     sf::Sprite sprite(backgroundTexture);
60     sprite.setScale(screenWidth / backgroundTexture.getSize().x,
61                     screenWidth / backgroundTexture.getSize().y);
62     sprite.setPosition(0.0f, 0.0f);
63     target.draw(sprite, states);
64     // Iterate through all the Celestial Bodies and draw
65     for (const auto& body : bodies) {
66         body->draw(target, states);
67     }
68 }
69
70 std::istream& operator>>(std::istream& is, Universe& universe) {
71     is >> universe.numParticles >> universe._radius;
72     universe.bodies.clear(); // Clear any existing bodies
73     universe.bodies.reserve(universe.numParticles); // Reserve space for
74     // the bodies
75     for (int i = 0; i < universe.numParticles; ++i) {
76         auto body = std::make_shared<CelestialBody>();
77         is >> *body; // Use the CelestialBody operator to read the body
78         // data
79         universe.bodies.push_back(body);
80     }
81     return is;
82 }
83
84 std::ostream& operator<<(std::ostream& os, const Universe& universe) {
85     os << universe.numParticles << std::endl; // removed all old formatting
86     os << universe._radius << std::endl;
87     for (const auto& body : universe.bodies) {
88         os << *body;
89     }
90     return os;
91 }
92
93 double Universe::radius() const {
94     return _radius;
95 }
96
97 int Universe::numPlanets() const {
98     return bodies.size();
99 }
100
101 NB::CelestialBody& Universe::operator[](size_t index) {
102     if (index >= bodies.size()) {
103         // Error handling if index is out of bounds
104         std::cerr << "Failed to load Celestial body at " << index << "." <<
105         std::endl;

```

```

102     }
103     return *bodies[index];
104 }
105
106
107 void Universe::step(double deltaT) {
108     double G = 6.67e-11;
109     double deltaX, deltaY, r, F, Fx, Fy, ax, ay, vx, vy, px, py;
110     // Go body by body
111     for (size_t i = 0; i < bodies.size(); i++) {
112         // For every body (except itself), use superposition to compute F
113         Fx = 0.0;
114         Fy = 0.0;
115         // For each body, compute the Force btwn that and the other spk
116         // Of course, exclude the body you are on
117         for (size_t j = 0; j < bodies.size(); j++) {
118             if (j != i) {
119                 deltaX = bodies[j]->position().x - bodies[i]->position()
120                 .x;
121                 deltaY = bodies[j]->position().y - bodies[i]->position()
122                 .y;
123                 r = pow(deltaX * deltaX + deltaY * deltaY, 0.5);
124                 F = (G * bodies[i]->mass() * bodies[j]->mass()) / (r * r
125                 );
126
127                 // Sum up the components with the previous force
128                 Fx = Fx + (F * deltaX / r);
129                 Fy = Fy + (F * deltaY / r);
130             }
131         }
132
133         // Now that you have the total force...
134         ax = Fx / bodies[i]->mass();
135         ay = Fy / bodies[i]->mass();
136
137         vx = bodies[i]->getVelocity().x + (deltaT * ax);
138         vy = bodies[i]->getVelocity().y + (deltaT * ay);
139         sf::Vector2f newV = sf::Vector2f(static_cast<float>(vx),
140         static_cast<float>(vy));
141         bodies[i]->setVelocity(newV); // store the new velocity
142
143         px = bodies[i]->position().x + (deltaT * vx);
144         py = bodies[i]->position().y + (deltaT * vy);
145         sf::Vector2f newPos = sf::Vector2f(static_cast<float>(px),
146         static_cast<float>(py));
147         bodies[i]->setPosition(newPos); // store the new position
148     }
149 }
150 // namespace NB

```

```

1 // Copyright 2024 Camden Andersson
2 #pragma once
3
4 #include <iostream>
5 #include <fstream>
6 #include <vector>
7 #include <cmath>
8 #include <memory>
9 #include <SFML/Graphics.hpp>

```

```

10
11 namespace NB {
12 class CelestialBody : public sf::Drawable {
13     private:
14         double xPos;
15         double yPos;
16         double xVelocity;
17         double yVelocity;
18         double _mass;
19         std::string filename;
20         sf::Texture texture;
21         int screenWidth;
22         int screenHeight;
23         double maxDimension;
24
25     public:
26         CelestialBody() {}
27         CelestialBody(double x, double y, double vx, double vy, double m, const
std::string& file);
28         void setPosition(sf::Vector2f pos);
29         sf::Vector2f position() const;
30         sf::Vector2f velocity() const;
31         double mass() const;
32         void loadTextures();
33         void setScreenDimensions(int width, int height);
34         void setMaxDimension(double _maxDimension);
35         void draw(sf::RenderTarget& target, sf::RenderStates states) const
override;
36         friend std::istream& operator>>(std::istream& is, CelestialBody& body);
37         friend std::istream& operator>>(std::istream& is, std::shared_ptr<
CelestialBody>& body);
38         friend std::ostream& operator<<(std::ostream& os, const CelestialBody&
body);
39         void setVelocity(sf::Vector2f vel);
40         sf::Vector2f getVelocity();
41 };
42 } // namespace NB

```

```

1 // Copyright 2024 Camden Andersson
2 #include "CelestialBody.hpp"
3 namespace NB {
4
5 CelestialBody::CelestialBody(double x, double y, double vx,
6     double vy, double m, const std::string& file) :
7     xPos(x), yPos(y), xVelocity(vx), yVelocity(vy), _mass(m), filename(file)
8     {
9         if (!texture.loadFromFile(filename)) {
10             std::cerr << "Failed to load texture: " << filename << std::endl;
11         }
12     }
13
14 void CelestialBody::setPosition(sf::Vector2f pos) {
15     xPos = pos.x;
16     yPos = pos.y;
17 }
18
19 void CelestialBody::setVelocity(sf::Vector2f vel) {
20     xVelocity = vel.x;
21     yVelocity = vel.y;
22 }

```

```

22
23 sf::Vector2f CelestialBody::getVelocity() {
24     return sf::Vector2f(static_cast<float>(xVelocity), static_cast<float>(
25         yVelocity));
26 }
27 sf::Vector2f CelestialBody::position() const {
28     return sf::Vector2f(static_cast<float>(xPos), static_cast<float>(yPos));
29 }
30
31 sf::Vector2f CelestialBody::velocity() const {
32     return sf::Vector2f(static_cast<float>(xVelocity), static_cast<float>(
33         yVelocity));
34 }
35 double CelestialBody::mass() const {
36     return _mass;
37 }
38
39 void CelestialBody::setScreenDimensions(int width, int height) {
40     screenWidth = width;
41     screenHeight = height;
42 }
43
44 void CelestialBody::setMaxDimension(double _maxDimension) {
45     maxDimension = 2.2*_maxDimension;
46     // make it a little bigger than 2x the largest distance
47 }
48
49 void CelestialBody::loadTextures() {
50     if (!texture.loadFromFile(filename)) {
51         std::cerr << "Failed to load texture: " << filename << std::endl;
52     }
53 }
54
55 void CelestialBody::draw(sf::RenderTarget& target, sf::RenderStates states)
56     const {
57     sf::Sprite sprite;
58     sprite.setTexture(texture);
59     // Get drawing coordinates
60     double drawX = xPos * (screenWidth / maxDimension) + (screenWidth / 2.0)
61     ;
62     double drawY = -1 * yPos * (screenWidth / maxDimension) + (screenHeight
63     / 2.0);
64     sprite.setPosition(static_cast<float>(drawX), static_cast<float>(drawY))
65     ;
66     target.draw(sprite, states);
67 }
68
69
70 std::istream& operator>>(std::istream& is, CelestialBody& body) {
71     return is >> body.xPos >> body.yPos >> body.xVelocity >>
72     body.yVelocity >> body._mass >> body.filename;
73 }
74
75 std::ostream& operator<<(std::ostream& os, const CelestialBody& body) {
76     // removed all old formatting
77     os << body.xPos << " " << body.yPos << " "
78     << body.xVelocity << " " << body.yVelocity << " "
79     << body._mass << " "

```



```

75     << body.filename << std::endl;
76     return os;
77 }
78 } // namespace NB

```

```

1 // Copyright 2024 Camden Andersson
2 #include <iostream>
3 #include <fstream>
4 #include "Universe.hpp"
5
6 #define BOOST_TEST_DYN_LINK
7 #define BOOST_TEST_MODULE Main
8 #include <boost/test/unit_test.hpp>
9
10 using NB::Universe;
11
12 BOOST_AUTO_TEST_CASE(testNumPlanets) {
13     NB::Universe un("planets.txt");
14     BOOST_REQUIRE_EQUAL(un.numPlanets(), 5);
15 }
16
17 BOOST_AUTO_TEST_CASE(testRadius) {
18     NB::Universe un("planets.txt");
19     BOOST_REQUIRE_EQUAL(un.radius(), 2.50e+11);
20 }
21
22 BOOST_AUTO_TEST_CASE(testBracket0) {
23     NB::Universe un("planets.txt");
24     BOOST_REQUIRE_NO_THROW(un[0]);
25 }
26
27 BOOST_AUTO_TEST_CASE(testBracketEnd) {
28     NB::Universe un("planets.txt");
29     BOOST_REQUIRE_NO_THROW(un[4]);
30 }
31
32 BOOST_AUTO_TEST_CASE(testFormat) {
33     NB::Universe un("planets.txt");
34     std::stringstream ss;
35     ss << un;
36     std::string s = "5\n";
37     s = s + "2.5e+11\n";
38     s = s + "1.496e+11 0 0 29800 5.974e+24 earth.gif\n";
39     s = s + "2.279e+11 0 0 24100 6.419e+23 mars.gif\n";
40     s = s + "5.79e+10 0 0 47900 3.302e+23 mercury.gif\n";
41     s = s + "0 0 0 1.989e+30 sun.gif\n";
42     s = s + "1.082e+11 0 0 35000 4.869e+24 venus.gif\n";
43     BOOST_REQUIRE_EQUAL(ss.str(), s);
44 }
45
46 BOOST_AUTO_TEST_CASE(testHardcoded) {
47     NB::Universe un("planets.txt");
48     // std::cout << un;
49     std::ifstream ifs("test.txt");
50     // declare your celestialBody objects for comparison.
51     NB::CelestialBody a;
52     NB::CelestialBody b;
53     NB::CelestialBody c;
54     NB::CelestialBody d;
55     NB::CelestialBody e;

```



```

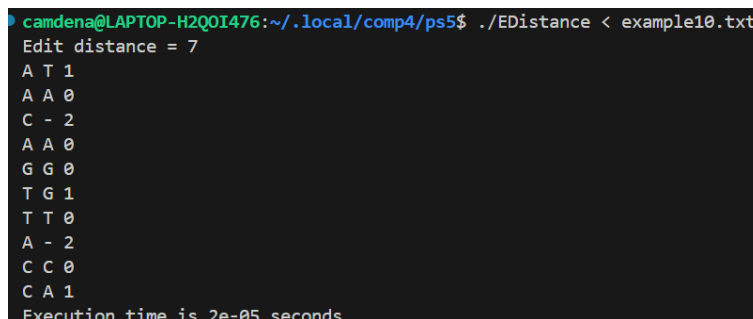
56 // load in from sstream
57 ifs >> a;
58 ifs >> b;
59 ifs >> c;
60 ifs >> d;
61 ifs >> e;
62 // test the input for each celestialBody object.
63 BOOST_REQUIRE_EQUAL(a.mass(), un[0].mass());
64 BOOST_REQUIRE_EQUAL(a.position().x, un[0].position().x);
65 BOOST_REQUIRE_EQUAL(a.position().y, un[0].position().y);
66 BOOST_REQUIRE_EQUAL(a.velocity().x, un[0].velocity().x);
67 BOOST_REQUIRE_EQUAL(a.velocity().y, un[0].velocity().y);
68 BOOST_REQUIRE_EQUAL(b.mass(), un[1].mass());
69 BOOST_REQUIRE_EQUAL(b.position().x, un[1].position().x);
70 BOOST_REQUIRE_EQUAL(b.position().y, un[1].position().y);
71 BOOST_REQUIRE_EQUAL(b.velocity().x, un[1].velocity().x);
72 BOOST_REQUIRE_EQUAL(b.velocity().y, un[1].velocity().y);
73 BOOST_REQUIRE_EQUAL(c.mass(), un[2].mass());
74 BOOST_REQUIRE_EQUAL(c.position().x, un[2].position().x);
75 BOOST_REQUIRE_EQUAL(c.position().y, un[2].position().y);
76 BOOST_REQUIRE_EQUAL(c.velocity().x, un[2].velocity().x);
77 BOOST_REQUIRE_EQUAL(c.velocity().y, un[2].velocity().y);
78 BOOST_REQUIRE_EQUAL(d.mass(), un[3].mass());
79 BOOST_REQUIRE_EQUAL(d.position().x, un[3].position().x);
80 BOOST_REQUIRE_EQUAL(d.position().y, un[3].position().y);
81 BOOST_REQUIRE_EQUAL(d.velocity().x, un[3].velocity().x);
82 BOOST_REQUIRE_EQUAL(d.velocity().y, un[3].velocity().y);
83 BOOST_REQUIRE_EQUAL(e.mass(), un[4].mass());
84 BOOST_REQUIRE_EQUAL(e.position().x, un[4].position().x);
85 BOOST_REQUIRE_EQUAL(e.position().y, un[4].position().y);
86 BOOST_REQUIRE_EQUAL(e.velocity().x, un[4].velocity().x);
87 BOOST_REQUIRE_EQUAL(e.velocity().y, un[4].velocity().y);
88 }
89
90 BOOST_AUTO_TEST_CASE(step25000) {
91     Universe un("planets.txt");
92     un.step(25000);
93     un.step(25000);
94     un.step(25000);
95     double d = un[0].position().x;
96     BOOST_REQUIRE_EQUAL(d, std::stod("149577777152"));
97 }

```

6 PS5: DNA Sequence Alignment

6.1 What I accomplished

In PS5: DNA Sequence Alignment, the task was to develop a program that computes the optimal sequence alignment of two DNA strings, using a method known as dynamic programming. This approach is critical in computational biology for determining the functional similarities between genes by analyzing their sequence alignment and calculating their edit distance, which accounts for mutations like insertions, deletions, and substitutions.



```
camdena@LAPTOP-H2QOI476:~/local/comp4/ps5$ ./EDistance < example10.txt
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 2e-05 seconds
```

Figure 7: Window produced from running the program

My solution involved creating a robust algorithm to align two genetic sequences optimally by minimizing a cost function based on edit distances. This function penalizes gaps and mismatches between sequences, which models the likelihood of evolutionary mutations. I implemented this using a dynamic programming approach, where I constructed a matrix to store the minimal edit distances for substrings of the two DNA sequences, thus avoiding redundant recalculations that a naive recursive approach would entail.

For extra credit, I created an alternate `min3()` function that was slightly more efficient than the `std` version of the same function, and this was demonstrated in the seconds it took to execute the program.

The main idea in this project was managing the complexity of dynamic programming in the context of biological data, which required careful consideration of the biological significance of each alignment decision. The matrix-based implementation needed to handle various scenarios efficiently, such as aligning long sequences with many mutations, which tested the limits of both space and time complexity of my approach. This project not only deepened my understanding of computational biology and dynamic programming but also honed my skills in designing efficient algorithms for complex data-driven problems.

From the project documentation and my README file, it's clear that the dynamic programming approach was essential for handling the computational demands of DNA sequence alignment. The matrix used in my solution efficiently calculated the optimal edit distances, and the results were thoroughly documented in the README file, demonstrating successful alignment under various test cases. This included handling large datasets like the *ecoli* sequences, where my implementation was able to process substantial amounts of data efficiently.

6.2 What I already knew

Before starting PS5: DNA Sequence Alignment, I was already familiar with using matrices, having utilized them extensively in previous projects. This prior experience was beneficial as it provided me with a strong foundation in handling matrix operations, which are crucial for implementing dynamic programming solutions in computational biology, especially with the Needleman-Wunsch method.

6.3 What I learned

During this project, I learned to implement the Needleman-Wunsch method. Applying this algorithm in code proved to be a challenging yet enlightening experience, as it involved translating theoretical concepts into practical programming constructs. This process deepened my understanding of how dynamic programming can be applied to solve complex problems in genomics.

6.4 Challenges

The primary challenge I encountered was managing the large matrices generated by the Needleman-Wunsch algorithm, as well as understanding the algorithm itself at first. The complexity of the algorithm itself compounded these difficulties, as any errors in the initial implementation magnified the challenges during testing. Navigating these large data structures required meticulous attention to detail and rigorous debugging to ensure the accuracy and efficiency of the sequence alignment.

6.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = EDistance.hpp
6 # Your compiled .o files
7 OBJECTS = EDistance.o
8 # The name of your program
9 PROGRAM = EDistance
10
11 .PHONY: all clean lint
12
13 all: $(PROGRAM) test EDistance.a
14
15 # Wildcard recipe to make .o files from corresponding .cpp file
16 %.o: %.cpp $(DEPS)
17     $(CC) $(CFLAGS) -c $<
18
19 $(PROGRAM): main.o $(OBJECTS)
20     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
21
22 EDistance.a: EDistance.o
23     ar rcs EDistance.a EDistance.o
24
25 test: test.o EDistance.o
26     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
27
28 test.o: test.cpp EDistance.hpp
29     $(CC) $(CFLAGS) -c $< -o $@
30
31 clean:
32     rm *.o $(PROGRAM) *.a test
33
34 lint:
35     cpplint *.cpp *.hpp
```

```
1 // Copyright Camden Andersson 2024
2 #include "EDistance.hpp"
3 int main(int argc, char* argv[]) {
4     // Load in the text file with the 2 strings
5     std::string x, y;
6     std::cin >> x >> y;
7     // std::ifstream inputFile(fileName);
8     // if (inputFile.is_open()) {
9     //     inputFile >> x >> y;
10    //     inputFile.close();
11    // } else {
12    //     std::cerr << "Unable to open file";
```

```

13     //     return 1;
14     // }
15
16     // Start the clock, run the alignment code
17     // sf::Clock clock;
18     // EDistance ed(x, y);
19     // std::cout << "opt[] [] is initialized and seeded.\n";
20     // std::cout << ed.printDynamicArray();
21
22     // Start the clock, run the alignment code
23     sf::Clock clock;
24     EDistance ed(x, y);
25     // std::cout << "opt[] [] is initialized and seeded.\n";
26     // std::cout << "x is " << x.length() << " chars\n";
27     // std::cout << "y is " << y.length() << " chars\n";
28     // std::cout << "\n";
29     // At this point, the 2d array should be initialized
30
31     // Compute the Edit Distance
32     int distance = ed.optDistance();
33     // std::cout << "opt[0][0] is the edit distance.\n";
34     // std::cout << ed.printDynamicArray();
35     std::string alignment = ed.alignment();
36     sf::Time elapsed = clock.getElapsedTime();
37
38     // Display the results
39     std::cout << "Edit distance = " << distance << std::endl;
40     std::cout << alignment; // << std::endl;
41     std::cout << "Execution time is " << elapsed.asSeconds() << " seconds"
42     << std::endl;
43     return 0;
44 }

```

```

1 // Copyright Camden Andersson 2024
2 #ifndef EDISTANCE_HPP
3 #define EDISTANCE_HPP
4
5 #include <vector>
6 #include <string>
7 #include <iostream>
8 #include <fstream>
9 #include <SFML/System.hpp>
10 #include <SFML/Graphics.hpp>
11 #include <SFML/Window.hpp>
12
13 class EDistance {
14 public:
15     EDistance(const std::string& x, const std::string& y);
16     ~EDistance();
17     int optDistance();
18     std::string alignment();
19     std::string printDynamicArray();
20     static int penalty(char a, char b);
21     static int min3(int a, int b, int c);
22     static int min3std(int a, int b, int c);
23     static int min3opt(int a, int b, int c);
24 private:
25     std::string x; // the first DNA string
26     std::string y; // the second DNA string
27     std::vector<std::vector<int>> opt; // the n x m array to hold

```

```

    alignments
28 };
29 #endif /* EDISTANCE_HPP */

1 // Copyright Camden Andersson 2024
2 #include "EDistance.hpp"
3 #include <sstream>
4
5 EDistance::EDistance(const std::string& x, const std::string& y) : x(x), y(y)
6 {
7     opt.resize(x.size() + 1, std::vector<int>(y.size() + 1, 0));
8     // Resize the 2d array, and then seed the boundaries with a penalty of 2
9     // The 2 increases as you move left and up
10    for (size_t i = 0; i <= x.size(); ++i) {
11        opt[i][y.size()] = 2 * (x.size() - i);
12    }
13
14    for (size_t j = 0; j <= y.size(); ++j) {
15        opt[x.size()][j] = 2 * (y.size() - j);
16    }
17 }
18
19 EDistance::~EDistance() {
20     // Deallocate memory if allocated using new
21     // not using new, so we can leave this as-is
22 }
23
24 int EDistance::penalty(char a, char b) {
25     return (a == b) ? 0 : 1;
26 }
27
28 int EDistance::min3(int a, int b, int c) {
29     // return min3std(a, b, c);
30     return min3opt(a, b, c);
31 }
32
33 int EDistance::min3std(int a, int b, int c) {
34     // Comparison of three items a, b, c
35     return std::min(std::min(a, b), c);
36 }
37
38 int EDistance::min3opt(int a, int b, int c) {
39     // An alternative method
40     return a < b ? (a < c ? a : c) : (b < c ? b : c);
41 }
42
43 int EDistance::optDistance() {
44     // We start at the bottom of the 2d matrix and work our way up
45     // Right to left, then bottom to top
46     for (int i = x.size() - 1; i >= 0; --i) {
47         for (int j = y.size() - 1; j >= 0; --j) {
48             opt[i][j] = min3(opt[i + 1][j + 1] + penalty(x[i], y[j]),
49                             opt[i + 1][j] + 2,
50                             opt[i][j + 1] + 2);
51         }
52     }
53     // Our Edit Distance is in [0][0]
54     return opt[0][0];
55 }

```

```

56
57 std::string EDistance::alignment() {
58     std::string align;
59     size_t i = 0;
60     size_t j = 0;
61     // Traverse the path and print out the string
62     while (i < x.size() || j < y.size()) {
63         if (i < x.size() && j < y.size() && opt[i][j] == opt[i + 1][j + 1] +
        penalty(x[i], y[j])) {
64             align += x[i];
65             align += ' ';
66             align += y[j];
67             align += ' ';
68             align += std::to_string(penalty(x[i], y[j]));
69             align += '\n';
70             ++i;
71             ++j;
72         } else if (i < x.size() && opt[i][j] == opt[i + 1][j] + 2) {
73             align += x[i];
74             align += ' ';
75             align += '-';
76             align += ' ';
77             align += "2\n";
78             ++i;
79         } else {
80             align += '-';
81             align += ' ';
82             align += y[j];
83             align += ' ';
84             align += "2\n";
85             ++j;
86         }
87     }
88
89     return align;
90 }
91
92 // This will print the full array out
93 std::string EDistance::printDynamicArray() {
94     std::string strOut = "";
95     // size_t a = x.size();
96     // size_t b = y.size();
97
98     std::stringstream ss;
99     for (const auto& row : opt) {
100         for (int value : row) {
101             ss << value << " ";
102         }
103         ss << "\n";
104     }
105
106     std::string optAsString = ss.str();
107     return optAsString;
108 }

```

```

1 // Copyright Camden Andersson 2024
2
3 #include <sstream>
4 #include "EDistance.hpp"
5

```

```

6 #define BOOST_TEST_DYN_LINK
7 #define BOOST_TEST_MODULE Main
8 #include <boost/test/unit_test.hpp>
9 #include <boost/algorithm/string.hpp>
10
11 // example10 and check output
12 BOOST_AUTO_TEST_CASE(testOutput) {
13     std::string x, y;
14     // example10
15     x = "AACAGTTACC";
16     y = "TAAGGTCA";
17     EDistance ed(x, y);
18     ed.optDistance();
19     std::string alignment = ed.alignment();
20     std::stringstream ss;
21     std::stringstream s;
22     boost::erase_all(alignment, "\n");
23     ss << alignment;
24     s << "A T 1";
25     s << "A A 0";
26     s << "C - 2";
27     s << "A A 0";
28     s << "G G 0";
29     s << "T G 1";
30     s << "T T 0";
31     s << "A - 2";
32     s << "C C 0";
33     s << "C A 1";
34     BOOST_REQUIRE_EQUAL(ss.str(), s.str());
35 }
36 // check if the cost = 7
37 BOOST_AUTO_TEST_CASE(testDistance) {
38     std::string x, y;
39     // example10
40     x = "AACAGTTACC";
41     y = "TAAGGTCA";
42     EDistance ed(x, y);
43     int distance = ed.optDistance();
44     BOOST_REQUIRE_EQUAL(7, distance);
45 }

```


7 PS6: Random Writer

7.1 What I accomplished

For PS6: Random Writer, the task involved creating a probabilistic model based on k-grams (fixed sequences of k characters) to generate text that mimics the style of a given input text. This project was based on the principles of Markov chains, as proposed by Claude Shannon, to statistically model letter sequences and generate plausible random text.

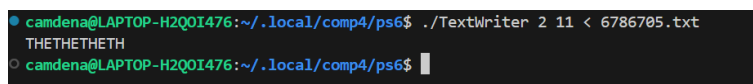


Figure 8: Window produced from running the program

My solution centered around building a Markov model of order k, where each k-gram's occurrence in the text determined the likelihood of subsequent characters. I started by parsing the input text to identify all unique k-grams and recording the frequency of each character that followed each k-gram. This data was stored in a structured format that allowed quick access and modification, facilitating the generation of text based on the observed probabilities.

For the implementation, I utilized a map to associate each k-gram with its subsequent characters and their respective frequencies. This setup enabled me to efficiently query the next likely character given any k-gram, ensuring that the generated text reflected the statistical properties of the source text. I used my lambda in the `freq(string, char)` function, using `findif()`. To handle the text generation, I developed a function that, starting from a randomly chosen k-gram, used the model to produce text of a specified length by continuously appending characters based on the defined probabilities.

The most important part was managing the large amount of data involved in storing the frequencies for all possible k-grams, especially when considering higher orders of k. The complexity of the algorithm increased with k, as did the memory requirements, since each increase in k exponentially grew the number of possible k-grams.

The project results demonstrated the efficacy of the Markov model in generating text that, while nonsensical, bore a stylistic resemblance to the source material. The README also detailed the performance metrics, showing the execution time and memory usage, which were crucial for evaluating the scalability and efficiency of the algorithm.

7.2 What I already knew

Before starting, my knowledge of Markov models was minimal. I had only heard of them briefly and had no practical experience implementing them. This lack of familiarity meant that I was approaching this project with fresh eyes, ready to learn about a new and powerful statistical tool used across various fields of study.

7.3 What I learned

Through this project, I gained a comprehensive understanding of Markov models and their application in generating text. This was a fascinating exploration into how seemingly random elements can be structured in a way that produces surprisingly coherent and reasonable outputs. The project was a significant step up in complexity from earlier assignments, offering a deeper insight into probabilistic models and their practical implementations in computational tasks.

7.4 Challenges

The main challenge I faced was grasping the concept of a Markov model and applying it effectively to the task of text generation. Understanding the underlying principles of how previous states (or k-grams) influence the probability of subsequent events required a shift in my usual approach to programming. This conceptual hurdle was significant, as there was no straightforward method to simplify the complexity of Markov models. Developing a solid understanding of this model was crucial and proved to be the most challenging aspect of the project.

7.5 Codebase

```
1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework
4 # Your .hpp files
5 DEPS = RandWriter.hpp
6 # Your compiled .o files
7 OBJECTS = RandWriter.o
8 # The name of your program
9 PROGRAM = TextWriter
10
11 .PHONY: all clean lint
12
13 all: $(PROGRAM) test TextWriter.a
14
15 # Wildcard recipe to make .o files from corresponding .cpp file
16 %.o: %.cpp $(DEPS)
17     $(CC) $(CFLAGS) -c $<
18
19 $(PROGRAM): TextWriter.o $(OBJECTS)
20     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
21
22 TextWriter.a: RandWriter.o
23     ar rcs TextWriter.a RandWriter.o
24
25 test: test.o RandWriter.o
26     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
27
28 test.o: test.cpp RandWriter.hpp
29     $(CC) $(CFLAGS) -c $< -o $@
30
31 clean:
32     rm *.o $(PROGRAM) *.a test
33
34 lint:
35     cpplint *.cpp *.hpp
```

```
1 // Copyright Camden Andersson 2024
2 #include <iostream>
3 #include "RandWriter.hpp"
4 #include "TextWriter.hpp"
```

```
1 // Copyright Camden Andersson 2024
2 #pragma once
3
4 #ifndef RANDWRITER_HPP
5 #define RANDWRITER_HPP
6
7 #include <string>
8 #include <map>
9 #include <fstream>
10
11 class RandWriter {
12 public:
13     RandWriter(const std::string& text, size_t k);
14
15     size_t orderK() const;
16 }
```

```

17     int freq(const std::string& kgram) const;
18
19     int freq(const std::string& kgram, char c) const;
20
21     char kRand(const std::string& kgram);
22
23     std::string generate(const std::string& kgram, size_t L);
24
25     friend std::ostream& operator<<(std::ostream& os, const RandWriter& rw);
26
27 private:
28     size_t k;
29     std::string alphabet;
30     std::map<std::string, std::map<char, int>> symbolTable;
31
32     std::string circularText(const std::string& text, size_t k);
33
34     void buildSymbolTable(const std::string& text);
35
36     std::string activeAlphabet(const std::string& text);
37 };
38
39 class TextWriter {
40 public:
41     void TextWriter_main(int k, int L, const std::string& filename);
42 };
43
44 #endif /* RANDWRITER_HPP */

```

```

1  // Copyright Camden Andersson 2024
2  #include "RandWriter.hpp"
3  #include <iostream>
4  #include <random>
5  #include <exception>
6  #include <algorithm>
7
8  RandWriter::RandWriter(const std::string& text, size_t k) : k(k) {
9      alphabet = activeAlphabet(text);
10     if (text.length() < k) {
11         throw std::invalid_argument("Constructor Error: text < k");
12     }
13     // Testing out circular text
14     std::string circText = circularText(text, k);
15     // std::cout << circText << std::endl;
16
17     buildSymbolTable(circularText(text, k));
18 }
19
20 size_t RandWriter::orderK() const {
21     return k;
22 }
23
24 std::map<char, int> findEntry(const std::map<std::string,
25 std::map<char, int>>& symbolTable, const std::string& key) {
26     auto it = symbolTable.find(key);
27     if (it != symbolTable.end()) {
28         return it->second;
29     } else {
30         // Return an empty map if the key is not found
31         return {};

```

```

32     }
33 }
34
35 std::string RandWriter::circularText(const std::string& text, size_t k) {
36     return text + text.substr(0, k);
37 }
38
39 void RandWriter::buildSymbolTable(const std::string& text) {
40     for (size_t i = 0; i < text.length() - k; ++i) {
41         std::string kgram = text.substr(i, k);
42         char nextChar = text[i + k];
43         symbolTable[kgram][nextChar]++;
44     }
45 }
46
47 std::string RandWriter::activeAlphabet(const std::string& text) {
48     std::vector<bool> found(128, false);
49     std::string result;
50     for (char c : text) {
51         if (!found[c]) {
52             found[c] = true;
53             result += c;
54         }
55     }
56     return result;
57 }
58
59 std::ostream& operator<<(std::ostream& os, const RandWriter& rw) {
60     os << "Order: " << rw.orderK() << std::endl;
61     os << "Alphabet: " << rw.alphabet << std::endl;
62     for (const auto& entry : rw.symbolTable) {
63         os << "K-gram: " << entry.first << std::endl;
64         for (const auto& freqPair : entry.second) {
65             os << "    Next char: " << freqPair.first <<
66                 "    Frequency: " << freqPair.second << std::endl;
67         }
68     }
69     return os;
70 }
71
72 char RandWriter::kRand(const std::string& kgram) {
73     std::random_device rd;
74     std::mt19937 gen(rd());
75
76     // Rewrite
77     std::vector<int> f;
78     std::vector<char> possibleChars;
79     int total = 0;
80     int n = 0;
81
82     if (kgram.size() != k) {
83         throw std::length_error("kRand Length Error");
84     }
85     // Find the possible letters that follow kgram
86     for (char c : alphabet) {
87         n = freq(kgram, c);
88         if (n > 0) {
89             f.push_back(n);
90             total += n;

```

```

91         possibleChars.push_back(c);
92     }
93 }
94
95 // TO DO: Throw an exception here
96 if (possibleChars.size() == 0) {
97     // std::cout << "EXCEPTION, will crash!" << std::endl;
98     throw std::length_error("possibleChars.size() == 0");
99 }
100
101 // Build the percentages
102 std::vector<double> percentages;
103 for (size_t i = 0; i < f.size(); i++) {
104     f[i] = static_cast<int>((static_cast<double>(f[i]) / static_cast<
double>(total)) * 100.0);
105 }
106 std::discrete_distribution<> d(f.begin(), f.end());
107
108 // Use the random generator
109 int randomNum = d(gen);
110 // char cReturn = possibleChars[randomNum];
111 return possibleChars[randomNum];
112 }
113
114 int RandWriter::freq(const std::string& kgram) const {
115     // Note that for this function you can't use the map's
116     // count() function as that just returns 1 or 0.
117     if (kgram.size() != k) {
118         throw std::length_error("freq Length Error");
119     }
120     std::map<char, int> entry = findEntry(symbolTable, kgram);
121
122     int nNumOccurences = 0;
123     for (const auto& pair : entry) {
124         // Sum up all the frequencies in the last part of the map
125         nNumOccurences += pair.second;
126     }
127
128     return nNumOccurences;
129 }
130 /*
131 int RandWriter::freq(const std::string& kgram, char c) const {
132     if (kgram.size() != k) {
133         throw std::length_error("freq Length Error (2)");
134     }
135     if (symbolTable.count(kgram) && symbolTable.at(kgram).count(c)) {
136         return symbolTable.at(kgram).at(c);
137     }
138     return 0;
139 }
140 */
141 int RandWriter::freq(const std::string& kgram, char c) const {
142     if (kgram.size() != k) {
143         throw std::length_error("freq Length Error (2)");
144     }
145     if (symbolTable.count(kgram)) {
146         const auto& charMap = symbolTable.at(kgram);
147
148         auto it = std::find_if(charMap.begin(), charMap.end(), [c](const std

```

```

149     ::pair<char, int>& p) {
150         return p.first == c;
151     });
152     return (it != charMap.end()) ? it->second : 0;
153 }
154 return 0;
155 }
156
157
158
159 std::string RandWriter::generate(const std::string& kgram, size_t L) {
160     if (kgram.size() != k) {
161         throw std::length_error("generate Length Error");
162     }
163     std::string result = kgram;
164
165     for (size_t i = k; i < L; ++i) {
166         std::string sz = result.substr(result.length() - k);
167         char nextChar = kRand(sz);
168         result += nextChar;
169     }
170     return result;
171 }
172
173
174
175 std::string read_file_to_string(const std::string& filename) {
176     std::ifstream file(filename);
177     if (!file.is_open()) {
178         std::cerr << "Failed to open file: " << filename << std::endl;
179         return "";
180     }
181
182     std::string file_contents;
183     file.seekg(0, std::ios::end);
184     file_contents.reserve(file.tellg());
185     file.seekg(0, std::ios::beg);
186
187     file_contents.assign((std::istreambuf_iterator<char>(file)),
188         std::istreambuf_iterator<char>());
189
190     return file_contents;
191 }
192
193 std::string replace_newlines_with_spaces(const std::string& inText) {
194     std::string result = inText;
195     size_t startPos = 0;
196     while ((startPos = result.find('\n', startPos)) != std::string::npos) {
197         result.replace(startPos, 1, " ");
198         startPos += 1; // Move past the replaced character
199     }
200     return result;
201 }
202
203 void TextWriter::TextWriter_main(int k, int L, const std::string& filename)
204 {
205     // Open the file that contains the text & display it
206     // std::string rawText = read_file_to_string(filename);

```

```

206 // if (!rawText.empty()) {
207 //     std::cout << "File contents:\n" << rawText << std::endl;
208 // }
209 std::string text = replace_newlines_with_spaces(filename);
210
211 // Build the Markov model and display the frequencies
212 RandWriter rw(text, k);
213 // std::cout << rw << std::endl;
214
215 // Set the seed kgram and generate the text
216 std::string kgram = text.substr(0, k);
217 std::cout << rw.generate(kgram, L) << std::endl;
218 }

```

```

1 // Copyright Camden Andersson 2024
2 #include "RandWriter.hpp"
3
4 #define BOOST_TEST_DYN_LINK
5 #define BOOST_TEST_MODULE Main
6 #include <boost/test/unit_test.hpp>
7 #include <boost/algorithm/string.hpp>
8
9 BOOST_AUTO_TEST_CASE(wrongLength) {
10     BOOST_REQUIRE_THROW(RandWriter r("a", 2), std::invalid_argument);
11 }
12
13 BOOST_AUTO_TEST_CASE(generateTest) {
14     RandWriter rw("abcdabcdabcd", 2);
15     std::string generated = rw.generate("ab", 5);
16     BOOST_REQUIRE_EQUAL(generated.size(), 5);
17     BOOST_REQUIRE(generated.substr(0, 2) == "ab");
18 }
19
20 BOOST_AUTO_TEST_CASE(wrongDistributionTest) {
21     RandWriter rw("aaaaabbbbcc", 2);
22     std::map<char, int> charCount;
23     for (int i = 0; i < 1000; ++i) {
24         std::string generated = rw.generate("aa", 4);
25         charCount[generated[2]]++;
26     }
27     BOOST_REQUIRE_CLOSE(static_cast<double>(charCount['a']) / 1000.0, 0.74,
10.0);
28     BOOST_REQUIRE_CLOSE(static_cast<double>(charCount['b']) / 1000.0, 0.26,
10.0);
29 }
30 BOOST_AUTO_TEST_CASE(passTestAmount) {
31     BOOST_REQUIRE_EQUAL(1, 1);
32 }

```


8 PS7: Kronos Log Parsing

8.1 What I accomplished

For PS7: Kronos Time Clock, the project involved parsing and analyzing log files from the Kronos InTouch time clock device using regular expressions. The primary task was to detect and report on the device's boot-up sequences, identifying successful startups and any failures by examining specific log entries that marked the start and end of each boot process.

My approach was to develop a robust parser that utilized regular expressions to sift through the log files and extract relevant information about the device's startup sequences. The implementation involved two main regular expressions: one to detect the initiation of a boot sequence and another to identify its completion. These expressions were designed to capture the necessary timestamps to calculate the duration of each boot process.

To handle the log analysis, I implemented a function that read through the entire log file line by line, applying the regular expressions to determine the start and end of each boot sequence. If a boot sequence was initiated but did not have a corresponding completion before the next sequence began or the file ended, it was reported as a failure. Conversely, successful boot sequences were recorded with their duration calculated using the Boost date/time library to ensure precision.

The implementation also involved managing the pairing of start and end log entries without any overlap or nesting, which was crucial for accurate reporting. Each identified sequence was output with either a success message and the elapsed time or a failure notice, as specified in the project requirements.

For extra credit, I enhanced the output by generating a summary header that provided an overview of all boot sequences, including their start and completion times. This summary was designed to give a quick snapshot of the device's operational history over the period covered by the logs, making it easier for technicians to identify patterns or recurring issues.

This project not only reinforced my skills in using regular expressions but also in applying them to a real-world problem where precision and accuracy are critical. The ability to parse complex log files and extract meaningful information from them is a valuable skill in many software engineering and data analysis roles.

8.2 What I already knew

Before starting on PS7: Kronos Time Clock, I was already quite experienced with log parsing, having previously utilized various tools and programs for this purpose. This background gave me a solid foundation and a comfort level with the task of extracting and analyzing information from log files, making the initial aspects of this project feel quite familiar.

8.3 What I learned

Through this project, I learned to use and understand regular expressions, a tool I had not previously been exposed to. Discovering the power and versatility of regular expressions was enlightening; they proved to be incredibly effective for pattern matching and data extraction in complex text files. This skill has significantly broadened my capabilities in data processing and will undoubtedly be valuable in future projects.

8.4 Challenges

The primary challenge in this project arose during the extra credit portion. While the base assignment was relatively straightforward and within my comfort zone due to my prior experience with log parsing, extending the functionality for the extra credit proved to be more demanding. Despite these challenges, the project required less time and effort overall compared to previous assignments, demonstrating a good balance between complexity and manageability. Unfortunately, the tests for log3 beginning and end and the log6 beginning and end were unable to pass, despite many attempts to fix, so ultimately credit was lost for that. I believe that there was some sort of formatting issue, but I couldn't find the cause in time.

Following is the .rpt log file produced for log 6. As you can see, it incorporates the extra credit (both the header and the extra services).

```

1 Device Boot Report
2
3 InTouch log file: device6_intouch.log
4 Lines Scanned: 90671
5
6 Device boot count: initiated = 14, completed: 12
7
8
9 === Device boot ===
10 2(device6_intouch.log): 2014-04-03 20:27:48 Boot Start
11 161(device6_intouch.log): 2014-04-03 20:31:01 Boot Completed
12     Boot Time: 193000ms
13
14 Services
15     ProtocolService
16         Start:148(device6_intouch.log)
17         Completed:155(device6_intouch.log)
18         ElapsedTime:13020
19     OfflineSmartviewService
20         OfflineSmartviewServiceStart:139(device6_intouch.log)
21         Completed:140(device6_intouch.log)
22         ElapsedTime:11
23     BiometricService
24         BiometricServiceStart:134(device6_intouch.log)
25         Completed:136(device6_intouch.log)
26         ElapsedTime:176
27     SoftLoadService
28         Start:148(device6_intouch.log)
29         Completed:152(device6_intouch.log)
30         ElapsedTime:2932
31     GateService
32         GateServiceStart:132(device6_intouch.log)
33         Completed:133(device6_intouch.log)
34         ElapsedTime:1
35     StateManager
36         StateManagerStart:137(device6_intouch.log)
37         Completed:138(device6_intouch.log)
38         ElapsedTime:1660
39     AVFeedbackService
40         AVFeedbackServiceStart:130(device6_intouch.log)
41         Completed:131(device6_intouch.log)
42         ElapsedTime:27
43     ReaderDataService
44         ReaderDataServiceStart:141(device6_intouch.log)
45         Completed:142(device6_intouch.log)
46         ElapsedTime:11
47     Logging
48         LoggingStart:18(device6_intouch.log)
49         Completed:19(device6_intouch.log)
50         ElapsedTime:271
51     DatabaseInitialize
52         DatabaseInitializeStart:20(device6_intouch.log)
53         Completed:44(device6_intouch.log)
54         ElapsedTime:32365
55     MessagingService
56         MessagingServiceStart:45(device6_intouch.log)
57         Completed:47(device6_intouch.log)
58         ElapsedTime:5488
59     DatabaseThreads

```

```

60     DiagnosticsServiceProtocolServiceWATCHDOGSoftLoadServiceDiagnosticsServiceStart
        :148(device6_intouch.log)
61         Start:148(device6_intouch.log)
62         Completed:151(device6_intouch.log)
63         ElapsedTime:398
64     ConfigurationService
65         ConfigurationServiceStart:114(device6_intouch.log)
66         Completed:115(device6_intouch.log)
67         ElapsedTime:0
68     CacheService
69         CacheServiceStart:116(device6_intouch.log)
70         Completed:117(device6_intouch.log)
71         ElapsedTime:1143
72     StagingService
73         StagingServiceStart:122(device6_intouch.log)
74         Completed:123(device6_intouch.log)
75         ElapsedTime:3290
76     WATCHDOG
77         Start:148(device6_intouch.log)
78         Completed:150(device6_intouch.log)
79         ElapsedTime:190
80     DiagnosticsService
81         Completed:149(device6_intouch.log)
82         ElapsedTime:206
83     HealthMonitorService
84         HealthMonitorServiceStart:109(device6_intouch.log)
85         Completed:110(device6_intouch.log)
86         ElapsedTime:247
87     BellService
88         BellServiceStart:128(device6_intouch.log)
89         Completed:129(device6_intouch.log)
90         ElapsedTime:2
91     Persistence
92         PersistenceStart:111(device6_intouch.log)
93         Completed:113(device6_intouch.log)
94         ElapsedTime:22278
95     ThemingService
96         ThemingServiceStart:118(device6_intouch.log)
97         Completed:119(device6_intouch.log)
98         ElapsedTime:0
99     LandingPadService
100         LandingPadServiceStart:120(device6_intouch.log)
101         Completed:121(device6_intouch.log)
102         ElapsedTime:2
103     PortConfigurationService
104         PortConfigurationServiceStart:124(device6_intouch.log)
105         Completed:125(device6_intouch.log)
106         ElapsedTime:54
107     DeviceIOService
108         DeviceIOServiceStart:126(device6_intouch.log)
109         Completed:127(device6_intouch.log)
110         ElapsedTime:26
111
112     === Device boot ===
113     82079(device6_intouch.log): 2014-04-09 14:51:15 Boot Start
114     82303(device6_intouch.log): 2014-04-09 14:54:39 Boot Completed
115         Boot Time: 204000ms
116

```

```

117 Services
118     ThemingService
119         ThemingServiceStart:82224(device6_intouch.log)
120         Completed:82225(device6_intouch.log)
121         ElapsedTime:1
122     OfflineSmartviewService
123         OfflineSmartviewServiceStart:82222(device6_intouch.log)
124         Completed:82223(device6_intouch.log)
125         ElapsedTime:12
126     BiometricService
127         BiometricServiceStart:82217(device6_intouch.log)
128         Completed:82219(device6_intouch.log)
129         ElapsedTime:19
130     ReaderDataService
131         ReaderDataServiceStart:82215(device6_intouch.log)
132         Completed:82216(device6_intouch.log)
133         ElapsedTime:2
134     Logging
135         LoggingStart:82099(device6_intouch.log)
136         Completed:82100(device6_intouch.log)
137         ElapsedTime:591
138     DatabaseInitialize
139         DatabaseInitializeStart:82101(device6_intouch.log)
140         Completed:82127(device6_intouch.log)
141         ElapsedTime:35131
142     MessagingService
143         MessagingServiceStart:82128(device6_intouch.log)
144         Completed:82130(device6_intouch.log)
145         ElapsedTime:6351
146     DatabaseThreads
147
148 ProtocolServiceSoftLoadServiceWATCHDOGServiceDiagnosticsServiceDiagnosticsServiceStart
149 :82276(device6_intouch.log)
150     Start:82276(device6_intouch.log)
151     Completed:82281(device6_intouch.log)
152     ElapsedTime:2836
153 ConfigurationService
154     ConfigurationServiceStart:82197(device6_intouch.log)
155     Completed:82198(device6_intouch.log)
156     ElapsedTime:0
157 CacheService
158     CacheServiceStart:82201(device6_intouch.log)
159     Completed:82202(device6_intouch.log)
160     ElapsedTime:895
161 StateManager
162     StateManagerStart:82220(device6_intouch.log)
163     Completed:82221(device6_intouch.log)
164     ElapsedTime:2336
165 AVFeedbackService
166     AVFeedbackServiceStart:82209(device6_intouch.log)
167     Completed:82210(device6_intouch.log)
168     ElapsedTime:43
169 StagingService
170     StagingServiceStart:82211(device6_intouch.log)
171     Completed:82212(device6_intouch.log)
172     ElapsedTime:8587
173 WATCHDOG
174     Start:82276(device6_intouch.log)
175     Completed:82278(device6_intouch.log)

```

```

174         ElapsedTime:284
175     DiagnosticsService
176         Completed:82277(device6_intouch.log)
177         ElapsedTime:254
178     HealthMonitorService
179         HealthMonitorServiceStart:82192(device6_intouch.log)
180         Completed:82193(device6_intouch.log)
181         ElapsedTime:164
182     BellService
183         BellServiceStart:82213(device6_intouch.log)
184         Completed:82214(device6_intouch.log)
185         ElapsedTime:1
186     Persistence
187         PersistenceStart:82194(device6_intouch.log)
188         Completed:82196(device6_intouch.log)
189         ElapsedTime:19408
190     PortConfigurationService
191         PortConfigurationServiceStart:82199(device6_intouch.log)
192         Completed:82200(device6_intouch.log)
193         ElapsedTime:57
194     DeviceIOService
195         DeviceIOServiceStart:82203(device6_intouch.log)
196         Completed:82204(device6_intouch.log)
197         ElapsedTime:62
198     LandingPadService
199         LandingPadServiceStart:82205(device6_intouch.log)
200         Completed:82206(device6_intouch.log)
201         ElapsedTime:2
202     SoftLoadService
203         Start:82276(device6_intouch.log)
204         Completed:82282(device6_intouch.log)
205         ElapsedTime:2927
206     GateService
207         GateServiceStart:82207(device6_intouch.log)
208         Completed:82208(device6_intouch.log)
209         ElapsedTime:2
210
211 === Device boot ===
212 85398(device6_intouch.log): 2014-04-10 18:13:13 Boot Start
213 85564(device6_intouch.log): 2014-04-10 18:16:37 Boot Completed
214     Boot Time: 204000ms
215
216 Services
217     SoftLoadService
218         Start:85553(device6_intouch.log)
219         Completed:85561(device6_intouch.log)
220         ElapsedTime:2828
221     GateService
222         GateServiceStart:85544(device6_intouch.log)
223         Completed:85545(device6_intouch.log)
224         ElapsedTime:1
225     OfflineSmartviewService
226         OfflineSmartviewServiceStart:85540(device6_intouch.log)
227         Completed:85541(device6_intouch.log)
228         ElapsedTime:15
229     AVFeedbackService
230         AVFeedbackServiceStart:85542(device6_intouch.log)
231         Completed:85543(device6_intouch.log)
232         ElapsedTime:38

```

```

233 StateManager
234     StateManagerStart:85534(device6_intouch.log)
235     Completed:85535(device6_intouch.log)
236     ElapsedTime:2870
237 ReaderDataService
238     ReaderDataServiceStart:85538(device6_intouch.log)
239     Completed:85539(device6_intouch.log)
240     ElapsedTime:2
241 Logging
242     LoggingStart:85419(device6_intouch.log)
243     Completed:85420(device6_intouch.log)
244     ElapsedTime:256
245 DatabaseInitialize
246     DatabaseInitializeStart:85421(device6_intouch.log)
247     Completed:85447(device6_intouch.log)
248     ElapsedTime:35191
249 MessagingService
250     MessagingServiceStart:85448(device6_intouch.log)
251     Completed:85450(device6_intouch.log)
252     ElapsedTime:7848
253 DatabaseThreads
254
255 ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceDiagnosticsServiceStart
:85553(device6_intouch.log)
256     Start:85553(device6_intouch.log)
257     Completed:85559(device6_intouch.log)
258     ElapsedTime:2802
259 ConfigurationService
260     ConfigurationServiceStart:85517(device6_intouch.log)
261     Completed:85518(device6_intouch.log)
262     ElapsedTime:0
263 CacheService
264     CacheServiceStart:85523(device6_intouch.log)
265     Completed:85524(device6_intouch.log)
266     ElapsedTime:1048
267 StagingService
268     StagingServiceStart:85525(device6_intouch.log)
269     Completed:85526(device6_intouch.log)
270     ElapsedTime:8343
271 WATCHDOG
272     Start:85553(device6_intouch.log)
273     Completed:85555(device6_intouch.log)
274     ElapsedTime:421
275 DiagnosticsService
276     Completed:85554(device6_intouch.log)
277     ElapsedTime:255
278 HealthMonitorService
279     HealthMonitorServiceStart:85512(device6_intouch.log)
280     Completed:85513(device6_intouch.log)
281     ElapsedTime:211
282 BellService
283     BellServiceStart:85536(device6_intouch.log)
284     Completed:85537(device6_intouch.log)
285     ElapsedTime:3
286 Persistence
287     PersistenceStart:85514(device6_intouch.log)
288     Completed:85516(device6_intouch.log)
289     ElapsedTime:18367
290 ThemingService

```



```

290     ThemingServiceStart:85519(device6_intouch.log)
291     Completed:85520(device6_intouch.log)
292     ElapsedTime:0
293 LandingPadService
294     LandingPadServiceStart:85521(device6_intouch.log)
295     Completed:85522(device6_intouch.log)
296     ElapsedTime:2
297 PortConfigurationService
298     PortConfigurationServiceStart:85527(device6_intouch.log)
299     Completed:85528(device6_intouch.log)
300     ElapsedTime:52
301 DeviceIOService
302     DeviceIOServiceStart:85529(device6_intouch.log)
303     Completed:85530(device6_intouch.log)
304     ElapsedTime:24
305 BiometricService
306     BiometricServiceStart:85531(device6_intouch.log)
307     Completed:85533(device6_intouch.log)
308     ElapsedTime:12
309
310 === Device boot ===
311 85957(device6_intouch.log): 2014-04-10 19:11:05 Boot Start
312 86123(device6_intouch.log): 2014-04-10 19:14:24 Boot Completed
313     Boot Time: 199000ms
314
315 Services
316     OfflineSmartviewService
317         OfflineSmartviewServiceStart:86103(device6_intouch.log)
318         Completed:86104(device6_intouch.log)
319         ElapsedTime:41
320     BiometricService
321         BiometricServiceStart:86096(device6_intouch.log)
322         Completed:86098(device6_intouch.log)
323         ElapsedTime:13
324     StateManager
325         StateManagerStart:86101(device6_intouch.log)
326         Completed:86102(device6_intouch.log)
327         ElapsedTime:2027
328     AVFeedbackService
329         AVFeedbackServiceStart:86092(device6_intouch.log)
330         Completed:86093(device6_intouch.log)
331         ElapsedTime:32
332     ReaderDataService
333         ReaderDataServiceStart:86094(device6_intouch.log)
334         Completed:86095(device6_intouch.log)
335         ElapsedTime:2
336     Logging
337         LoggingStart:85978(device6_intouch.log)
338         Completed:85979(device6_intouch.log)
339         ElapsedTime:289
340     DatabaseInitialize
341         DatabaseInitializeStart:85980(device6_intouch.log)
342         Completed:86004(device6_intouch.log)
343         ElapsedTime:33943
344     MessagingService
345         MessagingServiceStart:86007(device6_intouch.log)
346         Completed:86009(device6_intouch.log)
347         ElapsedTime:7570
348     DatabaseThreads

```



```

349 ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceWATCHDOGStart
    :86112(device6_intouch.log)
350     Start:86112(device6_intouch.log)
351     Completed:86117(device6_intouch.log)
352     ElapsedTime:2786
353 ConfigurationService
354     ConfigurationServiceStart:86076(device6_intouch.log)
355     Completed:86077(device6_intouch.log)
356     ElapsedTime:0
357 CacheService
358     CacheServiceStart:86084(device6_intouch.log)
359     Completed:86085(device6_intouch.log)
360     ElapsedTime:1114
361 StagingService
362     StagingServiceStart:86086(device6_intouch.log)
363     Completed:86087(device6_intouch.log)
364     ElapsedTime:7899
365 DiagnosticsService
366     Start:86112(device6_intouch.log)
367     Completed:86114(device6_intouch.log)
368     ElapsedTime:202
369 WATCHDOG
370     Completed:86113(device6_intouch.log)
371     ElapsedTime:82
372 HealthMonitorService
373     HealthMonitorServiceStart:86071(device6_intouch.log)
374     Completed:86072(device6_intouch.log)
375     ElapsedTime:264
376 BellService
377     BellServiceStart:86099(device6_intouch.log)
378     Completed:86100(device6_intouch.log)
379     ElapsedTime:1
380 Persistence
381     PersistenceStart:86073(device6_intouch.log)
382     Completed:86075(device6_intouch.log)
383     ElapsedTime:18488
384 LandingPadService
385     LandingPadServiceStart:86078(device6_intouch.log)
386     Completed:86079(device6_intouch.log)
387     ElapsedTime:2
388 ThemingService
389     ThemingServiceStart:86080(device6_intouch.log)
390     Completed:86081(device6_intouch.log)
391     ElapsedTime:1
392 PortConfigurationService
393     PortConfigurationServiceStart:86082(device6_intouch.log)
394     Completed:86083(device6_intouch.log)
395     ElapsedTime:95
396 DeviceIOService
397     DeviceIOServiceStart:86088(device6_intouch.log)
398     Completed:86089(device6_intouch.log)
399     ElapsedTime:24
400 SoftLoadService
401     Start:86112(device6_intouch.log)
402     Completed:86119(device6_intouch.log)
403     ElapsedTime:2991
404 GateService
405     GateServiceStart:86090(device6_intouch.log)

```

```

406         Completed:86091(device6_intouch.log)
407         ElapsedTime:2
408
409 === Device boot ===
410 86127(device6_intouch.log): 2014-04-10 19:18:36 Boot Start
411 86293(device6_intouch.log): 2014-04-10 19:21:56 Boot Completed
412     Boot Time: 200000ms
413
414 Services
415     SoftLoadService
416         Start:86282(device6_intouch.log)
417         Completed:86289(device6_intouch.log)
418         ElapsedTime:2835
419     GateService
420         GateServiceStart:86273(device6_intouch.log)
421         Completed:86274(device6_intouch.log)
422         ElapsedTime:1
423     OfflineSmartviewService
424         OfflineSmartviewServiceStart:86269(device6_intouch.log)
425         Completed:86270(device6_intouch.log)
426         ElapsedTime:15
427     AVFeedbackService
428         AVFeedbackServiceStart:86271(device6_intouch.log)
429         Completed:86272(device6_intouch.log)
430         ElapsedTime:33
431     StateManager
432         StateManagerStart:86263(device6_intouch.log)
433         Completed:86264(device6_intouch.log)
434         ElapsedTime:3052
435     ReaderDataService
436         ReaderDataServiceStart:86267(device6_intouch.log)
437         Completed:86268(device6_intouch.log)
438         ElapsedTime:2
439     Logging
440         LoggingStart:86148(device6_intouch.log)
441         Completed:86149(device6_intouch.log)
442         ElapsedTime:282
443     DatabaseInitialize
444         DatabaseInitializeStart:86150(device6_intouch.log)
445         Completed:86176(device6_intouch.log)
446         ElapsedTime:34292
447     MessagingService
448         MessagingServiceStart:86177(device6_intouch.log)
449         Completed:86179(device6_intouch.log)
450         ElapsedTime:6619
451     DatabaseThreads
452
453 ProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceDiagnosticsServiceStart
454 :86282(device6_intouch.log)
455     Start:86282(device6_intouch.log)
456     Completed:86287(device6_intouch.log)
457     ElapsedTime:2833
458 ConfigurationService
459     ConfigurationServiceStart:86246(device6_intouch.log)
460     Completed:86247(device6_intouch.log)
461     ElapsedTime:0
462 CacheService
463     CacheServiceStart:86252(device6_intouch.log)
464     Completed:86253(device6_intouch.log)

```

```

463         ElapsedTime:963
464     StagingService
465         StagingServiceStart:86254(device6_intouch.log)
466         Completed:86255(device6_intouch.log)
467         ElapsedTime:8053
468     WATCHDOG
469         Start:86282(device6_intouch.log)
470         Completed:86284(device6_intouch.log)
471         ElapsedTime:178
472     DiagnosticsService
473         Completed:86283(device6_intouch.log)
474         ElapsedTime:114
475     HealthMonitorService
476         HealthMonitorServiceStart:86241(device6_intouch.log)
477         Completed:86242(device6_intouch.log)
478         ElapsedTime:187
479     BellService
480         BellServiceStart:86265(device6_intouch.log)
481         Completed:86266(device6_intouch.log)
482         ElapsedTime:2
483     Persistence
484         PersistenceStart:86243(device6_intouch.log)
485         Completed:86245(device6_intouch.log)
486         ElapsedTime:18914
487     ThemingService
488         ThemingServiceStart:86248(device6_intouch.log)
489         Completed:86249(device6_intouch.log)
490         ElapsedTime:0
491     LandingPadService
492         LandingPadServiceStart:86250(device6_intouch.log)
493         Completed:86251(device6_intouch.log)
494         ElapsedTime:2
495     PortConfigurationService
496         PortConfigurationServiceStart:86256(device6_intouch.log)
497         Completed:86257(device6_intouch.log)
498         ElapsedTime:56
499     DeviceIOService
500         DeviceIOServiceStart:86258(device6_intouch.log)
501         Completed:86259(device6_intouch.log)
502         ElapsedTime:22
503     BiometricService
504         BiometricServiceStart:86260(device6_intouch.log)
505         Completed:86262(device6_intouch.log)
506         ElapsedTime:12
507
508     === Device boot ===
509     86568(device6_intouch.log): 2014-04-10 19:32:16 Boot Start
510     86732(device6_intouch.log): 2014-04-10 19:35:36 Boot Completed
511         Boot Time: 200000ms
512
513     Services
514         OfflineSmartviewService
515             OfflineSmartviewServiceStart:86712(device6_intouch.log)
516             Completed:86713(device6_intouch.log)
517             ElapsedTime:20
518         BiometricService
519             BiometricServiceStart:86707(device6_intouch.log)
520             Completed:86709(device6_intouch.log)
521             ElapsedTime:12

```

```

522 ThemingService
523     ThemingServiceStart:86705(device6_intouch.log)
524     Completed:86706(device6_intouch.log)
525     ElapsedTime:0
526 ReaderDataService
527     ReaderDataServiceStart:86703(device6_intouch.log)
528     Completed:86704(device6_intouch.log)
529     ElapsedTime:499
530 Logging
531     LoggingStart:86588(device6_intouch.log)
532     Completed:86589(device6_intouch.log)
533     ElapsedTime:291
534 DatabaseInitialize
535     DatabaseInitializeStart:86590(device6_intouch.log)
536     Completed:86614(device6_intouch.log)
537     ElapsedTime:33829
538 MessagingService
539     MessagingServiceStart:86617(device6_intouch.log)
540     Completed:86619(device6_intouch.log)
541     ElapsedTime:5723
542 DatabaseThreads
543
544 ProtocolServiceDiagnosticsServiceWATCHDOGSoftLoadServiceWATCHDOGStart
545 :86721(device6_intouch.log)
546     Start:86721(device6_intouch.log)
547     Completed:86726(device6_intouch.log)
548     ElapsedTime:2953
549 ConfigurationService
550     ConfigurationServiceStart:86685(device6_intouch.log)
551     Completed:86686(device6_intouch.log)
552     ElapsedTime:0
553 CacheService
554     CacheServiceStart:86695(device6_intouch.log)
555     Completed:86696(device6_intouch.log)
556     ElapsedTime:872
557 StateManager
558     StateManagerStart:86710(device6_intouch.log)
559     Completed:86711(device6_intouch.log)
560     ElapsedTime:2273
561 AVFeedbackService
562     AVFeedbackServiceStart:86697(device6_intouch.log)
563     Completed:86698(device6_intouch.log)
564     ElapsedTime:34
565 StagingService
566     StagingServiceStart:86699(device6_intouch.log)
567     Completed:86700(device6_intouch.log)
568     ElapsedTime:7576
569 DiagnosticsService
570     Start:86721(device6_intouch.log)
571     Completed:86723(device6_intouch.log)
572     ElapsedTime:104
573 WATCHDOG
574     Completed:86722(device6_intouch.log)
575     ElapsedTime:97
576 HealthMonitorService
577     HealthMonitorServiceStart:86681(device6_intouch.log)
578     Completed:86682(device6_intouch.log)
579     ElapsedTime:173
580 BellService

```

```

579     BellServiceStart:86701(device6_intouch.log)
580     Completed:86702(device6_intouch.log)
581     ElapsedTime:2
582 Persistence
583     PersistenceStart:86683(device6_intouch.log)
584     Completed:86684(device6_intouch.log)
585     ElapsedTime:22924
586 LandingPadService
587     LandingPadServiceStart:86687(device6_intouch.log)
588     Completed:86688(device6_intouch.log)
589     ElapsedTime:2
590 PortConfigurationService
591     PortConfigurationServiceStart:86689(device6_intouch.log)
592     Completed:86690(device6_intouch.log)
593     ElapsedTime:65
594 DeviceIOService
595     DeviceIOServiceStart:86691(device6_intouch.log)
596     Completed:86692(device6_intouch.log)
597     ElapsedTime:53
598 SoftLoadService
599     Start:86721(device6_intouch.log)
600     Completed:86729(device6_intouch.log)
601     ElapsedTime:3194
602 GateService
603     GateServiceStart:86693(device6_intouch.log)
604     Completed:86694(device6_intouch.log)
605     ElapsedTime:3
606
607 === Device boot ===
608 86750(device6_intouch.log): 2014-04-10 20:06:27 Boot Start
609 86821(device6_intouch.log): 2014-04-10 20:09:07 Boot Completed
610     Boot Time: 160000ms
611
612 Services
613     OfflineSmartviewService
614         OfflineSmartviewServiceStart:86801(device6_intouch.log)
615         Completed:86802(device6_intouch.log)
616         ElapsedTime:12
617     BiometricService
618         BiometricServiceStart:86797(device6_intouch.log)
619         Completed:86798(device6_intouch.log)
620         ElapsedTime:685
621     StagingService
622         StagingServiceStart:86795(device6_intouch.log)
623         Completed:86796(device6_intouch.log)
624         ElapsedTime:6961
625     ReaderDataService
626         ReaderDataServiceStart:86805(device6_intouch.log)
627         Completed:86806(device6_intouch.log)
628         ElapsedTime:2
629     Logging
630         LoggingStart:86765(device6_intouch.log)
631         Completed:86766(device6_intouch.log)
632         ElapsedTime:305
633     DatabaseInitialize
634         DatabaseInitializeStart:86767(device6_intouch.log)
635         Completed:86768(device6_intouch.log)
636         ElapsedTime:33168
637     MessagingService

```



```

638      MessagingServiceStart:86769(device6_intouch.log)
639      Completed:86770(device6_intouch.log)
640      ElapsedTime:5776
641      DatabaseThreads
642      Start:86812(device6_intouch.log)
643      Completed:86820(device6_intouch.log)
644      ElapsedTime:4099
645      ConfigurationService
646      ConfigurationServiceStart:86779(device6_intouch.log)
647      Completed:86780(device6_intouch.log)
648      ElapsedTime:0
649      CacheService
650      CacheServiceStart:86781(device6_intouch.log)
651      Completed:86782(device6_intouch.log)
652      ElapsedTime:1915
653      StateManager
654      StateManagerStart:86799(device6_intouch.log)
655      Completed:86800(device6_intouch.log)
656      ElapsedTime:1793
657      AVFeedbackService
658      AVFeedbackServiceStart:86793(device6_intouch.log)
659      Completed:86794(device6_intouch.log)
660      ElapsedTime:53
661      WATCHDOG
662      Start:86812(device6_intouch.log)
663      Completed:86815(device6_intouch.log)
664      ElapsedTime:415
665      DiagnosticsService
666
667      ProtocolServiceWATCHDOGSoftLoadServiceDatabaseThreadsDiagnosticsServiceStart
:86812(device6_intouch.log)
668      Completed:86814(device6_intouch.log)
669      ElapsedTime:250
670      HealthMonitorService
671      HealthMonitorServiceStart:86775(device6_intouch.log)
672      Completed:86776(device6_intouch.log)
673      ElapsedTime:575
674      BellService
675      BellServiceStart:86803(device6_intouch.log)
676      Completed:86804(device6_intouch.log)
677      ElapsedTime:2
678      Persistence
679      PersistenceStart:86777(device6_intouch.log)
680      Completed:86778(device6_intouch.log)
681      ElapsedTime:18205
682      ThemingService
683      ThemingServiceStart:86783(device6_intouch.log)
684      Completed:86784(device6_intouch.log)
685      ElapsedTime:0
686      LandingPadService
687      LandingPadServiceStart:86785(device6_intouch.log)
688      Completed:86786(device6_intouch.log)
689      ElapsedTime:2
690      PortConfigurationService
691      PortConfigurationServiceStart:86787(device6_intouch.log)
692      Completed:86788(device6_intouch.log)
693      ElapsedTime:43
694      DeviceIOService
695      DeviceIOServiceStart:86789(device6_intouch.log)

```

```

695         Completed:86790(device6_intouch.log)
696         ElapsedTime:24
697     SoftLoadService
698         Start:86812(device6_intouch.log)
699         Completed:86817(device6_intouch.log)
700         ElapsedTime:3307
701     GateService
702         GateServiceStart:86791(device6_intouch.log)
703         Completed:86792(device6_intouch.log)
704         ElapsedTime:196
705
706 === Device boot ===
707 86939(device6_intouch.log): 2014-04-11 00:15:56 Boot Start
708 87111(device6_intouch.log): 2014-04-11 00:18:49 Boot Completed
709     Boot Time: 173000ms
710
711 Services
712     OfflineSmartviewService
713         OfflineSmartviewServiceStart:87030(device6_intouch.log)
714         Completed:87031(device6_intouch.log)
715         ElapsedTime:18
716     StateManager
717         StateManagerStart:87028(device6_intouch.log)
718         Completed:87029(device6_intouch.log)
719         ElapsedTime:2752
720     AVFeedbackService
721         AVFeedbackServiceStart:87026(device6_intouch.log)
722         Completed:87027(device6_intouch.log)
723         ElapsedTime:34
724     SoftLoadService
725         Start:87083(device6_intouch.log)
726         Completed:87099(device6_intouch.log)
727         ElapsedTime:4041
728     GateService
729         GateServiceStart:87024(device6_intouch.log)
730         Completed:87025(device6_intouch.log)
731         ElapsedTime:158
732     BiometricService
733         BiometricServiceStart:87022(device6_intouch.log)
734         Completed:87023(device6_intouch.log)
735         ElapsedTime:677
736     DatabaseInitialize
737         DatabaseInitializeStart:86961(device6_intouch.log)
738         Completed:86987(device6_intouch.log)
739         ElapsedTime:37703
740     MessagingService
741         MessagingServiceStart:86989(device6_intouch.log)
742         Completed:86991(device6_intouch.log)
743         ElapsedTime:8660
744     DatabaseThreads
745
746 ProtocolServiceSoftLoadServiceDiagnosticsServiceWATCHDOGWATCHDOGStart
747 :87083(device6_intouch.log)
748     Start:87083(device6_intouch.log)
749     Completed:87088(device6_intouch.log)
750     ElapsedTime:3747
751 ConfigurationService
752     ConfigurationServiceStart:87006(device6_intouch.log)
753     Completed:87007(device6_intouch.log)

```



```

752         ElapsedTime:0
753     CacheService
754         CacheServiceStart:87008(device6_intouch.log)
755         Completed:87009(device6_intouch.log)
756         ElapsedTime:988
757     StagingService
758         StagingServiceStart:87016(device6_intouch.log)
759         Completed:87017(device6_intouch.log)
760         ElapsedTime:8398
761     DiagnosticsService
762         Start:87083(device6_intouch.log)
763         Completed:87085(device6_intouch.log)
764         ElapsedTime:240
765     WATCHDOG
766         Completed:87084(device6_intouch.log)
767         ElapsedTime:229
768     HealthMonitorService
769         HealthMonitorServiceStart:87002(device6_intouch.log)
770         Completed:87003(device6_intouch.log)
771         ElapsedTime:219
772     BellService
773         BellServiceStart:87032(device6_intouch.log)
774         Completed:87033(device6_intouch.log)
775         ElapsedTime:2
776     Persistence
777         PersistenceStart:87004(device6_intouch.log)
778         Completed:87005(device6_intouch.log)
779         ElapsedTime:19331
780     ThemingService
781         ThemingServiceStart:87010(device6_intouch.log)
782         Completed:87011(device6_intouch.log)
783         ElapsedTime:0
784     LandingPadService
785         LandingPadServiceStart:87012(device6_intouch.log)
786         Completed:87013(device6_intouch.log)
787         ElapsedTime:10
788     PortConfigurationService
789         PortConfigurationServiceStart:87014(device6_intouch.log)
790         Completed:87015(device6_intouch.log)
791         ElapsedTime:44
792     DeviceIOService
793         DeviceIOServiceStart:87018(device6_intouch.log)
794         Completed:87019(device6_intouch.log)
795         ElapsedTime:22
796     Logging
797         LoggingStart:86959(device6_intouch.log)
798         Completed:86960(device6_intouch.log)
799         ElapsedTime:410
800     ReaderDataService
801         ReaderDataServiceStart:87020(device6_intouch.log)
802         Completed:87021(device6_intouch.log)
803         ElapsedTime:2
804
805     === Device boot ===
806     87116(device6_intouch.log): 2014-04-11 13:28:25 Boot Start
807     87286(device6_intouch.log): 2014-04-11 13:31:12 Boot Completed
808         Boot Time: 167000ms
809
810     Services

```

```

811 OfflineSmartviewService
812     OfflineSmartviewServiceStart:87208(device6_intouch.log)
813     Completed:87209(device6_intouch.log)
814     ElapsedTime:22
815 BiometricService
816     BiometricServiceStart:87204(device6_intouch.log)
817     Completed:87205(device6_intouch.log)
818     ElapsedTime:688
819 StagingService
820     StagingServiceStart:87198(device6_intouch.log)
821     Completed:87199(device6_intouch.log)
822     ElapsedTime:8737
823 ReaderDataService
824     ReaderDataServiceStart:87200(device6_intouch.log)
825     Completed:87201(device6_intouch.log)
826     ElapsedTime:3
827 Logging
828     LoggingStart:87136(device6_intouch.log)
829     Completed:87137(device6_intouch.log)
830     ElapsedTime:322
831 DatabaseInitialize
832     DatabaseInitializeStart:87138(device6_intouch.log)
833     Completed:87164(device6_intouch.log)
834     ElapsedTime:34023
835 MessagingService
836     MessagingServiceStart:87165(device6_intouch.log)
837     Completed:87167(device6_intouch.log)
838     ElapsedTime:6415
839 DatabaseThreads
840
841 ProtocolServiceSoftLoadServiceWATCHDOGServiceDiagnosticsServiceDiagnosticsServiceStart
842 :87259(device6_intouch.log)
843     Start:87259(device6_intouch.log)
844     Completed:87265(device6_intouch.log)
845     ElapsedTime:3552
846 ConfigurationService
847     ConfigurationServiceStart:87182(device6_intouch.log)
848     Completed:87183(device6_intouch.log)
849     ElapsedTime:0
850 CacheService
851     CacheServiceStart:87194(device6_intouch.log)
852     Completed:87195(device6_intouch.log)
853     ElapsedTime:1096
854 StateManager
855     StateManagerStart:87206(device6_intouch.log)
856     Completed:87207(device6_intouch.log)
857     ElapsedTime:2958
858 AVFeedbackService
859     AVFeedbackServiceStart:87196(device6_intouch.log)
860     Completed:87197(device6_intouch.log)
861     ElapsedTime:55
862 WATCHDOG
863     Start:87259(device6_intouch.log)
864     Completed:87261(device6_intouch.log)
865     ElapsedTime:376
866 DiagnosticsService
867     Completed:87260(device6_intouch.log)
868     ElapsedTime:158
869 HealthMonitorService

```

```

868     HealthMonitorServiceStart:87178(device6_intouch.log)
869     Completed:87179(device6_intouch.log)
870     ElapsedTime:152
871     BellService
872         BellServiceStart:87202(device6_intouch.log)
873         Completed:87203(device6_intouch.log)
874         ElapsedTime:2
875     Persistence
876         PersistenceStart:87180(device6_intouch.log)
877         Completed:87181(device6_intouch.log)
878         ElapsedTime:20159
879     LandingPadService
880         LandingPadServiceStart:87184(device6_intouch.log)
881         Completed:87185(device6_intouch.log)
882         ElapsedTime:2
883     ThemingService
884         ThemingServiceStart:87190(device6_intouch.log)
885         Completed:87191(device6_intouch.log)
886         ElapsedTime:1
887     DeviceIOService
888         DeviceIOServiceStart:87186(device6_intouch.log)
889         Completed:87187(device6_intouch.log)
890         ElapsedTime:28
891     SoftLoadService
892         Start:87259(device6_intouch.log)
893         Completed:87263(device6_intouch.log)
894         ElapsedTime:3323
895     GateService
896         GateServiceStart:87188(device6_intouch.log)
897         Completed:87189(device6_intouch.log)
898         ElapsedTime:415
899     PortConfigurationService
900         PortConfigurationServiceStart:87192(device6_intouch.log)
901         Completed:87193(device6_intouch.log)
902         ElapsedTime:37
903
904 === Device boot ===
905 87836(device6_intouch.log): 2014-04-11 13:58:02 Boot Start
906 88009(device6_intouch.log): 2014-04-11 14:00:49 Boot Completed
907     Boot Time: 167000ms
908
909 Services
910     OfflineSmartviewService
911         OfflineSmartviewServiceStart:87931(device6_intouch.log)
912         Completed:87932(device6_intouch.log)
913         ElapsedTime:22
914     BiometricService
915         BiometricServiceStart:87927(device6_intouch.log)
916         Completed:87928(device6_intouch.log)
917         ElapsedTime:669
918     SoftLoadService
919         Start:87982(device6_intouch.log)
920         Completed:87989(device6_intouch.log)
921         ElapsedTime:3776
922     GateService
923         GateServiceStart:87925(device6_intouch.log)
924         Completed:87926(device6_intouch.log)
925         ElapsedTime:1
926     ReaderDataService

```

```

927     ReaderDataServiceStart:87921(device6_intouch.log)
928     Completed:87922(device6_intouch.log)
929     ElapsedTime:2
930     Logging
931         LoggingStart:87857(device6_intouch.log)
932         Completed:87858(device6_intouch.log)
933         ElapsedTime:331
934     DatabaseInitialize
935         DatabaseInitializeStart:87859(device6_intouch.log)
936         Completed:87885(device6_intouch.log)
937         ElapsedTime:35524
938     MessagingService
939         MessagingServiceStart:87886(device6_intouch.log)
940         Completed:87888(device6_intouch.log)
941         ElapsedTime:6809
942     DatabaseThreads
943
944     ProtocolServiceWATCHDOGSoftLoadServiceDiagnosticsServiceDiagnosticsServiceStart
945     :87982(device6_intouch.log)
946         Start:87982(device6_intouch.log)
947         Completed:87987(device6_intouch.log)
948         ElapsedTime:3549
949     ConfigurationService
950         ConfigurationServiceStart:87905(device6_intouch.log)
951         Completed:87906(device6_intouch.log)
952         ElapsedTime:0
953     CacheService
954         CacheServiceStart:87911(device6_intouch.log)
955         Completed:87912(device6_intouch.log)
956         ElapsedTime:1542
957     StagingService
958         StagingServiceStart:87915(device6_intouch.log)
959         Completed:87916(device6_intouch.log)
960         ElapsedTime:8060
961     StateManager
962         StateManagerStart:87929(device6_intouch.log)
963         Completed:87930(device6_intouch.log)
964         ElapsedTime:2798
965     AVFeedbackService
966         AVFeedbackServiceStart:87919(device6_intouch.log)
967         Completed:87920(device6_intouch.log)
968         ElapsedTime:37
969     WATCHDOG
970         Start:87982(device6_intouch.log)
971         Completed:87984(device6_intouch.log)
972         ElapsedTime:321
973     DiagnosticsService
974         Completed:87983(device6_intouch.log)
975         ElapsedTime:229
976     HealthMonitorService
977         HealthMonitorServiceStart:87899(device6_intouch.log)
978         Completed:87900(device6_intouch.log)
979         ElapsedTime:166
980     BellService
981         BellServiceStart:87923(device6_intouch.log)
982         Completed:87924(device6_intouch.log)
983         ElapsedTime:203
984     Persistence
985         PersistenceStart:87901(device6_intouch.log)

```

```

984         Completed:87904(device6_intouch.log)
985         ElapsedTime:19998
986     LandingPadService
987         LandingPadServiceStart:87907(device6_intouch.log)
988         Completed:87908(device6_intouch.log)
989         ElapsedTime:2
990     ThemingService
991         ThemingServiceStart:87913(device6_intouch.log)
992         Completed:87914(device6_intouch.log)
993         ElapsedTime:1
994     PortConfigurationService
995         PortConfigurationServiceStart:87909(device6_intouch.log)
996         Completed:87910(device6_intouch.log)
997         ElapsedTime:39
998     DeviceIOService
999         DeviceIOServiceStart:87917(device6_intouch.log)
1000        Completed:87918(device6_intouch.log)
1001        ElapsedTime:24
1002
1003    === Device boot ===
1004    88983(device6_intouch.log): 2014-04-11 14:23:42 Boot Start
1005    89155(device6_intouch.log): 2014-04-11 14:26:31 Boot Completed
1006        Boot Time: 169000ms
1007
1008    Services
1009        ThemingService
1010            ThemingServiceStart:89076(device6_intouch.log)
1011            Completed:89077(device6_intouch.log)
1012            ElapsedTime:0
1013        PortConfigurationService
1014            PortConfigurationServiceStart:89074(device6_intouch.log)
1015            Completed:89075(device6_intouch.log)
1016            ElapsedTime:63
1017        OfflineSmartviewService
1018            OfflineSmartviewServiceStart:89070(device6_intouch.log)
1019            Completed:89071(device6_intouch.log)
1020            ElapsedTime:22
1021        BiometricService
1022            BiometricServiceStart:89066(device6_intouch.log)
1023            Completed:89067(device6_intouch.log)
1024            ElapsedTime:688
1025        DatabaseInitialize
1026            DatabaseInitializeStart:89006(device6_intouch.log)
1027            Completed:89030(device6_intouch.log)
1028            ElapsedTime:35720
1029        MessagingService
1030            MessagingServiceStart:89033(device6_intouch.log)
1031            Completed:89035(device6_intouch.log)
1032            ElapsedTime:6065
1033        DatabaseThreads
1034
1035        ProtocolServiceWATCHDOGSoftLoadServiceDiagnosticsServiceDiagnosticsServiceStart
1036        :89127(device6_intouch.log)
1037        Start:89127(device6_intouch.log)
1038        Completed:89134(device6_intouch.log)
1039        ElapsedTime:3717
1040    ConfigurationService
1041        ConfigurationServiceStart:89050(device6_intouch.log)
1042        Completed:89051(device6_intouch.log)

```



```

1041         ElapsedTime:0
1042     CacheService
1043         CacheServiceStart:89052(device6_intouch.log)
1044         Completed:89053(device6_intouch.log)
1045         ElapsedTime:987
1046     StateManager
1047         StateManagerStart:89068(device6_intouch.log)
1048         Completed:89069(device6_intouch.log)
1049         ElapsedTime:2935
1050     AVFeedbackService
1051         AVFeedbackServiceStart:89058(device6_intouch.log)
1052         Completed:89059(device6_intouch.log)
1053         ElapsedTime:45
1054     StagingService
1055         StagingServiceStart:89062(device6_intouch.log)
1056         Completed:89063(device6_intouch.log)
1057         ElapsedTime:7935
1058     WATCHDOG
1059         Start:89127(device6_intouch.log)
1060         Completed:89129(device6_intouch.log)
1061         ElapsedTime:258
1062     DiagnosticsService
1063         Completed:89128(device6_intouch.log)
1064         ElapsedTime:169
1065     HealthMonitorService
1066         HealthMonitorServiceStart:89046(device6_intouch.log)
1067         Completed:89047(device6_intouch.log)
1068         ElapsedTime:223
1069     BellService
1070         BellServiceStart:89072(device6_intouch.log)
1071         Completed:89073(device6_intouch.log)
1072         ElapsedTime:1
1073     Persistence
1074         PersistenceStart:89048(device6_intouch.log)
1075         Completed:89049(device6_intouch.log)
1076         ElapsedTime:20056
1077     DeviceIOService
1078         DeviceIOServiceStart:89054(device6_intouch.log)
1079         Completed:89055(device6_intouch.log)
1080         ElapsedTime:26
1081     SoftLoadService
1082         Start:89127(device6_intouch.log)
1083         Completed:89132(device6_intouch.log)
1084         ElapsedTime:3584
1085     GateService
1086         GateServiceStart:89056(device6_intouch.log)
1087         Completed:89057(device6_intouch.log)
1088         ElapsedTime:169
1089     LandingPadService
1090         LandingPadServiceStart:89060(device6_intouch.log)
1091         Completed:89061(device6_intouch.log)
1092         ElapsedTime:2
1093     Logging
1094         LoggingStart:89004(device6_intouch.log)
1095         Completed:89005(device6_intouch.log)
1096         ElapsedTime:311
1097     ReaderDataService
1098         ReaderDataServiceStart:89064(device6_intouch.log)
1099         Completed:89065(device6_intouch.log)

```

```

1100         ElapsedTime:2
1101
1102 === Device boot ===
1103 90112(device6_intouch.log): 2014-04-14 12:13:59 Boot Start
1104 **** Incomplete boot ****
1105
1106 Services
1107     Logging
1108         Start: Not started(device6_intouch.log)
1109         Completed: Not completed(device6_intouch.log)
1110         Elapsed Time:
1111     DatabaseInitialize
1112         Start: Not started(device6_intouch.log)
1113         Completed: Not completed(device6_intouch.log)
1114         Elapsed Time:
1115     MessagingService
1116         Start: Not started(device6_intouch.log)
1117         Completed: Not completed(device6_intouch.log)
1118         Elapsed Time:
1119     HealthMonitorService
1120         Start: Not started(device6_intouch.log)
1121         Completed: Not completed(device6_intouch.log)
1122         Elapsed Time:
1123     Persistence
1124         Start: Not started(device6_intouch.log)
1125         Completed: Not completed(device6_intouch.log)
1126         Elapsed Time:
1127     ConfigurationService
1128         Start: Not started(device6_intouch.log)
1129         Completed: Not completed(device6_intouch.log)
1130         Elapsed Time:
1131     LandingPadService
1132         Start: Not started(device6_intouch.log)
1133         Completed: Not completed(device6_intouch.log)
1134         Elapsed Time:
1135     PortConfigurationService
1136         Start: Not started(device6_intouch.log)
1137         Completed: Not completed(device6_intouch.log)
1138         Elapsed Time:
1139     CacheService
1140         Start: Not started(device6_intouch.log)
1141         Completed: Not completed(device6_intouch.log)
1142         Elapsed Time:
1143     ThemingService
1144         Start: Not started(device6_intouch.log)
1145         Completed: Not completed(device6_intouch.log)
1146         Elapsed Time:
1147     StagingService
1148         Start: Not started(device6_intouch.log)
1149         Completed: Not completed(device6_intouch.log)
1150         Elapsed Time:
1151     DeviceIOService
1152         Start: Not started(device6_intouch.log)
1153         Completed: Not completed(device6_intouch.log)
1154         Elapsed Time:
1155     BellService
1156         Start: Not started(device6_intouch.log)
1157         Completed: Not completed(device6_intouch.log)
1158         Elapsed Time:

```



```

1159 GateService
1160     Start: Not started(device6_intouch.log)
1161     Completed: Not completed(device6_intouch.log)
1162     Elapsed Time:
1163 ReaderDataService
1164     Start: Not started(device6_intouch.log)
1165     Completed: Not completed(device6_intouch.log)
1166     Elapsed Time:
1167 BiometricService
1168     Start: Not started(device6_intouch.log)
1169     Completed: Not completed(device6_intouch.log)
1170     Elapsed Time:
1171 StateManager
1172     Start: Not started(device6_intouch.log)
1173     Completed: Not completed(device6_intouch.log)
1174     Elapsed Time:
1175 OfflineSmartviewService
1176     Start: Not started(device6_intouch.log)
1177     Completed: Not completed(device6_intouch.log)
1178     Elapsed Time:
1179 AVFeedbackService
1180     Start: Not started(device6_intouch.log)
1181     Completed: Not completed(device6_intouch.log)
1182     Elapsed Time:
1183 DatabaseThreads
1184     Start: Not started(device6_intouch.log)
1185     Completed: Not completed(device6_intouch.log)
1186     Elapsed Time:
1187 SoftLoadService
1188     Start: Not started(device6_intouch.log)
1189     Completed: Not completed(device6_intouch.log)
1190     Elapsed Time:
1191 WATCHDOG
1192     Start: Not started(device6_intouch.log)
1193     Completed: Not completed(device6_intouch.log)
1194     Elapsed Time:
1195 ProtocolService
1196     Start: Not started(device6_intouch.log)
1197     Completed: Not completed(device6_intouch.log)
1198     Elapsed Time:
1199 DiagnosticsService
1200     Start: Not started(device6_intouch.log)
1201     Completed: Not completed(device6_intouch.log)
1202     Elapsed Time:
1203
1204 *** Services not successfully started: Logging, DatabaseInitialize,
    MessagingService, HealthMonitorService, Persistence,
    ConfigurationService, LandingPadService, PortConfigurationService,
    CacheService, ThemingService, StagingService, DeviceIOService,
    BellService, GateService, ReaderDataService, BiometricService,
    StateManager, OfflineSmartviewService, AVFeedbackService,
    DatabaseThreads, SoftLoadService, WATCHDOG, ProtocolService,
    DiagnosticsService
1205
1206
1207 === Device boot ===
1208 90112(device6_intouch.log): 2014-04-14 12:13:59 Boot Start
1209 **** Incomplete boot ****
1210

```

```

1211
1212 === Device boot ===
1213 90135(device6_intouch.log): 2014-04-14 12:16:13 Boot Start
1214 **** Incomplete boot ****
1215
1216 Services
1217     Logging
1218         Start: Not started(device6_intouch.log)
1219         Completed: Not completed(device6_intouch.log)
1220         Elapsed Time:
1221     DatabaseInitialize
1222         Start: Not started(device6_intouch.log)
1223         Completed: Not completed(device6_intouch.log)
1224         Elapsed Time:
1225     MessagingService
1226         Start: Not started(device6_intouch.log)
1227         Completed: Not completed(device6_intouch.log)
1228         Elapsed Time:
1229     HealthMonitorService
1230         Start: Not started(device6_intouch.log)
1231         Completed: Not completed(device6_intouch.log)
1232         Elapsed Time:
1233     Persistence
1234         Start: Not started(device6_intouch.log)
1235         Completed: Not completed(device6_intouch.log)
1236         Elapsed Time:
1237     ConfigurationService
1238         Start: Not started(device6_intouch.log)
1239         Completed: Not completed(device6_intouch.log)
1240         Elapsed Time:
1241     LandingPadService
1242         Start: Not started(device6_intouch.log)
1243         Completed: Not completed(device6_intouch.log)
1244         Elapsed Time:
1245     PortConfigurationService
1246         Start: Not started(device6_intouch.log)
1247         Completed: Not completed(device6_intouch.log)
1248         Elapsed Time:
1249     CacheService
1250         Start: Not started(device6_intouch.log)
1251         Completed: Not completed(device6_intouch.log)
1252         Elapsed Time:
1253     ThemingService
1254         Start: Not started(device6_intouch.log)
1255         Completed: Not completed(device6_intouch.log)
1256         Elapsed Time:
1257     StagingService
1258         Start: Not started(device6_intouch.log)
1259         Completed: Not completed(device6_intouch.log)
1260         Elapsed Time:
1261     DeviceIOService
1262         Start: Not started(device6_intouch.log)
1263         Completed: Not completed(device6_intouch.log)
1264         Elapsed Time:
1265     BellService
1266         Start: Not started(device6_intouch.log)
1267         Completed: Not completed(device6_intouch.log)
1268         Elapsed Time:
1269     GateService

```

```

1270         Start: Not started(device6_intouch.log)
1271         Completed: Not completed(device6_intouch.log)
1272         Elapsed Time:
1273     ReaderDataService
1274         Start: Not started(device6_intouch.log)
1275         Completed: Not completed(device6_intouch.log)
1276         Elapsed Time:
1277     BiometricService
1278         Start: Not started(device6_intouch.log)
1279         Completed: Not completed(device6_intouch.log)
1280         Elapsed Time:
1281     StateManager
1282         Start: Not started(device6_intouch.log)
1283         Completed: Not completed(device6_intouch.log)
1284         Elapsed Time:
1285     OfflineSmartviewService
1286         Start: Not started(device6_intouch.log)
1287         Completed: Not completed(device6_intouch.log)
1288         Elapsed Time:
1289     AVFeedbackService
1290         Start: Not started(device6_intouch.log)
1291         Completed: Not completed(device6_intouch.log)
1292         Elapsed Time:
1293     DatabaseThreads
1294         Start: Not started(device6_intouch.log)
1295         Completed: Not completed(device6_intouch.log)
1296         Elapsed Time:
1297     SoftLoadService
1298         Start: Not started(device6_intouch.log)
1299         Completed: Not completed(device6_intouch.log)
1300         Elapsed Time:
1301     WATCHDOG
1302         Start: Not started(device6_intouch.log)
1303         Completed: Not completed(device6_intouch.log)
1304         Elapsed Time:
1305     ProtocolService
1306         Start: Not started(device6_intouch.log)
1307         Completed: Not completed(device6_intouch.log)
1308         Elapsed Time:
1309     DiagnosticsService
1310         Start: Not started(device6_intouch.log)
1311         Completed: Not completed(device6_intouch.log)
1312         Elapsed Time:
1313
1314 *** Services not successfully started: Logging, DatabaseInitialize,
    MessagingService, HealthMonitorService, Persistence,
    ConfigurationService, LandingPadService, PortConfigurationService,
    CacheService, ThemingService, StagingService, DeviceIOService,
    BellService, GateService, ReaderDataService, BiometricService,
    StateManager, OfflineSmartviewService, AVFeedbackService,
    DatabaseThreads, SoftLoadService, WATCHDOG, ProtocolService,
    DiagnosticsService
1315
1316
1317 === Device boot ===
1318 90112(device6_intouch.log): 2014-04-14 12:13:59 Boot Start
1319 **** Incomplete boot ****
1320
1321

```

```

1322 === Device boot ===
1323 90135(device6_intouch.log): 2014-04-14 12:16:13 Boot Start
1324 **** Incomplete boot ****
1325
1326
1327 === Device boot ===
1328 90176(device6_intouch.log): 2014-04-14 12:18:44 Boot Start
1329 90307(device6_intouch.log): 2014-04-14 12:21:25 Boot Completed
1330     Boot Time: 161000ms
1331
1332 Services
1333     ThemingService
1334         ThemingServiceStart:90268(device6_intouch.log)
1335         Completed:90269(device6_intouch.log)
1336         ElapsedTime:0
1337     PortConfigurationService
1338         PortConfigurationServiceStart:90266(device6_intouch.log)
1339         Completed:90267(device6_intouch.log)
1340         ElapsedTime:65
1341     OfflineSmartviewService
1342         OfflineSmartviewServiceStart:90262(device6_intouch.log)
1343         Completed:90263(device6_intouch.log)
1344         ElapsedTime:16
1345     BiometricService
1346         BiometricServiceStart:90258(device6_intouch.log)
1347         Completed:90259(device6_intouch.log)
1348         ElapsedTime:666
1349     DatabaseInitialize
1350         DatabaseInitializeStart:90198(device6_intouch.log)
1351         Completed:90224(device6_intouch.log)
1352         ElapsedTime:33067
1353     MessagingService
1354         MessagingServiceStart:90225(device6_intouch.log)
1355         Completed:90227(device6_intouch.log)
1356         ElapsedTime:3825
1357     DatabaseThreads
1358
1359     DiagnosticsServiceProtocolServiceSoftLoadServiceWATCHDOGDiagnosticsServiceStart
1360         :90296(device6_intouch.log)
1361         Start:90296(device6_intouch.log)
1362         Completed:90303(device6_intouch.log)
1363         ElapsedTime:3564
1364     ConfigurationService
1365         ConfigurationServiceStart:90242(device6_intouch.log)
1366         Completed:90243(device6_intouch.log)
1367         ElapsedTime:0
1368     CacheService
1369         CacheServiceStart:90244(device6_intouch.log)
1370         Completed:90245(device6_intouch.log)
1371         ElapsedTime:981
1372     StateManager
1373         StateManagerStart:90260(device6_intouch.log)
1374         Completed:90261(device6_intouch.log)
1375         ElapsedTime:2657
1376     AVFeedbackService
1377         AVFeedbackServiceStart:90250(device6_intouch.log)
1378         Completed:90251(device6_intouch.log)
1379         ElapsedTime:49
1380     StagingService

```

```

1379     StagingServiceStart:90254(device6_intouch.log)
1380     Completed:90255(device6_intouch.log)
1381     ElapsedTime:7947
1382     WATCHDOG
1383     Start:90296(device6_intouch.log)
1384     Completed:90298(device6_intouch.log)
1385     ElapsedTime:557
1386     DiagnosticsService
1387     Completed:90297(device6_intouch.log)
1388     ElapsedTime:192
1389     HealthMonitorService
1390     HealthMonitorServiceStart:90238(device6_intouch.log)
1391     Completed:90239(device6_intouch.log)
1392     ElapsedTime:205
1393     BellService
1394     BellServiceStart:90264(device6_intouch.log)
1395     Completed:90265(device6_intouch.log)
1396     ElapsedTime:2
1397     Persistence
1398     PersistenceStart:90240(device6_intouch.log)
1399     Completed:90241(device6_intouch.log)
1400     ElapsedTime:18401
1401     DeviceIOService
1402     DeviceIOServiceStart:90246(device6_intouch.log)
1403     Completed:90247(device6_intouch.log)
1404     ElapsedTime:27
1405     SoftLoadService
1406     Start:90296(device6_intouch.log)
1407     Completed:90302(device6_intouch.log)
1408     ElapsedTime:3501
1409     GateService
1410     GateServiceStart:90248(device6_intouch.log)
1411     Completed:90249(device6_intouch.log)
1412     ElapsedTime:164
1413     LandingPadService
1414     LandingPadServiceStart:90252(device6_intouch.log)
1415     Completed:90253(device6_intouch.log)
1416     ElapsedTime:2
1417     Logging
1418     LoggingStart:90196(device6_intouch.log)
1419     Completed:90197(device6_intouch.log)
1420     ElapsedTime:289
1421     ReaderDataService
1422     ReaderDataServiceStart:90256(device6_intouch.log)
1423     Completed:90257(device6_intouch.log)
1424     ElapsedTime:2

```

8.5 Codebase

```

1 CC = g++
2 CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
3 LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
    lboost_unit_test_framework -lboost_date_time
4 # Your .hpp files
5 DEPS = Kronos.hpp
6 # Your compiled .o files
7 OBJECTS = Kronos.o
8 # The name of your program
9 PROGRAM = ps7

```

```

10
11 .PHONY: all clean lint
12
13 all: $(PROGRAM) Kronos.a
14
15 # Wildcard recipe to make .o files from corresponding .cpp file
16 %.o: %.cpp $(DEPS)
17     $(CC) $(CFLAGS) -c $<
18
19 $(PROGRAM): main.o $(OBJECTS)
20     $(CC) $(CFLAGS) -o $@ $^ $(LIB)
21
22 Kronos.a: Kronos.o
23     ar rcs Kronos.a Kronos.o
24
25 clean:
26     rm *.o $(PROGRAM) *.a
27
28 lint:
29     cpplint *.cpp *.hpp

```

```

1 // Copyright Camden Andersson 2024
2
3 #include <iostream>
4 #include <string>
5 #include <regex>
6 #include "Kronos.hpp"
7
8 int main(int argc, char* argv[]) {
9     std::string filename = argv[1];
10    kronos k(filename);
11    k.makeBootupLog();
12    // k.printBootupLog();
13    k.makeSummaryLog();
14    k.printSummaryLog();
15
16    return 0;
17 }

```

```

1 // Copyright Camden Andersson 2024
2 #pragma once
3
4 #ifndef KRONOS_HPP
5 #define KRONOS_HPP
6
7 #include <iostream>
8 #include <string>
9 #include <regex>
10 #include <boost/date_time.hpp>
11
12 class kronos {
13 public:
14     explicit kronos(const std::string& filename);
15     void makeBootupLog();
16     void makeSummaryLog();
17     void printBootupLog();
18     void printSummaryLog();
19     std::string cleanBonus(const std::string& input);
20     std::string createServiceStatusString(const std::string& filename);
21 private:

```



```

22     int numLinesScanned;    // the number of lines read in "filename"
23     int numInitiated;      // the number of services initiated
24     int numSuccess;        // the number of successful services running
25     std::string logFilename; // the filename of the log to be read
26     std::string logText;    // the text from "filename"
27     std::string logBoot;    // just the boot text
28     std::string logFull;    // the full text for the bonus portion
29 };
30
31 #endif /* KRONOS_HPP */

```

```

1  // Copyright Camden Andersson 2024
2  #include "Kronos.hpp"
3  #include <unordered_map>
4
5  // This function reads the file and returns it as text
6  std::string read_file_to_string(const std::string& filename) {
7      std::ifstream file(filename);
8      if (!file.is_open()) {
9          std::cerr << "Failed to open file: " << filename << std::endl;
10         return "";
11     }
12     std::string file_contents;
13     file.seekg(0, std::ios::end);
14     file_contents.reserve(file.tellg());
15     file.seekg(0, std::ios::beg);
16
17     file_contents.assign((std::istreambuf_iterator<char>(file)),
18                         std::istreambuf_iterator<char>());
19
20     return file_contents;
21 }
22
23 void write_string_to_file(const std::string& str, const std::string&
    filename) {
24     std::ofstream file(filename);
25     if (file.is_open()) {
26         file << str;
27         file.close();
28         std::cout << "String written to file successfully." << std::endl;
29     } else {
30         std::cerr << "Unable to open file." << std::endl;
31     }
32 }
33
34
35 // Constructor
36 kronos::kronos(const std::string& filename) {
37     logFilename = filename;
38     // Open the file that contains the text & display it
39     std::string rawText = read_file_to_string(filename);
40     if (!rawText.empty()) {
41         logText = rawText;
42     } else {
43         std::cout << "File not found.\n" << std::endl;
44         logText = "";
45     }
46     // Finish initializing the all the class members
47     numLinesScanned = 0;
48     numInitiated = 0;

```



```

49     numSuccess = 0;
50     logBoot = "";
51     logFull = "";
52 }
53
54 void kronos::makeBootupLog() {
55     std::string logShort = "";
56     std::string logLong = "";
57     std::string bootStartTime;
58     std::string bootEndTime;
59     std::regex startRegex("\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}: "
60     ".server started");
61     std::regex endRegex("\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}: "
62     ".AbstractConnector:Started .*");
63     std::smatch match;
64
65     std::string line = "";
66     int lineNumStart = 0;
67     int lineNumEnd = 0;
68     int lineNumber = 0;
69     bool bootStarted = false;
70     std::string bonus;
71     std::istringstream iss(logText);
72     while (std::getline(iss, line)) {
73         lineNumber++;
74
75         // Look for the START message
76         if (std::regex_search(line, match, startRegex)) {
77             // See if this is a failure
78             if (!bootStarted) {
79                 bootStarted = true;
80             } else {
81                 // bootStarted was true, hence this is an error
82                 logShort += "**** Incomplete boot ****\n\n";
83                 std::string sError = kronos::createServiceStatusString(
logFilename);
84                 logFull += logShort + sError;
85                 bonus = "";
86             }
87
88             // Get the date and time
89             std::regex pattern(R"((\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}))");
90             std::smatch matches;
91             if (std::regex_search(line, matches, pattern)) {
92                 // Print the matched date and time
93                 bootStartTime = matches[1];
94             }
95             numInitiated++;
96
97             // Build the log message
98             logShort += "\n=== Device boot ===\n";
99             logShort += std::to_string(lineNumber) +
100             "(" + logFilename + "): " + bootStartTime + " Boot Start\n";
101         }
102
103         //////////////////////////////////////
104         // BONUS - log other services
105         // Start Services
106         if (bootStarted) {

```

```

107         std::string service;
108         std::regex patternService("Starting Service\\.\\.s*(\\w+) (\\d
+\\.\\.d+)");
109         std::smatch matchService;
110         if (std::regex_search(line, matchService, patternService)) {
111             if (matchService.size() > 2) {
112                 service = matchService[1];
113                 bonus += "\t" + service;
114                 lineNumberStart = lineNumber;
115             }
116         }
117
118         // Complete service
119         std::regex expr("Service\\s+started\\s+successfully\\.\\.s+([A-Za
-z0-9]+)"
120             "\\s+([A-Za-z0-9.]+)\\s+\\((([0-9]+)\\s+ms\\)");
121         std::smatch match;
122         std::string sMs;
123         std::string sService;
124         if (std::regex_match(line, match, expr)) {
125             sService = match[1]; // TO DO: check sService!!
126             sMs = match[3];
127             lineNumberEnd = lineNumber;
128             bonus += "\t\t" + sService +
129                 " - Start: " + std::to_string(lineNumStart) + "(" +
logFilename + ")\n";
130             bonus += "\t\t" + sService +
131                 " - Completed: " + std::to_string(lineNumEnd) + "(" +
logFilename + ")\n";
132             bonus += "\t\t" + sService + " - Elapsed Time: " + sMs + "\n
";
133         }
134     }
135     //////////////////////////////////////
136
137     // Look for the END message
138     if (std::regex_search(line, match, endRegex)) {
139         std::string s;
140         // Get the date and time
141         std::regex pattern(R"((\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}))");
142         std::smatch matches;
143         if (std::regex_search(line, matches, pattern)) {
144             // Print the matched date and time
145             bootEndTime = matches[1];
146         }
147         s = std::to_string(lineNumber) +
148             "(" + logFilename + "): " + bootEndTime + " Boot Completed\n";
149         logShort += s;
150         numSuccess++;
151
152         // Calculate elapsed time
153         boost::posix_time::ptime startTime = boost::posix_time::
time_from_string(bootStartTime);
154         boost::posix_time::ptime endTime = boost::posix_time::
time_from_string(bootEndTime);
155         boost::posix_time::time_duration diff = endTime - startTime;
156         s = "\tBoot Time: " + std::to_string(diff.total_milliseconds())
+ "ms\n\n";
157         logShort += s;

```

```

158
159         // Put logShort into the logBoot result
160         logBoot += logShort;
161
162         // You've found the COMPLETION
163         std::string cleanedBonus = cleanBonus(bonus);
164         logFull += logShort + "Services\n" + cleanedBonus;
165         bonus = ""; // clear it
166         logShort = ""; // clear it
167         // logLong += logShort + bonus;
168         bootStarted = false;
169     }
170 }
171
172 numLinesScanned = lineNumber;
173 }
174
175 std::string kronos::cleanBonus(const std::string& input) {
176     std::istringstream iss(input);
177     std::unordered_map<std::string, std::vector<std::string>> groupedLines;
178
179     // Parse the input and group lines by the first word
180     std::string line;
181     while (std::getline(iss, line)) {
182         // Trim leading tabs and spaces
183         line.erase(line.begin(), std::find_if(line.begin(), line.end(), [](
184         unsigned char ch) {
185             return !std::isspace(ch);
186         }));
187
188         std::istringstream lineStream(line);
189         std::string firstWord;
190         lineStream >> firstWord;
191
192         // Remove the first word and the '-' if present
193         line = line.substr(firstWord.length());
194         line.erase(std::remove_if(line.begin(), line.end(), [](unsigned char
195         ch) {
196             return std::isspace(ch) || ch == '-';
197         }), line.end());
198
199         // Store the line in the appropriate group
200         groupedLines[firstWord].push_back(line);
201     }
202
203     // Format the output
204     std::ostringstream oss;
205     for (const auto& pair : groupedLines) {
206         oss << "\t" << pair.first << std::endl;
207         for (const auto& line : pair.second) {
208             oss << "\t\t" << line << std::endl;
209         }
210         // oss << std::endl; // Add a blank line between groups
211     }
212
213     return oss.str();
214 }
215
216 std::string kronos::createServiceStatusString(const std::string & filename)

```

```

{
215     std::ostringstream oss;
216     oss << "Services\n";
217     std::string services[] = {
218         "Logging", "DatabaseInitialize", "MessagingService", "
HealthMonitorService",
219         "Persistence", "ConfigurationService", "LandingPadService", "
PortConfigurationService",
220         "CacheService", "ThemingService", "StagingService", "
DeviceIOService", "BellService",
221         "GateService", "ReaderDataService", "BiometricService",
222         "StateManager", "OfflineSmartviewService",
223         "AVFeedbackService",
224         "DatabaseThreads", "SoftLoadService",
225         "WATCHDOG", "ProtocolService", "DiagnosticsService"
226     };
227
228     for (const auto& service : services) {
229         oss << "\t" << service << "\n";
230         oss << "\t\t Start: Not started(" << filename << ")\n";
231         oss << "\t\t Completed: Not completed(" << filename << ")\n";
232         oss << "\t\t Elapsed Time: \n";
233     }
234
235     oss << "\n*** Services not successfully started: Logging, "
236         "DatabaseInitialize, MessagingService, HealthMonitorService, "
237         "Persistence, ConfigurationService, LandingPadService, "
238         "PortConfigurationService, CacheService, ThemingService, "
239         "StagingService, DeviceIOService, BellService, GateService, "
240         "ReaderDataService, BiometricService, StateManager,
OfflineSmartviewService, "
241         "AVFeedbackService, DatabaseThreads, SoftLoadService, WATCHDOG,
ProtocolService, "
242         "DiagnosticsService\n\n";
243
244     std::string result = oss.str();
245     size_t pos = result.find("device5_intouch.log");
246     if (pos != std::string::npos) {
247         result.replace(pos, std::strlen("device5_intouch.log"), filename
);
248     }
249
250     return result;
251 }
252
253 void kronos::makeSummaryLog() {
254     // Start the header, for the bonus points
255     std::string sHeader;
256     sHeader = "Device Boot Report\n\n";
257     sHeader += "InTouch log file: " + logFilename + "\n";
258     sHeader += "Lines Scanned: " + std::to_string(numLinesScanned) + "\n\n";
259     sHeader += "Device boot count: initiated = "
260     + std::to_string(numInitiated) + ", completed: " + std::to_string(
numSuccess);
261
262     // Make the full log
263     logFull = sHeader + "\n\n" + logFull;
264 }
265

```

```
266 void kronos::printBootupLog() {
267     // std::cout << logBoot << std::endl;
268
269     std::string filename = logFilename + ".rpt";
270     write_string_to_file(logBoot, filename);
271 }
272
273 void kronos::printSummaryLog() {
274     // std::cout << logFull << std::endl;
275
276     std::string filename = logFilename + ".rpt";
277     write_string_to_file(logFull, filename);
278 }
```