Yoo Min Cha
91.204.202 Project Portfolio
May 6, 2013

Contents:

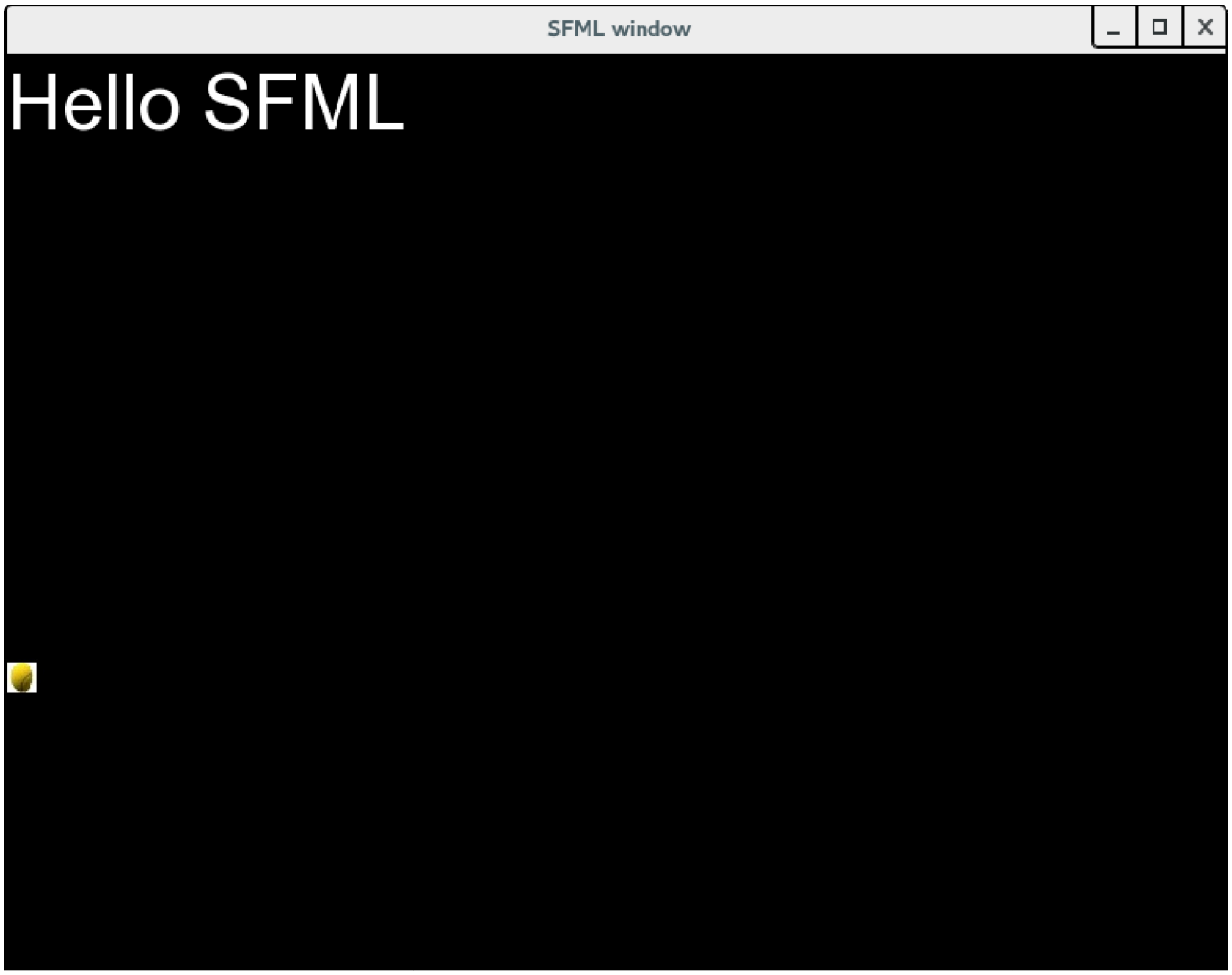
PS0 Hello World

For the first part of this assignment, I copied the code from the given link to create the big green sprite. The second part was more involved and required me to create a texture from a jpg file and then use this texture to paint the sprite. The sprite is not in the window and I can move it around from side to side or up and down.

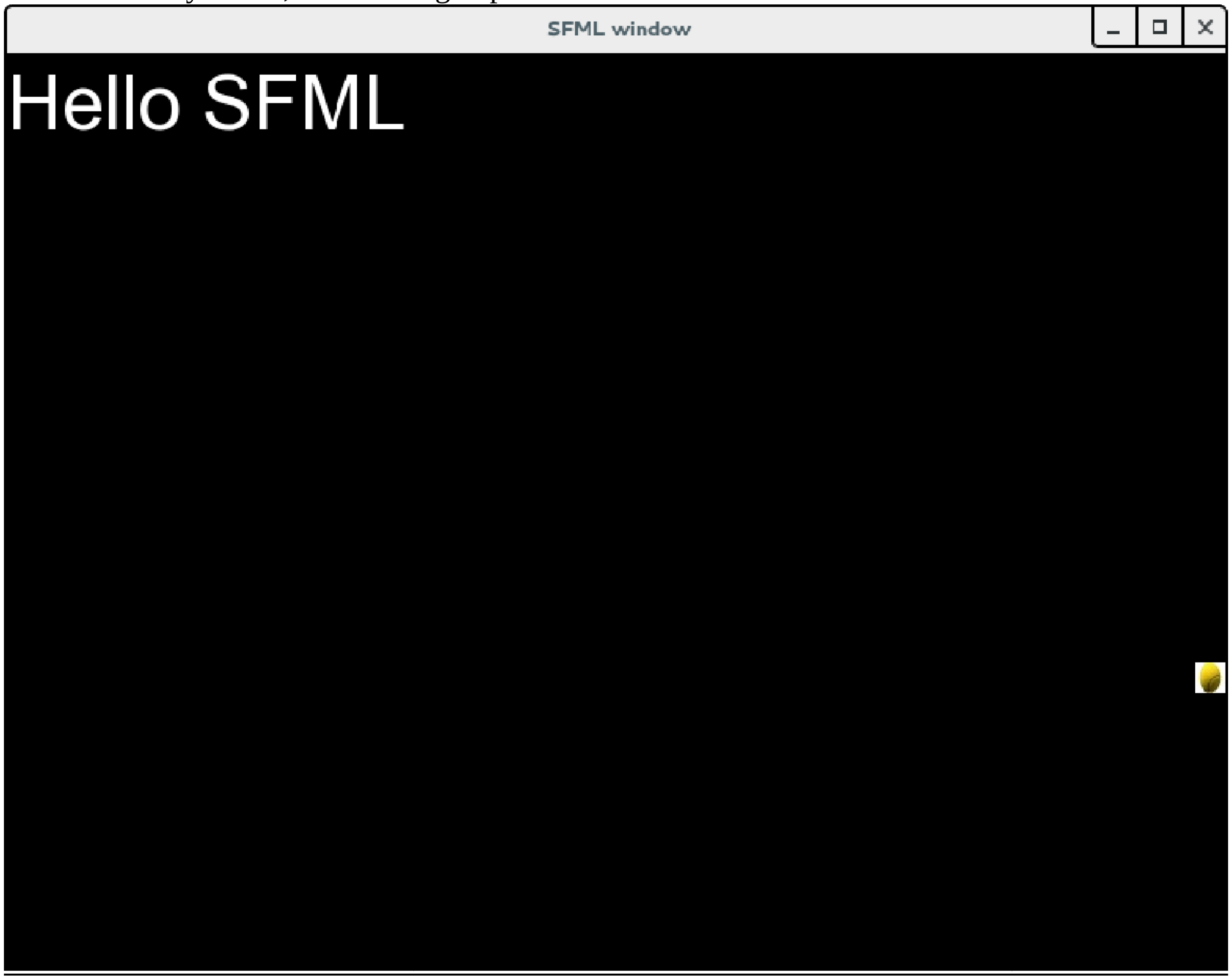
Design aspects I implemented in this assignment involved creating a window with a pretty black background and giant green sprite in the middle. Another thing I did was to create a giant Text inside the window that read out whatever message I wanted to insert there. I use a picture of a ball and used this image because it represented the object that was moving around in the program.

From this assignment, I learned how to utilize the tools in SFML to create a window, texture, sprite, event loop, keypressed events, and how to get them to do different things. I learned how to clear and draw the sprite for each event loop. I learned about SFML coordinates and using them as boundaries for which my ball was allowed to move in.

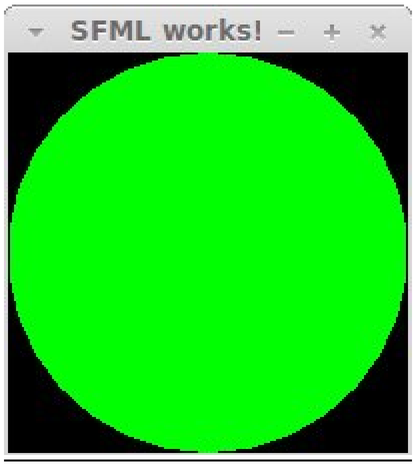
Ball starting position



Ball all the way to left, it can also go up or down.



Hello SFML



<u>PS1 N-Body Simulation</u>

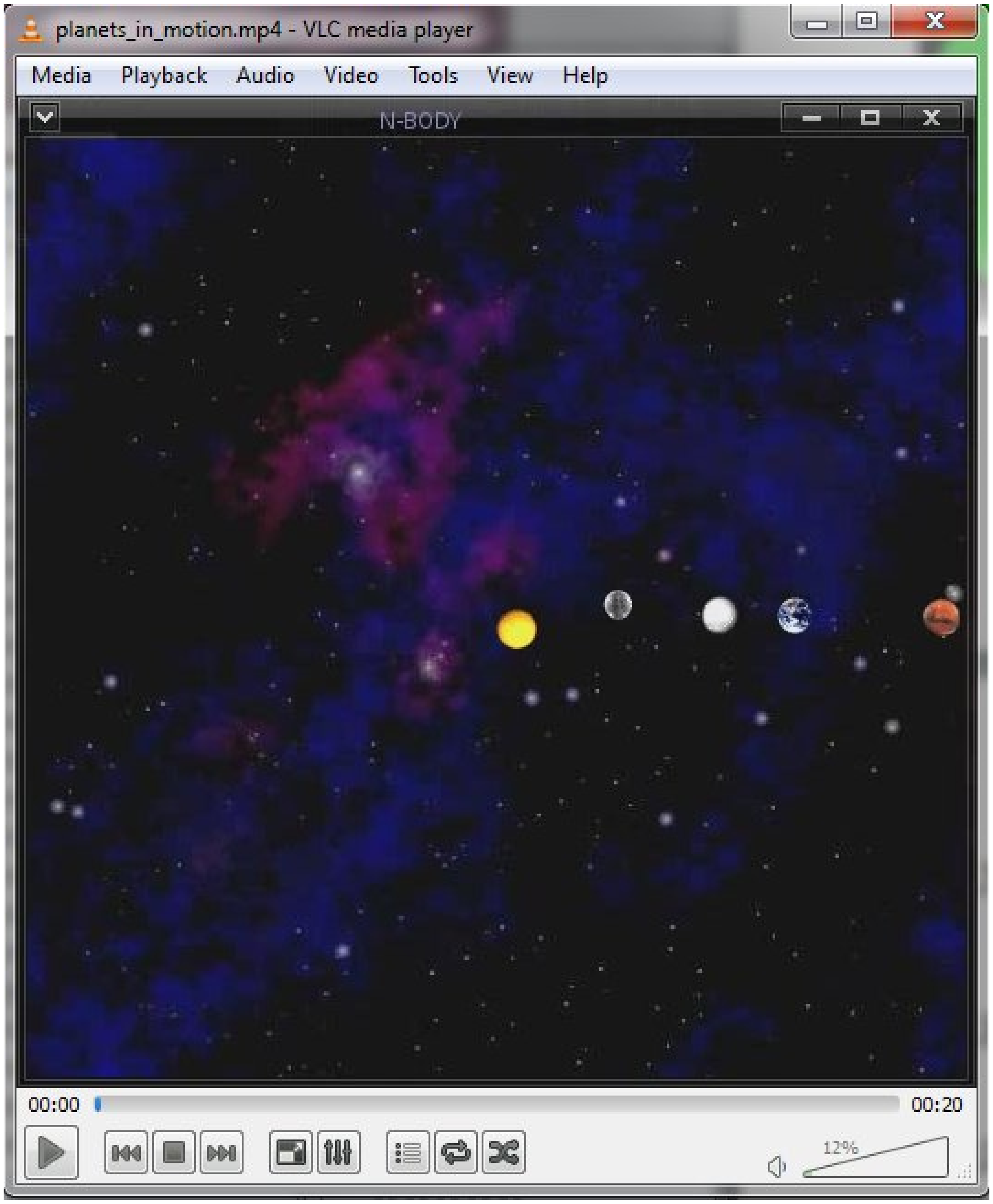
In this assignment, we had to create a solar system simulation by using the given images and data from the planets.txt file. We were also given the leapfrog formula to instill into our SolarSystem class so that it would calculate the position of each body by using force, acceleration, mass, gravity, and distance.

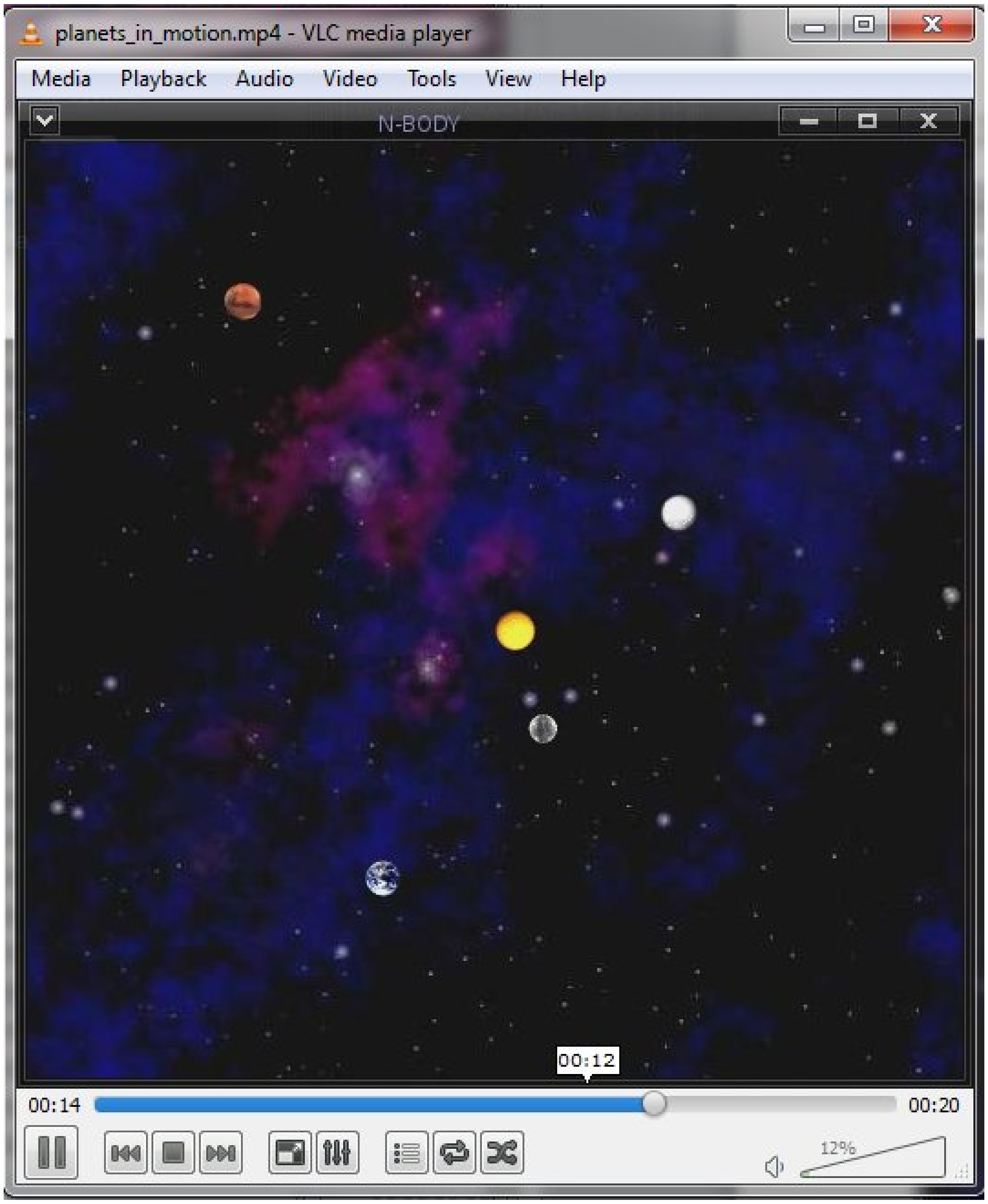
There were many factors to take into consideration when getting this simulation to work. We had to create a 2D matrix that contained all the forces for each of the bodies. In this matrix we had to use the leapfrog to obtain all the forces for each body.

```
        for (int i = 0; i < numPlanets; i++){
                        // add all the forces for matrix i
                pair<double,double> force(0.0, 0.0);
                for (int k = 0; k < numPlanets-1; k++)
                {
                                //cout<<force.x<<"_"<<forces[i][k].x<<endl;
                        force.x += forces[i][k].x;
                                //cout<<force.y<<"."<<forces[i][k].y<<endl;
                        force.y += forces[i][k].y;
                }
                    // update acceleration
                pair<double,double> a((force.x / stuff[i].getMass()), (force.y /
stuff[i].getMass()));
                    // update velocity
                pair<double,double> v(stuff[i].getVelocity());
                v.x += dT * a.x;
                v.y += dT * a.y;
                stuff[i].setVelocity(v.x, v.y);
                    // update positions using the new x and y components of velocity
                pair<double,double> p(stuff[i].getPosition());
                p.x += dT * v.x;
                p.y += dT * v.y;
                stuff[i].setPosition(p.x, p.y);
        }
}
```

I learned that you had to make the y position negative in the constructor in order to planets to correctly align vertically. I also had a hard time getting the textures to remain statically in the stack so I had to create a map that would dynamically allocate a node for each planet in the heap, map is implemented as balanced tree and can achieve logarithmic lookup, insertion, and deletion times. The nodes are referenced by planet name and the associating .jpg texture. The sprite would have a constant reference from now and all I had to do was update the positions each event loop.

For testing purpose I created a pause flag that would stop the leapfrog/position updates whenever I press P. I also had to comment out the leapfrog/position update in the main event loop and move it to the pollEvent loop so that it would only move run the simulation when the main window was active(such as pointing to the SFML window). This allowed me to pause the simulation and output the forces to standard output. The hardest part of this assignment was the initial process of creating the planet sprites and background because Windows is extremely intolerant and working was visual studios was quite the hassle. I had to install linux on my own machine and had to learn how to use linux in order smoothly compile my program. I also had issues with installing SFML the first time around but now I am better adept at it.


planets_in_motion.mp4 - VLC media player
Media
Playback
Audio
Video
Tools
View
Help
N-BODY
00:00
00:20
12%

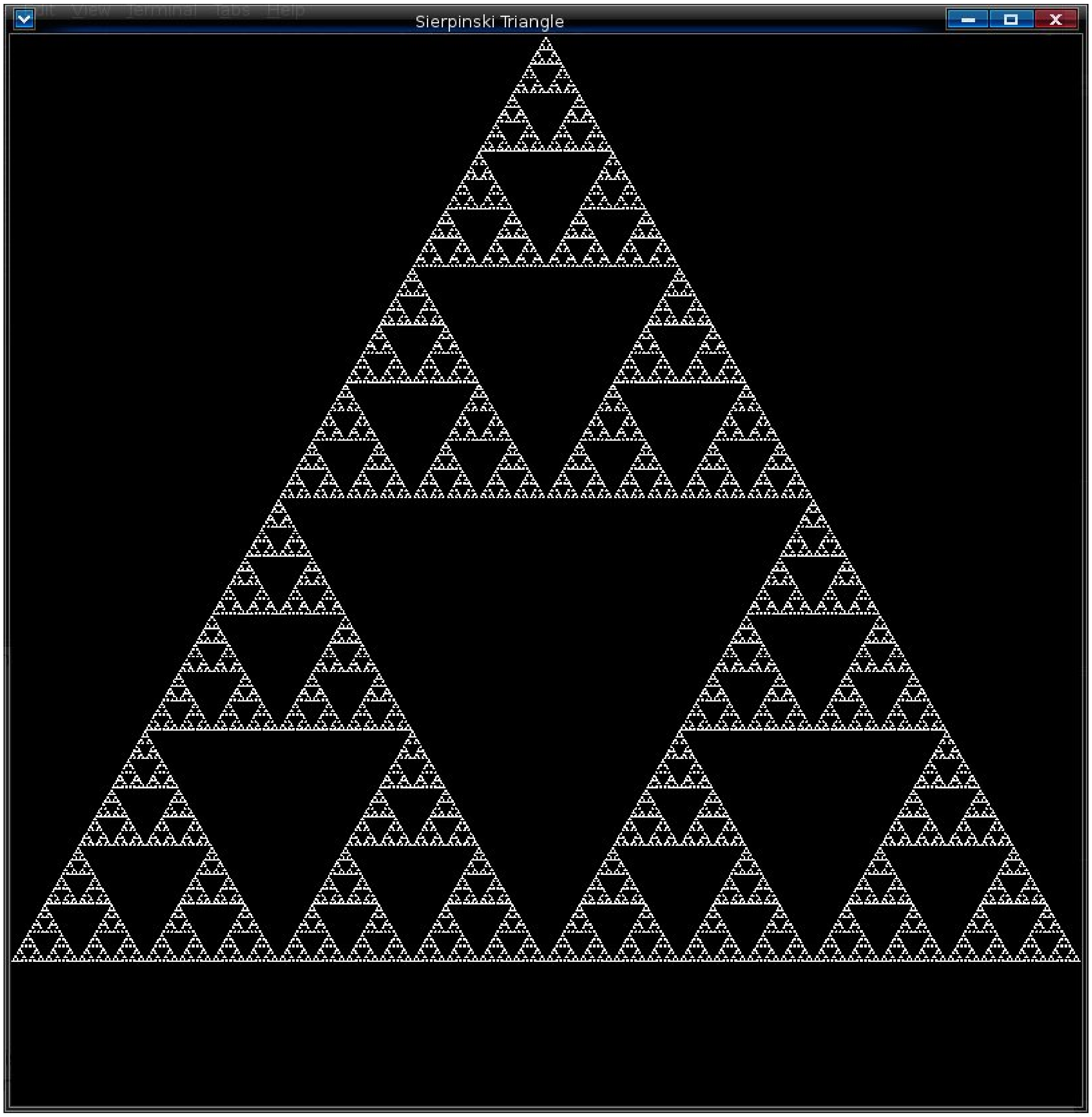

planets_in_motion.mp4 - VLC media player
Media
Playback
Audio
Video
Tools
View
Help
N-BODY
00:12
00:14
00:20
12%

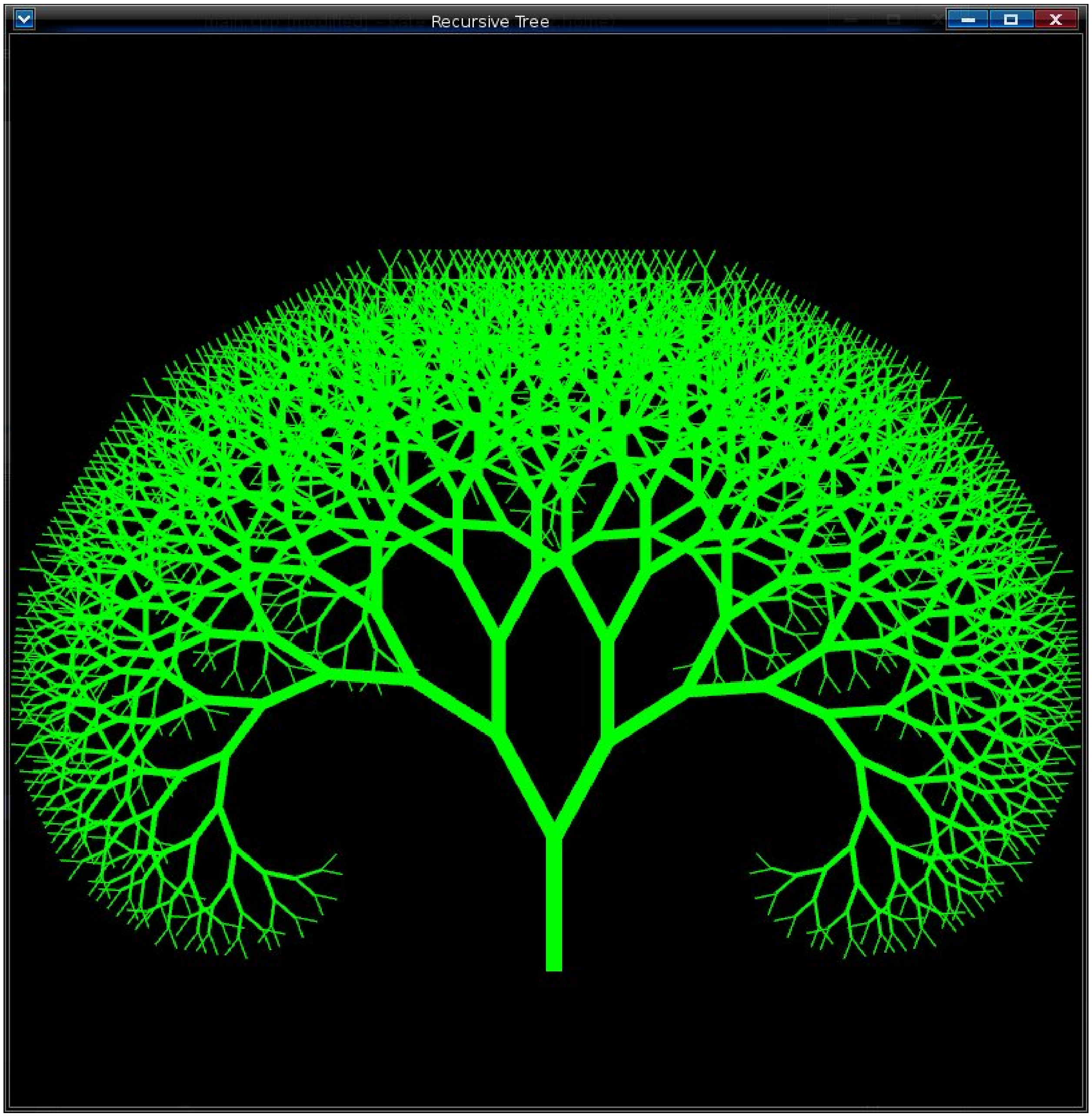
PS2 Recursive Graphics

This assignment required us to recursively create triangles within a triangle to create the Droste effect (mise en abyme). This first part of the assignment was to create a fractal pattern called Sierpinski's triangle, designed by the Polish mathematician Waclaw Sierpinski in 1915. The second part required using the same methodology to create a design of our own. I created the triangle getting the half way points of each triangle both vertically and horizontally and then creating another triangle inside that triangle. For my other design, I created a tree that branches out a given amount equivalent to the depth n.

In order to get the height of a triangle, I used the sin of pi/3 which is sqrt(3)/2*radius. The width was simply the radius or the bottom length of the triangle. The recursion was simple, I pass an integer n that would decrement each time the recursion was called until it would hit zero. Inside the recursion I would cut the delta x and delta y by half and then set these as the points, fill in the color and then push the triangle onto a global vector of pointers to Convexshapes. For my own design, I used the same driver as Sierpinski's program but changed a few things in the recursion function. I only had one length to worry about so I cut each length by 15% each recursive call and then I changed the angle by tilting the angle .5 radians for each opposing side. I converted the radians to degree to construct the line angle and I adjusted the width based on the depth of the recursion.

In this assignment, I learned how to use recursion to create art and also how it is also naturally occurring in nature. I also used antialiasing in my program to make the lines look cleaner.


Sierpinski Triangle



PS3 DNA Sequence Alignment

This program required taking in two strings and matching the two strings with the minimal edit distance. I created a EditDistance class that would take the two strings and compare them and output the cost to edit the two strings.

My approach to this assignment was to create a 1x1 dimensional array with the first and second strings as the x and y of the matrix. I used the initializer list in my constructor to create my matrix : `CostMatrix(a.size()+1, vector<int> (b.size()+1, 0))` to fill in the matrix with all zeroes. I then fill in the last row and columns by incrementing each step up from the bottom right most cell by 2. Then I allocated each of the cells with the costs by comparing the bottom, right, and bottom right cells with the originating cell.

```
CostMatrix[j][i] =
min((CostMatrix[j+1][i+1]+b),(CostMatrix[j+1][i]+2),(CostMatrix[j][i+1]+2));
```

After the matrix is complete, I trace my way up to [0][0] from the [x.size+1][y.size+1] cell. I do this by first testing my location by making sure the index is not the right most column or the bottom column first. I then proceed to get the difference from the adjacent cells to find out where my cell came from.

```
  while ((i!=Y.size()) || (j!=X.size()))
       {
        if ((i<Y.size()) && (CostMatrix[j][i] - CostMatrix[j][i+1] == 2))
              {
            output = output + '-' + ' ' + Y[i] + ' ' + '2' + '\n';
            i++;
        }
        else if ((j<X.size()) && (CostMatrix[j][i] - CostMatrix[j+1][i] == 2))
              {
            output = output + X[j] + ' ' + '-' + ' ' + '2' + '\n';
            j++;
        }
        else
              {
                     output = output + X[j] + ' ' + Y[i] + ' ' +
char('0'+penalty(X[j],Y[i])) + '\n';
                     i++;
                     j++;
        }
```

From this assignment, I learned how to match up two strings with utmost efficiency by using a scalar matrix.

```
Input: atattat
tattata


Expected output:
Edit distance = 4
----------------
a - 2
t t 0
a a 0
t t 0
t t 0
a a 0
t t 0
- a 2

Execution time is 0.066961seconds.
```

PS4a Linear Feedback Shift Register

This assignment required us to learned how to use a key bit in a string to encrypt itself. It also required us to use the boost lib for the first time. For this assignment, I implemented a Linear Feedback Shift Register class called LFSR and created a bunch of test cases to test my implementation.

The implementation for each step in the LFSR was quite simple, all I had to was take the left most bit and the keybit and use the bitwise XOR operator to produce the output which would be appended to the string. I also had to remove the first item in the string after appending the last character. I also had to implement my own test cases for testing my new class. For each class I instantiated a LFSR class with a string and keybit and then made sure that the returning item for generate produced the correct outcome after having done the LFSR on my own with pen and paper.

I learned how to create a Boost test case in this assignment and how to encrypt text. I also got better at working with strings but in the end I learned the string implementation for C++ is similar to many other container times. I learned strings are slower than arrays because it literally has to make a copy of itself every time I want to manipulate it. I learned there are ways around this by using emplace_back() instead of .append().

A problem I had with this assignment was that I forgot to create separate cases for each test case and instead used the one test case and implemented all my tests into. I fixed this issue and now I realized each test case has to be its own function to better distinguish the source of an error.

PS4b Image Encoding

This assignment was an extension of the LFSR. I use the LFSR class to encrypt an image and use the same keybit and string to decrypt the image. I was able to do this and have the before and after image in the same window.

I created a program that would utilize the LFSR class to encrypt the pixels in the image with the keybit. I used SFML to create a window with both images. I pretty much did the same thing I did in the first assignment but now with the LFSR feature. In order to use LFSR to encrypt and decrypt I simply had to use the xor bitwise operator with the generate() function of LFSR.

```
p = image.getPixel(x, y);
p.r ^= lfsr.generate(tap);
p.g ^= lfsr.generate(tap);
p.b ^= lfsr.generate(tap);
image.setPixel(x, y, p);
```

I learned I could filter images by using xor with rgb values. I also learned SFML can create files with the image by using the image.saveToFile() function.

I had an error in my submission because I played around with my code too much after finishing the assignment and not returning it back to normal before submitting the code. I had to fix it again but I think the TA might have not been able to compile it, but it does work, and the working code is provided in this pdf.

Encrypted

Decrypted

PS5a Ring Buffer

For this assignment, we were to create a ring buffer to hold Int16 values. We used Boost to test our new class.

The primary functions that we had to work on for this assignment were enqueue and dequeue. I created a vector of Int16s that would could indexed from first and last depending on the number of times it had been enqueue/dequeued. I had to keep track of the index each time I would add items to the vector and remove them. I also had functions to inform the full or empty status of the buffer. This was mostly an object orient assignment along with some testing.

I learned how to enqueue and dequeue items from a buffer. We also used exceptions in this assignment which we had learned to use in Computing III.

PS5b Karplus-Strong String Simulation and Guitar Hero

I created a Guitar String class with the aid of the RingBuffer class and the Karplus-Strong algorithm. We also had to create a Guitar Hero player that would be able to play certain notes depending on keys pressed on our keypad. I used SFML to create my music synthesizer and was able to output the correct note depending on the key being pressed.

Basically, the Guitar String has a ring buffer and I take the average of the first two Int16s in the container and then apply a decay to this value. Pluck() empties whatever is inside the ringbuffer and then fills it in with random Int16s .

In order to create each note to be played I had to create a temporary vector of Int16s as the samples and then store these samples into the sound buffer buf. Buf would permanently hold these values so I did not need to access the vector to play the sample.

```
for(int i=0;i<37;i++){
66: double freq = CONCERT_A*pow(2, ((double)(i-24)/12.0));
67: GuitarString gs = GuitarString(freq);
68: /*sf::Sound sound;
69: sf::SoundBuffer buf;
70: sounds.push_back(sound);
71: bufs.push_back(buf);
72: */
73: vector<sf::Int16> samples = makeSamplesFromString(gs);
74: if (!bufs[i].loadFromSamples(&samples[0], samples.size(), 2, SAMPLES_
PER_SEC))
75: throw std::runtime_error("sf::SoundBuffer: failed to load from samp
les.");
```

For the keyboard I created an array of ints posMap filled with -1 integer values.

```
57: string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
58: int posMap[128];
59: for(int i=0;i<128;i++) posMap[i]=-1;
```

I then took the ascii value from each of the characters in the string keyboard and used it as the index for my posMap, depending on the location of the character, it would be assigned the appropriating sound buffer later on in line 66.

```
61: for(int i=0;i<keyboard.size();i++){
62: posMap[keyboard[i]] = i;
63: }
```

This condition allows me to play the notes that have been assigned to a key.

```
if (posMap[static_cast<char>(event.text.unicode)]>=0)
sounds[posMap[static_cast<char>(event.text.unicode)]].play();
```

For this assignment, I learned about soundbuffers in SFML and how they work. I learned how samples and frequencies work and how to make sounds with them.

PS6 Markov Model of Natural Language

In this assignment, we learned to use Markovs models of Natural Language to generate pseudo-random text. Depending on the order of the Markov's model, we create k consecutive letters and we keep track of the number of times they occur; their frequencies. We also keep track of the frequencies for the number of times a character appears after this k-gram. For this assignment I created a text generator that takes MarkovModel class and uses the generator function to output pseudo random phrases.

My constructor takes in the value k and the string from text and creates an array of maps for each of the characters in the text. I start off by taking the first kgram from the text and then I store the kgrams for each kgram+1 character and increment their frequencies. By default the count is always zero but if it is not zero I create a key with the kgram and increase its frequency. I do this for each kgram.

```
if (k>0) temp = text.substr(0,k);
for (int i = k; i < text.size()+k; i++)
{
if (_kgrams[text[i%text.size()]].count(temp) == 0){
kgramList.insert(temp);
_kgrams[text[i%text.size()]][temp] = 1;
} else {
_kgrams[text[i%text.size()]][temp]++;
}
temp += text[i%text.size()];
temp = temp.substr(1);

}
```

My generator works by taking in the first kgram substring from the text and the number of chars to append to my string. I use randk to get a random character and append it to my resulting string. It updates the kgram and keeps generating random characters using rand.

```
string result = "";
for(int i=0;i<T;i++){
//cout<<kgram<<"-\n";
char nextChar = randk(kgram);
//cout<<"next char="<<nextChar<<endl;
result += nextChar;
kgram += nextChar;
kgram = kgram.substr(1);
}
return result;
```

Randk takes the kgram and generates a random character. Rand finds the total frequency of the kgram and grabs a random frequency and uses this random frequency to find the first kgram+1 greater than ran and returns it to generator.

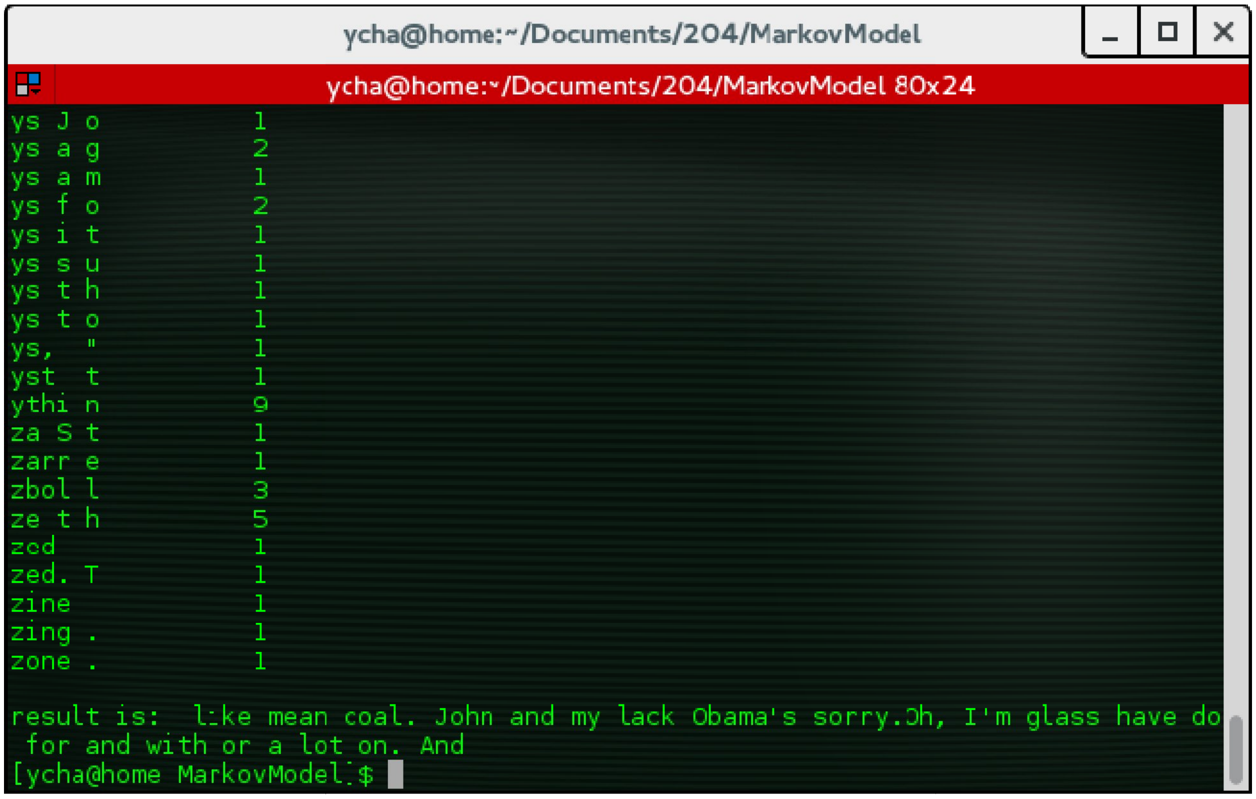
```
int freqArr[128], total=0;
for(int i=0;i<128;i++){
freqArr[i] = MarkovModel::freq(kgram,i);
```

```
total += freqArr[i];
}
if
if (total!=0){
int ran = rand()%total+1;

for(int i=0;i<128;i++){
// cout<<"ran="<<ran<<endl;
ran -= freqArr[i];
if (ran<=0) {
//cout<<"return "<<i <<" "<<freqArr[i]<<endl;
return (char) i;
}
}
```

I learned how to randomly create text from this assignment. This assignment took a lot of time, effort and testing to work. If anything this assignment helped me write code incrementally and slowly so that everything works and testing it along the way.

The hardest part of this assignment was the amount of hours, concentration and effort it took to get all the pieces working. It took a lot of time trying to debug and I had to constantly compile and print out values to figure out why things were not working.


ycha@home:~/Documents/204/MarkovModel

ycha@home:~/Documents/204/MarkovModel 80x24

```
ys J o          1
ys a g          2
ys a m          1
ys f o          2
ys i t          1
ys s u          1
ys t h          1
ys t o          1
ys,   "         1
yst  t          1
ythi n          9
za S t          1
zarr e          1
zbol l          3
ze t h          5
zcd             1
zed. T          1
zine            1
zing .          1
zone .          1

result is:  l:ke mean coal. John and my lack Obama's sorry.Oh, I'm glass have do
 for and with or a lot on. And
[ycha@home MarkovModel]$ 
```

PS7a Kronos Time Clock Log Parsing Boot Parsing/PS7b Kronos Time Clock Log Services and Software Updates Parsing

This assignment required learning about deterministic finite automaton and how to use regex to parse files. My program takes log files and finds out whether or a not a service/upgrade started and whether or not it finished.

My program takes a bunch of regexs and uses regex_match to store the match result into an array of strings called smatch. Once a service starts I set a flag and if it restarts without completing I return a faulty service startup message. I used instantiate a temporary date object with the date then offset it the hours, minutes, and seconds and store the result into a ptime variable. I use the ptime variables to calculate the elapsed time.

```
70: int h = stoi(sm[5]);
71: int m = stoi(sm[6]);
72: int s = stoi(sm[7]);
73: ptime temp(d, time_duration(h,m,s));
97: time_duration td = endTime- startTime;
```

I learned how to use regex in this assignment. I learned that boost is a great library full of very use containers and data types.

```
Parsed Log

-----------------------------------------



Line 739:Initial boot(2013-Dec-04 15:53:55)


Line 900:Boot sequence ended(2013-Dec-04 15:57:18)


Boot successful! Elapsed time(00:03:23)




Line 14282:Initial boot(2014-Jan-24 16:30:20)


Line 30190:Boot sequence ended(2014-Jan-24 16:32:51)


Boot successful! Elapsed time(00:02:31)
```

```
Line 30199:Initial boot(2014-Jan-24 17:41:46)

Line 30427:Boot sequence ended(2014-Jan-24 17:43:47)

Boot successful! Elapsed time(00:02:01)

Line 30778:Initial boot(2014-Jan-27 13:38:20)
```

PS0

Makefile Tue May 06 22:58:08 2014 1

```
main: main.o
g++ -g main.o -o main -lsfml-graphics -lsfml-window -lsfml-system --std=c++0x

main.o:
g++ -g -c main.cpp

demo: demo.o
g++ -g demo.o -o demo -lsfml-graphics -lsfml-window -lsfml-system --std=c++0x

demo.o:
g++ -g -c demo.cpp

clean:
rm -rf *.o *.gch *~
```

main.cpp Tue May 06 22:50:25 2014 1

```

#include <SFML/Graphics.hpp>

```

```
int main()
{
sf::RenderWindow window(sf::VideoMode(200, 200), "SFML works!");
sf::CircleShape shape(100.f);
shape.setFillColor(sf::Color::Green);

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

window.clear();
window.draw(shape);
window.display();
}
return 0;
}
```

demo.cpp Wed May 07 02:00:58 2014 1

```

#include <SFML/Graphics.hpp>

int main()
{
const int width = 800;
const int height = 600;
// Create the main window
sf::RenderWindow window(sf::VideoMode(width, height), "SFML window");
// Enable vertical synchronization to sync with graphics card
window.setVerticalSyncEnabled(true);

// Load a sprite to display
sf::Texture texture;
if (!texture.loadFromFile("cute_image.jpg"))
return EXIT_FAILURE;
sf::Sprite sprite(texture);
sprite.setPosition(0, width / 2);
// Create a graphical text to display
sf::Font font;
if (!font.loadFromFile("arial.ttf"))
return EXIT_FAILURE;
sf::Text text("Hello SFML", font, 50);

// Start the game loop
while (window.isOpen())
{
// Process events
sf::Event event;
while (window.pollEvent(event))
{
// Close window : exit
if (event.type == sf::Event::Closed)
window.close();

// Use the arrow keys to move the sprite
// in that direction.
if ((sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) && (sprite.getPosition().x < (width - 20)))
sprite.move(30, 0);
if ((sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) && (sprite.getPosition().x > 0))
sprite.move(-30, 0);
if ((sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) && (sprite.getPosition().y > 100))
sprite.move(0, -30);
if ((sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) && (sprite.getPosition().y < height - 20))
sprite.move(0, 30);

}

// Clear screen
window.clear();
// Draw the sprite and string
window.draw(sprite);
window.draw(text);
// Update the window
window.display();
}
return EXIT_SUCCESS;
}
```

PS1 N-Body Simulation

Body.hpp Wed May 07 09:04:16 2014 1

```

#ifndef _BODY_H
#define _BODY_H

#include <SFML/System.hpp>
#include <cstring>

static const double gravity = 6.67e-11;

class Body
{
private:
std::pair<double,double> position;
std::pair<double,double> velocity;
double mass;
std::string name;
public:
Body(std::pair<double,double> p, std::pair<double,double> v, double n, std::string s);
std::pair<double,double> getPosition() const;
std::pair<double,double> getVelocity() const;
double getMass() const;
```

```cpp
std::string getName() const;
void setPosition(double x, double y);
void setVelocity(double vx, double vy);
};

#endif



```

Body.cpp Wed May 07 09:01:55 2014 1

```cpp

#include "Body.hpp"

#define x first
#define y second

using namespace std;

Body::Body(pair<double,double> p, pair<double,double> v, double n, string s):
position(p), velocity(v), mass(n), name(s)
{}

pair<double,double> Body::getPosition() const
{
return position;
}
pair<double,double> Body::getVelocity() const
{
return velocity;
}
double Body::getMass() const
{
return mass;
}
string Body::getName() const
{
return name;
}
void Body::setPosition(double x, double y)
{
position.x = x;
position.y = y;
}
void Body::setVelocity(double vx, double vy)
{
velocity.x = vx;
velocity.y = vy;
}

```

SolarSystem.hpp Wed May 07 09:04:38 2014 1

```cpp

#ifndef _SOLARSYSTEM_H
#define _SOLARSYSTEM_H
```

```cpp

#include "Body.hpp"
#include <string>
#include <cstdlib>
#include <iostream>
#include <vector>
#include <cmath>

class SolarSystem
{
private:
int numPlanets;
double radius;
std::vector<Body> stuff;
public:
SolarSystem();
std::pair<double,double> getPosition(int i) const;
std::string getName(int i) const;
double getRadius() const;
int getNumPlanets() const;
double getDistance(std::pair<double,double> a, std::pair<double,double> b);
void leapFrog(double dT);
};

#endif


```

SolarSystem.cpp Wed May 07 10:36:14 2014 1

```cpp

#include "SolarSystem.hpp"
#include <cstdio>

#define x first
#define y second

using namespace std;
using namespace sf;

SolarSystem::SolarSystem()
{
cin >> numPlanets;
cin >> radius;
for (int i = 0; i < numPlanets; i++)
{
pair<double,double> p, v;
double m;
string name;
cin >> p.x;
cin >> p.y;
cin >> v.x;
cin >> v.y;
v.y = -v.y;
cin >> m;
cin >> name;
stuff.push_back(Body(p, v, m, name));
```

```
}
}

pair<double,double> SolarSystem::getPosition(int i) const
{
pair<double,double> p = stuff[i].getPosition();
p.first = p.first * 256 /radius +256;
p.second = p.second * 256 / radius + 256;
return p;
}

string SolarSystem::getName(int i) const
{
return stuff[i].getName();
}

double SolarSystem::getRadius() const
{
return radius;
}

int SolarSystem::getNumPlanets() const
{
return numPlanets;
}

double SolarSystem::getDistance(pair<double,double> a, pair<double,double> b)
{
double dx = a.x - b.x;
double dy = a.y - b.y;
return (sqrt((dx*dx)+(dy*dy)));
}

void SolarSystem::leapFrog(double dT)
{
vector<pair<double,double> > forces[numPlanets];

for (int i = 0; i < numPlanets; i++)
{
// update inputs for each of the matrices
for (int j = 0; j < numPlanets; j++)
{
// do not calculate force of body on itself
if (i != j)
{
double r, F, Fx, Fy;
// obtain distance using pythagorean theorem
r = getDistance((stuff[i].getPosition()),(stuff[j].getPosition()));
// obtain force from G(Mm/r^2)
F = (gravity*(stuff[i].getMass()/r)*(stuff[j].getMass()/r));
// obtain x and y components of force
Fx = ((F*(stuff[j].getPosition().x - stuff[i].getPosition().x)) / r);
Fy = ((F*(stuff[j].getPosition().y - stuff[i].getPosition().y)) / r);
// push new forces to temporary vector
forces[i].push_back(pair<double,double>(Fx, Fy));
}
}
}
for (int i = 0; i < numPlanets; i++){
// add all the forces for matrix i
pair<double,double> force(0.0, 0.0);
for (int k = 0; k < numPlanets-1; k++)
{
//cout<<force.x<<"_"<<forces[i][k].x<<endl;
force.x += forces[i][k].x;
//cout<<force.y<<"."<<forces[i][k].y<<endl;
force.y += forces[i][k].y;
}
// update acceleration
pair<double,double> a((force.x / stuff[i].getMass()), (force.y / stuff[i].getMass()));
// update velocity
pair<double,double> v(stuff[i].getVelocity());
v.x += dT * a.x;
v.y += dT * a.y;
stuff[i].setVelocity(v.x, v.y);
// update positions using the new x and y components of velocity
pair<double,double> p(stuff[i].getPosition());
p.x += dT * v.x;
p.y += dT * v.y;
stuff[i].setPosition(p.x, p.y);
}
}

```

nbody.cpp Wed May 07 09:41:16 2014 1

```

// build with make

#include "Body.hpp"
#include "SolarSystem.hpp"
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <map>
#include <memory>
#include <vector>
#include <iostream>
#include <cstdlib>
#include <string>

using namespace std;
using namespace sf;

static std::map<string, Texture*> textures;

#define WIDTH 512
#define HEIGHT 512

int main(int argc, char *argv[])
{
int pause = 1;
```

```
// create render object
RenderWindow window(VideoMode(WIDTH, HEIGHT, 32), "N-BODY");
window.setVerticalSyncEnabled(true);
window.setFramerateLimit(60);

// input time and deltatime and set up timer
Clock timer;
static double T(15034523.0), delayTime(25000.0);
switch(argc){
case 2:
T = atof(argv[1]);
break;
case 3:
T = atof(argv[1]);
delayTime = atof(argv[2]);
break;
default:
break;
}

//load in background image

Texture background;
if (!background.loadFromFile("starfield.jpg"))
{
return EXIT_FAILURE+ 11;
}
Sprite starfield(background);
starfield.setPosition(0, 0);

// create Solar System
SolarSystem universe;
vector<Sprite> sprites;
// create sprites for each body in the solar system
for (int i = 0; i < universe.getNumPlanets(); i++)
{
Texture* texture;
texture = new Texture();
texture->loadFromFile(universe.getName(i));
textures[universe.getName(i)]= texture;
Sprite sprite(*textures[universe.getName(i)]);
sprites.push_back(sprite);
pair<double,double> p = universe.getPosition(i);
Vector2f point;
point.x = p.first;
point.y = p.second;
sprites[i].setPosition(point);
}

while (window.isOpen())
{
// start of event loop
Event event;
double elapsedTime=0;
while (window.pollEvent(event))
{
// Close window
if ((event.type == Event::Closed) && (timer.getElapsedTime().asSeconds() < T))
window.close();
// Keypressed event P will pause simulation
if ((event.type == sf::Event::KeyPressed) && (event.key.code == sf::Keyboard::P))
{
if (pause == 0)
pause = 1;
else
pause = 0;
}
/*universe.leapFrog(delayTime);
for (int i = 0; i < universe.getNumPlanets(); i++)
{
pair<double,double> p = universe.getPosition(i);
Vector2f point;
point.x = p.first;
point.y = p.second;
sprites[i].setPosition(point);
}*/
//if (timer.getElapsedTime().asMilliseconds() - elapsedTime > delayTime)
//elapsedTime = timer.getElapsedTime().asMilliseconds();
}
// run simulation
if (pause)
universe.leapFrog(delayTime);
for (int i = 0; i < universe.getNumPlanets(); i++)
{
pair<double,double> p = universe.getPosition(i);
Vector2f point;
point.x = p.first;
point.y = p.second;
sprites[i].setPosition(point);
}

window.clear(Color::Black);
window.draw(starfield);
for (int i = 0; i < universe.getNumPlanets(); i++)
{
window.draw(sprites[i]);
}
window.display();
}
return EXIT_SUCCESS;
}
```

PS2 Recursive Graphics

SIERPINSKIS

main.cpp Tue May 06 22:33:32 2014 1

```

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

```
#include <vector>

using namespace std;
using namespace sf;

// Vector container to hold triangles
vector<ConvexShape*> triangles;

// Calculates the height of each triangle
inline double getHeight(double h)
{
return (sqrt(3)/2)*h;
}

// Create triangles for each instance of the recursive call
void fillTriangle(Vector2f p1, Vector2f p2, Vector2f p3)
{
ConvexShape* triangle = new ConvexShape(3);
triangle->setFillColor(Color::Black);
triangle->setPoint(0, p1);
triangle->setPoint(1, p2);
triangle->setPoint(2, p3);
triangles.push_back(triangle);
}

// Recursive function to create Sierpinski’s triangle
void sierpinski(int n, Vector2f p1, Vector2f p2, Vector2f p3)
{
if (n > 0)
{
Vector2f m1((p1.x+p2.x)/2, (p1.y+p2.y)/2);
Vector2f m2((p2.x+p3.x)/2, (p2.y+p3.y)/2);
Vector2f m3((p3.x+p1.x)/2, (p3.y+p1.y)/2);

fillTriangle(m1,m2, m3);
sierpinski(n-1, p1, m1, m3);
sierpinski(n-1, m2, p2, m1);
sierpinski(n-1, m3, m2, p3);
}
}

int main(int argc, char *argv[])
{
// Get number of recursive calls to take
int N;
if (argc == 2)
{
if ((N = atoi(argv[1])) > 10 )
{
cerr << "Too many recursive calls, please enter a smaller number \n";
return EXIT_FAILURE;
}
}

// Create window for app
RenderWindow app(VideoMode(800, 800), "SFML window");

// Define a triangle using ConvexShape class with 3 points
ConvexShape triangle(3);
triangle.setFillColor(Color::White);
triangle.setPoint(0, Vector2f(400, 0));
triangle.setPoint(1, Vector2f(0, getHeight(800)));
triangle.setPoint(2, Vector2f(800, getHeight(800)));

// Start the sierpinski recursion call
sierpinski(N, Vector2f(400, 0), Vector2f(0, getHeight(800)), Vector2f(
800, getHeight(800)));

// Start the app loop
while (app.isOpen())
{
Event event;
// Start of event loop
while (app.pollEvent(event))
{
if (event.type == Event::Closed)
app.close();
}

app.clear();

// Draw all existing triangles
app.draw(triangle);
for (int i = 0; i != triangles.size(); i++)
app.draw(*triangles[i]);
// Update the window
app.display();
}
return EXIT_SUCCESS;
}
```

RECURSIVE TREE

main.cpp Tue May 06 22:33:58 2014 1

```

#include <SFML/Graphics.hpp>
#include <vector>
#include <iostream>
#include <cmath>

using namespace std;
using namespace sf;

// Vector container to hold triangles
vector<RectangleShape> branches;

// Recursive function to create branches
void sierpinski(int n, Vector2f p, double length, double angle)
{
if (n > 0)
{
RectangleShape line(Vector2f(length, n));
line.rotate(angle*180/3.14);
line.setFillColor(Color::Green);
line.setPosition(p);

branches.push_back(line);
Vector2f nextP(p.x+length*cos(angle), p.y+length*sin(angle));
sierpinski(n-1, nextP, length*.85, angle+.5);
sierpinski(n-1, nextP, length*.85, angle-.5);
}
}

int main(int argc, char *argv[])
{
// Get number of recursive calls to take
int N;
if (argc == 2)
{
if ((N = atoi(argv[1])) > 18 )
{
cerr << "Too many recursive calls, please enter a smaller number \n";
return EXIT_FAILURE;
}
}

// anti aliasing for picture
ContextSettings settings;
settings.antialiasingLevel = 16;

// Create window for app
RenderWindow app(VideoMode(800, 800), "Recursive Tree", Style::Default, settings);

// Start the sierpinski recursion call
sierpinski(N, Vector2f(400, 700), 100, -3.14/2);

// Start the app loop
while (app.isOpen())
{
Event event;
// Start of event loop
while (app.pollEvent(event))
{
if (event.type == Event::Closed)
app.close();
}
// Clear screen
app.clear();

// Draw all existing branches
for (int i = 0; i != branches.size(); i++)
app.draw(branches[i]);

// Update the window
app.display();
}
return EXIT_SUCCESS;
}
```

Makefile Wed May 07 01:50:11 2014 1

```
all: main clean

main: main.o
g++ -g main.o --std=c++0x -lsfml-graphics -lsfml-window -lsfml-system -lm -s -o main

main.o:
g++ -g -c main.cpp

clean:
rm -rf *.o *.gch *˜
```

PS3 DNA Sequence Alignment

EditDistance.hpp Tue May 06 22:32:51 2014 1

```

#ifndef _EDITDISTANCE_H
#define _EDITDISTANCE_H

#include <vector>
#include <iostream>
#include <cstring>

class EditDistance
{
private:
std::vector<char> X;
std::vector<char> Y;
std::vector<std::vector<int> > CostMatrix;
public:
void printMatrix() const;
static int penalty(char a, char b);
static int min(int a, int b, int c);
std::string Alignment();
EditDistance(std::string a, std::string b);
int OptDistance(); // populates the NxM matrix
int getXsize() const;
int getYsize() const;
char getX(int i) const;
char getY(int i) const;
};

#endif
```

EditDistance.cpp Tue May 06 22:32:57 2014 1

```

#include "EditDistance.hpp"

using namespace std;

EditDistance::EditDistance(string a, string b):
X(a.begin(), a.end()), Y(b.begin(), b.end()),
CostMatrix(a.size()+1, vector<int> (b.size()+1, 0))
```

```
{
}

void EditDistance::printMatrix() const
{
for (int i = 0; i < CostMatrix.size(); i++)
{
for (int j = 0; j < CostMatrix[i].size(); j++)
cout << CostMatrix[i][j] << ’ ’;
cout << endl;
}
}

int EditDistance::penalty(char a, char b)
{
return (a == b ? 0 : 1);
}

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

string EditDistance::Alignment()
{
string output = "";
int i(0), j(0);

while ((i!=Y.size()) || (j!=X.size()))
{
if ((i<Y.size()) && (CostMatrix[j][i] - CostMatrix[j][i+1] == 2))
{
output = output + ’-’ + ’ ’ + Y[i] + ’ ’ + ’2’ + ’\n’;
i++;
}
else if ((j<X.size()) && (CostMatrix[j][i] - CostMatrix[j+1][i] == 2))
{
output = output + X[j] + ’ ’ + ’-’ + ’ ’ + ’2’ + ’\n’;
j++;
}
else
{
output = output + X[j] + ’ ’ + Y[i] + ’ ’ + char(’0’+penalty(X[j],Y[i])) + ’\n’;
i++;
j++;
}
}
return output;
}

```

```
// populate matrix and return the optimal distance
int EditDistance::OptDistance()
{
for (int i = X.size()-1; i >= 0; i--)
CostMatrix[i][Y.size()] = CostMatrix[i+1][Y.size()]+2;
for (int i = Y.size()-1; i >= 0; i--)
CostMatrix[X.size()][i] = CostMatrix[X.size()][i+1]+2;
for (int i = Y.size()-1; i >= 0; i--)
{
for (int j = X.size()-1; j >= 0; j--)
{
int b = penalty(X[j], Y[i]);
CostMatrix[j][i] = min((CostMatrix[j+1][i+1]+b),(CostMat
rix[j+1][i]+2),(CostMatrix[j][i+1]+2));
}
}
return CostMatrix[0][0];
}

int EditDistance::getXsize() const
{
return X.size();
}

int EditDistance::getYsize() const
{
return Y.size();
}

char EditDistance::getX(int i) const
{
return X[i];
}

char EditDistance::getY(int i) const
{
return Y[i];
}
```

main.cpp Tue May 06 22:32:59 2014 1

```

#include "EditDistance.hpp"
#include <SFML/System.hpp>
#include <iostream>
#include <vector>
#include <cstdlib>
#include <cstring>

using namespace std;
using namespace sf;

int main(int argc, char *argv[])
{
Clock clock;
Time t;
string xs, ys;
cin >> xs;
cin >> ys;
EditDistance ED(xs, ys);
cout << "Edit distance = " << ED.OptDistance() << endl;
cout << "----------------" << endl;
//ED.printMatrix();
//for (int i = 0; i < ED.getXsize(); i++)
// cout << ED.getX(i);
//cout << endl;
//for (int i = 0; i < ED.getYsize(); i++)
// cout << ED.getY(i);
// cout << endl;
cout << ED.Alignment() << endl;
t = clock.getElapsedTime();
cout << "Execution time is " << t.asSeconds() << "seconds. \n";
return 0;
}
```

Makefile Tue May 06 23:04:05 2014 1

```
all: main clean

main: main.o EditDistance.o
g++ -g main.o EditDistance.o --std=c++0x -lsfml-system -o1 -g -o main

main.o:
g++ -g -c main.cpp

EditDistance.o:
g++ -g -c EditDistance.cpp

clean:
rm -rf *.o *.gch *~

```

PS4a Linear Feedback Shift Register

LFSR.hpp Wed May 07 00:56:23 2014 1

```

#ifndef _LFSR_H
#define _LFSR_H

#include <cstring>
#include <deque>
#include <iostream>
#include <cstdlib>

class LFSR
{
private:
int keyBit;
std::string sequence;
public:
LFSR(std::string seed, int t);
int step();
int generate(int k);
```

```cpp
std::string toString();
int getKeyBit() const;
friend std::ostream& operator << (std::ostream &out, const LFSR &cLFSR);
};

#endif
```

LFSR.cpp Wed May 07 09:58:05 2014 1

```cpp

#include "LFSR.hpp"

using namespace std;

LFSR::LFSR(string seed, int t):
sequence(seed),
keyBit(t)
{}

int LFSR::step()
{
int result = ((sequence[sequence.size()-keyBit-1]-’0’) ^ (sequence[0] -’0’));
cout << (sequence[sequence.size()-keyBit-1]-’0’) << " " << (sequence[0] -’0’) << endl;
char intchar = result + ’0’;
string ctemp(1, intchar);
sequence.erase(sequence.begin(), sequence.begin()+1);
sequence.append(ctemp);
return result;
}

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

}
return result;
}

string LFSR::toString()
{
return sequence;
}

int LFSR::getKeyBit() const
{
return keyBit;
}

ostream& operator << (ostream &out, const LFSR &cLFSR)
{
out << cLFSR.sequence;
return out;
}
```

test.cpp Tue May 06 23:19:44 2014 1

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

#include "LFSR.hpp"

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

using namespace std;

BOOST_AUTO_TEST_CASE(fiveBitsTapAtTwo) {

LFSR l("00111", 2);
BOOST_REQUIRE(l.step() == 1);
BOOST_REQUIRE(l.step() == 1);
BOOST_REQUIRE(l.step() == 0);
BOOST_REQUIRE(l.step() == 0);
BOOST_REQUIRE(l.step() == 0);
BOOST_REQUIRE(l.step() == 1);
BOOST_REQUIRE(l.step() == 1);
BOOST_REQUIRE(l.step() == 0);

LFSR l2("00111", 2);
BOOST_REQUIRE(l2.generate(8) == 198);

}

BOOST_AUTO_TEST_CASE(sevenBitsTapAtFour) {
LFSR l3("001010", 4);
BOOST_REQUIRE(l3.generate(8) == 122);
}

BOOST_AUTO_TEST_CASE(eightBitsTapAtThree) {
LFSR l4("11111111", 3);
BOOST_REQUIRE(l4.generate(8) == 15);
}

BOOST_AUTO_TEST_CASE(sixBitsTapAtTwo) {
LFSR l5("101110", 2);
BOOST_REQUIRE(l5.generate(8) == 119);
}
```

Makefile Tue May 06 23:31:07 2014 1

```
all: main clean

main: LFSR.o test.o
g++ test.o LFSR.o --std=c++0x -lboost_unit_test_framework -s -o main

LFSR.o:
g++ -g -c LFSR.cpp
```

```

test.o:
g++ -g -c test.cpp

clean:
rm -rf *.o *.gch *~
```

PS4b Image Encoding

LFSR.hpp Wed May 07 00:56:23 2014 1

```

#ifndef _LFSR_H
#define _LFSR_H

#include <cstring>
#include <deque>
#include <iostream>
#include <cstdlib>

class LFSR
{
private:
int keyBit;
std::string sequence;
public:
LFSR(std::string seed, int t);
int step();
int generate(int k);
std::string toString();
int getKeyBit() const;
friend std::ostream& operator << (std::ostream &out, const LFSR &cLFSR);
};

#endif
```

LFSR.cpp Wed May 07 01:01:51 2014 1

```

#include "LFSR.hpp"

using namespace std;

LFSR::LFSR(string seed, int t):
sequence(seed),
keyBit(t)
{}

int LFSR::step()
{
int result = ((sequence[sequence.size()-keyBit-1]-'0') ^ (sequence[0]-'0'));
char intchar = result + '0';
string ctemp(1, intchar);
sequence.erase(sequence.begin(), sequence.begin()+1);
sequence.append(ctemp);
return result;
}

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

}
return result;
}

string LFSR::toString()
{
return sequence;
}

int LFSR::getKeyBit() const
{
return keyBit;
}

ostream& operator << (ostream &out, const LFSR &cLFSR)
{
out << cLFSR.sequence;
return out;
}

```

pixels.cpp Wed May 07 01:09:01 2014 1

```

#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include <string>
#include <cstdlib>
#include "LFSR.hpp"

using namespace std;
using namespace sf;

int main(int argc, char *argv[])
{
// input arguments
if(argc != 5)
{
cerr << "Need source image filename, output image filename, and LFSR seed and tap position.\n";
return EXIT_FAILURE + 4;
}
string sourceFile = argv[1];
string outputFile = argv[2];
string seed = argv[3];
unsigned int tap = atoi(argv[4]);

```

```
// load image from file
Image image;
if (!image.loadFromFile(sourceFile))
return EXIT_FAILURE + 6;
Image image2;
if (!image2.loadFromFile(sourceFile))
return EXIT_FAILURE + 7;

// instantiate a new lfsr object
LFSR lfsr(seed, tap);

// p is a pixel
Color p;

Vector2u size = image.getSize();
RenderWindow window(VideoMode(size.x *2, size.y), "Image Before/After");

// tranform image using lfsr method generate
for (int x=0; x < size.x; x++) {
for (int y= 0; y < size.y; y++) {
p = image.getPixel(x, y);
p.r ^= lfsr.generate(tap);
p.g ^= lfsr.generate(tap);
p.b ^= lfsr.generate(tap);
image.setPixel(x, y, p);
}
}

Texture texture;
texture.loadFromImage(image);

Texture texture2;
texture2.loadFromImage(image2);

Sprite sprite;
sprite.setTexture(texture);
sprite.setPosition(size.x, 0);

Sprite sprite2;
sprite2.setTexture(texture2);

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

window.clear(Color::White);
window.draw(sprite);
window.draw(sprite2);
window.display();
}

// write to new file
```

```
if (!image.saveToFile(outputFile))
return EXIT_FAILURE + 8;

return EXIT_SUCCESS;
}
```

Makefile Wed May 07 00:54:56 2014 1

```
all: pixels clean

pixels: pixels.o LFSR.o
g++ -g pixels.o LFSR.o --std=c++0x -lsfml-graphics -lsfml-window
-lsfml-system -s -o pixels

pixels.o:
g++ -g -c pixels.cpp

LFSR.o:
g++ -g -c LFSR.cpp

clean:
rm -rf *.o *.gch *˜
```

PS5 Karplus-Strong String Simulation and Guitar Hero(Ring Buffer)

RingBuffer.hpp Tue May 06 22:32:42 2014 1

```

#ifndef _RINGBUFFER_HPP
#define _RINGBUFFER_HPP

#include <SFML/System.hpp>
#include <vector>

class RingBuffer
{
private:
int time;
int _first;
int _last;
int _capacity;
bool _full;
std::vector<sf::Int16> ringBuff;
public:
RingBuffer(int capacity);
int size();
void empty();
bool isEmpty();
bool isFull();
void enqueue(sf::Int16 x);
sf::Int16 dequeue();
sf::Int16 peek();
};

#endif
```

RingBuffer.cpp Wed May 07 09:53:37 2014 1

```

#include <iostream>
#include <exception>
#include <stdexcept>
#include "RingBuffer.hpp"

using namespace std;
using namespace sf;

RingBuffer::RingBuffer(int capacity):
ringBuff(capacity, 0), _first(0), _last(0), _capacity(capacity), _full(false)
{
if(capacity < 1)
throw invalid_argument("Capacity must be larger than zero");
}
int RingBuffer::size()
{
return _capacity;
}
void RingBuffer::empty()
{
_first = 0;
_last = 0;
_full = false;
}
bool RingBuffer::isEmpty()
{
return !_full && (_first == _last);
}
bool RingBuffer::isFull()
{
return _full;
}
void RingBuffer::enqueue(Int16 x)
{
if (this->isFull())
throw runtime_error("Ring Buffer is full!");
ringBuff[_last] = x;
++_last;
if (_last == this->size()) _last = 0;
if (_last == _first) _full = true;
}
Int16 RingBuffer::dequeue()
{
if (this->isEmpty())
throw runtime_error("Ring Buffer is empty!");
Int16 x = ringBuff[_first];
++_first;
if (_first == this->size()) _first = 0;
if (_full) _full = false;
return x;
}
Int16 RingBuffer::peek()
{
if(isEmpty())
throw runtime_error("Ring Buffer is empty!");
return ringBuff[_first];
}
```

GuitarString.cpp Tue May 06 22:32:47 2014 1

```

#include <iostream>
#include <cstring>
#include <cstdlib>
#include <ctime>
#include "GuitarString.hpp"

using namespace std;
using namespace sf;

GuitarString::GuitarString(double frequency)
{
_time = 0;
_rb = new RingBuffer(frequency);
}
GuitarString::GuitarString(vector<Int16> init)
{
_rb = new RingBuffer(init.size());
for (unsigned int i = 0; i < init.size(); i++)
_rb->enqueue(init[i]);
}
GuitarString::~GuitarString()
{
}
void GuitarString::pluck()
{
_rb->empty();
while (!_rb->isFull())
{
_rb->enqueue((Int16)(rand() & 0xffff));
}
}
void GuitarString::tic()
{
Int16 x = _rb->dequeue();
double decay = .996*.5;
_rb->enqueue((x+_rb->peek()) * decay);
_time++;
}
Int16 GuitarString::sample()
{
return (_rb->peek());
}
int GuitarString::time()
{
```

```
return _time;
}
```

main.cpp Tue May 06 22:32:43 2014 1

```

// compile with
// make

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#define _USE_MATH_DEFINES
#include <math.h>
#include <limits.h>
#include <vector>

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

#include "RingBuffer.hpp"
#include "GuitarString.hpp"

using namespace std;

#define CONCERT_A 440.0
#define SAMPLES_PER_SEC 44100

// this makes a vector of <sf::Int16> from the Karplus-Strong string simulation
vector<sf::Int16> makeSamplesFromString(GuitarString gs) {
vector<sf::Int16> samples;

gs.pluck();
int duration = 8; // seconds
int i;
for (i= 0; i < SAMPLES_PER_SEC * duration; i++) {
gs.tic();
samples.push_back(gs.sample());
}

return samples;
}

int main()
{
sf::RenderWindow window(sf::VideoMode(300, 200), "SFML Guitar Hero Lite");
sf::Event event;

sf::Sound sounds[37];
sf::SoundBuffer bufs[37];

string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/’ ";
int posMap[128];
for(int i=0;i<128;i++) posMap[i]=-1;

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

for(int i=0;i<37;i++){
double freq = CONCERT_A*pow(2, ((double)(i-24)/12.0));
GuitarString gs = GuitarString(freq);
/*sf::Sound sound;
sf::SoundBuffer buf;
sounds.push_back(sound);
bufs.push_back(buf);
*/
vector<sf::Int16> samples = makeSamplesFromString(gs);
if (!bufs[i].loadFromSamples(&samples[0], samples.size(), 2, SAMPLES_
PER_SEC))
throw std::runtime_error("sf::SoundBuffer: failed to load from samp
les.");
sounds[i].setBuffer(bufs[i]);
}

while (window.isOpen()) {

while (window.pollEvent(event)) {

switch (event.type) {
case sf::Event::Closed:
window.close();
break;

case sf::Event::TextEntered:
cout << static_cast<char>(event.text.unicode) << " "
<< posMap[static_cast<char>(event.text.unicode)] << endl;

if (posMap[static_cast<char>(event.text.unicode)]>=0)
sounds[posMap[static_cast<char>(event.text.unicode)]].play();
break;

default:
break;

}

window.clear();
window.display();

}
}
return 0;
}
```

Makefile Tue May 06 22:32:41 2014 1

```
all: main

main: main.o GuitarString.o RingBuffer.o
g++ -g main.o GuitarString.o RingBuffer.o -lsfml-system -lsfml-wi
```

```
ndow -lsfml-graphics -lsfml-audio --std=c++0x -o main

main.o: main.cpp
g++ -g -c main.cpp -lsfml-system -lsfml-window -lsfml-graphics -lsfml-audio --std=c++0x

test: test.o RingBuffer.o
g++ -g test.o RingBuffer.o -o test -lboost_unit_test_framework

test.o: test.cpp
g++ -g -c test.cpp -lboost_unit_test_framework

testGS: testGS.o GuitarString.o RingBuffer.o
g++ -g testGS.o GuitarString.o RingBuffer.o -o testGS -lboost_unit_test_framework

testGS.o: testGS.cpp
g++ -g -c testGS.cpp -lboot_unit_test_framework

GuitarString.o: GuitarString.cpp GuitarString.hpp
g++ -g -c GuitarString.cpp

RingBuffer.o: RingBuffer.cpp RingBuffer.hpp
g++ -g -c RingBuffer.cpp --std=c++0x

clean:
rm -rf *.o *.gch *~

```

PS6 Markov Model of Natural Language

MarkovModel.cpp Wed May 07 00:37:54 2014 1

```

#include <iostream>
#include <ctime>
#include <random>
#include <vector>
#include <cstring>
#include "MarkovModel.hpp"

using namespace std;

MarkovModel::MarkovModel(string text, int k):_order(k), _alphabet(text)
{
srand(time(0));
if (k < 0)
throw invalid_argument("Order k must be higher than zero.");
if (k >= text.size())
throw invalid_argument("Order k must be less than the length of text.");

string temp = "";
if (k>0) temp = text.substr(0,k);

for (int i = k; i < text.size()+k; i++)
```

```
{
if (_kgrams[text[i%text.size()]].count(temp) == 0){
kgramList.insert(temp);
//kplusList.push_back(i);
_kgrams[text[i%text.size()]][temp] = 1;
} else {
_kgrams[text[i%text.size()]][temp]++;
}
temp += text[i%text.size()];
temp = temp.substr(1);

}
}

int MarkovModel::order()
{
return _order;
}

int MarkovModel::freq(string kgram)
{
// cout << "+" << kgram << "+" <<kgram.size()<<"+"<<_order<<endl;

if (kgram.size() != _order)
throw runtime_error("kgram must be the same size as k1");

int ans = 0;
for(int i=0;i<128;i++)
{
ans += _kgrams[i][kgram];
}
return ans;
}
int MarkovModel::freq(string kgram, char c)
{
if (kgram.size() != _order)
throw runtime_error("kgram must be the same size as k2");
if (!_kgrams[c].count(kgram))
return 0;
else
return _kgrams[c][kgram];
}

char MarkovModel::randk(string kgram)
{
if (kgram.size() != _order)
throw runtime_error("kgram must be the same size as k3");
int freqArr[128], total=0;
for(int i=0;i<128;i++){
freqArr[i] = MarkovModel::freq(kgram,i);
total += freqArr[i];
}
if (total!=0){
int ran = rand()%total+1;

for(int i=0;i<128;i++){
// cout<<"ran="<<ran<<endl;
ran -= freqArr[i];
if (ran<=0) {
//cout<<"return "<<i <<" "<<freqArr[i]<<endl;
return (char) i;
}
}
} else
throw runtime_error("No such kgram");

}

string MarkovModel::gen(string kgram, int T)
{
if (kgram.size() != _order)
throw runtime_error("kgram must be the same size as k");

string result = "";
for(int i=0;i<T;i++){
//cout<<kgram<<"-\n";
char nextChar = randk(kgram);
//cout<<"next char="<<nextChar<<endl;
result += nextChar;
kgram += nextChar;
kgram = kgram.substr(1);
}
return result;
}

ostream& operator<< (ostream& out, MarkovModel& mm)
{
out << "k-gram\tfreq" << endl;
out << "------------" << endl;
for (auto it = mm.kgramList.begin(); it != mm.kgramList.end(); it++)
{
int a = mm.freq(*it);
out << *it << ’\t’ << a << endl;
}
out << endl;
out << "k+1 gram\tfreq" << endl;
out << "--------------------" << endl;
for (auto it = mm.kgramList.begin(); it != mm.kgramList.end(); it++)
{
int a;
for(int i = 0; i < 128; i++) if ((a=mm.freq(*it, (char) (i)))>0)
{
out << *it << " " << (char)(i) << "\t\t" << a << endl;
}
}
return out;
}
```

TextGenerator.cpp Tue May 06 22:33:26 2014 1

```

// build with $make

#include <iostream>
#include <ctime>
```

```
#include <random>
#include <vector>
#include <cstring>
#include "MarkovModel.hpp"

using namespace std;

int main(int argc, char *argv[])
{
int k = atoi(argv[1]);
int T = atoi(argv[2]);
string text = "";
char c;
while(scanf("%c",&c)!=EOF){
if (c!='\n') text += c;
}
cout << endl;

MarkovModel mm(text, k);
cout << mm << endl;
//cout << k<< " "<< T <<" "<<text.substr(0,k) <<endl;
cout << "result is: ";
cout << mm.gen(text.substr(0,k),T) << endl;

return 0;
}
```

```

// build mmtest with $make mmtest

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

#include "MarkovModel.hpp"

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

using namespace std;

BOOST_AUTO_TEST_CASE(order0) {
// normal constructor
BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 0));

MarkovModel mm("gagggagaggcgagaaa", 0);

BOOST_REQUIRE(mm.order() == 0);
BOOST_REQUIRE(mm.freq("") == 17); // length of input in constructor
BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);

BOOST_REQUIRE(mm.freq("", 'g') == 9);
BOOST_REQUIRE(mm.freq("", 'a') == 7);
BOOST_REQUIRE(mm.freq("", 'c') == 1);
BOOST_REQUIRE(mm.freq("", ’x’) == 0);

}

BOOST_AUTO_TEST_CASE(order1) {
// normal constructor
BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 1));

MarkovModel mm("gagggagaggcgagaaa", 1);

BOOST_REQUIRE(mm.order() == 1);
BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
BOOST_REQUIRE_THROW(mm.freq("xx"), std::runtime_error);

BOOST_REQUIRE(mm.freq("a") == 7);
BOOST_REQUIRE(mm.freq("g") == 9);
BOOST_REQUIRE(mm.freq("c") == 1);

BOOST_REQUIRE(mm.freq("a", ’a’) == 2);
BOOST_REQUIRE(mm.freq("a", ’c’) == 0);
BOOST_REQUIRE(mm.freq("a", ’g’) == 5);

BOOST_REQUIRE(mm.freq("c", ’a’) == 0);
BOOST_REQUIRE(mm.freq("c", ’c’) == 0);
BOOST_REQUIRE(mm.freq("c", ’g’) == 1);

BOOST_REQUIRE(mm.freq("g", ’a’) == 5);
BOOST_REQUIRE(mm.freq("g", ’c’) == 1);
BOOST_REQUIRE(mm.freq("g", ’g’) == 3);

BOOST_REQUIRE_NO_THROW(mm.randk("a"));
BOOST_REQUIRE_NO_THROW(mm.randk("c"));
BOOST_REQUIRE_NO_THROW(mm.randk("g"));

BOOST_REQUIRE_THROW(mm.randk("x"), std::runtime_error);

BOOST_REQUIRE_THROW(mm.randk("xx"), std::runtime_error);

}

BOOST_AUTO_TEST_CASE(order2) {
// normal constructor
BOOST_REQUIRE_NO_THROW(MarkovModel("gagggagaggcgagaaa", 2));

MarkovModel mm("gagggagaggcgagaaa", 2);

BOOST_REQUIRE(mm.order() == 2);

BOOST_REQUIRE_THROW(mm.freq(""), std::runtime_error);
BOOST_REQUIRE_THROW(mm.freq("x"), std::runtime_error);
BOOST_REQUIRE_NO_THROW(mm.freq("xx"));
BOOST_REQUIRE_THROW(mm.freq("", ’g’), std::runtime_error); // kgram is wrong length
BOOST_REQUIRE_THROW(mm.freq("x", ’g’), std::runtime_error); // kgram is wrong length
BOOST_REQUIRE_THROW(mm.freq("xxx", ’g’), std::runtime_error); // kgram is wrong length
```

```
BOOST_REQUIRE(mm.freq("aa") == 2);
BOOST_REQUIRE(mm.freq("aa", ’a’) == 1);
BOOST_REQUIRE(mm.freq("aa", ’c’) == 0);
BOOST_REQUIRE(mm.freq("aa", ’g’) == 1);

BOOST_REQUIRE(mm.freq("ag") == 5);
BOOST_REQUIRE(mm.freq("ag", ’a’) == 3);
BOOST_REQUIRE(mm.freq("ag", ’c’) == 0);
BOOST_REQUIRE(mm.freq("ag", ’g’) == 2);

BOOST_REQUIRE(mm.freq("cg") == 1);
BOOST_REQUIRE(mm.freq("cg", ’a’) == 1);
BOOST_REQUIRE(mm.freq("cg", ’c’) == 0);
BOOST_REQUIRE(mm.freq("cg", ’g’) == 0);

BOOST_REQUIRE(mm.freq("ga") == 5);
BOOST_REQUIRE(mm.freq("ga", ’a’) == 1);
BOOST_REQUIRE(mm.freq("ga", ’c’) == 0);
BOOST_REQUIRE(mm.freq("ga", ’g’) == 4);

BOOST_REQUIRE(mm.freq("gc") == 1);
BOOST_REQUIRE(mm.freq("gc", ’a’) == 0);
BOOST_REQUIRE(mm.freq("gc", ’c’) == 0);
BOOST_REQUIRE(mm.freq("gc", ’g’) == 1);

BOOST_REQUIRE(mm.freq("gg") == 3);
BOOST_REQUIRE(mm.freq("gg", ’a’) == 1);
BOOST_REQUIRE(mm.freq("gg", ’c’) == 1);
BOOST_REQUIRE(mm.freq("gg", ’g’) == 1);

}
```

Makefile Tue May 06 23:32:57 2014 1

```
all: main clean

main: TextGenerator.o MarkovModel.o
g++ -g TextGenerator.o MarkovModel.o --std=c++0x -o main

TextGenerator.o: TextGenerator.cpp
g++ -g -c TextGenerator.cpp --std=c++0x

mmtest: mmtest.o MarkovModel.o
g++ -g mmtest.o MarkovModel.o -o test -lboost_unit_test_framework

mmtest.o: mmtest.cpp
g++ -g -c mmtest.cpp -lboost_unit_test_framework

MarkovModel.o: MarkovModel.cpp MarkovModel.hpp
g++ -g -c MarkovModel.cpp --std=c++0x

clean:
rm -rf *.o *.gch *~

```

PS7 A + B

```

// Build file with $make

#include <iostream>
#include <exception>
#include <stdexcept>
#include <string>
#include <cstdlib>
#include <boost/regex.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>

using namespace std;
using namespace boost;
using namespace boost::gregorian;
using namespace boost::posix_time;

int main ()
{
smatch sm;
int line = 0;
string outFile = " ";
string s;
ptime startTime, endTime, theTime;
int upgradeStart = 0;
int bootStart = 0;
int bootEnd = 0;
int nprocess = 0;
regex startBootRegex, endBootRegex, timeRegex, serviceStartRegex, successRegex, startUpgrade, endUpgrade;
regex oldApp, newApp;
// check for regex construction errors
try {
startBootRegex=".*((\\d{4})-([0-9]{2})-(\\d{2})).(\\d{2}):(\\d{2}):(\\d{2}): (\\(log\\.c\\.166\\) server started).*";
} catch (regex_error& exc) {
cerr << "Regex constructor failed with code " << exc.code() << endl;
exit(1);
}
try {
endBootRegex=".*((\\d{4})-(\\d{2})-(\\d{2})).(\\d{2}):(\\d{2}):(\\d{2})(.*)(oejs\\.AbstractConnector:Started SelectChannelConnector)(.*)";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}
try {
timeRegex=".*((\\d{4})-(\\d{2})-(\\d{2})).(\\d{2}):(\\d{2}):(\\d{2}).*";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}
```

```
try {
serviceStartRegex=".*(Starting Service).*";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}
try {
successRegex=".*(Service started).*";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}
try {
startUpgrade=".*(Install started).*";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}
try {
endUpgrade=".*(ExitValue from install command : 0).*";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}
try {
newApp=".*(Processing)(.*)(intouch-application-base-)(.*)(\\.armv6jel_vfp\\.rpm)(.*)";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}
try {
oldApp=".*(removing intouch-application-base-)(.*)(\\.armv6jel_vfp\\.rpm)(.*)";
} catch (regex_error& exc) {
cerr << "Regex constructor1 failed with code " << exc.code() << endl;
exit(1);
}

cout << "Parsed Log " << endl;
cout << "------------------------------------------" << endl;

while (getline(cin, s)) {
line++;
// Keep track of time
if (regex_match(s, sm, timeRegex))
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
date d(from_string(sm[1]));
int h = stoi(sm[5]);
int m = stoi(sm[6]);
int s = stoi(sm[7]);
ptime temp(d, time_duration(h,m,s));
theTime = temp;
```

```
}
// Check for Upgrade
if (regex_match(s, sm, startUpgrade))
{
if (upgradeStart == 1)
{
cout << "Upgrade Failed at line: " << line << "at (" << theTime <
< ")\n";
}
if (upgradeStart == 0)
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
upgradeStart = 1;
cout << "\n" << theTime << " " << sm[1] << endl;
}
}
if (regex_match(s, sm, newApp))
{
if (nprocess == 0)
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
cout << "\nNew Version: " << sm[4] << endl;
nprocess = 1;
}
}
if (regex_match(s, sm, oldApp))
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
nprocess = 0;
cout << "\nPrevious Version: " << sm[2] << endl;
}
// Look for uprade completion
if (regex_match(s, sm, endUpgrade))
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
upgradeStart = 0;
cout << "\n" << theTime << " " << sm[1] << endl;
}
// Check for Service startup
if (regex_match(s, sm, serviceStartRegex))
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
```

```
cout << "\n" << theTime << " " << sm[0] << endl;
}
// Check for successful startup
if (regex_match(s, sm, successRegex))
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
cout << "\n" << theTime << " " << sm[0] << endl;
}

// Check for Startup Sequence of Device
if (regex_match(s, sm, startBootRegex))
{
if (bootStart == 1)
{
cout << "\nBoot failed at line: " << line << endl << endl;
bootStart = 0;
}
if (bootStart == 0)
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
} */
bootStart = 1;
date d(from_string(sm[1]));
int h = stoi(sm[5]);
int m = stoi(sm[6]);
int s = stoi(sm[7]);
ptime temp(d, time_duration(h,m,s));
startTime = temp;
cout << "\nLine " << line << ":Initial boot(" << startTime << ")\n";
}
}
// Check for End of Sequence
if (regex_match(s, sm, endBootRegex))
{
/*for (int i = 0; i < sm.size(); i++)
{
cout << "sm[" << i << "]= " << sm[i] << endl;
}*/
bootEnd = 1;
date d(from_string(sm[1]));
int h = stoi(sm[5]);
int m = stoi(sm[6]);
int s = stoi(sm[7]);
ptime temp(d, time_duration(h,m,s));
endTime = temp;
cout <<"\nLine " << line << ":Boot sequence ended("<< endTime << ")\n";
}
// Check for Successful Boot
if (bootStart == 1 && bootEnd == 1)
{
```

```
time_duration td = endTime- startTime;
cout << "\nBoot successful! Elapsed time(" << td << ")\n\n";
bootStart = 0;
bootEnd = 0;
}
}
return 0;
}
```

Makefile Wed May 07 01:42:50 2014 1

```
all: main clean

main: main.o
g++ -g main.o --std=c++0x -o main -lboost_regex -lboost_date_time

main.o: main.cpp
g++ -g -c main.cpp --std=c++0x

clean:
rm -rf *.o *.gch *~

```