Michael Budney

COMP IV Section 201: Project Portfolio

Wednesday 27th April, 2022

Table of Contents:

PS0 Hello World with SFML

PS1 Linear Feedback Shift Register and Image Encoding

PS2 N-Body Simulation

PS3 Recursive Graphics - Sierpinski Triangles

PS4 Karplus-Strong Guitar String Simulation

PS5 DNA Sequence Alignment

PS6 Markov Model Random Writer

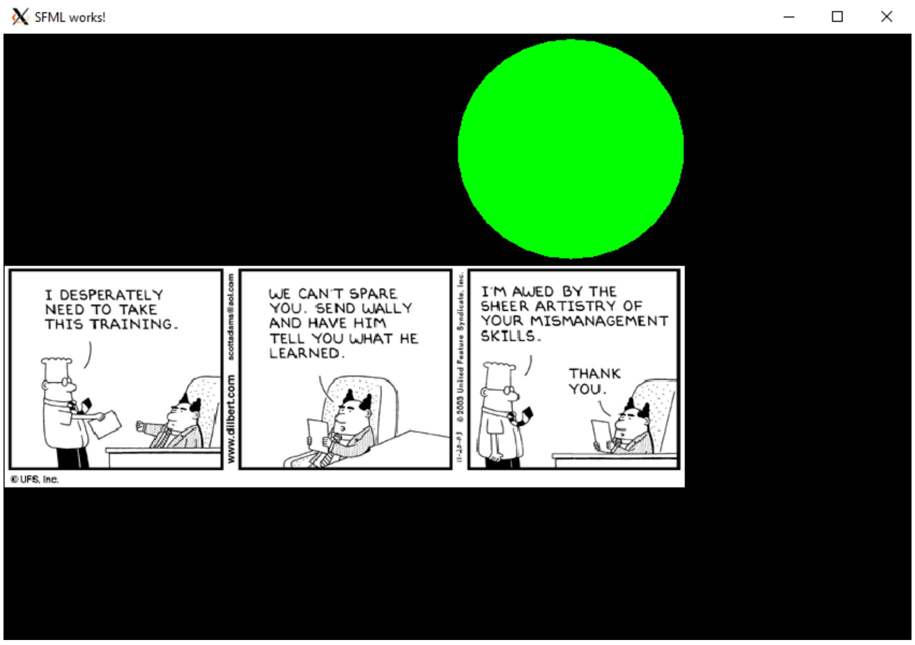
PS7 Kronos Time Clock Parsing w/ Regex

Time to Complete: 20 Hours

1 PS0 Hello World with SFML

1.1 Discussion

We used this introductory assignment to install the Simple Fast Media Library (SFML) and begin to understand its many functions. As well as prepare our systems for future projects that would require SFML.



SFML works!
I DESPERATELY NEED TO TAKE THIS TRAINING.
WE CAN'T SPARE YOU. SEND WALLY AND HAVE HIM TELL YOU WHAT HE LEARNED.
I'M AWED BY THE SHEER ARTISTRY OF YOUR MISMANAGEMENT SKILLS.
THANK YOU.
www.dilbert.com
scottadams@aol.com
© 2003 United Feature Syndicate, Inc.
© UFS, Inc.

Figure 1: PS0

1.2 Discussion of the assignment and what was accomplished

This assignment was used to set up our programming environment. I installed Windows Subsystem for Linux which is an Ubuntu distribution of Linux. In order to fully utilize the packages SFML has to offer, I also installed XServer to handle access to input and output devices. We were to display a green circle in the SFML window along with a sprite of our choosing.

1.3 What I already knew

As preparation, I underwent previous coursework in computer science using the C and C++ language. The basics of data structures, algorithms, and object oriented programming were well established before this course began.

1.4 What I learned

This was my first time installing a secondary operating system and input/output handling software. I learned a lot about the function of the event driven SFML library. This included displaying the green circle and a sprite of our choosing. I also did the extra credit to make the sprite moveable. It can be scrolled up, down, and rotated clockwise with event driven keystrokes.

1.5 Challenges

I had trouble figuring out the XServer usage. Turns out Windows 11 has its own built-in input/output handling device and will not cooperate with XServer. This was after I installed it on my main machine. There was some difficulty in getting the sprite to load with the SFML library.

1.6 A discussion of one or more key algorithms, data structures, or object oriented designs that were central to the assignment

Mostly just learning about the objects in the SFML library that pertained to the assignment. Event objects, window objects, sprite objects, etc.

1.7 Does it work? If not, what problem exist?

Works as intended.

1.8 Codebase

1.8.1 Makefile

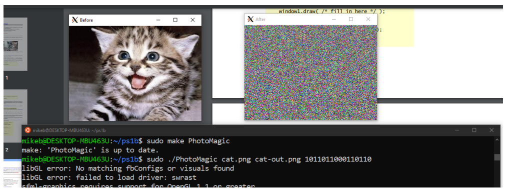
```
CC = g++
CFLAGS = -Wall -Werror -pedantic --std=c++17
LIBS = -lboost_unit_test_framework -lsfml-graphics -lsfml-window -lsfml-system

all: sfml-app
sfml-app: main.o
	$(CC) $(CFLAGS) -o sfml-app main.o $(LIBS)
main.o: main.cpp
	$(CC) $(CFLAGS) -c main.cpp $(LIBS)
lint:
	cpplint main . cpp
clean:
	rm -f main *.o
```

1.8.2 main.cpp

```
/***********************************
Name: Michael Budney
Computing 4
Jan 24th, 2021
***********************************/

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

```
#include<iostream>

int main()
{
    sf::Sprite Sprite;
    sf::RenderWindow window(sf::VideoMode(800, 800), ”SFML works!”);
    sf::CircleShape shape(100.f);
    shape.setFillColor(sf::Color::Green);
    sf::Texture texture;
    if (!texture.loadFromFile(”sprite.png”))
      {
 return EXIT_FAILURE;
      }
    shape.setPosition(400.f, 5.f);
    sf::Sprite sprite(texture);
    sprite.setPosition(1.f, 2.f);
    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }
    //make sprite move & respond to keystrokes - U for up and D for down
    if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::U)){
     sprite.move(sf::Vector2f(0.0, -0.6f));
 }
    if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::D)){
     sprite.move(sf::Vector2f(0.0, 0.6f));
 }

    //make sprite do something else - will rotate when R is pressed
    if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::R)){
     sprite.rotate(0.2f);
 }

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

    return EXIT_SUCCESS;
}
```

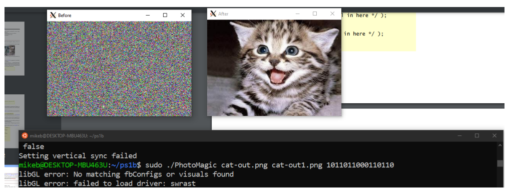
2 PS1 Linear Feedback Shift Register (LFSR) and Photomagic



```
window1.draw( /* fill in here */ );
```

```
mikeb@DESKTOP-MBU463U:~/ps1b$ sudo make PhotoMagic
make: 'PhotoMagic' is up to date.
mikeb@DESKTOP-MBU463U:~/ps1b$ sudo ./PhotoMagic cat.png cat-out.png 1011011000110110
libGL error: No matching fbConfigs or visuals found
libGL error: failed to load driver: swrast
sfml-graphics requires support for OpenGL 1.1 or greater
```

(a) Scramble Cat



```
in here */ );
in here */ );
```

```
 false
Setting vertical sync failed
mikeb@DESKTOP-MBU463U:~/ps1b$ sudo ./PhotoMagic cat-out.png cat-out1.png 1011011000110110
libGL error: No matching fbConfigs or visuals found
libGL error: failed to load driver: swrast
```

(b) Unscramble Cat

Figure 2: Cat Unharmed

2.1 Discussion of the assignment and what was accomplished

This assignment first had us create a Linear Feedback Shift Register as a means to encrypt our image. It functions as a psuedo-random number generator that left shifts bits, then replaces the rightmost bit with an XOR of the displaced bits and the tap bits. The program take 3 arguments, an input file, an output file, and a seed value for the pseudo generator.

2.2 What I already knew

We had a pair of assignments in Computing two where we dealt with bit fields as arrays in C. As well as binary (base two) system background knowledge from Computing one that came in handy.

2.3 What I learned

I learned about creating a pseudo-random number generator in the form of a linear feedback shift register. Additionally, gained an understanding of one possible usage of such as a basic encryption device. Gained a firmer understanding of bitwise operations and learned more about the SFML class objects.

2.4 Challenges

I did a lot of reading and research on the proper functioning of a linear feedback shift register in order to understand how to properly implement it. I struggled a fair amount in getting the LFSR class functioning properly, the step() and generate() functions required a lot of trial and error. In PhotoMagic.cpp it took me quite a while to get the image to decode, encoding is the easy part, but my functions needed a little tweaking to get it to decode properly.

2.5 Does it work?

It sure does.

2.6 Algorithms, data structures, or objected oriented designs central to the assignment?

The FibLFSR class uses arrays to construct, hold, and operate on the bit strings central to the operation of the linear feedback shift register.

2.7 Comments

Makes use of the SFML library to show a graphic representation of the image before and after the encoding process. Must call files in opposite argument positions with the same seed key in order to decode the image.

2.8 Codebase

2.8.1 Makefile

```
CC = g++
LIBR = -lboost_unit_test_framework -lsfml-graphics -lsfml-system -lsfml-window
CFLAGS = --std=c++17 -pedantic -Wall -Werror

all:PhotoMagic
PhotoMagic: FibLFSR.o PhotoMagic.o
 $(CC) $(CFLAGS) -o PhotoMagic PhotoMagic.o FibLFSR.o $(LIBR)
PhotoMagic.o: PhotoMagic.cpp
 $(CC) $(CFLAGS) -c PhotoMagic.cpp $(LIBR)
FibLFSR.o: FibLFSR.cpp FibLFSR.hpp
 $(CC) $(CFLAGS) -c FibLFSR.cpp
clean:
 rm -f *.o
distclean: clean
 rm -f PhotoMagic
```

2.8.2 FibLFSR.cpp

```
//name: Michael Budney
//date: 1/31/22

#include <iostream>
#include "FibLFSR.hpp"
#include <string>
using namespace std;

//construction
FibLFSR :: FibLFSR(string seed){
   size = seed.length();
   bit_string = new int[size];

   for(int i = 0; i < size; i++){
         bit_string[i] = seed[i] - '0';
      }
}

```

```
//destruction
FibLFSR :: ~FibLFSR(){
  delete[] bit_string;
}

int FibLFSR :: step(){
  int sBit = bit_string[0];
  int ten = bit_string[Size()-11];
  int twelve = bit_string[Size()-13];
  int thirteen = bit_string[Size()-14];
  int endB = bit_string[Size()-1];

  //bit shift left by 1 bit
  for(int i = 0; i < size; i++){
      bit_string[i] = bit_string[i+1];
    }
  endB = sBit ^ twelve ^ thirteen ^ ten;
  bit_string[Size()-1] = endB;
  return endB;
}

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

//length of bit-string
int FibLFSR :: Size(){
  return size;
}

//output operator
ostream& operator<<(ostream& out,  FibLFSR& obj){
  for(int i = 0; i < obj.Size(); i++){
      out << obj.bit_string[i];
    }
  cout << endl;
  return out;
}
```

2.8.3 PhotoMagic.cpp

```
#include <iostream>
#include <string>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>
#include "FibLFSR.hpp"
using namespace std;

```

```
void transform(sf::Image& im, FibLFSR* seed){
    sf::Color p;
 sf::Vector2u size = im.getSize();
 int x1 = size.x;
 int y1 = size.y;
 // encrypt/decrypt image
 for (int x = 0; x < x1; x++)
 {
  for (int y = 0; y < y1; y++)
  {
   p = im.getPixel(x, y);
   p.r = seed->generate(8) ^ p.r;
   p.g = seed->generate(8) ^ p.g;
   p.b = seed->generate(8) ^ p.b;
   im.setPixel(x, y, p);
  }
 }

}

int main(int arc, char* argv[]){
    string inName;
    string outName;
    string seedNum;

    inName = argv[1];
    outName = argv[2];
    seedNum = argv[3];

    FibLFSR seedObj(seedNum);


    //image 1
 sf::Image image;
 if (!image.loadFromFile(inName))
  return -1;

 //image 2
 sf::Image image2;
 if (!image2.loadFromFile(inName))
  return -1;

    transform(image, &seedObj);

    sf::Vector2u size = image.getSize();
    sf::Color p;

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

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

 sf::Texture texture2;
 texture2.loadFromImage(image2);

 sf::Sprite sprite2;
 sprite2.setTexture(texture2);

 sf::RenderWindow window1(sf::VideoMode(size.x, size.y), "After");
 sf::RenderWindow window2(sf::VideoMode(size.x, size.y), "Before");

 while (window1.isOpen() && window2.isOpen())
 {
  sf::Event event;
  while (window1.pollEvent(event))
  {
   if (event.type == sf::Event::Closed)
    window1.close();
  }
  while (window2.pollEvent(event))
  {
   if (event.type == sf::Event::Closed)
    window2.close();
  }
  window1.clear();
  window1.draw(sprite);
  window1.display();
  window2.clear();
  window2.draw(sprite2);
  window2.display();
 }

 //  write the file
 if (!image.saveToFile(outName))
  return -1;

 return 0;

}
```

3 PS2 NBody Simulation

3.1 Discussion of the assignment and what was accomplished

This assignment had us create a working model of the five innermost celestial bodies in our universe. In order to do this we had to use the open-source SFML library, and apply physics logic to sprite objects.

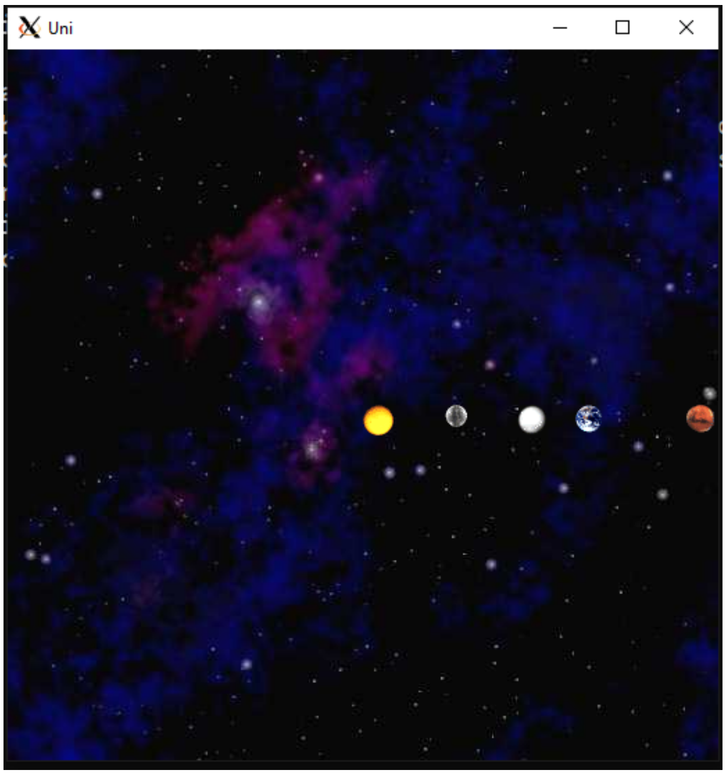


Figure 3: NBody Simulation

3.2 What I already knew

By now we have a decent understanding of the SFML library. Professor Daly gave us some advice on how to use the coordinates of the display port in relation to the coordinates of the planets. And of course some background calculus is more or less mandatory for physics.

3.3 What I learned

Beyond testing your calculus skills, this assignment really forced you to sharpen your object-oriented programming skills. It requires you to pass object references back and forth between classes. You had to write functions dependent on the other classes objects and methods.

3.4 Algorithm, data structures, or object-oriented designs central to the assignment?

The smart shared pointer was used for celestial body objects and is a private member variable of the universe class. This resolved ownership issues of the object references between the two classes.

3.5 Challenges

Had trouble getting new frames to consistently redraw to reflect the movement of the planets. They were either moving so fast that it essentially looked fixed, or just not redrawing at all. Most of the heartache was trial and error, pinpointing logical errors related to the step function.

3.6 Does it work?

It does. Though my planets are orbiting clockwise rather than counter-clockwise.

3.7 Comments

This simulation supports orbits of varying lengths, and will take as an argument a list of celestial bodies and their relative starting positions from a text file.

3.8 Codebase

3.8.1 Makefile

```
CC = g++ --std=c++17
LIBR = -lboost_unit_test_framework -lsfml-graphics -lsfml-system -lsfml-window
CFLAGS = -Wall -Werror -pedantic

all:NBody
NBody: Universe.o CelestialBody.o NBody.o lint
	$(CC) $(CFLAGS) -o NBody NBody.o Universe.o CelestialBody.o $(LIBR)
NBody.o: NBody.cpp
	$(CC) $(CFLAGS) -c NBody.cpp $(LIBR)
Universe.o: Universe.cpp
	$(CC) $(CFLAGS) -c Universe.cpp $(LIBR)
CelestialBody.o: CelestialBody.cpp CelestialBody.h
	$(CC) $(CFLAGS) -c CelestialBody.cpp $(LIBR)
clean:
	rm -f *.o
distclean: clean
	rm -f NBody
lint:
	cpplint Universe.cpp CelestialBody.cpp NBody.cpp
```

3.8.2 CelestialBody.h

```cpp

#pragma once
#include <iostream>
#include <vector>
#include <cmath>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
#include <memory>
#include <iomanip>
using std::vector;
using std::string;
using std::shared_ptr;
using std::istream;
using std::ostream;

class CelestialBody: public sf::Drawable  {
 public:
   CelestialBody();
   CelestialBody(double x, double y, double m, string fileName, double xvel,
      double yvel);
   void Initialize();
   void SetPos(const double& radius, const int& port);
   friend istream& operator>>(istream& in, CelestialBody& cBody);
   friend ostream& operator<<(ostream& out, CelestialBody& cBody);
   void NVelocity(const vector<shared_ptr<CelestialBody>>& planet  , double move
      );
   void NPosition(double stepLength);
   ~CelestialBody();
private:
   double xpos;
   double ypos;
   double mass;
   string file;
   double xvelocity;
   double yvelocity;
   sf::Texture cTexture;
   sf::Sprite cSprite;
   virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const;
};
```

3.8.3 CelestialBody.cpp

```cpp
// Copyright [2023] <MikeBudney>
#include "CelestialBody.h"
#include<string>
using std::setw;
using std::setprecision;
using std::istream;
using std::ostream;
using std::cout;
using std::vector;
using std::shared_ptr;
using std::sqrt;
using std::string;
using std::endl;

#define G 6.67e-11

CelestialBody::CelestialBody()      {}
CelestialBody::CelestialBody(double x, double y, double m, string fileName,
double xvel, double yvel) {
  xpos = x;
  ypos = y;
  mass = m;
  file = fileName;
  xvelocity = xvel;
  yvelocity = yvel;
  cTexture.loadFromFile(fileName);
  cSprite.setTexture(cTexture);
  cSprite.setPosition(x, y);
}
void CelestialBody::draw(sf::RenderTarget& target,
sf::RenderStates states) const {
  target.draw(cSprite, states);
}
void CelestialBody::SetPos(const double& radius, const int& port) {
  double xposition = (port / 2) + ((xpos / radius) * (port / 2));
  double yposition = (port / 2) + ((ypos / radius) * (port / 2));
  cSprite.setPosition(xposition, yposition);
}

void CelestialBody:: Initialize() {
  cTexture.loadFromFile(file);
  cSprite.setTexture(cTexture);
}

istream& operator>>(istream& input, CelestialBody& cBody) {
  input
  >> cBody.xpos >> cBody.ypos
  >> cBody.xvelocity >> cBody.yvelocity
  >> cBody.mass >> cBody.file;
  cBody.Initialize();
  return input;
}

ostream& operator<<(ostream& out, CelestialBody& CelBod) {
  out << setw(15) << std::left << setprecision(6) << CelBod.xpos  << setw(15)
      << std::left << setprecision(6) << CelBod.ypos  << setw(15)
      << std::left << setprecision(6) << CelBod.xvelocity  << setw(15)
      << std::left << setprecision(6) << CelBod.yvelocity << setw(15)
      << std::left << setprecision(6) << CelBod.mass << setw(15)
      << std::left << CelBod.file;
  return out;
}

void CelestialBody::NPosition(double stepLength)  {
  xpos += stepLength * xvelocity;
  ypos += stepLength * yvelocity;
}

void CelestialBody::NVelocity(const vector<shared_ptr
<CelestialBody>>& planet , double move) {
  double Fx = 0;
  double Fy = 0;
  for (auto i : planet) {
      if (xpos == i->xpos && ypos == i->ypos)
      continue;
      double dX = i->xpos - xpos;
      double dY = i->ypos - ypos;
      double objDistance = sqrt(pow(dX, 2) + pow(dY, 2) );
      double F = ((G * i->mass * this->mass) /  pow(objDistance, 2));
      Fx = F * (dX / objDistance);
      Fy = F * (dY / objDistance);
      double Ax = Fx / mass;
      double Ay = Fy / mass;
      this->xvelocity += move * Ax;
      this->yvelocity += move * Ay;
    }
}

CelestialBody::~CelestialBody() {}
```

3.8.4 Universe.h

```

#pragma once
#include "CelestialBody.h"
#include <vector>
#include <iostream>
#include <memory>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
using std::vector;
using std::shared_ptr;

class Universe : public sf::Drawable  {
public:
  Universe();
  void SetPlanetPosition(const int& port);
  void step(double seconds);
  ~Universe();
  void UniPrint();
  friend ostream& operator<<(ostream& out,  Universe& obj);
private:
 virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const
     override;
  double radius;
  int numPlanet;
  vector<shared_ptr<CelestialBody>> planet;
};
```

3.8.5 Universe.cpp

```
// Copyright [2023] <MikeBudney>
#include "Universe.h"
using std::vector;
using std::cout;
using std::cin;
using std::make_shared;
using std::endl;
using std::ostream;

void Universe::draw(sf::RenderTarget& target, sf::RenderStates states) const
    {
  for (int i = 0; i < numPlanet; i++) {
      target.draw(*planet[i], states);
    }
}

Universe::Universe() {
  cin >> numPlanet >> radius;
  for (int i = 0; i < numPlanet; i++) {
      planet.push_back(make_shared<CelestialBody>());
      cin >> *planet.back();
    }
}

void Universe::SetPlanetPosition(const int& port) {
  for (int i = 0; i < numPlanet; i++) {
      planet[i]->SetPos(radius, port);
    }
}
void Universe::step(double seconds) {
  for (int i = 0; i < numPlanet; i++) {
      planet[i]->NVelocity(planet, seconds);
      planet[i]->NPosition(seconds);
    }
}

ostream& operator<<(ostream& out, Universe& obj) {
  out << obj.numPlanet << endl;
  out << "r:" << obj.radius << endl;
  for (auto planetPtr : obj.planet) {
    out << *planetPtr << endl;
  }
  return out;
}

Universe::~Universe() {}
```

3.8.6 NBody.cpp

```
// Copyright [2023] <MikeBudney>
#include <iostream>
#include "CelestialBody.h"
```

```cpp
#include "Universe.h"
using std::endl;
using std::cout;
using std::ostringstream;
using std::string;
using std::stof;

int main(int argc, char **argv) {
  double start = 0;
  double end = stof(argv[1]);
  double timeFrame = stof(argv[2]);

  Universe uni;
  sf::Texture texture;
  texture.loadFromFile("starfield.jpg");
  if  (!texture.loadFromFile("starfield.jpg"))  {
      return 1;
    }
  sf::RenderWindow window1(sf::VideoMode(512, 512), "Uni");
  window1.setFramerateLimit(60);
  sf::Sprite sprite;
  sprite.setTexture(texture);
  uni.SetPlanetPosition(512);
  sf::Event event;
  while (window1.isOpen())  {
    if (start < end)  {
    uni.SetPlanetPosition(512);
    uni.step(timeFrame);
    start += timeFrame;
    }
    if (start >= end)
      break;
    while (window1.pollEvent(event))  {
      if (event.type == sf::Event::Closed)  {
        window1.close();
      }
    }
    window1.clear();
    window1.draw(sprite);
    window1.draw(uni);
    window1.display();
  }
cout << uni;
  return 0;
}
```

4 PS3 Recursive Graphics - Sierpinski Triangles

4.1 Discussion of the assignment and what was accomplished

PS3 required us to create a triangle class that we could use to implement a recursive triangle pattern known as the Sierpinski Triangles. It again required the use of the open-source SFML library.

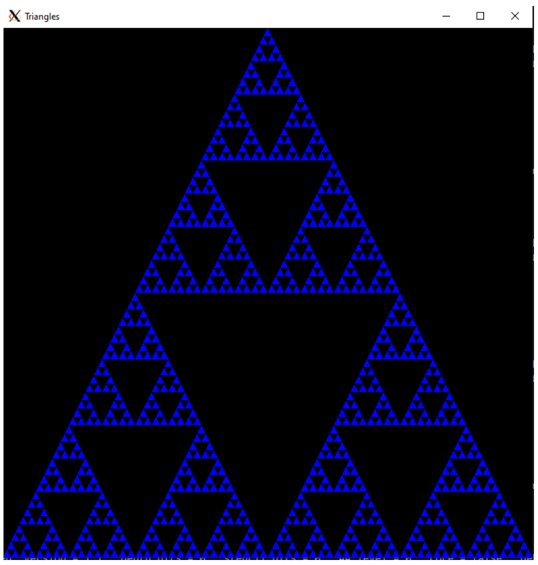


Figure 4: Triangle Fractals

4.2 What I already knew

Recursion is a common tool in computer science where there is an identifiable base case upon which to stop iterating. It's very efficient in terms of minimal lines of code and its drawbacks are felt only when there are many iterations on the call stack. The most common usage is probably in search trees of varying types.

4.3 What I learned

I learned how to draw fractals recursively using object oriented programming. The interplay between the drawer, the renderer, and the recursive function was particularly tough to figure out.

4.4 Algorithm, data structures, or object-oriented designs central to the assignment?

Used sf::RectangleShape as a parent class along with sf::Vector2f to build my class. The key insight was to use the rectangleshape to create a bounded area to draw each triangle in and draw the shape with the measures of the rectangle.

4.5 Challenges

Spent a long time trying to figure out how to get my triangle objects to draw. Spent an equally long time trying to get objects to draw permanently to a render window unsuccessfully. Figuring out I needed a separate rendering function held me back for a while.

4.6 Does it work?

Yes it does!

4.7 Comments

The program accepts two command line arguments, the first for the number of pixels to use for the base triangles side lengths, and the other allows you to select the number of iterations you would like to draw.

4.8 Codebase

4.8.1 Makefile

```
CC = g++ --std=c++17
LIBR = -lboost_unit_test_framework -lsfml-graphics -lsfml-system -lsfml-window
CFLAGS = -Wall -Werror -pedantic

all:TFractal
TFractal: Triangle.o TFractal.o lint
	$(CC) $(CFLAGS) -o TFractal TFractal.o Triangle.o $(LIBR)
TFractal.o: TFractal.cpp
	$(CC) $(CFLAGS) -c TFractal.cpp $(LIBR)
Triangle.o: Triangle.cpp
	$(CC) $(CFLAGS) -c Triangle.cpp $(LIBR)

clean:
	rm -f *.o
distclean: clean
	rm -f TFractal
lint:
	cpplint TFractal.cpp Triangle.cpp Triangle.h
```

4.8.2 Triangle.h

```
// Copyright [2023] <MikeBudney>
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
using sf::RectangleShape;
using sf::Vector2f;
using sf::RenderWindow;
using sf::Color;

class Sierpinski : public sf::RectangleShape, sf::Vector2f {
 public:
    Sierpinski()    {}
    ~Sierpinski()   {}
    void Render(sf::RenderWindow* window1);
    void fTree(const sf::Vector2f &bottom,
     const sf::Vector2f &left, const sf::Vector2f &right,
     int iteration, sf::RenderWindow* window);
    void setIterations(int newNumberOfIterations);
    void boundArea(float x1, float y1, double length);
    friend void drawTriangle(const sf::Vector2f &top1,
     const sf::Vector2f &left1, const sf::Vector2f &right1,
     const sf::Color &color1,  sf::RenderWindow* window1);


 private:
    int numIterations;
    sf::RectangleShape box;
    const sf::Color& m_color = (sf::Color::Blue);
};
```

4.8.3 Triangle.cpp

```
// Copyright [2023] <MikeBudney>

#include "Triangle.h"

void drawTriangle(const sf::Vector2f &top1, const sf::Vector2f &left1,
 const sf::Vector2f &right1, const sf::Color &color1,
 sf::RenderWindow* window1)  {
sf::ConvexShape shape;
  shape.setPointCount(3);
  shape.setPoint(0, top1);
  shape.setPoint(1, left1);
  shape.setPoint(2, right1);
  shape.setFillColor(color1);
  window1->draw(shape);
  return;
}

void Sierpinski::Render(sf::RenderWindow* window1)   {
  sf::Vector2f bottomRightPoint = sf::Vector2f(box.getPosition().x +
  box.getSize().x, box.getPosition().y + box.getSize().y);
  sf::Vector2f topLeftPoint = sf::Vector2f(box.getPosition());
  sf::Vector2f top = sf::Vector2f((bottomRightPoint.x +
  topLeftPoint.x)/2.0, topLeftPoint.y);
  sf::Vector2f left = sf::Vector2f(topLeftPoint.x, bottomRightPoint.y);
  sf::Vector2f right = sf::Vector2f(bottomRightPoint.x, bottomRightPoint.y);
  fTree(top, left, right, -1, window1);
}
```

```cpp

void Sierpinski::fTree(const sf::Vector2f &top,
 const sf::Vector2f &left, const sf::Vector2f &right,
 int iteration, sf::RenderWindow* window)    {
  if  (numIterations == 0) {
    drawTriangle(top, left, right, m_color, window);
    return;
  } else if (iteration == numIterations-1) {
      return;
  } else  {
    sf::Vector2f mLeft = sf::Vector2f((left.x + top.x)/2.0, (left.y+top.y)
        /2.0);
    sf::Vector2f mRight = sf::Vector2f((right.x + top.x)/2.0,
    (right.y + top.y)/2.0);
    sf::Vector2f mBottom = sf::Vector2f((left.x+right.x)/2.0,
    (left.y+right.y)/2.0);

    if  (iteration == numIterations-2) {
      drawTriangle(top, mLeft, mRight, m_color, window);
      drawTriangle(mLeft, left, mBottom, m_color, window);
      drawTriangle(mRight, mBottom, right, m_color, window);
    } else    {
      fTree(top, mLeft, mRight, iteration+1, window);
      fTree(mLeft, left, mBottom, iteration+1, window);
      fTree(mRight, mBottom, right, iteration+1, window);
    }
  }
}

void Sierpinski::setIterations(int number)   {
  numIterations = number;
}

void Sierpinski::boundArea(float x1, float y1, double length) {
  box = sf::RectangleShape();
  box.setPosition(sf::Vector2f(x1, y1));
  box.setSize(sf::Vector2f(length - x1, length - y1));
}
```

4.8.4 TFractal.cpp

```cpp
// Copyright [2023] <MikeBudney>
#include "Triangle.h"
#include <iostream>
#include <string>
using std::stod;
using std::stoi;
using sf::RenderWindow;
using sf::Event;

int main(int argc, char *argv[])    {
    const double L = stod(argv[1]);
    int N = stoi(argv[2]);
    Sierpinski triangle;
    triangle.boundArea(0.0, 0.0, L);
    triangle.setIterations(N);

    sf::RenderWindow window(sf::VideoMode(L, L), "Triangles");
    while (window.isOpen()) {
        for (sf::Event event; window.pollEvent(event);) {
            if (event.type == sf::Event::Closed)
                window.close();
        }
        window.clear();
        triangle.Render(&window);
        window.display();
    }
    return 0;
}
```

5 PS4 Karplus-Strong Guitar String Simulation

5.1 Discussion of the assignment and what was accomplished

The Karplus-Strong algorithm simulates the plucking of a guitar string. The vibration of the string is simulated by maintaining a ring buffer of N samples: the algorithm repeatedly deletes the first sample from the buffer and adds to the end of the buffer with the average of the deleted sample and the first sample, scaled by an energy decay factor of 0.996. This assignment takes the sound samples created to implement a piano on the keyboard that makes guitar sounds.

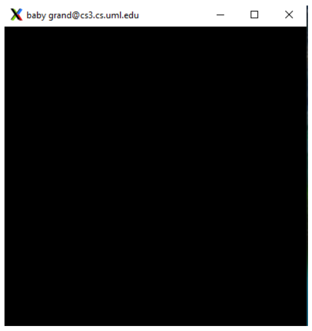


Figure 5: Window to indicate software is running

5.2 What I already knew

I knew how I would implement the circular buffer that would handle the string sounds. Along with the object oriented design concepts, we needed the sfml event class for the piano keystrokes, as well as some other data structures tied together with shared pointers.

5.3 What I learned

I didn't know anything about electronic sound synthesis before beginning this project, so there was a learning curve to understanding the material prior to implementation. Karplus-Strong synthesizes a new waveform from the existing circular buffer, and modifies the last sample by averaging.

5.4 Algorithm, data structures, or object-oriented designs central to the assignment?

Implementing the circular buffer as a queue allowed me to rely on the queue methods to write basic functions. The guitar plucking function uses a standard uniform distribution, along with the system clock and the Mersenne Twister algorithm to reload a new circular buffer. A trio of vectors makes the event driven keyboard possible.

5.5 Challenges

The biggest hurdle was coming to grips with the sound synthesis process before implementation. Equally hard was building the sound generation functions and vector structures that make the keyboard possible.

5.6 Does it work?

Rock on.

5.7 Comments

Keyboard layout is "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ".

5.8 Codebase

5.8.1 Makefile

```
CC = g++ --std=c++17
LIBR = -lboost_unit_test_framework -lsfml-graphics -lsfml-system -lsfml-window
    -lsfml-audio
CFLAGS = -Wall -Werror -pedantic

all:KSGuitarSim
KSGuitarSim: CircularBuffer.o StringSound.o KSGuitarSim.o test.o lint
 $(CC) $(CFLAGS) -o KSGuitarSim CircularBuffer.o StringSound.o KSGuitarSim.o $
    (LIBR)
KSGuitarSim.o: KSGuitarSim.cpp
 $(CC) $(CFLAGS) -c KSGuitarSim.cpp $(LIBR)
CircularBuffer.o: CircularBuffer.cpp
 $(CC) $(CFLAGS) -c CircularBuffer.cpp $(LIBR)
StringSound.o: StringSound.cpp
 $(CC) $(CFLAGS) -c StringSound.cpp $(LIBR)
test.o: test.cpp
 $(CC) $(CFLAGS) -c test.cpp $(LIBR)
clean:
 rm -f *.o
distclean: clean
 rm -f KSGuitarSim
lint:
 cpplint CircularBuffer.cpp CircularBuffer.h test.cpp KSGuitarSim.cpp
    StringSound.cpp StringSound.h
```

5.8.2 CircularBuffer.h

```
//  Copyright <MikeBudney> 2022
#pragma once
#include <stdint.h>
#include <stdbool.h>
#include <iostream>
#include <vector>
#include <queue>
```

```cpp
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
using std::vector;
using std::queue;
class CircularBuffer {
 public:
  CircularBuffer() {}
  ~CircularBuffer() {}
  explicit CircularBuffer(size_t capacity);
  size_t size() const;
  void enqueue(int16_t x);
  int16_t dequeue();
  int16_t peek();
  void Unload();
  bool isEmpty();
  bool isFull();
 private:
  unsigned int buffCap;
  queue<int16_t> circBuff;
};
```

5.8.3 CircularBuffer.cpp

```cpp
//  Copyright <MikeBudney> 2022
#include "CircularBuffer.h"
using std::vector;
using std::invalid_argument;
using std::runtime_error;
using std::queue;

CircularBuffer:: CircularBuffer(size_t capacity) {
  if (capacity < 1)  {
    throw invalid_argument("must be < 1");
  }
  buffCap = capacity;
}

size_t CircularBuffer::size() const {
  return circBuff.size();
}

bool CircularBuffer::isEmpty() {
  return circBuff.size() == 0;
}

int16_t CircularBuffer::dequeue() {
  if (isEmpty())  {
    throw runtime_error("cannot dequeue empty ring");
  }
  int16_t item  = circBuff.front();
  circBuff.pop();
  return item;
}
```

```

int16_t CircularBuffer::peek() {
  if (!isEmpty())  {
    return circBuff.front();
  } else {
    throw runtime_error("peek: can't peek from empty ring");
  }
}

bool CircularBuffer::isFull() {
  return circBuff.size() == buffCap;
}

void CircularBuffer::enqueue(int16_t item) {
  if (this->circBuff.size() >= this->buffCap)  {
    throw runtime_error("cannot add to full ring");
  }
  circBuff.push(item);
}


void CircularBuffer::Unload() {
  while (!this->isEmpty()) {
    this->dequeue();
  }
}
```

5.8.4 StringSound.h

```
//  Copyright <MikeBudney> 2022
#pragma once
#include "CircularBuffer.h"
#include <iostream>
#include <queue>
#include <vector>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
class StringSound {
 public:
  explicit StringSound(double frequency);
  explicit StringSound(vector<sf::Int16> init);
  StringSound(const StringSound& obj) = delete;
  ~StringSound();
  void pluck();
  void tic();
  sf::Int16 sample();
  int time();
 private:
  CircularBuffer* _cb;
  int _time;
};
```

5.8.5 StringSound.cpp

```
//  Copyright <MikeBudney> 2022
#include "StringSound.h"
#include "CircularBuffer.h"
#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include <stdio.h>
#include <cmath>
#include <random>
#include <functional>
#include <stdexcept>
#include <chrono>  //NOLINT
#include <vector>
#include <exception>
#define rangeA -32768
#define rangeB 32767
#define ticNum 0.996
using std::function;
using std::invalid_argument;
using std::out_of_range;
using std::bad_alloc;

StringSound::StringSound(double frequency) {
  if (frequency < 1)  {
    throw invalid_argument("cannot be < 1");
    }
    size_t sizeBuff = ceil(44100 / frequency);
    _cb = new CircularBuffer(sizeBuff);
    _time = 0;
}

int StringSound::time() {
  return _time;
}

StringSound::~StringSound() {
  delete _cb;
}
StringSound::StringSound(vector<sf::Int16> init) {
  _time = 0;
  size_t cap = init.size();
  if (cap < 1) {
    throw invalid_argument("cap cannot be < 1");
  }
  _cb = new CircularBuffer(cap);
  for (unsigned int i = 0; i < cap; i++) {
    _cb->enqueue(init.at(i));
  }
}

void StringSound::pluck() {
  _cb->Unload();
  unsigned int myst = std::chrono::system_clock::now().time_since_epoch().
      count();  //NOLINT
  std::mt19937 gen(myst);
  while (!_cb->isFull()) {
      std::uniform_int_distribution<int16_t> dist(rangeA, rangeB);
      _cb->enqueue(dist(gen));
  }
}

void StringSound::tic() {
  if (_cb->isEmpty()) {
    return;
  }
  int16_t value1 = _cb->dequeue();
  if (_cb->size() < 2) {
    return;
  }
  int16_t value2 = _cb->peek();
  int16_t step = (value2 + value1) / 2.0 * ticNum;
  if (_time % 100 != 0) {
    _cb->enqueue(step);
  } else {
    // deliberately left blank
  }
  _time = _time + 1;
}

sf::Int16 StringSound::sample() {
  if (!_cb->isEmpty()) {
    return _cb->peek();
  } else {
    return 0;
  }
}
```

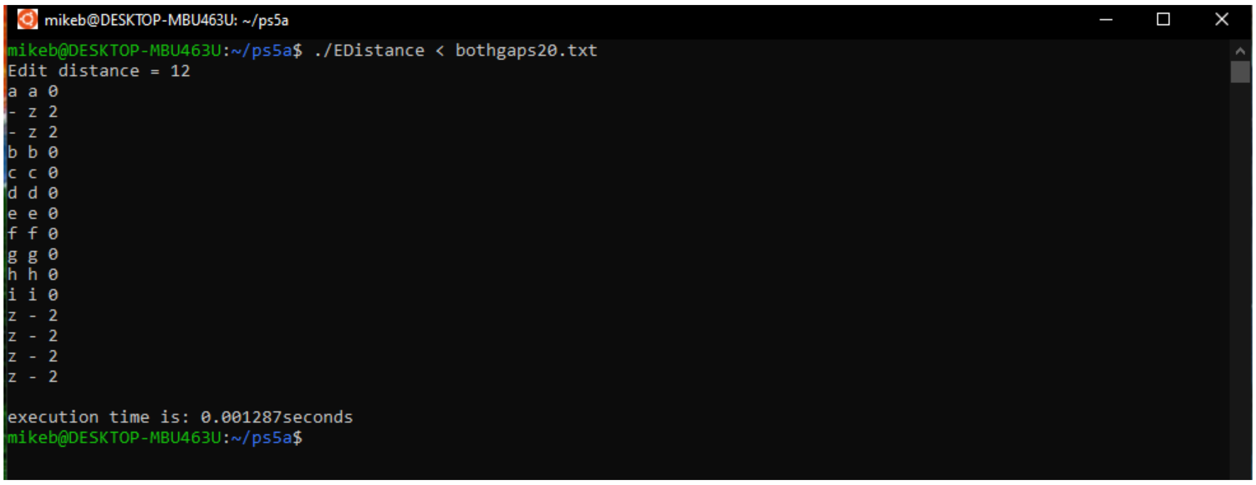
5.8.6 KSGuitarSim.cpp

```
// Copyright <MikeBudney> 2022
#include "CircularBuffer.h"
#include "StringSound.h"
#include <limits.h>
#include <math.h>
#include <iostream>
#include <string>
#include <vector>
#include <memory>
#include <exception>
#include <stdexcept>
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>
#include <SFML/Window.hpp>
#define CONCERT_A 220.0
#define SAMPLES_PER_SEC 44100
#define NUM_KEYS 37
using std::vector;
using std::string;
using std::shared_ptr;
using std::runtime_error;
using std::make_shared;

vector<sf::Int16> buildSample(StringSound& guitar);  //NOLINT

int main(int argc, char **argv)  {
    double goodVibrations = 0;
    string piano = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";
    sf::Event event;
    sf::RenderWindow window(sf::VideoMode(400, 400), "baby grand");
    sf::Sound sound;
    vector<sf::Sound> kboardSound;
    vector<vector<sf::Int16>> samp;
    vector<shared_ptr<sf::SoundBuffer>> kboardBuff;
    for (int i = 0; i < NUM_KEYS; i++)  {
        goodVibrations = 440.0 * pow(2.0, ((i - 24) / 12.0));
        StringSound item(goodVibrations);
        samp.push_back(buildSample(item));
        shared_ptr<sf::SoundBuffer> buffer_ = make_shared<sf::SoundBuffer>();
        kboardBuff.push_back(buffer_);
        if (!buffer_->loadFromSamples(&samp[i][0], samp[i].size(), 1,
        SAMPLES_PER_SEC))  {
            throw runtime_error("failed to load sample");
        }
        sound.setBuffer(*buffer_);
        kboardSound.push_back(sound);
    }
    while (window.isOpen())  {
        while (window.pollEvent(event))  {
            if (event.type == sf::Event::Closed)  {
                window.close();
            }
            if (event.type == sf::Event::TextEntered)  {
                for (int i = 0; i < NUM_KEYS; i++)  {
                    if (piano[i] == static_cast<unsigned char>
                    (event.text.unicode))  {
                        kboardSound[i].play();
                        break;
                    }
                }
            }
            window.clear();
            window.display();
        }
    }
    return 0;
}
vector<sf::Int16> buildSample(StringSound& guitar) {  //NOLINT
    vector<sf::Int16> samp;
    guitar.pluck();
    int seconds = 8;
    for (int i= 0; i < SAMPLES_PER_SEC * seconds; i++) {
        guitar.tic();
        samp.push_back(guitar.sample());
    }
    return samp;
}
```

6 PS5 DNA Sequence Alignment

6.1 Discussion of the assignment and what was accomplished

The goal of this assignment was to take real DNA sequence data and process them as two strings to be compared and aligned. It uses a 2D matrix and a scoring algorithm to identify the optimal alignment of the two DNA sequences. It then outputs the optimal alignment as well as the lowest score that determined that alignment.



```
mikeb@DESKTOP-MBU463U: ~/ps5a
mikeb@DESKTOP-MBU463U:~/ps5a$ ./EDistance < bothgaps20.txt
Edit distance = 12
a a 0
- z 2
- z 2
b b 0
c c 0
d d 0
e e 0
f f 0
g g 0
h h 0
i i 0
z - 2
z - 2
z - 2
z - 2

execution time is: 0.001287seconds
mikeb@DESKTOP-MBU463U:~/ps5a$
```

Figure 6: Sequence alignment

6.2 What I already knew

A genetic sequence is a string formed from a four-letter alphabet: Adenine (A), Thymine (T), Guanine(G), Cytosine (C). I also knew that I would need a 2D matrix structure and that there were a couple options for which algorithm to use for this assignment. Hirschberg's algorithm (1975) uses linear space and quadratic time, as opposed to the O(mn) space and O(mn) time of the Needleman-Wunsch.

6.3 What I learned

Dynamic programming involves taking a problem and dividing it into smaller sub-problems in which the optimal main problem is reliant on the solution to the optimal sub-problems. Learned about the Needleman-Wunsch algorithm for finding an optimal path to solve the alignment problem.

6.4 Algorithm, data structures, or object-oriented designs central to the assignment?

Needleman-Wunsch uses a scoring system with the help of a matrix designed from a vector in this case. It scores the alignment on the basis of match, mismatch, or gap and finds the minimum score (shortest path) to find the optimal alignment of two strings.

6.5 Challenges

Figuring out the logic to score the matrix and find the optimal distances was the biggest challenge. This is an instance where drawing was a valuable exercise to get a full picture down rather than trying to remember everything.

6.6 Does it work?

Yes. Works as intended.

6.7 Comments

This program takes a file redirect to a text file containing two strings to be compared as an argument. It then compares the strings and prints their proper alignment along with the edit distance to achieve the optimal alignment.

6.8 Codebase

6.8.1 Makefile

```
CC = g++ --std=c++17
LIBS = -lsfml-system
CFLAGS = -Wall -Werror -pedantic -ggdb

all:EDistance
EDistance: EDistance.o main.o
	$(CC) $(CFLAGS) -o EDistance EDistance.o main.o $(LIBS)
EDistance.o: EDistance.cpp
	$(CC) $(CFLAGS) -c EDistance.cpp $(LIBS)
main.o: main.cpp
	$(CC) $(CFLAGS) -c main.cpp $(LIBS)
clean:
	rm -f main.o EDistance.o test.o EDistance
lint:
	cpplint main.cpp EDistance.h EDistance.cpp
```

6.8.2 EDistance.h

```
// Copyright [2023] <MikeBudney>
#pragma once
#include <iostream>
#include <algorithm>
#include <vector>
#include <ctime>
#include <string>
using std::size_t;
using std::string;
using std::vector;
using std::istream;
using std::ostream;

```

```cpp
class EDistance {
 public:
  EDistance(string one, string two);
  int optDistance();
  static int penalty(char a, char b);
  static int min(int a, int b, int c);
  int& editValue(int r, int c);
  string alignment();
  // friend istream & operator >> (istream &in,  string &str);
  // friend ostream & operator << (ostream &out, const string &str);
 private:
  string str1;
  string str2;
  size_t column;
  size_t row;
  vector<int> matrix;
};
```

6.8.3 EDistance.cpp

```cpp
// Copyright [2023] <MikeBudney>
#include "EDistance.h"
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <algorithm>

using std::sort;
using std::string;
using std::stringstream;
using std::cout;
using std::vector;
using std::size_t;
using std::endl;

/*
istream & operator >> (istream &in,  string &str)  {
  in >> str;
    return in;
}

ostream & operator << (ostream &out, const string &str) {
    out << str;
    return out;
}
*/
EDistance::EDistance(string one, string two)  {
  str1 = one;
  str2 = two;
  row = one.length() + 1;
  column = two.length() + 1;
  matrix.resize((row) * (column));
}
```

```
int EDistance::min(int a, int b, int c) {
  vector<int> v {a, b, c};
  sort(v.begin(), v.end(), [v](const int& a, const int& b)  {
        return a < b;
    });
  sort(v.begin(), v.end(), [v](const int& b, const int& c)  {
        return b < c;
    });
    return v[0];
}
int& EDistance::editValue(int r, int c) {
    int num = 0;
    num = column * r + c;
    return matrix.at(num);
}
int EDistance::penalty(char a, char b)  {
  if (a == b) {
    return 0;
  } else {
      return 1;
    }
}
int EDistance::optDistance()  {
  for (size_t i = 0; i < row; i++)  {
      editValue(i, (column - 1)) = 2 * (row - 1 - i);
    }
  for (size_t j = 0; j < column; j++) {
      editValue((row - 1), j) = 2 * (column - 1 - j);
    }
  for (size_t i = row - 1; i > 0; i--) {
      for (size_t j = column - 1; j > 0; j--) {
        char a = str1[i - 1];
        char b = str2[j - 1];
        editValue(i - 1, j - 1) = min(editValue(i, j) + penalty(a, b),
         (editValue(i, j - 1) + 2), editValue(i - 1, j) + 2);
      }
  }
return editValue(0, 0);
}
string EDistance::alignment() {
  size_t i = 0;
  size_t j = 0;
  stringstream buffer;
  while (editValue(i, j) != 0) {
    if (i == row - 1)  {
      buffer << "- " << str2[j] << " 2\n";
      j++;
    } else if (j == column - 1)  {
        buffer << str1[i] << "- 2\n";
        i++;
      } else  {
          if (editValue(i, j) == editValue(i + 1, j) + 2)  {
            buffer << str1[i] << " - 2\n";
            i++;
            } else if (editValue(i, j) == editValue(i, j + 1) + 2) {
                buffer << "- " << str2[j] << " 2\n";
                j++;
            } else {
                buffer << str1[i] << " " << str2[j] << " " <<
                penalty(str1[i], str2[j]) << "\n";
                i++;
                j++;
            }
        }
    }
    return buffer.str();
}
```

6.8.4 main.cpp

```
#include "EDistance.h"
#include <iostream>
#include <SFML/System.hpp>

using sf::Clock;
using sf::Time;
using std::cout;
using std::cin;
using std::endl;

int main(int argc, char** argv){
  Clock clock;
  Time t;
  string x, y;
  cin >> x >> y;
  EDistance Edist(x, y);
  cout << "Edit distance = " << Edist.optDistance() << endl <<
  Edist.alignment() << endl;
  t = clock.getElapsedTime();
  cout << "execution time is: " << t.asSeconds() << "seconds \n";
  return 0;
}
```

7 PS6 Markov Model Random Writer

7.1 Discussion of the assignment and what was accomplished

An order-k Markov model uses strings of k-letters to predict text, these are called k-grams. This software program generates psuedo-random text using a markov model to create a distribution of occurrences of characters following each k-gram. In this assignment we used Mark Twain's *Tom Sawyer* as the input to test the model.

7.2 Random Text Examples

7.2.1 K-gram length of 3

"' THERE could with verathe sure to pround toward, Tom, by the made ally ind woulder his best. Then carrief, and you seemer boys with hight a come ove, so to lat. No, thankful little the pirath a his wel "'

7.2.2 K-gram length of 4

"'THE really over decided, and felt that be anybody'll forty-five escape, percritish with friend to wished a gloom. "I'd fellowing there curtain on the been for that the said Walters and a removal; but"'

7.3 What I already knew

Claude Shannon was a genius and that this would require some clever statistical coding skills. I also knew I needed a map or dictionary to track the occurrences of characters to create proper distributions.

7.4 What I learned

I first had to gain an understanding of how Markov models were intended to operate and then learn to apply the pattern recognition and statistical predictive modeling. I gained a deeper appreciation for maps and iterators and dug deeper into C++ standard library functionality. This assignment forces you to employ a number of data structures in concert with each other to achieve the desired result.

7.5 Algorithm, data structures, or object-oriented designs central to the assignment?

Krand(), the function responsible for determining which random character should appear next after the given k-gram would need uniform int distribution and random device from the standard library to create an equal likelihood integer distribution and generate the numbers for order 0 models. Then modifying the distribution with the map's character frequency data and generating a new pseudo-random number based on that distribution to create a new random character.

7.6 Challenges

This was another assignment that benefited greatly from doing some forward planning as opposed to writing a function and chasing your tail. I had some trouble writing some of the functions that accessed map data, it was sort of confusing since my map contained both strings and vectors as its key-value pairs.

7.7 Does it work?

Yes. Can also be humorous.

7.8 Comments

Takes the first k-gram of characters as first argument. Number of letters to generate based on the k-gram as second argument. The third argument is a redirect to a text file you would like to emulate.

7.9 Codebase

7.9.1 Makefile

```
CC = g++ --std=c++17
LIBS = -lsfml-graphics -lsfml-system -lsfml-window -lboost_unit_test_framework
CFLAGS = -g -Wall -Werror -pedantic

all:TextWriter Testrun
TextWriter: TextWriter.o RandWriter.o lint
	$(CC) $(CFLAGS) -o TextWriter TextWriter.o RandWriter.o $(LIBS)
Testrun: test.o RandWriter.o
	$(CC) $(CFLAGS) -o Testrun test.o RandWriter.o $(LIBS)
RandWriter.o: RandWriter.cpp
	$(CC) $(CFLAGS) -c RandWriter.cpp $(LIBS)
TextWriter.o: TextWriter.cpp
	$(CC) $(CFLAGS) -c TextWriter.cpp $(LIBS)
test.o: test.cpp
	$(CC) $(CFLAGS) -c test.cpp $(LIBS)
clean:
	rm -f *.o
lint:
	cpplint RandWriter.h RandWriter.cpp TextWriter.cpp
```

7.9.2 RandWriter.h

```
// Copyright <MikeBudney> 2022
#pragma once
#include <iostream>
#include <map>
#include <vector>
#include <string>
#include <numeric>
using std::vector;
using std::map;
using std::ostream;
using std::string;

class RandWriter {
 public:
  RandWriter() {}
  RandWriter(string text, int k);
  int freq(string kgram) const;
  int freq(string kgram, char c) const;
  friend ostream& operator<<(ostream& output, RandWriter& rWriter);
  string generate(string kgram, int L);
  ~RandWriter() {}
  int orderK() const;
  char kRand(string kgram);
 private:
  map<string, vector<char>> rwMap;
  string rwString;
  int rwOrder;
};
```

7.9.3 RandWriter.cpp

```
// Copyright <MikeBudney> 2022
#include "RandWriter.h"
#include <stdexcept>
#include <random>
#include <functional>
#include <vector>
#include <algorithm>
#include <numeric>
using std::cout;
using std::endl;
using std::map;
using std::runtime_error;
using std::ostream;
using std::uniform_int_distribution;
using std::string;
using std::random_device;
using std::accumulate;

RandWriter::RandWriter(string text, int k) {
  if (k < 0)  {
    throw runtime_error("cannot be negative");
  }
  rwString = text;
  rwOrder = k;
  string loop = rwString + rwString.substr(0, rwOrder);
  auto size = rwString.length();
  string transient = "";
  for (unsigned int i = 0; i < size; i++) {
    transient = loop.substr(i, rwOrder);
    char add = loop[i+rwOrder];
    rwMap[transient].push_back(add);
  }
}
ostream& operator<<(ostream& output,  RandWriter& rWriter) {
  for (const auto& iter : rWriter.rwMap) {
    output << iter.first << " ";
    for (size_t i = 0; i < (iter.second.size()); i++) {
      if (i == (iter.second.size() - 1))  {
       output << iter.second[i] << " ";
      } else  {
       output << iter.second[i] << ", ";
    }
    output << endl;
    }
  }
  return output;
}
int RandWriter::freq(string kgram, char c) const {
  if (static_cast<int>(kgram.length()) != rwOrder)  {
    throw runtime_error("length != order");
  }
  if (rwOrder  == 0) {
    string transient;
    transient = c;
    return rwMap.at(transient).size();
  } else {
    int temp = 0;
    for (int i = 0; i < static_cast<int>(rwMap.at(kgram).size()); i++) {
      if (rwMap.at(kgram)[i] == c)  {
       temp++;
      }
    }
    return temp;
  }
}
int RandWriter::freq(string kgram) const {
  if (static_cast<int>(kgram.length()) != rwOrder)  {
    throw runtime_error("length != order");
  }  else {
      return rwMap.at(kgram).size();
  }
}
int RandWriter::orderK() const {
  return rwOrder;
}
char RandWriter::kRand(string kgram) {
  if (static_cast<int>(kgram.length()) != rwOrder)  {
    throw runtime_error("length != order");
  }
  random_device rDevice;
  if (rwOrder == 0) {
    vector<int> num;
    int total = 0;
    total = total + rwMap.at(kgram).size();
    uniform_int_distribution<int> distance(0, total -1);
    int transient = distance(rDevice);
    int i = 0;
    for (auto const& iter : rwMap) {
      if (transient < num[i]) {
       return iter.second[0];
      }
      transient -= num[i];
      i++;
    }
  }
  if (rwMap.at(kgram).size() <= 0)  {
    throw runtime_error("DNE");
  }
  uniform_int_distribution<int> distance(0, (rwMap.at(kgram).size() - 1));
  return rwMap.at(kgram)[distance(rDevice)];
}
string RandWriter::generate(string kgram, int L) {
  if (static_cast<int>(kgram.size()) != rwOrder) {
    throw runtime_error("size != order");
  }
  string result = kgram;
  if (rwOrder == 0) {
    for (int i = 0; i < L; i++) {
      result.push_back(kRand(""));
    }
    return result;
  }
  if (rwMap[kgram].size() <= 0) {
    throw runtime_error("size <= 0");
  }
  while (static_cast<int>(result.size()) < L) {
    result.push_back(kRand(kgram));
    kgram = result.substr(result.size() - rwOrder, rwOrder);
  }
  return result;
}
```

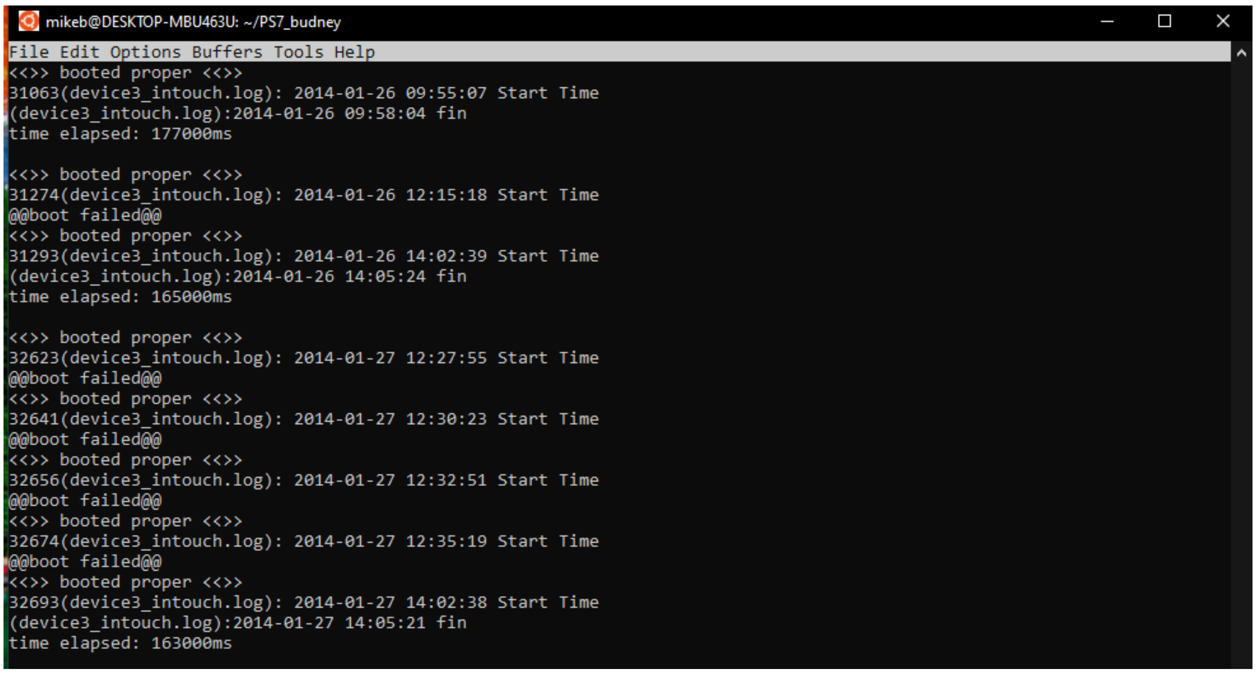
7.9.4 TextWriter.cpp

```
// Copyright <MikeBudney> 2022
#include "RandWriter.h"
#include <iostream>
#include <stdexcept>
#include <functional>
#include <algorithm>
#include <numeric>
using std::for_each;
using std::accumulate;
using std::string;
using std::cin;
using std::endl;
using std::cout;

int main(int argc, char** argv) {
  if (argc != 3)  {
    return -1;
  }
  int k = atoi(argv[1]);
  int L = atoi(argv[2]);
  string prose = "";
  string str;
  while (getline(cin, str)) {
    prose += str + "\n";
  }
  /*
  vector<char> s;
  vector<char> kgram = accumulate(str[0], str[k], s, [&]
   (int i) -> vector<char> {
   s.push_back(str[i]);
   return s;});*/
  RandWriter rWriter(prose, k);
  string gram = prose.substr(0, k);
  cout << rWriter.generate(gram, L) << endl;
return 0;
}
```

8 PS7 Kronos Time Clock Parsing w/ Regex

8.1 Discussion of the assignment and what was accomplished

This assignment uses the regular expression library (regex) to parse the time clock's output log for regex expressions. It identifies when the clock boots properly or fails to do so based on the log. It then generates a report based on its findings named after the initial input file.


mikeb@DESKTOP-MBU463U: ~/PS7_budney
File Edit Options Buffers Tools Help

```
<<>> booted proper <<>>
31063(device3_intouch.log): 2014-01-26 09:55:07 Start Time
(device3_intouch.log):2014-01-26 09:58:04 fin
time elapsed: 177000ms

<<>> booted proper <<>>
31274(device3_intouch.log): 2014-01-26 12:15:18 Start Time
@@boot failed@@
<<>> booted proper <<>>
31293(device3_intouch.log): 2014-01-26 14:02:39 Start Time
(device3_intouch.log):2014-01-26 14:05:24 fin
time elapsed: 165000ms

<<>> booted proper <<>>
32623(device3_intouch.log): 2014-01-27 12:27:55 Start Time
@@boot failed@@
<<>> booted proper <<>>
32641(device3_intouch.log): 2014-01-27 12:30:23 Start Time
@@boot failed@@
<<>> booted proper <<>>
32656(device3_intouch.log): 2014-01-27 12:32:51 Start Time
@@boot failed@@
<<>> booted proper <<>>
32674(device3_intouch.log): 2014-01-27 12:35:19 Start Time
@@boot failed@@
<<>> booted proper <<>>
32693(device3_intouch.log): 2014-01-27 14:02:38 Start Time
(device3_intouch.log):2014-01-27 14:05:21 fin
time elapsed: 163000ms
```

Figure 7: Report log for boot sequences on device 3 generated by the assignment

8.2 What I already knew

A little bit about regex expressions and their syntax. That though this assignment may seem mundane compared to some of the others, that it is incredibly handy in a variety of use cases.

8.3 What I learned

A more in-depth understanding of the boost regex library and why it is so useful. Also learned about the boost Posix time class and its methods in order to complete the assignment.

8.4 Algorithm, data structures, or object-oriented designs central to the assignment?

With the boost regex library you can create search strings by type of character rather than having to be explicit about what you are searching for. This allows you to use the boost regex library to search for time stamps with varying values. Searching the log files for boot start and boot completions was simpler since they are fixed messages. If the new boot message was found after the initial it indicated a boot failure. Or if no success message was found before the file ended, this was also an indication of failure.

8.5 Challenges

Mostly getting the regex expressions correct so that it would properly pick up the time stamps.

8.6 Does it work?

It does. No issues.

8.7 Comments

This was the last assignment of the semester. Short and sweet. If you made it this far, thanks for checking out my portfolio!

8.8 Codebase

8.8.1 Makefile

```
CC = g++ --std=c++17
LIBS = -lboost_regex -lboost_date_time
CFLAGS = -g -Wall -Werror -pedantic

all:PS7
PS7:kronos.o lint
	$(CC) $(CFLAGS) -o PS7 kronos.o $(LIBS)
kronos.o:kronos.cpp
	$(CC) $(CFLAGS) -c kronos.cpp $(LIBS)
clean:
	rm -f PS7 *.o
lint:
	cpplint kronos.cpp
```

8.8.2 Kronos.cpp

```
//  Copyright <MikeBudney> 2022
#include <fstream>
#include <iostream>
#include <string>
#include "boost/date_time/posix_time/posix_time.hpp"
#include <boost/regex.hpp>
using std::string;
using std::endl;
using std::ofstream;
using std::ifstream;
using std::getline;
```

```cpp
using std::endl;
using std::invalid_argument;
using std::runtime_error;
using boost::posix_time::ptime;
using boost::posix_time::time_from_string;
using boost::posix_time::time_duration;
using boost::regex_search;
using boost::regex;
using boost::smatch;
int main(int argc, char** argv) {
  if (argc != 2) {
    throw invalid_argument("invalid # of arguments");
  }
  string readFrom(argv[1]);
  string writeTo = readFrom + ".rpt";
  ofstream writeFile(writeTo);
  ifstream readFile(readFrom);
  if (!readFile.is_open()) {
    throw runtime_error("read file did not open");
  }
  if (!writeFile.is_open()) {
    throw runtime_error("write file did not open");
  }
  bool bootFlag = 0;
  regex timeStamp{"\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}"};
  regex bootOpen{"\\(log\\.c\\.166\\) server started"};
  regex bootFin{"oejs\\.AbstractConnector:Started SelectChannelConnector"};
  smatch regexMatch;
  auto getTime = [timeStamp, &regexMatch](string target)-> string {
      if (regex_search(target, regexMatch, timeStamp)) {
        return regexMatch.str();
      } else {
          return nullptr;
        }
    };
  string toParse;
  time_duration clock;
  ptime tSubZero;
  ptime tSubN;
  unsigned int lineCounter = 0;
  while (getline(readFile, toParse)) {
    lineCounter = lineCounter + 1;
    if (regex_search(toParse, bootOpen)) {
      tSubZero = time_from_string(getTime(toParse));
      if (bootFlag) {
       writeFile << "@@boot failed@@" << endl;
      }
      writeFile << "<<>> booted proper <<>>" << endl
      << lineCounter << "(" << readFrom << "):"
      << getTime(toParse) << " Start Time" << endl;
      bootFlag = 1;
    }
    if (regex_search(toParse, bootFin)) {
      tSubN = time_from_string(getTime(toParse));
```

```
        clock = tSubN - tSubZero;
        bootFlag = 0;
        writeFile << "(" << readFrom << "):" << getTime(toParse)
        << " fin" << endl << "time elapsed: "
        << clock.total_milliseconds() << "ms" << endl;
      }
    }
    readFile.close();
    writeFile.close();
    return 0;
}
```