**PROJECT REPORT**

*Data Structures And Algorithms (CSL 221)*



**BS(CS)-4A/B/C**

*Group Members*

|  |  |
| --- | --- |
| **Name** | **Enrollment** |
| 1. Moaaz Rafique | **02-134192-064** |
| 2. Shaheer Suleman | **02-134192-009** |
| 3.Ammar Anis | **02-134192-073** |
| 4. |  |

**Submitted to:**

**Saba Imtiaz**

**BAHRIA UNIVERSITY KARACHI CAMPUS**

*Department of Computer Science*

**Note: Project Report carries *10 Marks*. Report should be written in your own words. *Plagiarism* is not allowed.**

*Module wise Work/Task Distribution*

|  |  |  |
| --- | --- | --- |
| **Name** | **Enrollment** | **Task** |
| 1. Moaaz Rafique | **02-134192-064** | **Making and Managing Graph** |
| 2.Shaheer Suleman | **02-134192-009** | **handling inputs and outputs** |
| 3.Ammar Anis | **02-134192-073** | **Making Gui** |
| 4. |  |  |

**TABLE OF CONTENT**

1. Introduction.......................................................................................................................3

2. Requirements Specifications.............................................................................................5

3. System Implementation.....................................................................................................6

4. Testing................................................................................................................................19

5. Sample Screenshots Of System..........................................................................................20

**Google drive link**

**https://drive.google.com/drive/folders/1rzXuIGYNeUPC3k1hvTBarFC21g1lOb2A?usp=sharing**

**INTRODUCTION**

Project Description[1.1]

A program which will help people understand their math equations not only that it will show what is their equation is doing for example if someone writes y=x he will see the graphical solution of the equation which will help him understand what is going on in the equation. Not only that we will add the feature that allow us to put equations simultaneously hence they can compare the equations. This will help teacher and students likewise especially teaching online.

We will give the services to remove add using link list and buttons. A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers, basically a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

Active or disable options with the help of check boxes(queue). Basically, we are making a both polar and Cartesian co-ordinates system. By using sliders, we will also make zooming function which will help users to see clearly. We will also be adding priority hence we will be using swapping (The swap () function is used to swap two numbers. By using this function, you do that by adding third variable to swap two numbers.)

It will also have polar to cartesian conversion or vice versa y = r \* sin(θ) If it contains rs and θs, it is in polar form. If it contains xs and ys, it is in rectangular form, hence after finding out then substitute then Finally, we combine like terms and simplify the equation.

For example

**Cartesian form** y=f(x)=x2

U have to convert in polar formx=rcosθ and y=rsinθ, Divide both sides by r, Divide both sides by cos2θ, Hence u will get your answer which **is r=tan(θ)⋅sec(θ)**

Graph design will be made using x scale and y scale as grid distance but the users can edit that if they want according to their own requirements. graph will be implements using linked list and range of variables x,y,z. Only active graph will be shown and graph will have circles (points) and line will represent connection.

Scope of the Project[1.2]

A program designed to graph mathematical functions in both polar and Cartesian co-ordinates. The program should able to seamlessly add equations to and remove equations from the program. It will be able to display multiple equations at the same time with some priority setup. The program can be useful for mathematic teachers, students and researchers in understanding how graphs work and how to visualize solutions to different problems.

Modules in Project[1.3]

* The program will use C++ as the programming language.
* SFML will be used to draw all the graphics for the graph and graph sheet.
* TGUI: It is a cross-platform modern C++ GUI library. Although TGUI has always been a library for SFML, since TGUI 0.9 you can use custom backends and an experimental backend is provided that uses SDL2 with OpenGL 4., an SFML based GUI library will be used for user interface.
* ExprsTK: an expression parsing library will be used for equation parsing in the program. The C++ Mathematical Expression Toolkit Library (ExprTk) is a simple to use, easy to integrate and extremely efficient run-time mathematical expression parser and evaluation engine. ExprTk supports numerous forms of functional, logical and vector processing semantics and is very easily extendible.

Project Features[1.4]

It will provide a graphical representation of an equation. Plus, you can either add or remove multiple equations in time.

It will also have polar to cartesian conversion or vice versa y = r \* sin(θ)

If it contains rs and θs, it is in polar form. If it contains xs and ys, it is in rectangular form, hence after finding out then substitute then Finally, we combine like terms and simplify the equation.

For example

**Cartesian form** y=f(x)=x2

U have to convert in polar formx=rcosθ and y=rsinθ, Divide both sides by r, Divide both sides by cos2θ,Hence u will get your answer which **is r=tan(θ)⋅sec(θ)**

**REQUIREMENTS SPECIFICATION**

Hardware Requirements[2.1]

* Dual Core x86 Processor or above
* 1 GB RAM minimum or above.
* Minimum 40 GB Hardware required.

Software Requirements[2.2]

* Visual Studio
* TGUI – GUI Builder
* Microsoft Windows 7 or Above required

**SYSTEM IMPLEMENTATION**

Introduction[3.1]

A program which will show a graphical solution of the equation which will help user understand what is going on in the equation.

Code[3.2]

**Project.cpp**

#include "Source.h"

int noOfEquations = 2;

/// <summary>

/// Functions and variables for grid

/// </summary>

void removegt(tgui::Group::Ptr equationGroup, EquationList\* eqLink, bool para = false) {

equationGroup->setVisible(false);

if (para) {

tgui::CheckBox::Ptr checkBox = equationGroup->get<tgui::CheckBox>("CheckBox2");

checkBox->setChecked(false);

}

else {

tgui::CheckBox::Ptr checkBox = equationGroup->get<tgui::CheckBox>("CheckBox1");

checkBox->setChecked(false);

}

eqLink->removeEquationsGroup(equationGroup, para);

if (noOfEquations > 1)noOfEquations--;

}

int main()

{

sf::RenderWindow window(sf::VideoMode(800, 600), "Graph It");

window.setFramerateLimit(60);

tgui::Gui gui(window);

float zoom = 1;

gui.loadWidgetsFromFile("graphIt.txt");

bool a = false;

tgui::Button::Ptr zoomInButton = gui.get<tgui::Button>("zoomInButton");

tgui::Button::Ptr zoomOutButton = gui.get<tgui::Button>("zoomOutButton");

tgui::ScrollablePanel::Ptr sPanel = gui.get<tgui::ScrollablePanel>("sPanel");

EquationList\* eqLink = new EquationList(sPanel);

tgui::Button::Ptr addEquation = gui.get<tgui::Button>("addEquation");

tgui::Group::Ptr equationGroup = sPanel->get<tgui::Group>("equations");

tgui::EditBox::Ptr equationBox = equationGroup->get<tgui::EditBox>("equationBox");

tgui::CheckBox::Ptr active = equationGroup->get<tgui::CheckBox>("CheckBox1");

tgui::Button::Ptr removeEquation = equationGroup->get<tgui::Button>("removeEquation");

removeEquation->connect("MouseReleased", removegt, equationGroup, eqLink, false);

tgui::Group::Ptr equationPara = sPanel->get<tgui::Group>("equationPara");

tgui::Button::Ptr addPara = sPanel->get<tgui::Button>("addPara");

tgui::Button::Ptr removePara = gui.get<tgui::Button>("removePara");

removePara->connect("MouseReleased", removegt, equationPara, eqLink, true);

equationBox->setDefaultText("Enter f(x)");

equationBox->setText("x^3");

eqLink->addEquation(equationGroup);

eqLink->addEquation(equationPara, true);

addPara->connect("MouseReleased",

[&]() {

tgui::Group::Ptr equationGroup1 = tgui::Group::create();

equationGroup1 = tgui::Group::copy(equationPara);

equationGroup1->setVisible(true);

tgui::EditBox::Ptr number = equationGroup1->get<tgui::EditBox>("numbers");

noOfEquations++;

number->setText(std::to\_string(noOfEquations));

tgui::EditBox::Ptr equationBox1 = equationGroup1->get<tgui::EditBox>("xValue");

equationBox1->setDefaultText("Enter f(7)");

equationBox1->setText("t");

equationGroup1->setPosition(equationGroup->getPosition().x, equationGroup->getPosition().y + 95 \* ((std::stoi((std::string)number->getText())) - 1));

tgui::EditBox::Ptr equationBox2 = equationGroup1->get<tgui::EditBox>("yValue");

equationBox2->setDefaultText("Enter f(7)");

equationBox2->setText("t^2");

equationGroup1->setPosition(equationGroup->getPosition().x, equationGroup->getPosition().y + 95 \* ((std::stoi((std::string)number->getText())) - 1));

tgui::Button::Ptr removeEquation1 = equationGroup1->get<tgui::Button>("removePara");

removeEquation1->connect("MouseReleased", removegt, equationGroup1, eqLink, true);

sPanel->add(equationGroup1);

eqLink->addEquation(equationGroup1, true);

});

addEquation->connect("MouseReleased",

[&]() {

tgui::Group::Ptr equationGroup1 = tgui::Group::create();

equationGroup1 = tgui::Group::copy(equationGroup);

equationGroup1->setVisible(true);

tgui::EditBox::Ptr number = equationGroup1->get<tgui::EditBox>("number");

noOfEquations++;

number->setText(std::to\_string(noOfEquations));

tgui::EditBox::Ptr equationBox1 = equationGroup1->get<tgui::EditBox>("equationBox");

equationBox1->setDefaultText("Enter f(7)");

equationBox1->setText("x^1");

equationGroup1->setPosition(equationGroup->getPosition().x, equationGroup->getPosition().y + 95 \* ((std::stoi((std::string)number->getText())) - 1));

tgui::Button::Ptr removeEquation1 = equationGroup1->get<tgui::Button>("removeEquation");

removeEquation1->connect("MouseReleased", removegt, equationGroup1, eqLink, false);

sPanel->add(equationGroup1);

eqLink->addEquation(equationGroup1);

});

zoomInButton->connect("MouseReleased",

[&]() {

if (zoom > 0)zoom -= .05;

});

zoomOutButton->connect("MouseReleased",

[&]() {

zoom += .05;

});

while (window.isOpen())

{

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

window.close();

gui.handleEvent(event);

}

/\*if (sf::Mouse::isButtonPressed(sf::Mouse::Left)) {

addNode(&window);

}\*/

window.clear();

eqLink->display(&window, zoom);

gui.draw();

window.display();

}

return 0;

}

**Source.h**

#include <TGUI/TGUI.hpp>

#include "exprtk.hpp"

#include<iostream>

class EquationList

{

public:

struct GroupNode {

tgui::Group::Ptr g;

sf::String number;

sf::Color color;

bool para;

GroupNode\* next;

GroupNode\* prev;

};

GroupNode g;

struct GraphLinkLists

{

sf::CircleShape c;

GraphLinkLists\* next;

};

EquationList(tgui::ScrollablePanel::Ptr sPanel) {

this->head = NULL;

this->sPanel = sPanel;

}

~EquationList();

void removeEquationsGroup(tgui::Group::Ptr gt, bool para = false) {

sf::String number;

if (para) number = gt->get<tgui::EditBox>("numbers")->getText();

else number = gt->get<tgui::EditBox>("number")->getText();

std::cout << (std::string) number << "\n";

GroupNode\* del = findNode(number);

if (head == NULL || del == NULL)

return;

GroupNode\* temp = del;

while (temp->next != NULL) {

temp->g->setPosition(temp->g->getPosition().x, temp->g->getPosition().y - 95);

temp = temp->next;

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

}

temp->g->setPosition(temp->g->getPosition().x, temp->g->getPosition().y - 95);

/\* If node to be deleted is head node \*/

if (head == del)

head = del->next;

/\* Change next only if node to be

deleted is NOT the last node \*/

if (del->next != NULL)

del->next->prev = del->prev;

/\* Change prev only if node to be

deleted is NOT the first node \*/

if (del->prev != NULL)

del->prev->next = del->next;

displayEquations();

/\* Finally, free the memory occupied by del\*/

free(del);

return;

}

void addEquation(tgui::Group::Ptr gt, bool para = false) {

GroupNode\* eq = new GroupNode();

eq->g = gt;

eq->para = para;

eq->color = sf::Color::Color(rand() % 255, rand() % 255, rand() % 255);

tgui::EditBox::Ptr number;

if(para)number = gt->get<tgui::EditBox>("numbers");

else number = gt->get<tgui::EditBox>("number");

eq->number = number->getText();

if (head == NULL)

{

head = eq;

}

else {

GroupNode\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = eq;

eq->prev = temp;

}

}

GroupNode\* findNode(sf::String gt) {

if (head == NULL) {

return NULL;

}

if (head->number==gt)

return head;

GroupNode\* temp = head;

while (temp->next != NULL) {

if (temp->number == gt)

return temp;

temp = temp->next;

}

if (temp->number == gt)

return temp;

return NULL;

}

void display(sf::RenderWindow\* window, float z) {

this->zoom = z;

drawGrid(window);

//displayEquations();

if (head == NULL)

{

return;

}

GroupNode\* temp = this->head;

while (temp->next != NULL) {

if (!temp->para) {

tgui::CheckBox::Ptr checkBox = temp->g->get<tgui::CheckBox>("CheckBox1");

if(checkBox->isChecked())

{

tgui::EditBox::Ptr textBox = temp->g->get<tgui::EditBox>("equationBox");

function\_x<double>(textBox, temp->color);

displayList(window, manualNodeHead, temp->color);

deleteList(&manualNodeHead);

}

}

else {

tgui::CheckBox::Ptr checkBox = temp->g->get<tgui::CheckBox>("CheckBox2");

if (checkBox->isChecked())

{

tgui::EditBox::Ptr textBox1 = temp->g->get<tgui::EditBox>("xValue");

tgui::EditBox::Ptr textBox2 = temp->g->get<tgui::EditBox>("yValue");

function\_x\_y<double>(textBox1, textBox2, temp->color);

displayList(window, manualNodeHead, temp->color);

deleteList(&manualNodeHead);

}

}

temp = temp->next;

}

if (!temp->para) {

tgui::CheckBox::Ptr checkBox = temp->g->get<tgui::CheckBox>("CheckBox1");

if (checkBox->isChecked())

{

tgui::EditBox::Ptr textBox = temp->g->get<tgui::EditBox>("equationBox");

function\_x<double>(textBox, temp->color);

displayList(window, manualNodeHead, temp->color);

deleteList(&manualNodeHead);

}

}

else {

tgui::CheckBox::Ptr checkBox = temp->g->get<tgui::CheckBox>("CheckBox2");

if (checkBox->isChecked())

{

tgui::EditBox::Ptr textBox1 = temp->g->get<tgui::EditBox>("xValue");

tgui::EditBox::Ptr textBox2 = temp->g->get<tgui::EditBox>("yValue");

function\_x\_y<double>(textBox1, textBox2, temp->color);

displayList(window, manualNodeHead, temp->color);

deleteList(&manualNodeHead);

}

}

}

void displayEquations() {

if (head == NULL) {

return;

}

GroupNode\* temp = this->head;

while (temp->next != NULL) {

std::cout << temp->g->getAbsolutePosition().y<< "->";

temp = temp->next;

}

std::cout << temp->g->getAbsolutePosition().y << "\n";

}

private:

int x\_scale = 50;

int y\_scale = 50;

int x\_loc = 265;

int y\_loc = 80;

int thickness = 2;

int length = 500;

int gridsize = 10;

int graphNodeSize = 2;

float zoom = 1;

GraphLinkLists\* manualNodeHead = NULL;

GroupNode\* head;

tgui::ScrollablePanel::Ptr sPanel;

void drawGrid(sf::RenderWindow\* w) {

sf::RectangleShape line;

line.setFillColor(sf::Color::Color(255, 255, 255, 50));

for (int i = 0; i <= gridsize; i++)

{

line.setPosition(i \* x\_scale + x\_loc, y\_loc);

line.setSize(sf::Vector2f(thickness, length));

w->draw(line);

line.setPosition(x\_loc, i \* y\_scale + y\_loc);

line.setSize(sf::Vector2f(length, thickness));

w->draw(line);

}

}

void addNode(sf::RenderWindow\* w) {

GraphLinkLists\* g = new GraphLinkLists;

g->c.setPosition(sf::Mouse::getPosition(\*w).x + graphNodeSize, sf::Mouse::getPosition(\*w).y + graphNodeSize);

g->c.setRadius(graphNodeSize);

g->next = NULL;

if (manualNodeHead == NULL) {

manualNodeHead = g;

}

else {

GraphLinkLists\* temp = manualNodeHead;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = g;

}

}

void addNodeFromEquation(sf::Vector2f v,sf::Color color) {

GraphLinkLists\* g = new GraphLinkLists;

g->c.setPosition(v.x + graphNodeSize, v.y + graphNodeSize);

g->c.setRadius(graphNodeSize);

g->c.setFillColor(color);

g->next = NULL;

if (manualNodeHead == NULL) {

manualNodeHead = g;

}

else {

GraphLinkLists\* temp = manualNodeHead;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = g;

}

}

void displayList(sf::RenderWindow\* w, GraphLinkLists\* head, sf::Color color) {

if (head == NULL) {

return;

}

else {

GraphLinkLists\* temp = head;

while (temp->next != NULL) {

w->draw(temp->c);

sf::Vertex line[] =

{

sf::Vertex(sf::Vector2f(temp->c.getPosition().x + graphNodeSize,temp->c.getPosition().y + graphNodeSize)),

sf::Vertex(sf::Vector2f(temp->next->c.getPosition().x + graphNodeSize,temp->next->c.getPosition().y + graphNodeSize))

};

line->color = color;

w->draw(line, 2, sf::Lines);

temp = temp->next;

}

w->draw(temp->c);

}

}

void deleteList(GraphLinkLists\*\* head\_ref)

{

/\* deref head\_ref to get the real head \*/

GraphLinkLists\* temp = \*head\_ref;

GraphLinkLists\* next;

while (temp != NULL)

{

next = temp->next;

free(temp);

temp = next;

}

/\* deref head\_ref to affect the real head back

in the caller. \*/

\*head\_ref = NULL;

}

template <typename T> void function\_x(tgui::EditBox::Ptr editBox, sf::Color color)

{

typedef exprtk::symbol\_table<T> symbol\_table\_t;

typedef exprtk::expression<T> expression\_t;

typedef exprtk::parser<T> parser\_t;

std::string expression\_string = editBox->getText();

T x;

symbol\_table\_t symbol\_table;

symbol\_table.add\_variable("x", x);

symbol\_table.add\_constants();

expression\_t expression;

expression.register\_symbol\_table(symbol\_table);

parser\_t parser;

parser.compile(expression\_string, expression);

for (x = T(-5); x <= T(+5); x += T(0.1))

{

T y = expression.value();

addNodeFromEquation(sf::Vector2f(x \* x\_scale / zoom + x\_loc + gridsize / 2 \* x\_scale, - y \* y\_scale / zoom + y\_loc + gridsize / 2 \* y\_scale), color);

}

}

template <typename T> void function\_x\_y(tgui::EditBox::Ptr editBox, tgui::EditBox::Ptr editBox1,sf::Color color)

{

typedef exprtk::symbol\_table<T> symbol\_table\_t;

typedef exprtk::expression<T> expression\_t;

typedef exprtk::parser<T> parser\_t;

std::string expression\_string\_x = editBox->getText();

std::string expression\_string\_y = editBox1->getText();

T x;

symbol\_table\_t x\_func,y\_func;

x\_func.add\_variable("t", x);

x\_func.add\_constants();

y\_func.add\_variable("t", x);

y\_func.add\_constants();

expression\_t x\_expression, y\_expression;

x\_expression.register\_symbol\_table(x\_func);

y\_expression.register\_symbol\_table(y\_func);

parser\_t parser\_x, parser\_y;

parser\_x.compile(expression\_string\_x, x\_expression);

parser\_y.compile(expression\_string\_y, y\_expression);

for (x = T(-5); x <= T(+5); x += T(0.1))

{

T x1 = x\_expression.value();

T y1 = y\_expression.value();

addNodeFromEquation(sf::Vector2f(x1 \* x\_scale / zoom + x\_loc + gridsize / 2 \* x\_scale, -y1 \* y\_scale / zoom + y\_loc + gridsize / 2 \* y\_scale),color);

}

}

};

**TESTING**

Testing Methods[4.1]

We have tested the program with various different methods, by adding faulty data to trying to access or change equation form and have used extensive try catch methods to ensure that no data is vulnerable and the program runs efficiently and smoothly no matter what.

**SAMPLE SCREENSHOTS OF SYSTEM**





