Pulchowk Campus
Institute of Engineering
Tribhuvan University

C project report on

2D GRAPH PLOTTER

Submitted by
Krishant Timalsina
(PUL078BCT045)
Bishal Panta
(PUL078BCT036)
Prabin Adhikari
(PUL078BCT058)
Apil Chaudhary
(PUL078BCT017)

Submitted to
Department of Electronics
and Computer Engineering

ABSTRACT

In this mathematical-masterpiece world, patterns are everywhere. And patterns are written in the form of equations. But nowadays, scary equations are overshadowing the beauty of patterns (i.e. curves and lines) which are visually appealing to human eyes. So, visualization of equations is crucial to view the beauty of this world. To achieve this objective, a graph plotter software was determined to be programmed through discussion. And, a search for tools to build graphical grids and user-friendly Graphical User Interface (GUI) in C-programming language was initiated.

After some browsing through the web, 'Raylib' library was found to be the solution to our problem. The functions of this library were learnt through examples and projects of other programmers in GitHub and other platforms. Its auxiliary modules made for additional functionality were also thoroughly studied and practised, so that, an A-quality software could be built. Finally, a 2D graph plotter was developed in C with the ability to plot multiple graphs of different variety at the same time taking inputs from sliders with the help of mouse.

ACKNOWLEDGEMENT

We show profound gratitude to everyone who helped us complete our project. Words can't express our respect for them. Their suggestions and feedback have helped us make our goal come true. First of all, we are thankful to Mr. Ganesh Gautam, our computer programming guru. He whole-heartedly taught us each and every concept of C-programming in our syllabus with minute details. We are also grateful to him for providing us opportunity to boost our knowledge by assigning project to us.

We would also like to show deep appreciation to our second teachers, online communities and open-source contributors whose examples and projects inspired us to make one too. Above all, our parents and friends made feasible environment for us to complete our project. They hold special place in our hearts. Casual talks with friends provided us new insights and ideas to incorporate in our project. In addition, we owe a lot to the author of the book "Learning C by examples" for providing valuable knowledge about C; the practice problems in this book made us better problem solvers. Last but not the least, we are heartily thankful to all helping hands who are directly or indirectly connected to our project.

TABLE OF CONTENT PAGE NO.

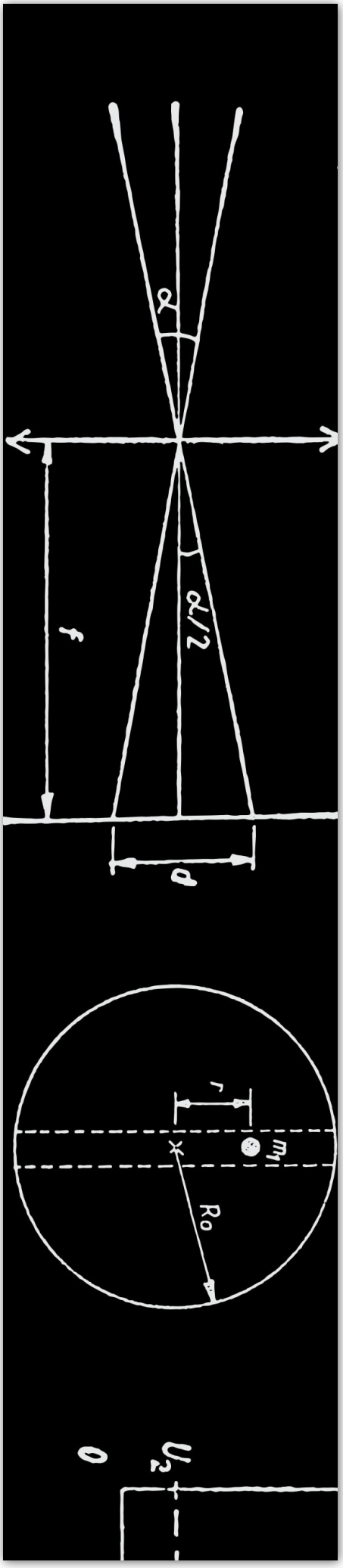
1. PROJECT INTRODUCTION

2D GRAPH PLOTTER

Our Project's name is **"2D-Graph Plotter"**. This project is prepared using **C-programming** language with the aid of an external library **"Raylib"** and its auxiliary module **"Raygui"** to apply a graphical user interface.

This program draws graphs for a wide variety of mathematical equations taking input from users in the form of slider values to assign them to variables.

In the starting stage, we discussed the approach for completing the project such as writing algorithm, source code... After the program was successfully completed, then only we wrote the documentation.



2.PROBLEM STATEMENT

2.1 Need

We, ourselves as well as our friends, faced problems, especially during math classes relating to equations in real-world scenarios. And the need for graphs was inevitable as it helps in visualizing mathematical language in a simple human-understandable form. And a light-weight easy-to-use graph plotter is an inseparable companion to each and every individual whether it's in classroom, or assigment sessions or just for self-study in engineering related topics.

2.2 Objective

- To demostrate our knowledge about C-programming.
- To learn about different library functions
- To learn about importance of group work
- To learn about about array, string, structure, pointer,..
- To use user defined functions and custom header files for better clarity and code management
- To create GUI to enhance user experience

3.PROBLEM ANALYSIS

3.1 Understanding the problem:

Here we try to understand the problem to be solved in totally. Before with the next stage or step, we should be absolutely sure about the objectives of the given problem. Problem analysis is the process of understanding real-world problems and userâ€™s needs and proposing solutions to meet those needs. The goal of problem analysis is to gain a better understanding of the problem being solved before developing a solution.

2.2 Input requirements:

When we say input, it means to feed some data into a program. An input can be given in the form of a file or from the command line. C programming provides a set of built-in functions to read the given input and feed it to the program as per requirement.

3.3 Output requirements:

When we say output, it means to display some data on screen, printer, or in any file. C programming provides a set of built-in functions to output the data on the computer screen as well as to save it in text or binary files.

3.4 Processing requirements:

All programming involves creating something that solves a problem. The first step is to examine the problem carefully to try to identify what qualifies as a solution. The second step is to then look at the list of requirements and to decide exactly what your solution should do to fulfill them.

3.5 Technical feasibility:

While solving the problem, a programmer must determine whether it is feasible or not. For example: if a programmer has to solve the problem of simulating the boundary of the universe, it might be infeasible. However, if he has to solve the problem of simulating end of solar system, it might be feasible. It is the responsibility of programmer to avoid the infeasible solutions to solve the problem at hand.

4. REVIEW OF RELATED LITERATURE

Introduction:

C is a very powerful high level programming language which can also be used for system programming. Famous operating System such as UNIX, LINUX, MINIX and many systems and application programs are written in C. C is a very robust machine independent, general purpose, concise programming language. C consists of very few numbers of keywords (re served words) in comparison to other programming languages. Although C is a high level language, several powerful operation such as bit manipulation, memory management etc. required for system programmers are available in C. The C combines the capabilities of an assembly language with the features of high level language and therefore it is suitable for writing both system software and application software. Programs written in C are efficient and run faster than any other languages. C is a general propose programming language which features economy of expressions, modern control flow and data structures, and a rich set of operators.C is a typical programming language for academic purposes. For this reasons, many universities have a course in C programming in their curriculum. C has a very typical syntax which can later be used to learn other languages. A programmer who has a strong command in C can learn other languages like C++, C#, JAVA, PERL, PHP, Java script etc very easily with less effort.

Structure of C program

S.N.	SECTION	REMARKS
1	Documentation	Optional
2	Link	Essential
3	Definition	Optional
4	Global Declaration Function Declaration	Optional
5	Main function	Essential
6	Sub program Function 1 Function 2 ..... Function n	Optional

CONTROL STATEMENTS

Control statements control the flow of an execution of instructions in a program. They make it possible to make decisions, to perform tasks repeatedly or to jump from one section of code to another. Following are the types of control statements:

1. Sequential: Sequential statements are a set of statements whose execution process happens in a sequence. The problem with sequential statements is that if the logic has broken in any one of the lines, then the complete source code execution will break.

2. Branching: In branching a program can take different courses of action depending upon different circumstances.

2.a if statements:

The if statement allows you to control if a program enters a section of code based on whether a given condition is true or false.

Syntax:

```
if (test_expression)
{
statement;
}
```

2.b if-else statement:

The if statement will execute a single statement or a compound statement, when the test expression is true (nonzero). It does nothing when the test expression is not true. C provides the if-else construct to perform some action even when the test expression is not true (zero). We can have situation where we need to execute one if the condition is true, and another when false. The format of the if-else construct is as follows:

Syntax:

```
If (test_condition)
{
true block of statement;
}
Else
{
false block of statement;
}
```

2.c nested if-else statement:

This statement is used to check series of conditions for writing multi way decision. It is the type of nesting in which there is an if-else statement in every else part except the last part.

Syntax:

```
if (expression 1)
statement 1;
else if (expression 2)
statement 2;
.....
.....
else
default statement
```

2.d The switch statement:

In computer programming language, a switch statement is a type of selection control mechanism used to allow the value of a variable or expression to change the control flow of program execution via search and map.

Syntax:

```
switch (expression)
case value_1:
statement_1;
break;
case value 2:
statement 2;
break;
.....
.....
case value_n:
```

```
 statement_n;
 break;
 default:
 default statement;
 break;
```

2.e Looping:

The process of repeating a block of statements until the some condition is satisfied is known as looping.

i. for loop:

The for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for (initialization; test expression; update_expression)
{
body of loop;
}
```

ii. while loop:

A loop is used for executing a block of statements repeatedly until a given condition returns false.

Syntax:

```
while (test_expression)
{
statement;
update expression;
}
```

iii. do-while loop:

The do-while loop is an exit control (post test) loop unlike for and while loop. The loop body is executed at least once even though the condition is false because the test expression is at the bottom of the loop.If the conditional expression in the while loop is false initially, then the body of the loop is not executed even once. Sometime, we need to test the condition at the bottom of the loop. For example: for the validation of input data. In this case we use another construct do-while loop.

Syntax:

```
do
{
statement;
}while(test_expression);
```

2.f Break and continue statement:

Break statement:

A break statement terminates the execution of the loop (while or do-while or for) and the control is transferred to the statement immediately following the loop. When break is encountered in a program, remaining part of the block is skipped and the loop is terminated passing control out of the loop. It is written as:

```
break;
```

Continue statement:

Unlike break, continue does not terminate a loop (while or do-while or for). It only interrupts a particular iteration. Whenever the continue statement is encountered, the remaining loop statement are skipped and the computation proceeds directly to the next pass of the loop. The condition which made to encounter with 'continue', that particular value is not executed, all other are executed. It is written as:

```
continue;
```

FUNCTIONS

A function is a self contained program segment or module which carries out a specified task. The function is one of the powerful features of C program. A function is a group of statement in a single logical unit to perform some specific task. A C program is usually

made up of many small functions, each performing a particular task, rather than one single large main() function. Once a function has been designed, its result can be used in computations without concern about its details of how the result was obtained. Functions can be developed, tested, and debugged independently. There are two types of function:

i. library function
ii. user defined function

Library function:
Library functions are built-in functions that are grouped together and placed in a common location called library. Each function here performs a specific operation. We can use this library functions to get the pre-defined output. Example of library function are printf(), scanf(), fopen(), fprintf(), etc.

User defined function:
A user-defined function (UDF) is a function provided by the user of a program or environment, in a context where the usual assumption is that functions are built into the program or environment. UDFs are usually written for the requirement of its creator.

Function prototype
A function prototype is simply the declaration of a function that specifies function's name, parameters and return type. It doesn't contain function body.

Function types
In C programming, function are of four types:
1. function with return type and passing arguments
2. function without return type and no arguments
3. function with return type and no arguments
4. function without return type and passing arguments

Pass by value and Pass by reference:
Pass by value: It is the method in which the value of the actual arguments is passed to the function from calling function.
Pass by reference: It is the method of function call in which the memory address of the actual argument is passed to the called function from calling function.

Recursive function
The function which calls itself is called recursive function and the process is called recursion. Here, any function that happens to call itself again and again (directly or indirectly), unless the program satisfies some specific condition/subtask is called a recursive function.

ARRAYS AND STRINGS
Arrays
Array is the collection of similar data items that can be represented by a single variable name.The individual data items in an array are called elements. A data item is stored in memory in one or more adjacent storage locations depending upon its type.
A pointer is a special type of variable which holds the address or location of another variable (or data item). Using pointer the data item is accessed indirectly through its address. In C, arrays and pointers are closely related. An array name is treated as a pointer constant, and pointers can be subscripted like arrays. In C, there is no formal data type for string; it is simply

one dimensional array of characters.

Declaration of Array:
The syntax for declaration of an one dimensional array is:
data_type array_name[size];

Initialization of Array:
Arrays may be of storage class automatic, external, or static, but not register. In traditional C, only external and static arrays can be initialized using an array initializer. In ANSI C, automatic arrays also can be initialized. An array can be initialized as:
data_type array_name[size]={element 1,element 2,.....};

Arrays as function arguments:
An array can be passed as argument to a function. Simply passing a value is different from passing an array as argument to called function. In a function definition, a formal parameter that is declared as an array is actually a pointer. When an array name is specified as argument to a function, the base address of the array is passed as "call-by-value" i.e. the address of the first element of the array is passed to the called function.

Strings
String are defined as an array of characters. The difference between a character array and a string is that the string is terminated by null character '\0'. The syntax for declaring syntax is:
char string_name[size];

String handling functions:
Although C does not have built-in string data type, it provides the programmer a rich set of library functions for handling strings. The function prototypes string-handling functions are given in the standard header file <string.h>. Some string handling functions in standard library are as follows:

strcpy(): It copies the content of one string into other.
syntax: strcpy(string 1, string 2);

strcmp(): It is used to compare two strings.
syntax: strcmp(string 1, string 2);
return 0 if both are same
return +1 if string 1 > string 2
return -1 if string 2 > string 1

strcat(): It is used to combined two strings into single.
syntax: strcat(string 1, string 2);

strlen(): It is used to calculate the length of a string.
syntax: strlen(string);

POINTER
Pointers are the special types of variables that can hold or store the memory location of other variable. Using the concept of pointer it is possible to write a very powerful and optimized code that can be run using very less memory space especially for a resource constraint environment.

Pointer declaration:
The general syntax for declaring a pointer variable is:
data_type *pointer_variable;

Pointer Arithmetic:
Suppose a pointer ptr is pointing an integer variable at memory location 2020. Suppose each memory location can store 8 bits (i.e. 1 byte) of data. Then it would take two consecutive memory

locations 2020 and 2021 to store the integer data assuming an integer data takes 2 bytes of memory. One can perform arithmetic operations on the ptr variable. The statement,

ptr = ptr +1;

increases the value of ptr to 2022 instead of 2021. This is because the pointer always points to starting address of a variable rather than intermediate one.

The statement,

ptr = ptr + 2;

increases the value of ptr to 2024.

Similarly, we can perform the decrement operations like (ptr-1), (ptr-2) etc on the pointer variables.

STRUCTURES

Structure is a collection of data items of same or different data types in contiguous memory location. Structure elements may contain simple variables, array, pointer or even other structure. It is a user defined data type. Use of structure in programs makes the programs more readable and efficient. A structure is defined or declared as:

```
struct structure_name
{
data_type 1 variable 1;
data_type 2 variable 2;
data_type 3 variable 3;
.....
.....
data_type n variable n;
};
```

Initializing a structure:

Suppose we a have structure named student already defined. The structure can be initialized like an array and string. The statement:

```
struct students = {"Ram", 12, 98,"Kathmandu"};
```

initializes s.name to "Ram", s.roll to 12, s. marks to 98 and s.address to"Kathmandu".

Array of Structures:

We can create the array of structure like we create the array of basic types. We have already known that an array is a group of identical data that are stored in consecutive memory locations in a common variable name. It is also possible to declare an array of structures. When declaring array of structures, the identical data type is the structure data type. But the structure of each array element may have identical data. For example, a programmer wants to store employee details of an office for processing different data of employees, array of structures are needed.

syntax:

```
struct structure_name
{
data_type 1 variable 1;
data_type 2 variable 2;
data_type 3 variable 3;
.....
.....
data_type n variable n;
}structure_variable[size];
```

Array within a structure:

The members of structure can be array. When an array is a member of a structure, it is called array within structure.

syntax:

```
struct structure_name
{
data_type 1 array_variable 1[size];
data_type 2 array_variable 2[size];
data_type 3 array_variable 3[size];
.....
.....
data_type n array_variable n[size];
```

};

Structures and Functions:
Structures can be passed to function similar to passing basics types to the function. It can be done in below 3 ways:
1. Passing structure to a function by value
2. Passing structure to a function by address(reference)
3. No need to pass a structure – Declare structure variable as global

Structure within structure(nested structure):
Sometime the member of the structure needs to have multiple values. For this case the member should be another variable of another structure type. C allows the member of structure to be thestructure. This mechanism is called the nesting of structure. When a structure has a member of another type of structure, it is called nested structure. Such nesting enables to build powerful data structures.

RAYLIB
Raylib is a programming library to enjoy videogames programming; no fancy interface, no visual helpers, no auto-debugging... just coding in pure spartan-programmers way. Raylib supports multiple target platforms, technically, any platform that **supports C language and OpenGL graphics** (or similar) can run raylib or it can be very easily ported to.
Raylib can be combined with several extra libraries like **raygui, raymath, raudio,** etc. for additional functionality, some of those libraries are already used internally while others are provided for user integration, most of these libraries are single-file header-only with no external dependencies.

// **Window-related functions**
void InitWindow(int width, int height, const char *title); // Initialize window and OpenGL context

int GetScreenWidth(void); // Get current screen width

int GetScreenHeight(void); // Get current screen height

//**Drawing-related functions**
void ClearBackground(Color color); // Set background color (framebuffer clear color)

void BeginDrawing(void); // Setup canvas (framebuffer) to start drawing

void EndDrawing(void); // End canvas drawing and swap buffers (double buffering

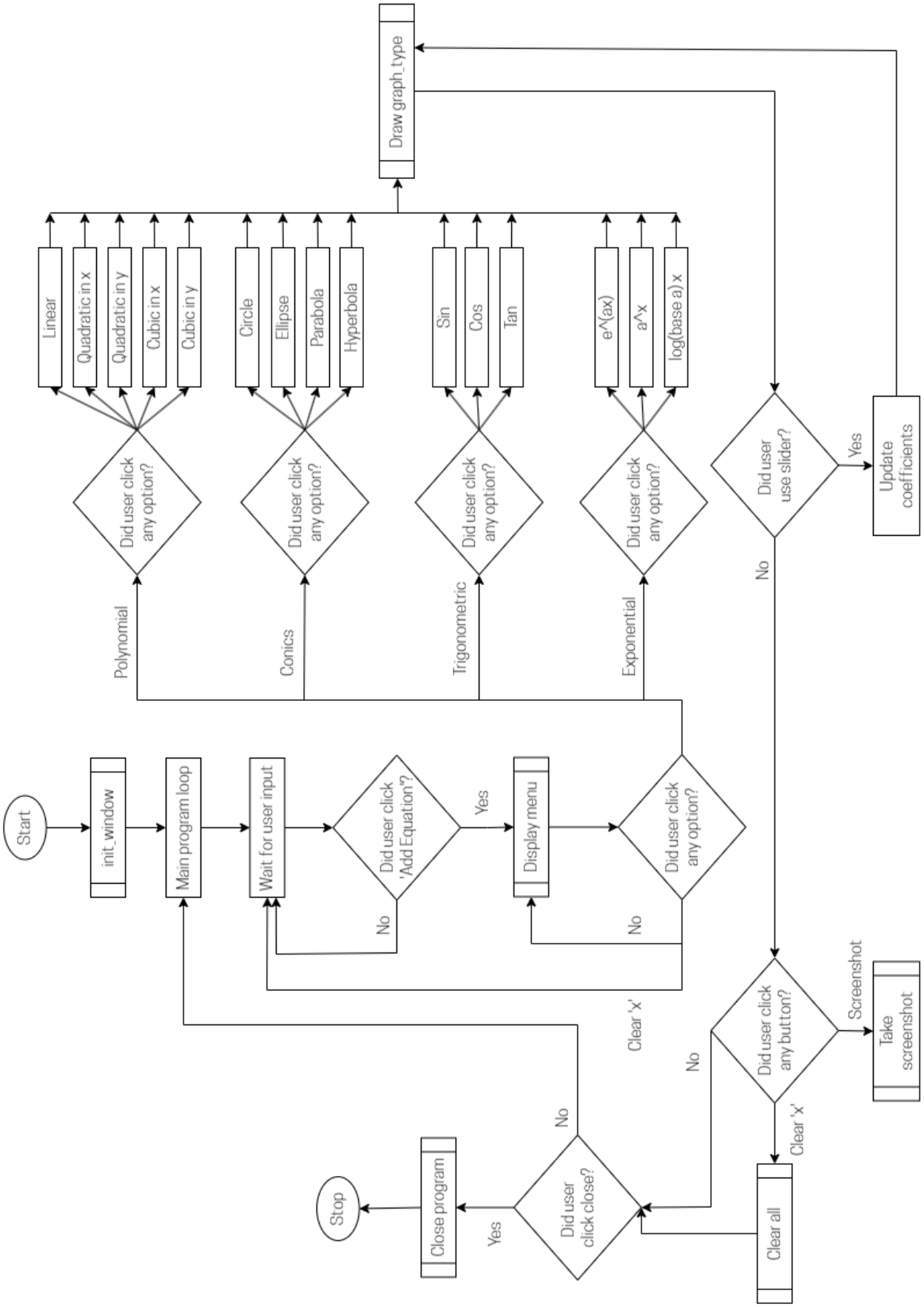
// **Basic shapes drawing functions**

```
void DrawPixel(int posX, int posY, Color color);                    // Draw a pixel
void DrawPixelV(Vector2 position, Color color);                     // Draw a pixel (Vector version)

void DrawLine(int startPosX, int startPosY, int endPosX, int endPosY, Color color); // Draw a line

void DrawLineV(Vector2 startPos, Vector2 endPos, Color color);      // Draw a line (Vector version)

void DrawLineEx(Vector2 startPos, Vector2 endPos, float thick, Color color); // Draw a line defining thickness

void DrawLineBezier(Vector2 startPos, Vector2 endPos, float thick, Color color); // Draw a line using cubic-bezier curves in-out

void DrawLineBezierQuad(Vector2 startPos, Vector2 endPos, Vector2 controlPos, float thick, Color color); // Draw line using quadratic bezier curves with a control point

void DrawLineBezierCubic(Vector2 startPos, Vector2 endPos, Vector2 startControlPos, Vector2 endControlPos, float thick, Color color); // Draw line using cubic bezier curves with 2 control points

void DrawLineStrip(Vector2 *points, int pointCount, Color color);   // Draw lines sequence

void DrawCircleLines(int centerX, int centerY, float radius, Color color); // Draw circle outline

void DrawEllipseLines(int centerX, int centerY, float radiusH, float radiusV, Color color);     // Draw ellipse outline
```

5.1 ALGORITHM

1. Start
2. Import libraries and header files
3. Initialise array of equations
4. Initialise variables for equations
5. Main program loop
6. Call init_window function to open program window
7. Is close button clicked or escape key pressed?
   Yes, Close window and exit
   No, next step
8. Call BeginDrawing function to make canvas for drawing
9. Call ClearBackground function
10. Call draw_axis function to draw grids in canvas
11. Call draw_graphs function : calls draw_graph for each equation
12. Call draw_sections function : draws equation section
13. Call draw_sliders - to input values through mouse for variables
14. Call draw_boxes - draws container for equation and its input
15. Call window_add - contains 'Add equation' button
16. Call draw_buttons - for clear screen and screenshot
17. Call EndDrawing
18. Goto step 7
19. Close window

5.2 FLOWCHART



6.RESULTS

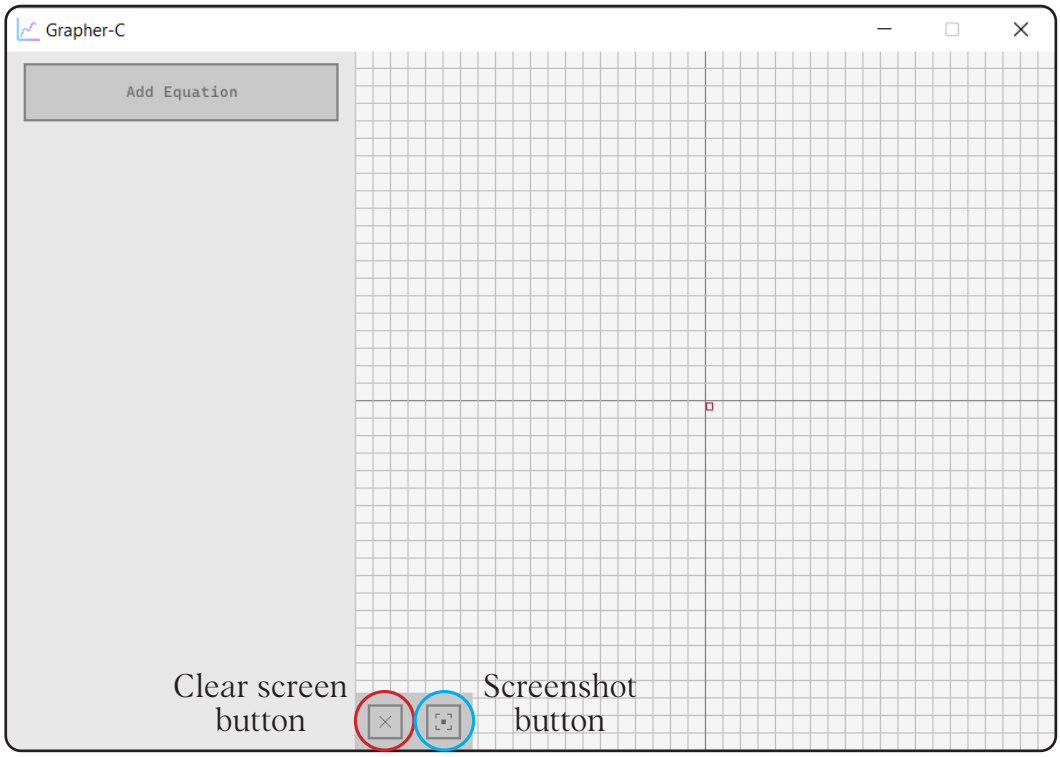


Fig 1. Starting Window

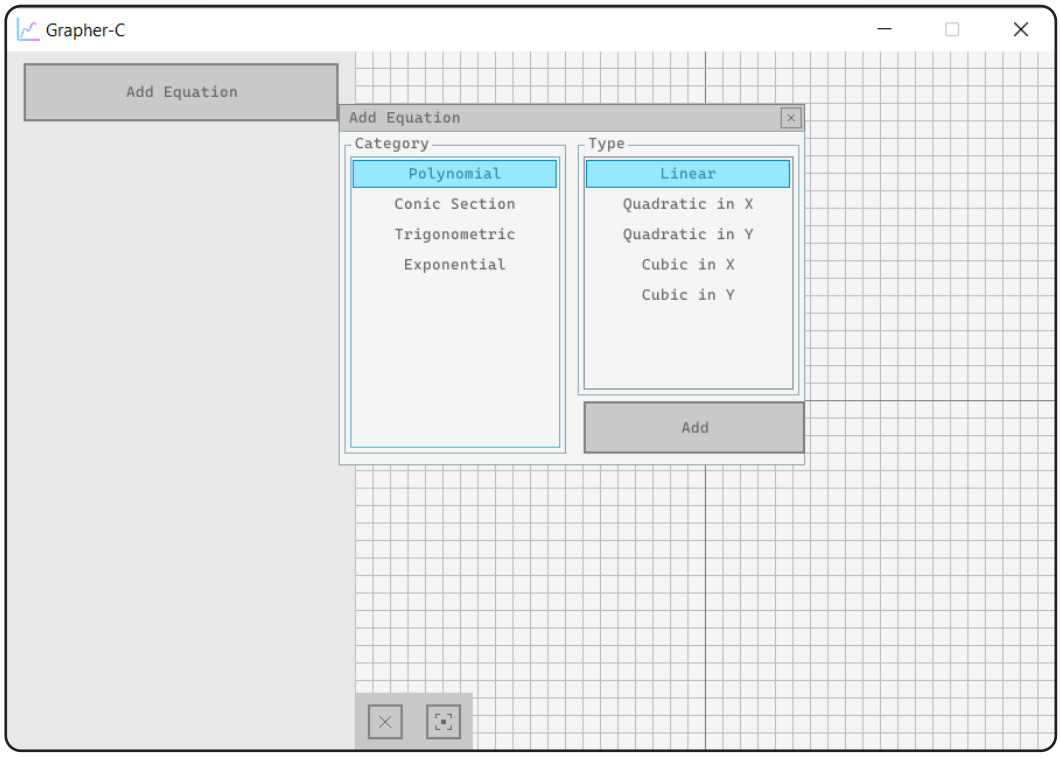


Fig 2. Pop-up window of equation types after clicking 'Add Equation' button

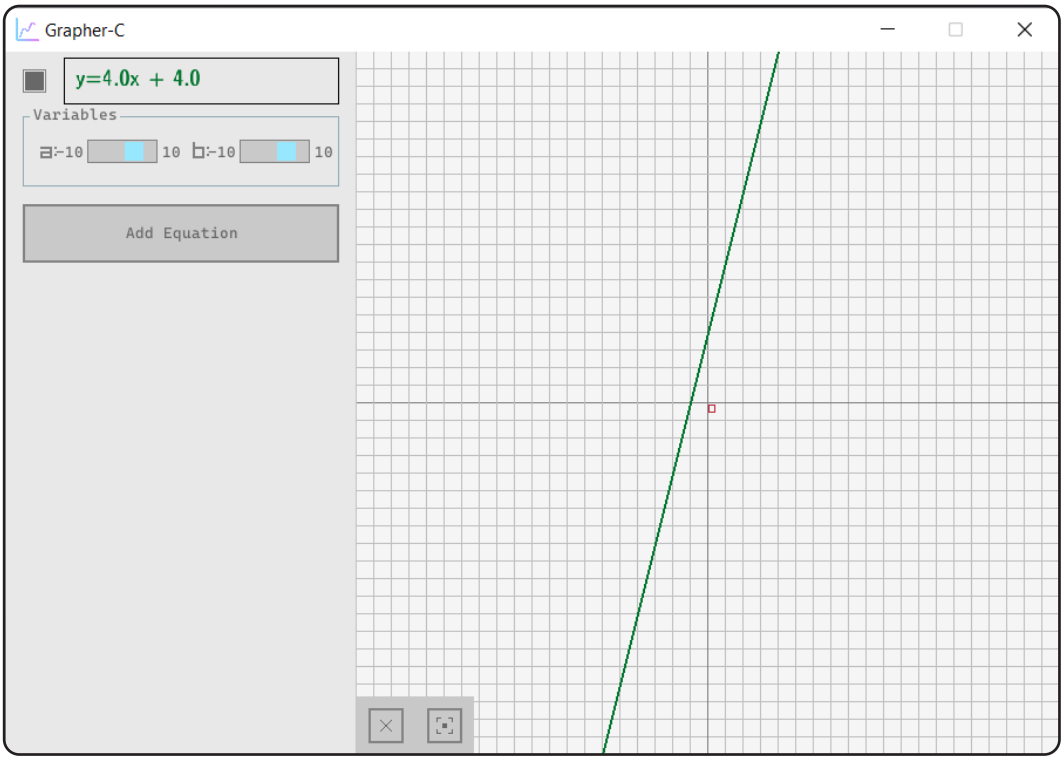


Fig 3. Graph of selected equation type

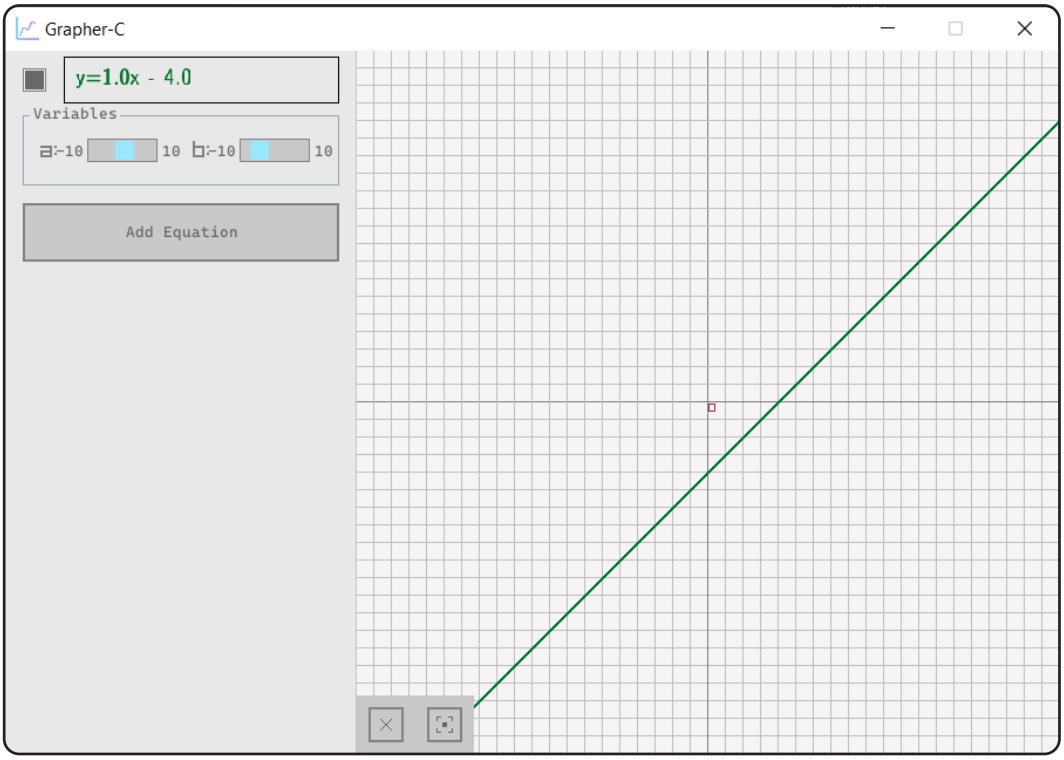


Fig 4. Modified graph after changing value of coefficients with sliders

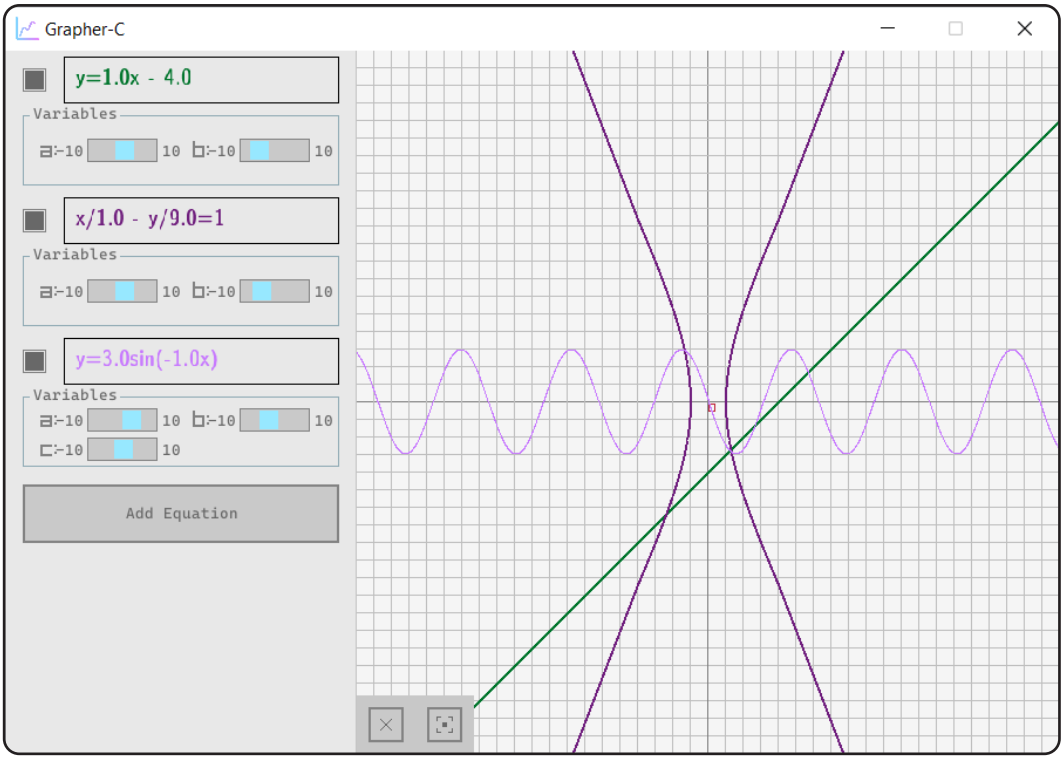


Fig 5. Multiple curves at the same time

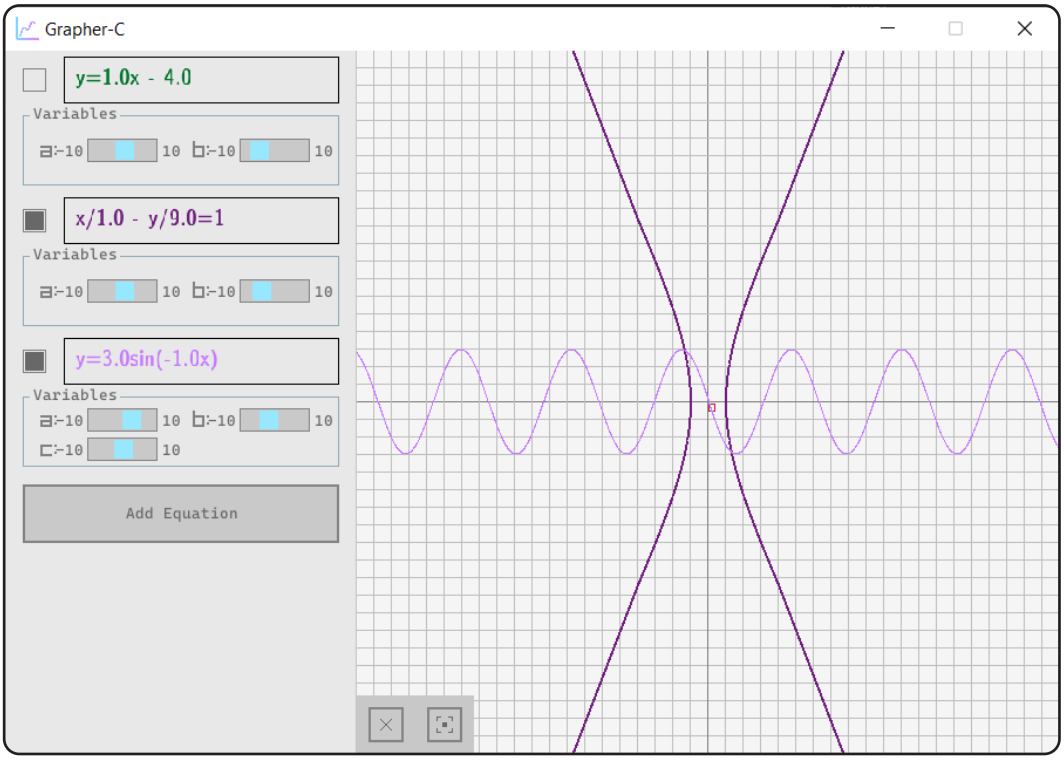


Fig 6. Toggle on/off curves through radio button on side of equation

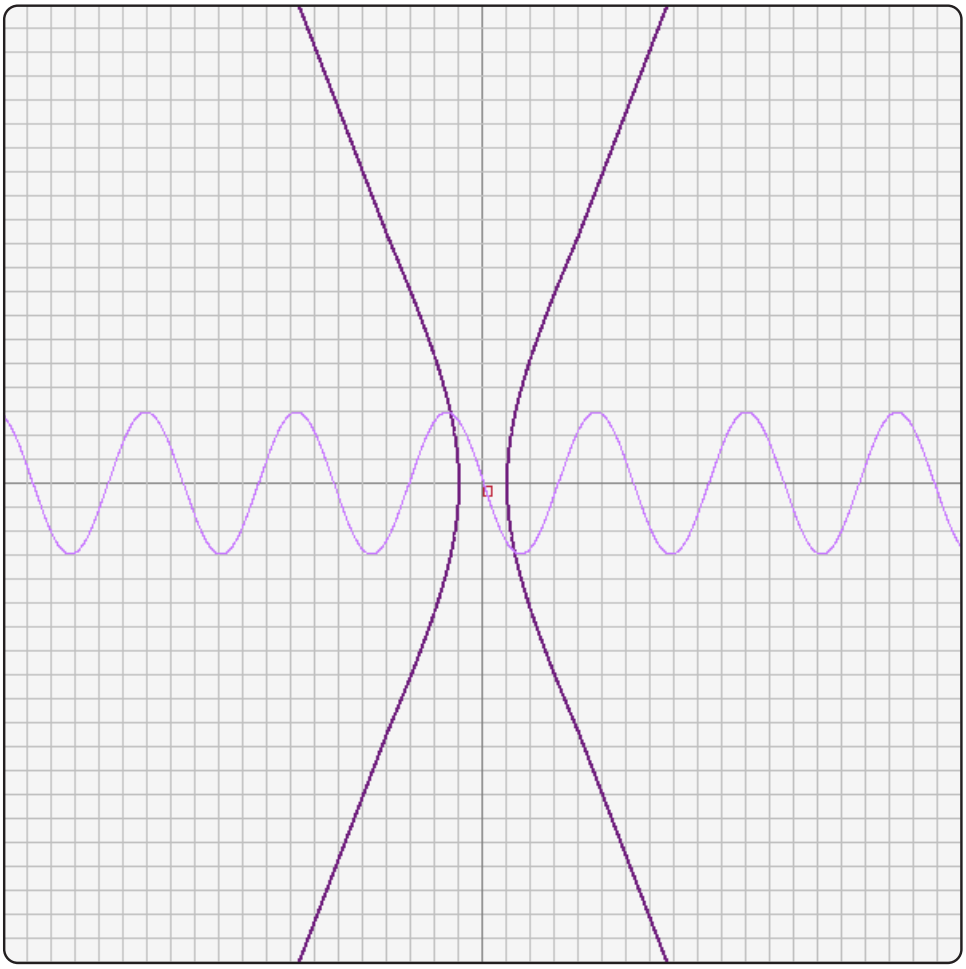


Fig 7. Screenshot captured and saved after clicking Screenshot button

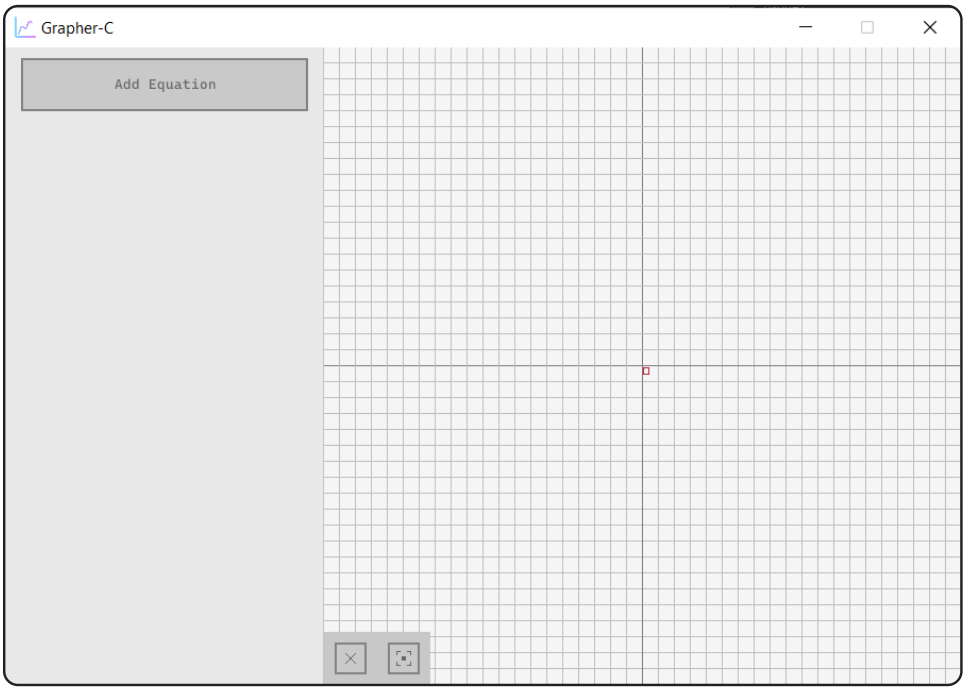


Fig 8. Clear canvas after clicking clear screen button

9.SOURCE CODE

main.c

```
#include <raylib.h>

#define RAYGUI_IMPLEMENTATION
#include "raygui.h" // Required for GUI controls

// custom header files
#include "structures.h"
#include "ui.h"
#include "graph.h"

// initialize array of euqations
Equation equation_arr[8];

char* vars[] = { "a:", "b:", "c:", "d:" };

 // Input Sliders
void draw_sliders();


int main(void)
{
	// init window
	init_window();

	// Main program loop
	while (!WindowShouldClose()) // Detect window close button or ESC key
	{
		BeginDrawing();
		ClearBackground(RAYWHITE);

		// draws grid
		draw_axis();
		// initialize euqations box num_eq times
		draw_graphs();
		// Draws sections of main window
		draw_sections();
		// Draws equations box
		draw_boxes();

		// Input sliders
		draw_sliders();

		//custom functions to draw window
		window_add();

		// screenshot and clear buttons
		draw_buttons();

		EndDrawing();
	}

	clean_up(); // Close window

	return 0;
}


void draw_sliders(){
	for (int i = 0; i < num_eq; i++) {
			GuiGroupBox((Rectangle) { 15, 55 + 120 * i, 270, 60 }, "Variables");
```

```
                                        for (int j = 0; j < equation_arr[i].type.var_num; j++) {
                                                float xcor = (j % 2 == 0) ? 30 : 160;
                                                float ycor = 120 * i + ((j / 2 != 0) ? 25 : 0);
                                                if (equation_arr[i].type.var_num <= 2)
                                                        ycor += 75;
                                                else
                                                        ycor += 65;
                                                DrawText(vars[j], xcor, ycor, 20, GRAY);
                                                equation_arr[i].type.value[j] = GuiSlider((Rectangle) { xcor + 40, ycor, 60, 20 },
“-10”, “10”, equation_arr[i].type.value[j], -10, 10);
                                        }
                                }
}
```

graph.h

```
#ifndef GRAPH_H
#define GRAPH_H
#include "raylib.h"

int draw_axis();
int show_eq(char *type, float a, float b, float c, float d, Color col);

#endif
```

ui.h

```
#ifndef UI_H
#define UI_H

int init_window();
int draw_sections();
int draw_graphs();
int draw_boxes();
int window_add();
int box_eq(int i);
int clean_up();

#endif
```

structures.h

```
#ifndef STRUCTURE_H
#define STRUCTURE_H
#include "raylib.h"


typedef struct Type {
   char* label;
   int var_num;
   float value[4];
}Type;


typedef struct Equation
{
   Type type;
   bool show;
   Color color;
} Equation;
```

```
extern Equation equation_arr[8];


typedef struct Window
{
   char *label;
   int typelen;
   Type types[5];
   Color col;
} Window;


extern Window windows[4];
extern int num_eq;
#endif
```

graph.c

```
#include "raylib.h"
#include <math.h>
#include <string.h>

#define PI_2 1.57079
#define EXP 2.71828

// polynomial
void linear(float a, float b, Color col);
void quadratic_y(float a, float b, float c, Color col);
void quadratic_x(float a, float b, float c, Color col);
void cubic_y(float a, float b, float c, float d, Color col);
void cubic_x(float a, float b, float c, float d, Color col);
// conic
void hyperbola(float a, float b, Color col);
void ellipse(float a, float b, Color col);
// trignometric
void draw_sine(float a, float b, float c, Color col);
void draw_tan(float a, float b, float c, Color col);
// exponential
void draw_log(float a, Color col);
void draw_exp(float a, Color col);

int draw_axis()
{
        int centerX = GetScreenWidth() - GetScreenHeight() / 2;
        int centerY = GetScreenHeight() / 2;

        // Line testing
        Vector3 start = { 0.0f, 0.0f, 0.0f };
        Vector3 end = { 10.0f, 10.0f, 10.0f };

        Camera3D camera = { 0 };
        camera.position = (Vector3){ 5.0f, 10.0f, 0.0f }; // Camera position
        camera.target = (Vector3){ 5.0f, 0.0f, 0.0f };    // Camera looking at point
        camera.up = (Vector3){ 0.0f, 0.0f, 10.0f };                 // Camera up vector (rotation towards target)
        camera.fovy = 90.0f;                                                                // Camera field-of-view Y
        camera.projection = CAMERA_PERSPECTIVE;
        // Making graph grid line
        BeginMode3D(camera);
        DrawGrid(40, 0.5f);
        EndMode3D();
```

```
        // Marking the origin
        DrawText("O", centerX, centerY, 2.0f, MAROON);
}

int show_eq(char* type, float a, float b, float c, float d, Color col)
{
        // type = tolower(type);
        floor(rand() * 10);
        //polynomial
        if (!strcmp(type, "linear"))
                linear(a, b, col);
        if (!strcmp(type, "quadratic_x"))
                quadratic_x(a, b, c, col);
        if (!strcmp(type, "quadratic_y"))
                quadratic_y(a, b, c, col);
        if (!strcmp(type, "cubic_x"))
                cubic_x(a, b, c, d, col);
        if (!strcmp(type, "cubic_y"))
                cubic_y(a, b, c, d, col);
        // Conic Section
        if (!strcmp(type, "hyperbola"))
                hyperbola(a, b, col);
        if (!strcmp(type, "ellipse"))
                ellipse(a, b, col);
        if (!strcmp(type, "circle"))
                ellipse(a, a, col);
        if (!strcmp(type, "parabola"))
                quadratic_y(1 / (float) (4 * a), 0, 0, col);
        // trignometric
        if (!strcmp(type, "sin"))
                draw_sine(a, b, c, col);
        if (!strcmp(type, "cos"))
                draw_sine(a, b, c + PI_2, col);
        if (!strcmp(type, "tan"))
                draw_tan(a, b, c, col);
        // exponential
        if (!strcmp(type, "log_a_x"))
                draw_log(a, col);
        if(!strcmp(type, "a_x"))
                draw_exp(a,col);
        if(!strcmp(type, "e_ax"))
                draw_exp(pow(EXP,a),col);
}

// conic
void ellipse(float a, float b, Color col)
{
        int centerX = GetScreenWidth() - GetScreenHeight() / 2;
        int centerY = GetScreenHeight() / 2;
        DrawEllipseLines(centerX, centerY, a * 15, b * 15, col);
}

void hyperbola(float a, float b, Color col)
{
        int centerX = GetScreenWidth() - GetScreenHeight() / 2;
        int centerY = GetScreenHeight() / 2;
        Vector2 P1 = { centerX + 30 * a * 2, centerY + 30 * b * sqrt(3) };
        Vector2 P2 = { centerX + 2 / 3 * 30 * a, centerY + 30 * b * (48 - 26 * sqrt(3)) / 18 };
        Vector2 P3 = { centerX + 2 / 3 * 30 * a, centerY - 30 * b * (48 - 26 * sqrt(3)) / 18 };
        Vector2 P4 = { centerX + 30 * 2 * a, centerY - 30 * b * sqrt(3) };
        Vector2 El = { centerX + 20 * 15, centerY + 20 * (b / a) * 13 };
        Vector2 El2 = { centerX + 20 * 15, centerY - 20 * (b / a) * 13 };

        Vector2 q1 = { centerX - 30 * a * 2, centerY + 30 * b * sqrt(3) };
```

```
		Vector2 q2 = { centerX - 2 / 3 * 30 * a, centerY + 30 * b * (48 - 26 * sqrt(3)) / 18 };
		Vector2 q3 = { centerX - 2 / 3 * 30 * a, centerY - 30 * b * (48 - 26 * sqrt(3)) / 18 };
		Vector2 q4 = { centerX - 30 * 2 * a, centerY - 30 * b * sqrt(3) };
		Vector2 Eql = { centerX - 20 * 15, centerY + 20 * (b / a) * 13 };
		Vector2 Eql2 = { centerX - 20 * 15, centerY - 20 * (b / a) * 13 };
		if (a <= 5 && a >= -5) {
			DrawLineBezierCubic(P1, P4, P2, P3, 2.0f, col);
			DrawLineBezierCubic(q1, q4, q2, q3, 2.0f, col);
			if (a >= 0) {
				DrawLineEx(P1, El, 2.0f, col);
				DrawLineEx(P4, El2, 2.0f, col);
				DrawLineEx(q1, Eql, 2.0f, col);
				DrawLineEx(q4, Eql2, 2.0f, col);
			}
			else {
				DrawLineEx(P1, Eql2, 2.0f, col);
				DrawLineEx(P4, Eql, 2.0f, col);
				DrawLineEx(q1, El2, 2.0f, col);
				DrawLineEx(q4, El, 2.0f, col);
			}
		}
		else {
			Vector2 q1 = { centerX - 30 * a * 2, centerY + 30 * b * sqrt(3) };
			Vector2 q4 = { centerX - 30 * 2 * a, centerY - 30 * b * sqrt(3) };
			DrawLineBezierCubic(P1, P4, P2, P3, 2.0f, col);
			DrawLineBezierCubic(q1, q4, q2, q3, 2.0f, col);
		}

}


// polynomial
void linear(float a, float b, Color col){
		int centerX = GetScreenWidth() - GetScreenHeight() / 2;
		int centerY = GetScreenHeight() / 2;


		int rightEdgeX = (centerX / 2) / 15;
		int rightEdgeY = a * rightEdgeX + b;
		int leftEdgeX = (-centerX / 2) / 15;
		int leftEdgeY = a * leftEdgeX + b;

		if (rightEdgeY <= 20 || leftEdgeY <= 20) {
			Vector2 start = { centerX + rightEdgeX * 15, centerY - rightEdgeY * 15 };
			Vector2 end = { centerX + leftEdgeX * 15, centerY - leftEdgeY * 15 };
			DrawLineEx(start, end, 2.0f, col);
		}

}
void quadratic_y(float a, float b, float c, Color col)
{
		int centerX = GetScreenWidth() - GetScreenHeight() / 2;
		int centerY = GetScreenHeight() / 2;

		/*for(float i=-10000;i<=10000;i=i+1)
					{
						int x=i/15+centerX-1;
						int y=((-a*i*i-b*i*229-c)/3500+centerY-1);
						if(-y+897+c*15>=300)
						DrawPixel(-y+897+c*15,x-300+122,col);
					}*/
		for (float i = -10000; i <= 10000; i = i + 1)
		{
			int x = i / 15 + centerX - 1;
```

```
                int y = ((-a * i * i + b * i * 229 - c) / 3500 + centerY);
                if (-y + 897 + c * 15 >= 300)
                        DrawPixel(-y + 898 + c * 15, x - 300 + 2, col);
        }
}
void quadratic_x(float a, float b, float c, Color col)
{
        int centerX = GetScreenWidth() - GetScreenHeight() / 2;
        int centerY = GetScreenHeight() / 2;
        for (float i = -10000; i < 10000; i = i + 1)
        {
                int x = i / 15 + centerX - 1;
                int y = ((-a * i * i - b * i * 230 - c) / 3500 + centerY - c * 15);
                DrawPixel(x, y, col);
        }
        for (float i = -10000; i <= 10000; i = i + 1)
        {
                int x = i / 15 + centerX - 1;
                int y = ((-a * i * i - b * i * 230 - c) / 3500 + centerY - 1 - c * 15);
                DrawPixel(x, y, col);
        }
}
void cubic_x(float a, float b, float c, float d, Color col)
{
        int centerX = GetScreenWidth() - GetScreenHeight() / 2;
        int centerY = GetScreenHeight() / 2;
        for (float i = -10000; i < 10000; i = i + 1)
        {
                int x = i / 15 + centerX - 1;
                int y = ((-a * i * i * i - b * i * i * 229 - c * i * 52500) / 800000 + centerY - d * 15);
                DrawPixel(x, y, col);
                DrawPixel(x, y - 1, col);
                DrawPixel(x, y + 1, col);
        }
}
void cubic_y(float a, float b, float c, float d, Color col)
{
        int centerX = GetScreenWidth() - GetScreenHeight() / 2;
        int centerY = GetScreenHeight() / 2;
        for (float i = -10000; i < 10000; i = i + 1)
        {
                int x = i / 15 + centerX - 1;
                int y = ((-a * i * i * i + b * i * i * 229 - c * i * 52500) / 800000 + centerY);
                if (y + 300 + 1 + d * 15 >= 300)
                {
                        DrawPixel(y + 300 + d * 15, x - 300, col);
                        DrawPixel(y + 300 + 1 + d * 15, x - 300, col);
                        if (b != 0)
                                DrawPixel(y + 300 - 1 + d * 15, x - 300, col);
                }
        }
}


// trignometric
void draw_sine(float a, float b, float c, Color col)
{
   float a1=a>=0 ? a: -a;
   float b1=b>=0 ? b:-b;
   int centerX = GetScreenWidth() - GetScreenHeight() / 2;
   int centerY = GetScreenHeight() / 2;
   for (float i = -10000; i < 100000; i = i + 1)
   {
      int x1, y1;
```

```
      int x = i / 5;
      int y = -a * 15 * sin((b * i) / 75);
      if (x + centerX - c * 15 >= 300)
      {
        if(a1+b1<=5)
        DrawPixel(x + centerX - c * 15, y + centerY - 1, col);
        else{
          DrawLine(x1 + centerX - c * 15, y1 + centerY - 1, x + centerX - c * 15, y + centerY - 1, col);
        }
      }
      x1 = x;
      y1 = y;
    }
}
void draw_tan(float a, float b, float c, Color col)
{
          int centerX = GetScreenWidth() - GetScreenHeight() / 2;
          int centerY = GetScreenHeight() / 2;
          for (float i = -10000; i < 10000; i = i + 1)
          {
                    int x1, y1;
                    int x = i / 5;
                    int y = -a * 15 * tan((b * i) / 75);
                    if (x + centerX - c * 15 >= 300)
                    {
                              DrawPixel(x + centerX - c * 15, y + centerY - 1, col);
                              // if(i>=-1000)
                              DrawLine(x1 + centerX - c * 15, y1 + centerY - 1, x + centerX - c * 15, y + centerY - 1, col);
                              // DrawLine(x2+centerX-c*15,y2+centerY-1,x+centerX-c*15,y+centerY-1,col);
                              // int x2=x1;
                              // int y2=y1;
                    }
                    x1 = x;
                    y1 = y;
          }
}


// exponential
void draw_log(float a, Color col) {
          int centerX = GetScreenWidth() - GetScreenHeight() / 2;
          int centerY = GetScreenHeight() / 2;

          if (a > 0 && a != 1) {
                    // from origin to 0.001
                    float x1, y1, x2, y2;
                    int j = 0;
                    x1 = 0.0001 * 15;
                    y1 = 600;
                    x2 = 0.001 * 15;
                    y2 = ((float)log(0.001) / (float)log(a)) * 15;


                    DrawLineEx((Vector2) { centerX + x1, y1 }, (Vector2) {
                              centerX + x2, centerY - y2
                    }, 2.0f, col);

                    // for range 0.001 to 1
                    for (float i = 0.001; i < 1; i += 0.001, j++) {

                              if (j % 2 == 0) {
                                        x1 = i;
                                        y1 = ((float)log(i) / (float)log(a));
                              }
```

```
                        else {
                                x2 = i;
                                y2 = ((float)log(i) / (float)log(a));
                        }
                        DrawLineEx((Vector2) { centerX + x1 * 15, centerY - y1 * 15 }, (Vector2) { centerX + x2 * 15,
centerY - y2 * 15 }, 2.0f, col);
                }

                // for range 1 to 20
                for (int i = 1; i <= 20; i += 1) {
                        if (i % 2 == 0) {
                                x1 = i;
                                y1 = ((float)log(i) / (float)log(a));
                        }
                        else {
                                x2 = i;
                                y2 = ((float)log(i) / (float)log(a));
                        }
                        DrawLineEx((Vector2) { centerX + x1 * 15, centerY - y1 * 15 }, (Vector2) { centerX + x2 * 15,
centerY - y2 * 15 }, 2.0f, col);
                }
        }

}
void draw_exp(float a,Color col)
{
   int centerX = GetScreenWidth()-GetScreenHeight()/2;
   int centerY = GetScreenHeight()/2;
   for(float i=-10000;i<10000;i=i+1)
        {
          int x1,y1;
          int x=i*16;
          int y=-pow(a,i)*15.999999;
         if(x+centerX>=300){
          //DrawPixel(x+centerX,y+centerY,col);
          DrawLine(x1+centerX,y1+centerY,x+centerX,y+centerY,col);
         }
          x1=x;
          y1=y;

}
}
```

ui.c

```
#include <raylib.h>
#include <string.h>
#include <time.h>

// TODO: SlideListView
// custom header
#include "ui.h"
#include "graph.h"
#include "structures.h"

Window windows[4] = {
        {"Polynomial", 5, {
                                {"linear", 2, {0}},
                                {"quadratic_x", 3, {0}},
                                { "quadratic_y",3, {0}},
                                {"cubic_x",4, {0}},
                                { "cubic_y",4, {0}}
                        }
                },
```

```
        {"Conic Section", 4, {
                                        {"ellipse",2, {0}},
                                        {"parabola",1, {0}},
                                        {"hyperbola",2, {0}},
                                        {"circle",1, {0}}
                                }
                },
        {"Trigonometric", 3, {
                                        {"sin",3, {0}},
                                        {"cos",3, {0}},
                                        {"tan",3, {0}}
                                }
                },
        {"Exponential", 3, {
                                        {"e_ax",1, {0}},
                                        {"a_x",1, {0}},
                                        {"log_a_x",1, {0}}
                                }
                }
};

//arrays for list view
char* window_category[4] = { "Polynomial", "Conic Section", "Trigonometric", "Exponential" };
char* window_type[4][5] = {
        {"Linear", "Quadratic in X", "Quadratic in Y", "Cubic in X", "Cubic in Y"},
        {"Ellipse", "Parabola", "Hyberbola", "Circle"},
        {"Sin", "Cos", "Tan"},
        {"e^(ax)", "a^x", "log(base a) x"}
};


// Function initialization
char *labelBuilder(char *label, float a, float b, float c, float d);


// no of equations
int num_eq = 0;
bool window_Active = false;
bool added = false;

//for list view
int current_category = 0;
int current_type = 0;

bool editMode = true;
Font font_label;
Font font_main;
Sound fx;

int init_window()
{
        const int screenWidth = 900;
        const int screenHeight = 600;
        InitWindow(screenWidth, screenHeight, "Grapher-C");
        InitAudioDevice();

        Image icon = LoadImage("res/icon.png");
        SetWindowIcon(icon);
        SetTargetFPS(GetMonitorRefreshRate(GetCurrentMonitor())); // Set our game to run at max refresh rate of
monitor
        font_main = LoadFontEx("res/code.otf", 15, 0, 250);
        font_label = LoadFontEx("res/cmunssdc.ttf", 30, 0, 250);
        GuiSetFont(font_main);
}
```

```
int draw_sections()
{
	DrawLine(GetScreenWidth() - GetScreenHeight(), 0, GetScreenWidth() - GetScreenHeight(), GetScreenHeight(),
Fade(LIGHTGRAY, 0.6f));
	DrawRectangle(0, 0, GetScreenWidth() - 600, GetScreenHeight(), Fade(LIGHTGRAY, 0.3f));
	DrawRectangle(300, 550, 100, 50 , LIGHTGRAY);

}

int draw_buttons() {
	if (GuiButton((Rectangle) { 310, 560, 30, 30 }, "#113#")) {
		for (int i = 0; i < 5; i++) {
			//equation_arr[i] = {{"\0", 0, {0}}, false, BLANK};
			num_eq = 0;
		}
	}
	if (GuiButton((Rectangle) { 360, 560, 30, 30 }, "#104#")) {
		time_t t;
		time(&t);
		int ti = t;
		system("mkdir screenshots");
		TakeScreenshot("screenshots/tmp.png");
		Image uncropped = LoadImage("screenshots/tmp.png");
		ImageCrop(&uncropped, (Rectangle) {300,0,600,600});
		ExportImage(uncropped, TextFormat("screenshots/Grapher_%d.png", ti));
		fx = LoadSound("res/screen.wav");
		system("cd screenshots && del tmp.png");
		PlaySound(fx);
	}
}
int draw_boxes() {
	for (int i = 0; i < num_eq; i++)
		box_eq(i);
}

int draw_graphs() {
	for (int i = 0; i < num_eq; i++)
		draw_graph(i);
}


int window_add()
{
	if (num_eq < 5 && GuiButton((Rectangle) { 15, num_eq * 120 + 10, 270, 50 }, "Add Equation"))
	{
		window_Active = true;
		//num_eq++;
	}

	if (window_Active)
	{
		//add data

		window_Active = !GuiWindowBox((Rectangle) { 285, 45, 400, 310 }, "Add Equation");
		GuiGroupBox((Rectangle) { 290, 80, 190, 265 }, "Category");
		current_category = GuiListViewEx((Rectangle) { 295, 90, 180, 250 }, window_category, 4, 0, NULL,
current_category);
		GuiGroupBox((Rectangle) { 490, 80, 190, 215 }, "Type");
		current_type = GuiListViewEx((Rectangle) { 495, 90, 180, 200 }, window_type[current_category],
windows[current_category].typelen, 0, NULL, current_type);
		added = GuiButton((Rectangle) { 495, 300, 190, 45 }, "Add");
		if (added) {
			window_Active = false;
```

```
                        Color colors[] = { RED, PURPLE, MAGENTA, DARKGREEN, DARKPURPLE ,BROWN, BEIGE
};
                        equation_arr[num_eq].color = colors[(GetRandomValue(0, 6))];
                        equation_arr[num_eq].type = windows[current_category].types[current_type];
                        equation_arr[num_eq].show = true;
                        //equation_arr[num_eq].type.value = {0,0,0,0};
                        for(int i=0; i< 4; i++)
                                equation_arr[num_eq].type.value[i] = GetRandomValue(-10, 10);
                        num_eq++;
                }
        }
}



int box_eq(int i)
{
        bool showEq = equation_arr[i].show;


        char label[80];
        strcpy(label,equation_arr[i].type.label);
        float a = equation_arr[i].type.value[0];
        float b = equation_arr[i].type.value[1];
        float c = equation_arr[i].type.value[2];
        float d = equation_arr[i].type.value[3];
        Color color = equation_arr[i].color;


        // Draw GUI controls
        //------------------------------------------------------------------------------
        equation_arr[i].show = GuiCheckBox((Rectangle) { 15, 15 + 120*i, 20, 20 }, "\0", equation_arr[i].show);
        DrawRectangleLines(50, 5 + 120 * i, 235, 40, BLACK);
        //DrawText(labelBuilder(label,a,b,c,d), 50 + 10,10 +120*i, 25, color);
        if(!strcmp(label, "cubic_x") || !strcmp(label, "cubic_y"))
                DrawTextEx(font_label, labelBuilder(label, a, b, c, d), (Vector2) { 45 + 10, 15 + 120 * i }, 19, 1, color);
        else
                DrawTextEx(font_label, labelBuilder(label, a, b, c, d), (Vector2) { 50 + 10, 12 + 120 * i }, 25, 1, color);
}

int draw_graph(int i)
{
        bool showEq = equation_arr[i].show;


        char label[80];
        strcpy(label, equation_arr[i].type.label);
        float a = equation_arr[i].type.value[0];
        float b = equation_arr[i].type.value[1];
        float c = equation_arr[i].type.value[2];
        float d = equation_arr[i].type.value[3];
        Color color = equation_arr[i].color;


        if (showEq)
        {
                show_eq(label, a, b, c, d,color);
                //DrawRectangle(0, 0, GetScreenWidth() - GetScreenHeight(), GetScreenHeight(), Fade(LIGHTGRAY,
0.6f));  //sliders(i);
        }
}

int clean_up() {
        UnloadFont(font_label);
```

```
        UnloadFont(font_main);
        UnloadSound(fx);
        CloseAudioDevice();
        CloseWindow();
};



char *labelBuilder(char *label, float a, float b, float c, float d){
        //polynomial
        if (!strcmp(label, "linear")) {
                if(!a&&!b) return (char*) TextFormat("y=0");
                if(!b) return (char*) TextFormat("y=%.1f",a);
                if(!a) return (char *)TextFormat("y=%.1f",b);
                if (b > 0) return (char *) TextFormat("y=%.1fx + %.1f", a, b);
                else return (char *)TextFormat("y=%.1fx - %.1f", a, -b);
        }


        if (!strcmp(label, "quadratic_x")) {
                if(!a&&!b&&!c) return (char *)TextFormat("y=0"); // a=b=c=0
                else if(!b&&!c) return (char *)TextFormat("y=%.1fx^2",a); //b=c=0
                else if(!a&&!c) return (char *)TextFormat("y=%.1fx",b); //a=c=0
                else if(!a&&!b) return (char *)TextFormat("y=%.1f",c); //a=b=0
                else if(!a) {
                        if(c>0) return (char *)TextFormat("y=%.1fx + %.1f",b,c); // a=0 c>0
                        else return (char *)TextFormat("y=%.1fx - %.1f",b,-1*c); //a=0 c<0
                }
                else if(!b) {
                        if(c>0) return (char *)TextFormat("y=%.1fx^2 + %.1f",a,c); // b=0 c>0
                        else return (char *)TextFormat("y=%.1fx^2 - %.1f",a,-1*c); // b=0 c<0
                }
                else if(!c) {
                        if(b>0) return (char *)TextFormat("y=%.1fx^2 + %.1fx",a,b); // c=0 b>0
                        else return (char *)TextFormat("y=%.1fx^2 - %.1fx",a,-1*b); // c=0 b<0
                }
                else{
                        if(b>0 && c>0 ) return (char *)TextFormat("y=%.1fx^2 + %.1fx + %.1f", a, b, c); // b and c both
positive
                        else if(c>0) return (char *)TextFormat("y=%.1fx^2 - %.1fx + %.1f", a, -1*b, c); // b negative
                        else if(b>0) return (char *)TextFormat("y=%.1fx^2 + %.1fx - %.1f", a, b, -1*c); // c negative
                        else return (char *)TextFormat("y=%.1fx^2 - %.1fx - %.1f", a, -1*b, -1*c); // b and c both negative
                        }
        }


        if (!strcmp(label, "quadratic_y")) {
                if(!a&&!b&&!c) return (char *)TextFormat("x=0"); // a=b=c=0
                else if(!b&&!c) return (char *)TextFormat("x=%.1fy^2",a); //b=c=0
                else if(!a&&!c) return (char *)TextFormat("x=%.1fy",b); //a=c=0
                else if(!a&&!b) return (char *)TextFormat("x=%.1f",c); //a=b=0
                else if(!a) {
                        if(c>0) return (char *)TextFormat("x=%.1fy + %.1f",b,c); // a=0 c>0
                        else return (char *)TextFormat("x=%.1fy - %.1f",b,-1*c); //a=0 c<0
                }
                else if(!b) {
                        if(c>0) return (char *)TextFormat("x=%.1fy^2 + %.1f",a,c); // b=0 c>0
                        else return (char *)TextFormat("x=%.1fy^2 - %.1f",a,-1*c); // b=0 c<0
                }
                else if(!c) {
                        if(b>0) return (char *)TextFormat("x=%.1fy^2 + %.1fy",a,b); // c=0 b>0
                        else return (char *)TextFormat("x=%.1fy^2 - %.1fy",a,-1*b); // c=0 b<0
                }
                else{
```

```
                        if(b>0 && c>0 ) return (char *)TextFormat("x=%.1fy^2 + %.1fy + %.1f", a, b, c); // b and c both
positive
                        else if(c>0) return (char *)TextFormat("x=%.1fy^2 - %.1fy + %.1f", a, -1*b, c); // b negative
                        else if(b>0) return (char *)TextFormat("x=%.1fy^2 + %.1fy - %.1f", a, b, -1*c); // c negative
                        else return (char *)TextFormat("x=%.1fy^2 - %.1fy - %.1f", a, -1*b, -1*c); // b and c both negative
                        }
        }

        if (!strcmp(label, "cubic_x")) {
                if(!a&&!b&&!c&&!d) return (char *)TextFormat("y=0");
                if(!b&&!c&&!d) return (char *)TextFormat("y=%.1fx^3", a);
                if(!a&&!c&&!d) return (char *)TextFormat("y=%.1fx^2", b);
                if(!a&&!b&&!d) return (char *)TextFormat("y=%.1fx", c);
                if(!a&&!b&&!c) return (char *)TextFormat("y=%.1f", d);
                if(!a&&!b) {
                        if(d>0) return (char *)TextFormat("y=%.1fx + %.1f", c, d);
                        else return (char *)TextFormat("y=%.1fx + %.1f", c, d);
                }
                if(!a&&!c) {
                        if(d>0) return (char *)TextFormat("y=%.1fx^2 + %.1f",  b,  d);
                        else return (char *)TextFormat("y=%.1fx^2 - %.1f", b, -1*d);
                }
                if(!a&&!d) {
                        if(c>0) return (char *)TextFormat("y=%.1fx^2 + %.1fx",  b,  c);
                        else return (char *)TextFormat("y=%.1fx^2 - %.1fx", b, -1*c);
                }
                if(!b&&!d) {
                        if(c>0) return (char *)TextFormat("y=%.1fx^3 + %.1fx",  a,  c);
                        else return (char *)TextFormat("y=%.1fx^3 - %.1fx", a, -1*c);
                }
                if(!b&&!c) {
                        if(d>0) return (char *)TextFormat("y=%.1fx^3 + %.1f",  a,  d);
                        else return (char *)TextFormat("y=%.1fx^3 - %.1f", a, -1*d);
                }
                if(!c&&!d) {
                        if(b>0) return (char *)TextFormat("y=%.1fx^3 + %.1fx^2",  a,  b);
                        else return (char *)TextFormat("y=%.1fx^3 - %.1fX^2", a, -1*b);
                }
                if(!a) {
                        if(c<0&&d<0) return (char *)TextFormat("y=%.1fx^2 - %.1fx - %.1f", b, -1*c, -1*d);
                        if(c<0) return (char *)TextFormat("y=%.1fx^2 - %.1fx + %.1f", b, -1*c, d);
                        if(d<0) return (char *)TextFormat("y=%.1fx^2 + %.1fx - %.1f", b, c, -1*d);
                        else return (char *)TextFormat("y=%.1fx^2 + %.1fx + %.1f", b, c, d);
                };
                if(!b) {
                        if(c<0&&d<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx - %.1f", a, -1*c, -1*d);
                        if(c<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx + %.1f", a, -1*c, d);
                        if(d<0) return (char *)TextFormat("y=%.1fx^3 + %.1fx - %.1f", a, c, -1*d);
                        else return (char *)TextFormat("y=%.1fx^3 + %.1fx + %.1f", a, c, d);
                }
                if(!c){
                        if(b<0&&d<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 - %.1f", a, -1*b, -1*d);
                        if(b<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 + %.1f", a, -1*b, d);
                        if(d<0) return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 - %.1f", a, b, -1*d);
                        else return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 + %.1f", a, b, d);
                }
                if(!d) {
                        if(b<0&&c<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 - %.1fx", a, -1*b, -1*c);
                        if(b<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 + %.1fx", a, -1*b, c);
                        if(c<0) return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 - %.1fx", a, b, -1*c);
                        else return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 + %.1fx", a, b, c);
                };

                        if(b<0&&c<0&&d<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 - %.1fx - %.1f", a, -1*b,
```

```
-1*c,-1*d);
                        if(b<0&&c<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 - %.1fx + %.1f", a, -1*b, -1*c,d);
                        if(b<0&&d<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 + %.1fx - %.1f", a, -1*b, c,-1*d);
                        if(c<0&&d<0) return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 - %.1fx - %.1f", a, b, -1*c,-1*d);
                        if(b<0) return (char *)TextFormat("y=%.1fx^3 - %.1fx^2 + %.1fx + %.1f", a, -1*b, c, d);
                        if(c<0) return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 - %.1fx + %.1f", a, b, -1*c,d);
                        if(d<0) return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 + %.1fx - %.1f", a, b, c,-1*d);
                        else return (char *)TextFormat("y=%.1fx^3 + %.1fx^2 + %.1fx + %.1f", a, b, c,d);

        }

        if (!strcmp(label, "cubic_y")) {
                if(!a&&!b&&!c&&!d) return (char *)TextFormat("x=0");
                if(!b&&!c&&!d) return (char *)TextFormat("x=%.1fy^3", a);
                if(!a&&!c&&!d) return (char *)TextFormat("x=%.1fy^2", b);
                if(!a&&!b&&!d) return (char *)TextFormat("x=%.1fy", c);
                if(!a&&!b&&!c) return (char *)TextFormat("x=%.1f", d);
                if(!a&&!b) {
                        if(d>0) return (char *)TextFormat("x=%.1fy + %.1f", c, d);
                        else return (char *)TextFormat("x=%.1fy + %.1f", c, d);
                }
                if(!a&&!c) {
                        if(d>0) return (char *)TextFormat("x=%.1fy^2 + %.1f",  b,  d);
                        else return (char *)TextFormat("x=%.1fy^2 - %.1f", b, -1*d);
                }
                if(!a&&!d) {
                        if(c>0) return (char *)TextFormat("x=%.1fy^2 + %.1fy",  b,  c);
                        else return (char *)TextFormat("x=%.1fy^2 - %.1fy", b, -1*c);
                }
                if(!b&&!d) {
                        if(c>0) return (char *)TextFormat("x=%.1fy^3 + %.1fy",  a,  c);
                        else return (char *)TextFormat("x=%.1fy^3 - %.1fy", a, -1*c);
                }
                if(!b&&!c) {
                        if(d>0) return (char *)TextFormat("x=%.1fy^3 + %.1f",  a,  d);
                        else return (char *)TextFormat("x=%.1fy^3 - %.1f", a, -1*d);
                }
                if(!c&&!d) {
                        if(b>0) return (char *)TextFormat("x=%.1fy^3 + %.1fy^2",  a,  b);
                        else return (char *)TextFormat("x=%.1fy^3 - %.1fy^2", a, -1*b);
                }
                if(!a) {
                        if(c<0&&d<0) return (char *)TextFormat("x=%.1fy^2 - %.1fy - %.1f", b, -1*c, -1*d);
                        if(c<0) return (char *)TextFormat("x=%.1fy^2 - %.1fy + %.1f", b, -1*c, d);
                        if(d<0) return (char *)TextFormat("x=%.1fy^2 + %.1fy - %.1f", b, c, -1*d);
                        else return (char *)TextFormat("x=%.1fy^2 + %.1fy + %.1f", b, c, d);
                };
                if(!b) {
                        if(c<0&&d<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy - %.1f", a, -1*c, -1*d);
                        if(c<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy + %.1f", a, -1*c, d);
                        if(d<0) return (char *)TextFormat("x=%.1fy^3 + %.1fy - %.1f", a, c, -1*d);
                        else return (char *)TextFormat("x=%.1fy^3 + %.1fy + %.1f", a, c, d);
                }
                if(!c){
                        if(b<0&&d<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 - %.1f", a, -1*b, -1*d);
                        if(b<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 + %.1f", a, -1*b, d);
                        if(d<0) return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 - %.1f", a, b, -1*d);
                        else return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 + %.1f", a, b, d);
                }
                if(!d) {
                        if(b<0&&c<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 - %.1fy", a, -1*b, -1*c);
                        if(b<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 + %.1fy", a, -1*b, c);
                        if(c<0) return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 - %.1fy", a, b, -1*c);
                        else return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 + %.1fy", a, b, c);
```

```
                };

                        if(b<0&&c<0&&d<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 - %.1fy - %.1f", a, -1*b,
-1*c,-1*d);
                        if(b<0&&c<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 - %.1fy + %.1f", a, -1*b, -1*c,d);
                        if(b<0&&d<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 + %.1fy - %.1f", a, -1*b, c,-1*d);
                        if(c<0&&d<0) return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 - %.1fy - %.1f", a, b, -1*c,-1*d);
                        if(b<0) return (char *)TextFormat("x=%.1fy^3 - %.1fy^2 + %.1fy + %.1f", a, -1*b, c, d);
                        if(c<0) return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 - %.1fy + %.1f", a, b, -1*c,d);
                        if(d<0) return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 + %.1fy - %.1f", a, b, c,-1*d);
                        else return (char *)TextFormat("x=%.1fy^3 + %.1fy^2 + %.1fy + %.1f", a, b, c,d);
        }

        // // Conic Section
        if (!strcmp(label, "hyperbola")){
                return (char *) TextFormat("x/%.1f - y/%.1f=1",a*a,b*b);
        }

        if (!strcmp(label, "ellipse")){
                return (char *) TextFormat("x/%.1f + y/%.1f=1",a*a,b*b);
        }

        if (!strcmp(label, "circle")){
                return (char *) TextFormat("x^2 + y^2=(%.1f)^2",a,b);
        }

        if(!strcmp(label, "parabola")){
                return (char *) TextFormat("y^2=%.1fx", 4*a);
        }

        // trignometric
        if (!strcmp(label, "sin")){
                if(!a) return (char *)TextFormat("y=0");
                else if(!b) return (char *)TextFormat("y=%.1fsin(%.1f)",a,c);
                else if(!c) return (char *)TextFormat("y=%.1fsin(%.1fx)",a,b);
                else if(c<0) return (char *)TextFormat("y=%.1fsin(%.1fx - %.1f)",a,b,-1*c);
                else return (char *)TextFormat("y=%.1fsin(%.1fx + %.1f)",a,b,c);
        }

        if (!strcmp(label, "cos")){
                if(!a) return (char *)TextFormat("y=0");
                else if(!b) return (char *)TextFormat("y=%.1fcos(%.1f)",a,c);
                else if(!c) return (char *)TextFormat("y=%.1fcos(%.1fx)",a,b);
                else if(c<0) return (char *)TextFormat("y=%.1fcos(%.1fx - %.1f)",a,b,-1*c);
                else return (char *)TextFormat("y=%.1fcos(%.1fx + %.1f)",a,b,c);
        }

        if (!strcmp(label, "tan")){
                if(!a) return (char *)TextFormat("y=0");
                else if(!b) return (char *)TextFormat("y=%.1ftan(%.1f)",a,c);
                else if(!c) return (char *)TextFormat("y=%.1ftan(%.1fx)",a,b);
                else if(c<0) return (char *)TextFormat("y=%.1ftan(%.1fx - %.1f)",a,b,-1*c);
                else return (char *)TextFormat("y=%.1ftan(%.1fx + %.1f)",a,b,c);
        }

        // // exponential
        if (!strcmp(label, "log_a_x")){
                if(a<=0|| a ==1) return (char *)TextFormat("doesn't exist!");
                else return (char *)TextFormat("y=log%.1f(x)",a);
        }

        if (!strcmp(label, "a_x")){
                return (char *)TextFormat("y=(%.1f)^x",a);
        }
```

```
        if (!strcmp(label, "e_ax")){
                return (char *)TextFormat("y=(e)^%.1fx",a);
        }

        return label;
}
```

7.CONCLUSION

Experience

When we started working on this project, we were sure it would be an interesting experience, but what we did not anticipate was the sheer number of challenges that would comealong. Challenges that proved the knowledge gained from our course to be insufficient fora project of this level. Algorithms that were unfamiliar to us had to be used for various parts of the program. But, along the way, we learned that these challenges were part of the learning experience. All in all, this was a very helpful project that introduced us to different paradigms of programming. As futile as it may be, we've tried to make a list of what we learned by doing this project:

1. Organizing the project and making the source code readable by using multiple files
and using a consistent naming convention.
2. Collaborative work with VCS like Git and remote hosting services like GitHub.
3. Using the official documentation of Libraries and the C language itself.
4. Unique algorithms for things like drawing curves, handling label, etc.
5. Avoiding memory leaks and other vulnerabilities that are exposed in low level language such as C.
6. Making sure the codes were methodical, operational and compatible.

Overview of the project

A one sentence definition of this project could be : "An optimal use of C programming language, learnt throughout the semester, to create a Graph Plotter which will get the basic tasks done." We can summarise the overall features of the project in these points:

1. Plots graphs for variety of equations
2. User friendly interactive approach to ask for input
3. Allows the modification of coefficients of variables through sliders to modify curve
4. Enables users to take screenshots too.
5. Allows users to compare two or more curves at the same time

8.REFERENCES

Book

Learning C by examples .... Krishna Kadel

Websites

www.stackoverflow.com
https://github.com/raysan5/raylib
https://www.raylib.com/cheatsheet/cheatsheet.html
https://github.com/raysan5/raygui/blob/master/src/raygui.h
https://www.raylib.com/examples.html