TRIBHUWAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A COMPLETE DOCUMENTATION ON "GRAPHULATOR"

Submitted to:

Dept. of Comp. Engineering

Submitted By:

Anusandhan Pokhrel (073BCT507)

Ashish Agarwal (073BCT509)

Pujan Karki (073BCT549)

GRATITUDE

Firstly, we would like to express our gratitude and great appreciation to Asst. Prof. Daya Sagar Baral for his continuous support and guidelines in completion of our project. The hard dedication, team work and efforts have finally collaborated to finalize the project. We are highly indebted to those who helped directly and indirectly to complete this program.

Also, we would like to thank each class mates for their valuable suggestions which contributed a lot in completion of project.

Last but not least, constructive criticisms and appraisal would be grateful.

"THE GRAPHULATOR"

Anusandhan Pokhrel (073BCT507)

Ashish Agarwal (073BCT509)

Pujan Karki (073BCT549)

CONTENTS

ABSTRACT

Graphulator – Graphing calculator is a numerical graphing calculator for 3D graphing generally. It is the visualization of the graphs of various equations which takes more time to even solve. The program uses object-oriented programming using C++. The extension library OpenGL was used in which Glut was included.

OBJECTIVES

This project is believed to be achieving listed objectives:

- ✓ Implementation of knowledge of object-oriented programming using C++.
- ✓ Being familiar with the graphics related programming knowledge using extensive library.
- ✓ Using various features of object-oriented programming in the project.
- ✓ Widening the concept of resource reusability by defining header files as per need.
- ✓ To develop the team-involvement, leadership qualities among the team members.

INTRODUCTION

Graphulator is the graphing calculator which visualizes the graphs of mathematical equations. The 3D graphs of various algebraic, trigonometrical, exponential expressions can be drawn using this calculator.

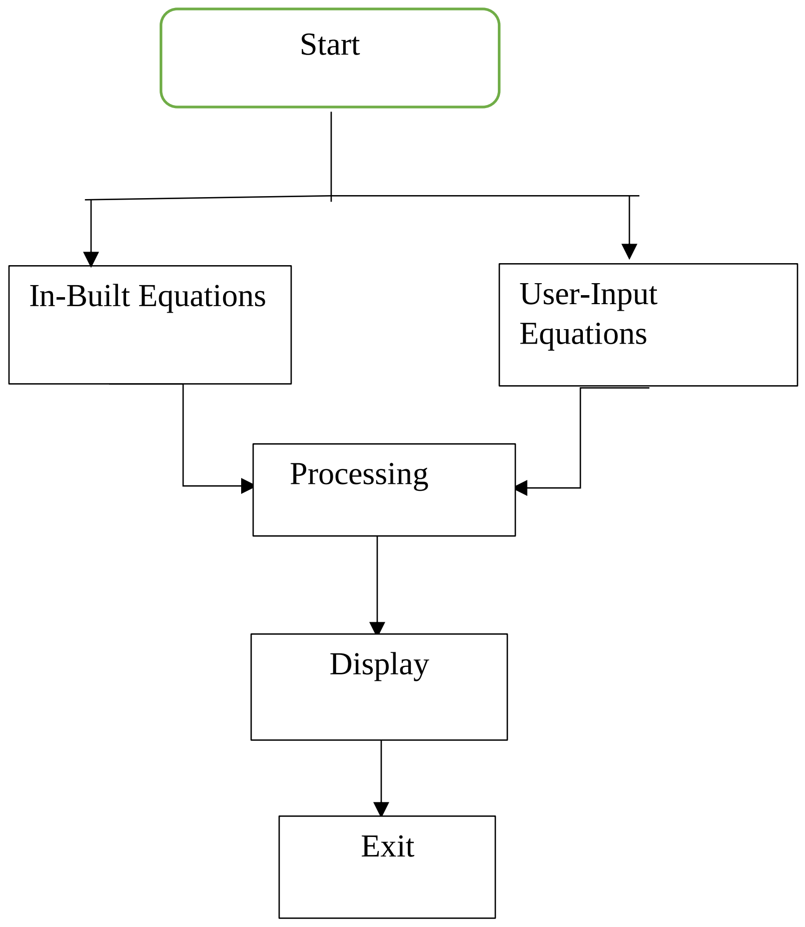
The user can observe the various in-built graph from the program. Also, user can input the desired graph using input window. The graph can be viewed in 3D co-ordinate system. Various key-board inputs can be given like to view next graphs, previous graphs and so on. The project members have tried best to provide user-friendly observation.

The amazing features of graphic library and C++ has been used in order to make the project more efficient.

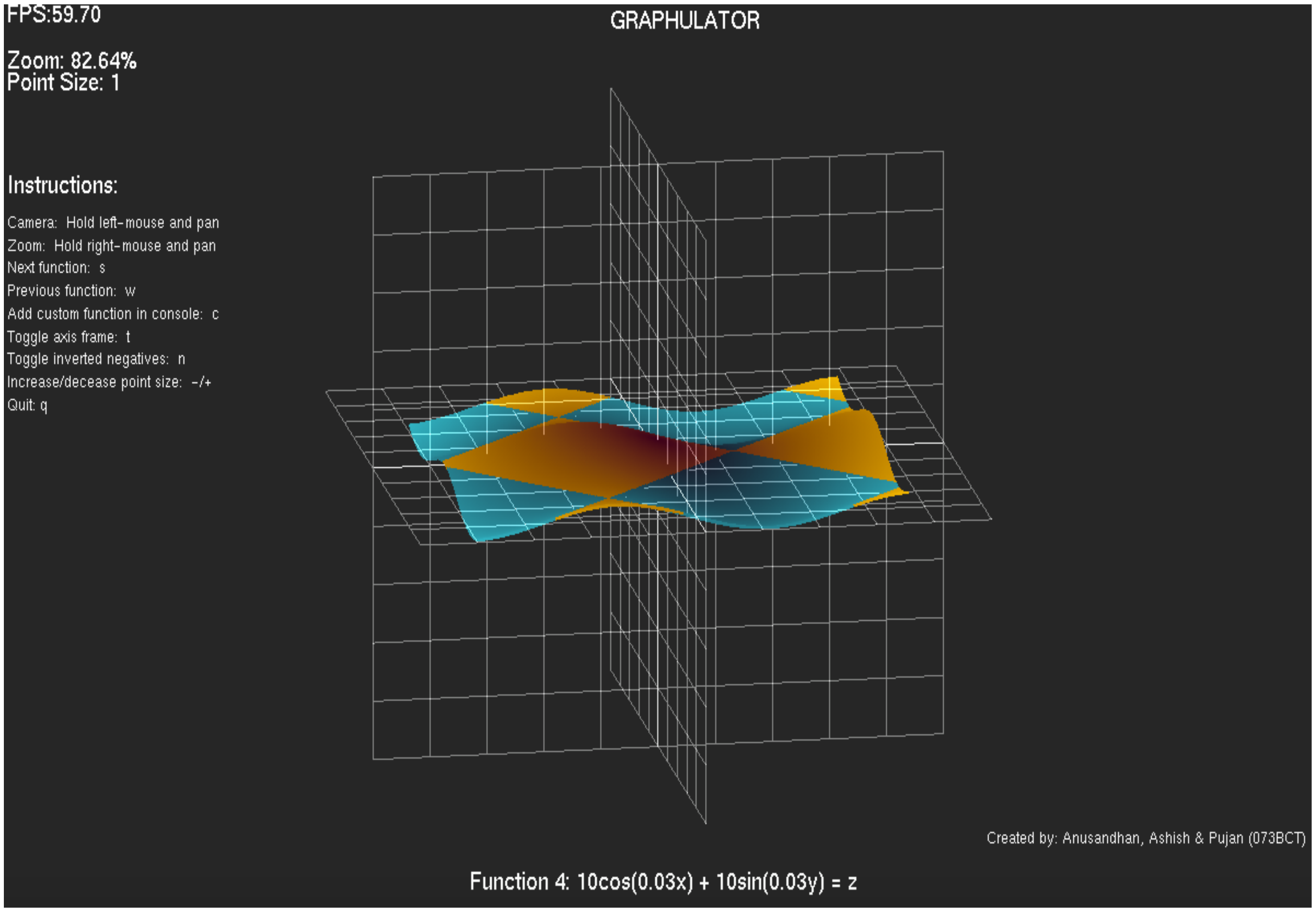
APPLICATIONS

Graphulator can be used in Mathematical equations problems. It can contribute a little for making easier in various complex problem. The nature of graphs shown by various equations can be observed. It can be implemented in solving problems which can save time of plotting the graphs in paper which is probably time consuming (let's say) thing to do. This can be used in Mathematics, the premise of modern education which can be applicable from low standards to higher level mathematics.

IMPLEMENTATION



APPENDIX:TEST DOCUMENTATION



Instructions:

Camera: Hold left-mouse and pan
Zoom: Hold right-mouse and pan
Next function: s
Previous function: w
Add custom function in console: c
Toggle axis frame: t
Toggle inverted negatives: n
Increase/decease point size: -/+
Quit: q

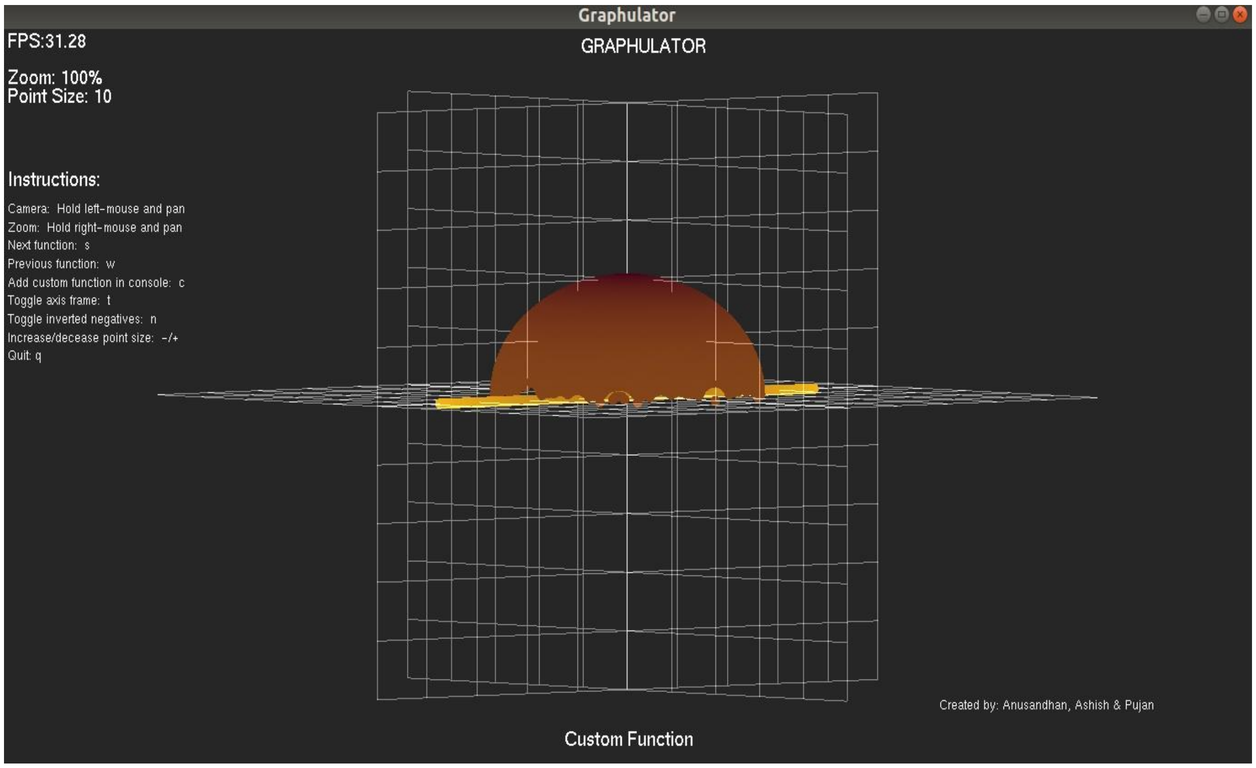
Created by: Anusandhan, Ashish & Pujan (073BCT)

Function 4: 10cos(0.03x) + 10sin(0.03y) = z

The above screen shows graph nature shown by equation:

$10\cos(0.03x) + 10\sin(0.03y) = z$;

Where x, y, z can have random values. The blue plots show negative axis part of the graph.


Graphulator
FPS:31.28
GRAPHULATOR
Zoom: 100%
Point Size: 10
Instructions:
Camera: Hold left-mouse and pan
Zoom: Hold right-mouse and pan
Next function: s
Previous function: w
Add custom function in console: c
Toggle axis frame: t
Toggle inverted negatives: n
Increase/decease point size: -/+
Quit: q
Created by: Anusandhan, Ashish & Pujan
Custom Function

Hemisphere in Graphulator

APPENDIX: REFERENCE LINKS

- ✓ https://en.wikipedia.org/wiki/OpenGL_Utility_Toolkit
- ✓ https://en.wikipedia.org/wiki/Shunting-yard_algorithm
- ✓ https://www.youtube.com/channel/UCiFAmp2Crv66cQA-9SPje1A
- ✓ https://www.youtube.com/user/sonarsystemslimited
- ✓ https://www.youtube.com/user/iamdavidwparker
- ✓ GOOGLE (Obvious)

RESULT

Hence, the project was completed by the team after active involvement, research and efforts. Despite hard work we couldn’t add all the features as proposed. But, the project developed the qualities like team work, time management, and co-operation in team members for backing up each other and to cope the challenges together.

CHALLENGES FACED:

We, team members faced various challenges during the completion of project. Since the Glut was new to every team member, it took a bit more time to learn it and write the programs using it. Due to which we got many bugs which were very difficult to debug properly. We couldn't give more time for project due to many internal and external reasons. Despite the challenges, we tried our best for the project.

CONCLUSION:

Finally, we are able to conclude our project which developed different skills among us and enlightened us the way of team co-operation. We were able to improvise our participation in the project. The sheer force of team work and companionship made the project successful and here we are concluding our project **“GRAPHULATOR”**.

APPENDIX: SOURCE CODE

```
#include
<cstdlib
>


            #ifndef __APPLE__
            #include <GL/glew.h>
            #endif


            #include "glut.h"
            #include <cstdio>
            #include <cstring>
            #include <iostream>
            #include <cmath>


            #include "expression/expressions.h"


            const int gWidth  = 1200;
            const int gHeight = 800;


            float gXAngle = 45.0f;
            float gYAngle = -45.0f;
            //float gZAngle = 90.0f;
            int   gXOrigin = -1;
            int   gYOrigin = -1;


            bool  gDoZooming = false;
            float gZoom = 1.0f;


            int  gFrame=0,gTime,gTimebase=0;
            char gFrameString[50];
char gZoomString[50] = "Zoom: 100%";
char gPointString[50] = "Point Size: 1";
char gFunctionString[1024] = "Function 1: ln(x^2) + ln(y) = z";
char main_name[50]= "GRAPHULATOR";


float  gGraphRes = 300;
float  gGraphBounds = 300;
int    gFrameScaling = gGraphRes/100+1;
int    gFrameSpacing = 10;
float* gPointCache = 0;
bool   gDrawGrid = true;
bool   gDrawNegatives = false;
int    gPointSize = 1;




int  gFunctionID = 0;
bool gFunctionChanged = true;
BaseExpression* gCustomFunction;


//Window's ID
static int gWinID;


//Renders strings
void renderCharacters(float pX, float pY, float pZ, void *pFont, char
*pCharArray)
{
       //Position sets here
       glRasterPos3f(pX, pY, pZ);


       //'\0' array loop renders
       for ( char* lChar = pCharArray; *lChar != '\0'; lChar++)
       {
              glutBitmapCharacter(pFont, *lChar);
       }
}
```

```
//renders Axes to RGB color scheme
void renderAxisFrame()
{
       //For transparency
    glEnable(GL_BLEND);
       glBlendFunc(GL_SRC_ALPHA,GL_ONE);


       //X  grid(YZ Plane)
       glBegin(GL_LINES);
       for(int i=0;i<=gFrameSpacing;i++)
       {
        glColor4f(0.8,0.8,0.8,0.5);
              //Grid Renders here
              glVertex3f((gFrameSpacing/2-i)*gFrameScaling,0,-
gFrameSpacing/2*gFrameScaling);
              glVertex3f((gFrameSpacing/2-
i)*gFrameScaling,0,gFrameSpacing/2*gFrameScaling);
              glVertex3f(-gFrameSpacing/2*gFrameScaling,0,(gFrameSpacing/2-
i)*gFrameScaling);
              glVertex3f(gFrameSpacing/2*gFrameScaling,0,(gFrameSpacing/2-
i)*gFrameScaling);


       }
       glEnd();




       //Y wala grid(XZ Plane)
       glBegin(GL_LINES);
       for(int i=0;i<=gFrameSpacing;i++)
       {
              glColor4f(0.8,0.8,0.8,0.5);


              //Render here
              glVertex3f(0,(gFrameSpacing/2-i)*gFrameScaling,-
gFrameSpacing/2*gFrameScaling);
              glVertex3f(0,(gFrameSpacing/2-
i)*gFrameScaling,gFrameSpacing/2*gFrameScaling);
```

```
		glVertex3f(0,-gFrameSpacing/2*gFrameScaling,(gFrameSpacing/2-
i)*gFrameScaling);
		glVertex3f(0,gFrameSpacing/2*gFrameScaling,(gFrameSpacing/2-
i)*gFrameScaling);
	}
	glEnd();


	//Z grid(XY Plane)
	glBegin(GL_LINES);
	for(int i=0;i<=gFrameSpacing;i++)
	{
		glColor4f(0.8,0.8,0.8,0.5);


		//Render
		glVertex3f(-gFrameSpacing/2*gFrameScaling,(gFrameSpacing/2-
i)*gFrameScaling,0);
		glVertex3f(gFrameSpacing/2*gFrameScaling,(gFrameSpacing/2-
i)*gFrameScaling,0);
		glVertex3f((gFrameSpacing/2-i)*gFrameScaling,-
gFrameSpacing/2*gFrameScaling,0);
		glVertex3f((gFrameSpacing/2-
i)*gFrameScaling,gFrameSpacing/2*gFrameScaling,0);
	}
	glEnd();


	//Transparency removed
	glDisable(GL_BLEND);
}




//Renders UI here
void renderUI()
{
	//FDisplays FPS
	gFrame++;
	gTime = glutGet(GLUT_ELAPSED_TIME);
```

```
        //Calculates in each sec
        if (gTime - gTimebase > 1000)
        {
                //FPS display
                sprintf(gFrameString,"FPS:%4.2f", gFrame*1000.0/(gTime-
gTimebase));
                gTimebase = gTime;
                gFrame = 0;
        }


        //Text color for frame rate
        glColor3f(1.0f,1.0f,1.0f);


        //GL ko camera lai 2D orthogonal plane ma render garna set gareko
        glMatrixMode(GL_PROJECTION);
        glPushMatrix();
        glLoadIdentity();
        gluOrtho2D(0, gWidth, gHeight, 0);
        //Go back the model view
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();


        //Frame rate renders
        renderCharacters(5,20,0,GLUT_BITMAP_HELVETICA_18,gFrameString);


        //Information  text render
    //Helvetica font used
        renderCharacters(5,60,0,GLUT_BITMAP_HELVETICA_18, gZoomString);
        renderCharacters(5,80,0,GLUT_BITMAP_HELVETICA_18, gPointString);
    renderCharacters(5,170,0,GLUT_BITMAP_HELVETICA_18,"Instructions:");
        renderCharacters(5,200,0,GLUT_BITMAP_HELVETICA_12,"Camera:  Hold left-
mouse and pan");
        renderCharacters(5,220,0,GLUT_BITMAP_HELVETICA_12,"Zoom:  Hold right-
mouse and pan");
        renderCharacters(5,240,0,GLUT_BITMAP_HELVETICA_12,"Next function:  s");
        renderCharacters(5,260,0,GLUT_BITMAP_HELVETICA_12,"Previous function:
w");
        renderCharacters(5,280,0,GLUT_BITMAP_HELVETICA_12,"Add custom function
in console:  c");
```

```
	renderCharacters(5,300,0,GLUT_BITMAP_HELVETICA_12,"Toggle axis frame: 
t");
	renderCharacters(5,320,0,GLUT_BITMAP_HELVETICA_12,"Toggle inverted 
negatives:  n");
	renderCharacters(5,340,0,GLUT_BITMAP_HELVETICA_12,"Increase/decease 
point size:  -/+");
    renderCharacters(5,360,0,GLUT_BITMAP_HELVETICA_12, "Quit: q");
    renderCharacters(gWidth-300,gHeight-60,0,GLUT_BITMAP_HELVETICA_12,"Created 
by: Anusandhan, Ashish & Pujan");
    renderCharacters((gWidth - 
strlen(main_name)*8)/2,25,0,GLUT_BITMAP_HELVETICA_18, main_name);
	//Render the function name
	renderCharacters((gWidth - strlen(gFunctionString)*8)/2,gHeight-
20,0,GLUT_BITMAP_HELVETICA_18, gFunctionString);


	glPopMatrix();
}


/*Yaha chai function render hune main kaam huncha
 function lai point haru ko cloud jasto dataset ma render garcha
 */
void renderPoints()
{
	//Agi projection load garejastai yaha model view matrix mode load garcha
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();


	//Aba points haru render huna suru vaio
    glBegin(GL_POINTS);





	int index = 0;
    /*
     Yaha chai x,y,z coordinates haru plot huncha function chai z=f(x,y) ko 
form ma huna paryo
     Coordinate plot garni bela max kati ota plot garne vanne tha hunna so k 
garne vanda graph ko resolution jati cha teskai over loop garera points haru 
banaune
```

```
     Jastai hamro default graph ko resolution 300 cha vanesi hamle -150 to 150
samma loop lagayera points plot garchau
     Yesle garda k faida huncha vane zoom in/out garda hamro graph pani change
huncha with respect to the grid
     */
       for(float i =-gGraphRes/2.0f; i<gGraphRes/2.0f;
i+=gGraphRes/gGraphBounds)
       {
              for(float j=-gGraphRes/2.0f; j<gGraphRes/2.0f;
j+=gGraphRes/gGraphBounds)
              {
            /*
             i ra j vaneko x ra z vaihalyo
             x ra z plot garisakesi aba teslai kati height ma elevate garne
vanne y coordinate le dincha
             kina vane opengl ko coordinate system vaneko right handed
coordinate system ho
             */
                     float y = 0;



                     /*Mathi bata x ra z ko value aaucha ani yaha aayera height
calculate huncha
             function aanusar
            */
                     if(gFunctionChanged)
                     {
                            switch(gFunctionID)
                            {
                                   case -1:
                                          /*Custom Function Banauda chai tyo
fuction ko calculation expressions vanne header file ma gayera huncha*/
                                          y= gCustomFunction->evaluate(i,j);
                                          strcpy(gFunctionString, "Custom
Function");

                                          break;
                                   case 0:
                                          y = log(i*i) + log(j);
                                          strcpy(gFunctionString,"Function 1:
ln(x^2) + ln(y) = z");

                                          break;
```

```
                                case 1:
                                        y = (i*i + (j * j))/200;
                                        strcpy(gFunctionString,"Function 2:
(x^2 + y^2)/200 = z");

                                        break;


                                case 2:
                                        y = pow((i*i+j*j),0.5);
                                        strcpy(gFunctionString,"Function 3:
z=(x^2+y^2)^0.5");

                                        break;


                                case 3:
                                        y= cos(i*0.03f)*10 +
sin(j*0.03f)*10;

                                        strcpy(gFunctionString,"Function 4:
10cos(0.03x) + 10sin(0.03y) = z");

                                        break;


                                case 4:
                        y= 1-std::abs(i+j)-std::abs(j-i);


                                        strcpy(gFunctionString,"Function 5:
z=1-abs(x+y)-abs(y-x)");

                                        break;


                                case 5:
                                        y = ((i*i)-(j*j));
                                        strcpy(gFunctionString,"Function 6:
x^2 - y^2 = z");

                                        break;


                                default:
                                        y = ((j*0.01*j*i + i * i)/300);
                                        strcpy(gFunctionString,"Function 7:
0.01y^3 + x^2 = z ");

                                        break;
```

```
                    }



                    gPointCache[index] = y;
                    index++;
                }
                else
                {
                    y = gPointCache[index];
                    index++;
                }
            /*
             yaha chai gradient haleko cha, gradient kati halne(2 ta colors)
vanne chai mathi bata aako i ra j ko value bata calculate gareko cha
             tesle garda point aanusar farak farak color shades aaucha ani
color blend vajasto dekincha (zoom out huda)
             */
                float xPartial = (i)/1200.0f;
                float yPartial = (j)/1200.0f;
                float gradient = xPartial*xPartial + yPartial * yPartial;
                gradient = sqrt(gradient);


            if(y<0 && gDrawNegatives)
                glColor3f(0.1,gradient*4,(gradient*4+0.05));
            else
                glColor3f((gradient*4+0.25),gradient*4,0.1);


                glVertex3f(j*0.1f,y*0.1f,i*0.1f);



              }
        }


        /*Ek choti banaisakesi feri banauna naparos vanera arko choti tyo
function ko case aauda cache bata load huncha*/
        gFunctionChanged = false;
        glEnd();
}


//Scene Render hune main loop
void renderScene()
{
    //Color ra Depth data lai clear gareko suru ma
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);


    //glMatrixMode feri projection mode ma lageko(reset gareko)
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();


    //suru ma 45 degree ko field of view, aspect ratio window ko
(width/height), ani near to far range 0.1 to 1000 units ko rakhera hercha
    //3D view ko 2D perspective view nikaleko
        //gluPerspective(45.0f*gZoom,(GLfloat)gWidth/(GLfloat)gHeight,0.1f,1000.
0f);
    glOrtho(-25*gZoom*(GLfloat)gWidth/(GLfloat)gHeight,
25*gZoom*(GLfloat)gWidth/(GLfloat)gHeight, -25*gZoom, 25*gZoom, 0.1f, 1000.0f);


    //origi lai chai (0,50,-35) vanne point bata herne
    //desired axis ko around ma rotate garni ani
        gluLookAt(0, 50*gGraphRes/gGraphBounds, -35*gGraphRes/gGraphBounds, 0,
0, 0, 0, 1, 0);
        glRotatef(gXAngle,0,1,0);


        //The axis we rotate on for the vertical component depends on our angle
around the vertical. The vector around which we rotate is therefore defined
        //by x = cos(angle) and z = sin(angle), as per the trig unit circle
        glRotatef(-
gYAngle,cos(gXAngle*3.14159/180.0f),0,sin(gXAngle*3.14159/180.0f)); //radian to
degree


    //Rendering the function now
        renderPoints();
        //Render the axis frame if needed
        if(gDrawGrid)
                renderAxisFrame();


        //Render the UI
        renderUI();


        //Swap the buffers and prepare to render again
    glutSwapBuffers();
        glutPostRedisplay();
}





//first screen
void firScr(){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);





}


//Keyboard ko input yaha handle huncha
void keyboard (unsigned char pKey, int pX, int pY)
{
        //s thichyo vane esle function ko id lai increment gardincha
    //last ko function id ma pugesi chai jati s thiche ni static nai rahancha
        if ((pKey=='s' || pKey =='s') && gFunctionID < 6)
    {
                gFunctionID++;
                gFunctionChanged = true;
    }
    //w thichyo vane esle function ko id lai decrement gardincha
    //first ko function id ma pugesi chai jati w thiche ni static nai rahancha
        else if((pKey =='w' || pKey == 'W') && gFunctionID > 0)
        {
                gFunctionID--;
                gFunctionChanged = true;
        }
        else if(pKey == 'c' || pKey == 'C')
        {
                std::string lString;
                getline (std::cin,lString);


                gCustomFunction = processString(lString);
                gFunctionID = -1;
                gFunctionChanged = true;
        }
        else if(pKey == 't' || pKey == 'T')
        {
                gDrawGrid = !gDrawGrid; //Grid on/off toggle gareko
        }
        else if(pKey == 'n' || pKey == 'N')
        {
                gDrawNegatives = !gDrawNegatives; //negative planes ko lai
chuttyaune on/off gareko
        }
    //point ko size badhaune
        else if(pKey == '=' || pKey == '+')
        {
                if(gPointSize < 10)
                        glPointSize(++gPointSize);


                sprintf(gPointString,"Point Size: %i",gPointSize); //gPointString
vanne string ma print garcha
        }
    //point ko size ghataune
        else if(pKey == '-' || pKey == '_')
        {
                if(gPointSize > 1)
                {
                        glPointSize(--gPointSize);
                }


                sprintf(gPointString,"Point Size: %i",gPointSize);
        }
    else if(pKey == 'q' || pKey == 'Q')
    {
        glutDestroyWindow(gWinID); // Window destroy garne
        delete gPointCache; //Memory sarsafai gareko
        delete gCustomFunction;
        exit(0);
    }
}


//Mouse ko motion yaha handle huncha
void mouseMove(int pX, int pY)
{
       if(gDoZooming)
       {
              if(gXOrigin - pX > 0 && gZoom <= 2)
                     gZoom += 0.03;
              else if(gZoom >= 0.1 && (gXOrigin - pX) < 0)
                     gZoom -= 0.03;


              gXOrigin = pX;
              sprintf(gZoomString,"Zoom: %4.2f%%", 100.0f/gZoom);
       }
       else
       {
        //Mouse le tyakka click garesi origin set huncha ani hamle rotate
garchau
              if (gXOrigin >= 0)
              {
            //angle milaune kaam (on the basis of x delta)
                     gXAngle += (pX - gXOrigin) * 0.25f;


                     //Reset the origin
                     gXOrigin = pX;
              }


              //Y position ko lagi ni same
              if(gYOrigin >= 0)
              {
                     gYAngle += (pY - gYOrigin) * 0.25f;
                     gYOrigin = pY;
```

```
                }
        }
}


//Yaha chai mouse ko button ko events haru handle huncha
void mouseButton(int pButton, int pState, int pX, int pY)
{
    //Left button click garecha vane
        if (pButton == GLUT_LEFT_BUTTON)
        {
         //left button thichirakheko cha vane gXOrigin ra gYOrigin lai mouse ko
point ko value set garne
         //natra button release garesi origin clear gardine
                if (pState == GLUT_UP)
                {
                        gXOrigin = -1;
                        gYOrigin = -1;
                }
                else
                {
                        gXOrigin = pX;
                        gYOrigin = pY;
                }
        }
    //Right Button Click Garda Ko Kura
        else if(pButton == GLUT_RIGHT_BUTTON)
        {
         //Zoom garne nagarne yaha decide huncha
                if(pState == GLUT_UP)
                {
                        gDoZooming = false;
                }
                else
                {
                        gDoZooming = true;
                        gXOrigin = pX;
                }
        }


}
```

```
int main(int argc, char **argv)
{


        //Glut init
        glutInit(&argc, argv);


        //Display mode yaha
        glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);


        //Window ko settings
        glutInitWindowPosition(100,100);
        glutInitWindowSize(gWidth,gHeight);
        gWinID = glutCreateWindow("Graphulator"); //Window ko name


        //Keyboard/Mouse input handlers
        glutKeyboardFunc(keyboard);
        glutMouseFunc(mouseButton);
        glutMotionFunc(mouseMove);


    //Display/Render calls haru handle garne function
    glutDisplayFunc(firScr);
    glutDisplayFunc(renderScene);


    //Main canvas ko color ra depth
        //glClearColor(0.15f, 0.15f, 0.15f, 0.1f);//R,G,B ra alpha
        glClearDepth(1.0f);


    //Smooth Banaune ninja technique :D
        glShadeModel(GL_SMOOTH);
        glEnable(GL_POINT_SMOOTH); //Point smooth banayena vane cube ko dhikka
jasto aaucha
        glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
        glPointSize(gPointSize);
    //GL ko depth test enable gareko
        glEnable(GL_DEPTH_TEST);
        glDepthFunc(GL_LEQUAL);


    //Yo chai cache ko lagi memory allocate gareko
        gPointCache = (float*)malloc(gGraphRes * gGraphRes * sizeof(float));


    TrigExpression *t = new TrigExpression(new LogExpression(new
AddExpression(XVAR,new NumberExpression(10.5)), 0),eCos);


    //GL ko main loop start garne, until window is closed.
        glutMainLoop();


        //Memory ra Cache Clean Gardeko
        if(gCustomFunction)
               delete gCustomFunction;


        delete gPointCache;


        //La sakkiyo
        return 0;
}
```