|  |
| --- |
| **NATIONAL INSTITUTE OF TECHNOLOGY ,SILCHAR** |
|  |
| **LIMITING VALUE PLOTTER** |
| COMPUTER GRAPHICS PROJECT |
|  |

**SUBMITTED BY:-**

1)**RIDDHIMAN PARASAR**(16-1-5-005)

2)**RAJDEEP SARMA** (16-1-5-039)

**INTRODUCTION**

The project ‘Limiting Value Plotter’ is based on the concept of value of the limit of a function where it approaches a particular value. It is represented mathematically as limx->af(x)=f(a), where f(x) is a function of x and a is the value of the constant to which the function approaches.

The project has been implemented using open source IDE Codeblocks with the help of the open source graphics library openGL.

The program consists of three limiting trigonometric functions viz sine,cos and tan plotter where the original graph of the system transforms itself into its limiting value using animation.

**OPENGL FUNCTIONS**

* void **glClearColor**(GLfloat  red, GLfloat  green, GLfloat  blue, GLfloat  alpha)

[glClearColor()](http://code.nabla.net/doc/OpenGL/api/OpenGL/man4/glClearColor.html#glClearColor) specifies the red, green, blue, and alpha values used by [glClear()](http://code.nabla.net/doc/OpenGL/api/OpenGL/man4/glClear.html#glClear) to clear the color buffers. Values specified by [glClearColor()](http://code.nabla.net/doc/OpenGL/api/OpenGL/man4/glClearColor.html#glClearColor) are clamped to the range [0,1].

* void **glClear**(GLbitfield  mask)

[glClear()](http://code.nabla.net/doc/OpenGL/api/OpenGL/man4/glClear.html?highlight=glclear#glClear) takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.

The values are as follows:

GL\_COLOR\_BUFFER\_BIT

Indicates the buffers currently enabled for color writing.

GL\_DEPTH\_BUFFER\_BIT

Indicates the depth buffer.

GL\_STENCIL\_BUFFER\_BIT

Indicates the stencil buffer.

* **glBegin**— delimit the vertices of a primitive or a group of like primitives

void glBegin(GLenum mode);

mode

Specifies the primitive or primitives that will be created from vertices

presented between glBegin and the subsequent glEnd.

Ten symbolic constants are accepted:

GL\_POINTS,GL\_LINES,GL\_LINE\_STRIP, GL\_LINE\_LOOP,GL\_TRIANGLES,GL\_TRIANGLE\_STRIP,GL\_TRIANGLE\_FAN,GL\_QUADS,GL\_QUAD\_STRIP, and GL\_POLYGON

* **glColor** — set the current color

red, green, blue

Specify new red, green, and blue values for the current color.

alpha

Specifies a new alpha value for the current color.

Included only in the four-argument glColor4 commands.

* **glVertex2d** – specifies a vertex

## Parameters

*x*

Specifies the x-coordinate of a vertex.

*y*

Specifies the y-coordinate of a vertex.

* **GlEnd** - delimit the vertices that define a primitive or a group of like primitives
* **glutPostRedisplay()**- essentially sets a flag so that on the next iteration of the mainloop, the registered display() function is called.
* **glFlush** - forces execution of OpenGL functions in finite time.
* **glutInit** -is used to initialize the GLUT library.
* **glutInitWindowSize**- sets the initial window size.
* **glutInitWindowPosition**-sets the initial window position.
* **glutCreateWindow**- creates a top-level window.
* **glutDisplayFunc**- sets the display callback for the current window.
* **glutMainLoop**- enters the GLUT event processing loop.

**USER DEFINED FUNCTIONS**

* **void PlotSine()** **-** A user defined function to initially plot sine curve and show its transformation to its limiting value of the function

limx->0 (sin(x)/x)=1.

* **void PlotCos() -** A user defined function to initially plot sine curve and show its transformation to its limiting value of the function

limx->0 cos(x)=1.

* **void PlotTan() -** A user defined function to initially plot sine curve and show its transformation to its limiting value of the function

limx->0 (tan(x)/x)=1.

**CODE**

**#include <gl/freeglut.h>**

**#include <iostream>**

**#include <cmath>**

**#define PRECISION 0.0001**

**#define ZOOM 0.01**

**#define PI 3.14159265359**

**double scaleFactorX = 1 /(4 \* PI), scaleFactorY = 0.25;**

**using namespace std;**

**void plotSine() {**

**double x = -4\*PI, y;**

**glClearColor(0.0f,0.0f,0.0f,1.0f);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glBegin(GL\_LINES);**

**glColor3f(0.5, 0.5, 0.5);**

**glVertex2d(-1, 0);**

**glVertex2d(1, 0);**

**glVertex2d(0, 1);**

**glVertex2d(0, -1);**

**glEnd();**

**glBegin(GL\_POINTS);**

**glColor3f(1, 1, 1);**

**do {**

**y = sin(x);**

**x = x + PRECISION;**

**glVertex2d(x \* scaleFactorX, y \* scaleFactorY);**

**} while(x<=4\*PI);**

**glEnd();**

**glFlush();**

**scaleFactorX += ZOOM;**

**scaleFactorY += ZOOM;**

**glutPostRedisplay();**

**}**

**void plotCos() {**

**double x = -4\*PI, y;**

**glClearColor(0.0f,0.0f,0.0f,1.0f);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glBegin(GL\_LINES);**

**glColor3f(0.5, 0.5, 0.5);**

**glVertex2d(-1, 0);**

**glVertex2d(1, 0);**

**glVertex2d(0, 1);**

**glVertex2d(0, -1);**

**glVertex2d(-1, 1 \* scaleFactorY);**

**glVertex2d(1, 1 \* scaleFactorY);**

**glEnd();**

**glBegin(GL\_POINTS);**

**glColor3f(1, 1, 1);**

**do {**

**y = cos(x);**

**x = x + PRECISION;**

**glVertex2d(x \* scaleFactorX, y \* scaleFactorY);**

**} while(x<=4\*PI);**

**glEnd();**

**glFlush();**

**scaleFactorX += ZOOM;**

**glutPostRedisplay();**

**}**

**void plotTan() {**

**double x = -4\*PI, y;**

**glClearColor(0.0f,0.0f,0.0f,1.0f);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glBegin(GL\_LINES);**

**glColor3f(0.5, 0.5, 0.5);**

**glVertex2d(-1, 0);**

**glVertex2d(1, 0);**

**glVertex2d(0, 1);**

**glVertex2d(0, -1);**

**glEnd();**

**glBegin(GL\_POINTS);**

**glColor3f(1, 1, 1);**

**do {**

**y = tan(x);**

**x = x + PRECISION;**

**glVertex2d(x \* scaleFactorX, y \* scaleFactorY);**

**} while(x<=4\*PI);**

**glEnd();**

**glFlush();**

**scaleFactorX += ZOOM;**

**scaleFactorY += ZOOM;**

**glutPostRedisplay();**

**}**

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

**int type;**

**glutInit(&argc, argv);**

**cout<< "Select option : \n 1) Sine \n 2) Cosine \n 3) Tangent \n ";**

**cin >> type;**

**glutInitWindowSize(500, 500);**

**glutInitWindowPosition(800, 100);**

**glutCreateWindow("Project MathemGL: Display");**

**switch(type) {**

**case 1 :**

**glutDisplayFunc(plotSine);**

**cout<<endl<<"lim (x->0) (sin x / x ) = 1"<<endl;**

**break ;**

**case 2 :**

**glutDisplayFunc(plotCos);**

**cout<<endl<<"lim (x->0) (cos x) = 1"<<endl;**

**break ;**

**case 3 :**

**glutDisplayFunc(plotTan);**

**cout<<endl<<"lim (x->0) (tan x / x ) = 1"<<endl;**

**break ;**

**default:**

**cout<<"Enter a value b/w 1 and 4"<<endl<<endl;**

**break;**

**}**

**glutMainLoop();**

**return 0;**

**}**

**CONCLUSION**

This project was well tested I both windows and Ubuntu(Linux) operating systems. It helps in understanding the concept of limit of a function in a graphical way.

The project enabled to work with midlevel OpenGl complexity and the project demonstrates the scope of OpenGl platform in solving mathematical problems. It serves as a stepping stone for venturing into other fields of Computer Graphics and applications.

**ACKNOWLEDGEMENT**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organisations. We would like to extend our sincere thanks to all of them.

Firstly, we are highly indebted to Mr Badal Soni for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

We would also like to thank our colleagues in developing the project and people who have willingly helped us with their abilities.