VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

Mini Project Report on

“SIMULATION OF CONIC SECTIONS”

Submitted in the partial fulfillment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work

Submitted By

P. ROHIT	USN: 1BY17CS106
RUTVI RAHUL GADRE	USN: 1BY17CS144

Under the guidance of

Mr. SHANKAR R
Assistant Professor, CSE, BMSIT&M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU - 560064.

2019-2020

BMS INSTITUTE OF TECHNOLOGY &MANAGEMENT
YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"SIMULATION OF CONIC SECTIONS"**is a bonafide work carried out by **P.Rohit (1BY17CS106) and Rutvi Rahul Gadre (1BY17CS144)** in partial fulfillment for *Mini Project* during the year 2019-2020. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*. It is also certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report.

Signature of the Guide with date	**Signature of HOD with date**
Mr. SHANKAR R	Dr. Anil G N
Assistant Professor	Prof & Head
CSE, BMSIT&M	CSE, BMSIT&M

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical, and environment friendly for the betterment of the society.

INSTITUTE MISSION

Accomplish a stimulating learning environment through high-quality academic instruction, innovation, and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as a valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance, and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analyzing, and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team-building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance, and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and everyone who has helped us make this project a success.

We heartily thank our Principal, Dr. MOHAN BABU G N, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, Dr. ANIL G N, Department of Computer Science and Engineering, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We thank our Project Guide, Mr. Shankar R, Assistant Professor, Department of Computer Science and Engineering for his intangible support and for being a constant backbone for our project.

Special thanks to all the staff members of the Computer Science Department for their help and kind cooperation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

P.ROHIT (1BY17CS106)

RUTVI RAHUL GADRE (1BY17CS144)

ABSTRACT

To implement standard math's interactive animations using OpenGL, to make understanding of concepts easier for students. This project aims to create a visual representation for concepts that are generally tough and pretty complex for students to understand in math such as the Conic Sections. This visual representation is done by using C/C++ and OpenGL. OpenGL is an open-source library for C/C++.

The OpenGL Utility Library (GLU) provides many modeling features to utilize computer graphics. GLU is a standard part of every OpenGL implementation. Users can select from a number of options to view the simulation of the following: Sine and Cosine graphs, locus of a circle, ellipse, Formation of conic sections from a cone.

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and the Utah teapot. GLUT also has some limited support for creating pop-up menus.

This simulation will show an animation to help explain the concepts of the formation of the mentioned topics to help clear understanding.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Brief Introduction

Conic Sections refer to the formation of a figure by the intersection of a plane and a circular cone. Depending on the angle of the plane with respect to the cone, a conic section maybe is an ellipse, a parabola, a hyperbola, etc. The real-world applications of conic sections are widely known. They are used for a better understanding of three-dimensional geometry which are helpful for the study of electromagnetic field theory, designing of antennas, parabolic reflectors, etc. They are also used in creating engineering drawings which is the basic plan of components designed by Mechanical Engineers. They are used in Space research to study the orbit of various plants and their relative motion. A visual aid is always helpful to gain an insight into the theoretical concepts that we study. Seeing images of what's being taught is a powerful way to build student engagement and boost retention. Not only do they provide supplementary information to students, but the visual aids show images that allow them to connect a topic to what it looks like. Further, visual aids can promote deeper thinking and build overall critical thinking skills. In fact, bringing a visual aid into your classroom opens up a whole new realm of educational opportunities. The simulation of conic sections allows us to study them in detail. Students can use this product to get clarity of the mathematical models and understand them in a better manner.

1.2 Motivation

The motivation behind this project is to try and help the students who are not able to understand difficult concepts of mathematics clearly, there are so many students studying in India and not everyone's the same. Each child is different, where one may find a subject very easy, another may find it extremely difficult. Topics in mathematics such as Conic Sections and Volumes & Areas, etc. require students to visualize them in order to understand the topic thoroughly. This project allows students to easily view a visualization in the form of a graphical animation which demonstrates the formation and cross-section of Conic sections and thus spends less time trying to get a mental picture and understand the concept clearly.

1.3 Scope

The aim is to allow widespread use of animation aids like this to use in schools so that students will be benefited, and to create more of such applications which will help the students understand the concepts more clearly and quickly.

1.4 Problem Statement

To implement Standard Math's Interactive Animations using OpenGL, to make understanding of the important concepts of mathematics using Animations.

1.5 Proposed System

The proposed system is an open-source one and does not charge students to run their own simulations. It is as sophisticated and advanced as the curriculum followed by the educational institutions and hence is useful to bridge the gap between theoretical learning and practical learning. We have shown graphical proofs of standard limits using the plot of trigonometric functions. We have also shown how different conic sections can be drawn using pencil, thread, and fixed pins (with fixed lines in case of a parabola). We have visualized how conic sections are formed from the cross-section of a cone.

1.6 Limitations

● Subscription-based learning Some of these websites and courses are available to students only after they pass a subscription paywall. This comes as a burden to those students who cannot afford to pay such premiums.

● Not different in terms of what they offer

● These methods of learning for the students are not all that different from what they learn in school and although they can understand concepts in a clearer way due to the self-paced nature, they sometimes will not understand complex topics as the explanations are not all different from what they learn in school.

CHAPTER 2: SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS:

- Programming Language: C++
- Graphics Library: FreeGLUT, GL/glut.h
- Operating System: Linux
- Compiler: GCC
- OpenGL

HARDWARE REQUIREMENTS:

- A computer
- Dual-core processor
- 2GB RAM
- 40GB HDD
- Keyboard
- Mouse

CHAPTER 3: LITERATURE SURVEY

There are quite a few options in the market when it comes to learning math in an innovative way like Khan Academy, online course websites, etc. All these technologies sure make it easier and allow students to get self-paced learning and get a better understanding of the concepts. They teach short lessons in the form of videos. These websites also include supplementary practice exercises and materials for educators. They provide a personalized learning experience, mainly built on the videos which are hosted on youtube. These websites are meant to be used as a supplement to their videos because they include other features such as progress tracking, practice exercises, and teaching tools.

CHAPTER 4: IMPLEMENTATION

PlotSine and PlotCos and PlotTan:

These functions plot the sine and cosine graphs respectively on the output window. The 2D orthographic scale is -1 to 1 on both x and y axes.

The y = x line is drawn to show how it forms a tangent to the curve. This is done using the GL_LINES command.

The curves are drawn with the help of the respective functions from the math.h standard library, the render is also continuously redrawn with a zoom feature which zooms in on the origin and we can control the precision of the curve drawn with the precision preprocessor directive.

PlotCircle:

The circle is drawn using the equation: x^2 + y^2 = 1

This function animates the drawing of a circle by using a circle parametric equation, ad drawing a line from the origin to the circle's circumference, and drawing each point on the circumference using GL_POINTS pixel by pixel.

The line used to draw the circle is denoted in different colors to be able to distinguish the animation.

The speed with which the circle is drawn can be controlled by adjusting the change in the parameters of the circle.

And as soon as the locus becomes the starting drawing point after starting the drawing the animation is concluded.

PlotEllipse:

The ellipse is drawn using the equation: x^2 / 64 + y^2 / 9 = 1

The animation uses the principal similar to the circle animation, but in case of the ellipse, the drawing arm extends from both foci together and draws the ellipse in one motion.

The speed of the drawing can also be controlled as with the circle animation.

PlotParabola:

The parabola is drawn using the equation: y^2 = 4 * a * x

The animation contains two arms, one from the y axis, and another from the focus of the parabola.

The arms contain different color to distinguish the animation with ease. The speed of the drawing can be controlled.

The Directrix/Latus Rectum is drawn with a light grey color, and so is the line joining the parabola and the Directrix.

Conic Sections:

The different parts of a Cone are the Circle, Ellipse, Parabola, and Hyperbola.

These are drawn with the help of the following functions:

- generateConic: This is the display callback function which makes use of the glut wire cone primitive in the GLUT library.
- reshapeConic: This is the reshape callback function which rotates the perspective according to the specified angle and shifts the axis to generate the specified Conic section.

The animation starts off by rendering the Wire Cone on the output window, then as per the choice selected, the cone perspective is modified to show the conic section on the screen.

The circle is obtained by cutting the cone with a plane parallel to the base.

The ellipse is obtained by cutting the cone with a plane at a 90° angle to the base.

Parabola is obtained by cutting the cone by a plane at angle > 0 (!= 0) with respect to its axis.

Hyperbola is obtained by cutting the cone by a plane parallel to its axis.

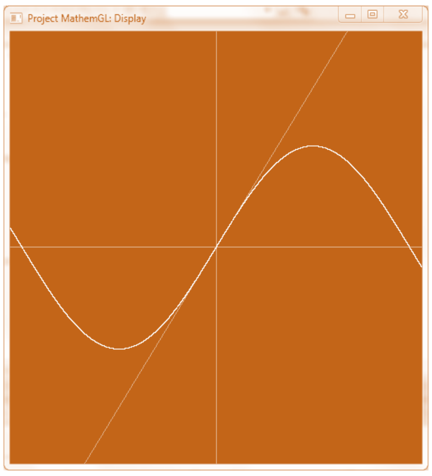
Code:

```
#include <gl/freeglut.h>
#include <iostream>
#include <cmath>

#define PRECISION 0.0001
#define ZOOM 0.01
#define PI 3.14159265359
#define MULT_FACTOR 0.1

#define PARABOLA_A 1.5          // Parabola semi-latus rectum (a)
#define PARABOLA_T 2.5          // Extremes of parabola parameter
#define PARABOLA_DEL 0.0004     // Difference b/w parameter value b/w frames

double scaleFactorX = 1 /(4 * PI), scaleFactorY = 0.25;
int clickPosX, clickPosY;
float t = 0;
float tp = PARABOLA_T; // Parameter for Parabola
float transZ = -6, rotAngle = 0; // Constants for Conic Generation

// maxAngle: Ellipse: -150, Circle: -180, Parabola: -110, Hyperbola: -90
// maxTranslation: Ellipse, Circle: -2.2, Parabola: -2.2, Hyperbola: -2.4
// yTranslation: Ellipse: -0.2, Circle: 0, Parabola: -0.2, Hyperbola: -0.4
float maxAngle = -150, maxTranslationZ = -2.2, yTranslation = -0.2;

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);
        for(int i=0 ; i<100 ; i++){
            for(int j=0 ; j<100 ; j++){
                if (i == j ){
                   glVertex2d(i * scaleFactorX, j * scaleFactorY);
                   glVertex2d(-i * scaleFactorX, -j * scaleFactorY);
                }
            }
        }
    glEnd();

    glBegin(GL_POINTS);
```

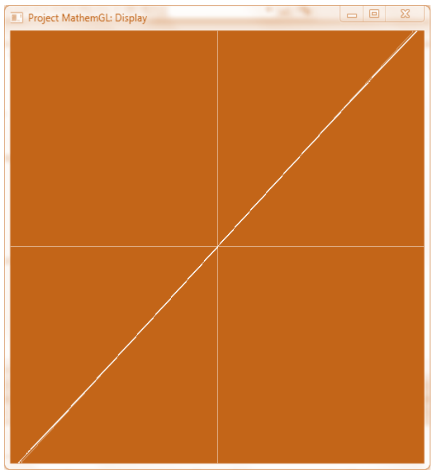
```
        glColor3f(1, 1, 1);
        do {
            y = sin(x);
            x = x + PRECISION;

            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);

        for(int i=0 ; i<100 ; i++){
            for(int j=0 ; j<100 ; j++){
                if (i == j ){
                   glVertex2d(i * scaleFactorX, j * scaleFactorY);
                   glVertex2d(-i * scaleFactorX, -j * scaleFactorY);
                }
            }
        }
    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();
}

void plotCircle(){

    // Circle is being made using this eqn. x^2 + y^2 = 1
    glClearColor(1,1,1,1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    float x,y,px,py,pt;
        glBegin(GL_LINES);


            // X Axis
            glColor3f(0.9, 0.9, 0.9);
            glVertex2d(-1, 0);
            glVertex2d(1, 0);

            // Y Axis
            glVertex2d(0, 1);
            glVertex2d(0, -1);

            //Parametric Equations
            x = 3*sin(t);
            y = 3*cos(t);
            glColor3f(1.0, 0, 0);
            glVertex2f(0 *MULT_FACTOR,0*MULT_FACTOR);
            glVertex2f(x*MULT_FACTOR,y*MULT_FACTOR);
        glEnd();

        glPointSize(4);
        glBegin(GL_POINTS);
            glColor3f(0.0, 0.0, 1.0);
            glVertex2f(0 *MULT_FACTOR,0*MULT_FACTOR);
            glColor3f(0, 0, 0);
            glVertex2f(x*MULT_FACTOR,y*MULT_FACTOR);
        glEnd();

       glPointSize(2);
       glBegin(GL_POINTS);
            pt = 0;

           //Loop to make locus till the point where drawing point has reached
            while(pt<=t){
              px = 3*sin(pt);
              py = 3*cos(pt);
              glColor3f(0.5, 0.5, 0.5);
              glVertex2f(px*MULT_FACTOR,py*MULT_FACTOR);
              pt = pt + PI / 5000;
            }
       glEnd();

     //Animation:Updating the circle parameter if the drawing hasn't finished.
        if(t < 2*PI) {
            t = t + PI / 5000;
            glutPostRedisplay();
        }

    glFlush();
}

void plotEllipse(){
    // Ellipse is being made using this eqn. x^2 / 64 + y^2 / 9 = 1
    glClearColor(1,1,1,1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    float x,y,px,py,pt;
        glBegin(GL_LINES);

            // X Axis
            glColor3f(0.9, 0.9, 0.9);
            glVertex2d(-1, 0);
            glVertex2d(1, 0);

            // Y Axis
            glVertex2d(0, 1);
            glVertex2d(0, -1);

            //parametric Equations
            x = 8*sin(t);
            y = 3*cos(t);

            // Lines joining Focii to Ellipse
            glColor3f(1.0, 0, 0);
            glVertex2f(-6 *MULT_FACTOR,0*MULT_FACTOR);
            glVertex2f(x*MULT_FACTOR,y*MULT_FACTOR);
            glColor3f(1.0, 0.0, 0.0);
            glVertex2f(x*MULT_FACTOR,y*MULT_FACTOR);
            glVertex2f(6*MULT_FACTOR,0*MULT_FACTOR);
        glEnd();


        glPointSize(4);
        glBegin(GL_POINTS);
            glColor3f(0.0, 0.0, 1.0);
            glVertex2f(6 *MULT_FACTOR,0*MULT_FACTOR);
            glVertex2f(-6 *MULT_FACTOR,0*MULT_FACTOR);
            glColor3f(0, 0, 0);
            glVertex2f(x*MULT_FACTOR,y*MULT_FACTOR);
        glEnd();

       glPointSize(2);
       glBegin(GL_POINTS);
            pt = 0;

          // Loop to make locus till the point where drawing point has reached
            while(pt<=t){

              //Parametric Equations
              px = 8*sin(pt);
              py = 3*cos(pt);
              glColor3f(0.5, 0.5, 0.5);
              glVertex2f(px*MULT_FACTOR,py*MULT_FACTOR);
              pt = pt + PI / 5000;

            }
        glEnd();

  // Animation: Updating the ellipse parameter if the drawing hasn't finished.
        if(t < 2*PI) {
            t = t + PI / 5000;
            glutPostRedisplay();
        }

    glFlush();
}

void plotParabola() {
    // Parabola is being made using this eqn. y^2 = 4 * a * x
    float x, y, px, py, pt;

    glClearColor(1, 1, 1, 1);
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
        x = PARABOLA_A * tp * tp;
        y = 2 * PARABOLA_A * tp;

        // X Axis
        glColor3f(0.9, 0.9, 0.9);
        glVertex2d(-1, 0);
        glVertex2d(1, 0);

        // Y Axis
        glVertex2d(0, 1);
        glVertex2d(0, -1);

        // Directrix
        glColor3f(0, 0, 0);
        glVertex2f(-PARABOLA_A * MULT_FACTOR, 1);
        glVertex2f(-PARABOLA_A * MULT_FACTOR, -1);

        // Line joining Directrix to Parabola
        glColor3f(1, 0, 0);
        glVertex2f(-PARABOLA_A * MULT_FACTOR, y * MULT_FACTOR);
        glVertex2f(x * MULT_FACTOR, y * MULT_FACTOR);

        // Line joining Focus to Parabola
        glVertex2f(x * MULT_FACTOR, y * MULT_FACTOR);
        glVertex2f(PARABOLA_A * MULT_FACTOR, 0 * MULT_FACTOR);
    glEnd();
    glPointSize(4);
    glBegin(GL_POINTS);

        // Focus
        glColor3f(0, 0, 1);
        glVertex2f(PARABOLA_A * MULT_FACTOR, 0 * MULT_FACTOR);

        // Point (wrt. y-coordinate) on Directrix
        glVertex2f(-PARABOLA_A * MULT_FACTOR, y * MULT_FACTOR);

        // Point on the Parabola (where it is being drawn)
        glColor3f(0, 0, 0);
        glVertex2f(x * MULT_FACTOR, y * MULT_FACTOR);
    glEnd();

    glPointSize(2);
    glBegin(GL_POINTS);
        pt = PARABOLA_T;
        // Loop to make locus till the point where drawing point has reached
        while(pt >= tp) {
          px = PARABOLA_A * pt * pt;
          py = 2 * PARABOLA_A * pt;

          glColor3f(0.5, 0.5, 0.5);

          glVertex2f(px * MULT_FACTOR, py * MULT_FACTOR);

          pt = pt - PARABOLA_DEL;
        }
    glEnd();

 // Animation: Updating the parabola parameter if the drawing hasn't finished.
    if(tp <= PARABOLA_T && tp >= -PARABOLA_T) {
        tp = tp - PARABOLA_DEL;
        glutPostRedisplay();
    }

    glFlush();
}

void generateConic() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);

    glLoadIdentity();
    glTranslatef(0.0f, yTranslation, transZ);
    glRotatef(rotAngle, 1, 0, 0);
    glColor3f(1, 1, 1);
    glutWireCone(1,2,150,50);
    if( rotAngle > maxAngle ) {
        rotAngle = rotAngle - 0.03;
        glutPostRedisplay();
    }

    if(rotAngle <= maxAngle && transZ <= maxTranslationZ) {
       transZ = transZ + 0.0005;
       glutPostRedisplay();
    }
    glFlush();
}

void reshapeConic(GLsizei width, GLsizei height) {
   if (height == 0) height = 1;

   GLfloat aspect = (GLfloat)width / (GLfloat)height;

   glViewport(0, 0, width, height);

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();

   gluPerspective(45.0f, aspect, 3.0f, 100.0f);
}

int main(int argc, char** argv) {
    int type;
    glutInit(&argc, argv);

    cout<< "Select option : \n 1) Sine \n 2) Cosine \n 3) Tangent \n 4) Circle
 \n 5) Ellipse \n 6) Parabola \n "
        << "7) Generate Ellipse \n 8) Generate Circle \n 9) Generate Parabola
\n 10) Generate Hyperbola \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 ;
        case 4 :
            glutDisplayFunc(plotCircle);
            cout<<endl<<"Circle is the locus of all the points\n at equidistan
t from a given \nfixed point, Center.";
            break ;
        case 5 :
            glutDisplayFunc(plotEllipse);
            cout<<endl<<"Ellipse is the locus of all the points\n such that su
m of distances of the point from \ntwo given fixed points, Focii is constant."
;
            break ;
        case 6 :
            glutDisplayFunc(plotParabola);
            cout<<endl<<"Parabola is the locus of all the points\n at equidist
ant from a given \nfixed point, Focus and a line, Directrix.";
            break ;
        case 7 :
            // Ellipse
            maxAngle = -150, maxTranslationZ = -2.2, yTranslation = -0.2;
            glutDisplayFunc(generateConic);
            glutReshapeFunc(reshapeConic);
            cout<<endl<<"Ellipse is a conic section \nobtained by cutting the
cone by a plane at angle < 90 wrt base.";
            break;
        case 8 :
            // Circle
            maxAngle = -180, maxTranslationZ = -2.2, yTranslation = 0;
            glutDisplayFunc(generateConic);
            glutReshapeFunc(reshapeConic);
            cout<<endl<<"Circle is a conic section \nobtained by cutting the c
one by a plane parallel to the base.";
            break;
        case 9 :
            // Parabola
            maxAngle = -110, maxTranslationZ = -2.2, yTranslation = -0.2;
            glutDisplayFunc(generateConic);
            glutReshapeFunc(reshapeConic);
            cout<<endl<<"Parabola is a conic section \nobtained by cutting the
 cone by a plane at angle > 0 (!= 0) wrt its axis.";
            break;
        case 10 :
            // Hyperbola
            maxAngle = -90, maxTranslationZ = -2.4, yTranslation = -0.4;
            glutDisplayFunc(generateConic);
            glutReshapeFunc(reshapeConic);
            cout<<endl<<"Hyperbola is a conic section \nobtained by cutting th
e cone by a plane parallel to its axis.";
            break;
        default:
            cout<<"Enter a value b/w 1 and 10"<<endl<<endl;
            break;
    }
    glutMainLoop();
    return 0;
}
```

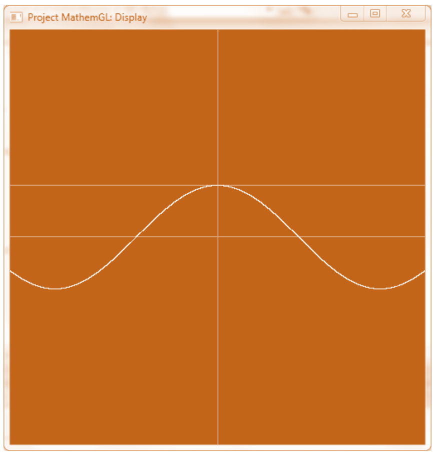
CHAPTER 5: RESULTS

Output of plotted graphs:

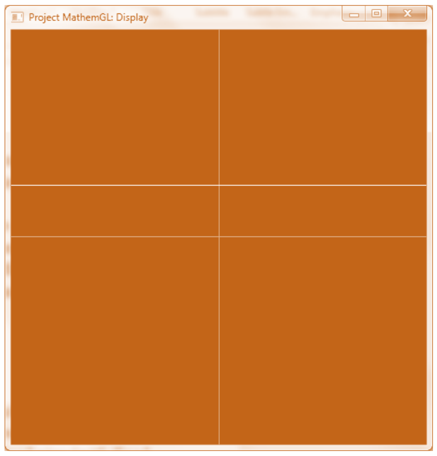


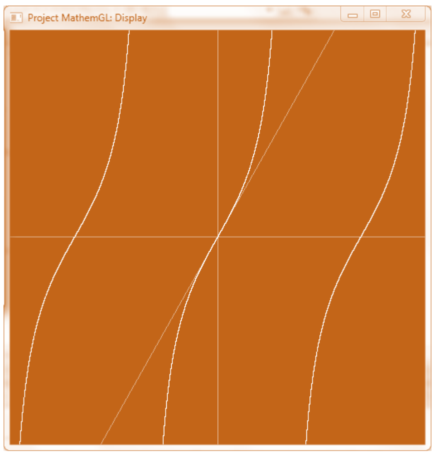
$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$



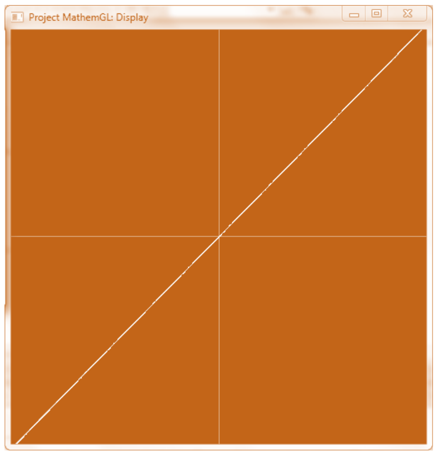


$$\lim_{x \to 0} \cos x = 1$$



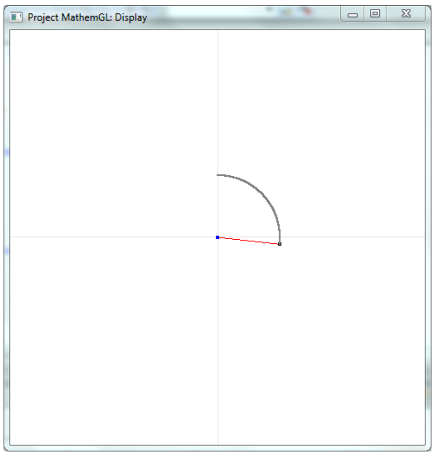


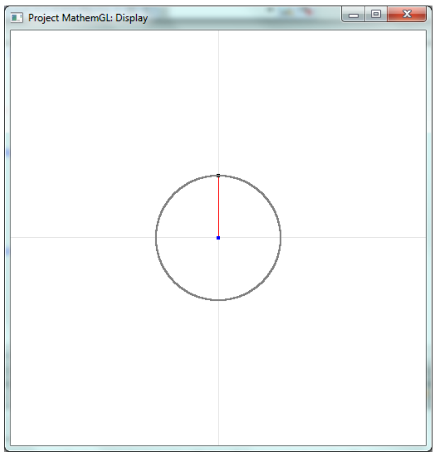
$$\lim_{x \to 0} \frac{\tan x}{x} = 1$$



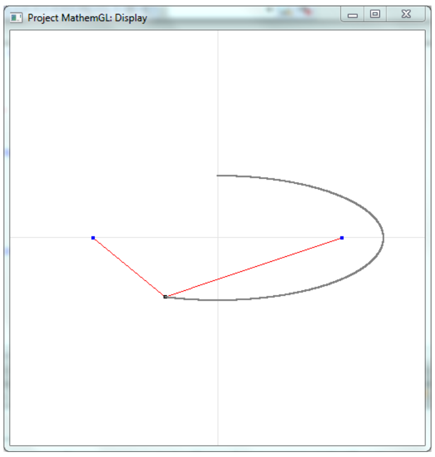
Locus of Conic Sections:

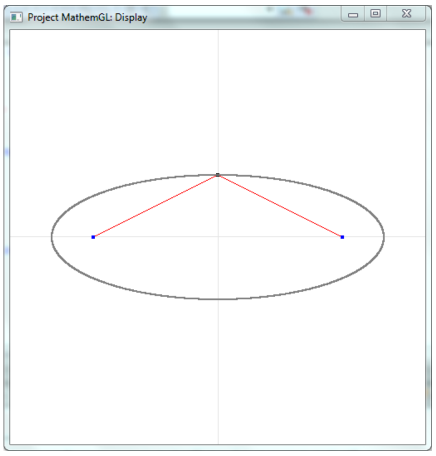
Circle: Circle is the locus of all the points at equidistant from a given fixed point, Center.



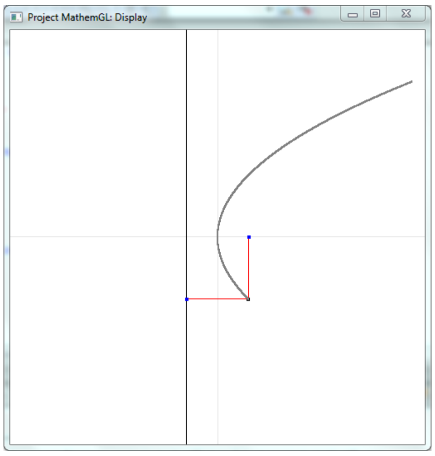


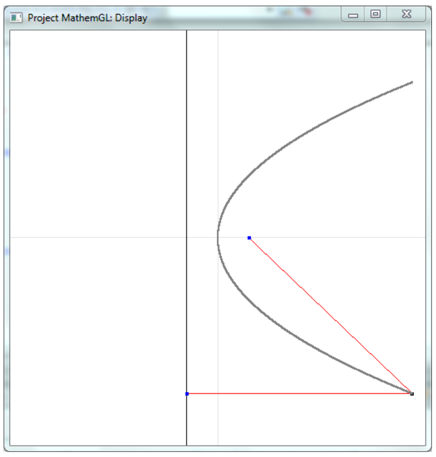
Ellipse: Ellipse is the locus of all the points such that sum of distances of the point from two given fixed points, Foci is constant.



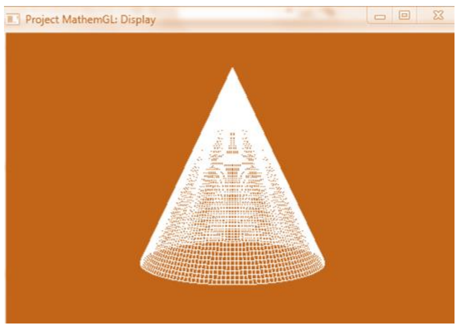


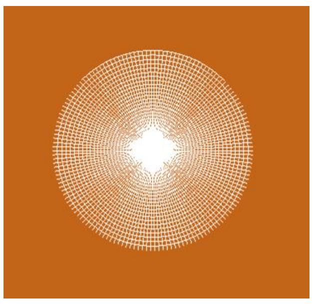
Parabola: Parabola is the locus of all the points at equidistant from a given fixed point, Focus and a line, Directrix.



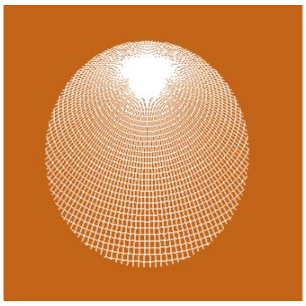


Formation of Conic Sections from Cone:

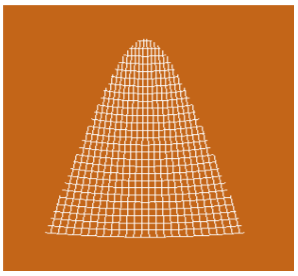




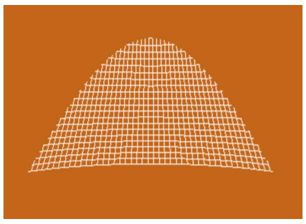
Circle: Circle is a conic section obtained by cutting the cone by a plane parallel to the base.



Ellipse: Ellipse is a conic section obtained by cutting the cone by a plane at angle < 90 with respect to base.



Parabola: Parabola is a conic section obtained by cutting the cone by a plane at angle > 0 (not equal to 0) with respect to its axis.



Hyperbola: Hyperbola is a conic section obtained by cutting the cone by a plane parallel to its axis.

CONCLUSION

We have shown graphical proofs of standard limits using plot of trigonometric functions. We have also shown how different conic sections can be drawn using pencil, thread and fixed pins (with fixed lines in case of parabola). We have visualized how conic sections are formed from cross-section of a cone.

FUTURE WORK

We intend to make many iterations of the application in the future and these are some of the concepts we have come up with:

- Visualization of Surface Area and Mensuration for grade school.
- Analysis of Calculus using graphs.
- Expansion to other subjects to allow 3D animations e.g. Science, Geography etc.

BIBLIOGRAPHY

1. Interactive Computer Graphics (a top down approach) by Edward Angel

2. OpenGL Programming Guide by Addison-Wesley

Online Resources:

https://en.wikipedia.org/wiki/Main_Page

https://shankarrajagopal.github.io/

https://opengl.org

https://cs.unm.edu/~angel