|  |  |
| --- | --- |
| **Logo  Description automatically generated** | **BÀI THỰC HÀNH ĐỒ HỌA MÁY TÍNH** |
|  | **CÁC ĐỐI TƯỢNG ĐỒ HỌA CƠ BẢN** |

**Họ và tên Sinh viên:**Phan Trần Nhật Hạ  

**Mã Sinh viên:**102210159     **Nhóm:** 21Nh12

[**1.**    **Cấu hình Project sử dụng thư viện OpenGL trong Dev C**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273256)

[**2.**    **Lý thuyết - Thư viện đồ họa OpenGL**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273257)

[2.1        Các đối tượng đồ họa cơ sở (primitive) của OpenGL](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273258)

[2.1.1         Điểm : GL\_POINTS](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273259)

[2.1.2         Đường thẳng : GL\_LINES](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273260)

[2.1.3         Tam giác : GL\_TRIANGLES](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273261)

[2.1.4         Tứ giác : GL\_QUADS](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273262)

[2.1.5         Đa giác : GL\_POLYGON](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273263)

[2.2        Xóa bộ đệm màu glClear()](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273264)

[2.3        Thiết lập lại màu vẽ cho đối tượng glColor3f()](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273265)

[2.4        Các hàm vẽ các đối tượng hình học phức tạp trong thư viện GLUT](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273266)

[**3.**    **Chương trình lab01hcn.cpp: Vẽ một hình chữ nhật màu trắng trên nền đen**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273267)

[**4.**    **Chương trình lab01tamgiac.cpp: Vẽ một tam giác**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273268)

[**5.**    **Chương trình lab01tudien.cpp: Vẽ một khối tứ diện**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273269)

[**6.**    **Chương trình lab01mouse.cpp: Xử lý các sự kiện phím và chuột trong OpenGL**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273270)

[**7.**    **Chương trình lab01hinhcau.cpp: Vẽ mặt cầu/bình trà**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273271)

[**8.**    **Bài tập**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-LapTrinhOpenGLCacDoiTuongDoHoaCoBan.htm#_Toc129273272)

**1.    Cấu hình Project sử dụng thư viện OpenGL trong Dev C**

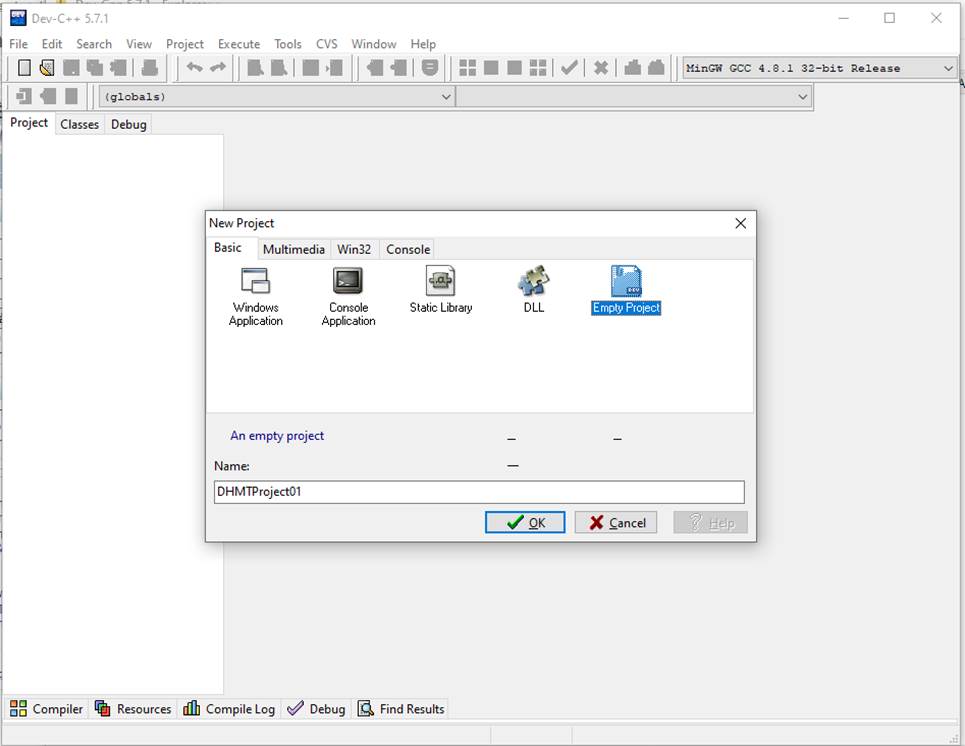
§  Tải file **Dev-Cpp 5.7.1.rar** và giải nén vào thư mục **C:\ Dev-Cpp 5.7.1**

§  Vào thư mục Dev-Cpp 5.7.1\libOpenGL: Sao chép các file thư viện **glut32.dll, glut.dll, glut32.dll**  vào hai thư mục sau:

**C:\Windows\SysWOW64\\*.\***

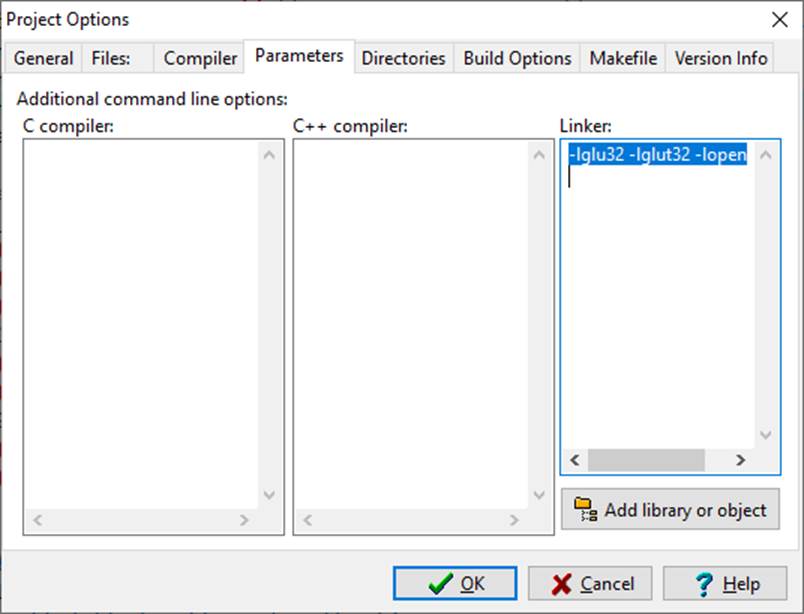
**C:\Windows\System32\\*.\***

§  Tạo mới một Project: Menu *File\New\Project* => Chọn *Empty Project*. Nhập tên. OK



§  Trong cửa sổ *Project Options* (hoặc nhấn Alt + P): Chọn *tab Parameter => Linker* khai báo như sau:

**-lglu32 -lglut32 -lopengl32**



          Chọn OK.

          Trong Project, tạo một tập tin chương trình và lưu Lab\*.cpp

**2.    Lý thuyết - Thư viện đồ họa OpenGL**

OpenGL cung cấp một số thành phần đồ hoạ cơ sở gọi là các primitive. Cần xác định các primitive trước khi vẽ trên màn hình.

**2.1       Các đối tượng đồ họa cơ sở (primitive) của OpenGL**

*2.1.1     Điểm : GL\_POINTS*

glBegin(GL\_POINTS);

       glVertex3f(1.0f, 0.0f, 0.0f); // A

       glVertex3f(0.0f, 1.0f, 0.0f); // B

       glVertex3f(0.0f, 0.0f, 1.0f); // C

glEnd();

*2.1.2     Đường thẳng : GL\_LINES*

glBegin(GL\_LINES);

       // AB

       glVertex3f(1.0f, 0.0f, 0.0f);

       glVertex3f(0.0f, 1.0f, 0.0f);

       // AC

       glVertex3f(1.0f, 0.0f, 0.0f);

       glVertex3f(0.0f, 0.0f, 1.0f);

glEnd();

Hai primitive được xây dựng dựa trên GL\_LINES :

§  GL\_LINE\_STRIP : các cạnh được vẽ liên tiếp theo thứ tự các điểm đưa ra bởi glVertex.

glBegin(GL\_LINES);

       glVertex3f(1.0f, 0.0f, 0.0f); // A

       glVertex3f(0.0f, 1.0f, 0.0f); // B

       glVertex3f(0.0f, 0.0f, 1.0f); // C

       glVertex3f(0.0f, -1.0f, 0.0f); // D

glEnd();

§  GL\_LINE\_LOOP : tương tự GL\_LINE\_STRIP nhưng điểm kết thúc được nối với điểm khởi đầu.

§  Các primitive GL\_LINE … cho phép vẽ mô hình khung của nhiều hình phức tạp. Muốn vẽ được các đối tượng với các mặt được tô màu, chúng ta phải sử dụng các primitive đa giác.

§  Hai mặt của đa giác trong OpenGL có thuộc tính chiều xoay của các đỉnh *(winding)* được xác định theo thứ tự các đỉnh liệt kê khi vẽ đa giác. Mặc định, mặt trước của đa giác tương ứng với chiều xoay ngược kim đồng hồ. Để thiết lập mặt trước / sau của đa giác theo chiều xoay, sử dụng hàm *glFrontFace* :

              glFrontFace (GL\_CW)           // Mặt trước theo chiều kim đồng hồ

              glFrontFace (GL\_CCW)  // Mặt trước ngược chiều kim đồng hồ

*2.1.3     Tam giác : GL\_TRIANGLES*

glBegin(GL\_TRIANGLES);

     // Tam giác V0V1V2

     glVertex2f(0.0f,0.0f);  // V0

     glVertex2f(1.0f,2.0f);  // V1

     glVertex2f(2.0f,0.0f);  // V2

     // Tam giác V3V4V5

     glVertex2f(-3.0f,0.0f);  // V3

     glVertex2f(-1.0f,0.0f);  // V4

     glVertex2f(-3.0f,2.0f);  // V5

glEnd();

          Hai primitive xây dựng dựa trên GL\_TRIANGLES :

§  GL\_TRIANGLE\_STRIP : vẽ liên tiếp các tam giác. Tam giác thứ *i+1* được tạo thành từ một đỉnh mới và hai đỉnh cũ thuộc tam giác thứ *i*.

     //  Tam giác V0V1V2

     glVertex2f ….    // V­0

     glVertex2f ….       // V1

     glVertex2f ….       // V2

          //  Tam giác V1V2V3

     glVertex2f ….       // V3

     //  Tam giác V2V3V4

     glVertex2f ….       // V4

glEnd();

Lưu ý : Hướng của các cạnh không nhất thiết bảo toàn trong các tam giác khác nhau.

§  GL\_TRIANGLE\_FAN : Vẽ liên tiếp các tam giác xoay quanh đỉnh xác định đầu tiên.

glBegin(GL\_TRIANGLE\_FAN);

     //  Đỉnh tâm

     glVertex2f …..   // V0

     //  Tam giác V0V1V2

     glVertex2f …..      // V1

     glVertex2f …..      // V2

     //  Tam giác V0V2V3

     glVertex2f …..      // V3

     //  Tam giác V0V3V4

     glVertex2f …..      // V4

glEnd();

*2.1.4     Tứ giác : GL\_QUADS*

Primitive này được xác định dựa trên 4 đỉnh liên tục được liệt kê, và không có đỉnh nào được phép sử dụng lại cho các tứ giác tiếp theo. Tương tự tam giác, GL\_QUAD\_STRIP cho phép vẽ nhiều tứ giác liên tiếp nhau, trong đó tứ giác thứ ­*i+1* được xác định từ 2 đỉnh mới và 2 đỉnh cũ của tứ giác thứ *i*.

*2.1.5     Đa giác : GL\_POLYGON*

glBegin(GL\_POLYGON);

     // Tam giác V0V1V2

     glVertex2f(0.0f,0.0f);  // V0

     glVertex2f(1.0f,2.0f);  // V1

     glVertex2f(2.0f,0.0f);  // V2

glEnd();

**2.2       Xóa bộ đệm màu glClear()**

Mỗi lần vẽ, nên dùng lệnh xóa bộ đệm màu glClear()

glClearColor(0.0, 0.0, 0.0, 0.0); /\* xác định màu để xóa color buffer (màu đen) \*/

glClearDepth(1.0); /\* xác định giá trị để xóa depth buffer \*/

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  /\* xóa color buffer và depth buffer \*/

**2.3       Thiết lập lại màu vẽ cho đối tượng glColor3f()**

Khi vẽ một đối tượng, OpenGL sẽ tự động sử dụng màu đã được xác định trước đó. Do đó, để vẽ đối tượng với màu mới, cần phải thiết lập lại màu vẽ bằng cách dùng hàm glColor3f()

glColor3f(0.0, 0.0, 0.0);           // black

glColor3f(1.0, 0.0, 0.0);           // red

glColor3f(0.0, 1.0, 0.0);           // green

glColor3f(1.0, 1.0, 0.0);           // yellow

glColor3f(0.0, 0.0, 1.0);           // blue

glColor3f(1.0, 0.0, 1.0);           // magenta

glColor3f(0.0, 1.0, 1.0);           // cyan

glColor3f(1.0, 1.0, 1.0);           // white

**2.4       Các hàm vẽ các đối tượng hình học phức tạp trong thư viện GLUT**

§  void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);

§  void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);

§  void glutWireCube(GLdouble size);

§  void glutSolidCube(GLdouble size);

§  void glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);

§  void glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);

§  void glutWireIcosahedron(void);

§  void glutSolidIcosahedron(void);

§  void glutWireOctahedron(void);

§  void glutSolidOctahedron(void);

§  void glutWireTetrahedron(void);

§  void glutSolidTetrahedron(void);

§  void glutWireDodecahedron(GLdouble radius);

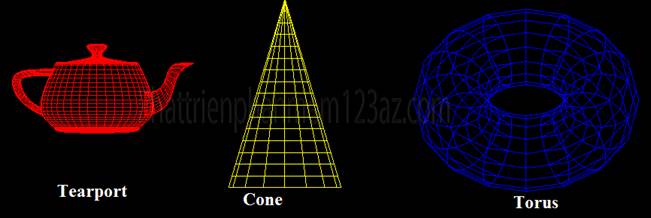
§  void glutSolidDodecahedron(GLdouble radius);

§  void glutWireCone(GLdouble radius, GLdouble height, GLint slices, GLint stacks);

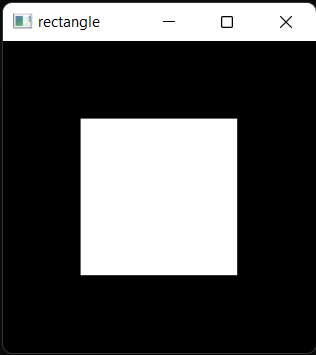
§  void glutSolidCone(GLdouble radius, GLdouble height, GLint slices, GLint stacks);

§  void glutWireTeapot(GLdouble size);

§  void glutSolidTeapot(GLdouble size);



**3.    Chương trình lab01hcn.cpp: Vẽ một hình chữ nhật màu trắng trên nền đen**



**4.    Chương trình lab01tamgiac.cpp: Vẽ một tam giác**

#include "GL/glut.h"

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 1.0, 0.5);

glBegin(GL\_TRIANGLES);

glVertex3f (0.25, 0.25, 0.0);

glVertex3f (0.75, 0.25, 0.0);

glVertex3f (0.75, 0.75, 0.0);

glEnd();

glFlush ();

}

void init (void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (250, 250);

glutInitWindowPosition (100, 100);

glutCreateWindow ("Triangles");

init ();

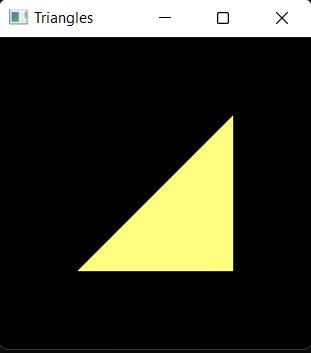
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

//main



**5.    Chương trình lab01tudien.cpp: Vẽ một khối tứ diện**

#include <GL/glut.h>

#include <math.h>

#include "stdio.h"

#define M\_PI 3.14159265358979323846

static GLfloat gocX = 0.0f;

static GLfloat gocY = 0.0f;

static GLfloat gocZ = 0.0f;

static GLfloat size = 0.5;

int HinhVe = 1;

void veTuDien(float r)

{

float x, y, z, Delta\_U, Delta\_V, u, v, Pi\_2;

Pi\_2 = M\_PI / 2;

Delta\_U = 0.1;

Delta\_V = 0.1;

glColor3f(1.0, 0.2, 0.5);

glBegin(GL\_POLYGON);

glVertex3f(r, 0.0f, 0.0f);

glVertex3f(0.0f, r, 0.0f);

glVertex3f(-r, 0.0f, 0.0f);

glEnd();

glFlush ();

glColor3f(0.3, 1.0, 0.5);

glBegin(GL\_POLYGON);

glVertex3f(-r, 0.0f, 0.0f);

glVertex3f(0.0f, r, 0.0f);

glVertex3f(0.0f, 0.0f, -r);

glEnd();

glFlush ();

glColor3f(1.0, 0.0, 0.5);

glBegin(GL\_POLYGON);

glVertex3f(0.0f, 0.0f, -r);

glVertex3f(0.0f, r, 0.0f);

glVertex3f(r, 0.0f, 0.0f);

glEnd();

glFlush ();

glColor3f(0.5, 0.0, 1.8);

glBegin(GL\_POLYGON);

glVertex3f(-r, 0.0f, 0.0f);

glVertex3f(0.0f, 0.0f, -r);

glVertex3f(r, 0.0f, 0.0f);

glEnd();

glFlush ();

}

void myDisplay(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glPushMatrix();

glRotatef(gocX, 1.0f, 0.0f, 0.0f);

glRotatef(gocY, 0.0f, 1.0f, 0.0f);

glRotatef(gocZ, 0.0f, 0.0f, 1.0f);

veTuDien(size);

glPopMatrix();

glutSwapBuffers();

}

void myKeyboard(int key, int x, int y)

{

if (key == GLUT\_KEY\_END) exit(0);

if (key == GLUT\_KEY\_DOWN) gocX -= 5.0f;

if (key == GLUT\_KEY\_UP) gocX += 5.0f;

if (key == GLUT\_KEY\_RIGHT) gocY -= 5.0f;

if (key == GLUT\_KEY\_LEFT) gocY += 5.0f;

if (key == GLUT\_KEY\_PAGE\_DOWN) gocZ -= 5.0f;

if (key == GLUT\_KEY\_PAGE\_UP) gocZ += 5.0f;

glutPostRedisplay();

}

void myColor(unsigned char k, int x, int y)

{

printf("k= %d", k);

switch (k) {

case 114: glColor3f(1, 0, 0);

glutPostRedisplay();

break;

case 103: glColor3f(0, 1, 0);

glutPostRedisplay();

break;

case 98: glColor3f(0, 0, 1);

glutPostRedisplay();

break;

case 49: HinhVe = 1;

glutPostRedisplay();

break;

case 50: HinhVe = 2;

glutPostRedisplay();

break;

default: break;

}

}

void myMouse(int button, int state, int x, int y)

{

if ((button == GLUT\_LEFT\_BUTTON) & (state == GLUT\_DOWN)) size = size + 0.05;

if ((button == GLUT\_RIGHT\_BUTTON) & (state == GLUT\_DOWN)) size = size - 0.05;

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(600, 600);

glutInitWindowPosition(50, 50);

glutCreateWindow("Khoi tu dien");

glClearColor(0.0, 0.0, 0.0, 0.0);

glColor3f(0, 0, 0);

glutDisplayFunc(myDisplay);

glutSpecialFunc(myKeyboard);

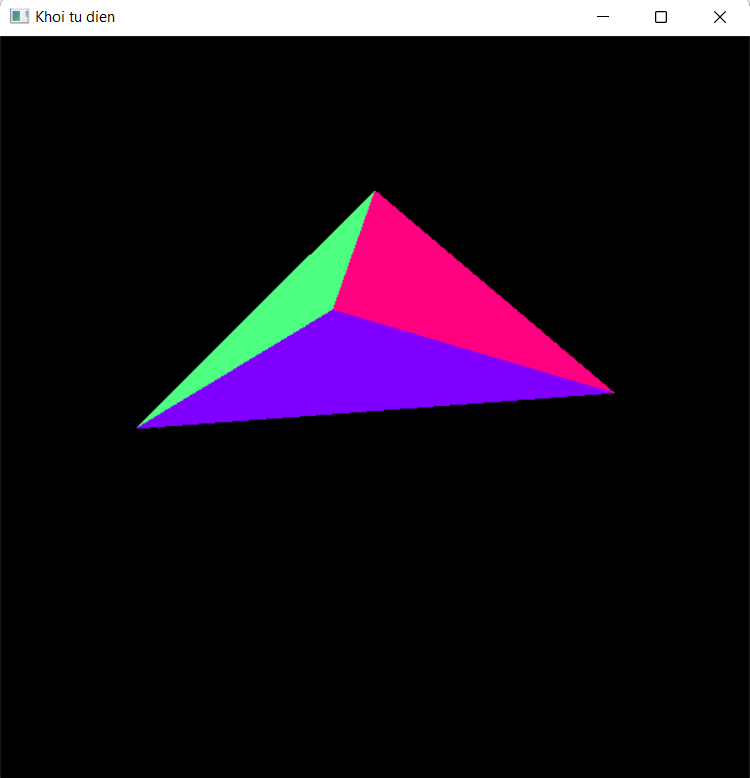
glutKeyboardFunc(myColor);

glutMouseFunc(myMouse);

glutMainLoop();

return 0;

}



**6.    Chương trình lab01mouse.cpp: Xử lý các sự kiện phím và chuột trong OpenGL**

#include<gl/glut.h>

#include<gl/gl.h>

#include<stdio.h>

void myDisplay(void)

{

       glClear(GL\_COLOR\_BUFFER\_BIT);

       glutSwapBuffers();

       glFlush();

}

void myMouse(int b, int s, int x, int y)

{

       switch (b)

       {

              case GLUT\_LEFT\_BUTTON:

                      if (s==GLUT\_DOWN) printf("\nLeft button pressed!");

                      else if (s==GLUT\_UP)printf("\nLeft button released!");

                      break;

              case GLUT\_RIGHT\_BUTTON:

                      if (s==GLUT\_DOWN)printf("\nRight button pressed!");

                      else if (s==GLUT\_UP) printf("\nRight button released");

                      break;

              default: break;

       }

}

void myKeyboard(unsigned char c, int x, int y)

{

       switch (c)

       {

              case 27:

                      exit(0);

                      break;

              default:

                      printf("\nKey %c is hit",c);

                      break;

       }

}

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

{

       glutInit(&argc, argv);

       glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGBA);

       glutInitWindowPosition(300,200);

       glutInitWindowSize(320,320);

       glutCreateWindow("Vi du 1.2");

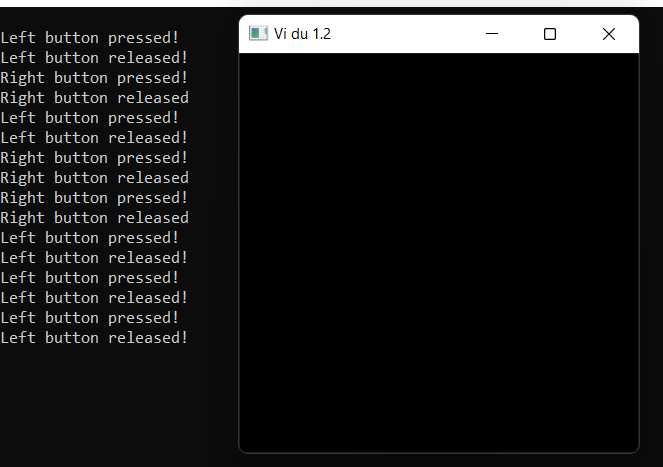
       glutDisplayFunc(myDisplay);

       glutMouseFunc(myMouse);

       glutKeyboardFunc(myKeyboard);

       glutMainLoop();

}



**7.    Chương trình lab01hinhcau.cpp: Vẽ mặt cầu/bình trà**

#include <GL/glut.h>

#include <math.h>

#include "stdio.h"

#define M\_PI    3.14159265358979323846

static GLfloat gocX = 0.0f;

static GLfloat gocY = 0.0f;

static GLfloat gocZ = 0.0f;

static GLfloat size = 0.5;

int HinhVe=1;

//-------cai dat cac hàm ve mat cong o day------------

void veBinhTra(GLfloat s)

{

       glutWireTeapot(s);

}

void veMatCong(float r) //ve mat cau

{

       float x,y,z,Delta\_U,Delta\_V,u,v,Pi\_2;

       Pi\_2 = M\_PI/2;

Delta\_U = 0.1;

Delta\_V=0.1;

       glBegin(GL\_POINTS);

       for(v = -Pi\_2;v<Pi\_2;v+=Delta\_V)

       {

              for(u=0;u<2\*M\_PI;u+=Delta\_U)

              {

                      x= r\*cos(u)\*cos(v);

                      y= r\*sin(u)\*cos(v);

                      z= r\*sin(v);

                      glVertex3f(x,y,z);

              }

       }

       glEnd();

}

//----------------------------------------------------

void myDisplay(void)

{

       glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

       glPushMatrix();

       glRotatef(gocX, 1.0f, 0.0f, 0.0f);

       glRotatef(gocY, 0.0f, 1.0f, 0.0f);

       glRotatef(gocZ, 0.0f, 0.0f, 1.0f);

       //Goi cac ham ve mat tron xoay o day

       switch (HinhVe) {

              case 1 :       veBinhTra(size); break;

              case 2 :       veMatCong (size); break;

              default:       break;

       }

       glPopMatrix();

       glutSwapBuffers();

}

void myKeyboard(int key, int x, int y)

{

       if(key == GLUT\_KEY\_END)              exit(0);

       if(key == GLUT\_KEY\_DOWN)             gocX-= 5.0f;

       if(key == GLUT\_KEY\_UP)               gocX += 5.0f;

       if(key == GLUT\_KEY\_RIGHT)            gocY -= 5.0f;

       if(key == GLUT\_KEY\_LEFT)             gocY += 5.0f;

       if(key == GLUT\_KEY\_PAGE\_DOWN)        gocZ -= 5.0f;

       if(key == GLUT\_KEY\_PAGE\_UP)          gocZ += 5.0f;

       glutPostRedisplay();

}

void myColor(unsigned char k, int x, int y)

{

       printf("k= %d",k);

       switch (k) {

              case 114 :     glColor3f(1, 0, 0);     //r

                             glutPostRedisplay();

                             break;

              case  103 : glColor3f(0, 1, 0);   //g

                             glutPostRedisplay();

                             break;

              case  98 :     glColor3f(0, 0, 1);   //b

                             glutPostRedisplay();

                             break;

              case 49 :      HinhVe=1;

                             glutPostRedisplay();

                             break;

              case 50 :      HinhVe=2;

                             glutPostRedisplay();

                             break;

              default:       break;

       }

 }

void myMouse(int button, int state, int x, int y)

{

       if ((button==GLUT\_LEFT\_BUTTON)  & (state==GLUT\_DOWN))       size = size + 0.05;

       if ((button==GLUT\_RIGHT\_BUTTON) & (state==GLUT\_DOWN))       size = size - 0.05;

       glutPostRedisplay();

}

//----------------------------------------------------

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

{

       glutInit(&argc, argv);

       glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

       glutInitWindowSize (600, 600);

       glutInitWindowPosition (50, 50);

       glutCreateWindow("Quan Sat Vat The 3D");

       glClearColor(0.0, 0.0, 0.0, 0.0 );

       glColor3f(0, 0, 1);

       glutDisplayFunc(myDisplay);

       glutSpecialFunc(myKeyboard);

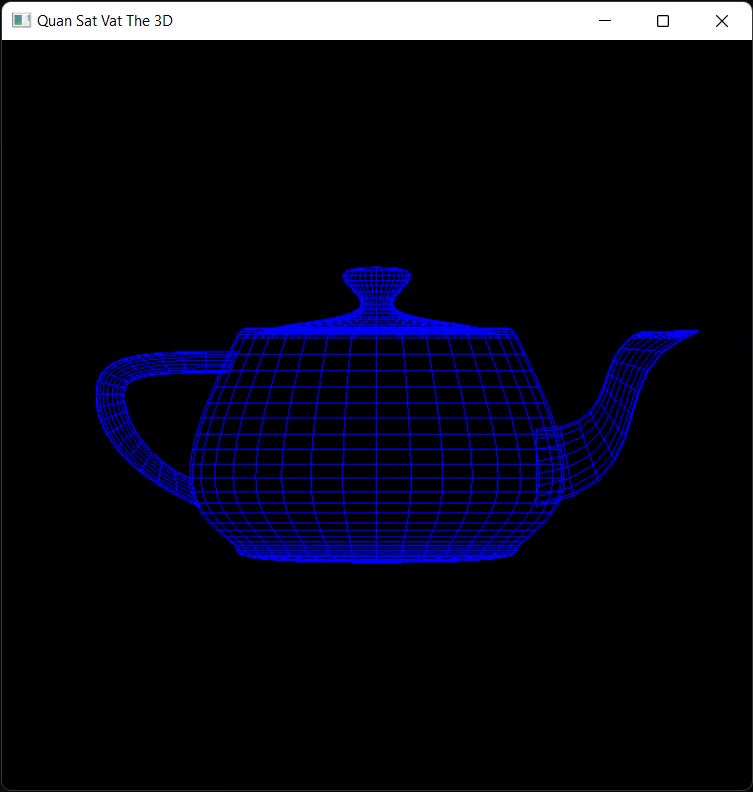
       glutKeyboardFunc(myColor);

       glutMouseFunc(myMouse);

       glutMainLoop();

       return 0;

}



**8.    Bài tập**

1.    Lập trình OpenGL vẽ một đối tượng đơn giản (ngôi nhà, hình hộp chữ nhật, …)

#include "GL/glut.h"

#include "GL/glut.h"

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 1.0, 0.7);

glBegin(GL\_POLYGON);

glVertex3f (0.2, 0.0, 0.0);

glVertex3f (0.6, 0.0, 0.0);

glVertex3f (0.6, 0.6, 0.0);

glVertex3f (0.2, 0.6, 0.0);

glEnd();

glColor3f (1.0, 0.3, 0.7);

glBegin(GL\_TRIANGLES);

glVertex3f(0.2, 0.6, 0.0);

glVertex3f(0.6, 0.6, 0.0);

glVertex3f(0.4, 0.8, 0.0);

glEnd();

glColor3f (0.5, 0.7, 1.0);

glBegin(GL\_POLYGON);

glVertex3f (0.35, 0.0, 0.0);

glVertex3f (0.45, 0.0, 0.0);

glVertex3f (0.45, 0.2, 0.0);

glVertex3f (0.35, 0.2, 0.0);

glEnd();

glFlush ();

}

void init (void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (250, 250);

glutInitWindowPosition (100, 100);

glutCreateWindow ("House");

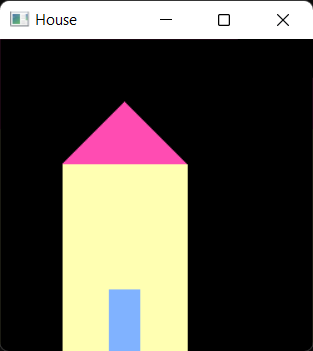
init ();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}



2.    Lập trình OpenGL 2D vẽ một đường tròn ngoại tiếp tam giác ABC có các điểm A(xa, ya), B(xb, yb), C(xc, yc) cho trước không thẳng hàng.

#include "GL/glut.h"

#include <stdio.h>

#include <math.h>

const float PI = 3.14159265358979323846;

void drawTriangle(double\* vertex1, double\* vertex2, double\* vertex3)

{

double c120 = vertex1[0] - vertex2[0];

double c121 = vertex1[1] - vertex2[1];

double c122 = vertex1[2] - vertex2[2];

double c130 = vertex1[0] - vertex3[0];

double c131 = vertex1[1] - vertex3[1];

double c132 = vertex1[2] - vertex3[2];

if (c120 / c130 != c121 / c131 || c120 / c130 != c122 / c132 || c121 / c131 != c122 / c132)

{

glColor3f(1, 0.5, 0.7);

glBegin(GL\_LINE\_LOOP);

glVertex3f(vertex1[0], vertex1[1], vertex1[2]);

glVertex3f(vertex2[0], vertex2[1], vertex2[2]);

glVertex3f(vertex3[0], vertex3[1], vertex3[2]);

glEnd();

}

else

{

printf("Khong phai tam giac!\n");

}

}

void drawCircle(double\* center, double radius)

{

double delta\_v = 0.01;

glColor3f(0, 1, 1.5);

glBegin(GL\_POINTS);

for(double v = 0.0; v < 2.0\*PI; v += delta\_v)

{

double x = center[0] + radius\*sin(v);

double y = center[1] + radius\*cos(v);

glVertex3f(x, y, 0.0);

}

glEnd();

}

void draw(void)

{

double tvertex1[3] = {0.5, 0.5, 0.0};

double tvertex2[3] = {0.5, 0.9, 0.0};

double tvertex3[3] = {0.1, 0.6, 0.0};

drawTriangle(tvertex1, tvertex2, tvertex3);

double a1 = 2.0\*(tvertex1[0] - tvertex2[0]);

double b1 = 2.0\*(tvertex1[1] - tvertex2[1]);

double c1 = tvertex1[0]\*tvertex1[0] + tvertex1[1]\*tvertex1[1] - tvertex2[0]\*tvertex2[0] - tvertex2[1]\*tvertex2[1];

double a2 = 2.0\*(tvertex1[0] - tvertex3[0]);

double b2 = 2.0\*(tvertex1[1] - tvertex3[1]);

double c2 = tvertex1[0]\*tvertex1[0] + tvertex1[1]\*tvertex1[1] - tvertex3[0]\*tvertex3[0] - tvertex3[1]\*tvertex3[1];

double center[2] = {(b2\*c1 - b1\*c2)/(b2\*a1 - b1\*a2), (a2\*c1 - a1\*c2)/(a2\*b1 - a1\*b2)};

double radius = sqrt((center[0] - tvertex1[0])\*(center[0] - tvertex1[0]) + (center[1] - tvertex1[1])\*(center[1] - tvertex1[1]));

drawCircle(center, radius) ;

glFlush();

}

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Duong tron ngoai tiep");

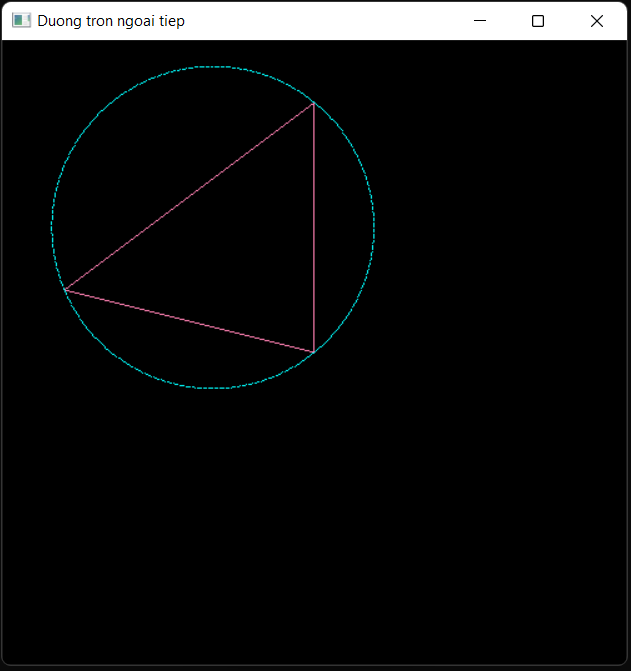
init();

glutDisplayFunc(draw);

glutMainLoop();

return 0;

}



3.    Lập trình OpenGL 2D vẽ một đường tròn 2D nội tiếp tam giác ABC có các điểm A(xa, ya), B(xb, yb), C(xc, yc) cho trước không thẳng hàng.

#include "GL/glut.h"

#include <stdio.h>

#include <math.h>

const float PI = 3.14159265358979323846;

double length(double\* vertex1, double\* vertex2)

{

return sqrt((vertex1[0] - vertex2[0]) \* (vertex1[0] - vertex2[0]) + (vertex1[1] - vertex2[1]) \* (vertex1[1] - vertex2[1]));

}

double distance(double\* point, double\* vertex1, double\* vertex2)

{

double a = (vertex2[1] - vertex1[1])/(vertex2[0] - vertex1[0]);

double b = -1;

double c = vertex1[1] - vertex1[0]\*a;

return (fabs(a\*point[0] + b\*point[1] + c)) / (sqrt(a\*a + b\*b));

}

void drawTriangle(double\* vertex1, double\* vertex2, double\* vertex3)

{

double c120 = vertex1[0] - vertex2[0];

double c121 = vertex1[1] - vertex2[1];

double c122 = vertex1[2] - vertex2[2];

double c130 = vertex1[0] - vertex3[0];

double c131 = vertex1[1] - vertex3[1];

double c132 = vertex1[2] - vertex3[2];

if (c120 / c130 != c121 / c131 || c120 / c130 != c122 / c132 || c121 / c131 != c122 / c132)

{

glColor3f(0, 1, 0.6);

glBegin(GL\_LINE\_LOOP);

glVertex3f(vertex1[0], vertex1[1], vertex1[2]);

glVertex3f(vertex2[0], vertex2[1], vertex2[2]);

glVertex3f(vertex3[0], vertex3[1], vertex3[2]);

glEnd();

}

else {

printf("Khong phai tam giac!\n");

}

}

void drawCircle(double\* center, double radius)

{

double delta\_v = 0.01;

glColor3f(1, 1, 0.6);

glBegin(GL\_POINTS);

for(double v = 0.0; v < 2.0\*PI; v += delta\_v)

{

double x = center[0] + radius\*sin(v);

double y = center[1] + radius\*cos(v);

glVertex3f(x, y, 0.0);

}

glEnd();

}

void draw(void)

{

double tvertex1[3] = {0.5, 0.5, 0.0};

double tvertex2[3] = {0.8, 0.8, 0.0};

double tvertex3[3] = {0.35, 0.8, 0.0};

drawTriangle(tvertex1, tvertex2, tvertex3);

double c12 = length(tvertex1, tvertex2);

double c13 = length(tvertex1, tvertex3);

double c23 = length(tvertex2, tvertex3);

double c123 = c12 +c13 + c23;

double center[2] = {(c12\*tvertex3[0] + c23\*tvertex1[0] + c13\*tvertex2[0])/c123, (c12\*tvertex3[1] + c23\*tvertex1[1] + c13\*tvertex2[1])/c123};

double radius = distance(center, tvertex1, tvertex2);

drawCircle(center, radius);

glFlush();

}

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Duong tron noi tiep tam giac");

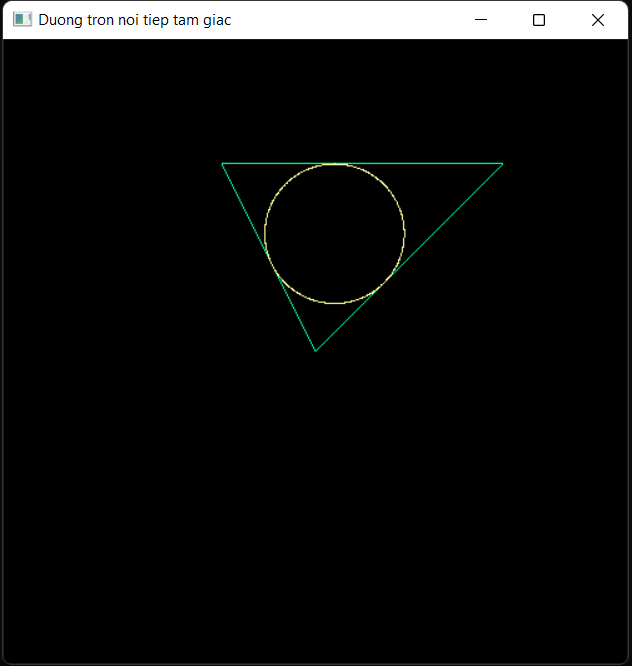
init();

glutDisplayFunc(draw);

glutMainLoop();

return 0;

}



4.    Lập trình OpenGL 2D vẽ các đường cong biểu diễn dưới dạng tham số như dưới đây. Lưu ý thiết lập cửa sổ, khung nhìn và khoảng biến thiên thích hợp cho tham số*t*, để được kết quả như hình vẽ 16.

§  Đồ thị hàm y=sin(x), hàm y=cos(x)

§  x = t + 2.0\*sin(2.0\*t); y = t + 2.0\*cos(5.0\*t);

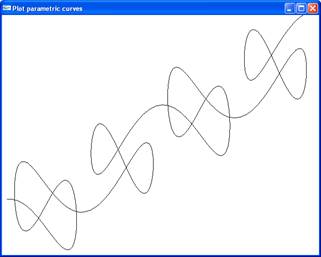
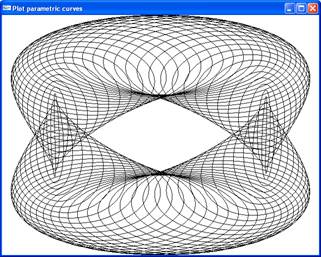
§  x = cos(t) - cos(80\*t)\*sin(t);  y = 2.0\*sin(t) - sin(80\*t);

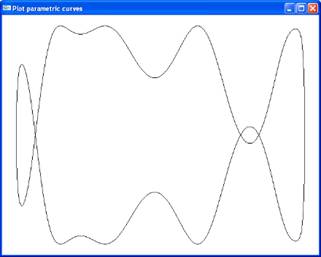
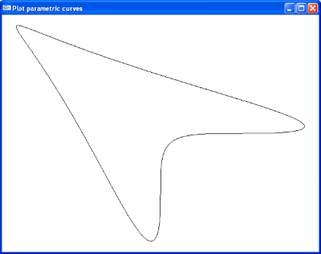
§  x = cos(t); y = sin(t + sin(5.0\*t));

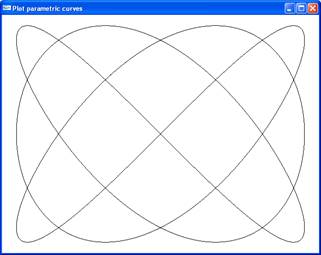
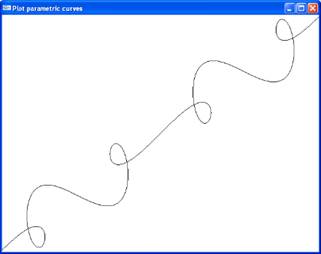
§  x = sin(t+sin(t)); y = cos(t + cos(t));

§  x = t + sin(2.0\*t); y = t + sin(3.0\*t);

§  x = sin(3.0\*t);  y = sin(4.0\*t);

**Hình 16** Đường cong tham số

- Code:

§  Đồ thị hàm y=sin(x), hàm y=cos(x)

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int x, y;

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(x=-360; x<=360; x++)

{

y = 100\*sin(x\*3.14/180);

glVertex3f(x, y, 0);

}

glEnd();

glColor3f (1.0, 0.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(x=-360; x<=360; x++)

{

y = 100\*cos(x\*3.14/180);

glVertex3f(x, y, 0);

}

glEnd();

glViewport(0,0,1500,700);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(1500, 700);

glutInitWindowPosition(0, 0);

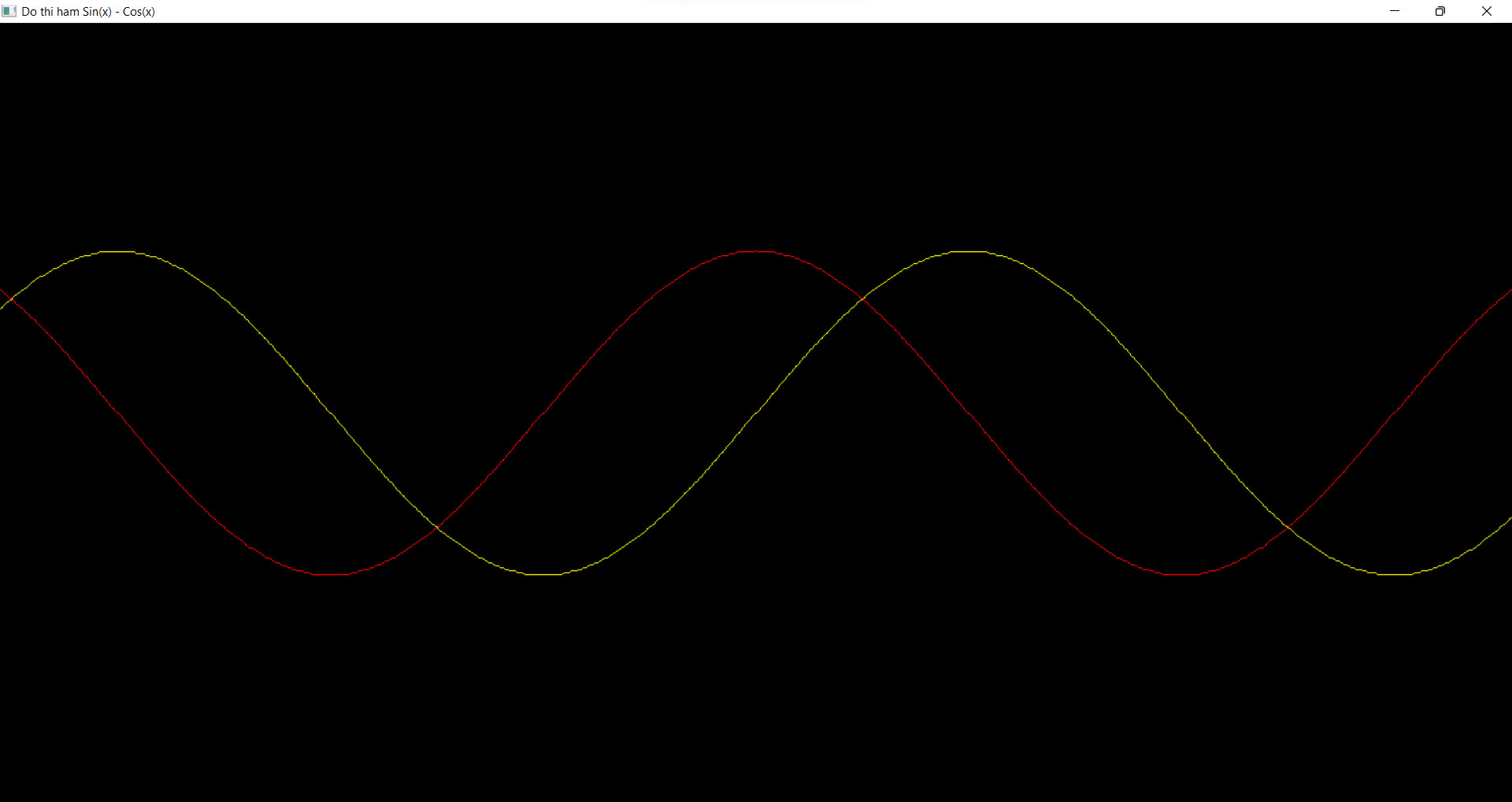
glutCreateWindow("Do thi ham Sin(x) - Cos(x)");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



§  x = t + 2.0\*sin(2.0\*t); y = t + 2.0\*cos(5.0\*t);

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int x, y, t;

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(t=-720; t<=720; t++)

{

x = t + 50\*2.0\*sin(2.0\*t\*3.14/180);

y = t + 50\*2.0\*cos(5.0\*t\*3.14/180);

glVertex3f(x, y, 0);

}

glEnd();

glViewport(0,0,1500,700);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(1500, 700);

glutInitWindowPosition(0, 0);

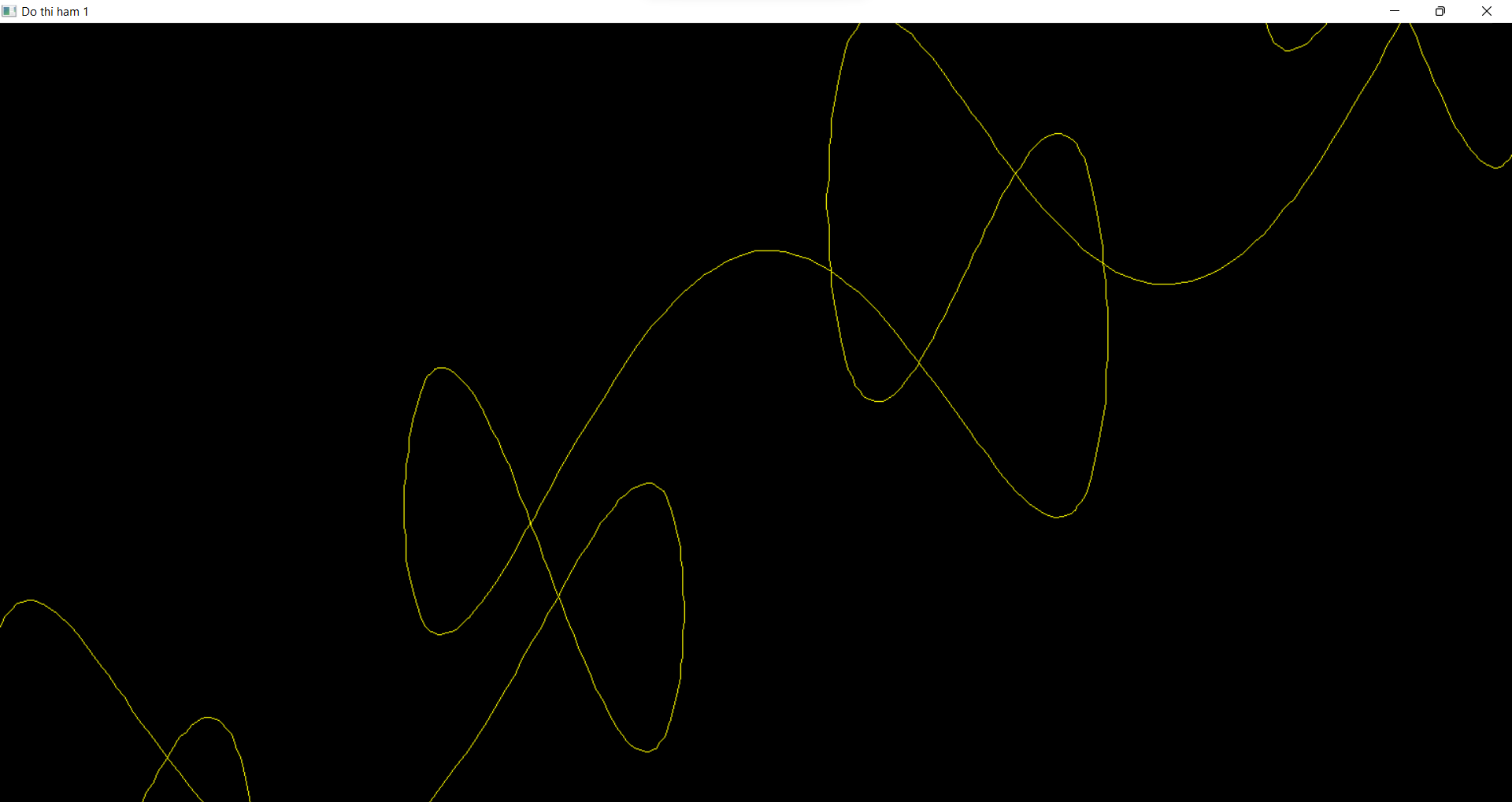
glutCreateWindow("Do thi ham 1");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



§  x = cos(t) - cos(80\*t)\*sin(t);  y = 2.0\*sin(t) - sin(80\*t);

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int x, y, t;

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(t=-720; t<=720; t++)

{

x = 70\*cos(t\*3.14/180) - 70\*cos(80\*t\*3.14/180)\*sin(t\*3.14/180);

y = 70\*2.0\*sin(t\*3.14/180) - 70\*sin(80\*t\*3.14/180);

glVertex3f(x, y, 0);

}

glEnd();

glViewport(0,0,1500,700);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(1500, 700);

glutInitWindowPosition(0, 0);

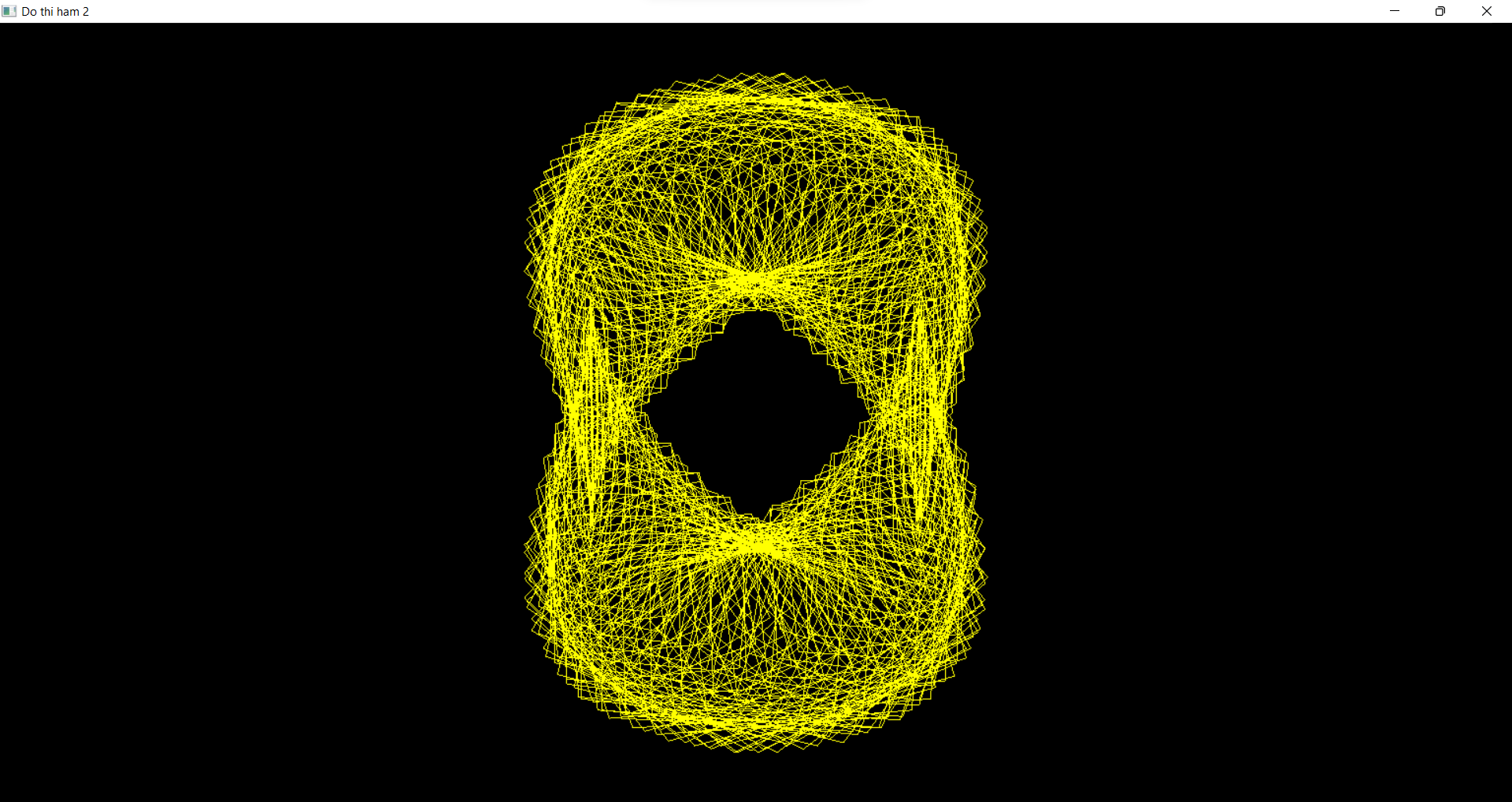
glutCreateWindow("Do thi ham 2");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



§  x = cos(t); y = sin(t + sin(5.0\*t));

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int x, y, t;

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(t=-720; t<=720; t++)

{

x = 100\*cos(t\*3.14/180);

y = 100\*sin(t\*3.14/180 + sin(5.0\*t\*3.14/180));

glVertex3f(x, y, 0);

}

glEnd();

glViewport(0,0,1500,700);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(1500, 700);

glutInitWindowPosition(0, 0);

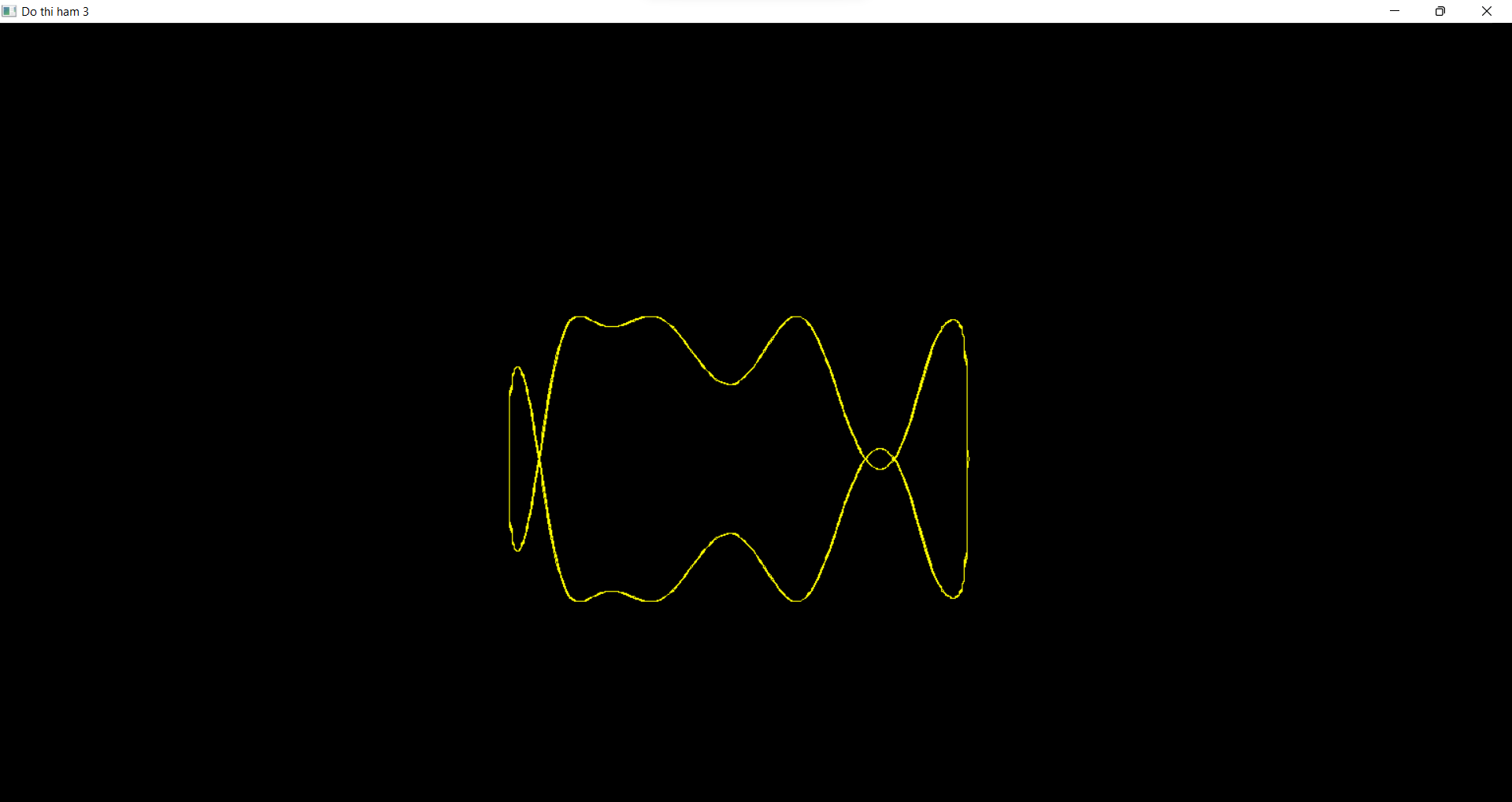
glutCreateWindow("Do thi ham 3");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



§  x = sin(t+sin(t)); y = cos(t + cos(t));

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int x, y, t;

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(t=-720; t<=720; t++)

{

x = 100\*sin(t\*3.14/180+sin(t\*3.14/180));

y = 100\*cos(t\*3.14/180 + cos(t\*3.14/180));

glVertex3f(x, y, 0);

}

glEnd();

glViewport(0,0,1500,700);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(1500, 700);

glutInitWindowPosition(0, 0);

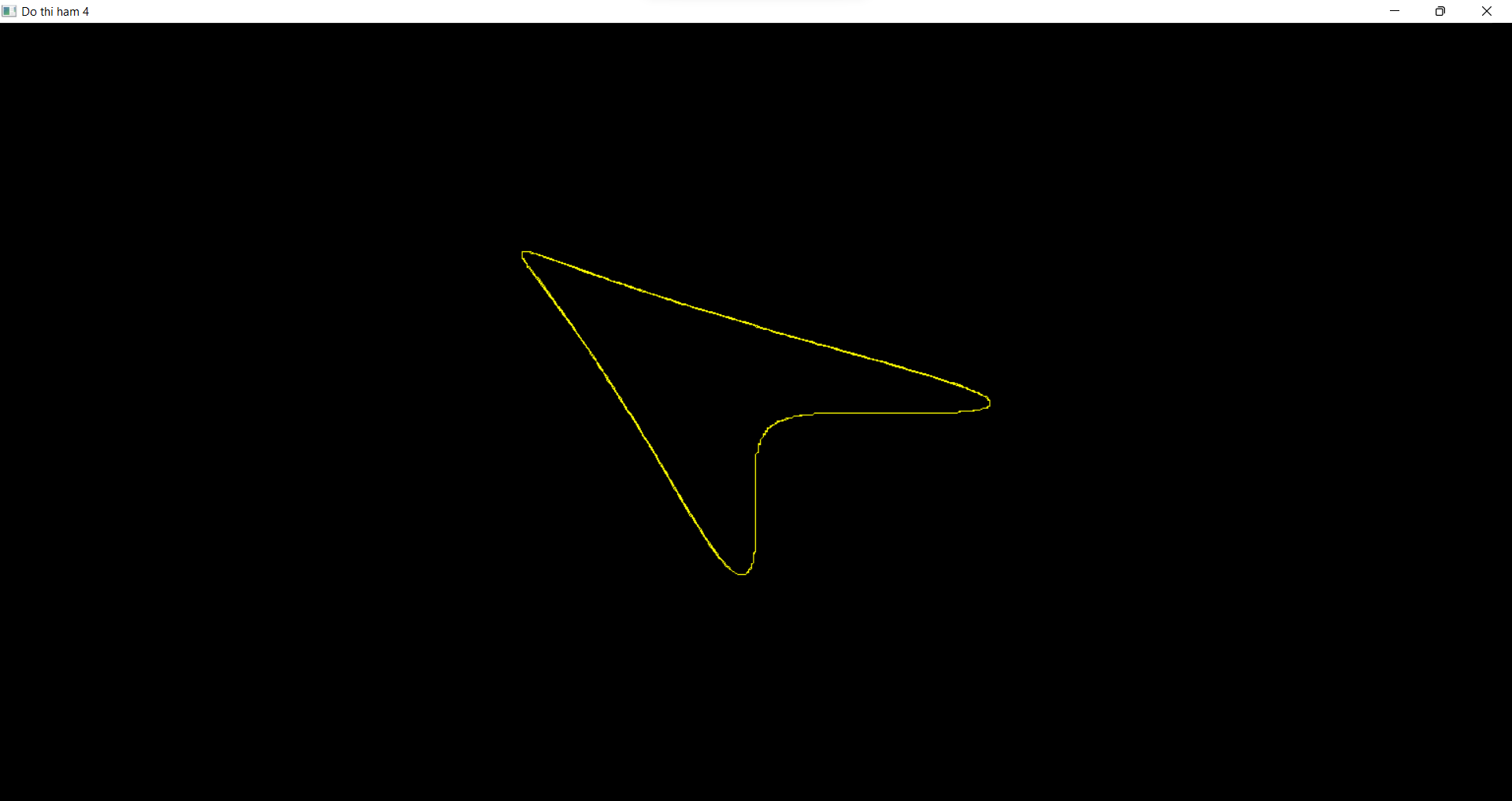
glutCreateWindow("Do thi ham 4");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



§  x = t + sin(2.0\*t); y = t + sin(3.0\*t);

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int x, y, t;

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(t=-720; t<=720; t++)

{

x = t + 50\*sin(2.0\*t\*3.14/180);

y = t + 50\*sin(3.0\*t\*3.14/180);

glVertex3f(x, y, 0);

}

glEnd();

glViewport(0,0,1500,700);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(1500, 700);

glutInitWindowPosition(0, 0);

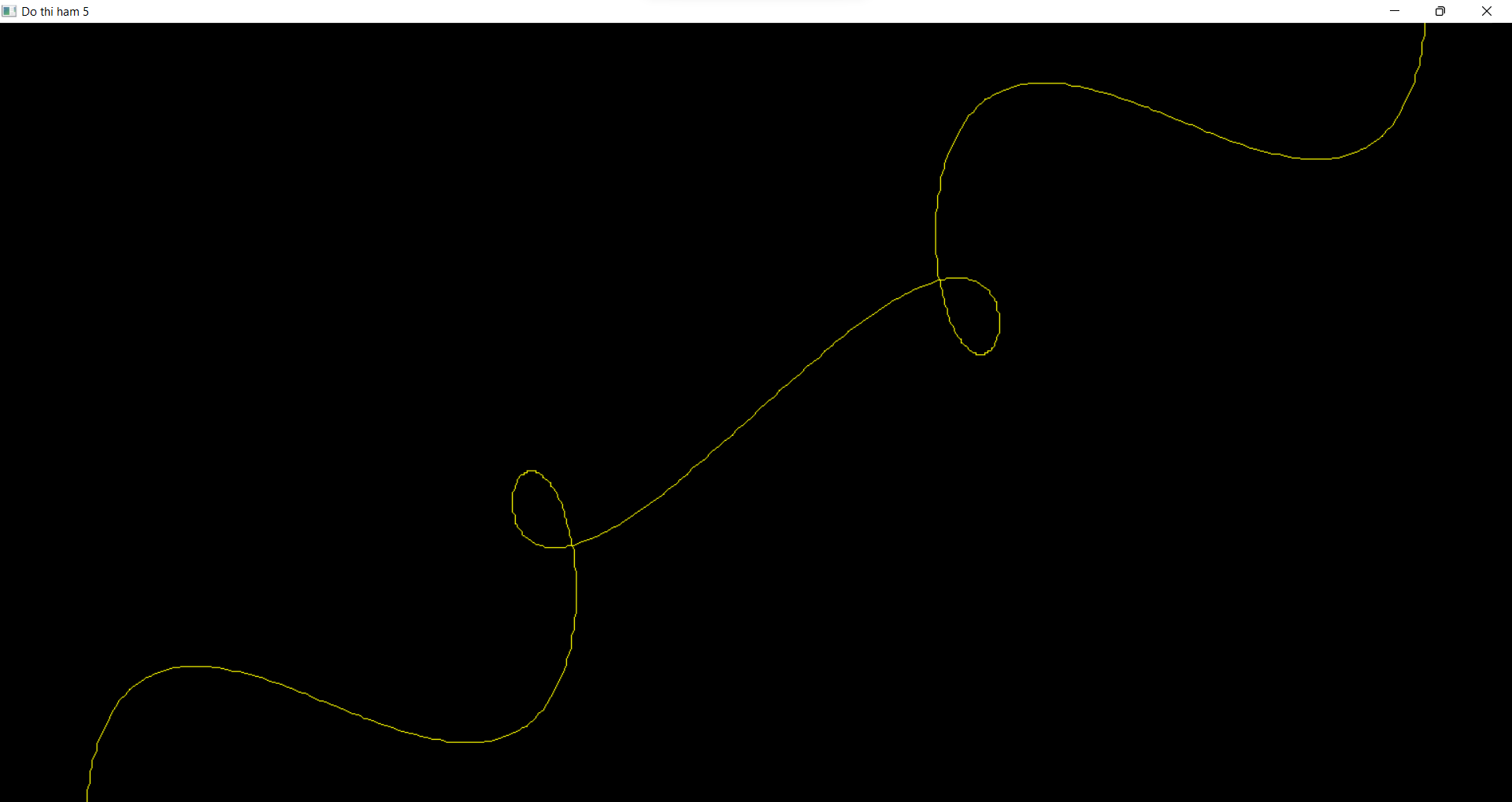
glutCreateWindow("Do thi ham 5");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



§  x = sin(3.0\*t);  y = sin(4.0\*t);

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int x, y, t;

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

for(t=-720; t<=720; t++)

{

x = 100\*sin(3\*t\*3.14/180);

y = 100\*sin(4\*t\*3.14/180);

glVertex3f(x, y, 0);

}

glEnd();

glViewport(0,0,1500,700);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(1500, 700);

glutInitWindowPosition(0, 0);

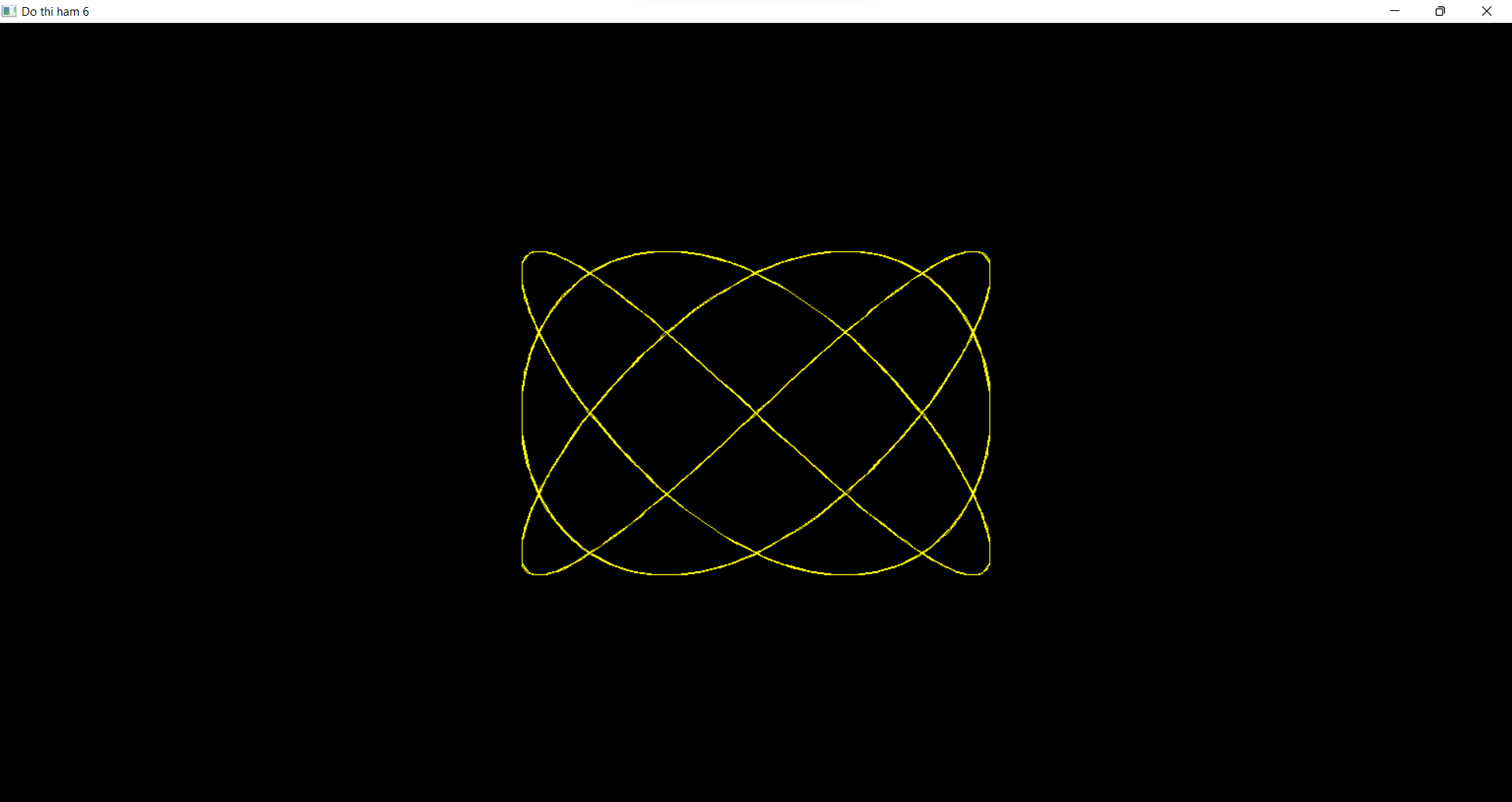
glutCreateWindow("Do thi ham 6");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}

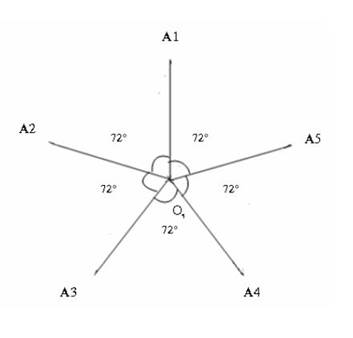


5.    Lập trình OpenGL 2D vẽ ngôi sao 5 cánh

-      Một ngôi sao có 5 đỉnh sẽ cách đều tâm, hai đỉnh kề nhau sẽ cách nhau một góc 72o.

-      Để xác định đỉnh, chúng ta chỉ cần chọn 1 điểm cách tâm 1 khoảng r, sau đó quay quanh tâm theo góc 72ota sẽ được đỉnh tiếp theo.

-      Từ 5 đỉnh của ngôi sao, sử dụng thuật toán nối các điểm thành các đường thẳng tạo thành ngôi sao.

[](http://ent.htu.edu.vn/images/2016/t2/t2_hoai1.png)

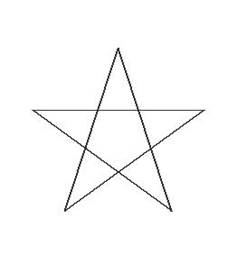
**Thuật toán**

+ Bước 1: Xác định đỉnh của ngôi sao bằng công thức

x[i] = x \*cos(i\* 72o) – y \*sin(y \* 72o) + (y +r)sin(i \* 72o) + x(1- cos(i\*72o))

y[i] = x \*sin(i\* 72o) + y \*cos(y \* 72o) + (y +r)(1-cos(i \* 72o)) + x\*sin(i\*72o)

+ Bước 2: Sử dụng thuật toán DDA nối các điểm: 1 và 3, 1 và 4, 2 và 3, 2 và 5, 3 và 5.

[](http://ent.htu.edu.vn/images/2016/t2/t2hoai2.png)

void ngoisao(Graphics g, intxt, intyt, intr) {

floatx[] = newfloat[5], y[] = newfloat[5], grad = (float) ((72 \* 3.14) / 180);

x[0] = xt;

y[0] = yt - r;

for (inti = 1; i < 5; i++) {

x[i] = (float) (x[0] \* Math.cos(i \* grad) - y[0] \* Math.sin(i \* grad) + yt \* Math.sin(i \* grad) + xt \* (1 - Math.cos(i \* grad)));

y[i] = (float) (x[0] \* Math.sin(i \* grad) + y[0] \* Math.cos(i \* grad) + yt \* (1 - Math.cos(i \* grad)) - xt \* Math.sin(i \* grad));

}//for

…

}

- Code:

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void ngoisao(int xt, int yt, int r)

{

float x[5], y[5], grad = (float) ((72 \* 3.14) / 180);

x[0] = xt;

y[0] = yt - r;

for (int i = 1; i < 5; i++)

{

x[i] = (float) (x[0] \* cos(i \* grad) - y[0] \* sin(i \* grad) + yt \* sin(i \* grad) + xt \* (1 - cos(i \* grad)));

y[i] = (float) (x[0] \* sin(i \* grad) + y[0] \* cos(i \* grad) + yt \* (1 - cos(i \* grad)) - xt \* sin(i \* grad));

}

for (int i = 0; i < 5; i++)

{

glColor3f (1.0, 0.3, 0.0);

glBegin(GL\_POINTS);

glVertex3f(-x[i], -y[i], 0);

glEnd();

}

glColor3f (1.0, 0.3, 0.0);

glBegin(GL\_LINE\_STRIP);

glVertex3f(-x[0], -y[0], 0);

glVertex3f(-x[2], -y[2], 0);

glVertex3f(-x[4], -y[4], 0);

glVertex3f(-x[1], -y[1], 0);

glVertex3f(-x[3], -y[3], 0);

glVertex3f(-x[0], -y[0], 0);

glEnd();

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

ngoisao(0, 0, 100);

glViewport(0,0,640,480);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(640, 480);

glutInitWindowPosition(0, 0);

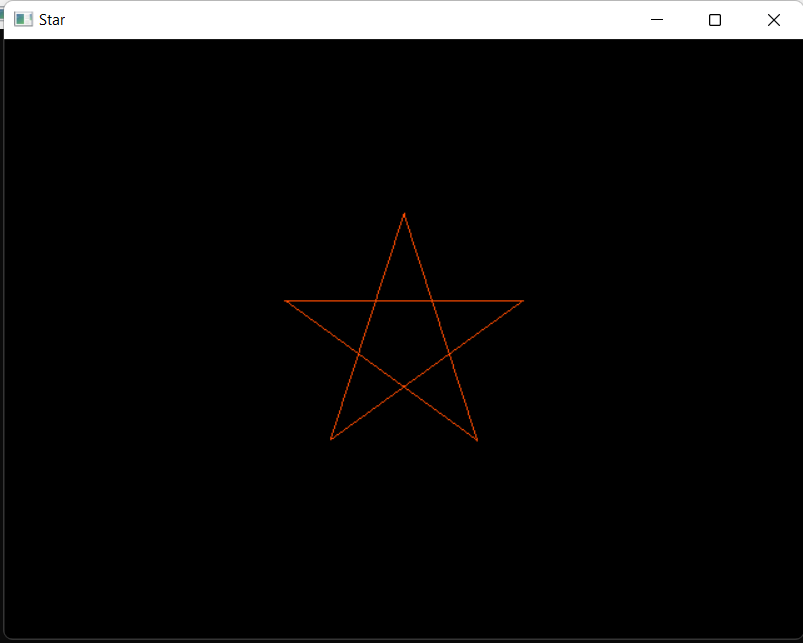
glutCreateWindow("Star");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



6.    Lập trình OpenGL 2D vẽ bông hoa 5 cánh

-      Bông hoa gồm 5 cánh là 5 hình tròn được xếp đều nhau quanh 1 hình tròn tâm và cách đều tâm

-      5 tâm của mỗi cánh cách đều tâm chính, hai tâm kề nhau sẽ cách nhau một góc 72o.

-      Cần chọn 1 điểm cách tâm 1 khoảng r (chúng ta có thể nâng khoảng r lên r +r /3 để các cánh hoa xa nhau hơn), sau đó quay quanh tâm theo góc 72ota sẽ được cái đỉnh tiếp theo.

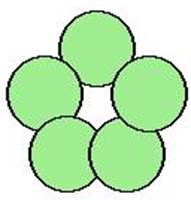
**Thuật toán:**

+ Bước 1: Xác vị trí của cánh theo công thức:

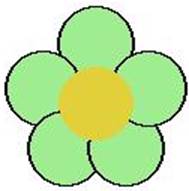
x[i] = x \*cos(i\* 72o) – y \*sin(y \* 72o) + y sin(i \* 72o) + x(1- cos(i\*72o))

y[i] = x \*sin(i\* 72o) + y \*cos(y \* 72o) + y (1-cos(i \* 72o)) + x\*sin(i\*72o)

+ Bước 2: Vẽ và tô màu các cánh

[](http://ent.htu.edu.vn/images/2016/t2/t2_hoai4.png)

+ Bước 3: Vẽ và tô màu tâm vào giữa các cánh.

[](http://ent.htu.edu.vn/images/2016/t2/t2_hoai5.png)

- Cài đặt

void veHoa(Graphics g, intxt, intyt, intr)

{

     floatx[] = newfloat[5], y[] = newfloat[5], grad = (float) ((72 \* 3.14) / 180);

         x[0] = xt;

         y[0] = yt - r;

         r += r/3;

         for (inti = 1; i < 5; i++) {

             x[i] = (float) (x[0] \* Math.cos(i \* grad) - y[0] \* Math.sin(i \* grad) + yt \* Math.sin(i \* grad) + xt \* (1 - Math.cos(i \* grad)));

             y[i] = (float) (x[0] \* Math.sin(i \* grad) + y[0] \* Math.cos(i \* grad) + yt \* (1 - Math.cos(i \* grad)) - xt \* Math.sin(i \* grad));

         }

….

    }

- Code:

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void draw8point(int xc, int yc, int x, int y)

{

glBegin(GL\_POLYGON);

glVertex3f(xc + x, yc + y, 0.0);

glVertex3f(xc + y, yc + x, 0.0);

glVertex3f(xc + y, yc - x, 0.0);

glVertex3f(xc + x, yc - y, 0.0);

glVertex3f(xc - x, yc - y, 0.0);

glVertex3f(xc - y, yc - x, 0.0);

glVertex3f(xc - y, yc + x, 0.0);

glVertex3f(xc - x, yc + y, 0.0);

glEnd();

}

void CircleMidpoint(int xc,int yc,int R )

{

float P;

int y = R;

int x = 0;

P = 5/4 - R;

draw8point(xc, yc, x, y);

while (x < y)

{

if (P < 0) // Chon diem P

{

P += 2\*x + 3;

}

else // Chon diem S

{

P += 2\*(x - y) + 5;

y--;

}

x++;

draw8point(xc,yc,x,y);

}

}

void hoa(int xt, int yt, int r)

{

float x[5], y[5], grad = (float) ((72 \* 3.14) / 180);

x[0] = xt;

y[0] = yt - r;

for (int i = 1; i < 5; i++)

{

x[i] = (float) (x[0] \* cos(i \* grad) - y[0] \* sin(i \* grad) + yt \* sin(i \* grad) + xt \* (1 - cos(i \* grad)));

y[i] = (float) (x[0] \* sin(i \* grad) + y[0] \* cos(i \* grad) + yt \* (1 - cos(i \* grad)) - xt \* sin(i \* grad));

}

for (int i = 0; i < 5; i++)

{

glColor3f (1.0, 1.0, 1.0);

glBegin(GL\_POINTS);

glVertex3f(-x[i], -y[i], 0);

glEnd();

}

glColor3f (0.0, 1.0, 0.7);

CircleMidpoint(-x[0], -y[0], 70 );

CircleMidpoint(-x[1], -y[1], 70 );

CircleMidpoint(-x[2], -y[2], 70 );

CircleMidpoint(-x[3], -y[3], 70 );

CircleMidpoint(-x[4], -y[4], 70 );

glColor3f (1.0, 1.0, 0.5);

CircleMidpoint(0, 0, 70 );

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

hoa(0, 0, 100);

glViewport(0,0,640,480);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(640, 480);

glutInitWindowPosition(0, 0);

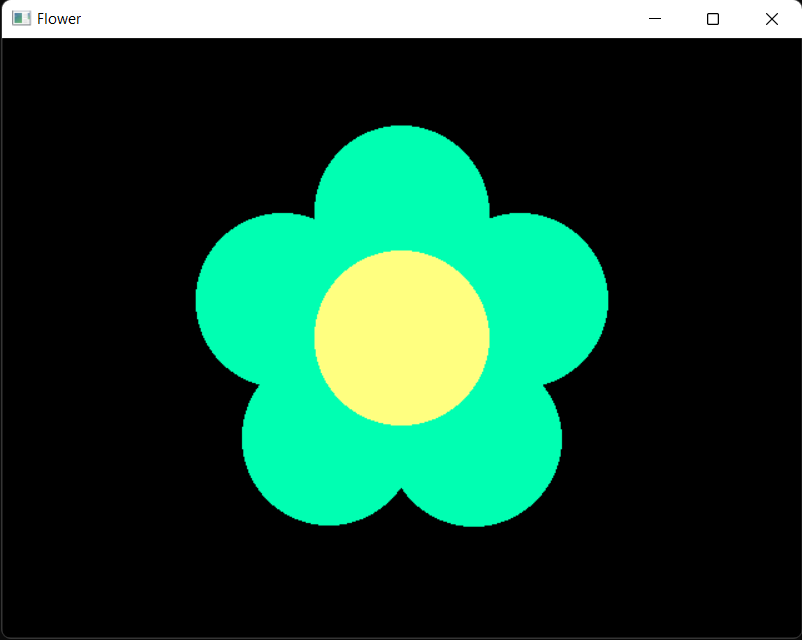
glutCreateWindow("Flower");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



7.    Vẽ quốc kỳ Việt nam

<https://giaphiep.com/blog/ve-quoc-ky-viet-nam-bang-flutter-28004>

#include <GL/glut.h>

#include <Math.h>

void initGL()

{

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

glOrtho(-320,320,-240,240,-1,1);

}

void ngoisao(int xt, int yt, int r)

{

float x[5], y[5], grad = (float) ((72 \* 3.14) / 180);

x[0] = xt;

y[0] = yt - r;

for (int i = 1; i < 5; i++)

{

x[i] = (float) (x[0] \* cos(i \* grad) - y[0] \* sin(i \* grad) + yt \* sin(i \* grad) + xt \* (1 - cos(i \* grad)));

y[i] = (float) (x[0] \* sin(i \* grad) + y[0] \* cos(i \* grad) + yt \* (1 - cos(i \* grad)) - xt \* sin(i \* grad));

}

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_LINE\_STRIP);

glVertex3f(-x[0], -y[0], 0);

glVertex3f(-x[2], -y[2], 0);

glVertex3f(-x[4], -y[4], 0);

glVertex3f(-x[1], -y[1], 0);

glVertex3f(-x[3], -y[3], 0);

glVertex3f(-x[0], -y[0], 0);

glEnd();

glColor3f (1.0, 1.0, 0.0);

glBegin(GL\_TRIANGLES);

glVertex3f(-x[1], -y[1], 0);

glVertex3f(0, -38, 0);

glVertex3f(-x[4], -y[4], 0);

glVertex3f(-x[0], -y[0], 0);

glVertex3f(-40, -10, 0);

glVertex3f(-x[3], -y[3], 0);

glVertex3f(-x[0], -y[0], 0);

glVertex3f(40, -10, 0);

glVertex3f(-x[2], -y[2], 0);

glEnd();

}

void mydisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 0.0, 0.0);

glBegin(GL\_POLYGON);

glVertex3f (-300, -200, 0.0);

glVertex3f (300, -200, 0.0);

glVertex3f (300, 200, 0.0);

glVertex3f (-300, 200, 0.0);

glEnd();

ngoisao(0, 0, 100);

glViewport(0,0,640,480);

glFlush();

}

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

glutInit(&argc, argv);

int mode=GLUT\_SINGLE | GLUT\_RGB;

glutInitDisplayMode(mode);

glutInitWindowSize(640, 480);

glutInitWindowPosition(0, 0);

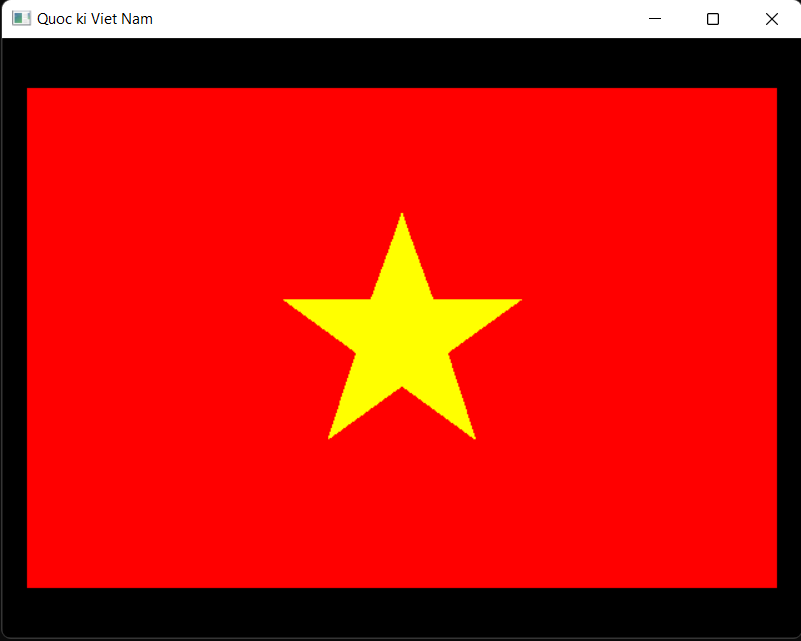
glutCreateWindow("Quoc ki Viet Nam");

initGL();

glutDisplayFunc(mydisplay);

glutMainLoop();

}



----------------------------------------------------------