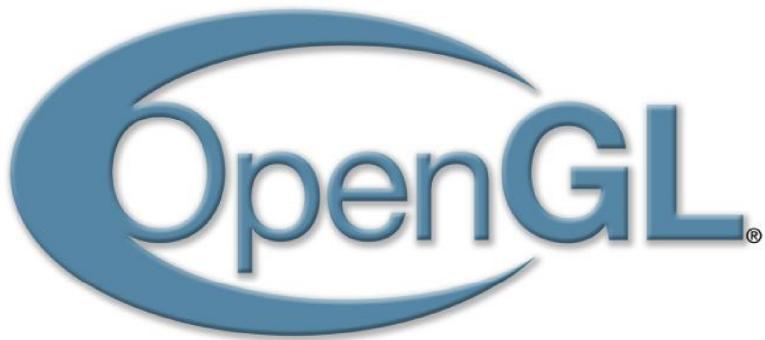


# **ĐỒ HỌA MÁY TÍNH**

Bài thực hành 1

## **OPENGL**

*Version 1.0*



**Lê Viết Tuấn**

TP. HCM, 2018

## MỤC LỤC

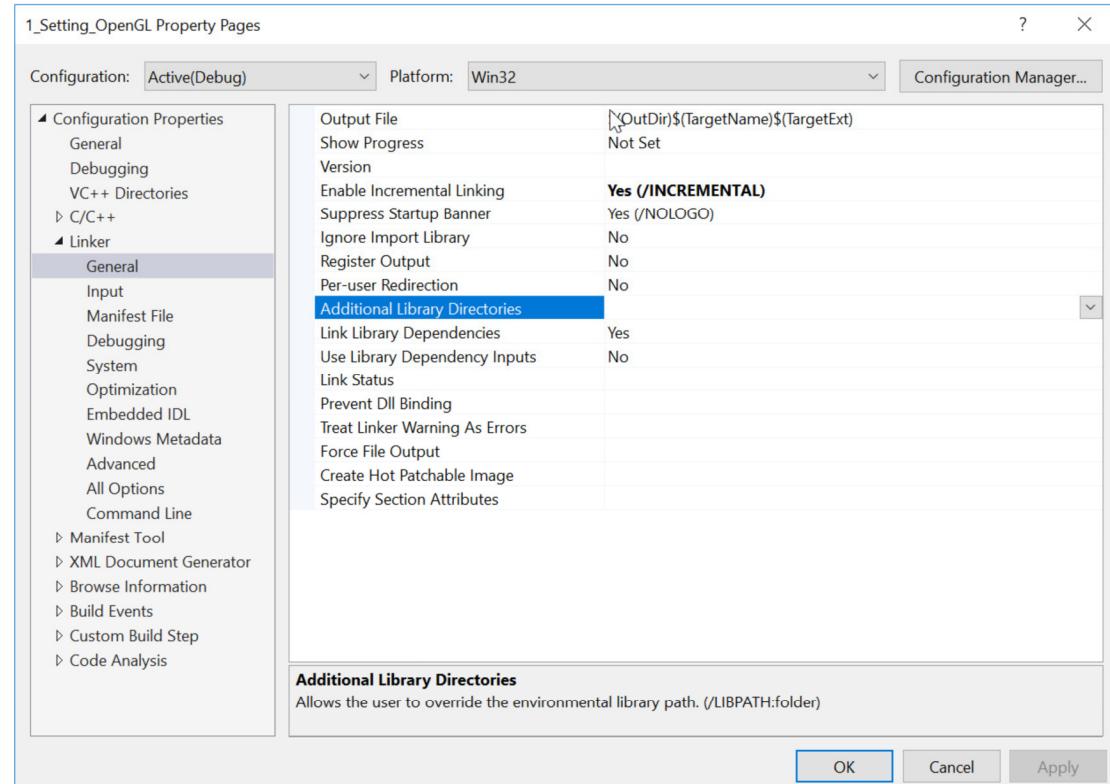
<b>Mục lục.....</b>	<b>i</b>
<b>1. Mục tiêu.....</b>	<b>1</b>
<b>2. Bài tập hướng dẫn.....</b>	<b>1</b>
<b>3. Bài tập tự giải.....</b>	<b>24</b>
<b>4. Tài liệu tham khảo .....</b>	<b>26</b>

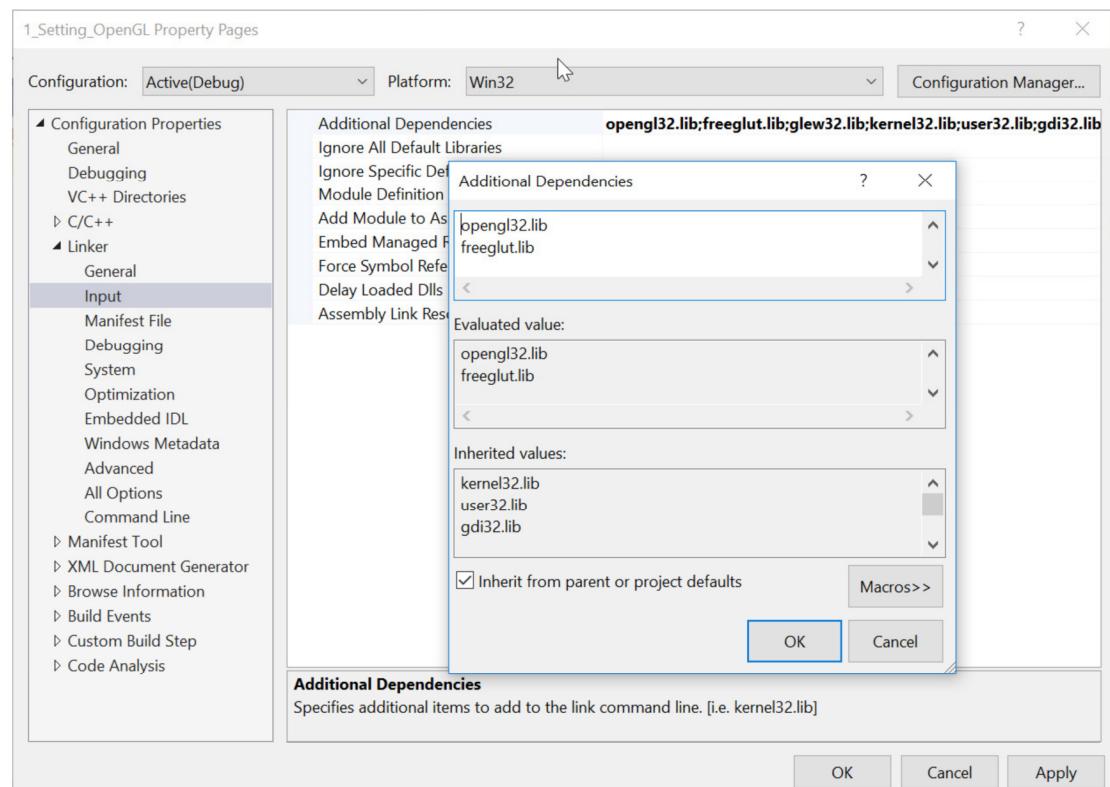
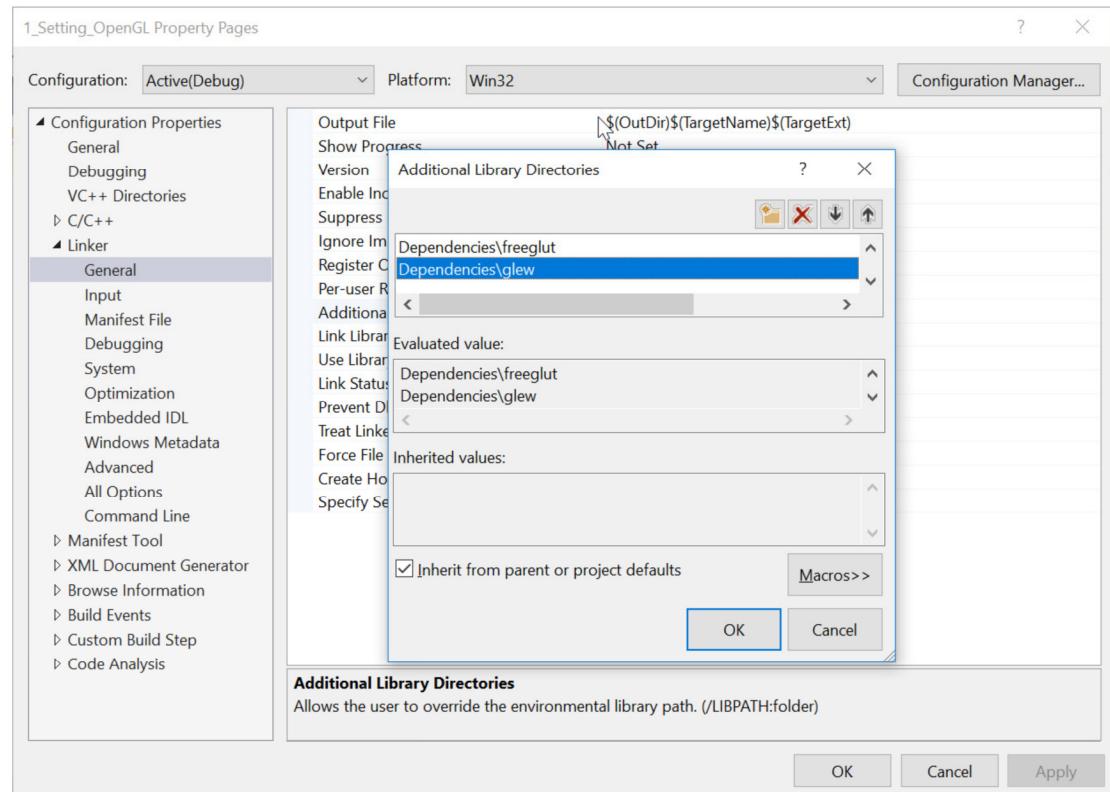
## 1. Mục tiêu

- Sử dụng OpenGL

## 2. Bài tập hướng dẫn

Cấu hình project



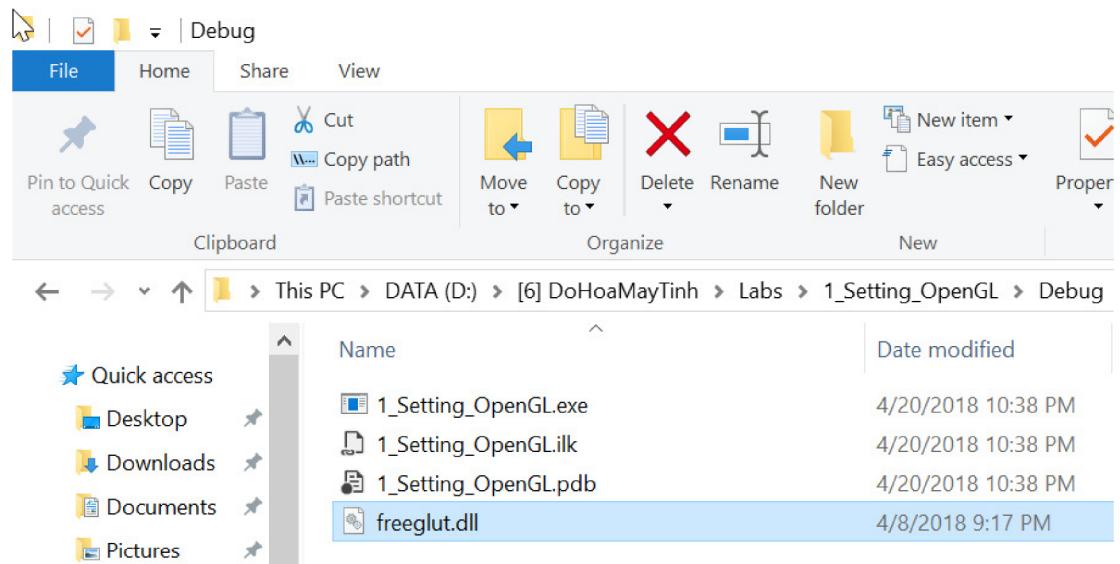


Gồm các file \*.lib sau:

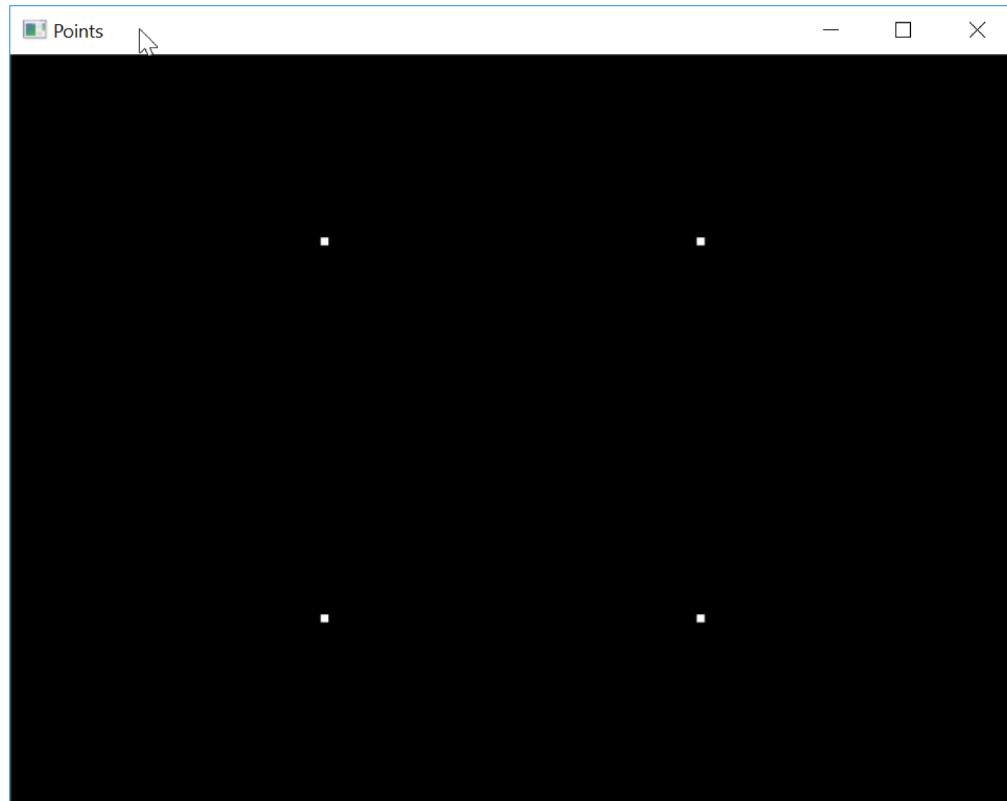
**freeglut.lib**

**glew32.lib**

Copy file freeglut.dll vào thư mục Debug của project



**Bài 1.** Vẽ 4 đỉnh của một hình vuông có tọa độ p1 (200, 120), p2(440, 120),  
p3(440, 360), p4(200, 360)



Hướng dẫn:

```

#include "Dependencies\glew\glew.h"
#include "Dependencies\freeglut\freeglut.h"

const int screenWidth = 640;
const int screenHeight = 480;

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0); //clear black
    glShadeModel(GL_FLAT);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); //Clears the color buffer
    glColor3f(1.0, 1.0, 1.0);
    glPointSize(5.0);

    glBegin(GL_POINTS);
        glVertex3f(200.0, 120.0, 0.0);
        glVertex3f(440.0, 120.0, 0.0);
        glVertex3f(440.0, 360.0, 0.0);
        glVertex3f(200.0, 360.0, 0.0);
    glEnd();

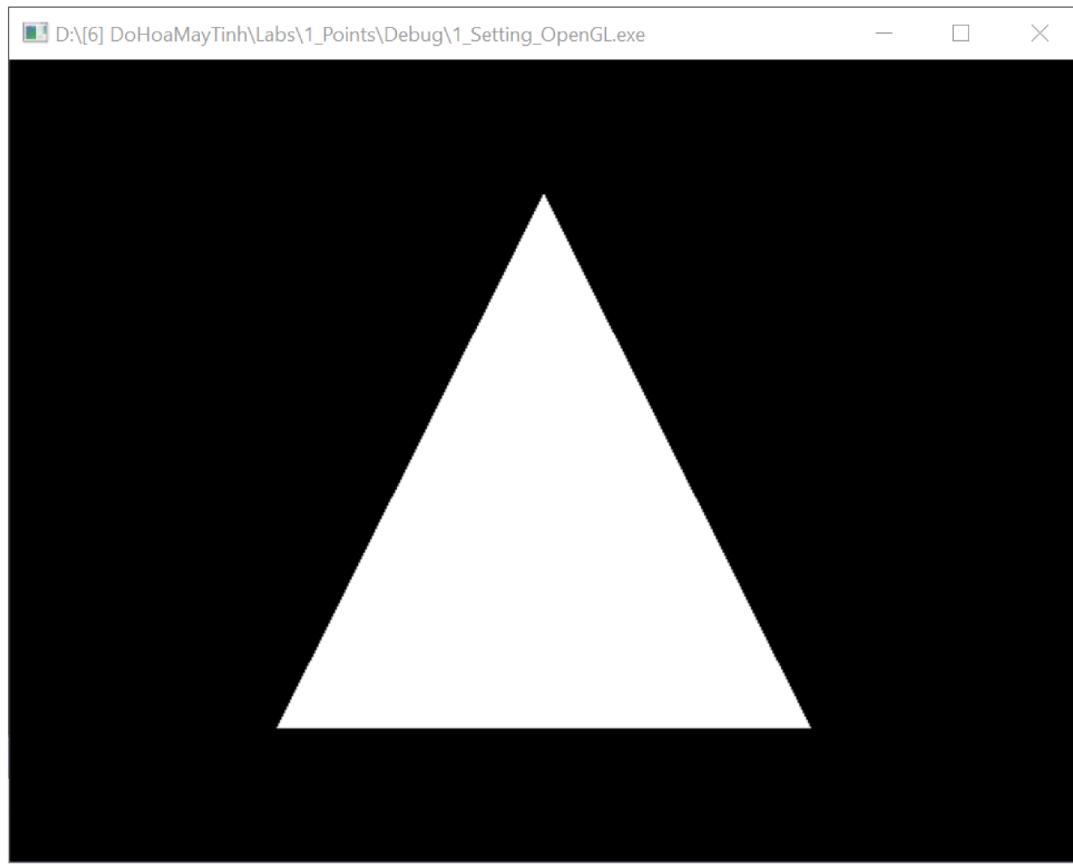
    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(screenWidth, screenHeight); //optional
    glutInitWindowPosition(100, 100); //optional
    glutCreateWindow("Points");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

```

**Bài 2.** Vẽ hình tam giác bất 3 đỉnh của tam giác có tọa độ lần lượt là  $v_1(160, 160)$ ,  $v_2(480, 160)$ ,  $v_3(320, 320)$



Hướng dẫn:

```
#include "Dependencies\glew\glew.h"
#include "Dependencies\freeglut\freeglut.h"

const int screenWidth = 640;
const int screenHeight = 480;

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0); //clear black
    glShadeModel(GL_FLAT);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); //Clears the color buffer
    glColor3f(1.0, 1.0, 1.0);

    glBegin(GL_TRIANGLES);
        glVertex2i(160, 80);
        glVertex2i(480, 80);
        glVertex2i(320, 400);
    glEnd();

    glFlush();
}
```

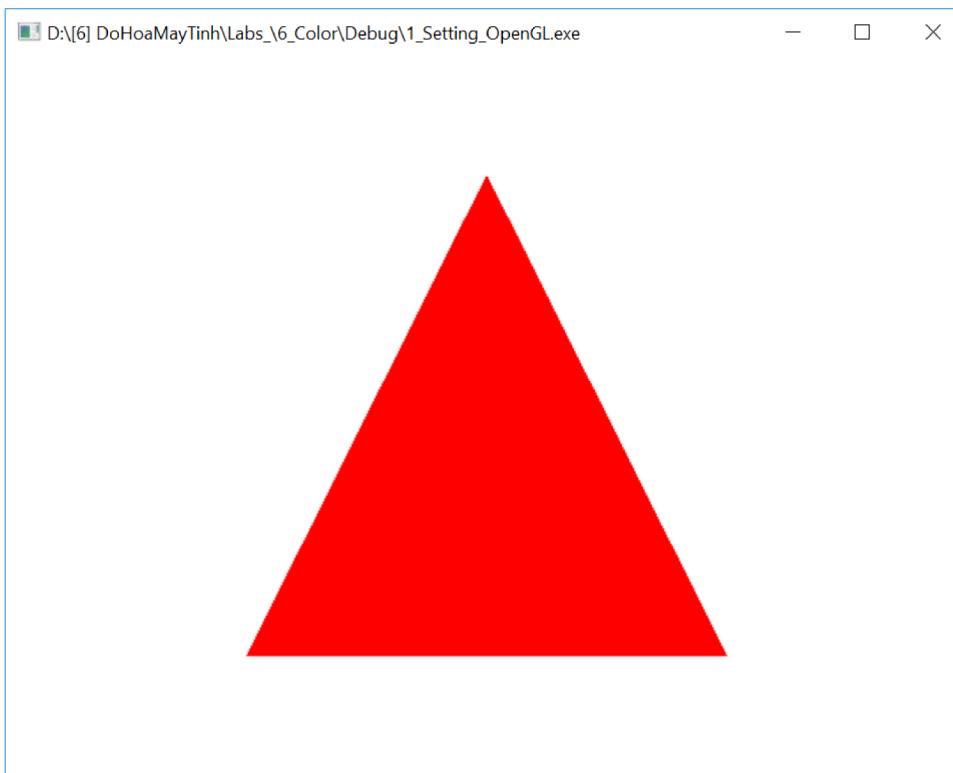
```

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv); // Khởi tạo glut
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // Khởi tạo chế độ vẽ
    single buffer và hệ màu RGB
    glutInitWindowSize(screenWidth, screenHeight); //optional
    glutInitWindowPosition(100, 100); //optional
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

```

### Bài 3. Tô màu đồng nhất cho tam giác



Hướng dẫn:

```

#include "Dependencies\glew\glew.h"
#include "Dependencies\freeglut\freeglut.h"

```

```
const int screenWidth = 640;
const int screenHeight = 480;

void init(void)
{
    glClearColor(1.0, 1.0, 1.0, 1.0); //clear white
}

void triangle()
{
    //To màu flat
    glBegin(GL_TRIANGLES);
        glColor3f(1.0, 0.0, 0.0); //màu đỏ
        glVertex2i(50, 50);
        glVertex2i(250, 50);
        glVertex2i(150, 250);
    glEnd();
}

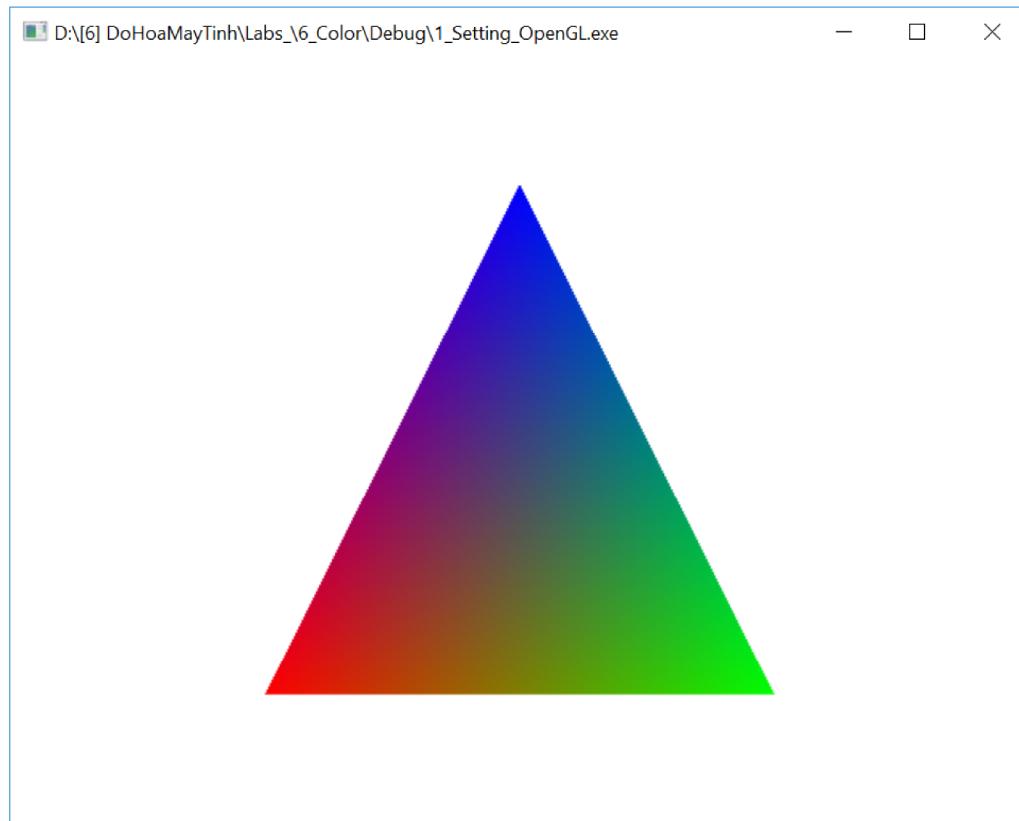
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    triangle();

    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(screenWidth, screenHeight); //optional
    glutInitWindowPosition(100, 100); //optional
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```

**Bài 4.** Tô màu cho tam giác như hình sau



### Hướng dẫn:

Vẽ tam giác với mỗi đỉnh lần lượt có các màu đỏ, lục và lam. Ta sử dụng hàm  `glColor3f` với 3 thành phần r, g, b trong đoạn [0..1] để gán trị màu cho từng đỉnh của tam giác.

```
#include "Dependencies\glew\glew.h"
#include "Dependencies\freeglut\freeglut.h"

const int screenWidth = 640;
const int screenHeight = 480;

void init(void)
{
    glClearColor(1.0, 1.0, 1.0, 1.0); //clear white
}

void triangle()
{
    glBegin(GL_TRIANGLES);
        glColor3f(1.0, 0.0, 0.0);
        glVertex2i(160, 80);
        glColor3f(0.0, 1.0, 0.0);
        glVertex2i(480, 80);
        glColor3f(0.0, 0.0, 1.0);
        glVertex2i(320, 400);
    glEnd();
}
```

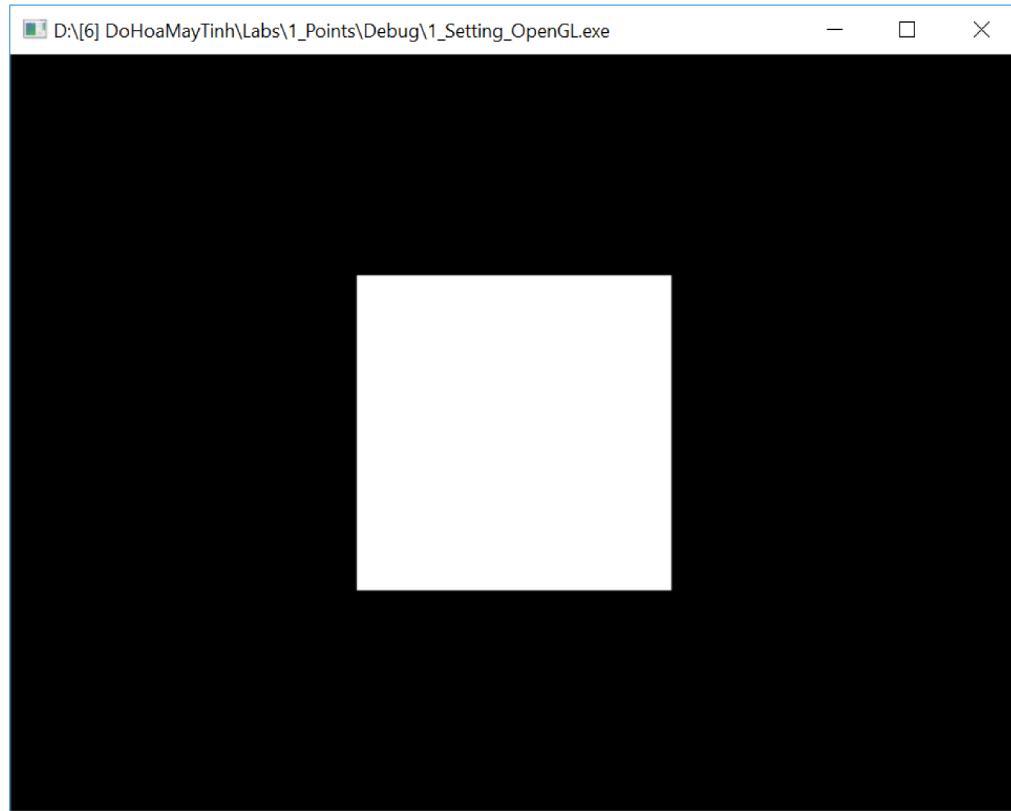
```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    triangle();

    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowSize(screenWidth, screenHeight); //optional
    glutInitWindowPosition(100, 100); //optional
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```

**Bài 5.** Xây dựng một hàm vẽ hình vuông có độ dài cạnh là a bắt đầu từ tọa độ (x, y). Áp dụng để vẽ hình vuông có độ dài cạnh là a = 200 và được đặt vào chính giữa cửa sổ OpenGL. Gợi ý: sử dụng glBegin và GL\_POLYGON.



Hướng dẫn:

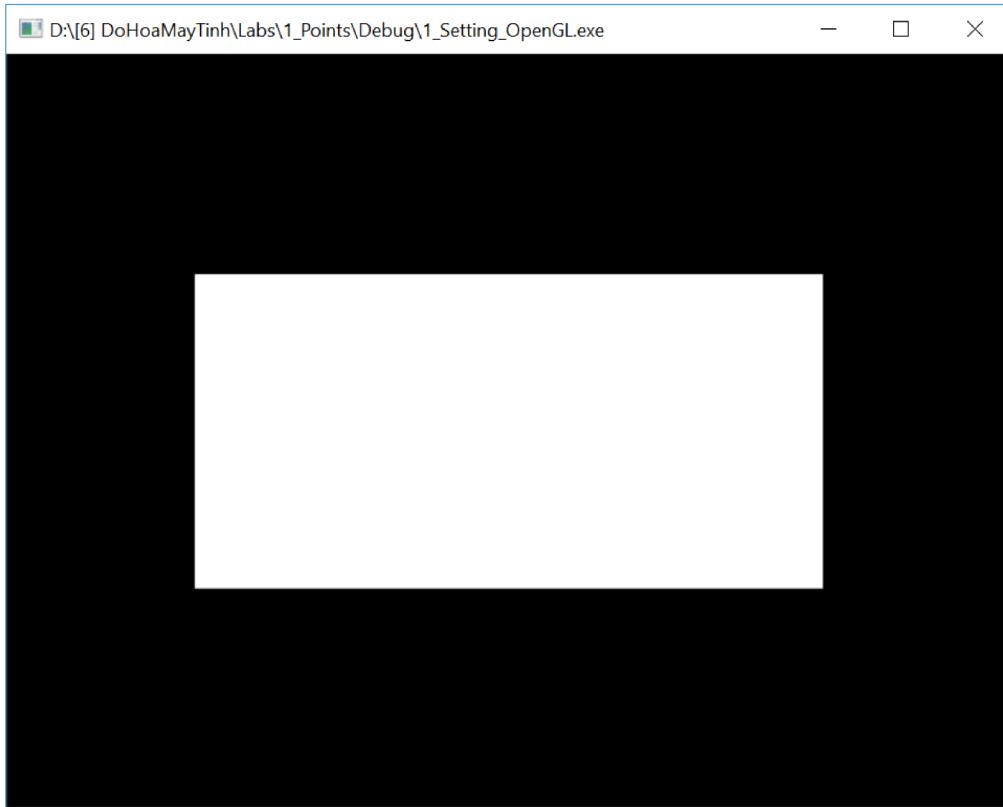
```
void drawSquare(GLint x, GLint y, GLint a)
{
    glBegin(GL_POLYGON);
        glVertex2i(x, y);
        glVertex2i(x + a, y);
        glVertex2i(x + a, y + a);
        glVertex2i(x, y + a);
    glEnd();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); //Clears the color buffer

    GLint size = 200; // độ dài cạnh = 200
    GLint x = (screenWidth - size) / 2; // cạnh vào giữa theo x
    GLint y = (screenHeight - size) / 2; // cạnh vào giữa theo y
    drawSquare(x, y, size);

    glFlush();
}
```

**Bài 6.** Xây dựng một hàm vẽ hình chữ nhật có độ dài chiều rộng là a và chiều cao là b, bắt đầu tại tọa độ (x, y). Áp dụng để vẽ một hình chữ nhật có chiều rộng 400, chiều cao 200 và được đặt vào chính giữa cửa sổ OpenGL



Hướng dẫn:

```

void drawRectangle(GLint x, GLint y, GLint width, GLint height)
{
    glBegin(GL_POLYGON);
    glVertex2i(x, y);           // (trên, trái) +-----+
    glVertex2i(x + width, y);   // (trên, phải) ^     |
    glVertex2i(x + width, y + height); // (dưới, phải) |     v
    glVertex2i(x, y + height);   // (dưới, trái) +-----+
    glEnd();
}

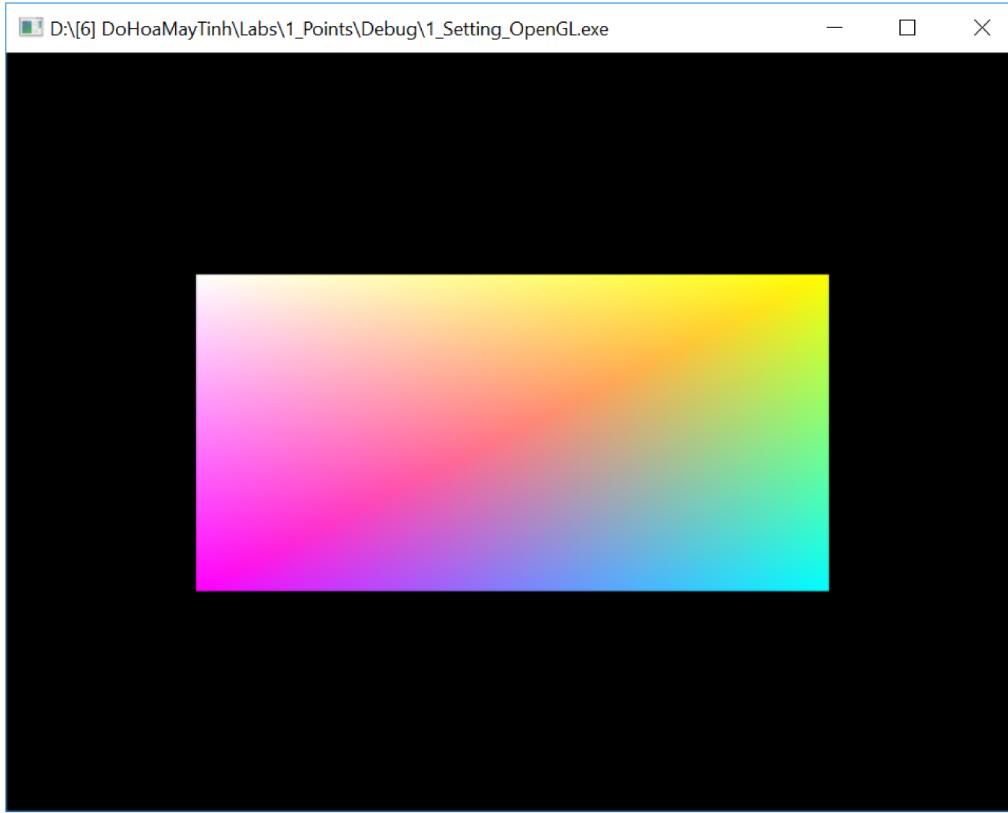
void display()
{
    glClear(GL_COLOR_BUFFER_BIT); //Clears the color buffer

    GLint w = 400; h = 200; // chiều rộng = 400, chiều cao = 200
    GLint x = (screenWidth - w) / 2; // cạnh vào giữa màn hình
    GLint y = (screenHeight - h) / 2;
    drawRectangle(x, y, w, h);

    glFlush();
}

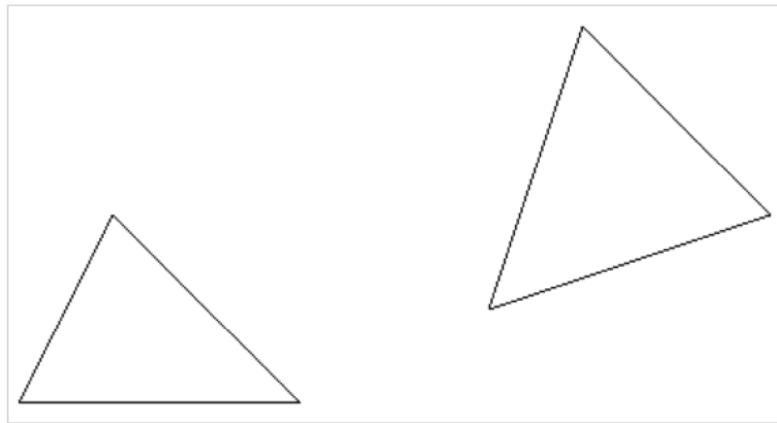
```

```
}
```

**Bài 7.** Tô màu cho hình chữ nhậtHướng dẫn:

```
void drawRectangle(GLint x, GLint y, GLint width, GLint height)
{
    glBegin(GL_POLYGON);
    glColor3f(1.0f, 0.0f, 1.0f); // đỉnh có màu màu hồng
    glVertex2i(x, y);
    glColor3f(0.0f, 1.0f, 1.0f); // đỉnh có màu xanh da trời
    glVertex2i(x + width, y);
    glColor3f(1.0f, 1.0f, 0.0f); // đỉnh có màu xanh vàng
    glVertex2i(x + width, y + height);
    glColor3f(1.0f, 1.0f, 1.0f); // đỉnh có màu trắng
    glVertex2i(x, y + height);
    glEnd();
}
```

**Bài 8.** Cho các điểm  $v_0 = (50, 50)$ ,  $v_1 = (100, 150)$ ,  $v_2 = (200, 50)$ ,  $v_3 = (180, 100)$ ,  $v_4 = (450, 150)$ ,  $v_5 = (350, 250)$ . Vẽ các hình sau

Hướng dẫn:

```
#include "Dependencies\glew\glew.h"
#include "Dependencies\freeglut\freeglut.h"

const int screenWidth = 640;
const int screenHeight = 480;

void init(void)
{
    //glClearColor(0.0, 0.0, 0.0, 0.0); //clear black
    glClearColor(1.0, 1.0, 1.0, 1.0); //clear white
    glShadeModel(GL_FLAT);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); // Clears the color buffer
    //glColor3f(1.0, 1.0, 1.0); // white
    glColor3f(0.0, 0.0, 0.0); // black
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

    glBegin(GL_TRIANGLES);
        glVertex2i(50, 50); //v0
        glVertex2i(200, 50); //v2
        glVertex2i(100, 150); //v1

        glVertex2i(300, 100); //v3
        glVertex2i(450, 150); //v4
        glVertex2i(350, 250); //v5
    glEnd();

    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
}

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

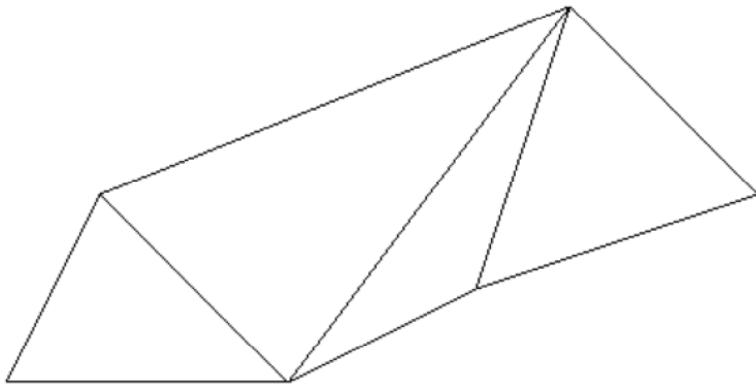
```

glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
glutInitWindowSize(screenWidth, screenHeight); //optional
glutInitWindowPosition(100, 100); //optional
glutCreateWindow(argv[0]);
init();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;

}

```

Tương tự vẽ hình sau:



Ta sẽ sử dụng đoạn mã lệnh sau để vẽ

```

glBegin(GL_TRIANGLE_STRIP);
    glVertex2i(50, 50);
    glVertex2i(100, 150);
    glVertex2i(200, 50);

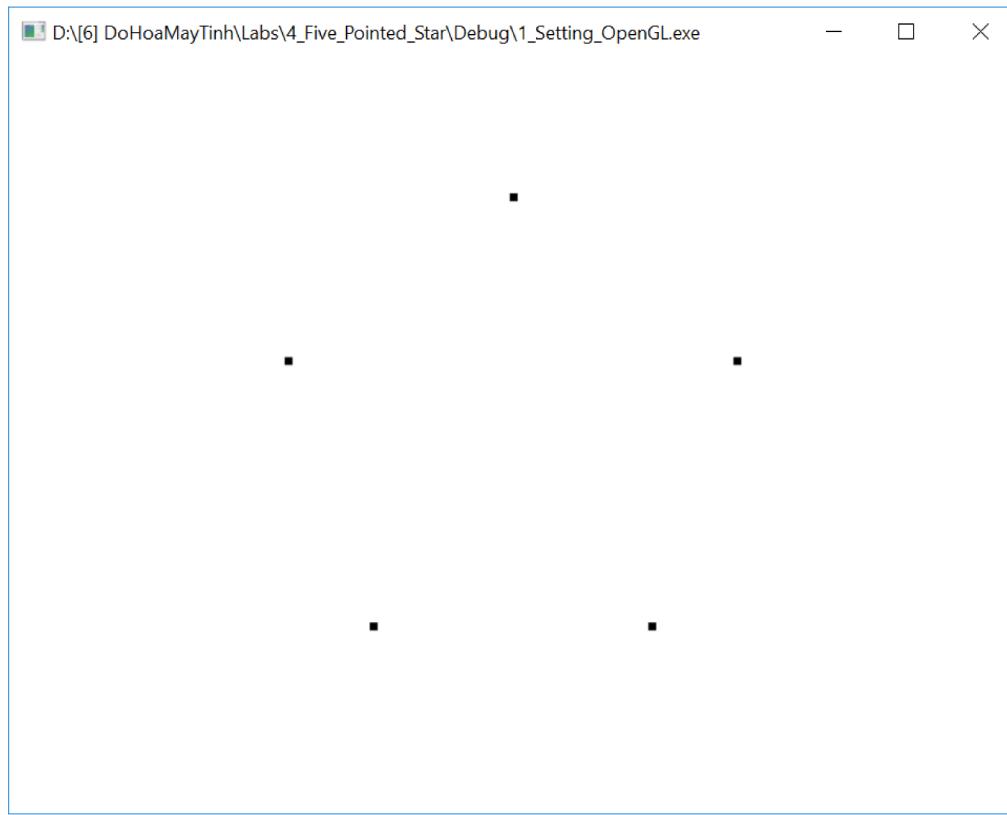
    glVertex2i(350, 250);
    glVertex2i(300, 100);
    glVertex2i(450, 150);

glEnd();

```

**Bài 9.** Vẽ 5 đỉnh của một ngôi sao như hình sau, biết

- Tâm ngôi sao nằm giữa màn hình
- Bán kính  $R = 150$  (khoảng cách từ tâm đến một đỉnh của ngôi sao).



Hướng dẫn:

```
#include <math.h>
#include "Dependencies\glew\glew.h"
#include "Dependencies\freeglut\freeglut.h"

const int screenWidth = 640;
const int screenHeight = 480;
const double R = 150;
const double pi = 3.141592654;

struct GLdoublePoint
{
    GLdouble x;
    GLdouble y;
};

void init(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0); // Thiết lập màu nền (trắng)
    glShadeModel(GL_FLAT);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); // Xóa màn hình
    glPointSize(5.0);
    glColor3f(0.0f, 0.0f, 0.0f); // Thiết lập màu vẽ (đen)
    glBegin(GL_POINTS);
        //glVertex2d((GLdouble)screenWidth / 2, (GLdouble)screenHeight
    / 2);
```

```

GLdoublePoint V1, V2, V3, V4, V5, V0;
V0.x = screenWidth / 2;
V0.y = screenHeight / 2;
V1.x = V0.x;
V1.y = V0.y + R;
V2.x = V0.x + R*sin(2 * pi / 5);
V2.y = V0.y + R*cos(2 * pi / 5);
V3.x = V0.x + R*sin(pi / 5);
V3.y = V0.y - R*cos(pi / 5);
V4.x = V0.x - R*sin(pi / 5);
V4.y = V0.y - R*cos(pi / 5);
V5.x = V0.x - R*sin(2 * pi / 5);
V5.y = V0.y + R*cos(2 * pi / 5);

//glVertex2d(V0.x, V0.y);
glVertex2d(V1.x, V1.y);
glVertex2d(V2.x, V2.y);
glVertex2d(V3.x, V3.y);
glVertex2d(V4.x, V4.y);
glVertex2d(V5.x, V5.y);

glEnd();

glFlush(); // Đẩy ra buffer
}

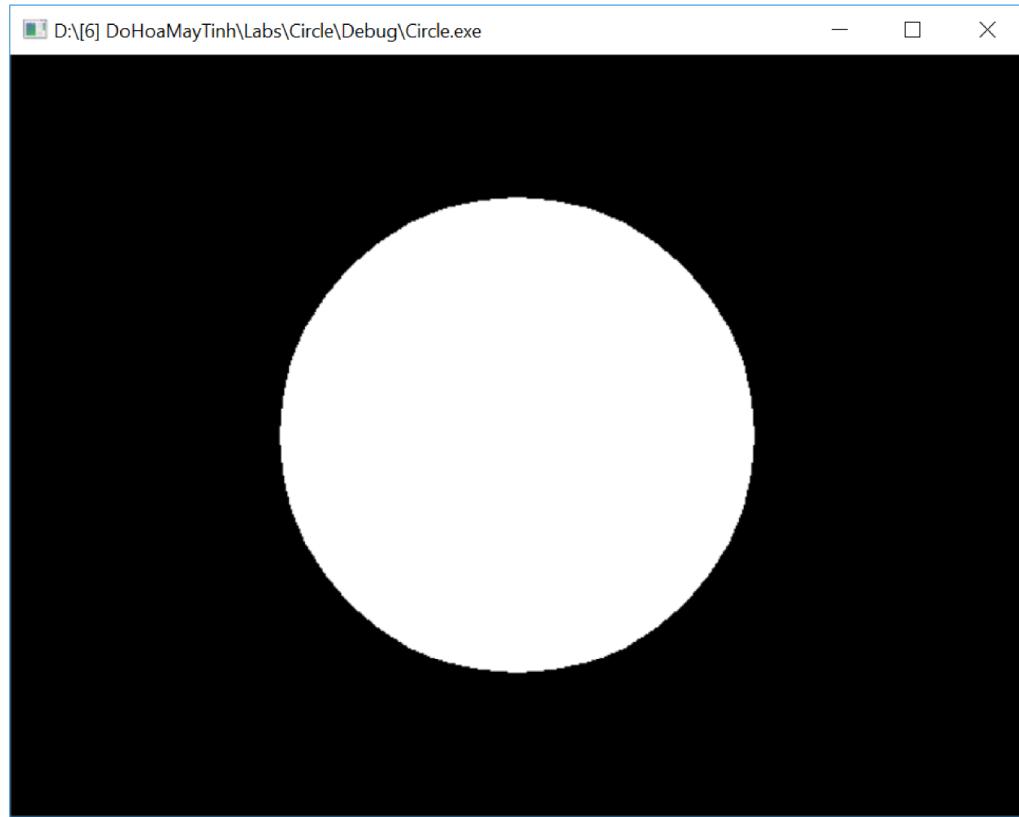
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv); // Khởi tạo glut
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA); // Khởi tạo chế độ vẽ
single buffer và hệ màu RGB
    glutInitWindowSize(screenWidth, screenHeight); //optional
    glutInitWindowPosition(100, 100); //optional
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

```

### Bài 10. Vẽ hình tròn

Sử dụng glBegin và GL\_TRIANGLE\_FAN, hãy xây dựng một hàm vẽ hình tròn có tâm (x, y) và bán kính r. Áp dụng vẽ hình tròn có  $r = 150$  và tâm tại tọa độ (320,240)



Hướng dẫn:

```
#define PI 3.14159265358979323846
#define STEPS 40

void glCircle(GLint x, GLint y, GLint radius)
{
    GLfloat twicePi = (GLfloat) 2.0f * PI;
    glBegin(GL_TRIANGLE_FAN);
    glVertex2i(x, y); // circle center
    for (int i = 0; i <= STEPS; i++) // draw a triangle fan
    {
        glVertex2i((GLint)(x + (radius * cos(i * twicePi / STEPS)) +
0.5),
                    (GLint)(y + (radius * sin(i * twicePi / STEPS)) + 0.5));
    }
    glEnd();
}

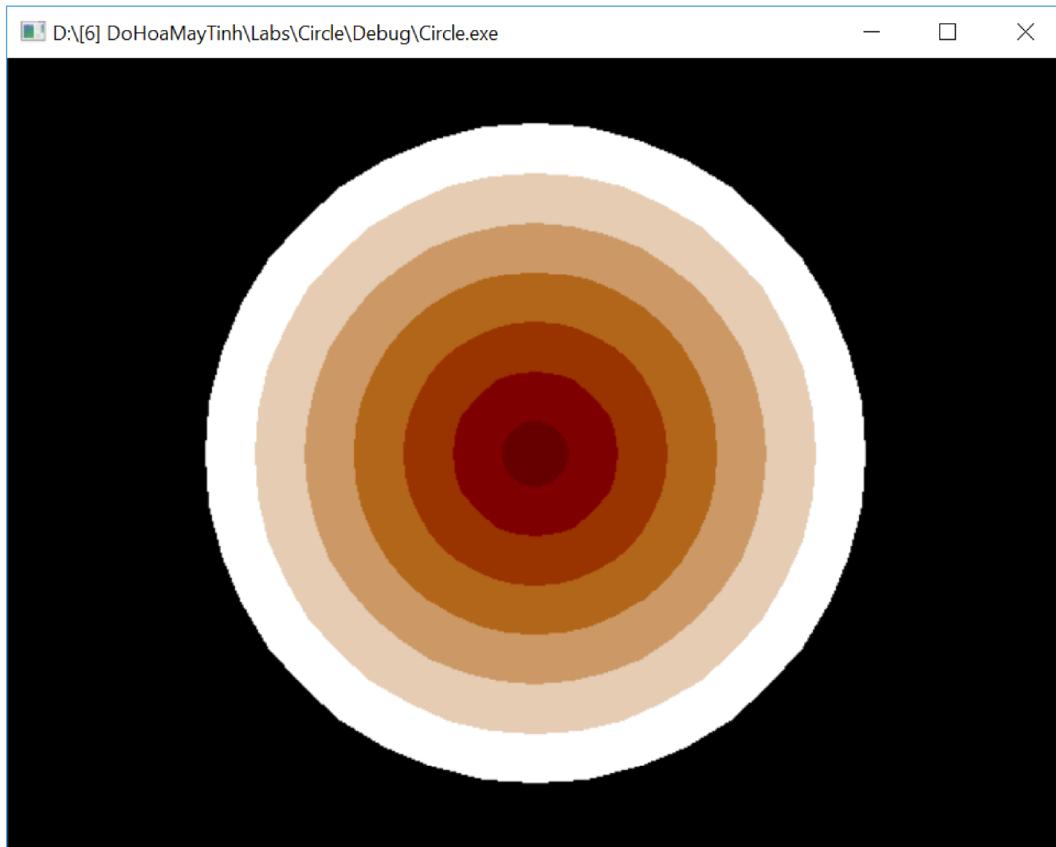
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);

    glCircle(320, 240, 150);

    glFlush();
}
```

{

**Bài 11.** Hãy vẽ các vòng tròn đồng tâm tại điểm (320,240). Bán kính ban đầu  $r = 100$ , giá trị màu ban đầu ( $\text{red} = 1.0$ ,  $\text{green} = 1.0$ ,  $\text{blue} = 1.0$ ). Sau mỗi bước lặp, ta giảm bán kính một lượng là  $\Delta r = 30$ ,  $\Delta \text{red} = 0.1$ ,  $\Delta \text{green} = 0.2$ ,  $\Delta \text{blue} = 0.3$ . Quá trình lặp này dừng khi  $r \leq 0$ .



Hướng dẫn:

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT); // Clears the color buffer

    GLfloat red = 1.0f; // initial R, G, B values
    GLfloat green = 1.0f;
    GLfloat blue = 1.0f;
    for (int r = 200; r > 0; r -= 30) // draw from bigger to smaller circle
    {
        glColor3f(red, green, blue);
        glCircle(320, 240, r); // all circles have the same center
        red -= 0.1f; // update the R, G, B value
        green -= 0.2f;
```

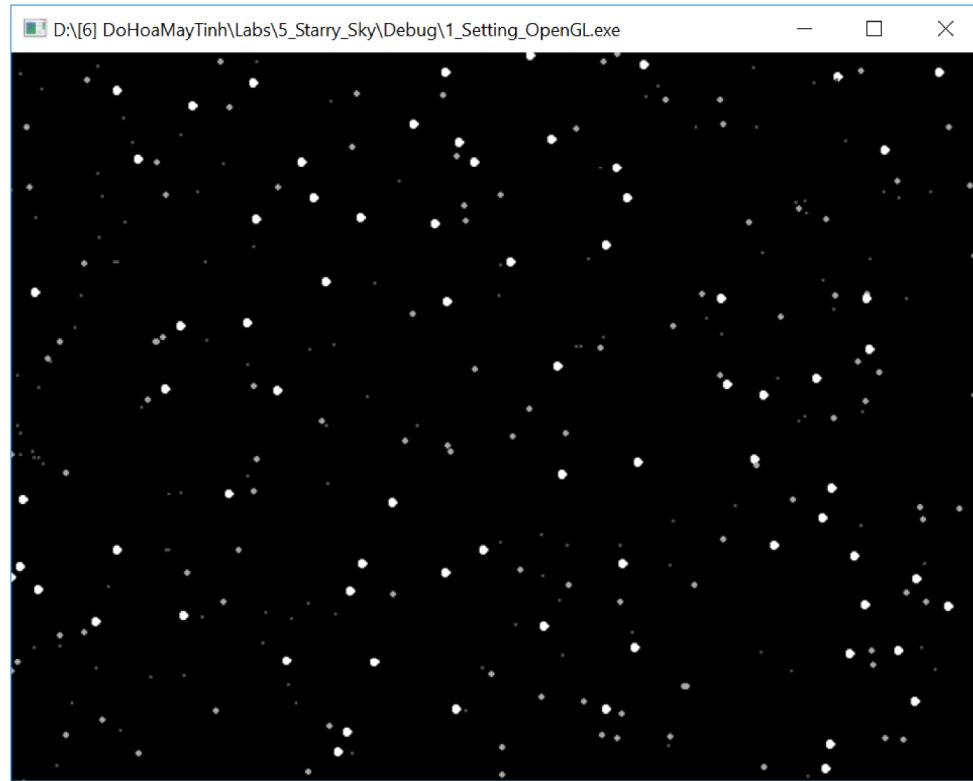
```

        blue -= 0.3f;
    }
    glFlush();
}

```

**Bài 12.** Sử dụng hàm glCircle đã cài đặt ở Bài trên, hãy viết ứng dụng đồ họa sử dụng OpenGL mô phỏng hoạt cảnh chuyển động ngang của một bầu trời sao với các yêu cầu sau:

- Tổng số lượng các ngôi sao trên bầu trời là 300.
- Mỗi một ngôi sao ban đầu được phát sinh với các đặc điểm sau:
  - Tọa độ (x, y) của các ngôi sao được phát sinh ngẫu nhiên trong phạm vi kích thước của cửa sổ OpenGL.
  - Bán kính r của các ngôi sao được phát sinh ngẫu nhiên trong khoảng từ 1 đến 3 pixel.
  - Cường độ sáng i của ngôi sao phụ thuộc vào bán kính của ngôi sao đó theo tỷ lệ  $i = r/3$ .
  - Vận tốc di chuyển ngang của ngôi sao thứ i, ký hiệu  $\Delta v_i$ , được tính dựa trên công thức:  $\Delta v_i = 2r_i + v$ . Với v là một giá trị ngẫu nhiên trong khoảng từ 1 đến 3.
- Tại bước lặp thứ k, các ngôi sao liên tục di chuyển ngang theo chiều từ trái sang phải dựa trên vận tốc  $\Delta v_i$  của chúng:  $x_i^k = x_i^{k-1} + \Delta v_i$ .
- Nếu ngôi sao thứ i di chuyển vượt quá chiều rộng cửa sổ, ta phát sinh lại ngẫu nhiên ngôi sao này theo các yêu cầu ở trên, ngoại trừ tọa độ  $x_i$  của ngôi sao này bắt đầu từ cạnh trái của cửa sổ:  $x_i = 0$ , tạo cảm giác một ngôi sao mới di chuyển ngang cửa sổ.



Hướng dẫn:

```
#include <math.h>
#include "Dependencies\glew\glew.h"
#include "Dependencies\freeglut\freeglut.h"

#define STEPS      40           // number of fans per circle
#define MAX_STARS  300          // total stars supported
#define PI 3.14159265358979323846

const int screenWidth = 640;
const int screenHeight = 480;

typedef struct star           // structure of a star
{
    GLint x, y;
    GLint radius;
    GLint velocity;
    GLfloat intensity;
} STAR;

STAR sky[MAX_STARS];           // our starfield

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0); //clear black
}

void glCircle(GLint x, GLint y, GLint radius)
{
    GLfloat twicePi = (GLfloat) 2.0f * PI;
```

```

glBegin(GL_TRIANGLE_FAN);
    glVertex2i(x, y); // circle center
    for (int i = 0; i <= STEPS; i++) // draw a triangle fan
    {
        glVertex2i((GLint)(x + (radius * cos(i * twicePi / STEPS)) +
0.5),
                    (GLint)(y + (radius * sin(i * twicePi / STEPS)) + 0.5));
    }
    glEnd();
}

void skyInit()
{
    for (int i = 0; i < MAX_STARS; i++)
    {
        sky[i].x = rand() % screenWidth;
        sky[i].y = rand() % screenHeight;
        sky[i].radius = 1 + rand() % 3;
        sky[i].intensity = sky[i].radius / 3.0f;
        sky[i].velocity = sky[i].radius * 2 + rand() % 3;
    }
}

void skyDraw() // put your drawing code here
{
    glClearColor(0, 0, 0, 0); // clear color buffer to black
    glClear(GL_COLOR_BUFFER_BIT);

    for (int i = 0; i < MAX_STARS; i++)
    {
        glColor3f(sky[i].intensity, sky[i].intensity,
sky[i].intensity);
        glCircle(sky[i].x, sky[i].y, sky[i].radius);

    }
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT); //Clears the color buffer
    glColor3f(1.0, 1.0, 1.0);

    glPushMatrix();
    skyDraw();
    glPopMatrix();
    glutSwapBuffers();
}

void update()
{
    for (int i = 0; i < MAX_STARS; i++)
    {
        sky[i].x += sky[i].velocity;
        if (sky[i].x >= screenWidth)
        {
            sky[i].x = 0;
            sky[i].y = rand() % screenHeight;
            sky[i].radius = 1 + rand() % 3;
            sky[i].intensity = sky[i].radius / 3.0f;
            sky[i].velocity = sky[i].radius * 2 + rand() % 3;
        }
    }
}

```

```

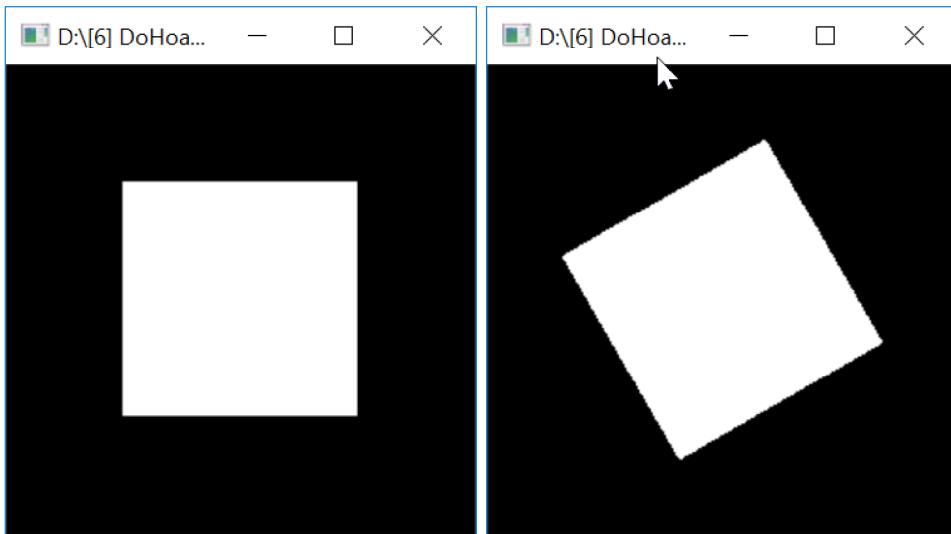
        Sleep(50);
        glutPostRedisplay();
    }

    void reshape(int w, int h)
    {
        glViewport(0, 0, (GLsizei)w, (GLsizei)h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
    }

    int main(int argc, char **argv)
    {
        glutInit(&argc, argv); // Khởi tạo glut
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA); // Khởi tạo chế độ vẽ
        single buffer và hệ màu RGB
        glutInitWindowSize(screenWidth, screenHeight); //optional
        glutInitWindowPosition(100, 100); //optional
        glutCreateWindow(argv[0]);
        init();
        skyInit();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutIdleFunc(update);
        glutMainLoop();
        return 0;
    }
}

```

### Bài 13. Vẽ hình chữ nhật quay quanh tâm



Khi bấm chuột trái hình chữ nhật sẽ quay quanh tâm, bấm chuột giữa sẽ dừng lại.

Hướng dẫn:

```
#include "Dependencies\glew\glew.h"
```

```
#include "Dependencies\freeglut\freeglut.h"

const int screenWidth = 250;
const int screenHeight = 250;

static GLfloat spin = 0.0;

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glRotatef(spin, 0.0, 0.0, 1.0);
    glColor3f(1.0, 1.0, 1.0);
    glRectf(-25.0, -25.0, 25.0, 25.0);
    glPopMatrix();
    glutSwapBuffers();
}

void spinDisplay(void)
{
    spin = spin + 2.0;
    if (spin > 360.0)
        spin = spin - 360.0;
    glutPostRedisplay();
}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void mouse(int button, int state, int x, int y)
{
    switch (button) {
    case GLUT_LEFT_BUTTON:
        if (state == GLUT_DOWN)
            glutIdleFunc(spinDisplay);
        break;
    case GLUT_MIDDLE_BUTTON:
        if (state == GLUT_DOWN)
            glutIdleFunc(NULL);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
```

```

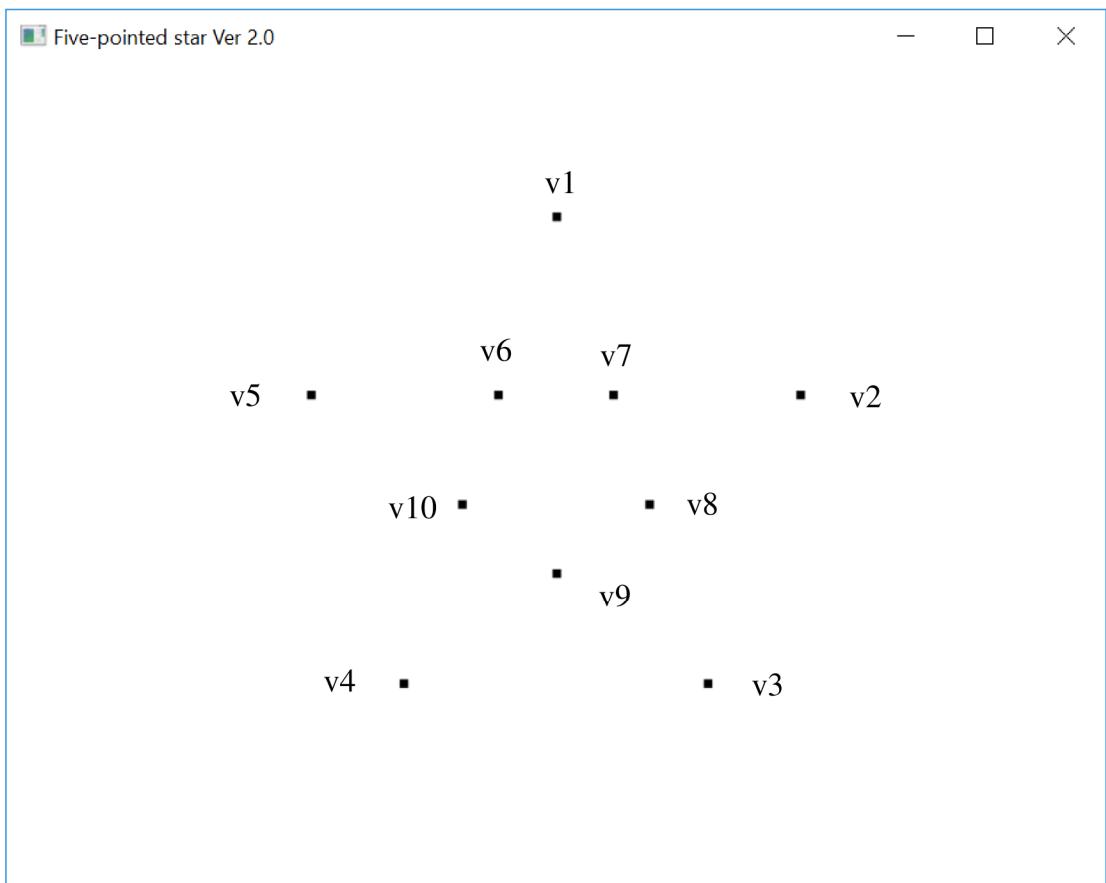
glutInitWindowSize(screenWidth, screenWidth);
glutInitWindowPosition(100, 100);
glutCreateWindow(argv[0]);
init();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMouseFunc(mouse);
glutMainLoop();
return 0;

}

```

### 3. Bài tập tự giải

**Bài 14.** Cải tiến bài tập 9 để vẽ hình ngôi sao như sau:



Hướng dẫn:

Trong bài tập 9 ta đã tính được tọa độ của các đỉnh  $v_1, v_2, v_3, v_4, v_5$ . Tương tự chúng ta tìm tọa độ của các đỉnh  $v_6, v_7, v_8, v_9, v_{10}$ .

**Bài 15.** Chính sửa bài 12 để đảo ngược chuyển động các ngôi sao theo chiều từ phải sang trái

Gợi ý:

- Ta giảm tọa độ x của các ngôi sao một lượng là  $\Delta v$ . Khi đó công thức tính vị trí của các ngôi sao ở mỗi lần lặp trở thành:  $x_i^k = x_i^{k-1} - \Delta v_i$ .
- Xét ngôi sao thứ i, sau khi di chuyển hết cạnh trái của cửa sổ, khi đó  $x_i < 0$ . Ta phát sinh lại ngôi sao này với  $x_i = SCREENWIDTH$ , tức là ở cạnh phải của cửa sổ OpenGL.

#### 4. Tài liệu tham khảo

[1] <http://www.glprogramming.com/red/>