

U18CO018
Shubham Shekhaliya
Computer Graphics
Assignment 5

1. Write a program to draw the following shapes:

Triangles (triples of vertices interpreted as triangles)

Triangle Strip (linked strip of triangles)

Triangle Fan (linked fan of triangles)

Quads (quadruples of vertices interpreted as four sided polygons)

Quad Strip (linked strip of quadrilaterals)

Polygon (boundary of a simple, convex polygon)

Code:

```
#include<windows.h>
#include<stdio.h>
#include<GL/glut.h>
#include<math.h>

void init() {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(0.0, 0.0, 1.0);
    glPointSize(7.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-780, 780, -420, 420);
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glEnable(GL_POINT_SMOOTH);
    // Triangles
    glBegin(GL_TRIANGLES);
    glVertex2i(-600,0+200);
    glVertex2i(-600,50+200);
    glVertex2i(-600+100,0+200);
    glVertex2i(-600+120,10+200);
    glVertex2i(-600+15,70+200);
    glVertex2i(-600+225,70+200);
    glEnd();

    // Triangle Strip
    glBegin(GL_TRIANGLE_STRIP);
    glVertex2i(0-100,100+180);
    glVertex2i(0-100,0+180);
```

```
glVertex2i(100-100,100+180);
glVertex2i(150-100,0+180);
glVertex2i(180-100,100+180);
glVertex2i(200-100,-20+180);
glEnd();
```

```
// Triangle FAN
glBegin(GL_TRIANGLE_FAN);
glVertex2i(0+450,0+180);
glVertex2i(0+450,100+180);
glVertex2i(70+450,85+180);
glVertex2i(100+450,35+180);
glVertex2i(100+450,-10+180);
glEnd();
```

```
glBegin(GL_QUADS);
glVertex2i(0-600,0-180);
glVertex2i(20-600,40-180);
glVertex2i(100-600,50-180);
glVertex2i(110-600,-10-180);
glVertex2i(130-600,-20-180);
glVertex2i(140-600,35-180);
glVertex2i(210-600,55-180);
glVertex2i(170-600,-20-180);
glEnd();
```

```
glBegin(GL_QUAD_STRIP);
glVertex2i(-10-100,50-180);
glVertex2i(0-100,0-180);
glVertex2i(60-100,55-180);
glVertex2i(60-100,0-180);
glVertex2i(90-100,60-180);
glVertex2i(110-100,15-180);
glVertex2i(150-100,65-180);
glVertex2i(155-100,20-180);
glEnd();
```

```
glBegin(GL_POLYGON);
glVertex2i(0+450,0-150);
glVertex2i(150+450,20-150);
glVertex2i(120+450,-25-150);
glVertex2i(-10+450,-60-150);
glVertex2i(-15+450,-30-150);
```

```
glEnd();
```

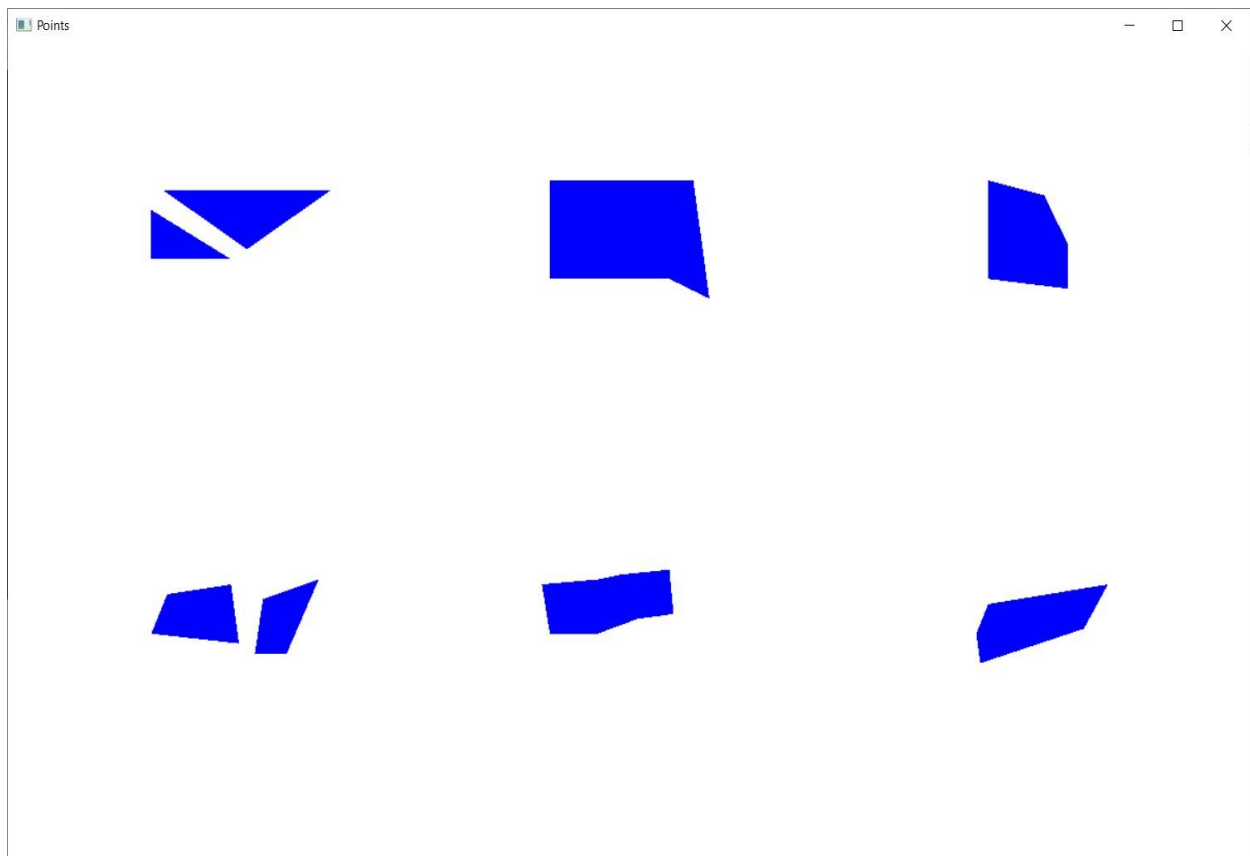
```
glFlush();
```

```

}
int main (int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1200, 750);
    glutInitWindowPosition(100, 0);
    glutCreateWindow("Points");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}

```

Output:



2. Write a menu driven program for following algorithms:

Mid Point Circle generating algorithm

Mid Point Ellipse generating algorithm

Code:

```

#include <graphics.h>
#include <iostream>

```

```

using namespace std;

void midPointCircleDraw(float a, float b, float r) {
    float x = r, y = 0;
    putpixel(x + a, y + b, WHITE);

    if (r > 0) {
        putpixel(x + a, -y + b, WHITE);
        putpixel(y + a, x + b, WHITE);
        putpixel(-y + a, x + b, WHITE);
    }
    float P = 1 - r;

    while (x > y) {
        y++;
        if (P <= 0)
            P = P + 2 * y + 1;
        else {
            x--;
            P = P + 2 * y - 2 * x + 1;
        }

        if (x < y)
            break;

        putpixel(x + a, y + b, WHITE);
        putpixel(-x + a, y + b, WHITE);
        putpixel(x + a, -y + b, WHITE);
        putpixel(-x + a, -y + b, WHITE);

        if (x != y) {
            putpixel(y + a, x + b, WHITE);
            putpixel(-y + a, x + b, WHITE);
            putpixel(y + a, -x + b, WHITE);
            putpixel(-y + a, -x + b, WHITE);
        }
    }
}

void midPointEllipseDraw(int xc, int yc, int rx, int ry) {
    float dx, dy, d1, d2, x, y;
    x = 0;
    y = ry;

    d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;

```

```

while (dx < dy) {
    putpixel(x + xc, y + yc, WHITE);
    putpixel(-x + xc, y + yc, WHITE);
    putpixel(x + xc, -y + yc, WHITE);
    putpixel(-x + xc, -y + yc, WHITE);

    if (d1 < 0) {
        x++;
        dx = dx + (2 * ry * ry);
        d1 = d1 + dx + (ry * ry);
    }
    else {
        x++;
        y--;
        dx = dx + (2 * ry * ry);
        dy = dy - (2 * rx * rx);
        d1 = d1 + dx - dy + (ry * ry);
    }
}

d2 = ((ry * ry) * ((x + 0.5) * (x + 0.5))) + ((rx * rx) * ((y - 1) * (y - 1))) -
(rx * rx * ry * ry);

while (y >= 0) {
    putpixel(x + xc, y + yc, WHITE);
    putpixel(-x + xc, y + yc, WHITE);
    putpixel(x + xc, -y + yc, WHITE);
    putpixel(-x + xc, -y + yc, WHITE);
    if (d2 > 0) {
        y--;
        dy = dy - (2 * rx * rx);
        d2 = d2 + (rx * rx) - dy;
    }
    else {
        y--;
        x++;
        dx = dx + (2 * ry * ry);
        dy = dy - (2 * rx * rx);
        d2 = d2 + dx - dy + (rx * rx);
    }
}
}

int main() {
    int gdriver = DETECT, gmode;
    initgraph(&gdriver, &gmode, " ");

    while (true) {

```

```

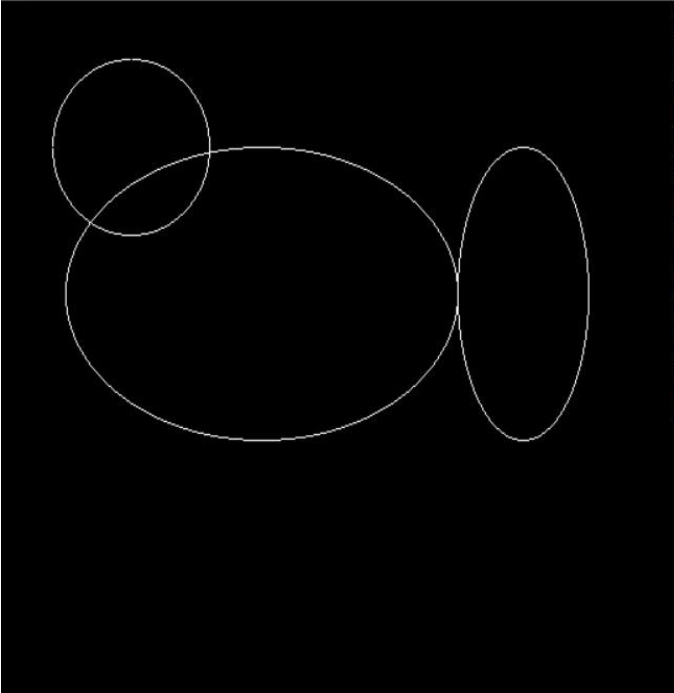
cout << "1 Mid Point Circle generating algorithm\n";
cout << "2 Mid Point Ellipse generating algorithm\n";
cout << "3 Exit\n";

int p = -1;
cin >> p;

if (p == 1) {
    cout << "Enter center x , y and radius space separated\n";
    int x0, y0, r;
    cin >> x0 >> y0 >> r;
    midPointCircleDraw(x0, y0, r);
    getch();
}
else if (p == 2) {
    cout << "Enter center x , y , xradius and yradius space separated\n";
    int x0, y0, xr, yr;
    cin >> x0 >> y0 >> xr >> yr;
    midPointEllipseDraw(x0, y0, xr, yr);
    getch();
}
else {
    break;
}
}
return 0;
}

```

Output:



```

E:\Asem6\cg\practical\main.exe
1 Mid Point Circle generating algorithm
2 Mid Point Ellipse generating algorithm
3 Exit
1
Enter center x , y and radius space separated
100 100 60
1 Mid Point Circle generating algorithm
2 Mid Point Ellipse generating algorithm
3 Exit
2
Enter center x , y , xradius and yradius space separated
200 200 150 100
1 Mid Point Circle generating algorithm
2 Mid Point Ellipse generating algorithm
3 Exit
2
Enter center x , y , xradius and yradius space separated
400 200 50 100

```

3. Write a program to generate the following figure

Code:

```
#include <bits/stdc++.h>
#include <graphics.h>
using namespace std;

void dda(float x0, float y0, float x1, float y1) {
    float i;
    float x, y, dx, dy, steps;
    dx = (float)(x1 - x0);
    dy = (float)(y1 - y0);
    if (dx >= dy) {
        steps = dx;
    }
    else {
        steps = dy;
    }
    dx = dx / steps;
    dy = dy / steps;
    x = x0;
    y = y0;
    i = 1;
    while (i <= steps) {
        putpixel(x, y, WHITE);
        x += dx;
        y += dy;
        i = i + 1;
    }
}

void draw(float x, float y, int f, float cx, float cy, float xx, float yy) {
    float v1x = xx - cx, v1y = yy - cy;
    float v2x = x - cx, v2y = y - cy;

    float angle = atan2(v1x * v2y - v2x * v1y, v1x * v2x + v1y * v2y);

    if (f == 0) { // down
        if (angle > 0)
        {
            putpixel(x, y, WHITE);
        }
    }
    else { // up
        if (angle < 0)
            putpixel(x, y, WHITE);
    }
}
```

```

}

void halfCircle(float a, float b, int f, float r, float xx, float yy) {
    float x = r, y = 0;
    putpixel(x + a, y + b, WHITE);

    if (r > 0) {
        putpixel(x + a, -y + b, WHITE);
        putpixel(y + a, x + b, WHITE);
        putpixel(-y + a, x + b, WHITE);
    }
    float P = 1 - r;

    while (x > y) {
        y++;
        if (P <= 0)
            P = P + 2 * y + 1;
        else {
            x--;
            P = P + 2 * y - 2 * x + 1;
        }

        if (x < y)
            break;

        draw(x + a, y + b, f, a, b, xx, yy);
        draw(-x + a, y + b, f, a, b, xx, yy);
        draw(x + a, -y + b, f, a, b, xx, yy);
        draw(-x + a, -y + b, f, a, b, xx, yy);

        if (x != y) {
            draw(y + a, x + b, f, a, b, xx, yy);
            draw(-y + a, x + b, f, a, b, xx, yy);
            draw(y + a, -x + b, f, a, b, xx, yy);
            draw(-y + a, -x + b, f, a, b, xx, yy);
        }
    }
}

int main() {
    int gdriver = DETECT, gmode;
    int midx, midy, i;

    initgraph(&gdriver, &gmode, "");
    cout << "Enter coordinates of the Point A and B\n";
    int x0, y0, x1, y1;
    cin >> x0 >> y0 >> x1 >> y1;
}

```



```

dda(x0, y0, x1, y1);

float mx = (x0 + x1) / 2, my = (y0 + y1) / 2;

float m1x = (mx + x0) / 2, m1y = (my + y0) / 2;
float r1 = sqrt(abs(mx - m1x) * abs(mx - m1x) + abs(my - m1y) * abs(my - m1y));
float m2x = (mx + x1) / 2, m2y = (my + y1) / 2;
float r2 = sqrt(abs(mx - m2x) * abs(mx - m2x) + abs(my - m2y) * abs(my - m2y));

halfCircle(m1x, m1y, 0, r1, mx, my);
halfCircle(m2x, m2y, 1, r2, x1, y1);

getch();
return 0;
}

```

Output:

