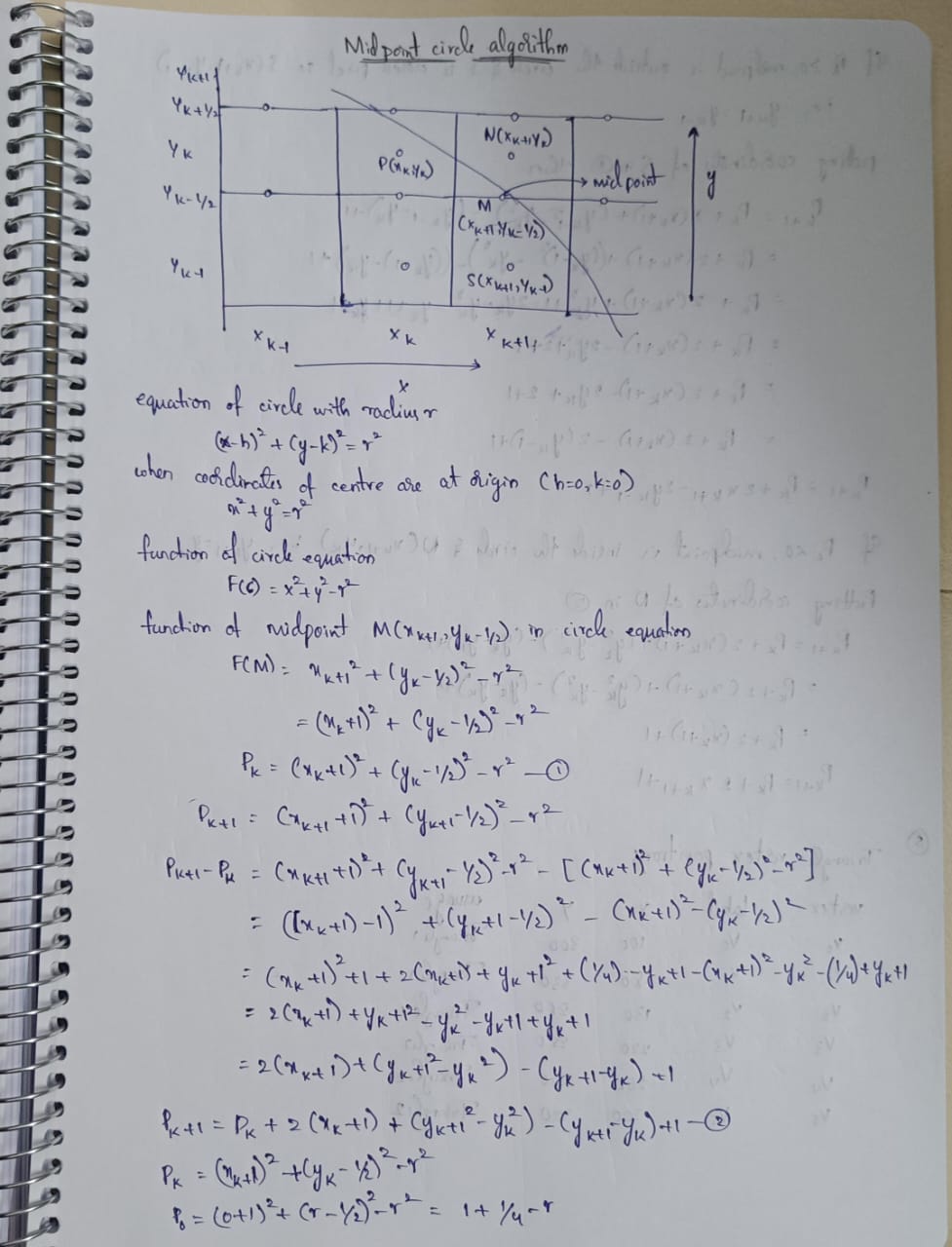
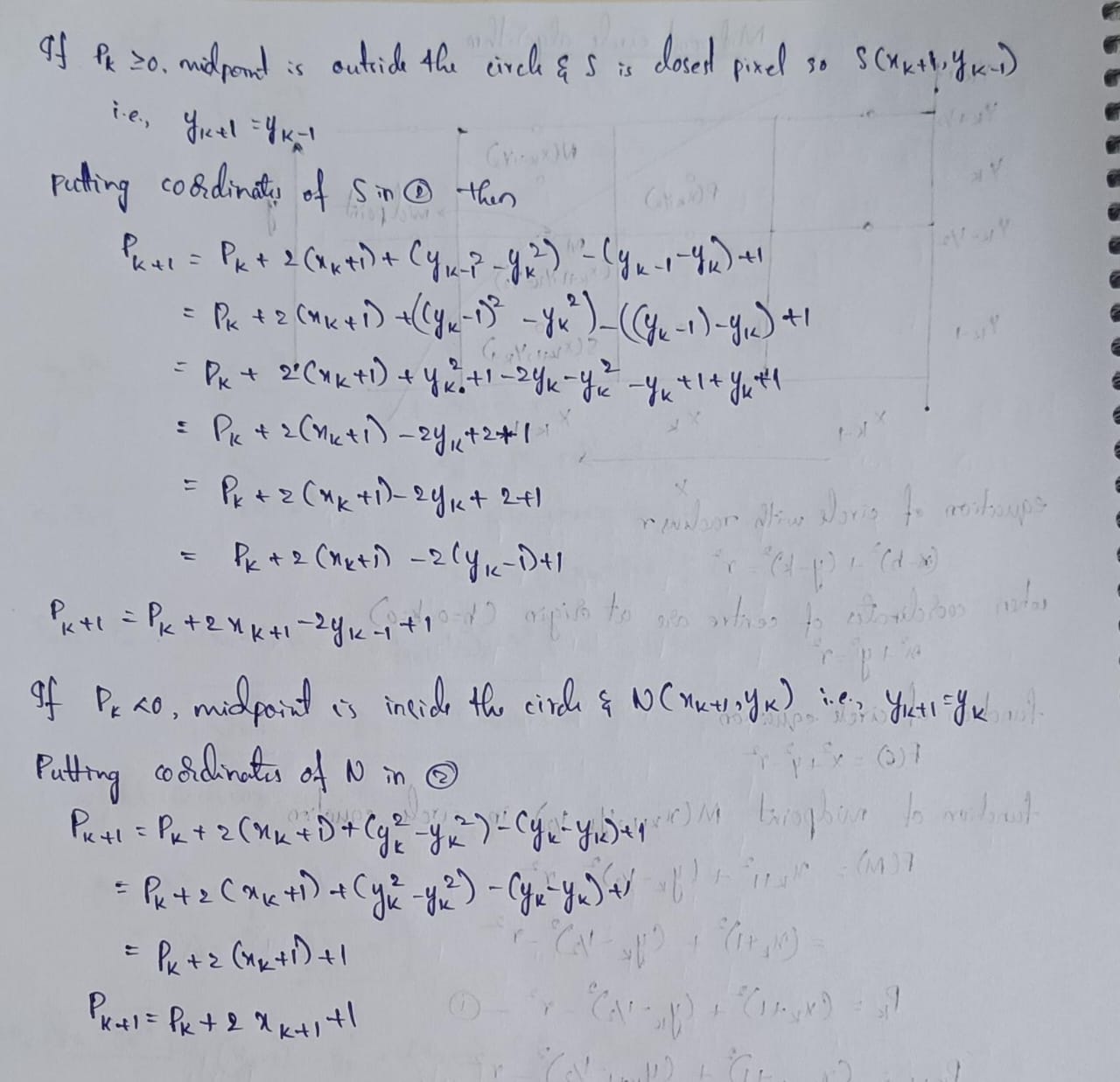
**19CSE433 – COMPUTER GRAPHICS AND VISUALIZATION – LAB3**

**CB.EN.U4CSE22328 K.Bhavya Naga Sai**

**q1) Derive the equations for the Midpoint circle algorithm.**

****



**2) Lineloop.c**

#include <windows.h>

#include <GL/glut.h>

#include <stdio.h>

#include <math.h>

#include <string.h>

// Vertex positions (v0 to v5)

int x[] = {100, 180, 240, 220, 140, 100};

int y[] = {300, 330, 270, 200, 180, 240};

// Colors for labels (matching the example image)

float labelColors[6][3] = {

{1, 0, 0}, // v0 - red

{0, 1, 0}, // v1 - green

{0, 0, 1}, // v2 - blue

{0, 1, 1}, // v3 - cyan

{1, 0, 1}, // v4 - magenta

{1, 1, 0.5} // v5 - yellowish

};

// Draw pixel

void setPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

// Midpoint Circle Algorithm

void midpointCircle(int xc, int yc, int r) {

int x = 0, y = r;

int p = 1 - r;

while (x <= y) {

setPixel(xc + x, yc + y);

setPixel(xc - x, yc + y);

setPixel(xc + x, yc - y);

setPixel(xc - x, yc - y);

setPixel(xc + y, yc + x);

setPixel(xc - y, yc + x);

setPixel(xc + y, yc - x);

setPixel(xc - y, yc - x);

x++;

if (p < 0)

p += 2 \* x + 1;

else {

y--;

p += 2 \* (x - y) + 1;

}

}

}

// Bresenham Line Drawing Algorithm

void drawLine(int x0, int y0, int x1, int y1) {

int dx = abs(x1 - x0), dy = abs(y1 - y0);

int sx = x0 < x1 ? 1 : -1;

int sy = y0 < y1 ? 1 : -1;

int err = dx - dy;

while (1) {

setPixel(x0, y0);

if (x0 == x1 && y0 == y1) break;

int e2 = 2 \* err;

if (e2 > -dy) { err -= dy; x0 += sx; }

if (e2 < dx) { err += dx; y0 += sy; }

}

}

// Draw vertex label

void drawLabel(int x, int y, const char \*label, float r, float g, float b) {

glColor3f(r, g, b);

glRasterPos2i(x - 6, y - 4);

for (int i = 0; i < strlen(label); i++) {

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, label[i]);

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw white LineLoop (v0→v1→...→v5→v0)

glColor3f(1, 1, 1);

for (int i = 0; i < 6; i++) {

int j = (i + 1) % 6;

drawLine(x[i], y[i], x[j], y[j]);

}

// Draw white circles and colored vertex labels

for (int i = 0; i < 6; i++) {

glColor3f(1, 1, 1); // white circle

midpointCircle(x[i], y[i], 12);

char label[4];

sprintf(label, "v%d", i);

drawLabel(x[i], y[i], label, labelColors[i][0], labelColors[i][1], labelColors[i][2]);

}

glFlush();

}

void init() {

glClearColor(0, 0, 0, 1); // Black bg

glPointSize(2.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 400, 0, 400);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutInitWindowPosition(100, 100);

glutCreateWindow("LineLoop with Midpoint Circle & Colored Labels");

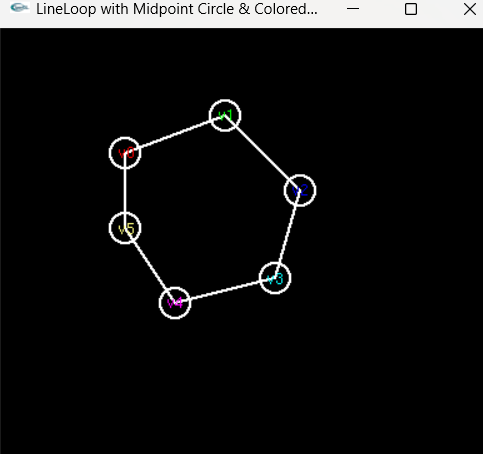
init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}  
Output:



**Triangle.c:**

#include <windows.h>

#include <GL/glut.h>

#include <stdio.h>

#include <math.h>

#include <string.h>

// Vertex coordinates

int x[] = {100, 180, 130, 220, 260, 200};

int y[] = {300, 330, 250, 250, 220, 200};

// Circle radius

int radius = 14;

// Set a pixel

void setPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

// Midpoint Circle Drawing Algorithm

void midpointCircle(int xc, int yc, int r) {

int x = 0, y = r;

int p = 1 - r;

while (x <= y) {

// 8 symmetric points

setPixel(xc + x, yc + y);

setPixel(xc - x, yc + y);

setPixel(xc + x, yc - y);

setPixel(xc - x, yc - y);

setPixel(xc + y, yc + x);

setPixel(xc - y, yc + x);

setPixel(xc + y, yc - x);

setPixel(xc - y, yc - x);

x++;

if (p < 0)

p += 2 \* x + 1;

else {

y--;

p += 2 \* (x - y) + 1;

}

}

}

// Bresenham's Line Drawing Algorithm

void drawLine(int x0, int y0, int x1, int y1) {

int dx = abs(x1 - x0), dy = abs(y1 - y0);

int sx = (x0 < x1) ? 1 : -1;

int sy = (y0 < y1) ? 1 : -1;

int err = dx - dy;

while (1) {

setPixel(x0, y0);

if (x0 == x1 && y0 == y1) break;

int e2 = 2 \* err;

if (e2 > -dy) { err -= dy; x0 += sx; }

if (e2 < dx) { err += dx; y0 += sy; }

}

}

// Draw labels near each circle

void drawLabel(int x, int y, const char \*label) {

glColor3f(1, 1, 1); // White

glRasterPos2i(x - 6, y - 4);

for (int i = 0; i < strlen(label); i++) {

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, label[i]);

}

}

// Main drawing function

void drawTrianglesOutline() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw triangle 1: v0 - v1 - v2

glColor3f(1, 1, 1);

drawLine(x[0], y[0], x[1], y[1]);

drawLine(x[1], y[1], x[2], y[2]);

drawLine(x[2], y[2], x[0], y[0]);

// Draw triangle 2: v3 - v4 - v5

drawLine(x[3], y[3], x[4], y[4]);

drawLine(x[4], y[4], x[5], y[5]);

drawLine(x[5], y[5], x[3], y[3]);

// Colors for circle vertices

float colors[6][3] = {

{1.0, 0.0, 0.0}, // v0 - red

{0.0, 1.0, 0.0}, // v1 - green

{0.0, 0.0, 1.0}, // v2 - blue

{0.0, 1.0, 1.0}, // v3 - cyan

{1.0, 0.0, 1.0}, // v4 - magenta

{1.0, 1.0, 0.6} // v5 - yellowish

};

for (int i = 0; i < 6; i++) {

glColor3f(colors[i][0], colors[i][1], colors[i][2]);

midpointCircle(x[i], y[i], radius);

char label[4];

sprintf(label, "v%d", i);

drawLabel(x[i], y[i], label);

}

glFlush();

}

void init() {

glClearColor(0, 0, 0, 1); // Black background

glColor3f(1, 1, 1); // White lines

glPointSize(2.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 400, 0, 400);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutInitWindowPosition(100, 100);

glutCreateWindow("Triangles with Circles and Labels");

init();

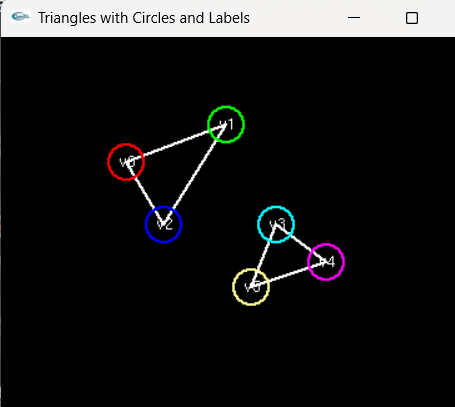
glutDisplayFunc(drawTrianglesOutline);

glutMainLoop();

return 0;

}

**Output:**



**3) Tabulate the pixel positions generated to create the selected structures in (Q No 2)**

