RNSIT-CSE Bengaluru

Computer Graphics Laboratory 18CSL67
Extra Programs

1.DDA Line Drawing Technique

```
#include <iostream>
#include <glut.h>
#include <math.h>
using namespace std;
int xx, yy, xend, yend;
void myinit() {
     //glClearColor(2.0, 2.0, 2.0, 4.0);
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluOrtho2D(0, 500, 0, 500);
     glMatrixMode(GL_MODELVIEW);
}
void setPixel(int x, int y)
{
     glBegin(GL_POINTS);
     glVertex2f(x, y);
     glEnd();
     glFlush();
}
void lineDDA(int x0,int y0,int xend,int yend)
{
     int dx =xend-x0, dy = yend-y0,steps,k;
     float xIncrement, yIncrement, x = x0, y = y0;
     glColor3f(1, 0, 0);
     glPointSize(3);
     if (fabs(dx) > fabs(dy))
     {
          steps = fabs(dx);
     }
     else {
          steps = fabs(dy);
     }
     xIncrement = float(dx) / float(steps);
     yIncrement = float(dy) / float(steps);

     setPixel(round(x), round(y));
     for (k = 0; k < steps; k++) {
          x = x + xIncrement;
          y = y + yIncrement;
          setPixel(round(x), round(y));
     }
```

```
}
void display()
{
     glClearColor(1, 1, 1, 1);
     glClear(GL_COLOR_BUFFER_BIT);
     lineDDA(xx,yy,xend,yend);
     glFlush();
}
int main() {
     //glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
     glutInitWindowSize(500, 500);
     glutInitWindowPosition(100, 100);
     glutCreateWindow("dda line");
     myinit();
     cout << "enter co-ordinates of first point: ";
     cin >> xx >> yy;
     cout << "enter co-ordinates of second point: ";
     cin >> xend >> yend;
     glutDisplayFunc(display);
     glutMainLoop();
     return 0;
}
```

2.Mid-Point Circle Generation

```
#include <gl/glut.h>
#include <iostream>
using namespace std;
void myinit()
{
     glClearColor(0.0, 0.0, 0.0, 0.0);
     glClear(GL_COLOR_BUFFER_BIT);
     gluOrtho2D(-500,500,-500,500);
     glMatrixMode(GL_PROJECTION);
}
void Draw() {
     glClear(GL_COLOR_BUFFER_BIT);
     glColor3d(1, 0, 0);
     glBegin(GL_TRIANGLE_FAN);
     glVertex2f(0.0, 0.0);
     glVertex2f(0.0, 30.0);
     glVertex2f(20.0, 30.0);
     glVertex2f(30.0, 20.0);
     glVertex2f(30.50, 0.50);
     glVertex2f(30.0, -10.0);
     glVertex2f(20.0, -20.0);
     glVertex2f(0.0, -20.50);
     glVertex2f(-20.0, -20.0);
     glVertex2f(-30.0, -10.0);
     glEnd();
     glFlush();
```

```
}
void circle()
{
	glColor3f(1.0, 0.0, 0.0);
	glPointSize(2.0);
	float r = 100;
	float x = 0, y = r;
	float p = 1 - r;
	glBegin(GL_POINTS);
	while (x != y)
	{
		x++;
		if (p < 0) {
			p += 2 * (x + 1) + 1;
		}
		else {
			y--;
			p += 2 * (x + 1) + 1 - 2 * (y - 1);
		}
		glVertex2i(x, y);
		glVertex2i(-x, y);
		glVertex2i(x, -y);
		glVertex2i(-x, -y);
		glVertex2i(y, x);
		glVertex2i(-y, x);
		glVertex2i(y, -x);
		glVertex2i(-y, -x);

	}
	glEnd();
	glFlush();
}
int main(int argc, char** argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(100, 100);
	glutCreateWindow("Line Draw OpenGL");
	glutDisplayFunc(Draw);
	myinit();
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(500, 500);
	glutCreateWindow("Draw OpenGL");
	glutDisplayFunc(circle);
	myinit();
	glutMainLoop();
	return 0;
}
```

3. Stroked Circle

```
#include<stdio.h>
#include<math.h>
#include<gl/glut.h>
```

```
void myinit()
{
     glMatrixMode(GL_PROJECTION_MATRIX);
     glLoadIdentity();
     gluOrtho2D(-50, 50, -50, 50);
     glMatrixMode(GL_MODELVIEW);
}

void display()
{
     glClearColor(1, 1, 1, 1);
     glClear(GL_COLOR_BUFFER_BIT);

     int i;
     float x1, x2, y1, y2, r1 = 15, r2 = 18, t;

     glColor3f(1, 0, 0);
     glBegin(GL_QUAD_STRIP);
     for (i = 0; i <= 24; i++)
     {
          t = 3.142 / 12 * i;
          x1 = r1 * cos(t);
          y1 = r1 * sin(t);
          x2 = r2 * cos(t);
          y2 = r2 * sin(t);
          glVertex2f(x1, y1);
          glVertex2f(x2, y2);
     }
     glEnd();

     glFlush();
}


void main()
{
     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
     glutInitWindowSize(500, 500);
     glutInitWindowPosition(300, 150);

     glutCreateWindow("Stroked O");
     myinit();
     glutDisplayFunc(display);
     glutMainLoop();
}
```

4. Raster Text Display

```
#include<gl/glut.h>
#include<iostream>
using namespace std;
```

```
char str[40];
int cx1 = 50, cy1 = 100, cx2 = 20, cy2 = 180, d = 10;

void myinit()
{
     glMatrixMode(GL_PROJECTION_MATRIX);
     glLoadIdentity();
     gluOrtho2D(0, 200, 0, 200);
     glMatrixMode(GL_MODELVIEW);
}

void display()
{
     glClearColor(1, 1, 1, 1);
     glClear(GL_COLOR_BUFFER_BIT);
     glColor3f(1, 0, 0);
     glRasterPos2i(cx1, cy1);
     int i;
     glRasterPos2i(cx1, cy1);
     for (i = 0; i < strlen(str); i++)
     {
       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, str[i]);
     }

     glColor3f(0, 0, 1);
     for (i = 0; i < strlen(str); i++)
     {
        glRasterPos2i(cx2, cy2 - d * i);
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);
     }

     glFlush();
}
void main()
{
     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
     glutInitWindowSize(500, 500);
     glutInitWindowPosition(300, 150);
     cout<<"Enter the string:\n";
     cin>>str;
     glutCreateWindow("Text");
     myinit();
     glutDisplayFunc(display);
     glutMainLoop();
}
```

5. 2D Gasket

```
#include<stdlib.h>
#include<GL/glut.h>
float v[3][2] = { {-25,-25},{0,25},{25,-25} };
void myinit()
{
     glMatrixMode(GL_PROJECTION_MATRIX);
```

```
	glLoadIdentity();
	gluOrtho2D(-50, 50, -50, 50);
	glMatrixMode(GL_MODELVIEW);
}
void display()
{
	glClearColor(1, 1, 1, 1);
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_LINE_LOOP);
	glVertex2fv(v[0]);
	glVertex2fv(v[1]);
	glVertex2fv(v[2]);
	glEnd();
	float p[2] = { 0,0 };
	int i, n = 5000, j;
	glPointSize(2);
	for (i = 0; i < n; i++)
	{
		j = rand() % 3;
		if (j == 0)
			glColor3f(1, 0, 0);
		else if (j == 1)
			glColor3f(0, 1, 0);
		else
			glColor3f(0, 0, 1);
		p[0] = (p[0] + v[j][0]) / 2;
		p[1] = (p[1] + v[j][1]) / 2;
		glBegin(GL_POINTS);
		glVertex2fv(p);
		glEnd();
		glFlush();
	}
}
void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(300, 150);
	glutCreateWindow("2D Sierpinkski Gasket");
	myinit();
	glutDisplayFunc(display);
	glutMainLoop();
}
```

6. Reshape Function

```
#include<stdio.h>
#include<gl/glut.h>
void myinit()
{
glMatrixMode(GL_PROJECTION_MATRIX);
glLoadIdentity();
gluOrtho2D(-100,100,-100,100);
glMatrixMode(GL_MODELVIEW);
```

```
}
void display()
{
glClearColor(0,0,0,1);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1,0,0);
glBegin(GL_POLYGON);
glVertex2f(-50,-50);
glVertex2f(-50,50);
glVertex2f(50,50);
glVertex2f(50,-50);
glEnd();
glFlush();
}
void reshape(int w,int h)
{
glViewport(0,0,w,h);
glMatrixMode(GL_PROJECTION_MATRIX);
glLoadIdentity();
float t1 = (float)w/(float)h;
float t2 = (float)h/(float)w;
if(w>h)
gluOrtho2D(-100*t1,100*t1,-100,100);
else
gluOrtho2D(-100,100,-100*t2,100*t2);
glMatrixMode(GL_MODELVIEW);
glutPostRedisplay();
}
void main()
{
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(200,200);
glutInitWindowPosition(300,150);
glutCreateWindow("Reshape");
myinit();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
}
```

7. Square shape drawn on Mouse Click

```
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
int wh = 500, ww = 500; float siz = 3;
void myinit()
{
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluOrtho2D(0, wh, 0, ww);// xmin, xmax, ymin, ymax
     glMatrixMode(GL_MODELVIEW);
}
```

```
void drawsq(int x, int y)
{
     y = wh - y;
     glColor3f(0.0, 1.0, 0.0);
     glBegin(GL_POLYGON);
     glVertex2f(x + siz, y + siz);
     glVertex2f(x - siz, y + siz);
     glVertex2f(x - siz, y - siz);
     glVertex2f(x + siz, y - siz);
     glEnd();
     glFlush();
}
void display()
{
     glClearColor(1, 1, 1, 1);
     glClear(GL_COLOR_BUFFER_BIT);
}
void myMouse(int button, int state, int x, int y)
{
     if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
          drawsq(x, y);
     if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
          exit(0);
}
void main(int argc, char** argv)
{
     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
     glutInitWindowSize(wh, ww);
     glutCreateWindow("square");
     glutDisplayFunc(display);
     glutMouseFunc(myMouse);
     myinit();
     glutMainLoop();
}
```

8. Display List Demonstration

```
#include<stdio.h>
#include<gl/glut.h>
#define sq 10
void myinit()
{
glMatrixMode(GL_PROJECTION_MATRIX);
glLoadIdentity();
gluOrtho2D(-100,100,-100,100);
glMatrixMode(GL_MODELVIEW);
}
void display()
{
glClearColor(0,0,0,1);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1,0,0);
glNewList(sq,GL_COMPILE);
```

```
glBegin(GL_POLYGON);
glVertex2f(-50,-50);
glVertex2f(-50,50);
glVertex2f(50,50);
glVertex2f(50,-50);
glEnd();
glEndList();
glCallList(sq);
glFlush();
}
void main()
{
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(600,600);
glutInitWindowPosition(300,150);
glutCreateWindow("Display List");
myinit();
glutDisplayFunc(display);
glutMainLoop();
}
```

9.Rubberband Technique

```
#include<stdio.h>
#include<gl/glut.h>
float xm, ym, xmm, ymm;
int first = 0, w = 600, h = 600;
void init()
{
     glMatrixMode(GL_PROJECTION_MATRIX);
     glLoadIdentity();
     gluOrtho2D(0, w, 0, h);
     glMatrixMode(GL_MODELVIEW);
}
void disp()
{
     glClearColor(0, 0, 0, 1);
     glClear(GL_COLOR_BUFFER_BIT);
     glColor3f(0, 0, 1);
     glLineWidth(5);
     glFlush();
}
void mouse(int b, int s, int x, int y)
{
     glColor3f(0, 0, 1);
     y = h - y;
     if (b == GLUT_LEFT_BUTTON && s == GLUT_DOWN)
     {
          xm = x;
          ym = y;
          first = 0;
     }
     if (b == GLUT_LEFT_BUTTON && s == GLUT_UP)
     {
```

```
            glLogicOp(GL_XOR);
            glBegin(GL_LINES);
            glVertex2f(xm, ym);
            glVertex2f(xmm, ymm);
            glEnd();
            glFlush();
            glLogicOp(GL_COPY);
            glBegin(GL_LINES);
            glVertex2f(xm, ym);
            glVertex2f(xmm, ymm);
            glEnd();
            glFlush();
      }
      if (b == GLUT_RIGHT_BUTTON && s == GLUT_DOWN)
      {
            glClearColor(0, 0, 0, 1);
            glClear(GL_COLOR_BUFFER_BIT);
            glFlush();
      }
      glFlush();
}
void move(int x, int y)
{
      y = h - y;
      if (first == 1)
      {
            glLogicOp(GL_XOR);
            glBegin(GL_LINES);
            glVertex2f(xm, ym);
            glVertex2f(xmm, ymm);
            glEnd();
      }
      xmm = x;
      ymm = y;
      glLogicOp(GL_XOR);
      glBegin(GL_LINES);
      glVertex2f(xm, ym);
      glVertex2f(xmm, ymm);
      glEnd();
      glFlush();
      first = 1;
}
void main()
{
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutInitWindowSize(600, 600);
      glutInitWindowPosition(300, 150);
      glutCreateWindow("Rubberband Technique");
      init();
      glEnable(GL_COLOR_LOGIC_OP);
      glutDisplayFunc(disp);
      glutMouseFunc(mouse);
```

```
     glutMotionFunc(move);
     glutMainLoop();
}
```

10.Menu based Interaction.

```
#include<stdlib.h>
#include<stdio.h>
#include<GL/glut.h>
float x1,x2,x3,x4,y1,y2,y3,y4;
void d_menu(int op)
{
     if(op==1)
          glColor3f(1.0,0.0,0.0);
     else if(op==2)
          glColor3f(0.0,1.0,0.0);
     else if(op==3)
          glColor3f(0.0,0.0,1.0);
     else if(op==4)
          exit(0);
     glutPostRedisplay();
}

void display()
{
     x1=200.0;y1=200.0;x2=100.0;y2=300.0;
     x3=200.0;y3=400.0;x4=300.0;y4=300.0;
     glClear(GL_COLOR_BUFFER_BIT);
     glBegin(GL_LINE_LOOP);
     glVertex2f(x1,y1);
     glVertex2f(x2,y2);
     glVertex2f(x3,y3);
     glVertex2f(x4,y4);
     glEnd();
     glFlush();
}

void myinit()
{
     glClearColor(1.0,1.0,1.0,1.0);
     glColor3f(1.0,0.0,1.0);
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluOrtho2D(0.0,499.0,0.0,499.0);
}

void main(int argc,char **argv)
{
     glutInit(&argc,argv);
     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
     glutInitWindowSize(500,500);
     glutInitWindowPosition(0,0);
     glutCreateWindow("polygon");
     glutCreateMenu(d_menu);
```

```
     glutAddMenuEntry("Red",1);
     glutAddMenuEntry("Green",2);
     glutAddMenuEntry("Blue",3);
     glutAddMenuEntry("Quit",4);
     glutAttachMenu(GLUT_RIGHT_BUTTON);
     myinit();
     glutDisplayFunc(display);
     glutMainLoop();
}
```

11.Simple Bezier Curve

```
#include<GL/glut.h>
#include <stdlib.h>
#include <math.h>
/* Set initial size of the display window. */
GLsizei winWidth = 600, winHeight = 600;
/* Set size of world-coordinate clipping window.*/
GLfloat xwcMin = -50.0, xwcMax = 50.0;
GLfloat ywcMin = -50.0, ywcMax = 50.0;
class wcPt3D
{
public:
     GLfloat x, y, z;
};
void init(void)
{
     /* Set color of display window to white. */
     glClearColor(1.0, 1.0, 1.0, 0.0);
}
void plotPoint(wcPt3D bezCurvePt)
{
     glBegin(GL_POINTS);
     glVertex2f(bezCurvePt.x, bezCurvePt.y);
     glEnd();
}
void binomialCoeffs(GLint n, GLint* C)
{

     GLint k, j;
     for (k = 0; k <= n; k++) {
          /* Compute n!/(k!(n - k)!). */
          C[k] = 1;
          for (j = n; j >= k + 1; j--)
               C[k] *= j;
          for (j = n - k; j >= 2; j--)
               C[k] /= j;
     }
}
void computeBezPt(GLfloat u, wcPt3D* bezPt, GLint nCtrlPts,
wcPt3D* ctrlPts, GLint* C)
{
     GLint k, n = nCtrlPts - 1;
     GLfloat bezBlendFcn;
```

```
        bezPt->x = bezPt->y = bezPt->z = 0.0;
        /* Compute blending functions and blend control points. */
        for (k = 0; k < nCtrlPts; k++) {
             bezBlendFcn = C[k] * pow(u, k) * pow(1 - u, n - k);
             bezPt->x += ctrlPts[k].x * bezBlendFcn;
             bezPt->y += ctrlPts[k].y * bezBlendFcn;
             bezPt->z += ctrlPts[k].z * bezBlendFcn;
        }
}
void bezier(wcPt3D* ctrlPts, GLint nCtrlPts, GLint nBezCurvePts)
{
        wcPt3D bezCurvePt;
        GLfloat u;
        GLint* C, k;
       C = new GLint[nCtrlPts];
        binomialCoeffs(nCtrlPts - 1, C);
        for (k = 0; k <= nBezCurvePts; k++)
        {
        u = GLfloat(k) / GLfloat(nBezCurvePts);
        computeBezPt(u, &bezCurvePt, nCtrlPts, ctrlPts,C);
        plotPoint(bezCurvePt);
       }
delete[] C;
}
void displayFcn(void)
{
/* Set example number of control points and number of curve
positions to be plotted along the Bezier curve. */
GLint nCtrlPts = 4, nBezCurvePts = 1000;
wcPt3D ctrlPts[4] = { {-40.0, -40.0, 0.0}, {-10.0, 200.0,
0.0},{10.0, -200.0, 0.0}, {40.0, 40.0, 0.0} };
glClear(GL_COLOR_BUFFER_BIT); // Clear display window.
glPointSize(4);
glColor3f(1.0, 0.0, 0.0); // Set point color to red.
bezier(ctrlPts, nCtrlPts, nBezCurvePts);
glFlush();
}
void winReshapeFcn(GLint newWidth, GLint newHeight)
{
        /* Maintain an aspect ratio of 1.0. */
        glViewport(0, 0, newHeight, newHeight);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(xwcMin, xwcMax, ywcMin, ywcMax);
        glClear(GL_COLOR_BUFFER_BIT);
}
void main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE |GLUT_RGB);
        glutInitWindowPosition(50, 50);
        glutInitWindowSize(winWidth, winHeight);
```

```
    glutCreateWindow("Bezier Curve");
    init();
    glutDisplayFunc(displayFcn);
    glutReshapeFunc(winReshapeFcn);
    glutMainLoop();
}
```