**Assignment No. 4**

4. Implement the following polygon filling methods : i) Flood fill / Seed fill ii) Boundary fill ; using

mouse click, keyboard interface and menu driven programming

|  |
| --- |
| **Aim** |
| Draw the polygons by using the mouse. Choose colors by clicking on the designed color pane. Use window port to draw. (Use DDA algorithm for line drawing) |

|  |  |
| --- | --- |
| **Objective(s)** | |
| **1** | To learn mouse events in open GL |
| **2** | To learn DDA line Algorithm |

|  |
| --- |
| **Theory** |
| **Mouse Interaction Example**  declare prototype  •myMouse(int button, int state, int x, int y)  •myMovedMouse  „ Register callbacks:  •glutMouseFunc(myMouse):when mouse button pressed  •glutMotionFunc(myMovedMouse):when mouse moves  „ Button returned values:  • GLUT\_LEFT\_BUTTON, GLUT\_MIDDLE\_BUTTON,  GLUT\_RIGHT\_BUTTON  „ State returned values:  • GLUT\_UP, GLUT\_DOWN  „ X,Y returned values:  „ x,y coordinates of mouse location  Each mouse click generates separate events and Store click points in static variable in your mouse function  „ Example: draw (or select ) rectangle on screen  „ Mouse y returned assumes y=0 at top of monitor  „ OpenGL assumes y=0 at bottom. Solution? Flip mouse y  void myMouse(int button, intstate, int x, int y)  {  static GLintPoint corner[2];  static int numCorners = 0; // initial value is 0  if(button == GLUT\_LEFT\_BUTTON &&  state == GLUT\_DOWN  {  corner[numCorners].x = x;  corner[numCorners].y= screenHeight – y; //flip y coord  numCorners++;  if(numCorners == 2)  {  // draw rectangle or do whatever you planned to do  glRecti(corner[0].x, corner[0].y  corner[1].x, corner[1].y);  numCorners == 0;  }  else if(button == GLUT\_RIGHT\_BUTTON &&  state == GLUT\_DOWN)  glClear(GL\_COLOR\_BUFFER\_BIT); // clear the window  glFlush( );  }    A sample code illustrats handling mouse events  1: #include <GL/glut.h>  2: int ww=600,wh=400;  3: int first=0;  4: int xi,yi,xf,yf;  5: void drawLine(int x1,int y1,int x2,int y2)  6: {  7: glClear(GL\_COLOR\_BUFFER\_BIT);  8: glLineWidth(5.0);  9: glBegin(GL\_LINES);  10: glVertex2i(x1,y1);  11: glVertex2i(x2,y2);  12: glEnd();  13: glFlush();  14: }  15: void display()  16: {  17: glClearColor(0.2, 0.4, 0.0, 1.0);  18: glColor3f(0.7, 0.4, 0.0);  19: glClear(GL\_COLOR\_BUFFER\_BIT);  20: glFlush();  21: }  22: void mouse(int btn,int state,int x,int y)  23: {  24: if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)  25: {  26: switch(first)  27: {  28: case 0:  29: xi=x;  30: yi=wh-y;  31: first=1;  32: break;  33: case 1:  34: xf=x;  35: yf=wh-y;  36: drawLine(xi,yi,xf,yf);  37: first=0;  38: break;  39: }  40: }  41: }  42: void myinit()  43: {  44: glViewport(0,0,ww,wh);  45: glMatrixMode(GL\_PROJECTION);  46: glLoadIdentity();  47: gluOrtho2D(0.0,(GLdouble)ww,0.0,(GLdouble)wh);  48: glMatrixMode(GL\_MODELVIEW);  49: }  50: int main(int argc,char\*\* argv)  51: {  52: glutInit(&argc,argv);  53: glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);  54: glutInitWindowSize(ww,wh);  55: glutCreateWindow("Draw Line With Mouse Click");  56: glutDisplayFunc(display);  57: myinit();  58: glutMouseFunc(mouse);  59: glutMainLoop();  60: return 0;  61: } |

|  |
| --- |
| **Input** |
| Starting point and Ending point of line  Select the color for line drawing |

|  |
| --- |
| **Output** |
| Colored line |