```
//draw rectangle on left mouse click
//programmed by Gaddisa Olani.  2015
#include <GL/glut.h>
GLsizei MOUSEx=0, MOUSEy=0;
GLfloat SIDE=50;
GLfloat BLUE[3] = {0,0,1};

void drawSquare1()
{
   glColor3fv(BLUE);
   glBegin(GL_POLYGON);
      glVertex3f(MOUSEx, MOUSEy,0);
      glVertex3f(MOUSEx+SIDE, MOUSEy,0);
      glVertex3f(MOUSEx+SIDE, MOUSEy+SIDE,0);
      glVertex3f(MOUSEx, MOUSEy+SIDE,0);
   glEnd();
   glFlush();
}

void display(void)
{
   glClearColor (0.0,0.0,0.0,1.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glLoadIdentity();
   drawSquare1();
   glFlush();
}
void reshape(int w, int h)
{
   glViewport(0,0,(GLsizei)w,(GLsizei)h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
    //gluPerspective (60, (GLfloat)w / (GLfloat)h, 1.0, 100.0);
   glOrtho(0.0,1368,768,0,-1.0,1.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void spindisplay(void)
{
   glutPostRedisplay();
}

void setX(int x)
{
   MOUSEx=x;
}
```

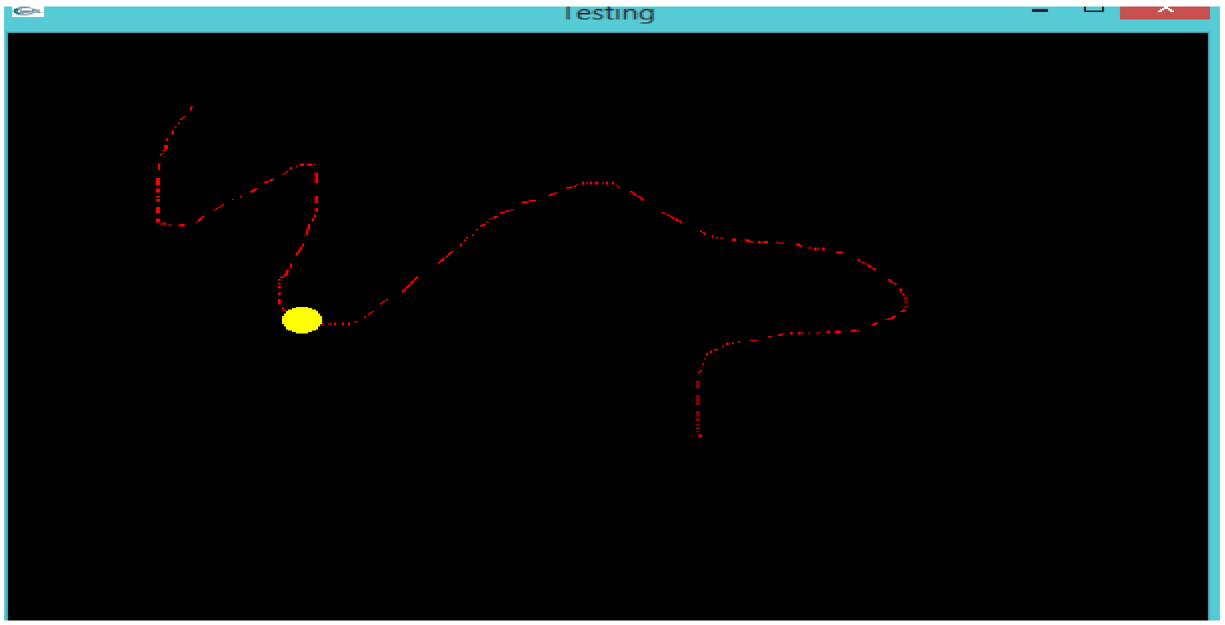
```
void setY(int y)
{
   MOUSEy=y;
}
void mouse(int btn, int state, int x, int y)
{
   if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
   {
      setX(x);
      setY(y);
      //drawSquare(MOUSEx,HEIGHT-MOUSEy);
      glutPostRedisplay();
   }
   if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
   {
      exit(1);   // To Exit the Program
   }
}
int main(int argc, char **argv)
{  glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowSize(1366,768);
   glutInitWindowPosition(0,0);
   glutCreateWindow("Moving squares");
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMouseFunc(mouse);
   glutIdleFunc(spindisplay);
   glutMainLoop();
}
```

## //Continuous Line On Mouse Drag

```
#include <iostream>
#include <glut.h>
#include <string.h>
#define WIDTH 600
#define HEIGHT 600

using namespace std;

double arr[5000][4];
int z=0;
int flag=0;
float radius=0.03;
int ptr=0;
int faltu_bit=1;
float color[3][3]={{1.0,1.0,1.0},{1.0,1.0,0.0},{0.0,1.0,0.0}};
```

```
void drawText(char *str,float x,float y,int id)
{
    int i;
    int len=strlen(str);
    //glLoadIdentity();
    glColor3f(color[id][0],color[id][1],color[id][2]);
    glRasterPos2f(x,y);
    for(i=0;i<len;i++)
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);
}


void init()
{
    glClearColor( 0.0, 0.0, 0.0, 1.0);
    glMatrixMode( GL_PROJECTION);
    gluOrtho2D(0.0,WIDTH,0.0,HEIGHT);
    memset(arr,0,5000);
    glPointSize(20.0);
}

void resetAll()
{
    memset(arr,0,5000);
    z=0;
}

///OPENGL MAPPING///
float getOpenGLX(int x)
{
    double ox = x/ (double)WIDTH*(WIDTH);
    return ox;
}

float getOpenGLY(int y)
{
    double oy = (1 - y/ (double) HEIGHT)*HEIGHT;
    return oy;
}

void drawPoints()
{
    glBegin( GL_POINTS );
    glColor3f( 0.0,1.0,0.0 );
    for ( int i = 0; i < z; i++ )
    {
        glVertex2f( arr[i][0],arr[i][1]);
    }
    glEnd();
}

void drawBall(float x,float y)
{
    glBegin( GL_POINTS);
    glColor3f( 1.0,1.0,0.0 );
    glVertex2f(x,y);
    glEnd();
```

```
}

void drawLines()
{
    glBegin(GL_LINES);
    glColor3f(1.0,0.0,0.0);
    for(int i=0;i<z;i++)
    {
        glVertex2f(arr[i][0],arr[i][1]);
    }
    glEnd();
}

void addValue(int x,int y)
{
    arr[z][0]=getOpenGLX(x);
    arr[z++][1]=getOpenGLY(y);
}
void trackBall()
{
    drawPoints();
}
void myDisplay()
{
    glClear( GL_COLOR_BUFFER_BIT);
    if(!flag)
    {
        drawLines();
        if(!faltu_bit)
        drawBall(arr[ptr][0],arr[ptr][1]);
    }
    if(faltu_bit)
    {
     drawText("Project by: Adil Ansar [10 CSS-32]",50.0,500.0,0);
     drawText("Welcome",250.0,300.0,1);
    drawText("Drag the Mouse Any Where in the Window to see the
Path",10.0,200.0,2);
    }
    glutSwapBuffers();
    glutPostRedisplay();
    glFlush();
}


void myMouseStat(int button,int state,int x, int y)
{
    if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
    {
        if(!flag)
        {
            if(faltu_bit)
            {
                faltu_bit=0;
            }
            resetAll();
            flag=1;
        }
    }
    else if(button==GLUT_LEFT_BUTTON && state==GLUT_UP)
    {
        if(flag)
        {
            ptr=0;
            flag=0;
        }
    }
}


void myPressedMove(int x,int y)
{
    if(flag)
    {
        addValue(x,y);
    }
}

void myTimer(int t)
{
    if(ptr!=z)
    {
         ptr++;
    }
    else
{
    ptr=0;
}
    glutTimerFunc(100,myTimer,0);
}

int main( int argc, char ** argv)
{
    glutInit( &argc, argv);
    glutInitDisplayMode( GLUT_DOUBLE| GLUT_RGB);
    glutInitWindowPosition( 100, 100);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutCreateWindow( "Testing");
    init();
    glutDisplayFunc(myDisplay);
    glutMouseFunc(myMouseStat);
    glutMotionFunc(myPressedMove);
    glutTimerFunc(100,myTimer,0);
    glutMainLoop();
    return 0;
}
```



## 1 Answer

In this code

```
void drawLines()
{
    glBegin(GL_LINES);
    glColor3f(1.0,0.0,0.0);
    for(int i=0;i<z;i++)
    {
        glVertex2f(arr[i][0],arr[i][1]);
    }
    glEnd();
}
```

`GL_LINES` is a drawing mode where two consecutive vertices make a line segment. Then the next two and so so. What you're drawing is a *line strip*; replacing `GL_LINES` with `GL_LINE_STRIP` will give you your expected result.

On a side note: You should abandon using immediate mode (glBegin, glVertex, glEnd). It's slow, it's cumbersome to work with and in the long term can become a major PITA. Use at least vertex arrays; they've been the recommended way of supplying geometry data for well over 15 years. Plus it makes your code much simpler. Your code above can be replaced with this:

```
/* vertex data comes from an array */
glEnableClientState(GL_VERTEX_ARRAY);

/* we want to use a common color for all vertices */
glDisableClientState(GL_COLOR_ARRAY);
glColor3f(1.0, 0.0, 0.0);

/* where to get the data from */
```

```
glVertexPointer(2, GL_DOUBLE, sizeof(double)*4, arr);

/* draw the whole thing */
glDrawArrays(GL_LINE_STRIP, 0, z);

/* cleanup */
glDisableClientState(GL_VERTEX_ARRAY);
```

As you can see it's shorter, more concise, easier to read and it avoids doing `z+3` function calls, each of which taking some time to execute.

**Important** that OpenGL can digest that multidimensional array `arr` is due to the fact, that statically allocated storage of multidimensional arrays is always contiguous. It would not work if you'd allocate an array of pointers to arrays (the naive way to allocate multidimensional arrays dynamically).