OpenGL

- OpenGL is mainly considered an API (an Application Programming Interface) that provides us with a large set of functions that we can use to manipulate graphics and images. However, OpenGL by itself is not an API, but merely a specification, developed and maintained by the **Khronos** Group.

- The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it is possible for the API to be implemented entirely in software, it is designed to be implemented mostly or entirely in hardware.

- Although the function definitions are superficially similar to those of the programming language C, they are language-independent. As such, OpenGL has many language bindings.

- In addition to being language-independent, OpenGL is also cross-platform.

GLUT

- GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs.
- It implements a simple windowing application programming interface (API) for OpenGL.
- GLUT makes it considerably easier to learn about and explore OpenGL programming.
- GLUT provides a portable API so you can write a single OpenGL program that works on both Win32 PCs and X11 workstations.
- GLUT is designed for constructing small to medium sized OpenGL programs.
- GLUT is well-suited to learning OpenGL and developing simple OpenGL applications,
- GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits like Motif.
- GLUT is simple, easy, and small.

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

Library

```
void display(void)
{
  glBegin(GL_POINTS);
  glVertex2f(x,y);
  glEnd();
  do
  {  if(abs(m)<=1){
        x=x+1;
        y=y+m;
        glBegin(GL_POINTS);
        glVertex2f(x,y);
        glEnd(); }
     if(abs(m)>1){
        y=y+1;
        x=x+(1/m);
        glBegin(GL_POINTS);
        glVertex2f(x,y);
        glEnd();   } }
  while(x<=xn && y<=yn);
  glFlush(); }
```

which are bracketed by **glBegin()** and **glEnd()**, define the object to be drawn . Here a point (x,y) will be drawn

It specifies the vertex's coordinates as single-precision floating-point numbers.

glFlush() ensures that the drawing commands are actually executed rather than stored in a *buffer* awaiting additional OpenGL commands

Algo Implement

```
void init(void)
{
   glClear(GL_COLOR_BUFFER_BIT);
   glClearColor(0,0,0,0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-100,100,-100,100);
}


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


   scanf("%f %f %f %f", &x,&y,&xn,&yn);
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(100,100);
   glutCreateWindow("___");
   init();
   glutDisplayFunc(display);
   glutMainLoop();


return 0;
}
```

glClearColor() establishes what color the window will be cleared to, and **glClear()** actually clears the window. Once the clearing color is set, the window is cleared to that color whenever **glClear()** is called. This clearing color can be changed with another call to **glClearColor()**.

glutInit(int **argc*, char ***argv*) initializes GLUT and processes any command line arguments geometry). **glutInit()** should be called before any other GLUT routine.

glutInitDisplayMode(unsigned int *mode*) specifies whether to use an *RGBA* or color-index color model. You can also specify whether you want a single- or double-buffered window

glutInitWindowPosition(int *x*, int *y*) specifies the screen location for the upper-left corner of your window.

glutInitWindowSize(int *width*, int *size*) specifies the size, in pixels, of your window.

int **glutCreateWindow**(char **string*) creates a window with an OpenGL context. It returns a unique identifier for the new window. Be warned:
Until **glutMainLoop()** is called (see next section), the window is not yet displayed.

You can check -
https://www.glprogramming.com/red/chapter01.html