Assignment - A5

Problem Statement - Write a C++ / Java program to draw a 4X4 chessboard rotated 45° with the horizontal axis. Use Bresenham's algorithm to draw the lines. Use seed fill algorithm to fill black squares of rotated chessboard.

Objective - ① To learn and understand seed fill and and boundary fill algorithm
② To implement seed fill algorithm recursively and non-recursively.
③ To implement rotation of polygon using 2D transformation.

Outcome - ① Students will be able to implement and seed fill (floor and boundary fill) algorithm
② students will be able to implement rotation of 2D figures using transformation

S/W and H/W requirements - Core i3 processor, Fedora OS, QT creator.

Theory -

Polygon filling is process of coloring in a fixed area or regions are defined at pixel level we have algorithms like

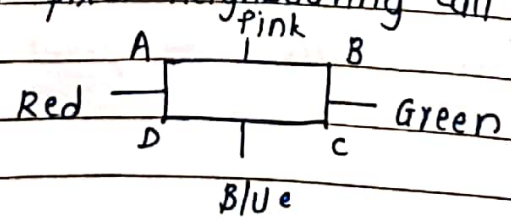
① Boundary fill
② Flood fill
③ Edge fill
④ Fence fill

Boundary fill algorithm

① This algorithm needs one point which is surely inside the polygon
② This starting point is called seed point which is nothing but a point from which the filling process starts.
③ It is recursive method. The algorithm checks to see if seed pixel has a boundary color pixel or not
④ If no, then fill the pixel boundary color and boundary make a recursive call to each of neighbouring pixel.
⑤ This algorithm work for any shaped polygon and fills that polygon with boundary color but it uses high power of recursive calls and takes more time and memory.

Flood fill Algorithm

① This algorithm also begins with seed point which must be surely inside the polygon original color is previous or old color.
② If yes, fill that pixel with new color and use each of pixel neighbouring call

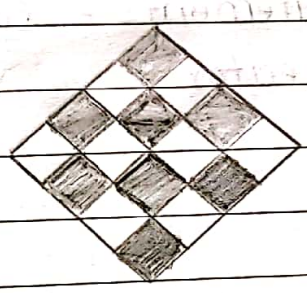


③ Sometimes we want to fill an area that is not defined with a single color boundary. Flood filed helps in their case as we do not check boundary pixel

Algorithm -

```
Boundary (x, y, newcolor) {
    current = getpixel (x, y);
    If ((current != newcolor) && (current != boundarycolor))
    { putpixel (x, y, newcolor);
      boundary (x+1, y, newcolor);
      boundary (x, y+1, newcolor);
      boundary (x-1, y, newcolor);
      boundary (x, y-1, newcolor);
    }
```

Testcase

Input	Output	Status
length : 100		Success

Conclusion - Thus the seed fill and boundary fill algorithm will be implemented successfully.