VARIABLES USED IN THE PROGRAM

------------------------------------------------------------------------------------------------------------------------------------------

(Structure POINTE

VARIABLES – Integer x,y )

(Structure LINE

VARIABLES- structure POINTE P1, structure POINTE P2, structure POINTE P3, structure POINTE P4

Integer flag // flag to indicate whether the line exists or not

Integer filled // used to check whether the line is filled with color

Integer horizontal //used to check for horizontal and vertical)

total\_squares = 16 //number of boxes in a 5\*5 matrix of dots and boxes

cur\_player = 0 // current player 0 = player1 and 1 = player2.

cur\_user\_flag = 0 // to know whether the current user has scored a point.

user\_points[3]. // user\_points[1] player 1 points similarly for player 2.

line\_size //stores number of lines drawn.

------------------------------------------------------------------------------------------------------------------------------------------

Algorithm\_Main

1. Start.
2. Call display() // displays the points and lines(circles and rectangles). Or displays the Grid (5x5).
3. Check for mouse activity and call mouse() .
4. End.

Algorithm\_Mouse

1. Get co ordinates of clicked points(x,y).
2. Call get\_line\_pos(x,y). // to check if points is inside a horizontal line or a vertical line.
3. Switch player
   1. If current player has scored a point then
      1. Don’t switch .
   2. Else if current player hasn’t scored any point then
      1. Switch player .
4. Check if total squares == 0
   1. If yes print\_winner() & game over.

Algorithm\_getlinepos()

1. Identify the line in which the point lies.
2. Call horizontal(line-num) if line is horizontal
3. Else call vertical(line-num).

Algorithm setPixel(x,y)

1. if cur\_player == 0 put color red

2. else if cur\_player == 1 put color green

Algorithm getPixel(x,y,color)

1. Return color at point x,y

Algorithm display()

1. draw a 5x5 set of lines representing the rectangles or lines

Start numbering the lines from horizontal and alternating between vertical and horizontal

(start from 0)

2. set filled = 0 ,flag = 1 for all lines

3. set horizontal = 1 for horizontal lines and horizontal =0 for vertical lines.

4. draw 25 circles to denote points // built in opengl function.

Algorithm boundary\_fill(x,y,fillcolor,bordercolor)

1. getPixel(x,y,interiorColor)

2. if((interiorColor!=borderColor

3. setPixel(x,y,fillColor);

4. boundaryFill4(x+1,y,fillColor,borderColor);

5. boundaryFill4(x-1,y,fillColor,borderColor);

6. boundaryFill4(x,y+1,fillColor,borderColor);

7. boundaryFill4(x,y-1,fillColor,borderColor);

8 end if

Algorithm horizontal(x) // x denotes line number

1. set filled of line[x].filled = 1

2. using boundary numbers check if corresponding 6 lines are filled or not if true then

/\*To check upper half forms a square\*/

3. Check if((i-9)>=0 && L[i-9].filled == 1)

4. Check if((i-8)>=0 && L[i-8].filled == 1)

5. Check if(((i-8)+1)<=42 && L[i-7].filled == 1)

6. color the square// color using boundary fill by specifying an interior point

7. else don’t color the square

/\*To check if lower half forms square\*/

8. Check if((i+9)<= 42 && L[i+9].filled == 1)

9. Check if((i+1)>=0 && L[i+1].filled == 1)

10. Check if((i+2)<=42 && L[i+2].filled == 1)

11. color the square// color using boundary fill by specifying an interior point

12. else don’t color the square

13. color the line //color using boundary fill by specifying an interior point

Algorithm vertical(x) // x denotes line number

1. set filled of line[x].filled = 1

2. using boundary numbers check if corresponding 6 lines are filled or not if true then

3. color the square // color using boundary fill by specifying an interior point

/\*To check if left half forms a square \*/

4. Check if((i-1)>=0 && L[i-1].filled == 1)

5. Check if((i-2)>=0 && L[i-2].filled == 1)

6. Check if((i+7)<=42 && L[i+7].filled == 1)

7. color the square // color using boundary fill by specifying an interior point

8. else don’t color the square

/\*To check if right half forms a square \*/

9. Check if((i-1)>=0 && L[i-1].filled == 1

10. Check if((i+8)<=42 && L[i+8].filled == 1)

11. Check if((i+1)<=42 && L[i+1].filled == 1)

12. color the square

13. else don’t color the square

14. color the line //color using boundary fill by specifying an interior point

//END OF PROCEDURE//

--------------------------------------------------------------------------------------------------------------------------

Submitted by Sunjay Calvvin P(260)

Anush Kumar(216)

Roja S Rajan(251)

Vishnu G(263)