CG Assignment 6

YASH SARANG D6AD/47

Aim: Scanline Polygon filling Algorithm

Theory: Scanline filling is basically filling up of polygons using horizontal lines or scanlines. The purpose of the SLPF algorithm is to fill (color) the interior pixels of a polygon given only the vertices of the figure.. The algorithm works by intersecting scanline with polygon edges & fills the polygon between pairs of intersecting.

* Components of polygon filling

① Edge brackets -
It contains an edge's info. The entries of edge bucket vary according to data structures which you have used.

② Edge table
It consists of several of edge lists which holds all of the edges that compose of the figure. When creating edges, the vertices of the edges

that compose the figure. When creating edges, the vertices of the edges need to be ordered from left to right. and the three edges are maintained in increasing y_{min} order.

The filling is complete once all of the edges are removed from the ET.

* Algorithm:

① We will process the polygon edge after edge, & store it in the edge table.

② Storing is done by storing the edge in the same scanline edge tuple is sorted using insertion sort, according to its x_{min} value.

③④ After the whole polygon is added to the edge table, the figure is now filled.

⑤ Filling is stored from the first scanline at the bottom & continued till the top.

⑥ Now, the active edge table is taken & the following things are repeated for each scanline.

(i) Copy all edge buckets of the designed scanline to the active edge table.

(ii) Perform an insertion sort according to the x of y_{min} values.

(iii) Remove all edge buckets whose y_{max} is equal to or greater than the scanline.

(iv) Fill up pairs of edges in active tuple, if any vertex is got, follow the instructions:-

- If both of the lines intersect at the vertex are on the same side of the scanline, consider it as two points.

- If lines intersecting at the vertex are at opposite sides of the scanline, consider it as only one point.

(v) Update the x of y_{min} by adding slope inverse for each bucket.

Conclusion:

We have studied scanline polygon filling algorithm. We learnt in detail about its algorithm, its pros & cons.

We have also implemented programs for this.

Code :-

```
#include <stdio.h>
 #include <conio.h>
 #include <graphics.h>

 void main() {
  int n, i, j, k, gd, gm, dy, dx;
  int x, y, temp;
  int a[20][2], xi[20];
  float slope[20];
  clrscr();
  printf("\n\n\tEnter the no. of edges of polygon : ");
  scanf("%d", & n);
  printf("\n\n\tEnter the cordinates of polygon :\n\n\n ");
  for (i = 0; i < n; i++) {
   printf("\tX%d Y%d : ", i, i);
   scanf("%d %d", & a[i][0], & a[i][1]);
  }
  a[n][0] = a[0][0];
  a[n][1] = a[0][1];
  detectgraph( & gd, & gm);
  initgraph( & gd, & gm, "C:\\TurboC3\\BGI");
  /*- draw polygon -*/
  for (i = 0; i < n; i++) {
   line(a[i][0], a[i][1], a[i + 1][0], a[i + 1][1]);
  }
  getch();
  for (i = 0; i < n; i++) {
   dy = a[i + 1][1] - a[i][1];
   dx = a[i + 1][0] - a[i][0];
   if (dy == 0) slope[i] = 1.0;
   if (dx == 0) slope[i] = 0.0;
   if ((dy != 0) && (dx != 0)) /*- calculate inverse slope -*/ {
    slope[i] = (float) dx / dy;
   }
  }
  for (y = 0; y < 480; y++) {
   k = 0;
   for (i = 0; i < n; i++) {
    if (((a[i][1] <= y) && (a[i + 1][1] > y)) ||
     ((a[i][1] > y) && (a[i + 1][1] <= y))) {
     xi[k] = (int)(a[i][0] + slope[i] * (y - a[i][1]));
     k++;
    }
   }
   for (j = 0; j < k - 1; j++) /*- Arrange x-intersections in order -*/
```

```
    for (i = 0; i < k - 1; i++) {
      if (xi[i] > xi[i + 1]) {
        temp = xi[i];
        xi[i] = xi[i + 1];
        xi[i + 1] = temp;
      }
    }
   setcolor(3);
   for (i = 0; i < k; i += 2) {
     line(xi[i], y, xi[i + 1] + 1, y);
     getch();
   }
  }
 }
```

Output:-

Co-ordinates :-

X0 Y0 = 200 300

X1 Y1 = 300 400

X2 Y2 = 200 400

X3 Y3 = 300 300