classmate

Date

Page

INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR

B.TECH 5th SEMESTER MID-TERM EXAMINATION, OCTOBER 2021

COMPUTER GRAPHICS [CS3121]

Name = Abhiroop Mukherjee

Enrolment No. : 510519109

G-Suite ID : 510519109.abhirup@students.iiests.ac.in

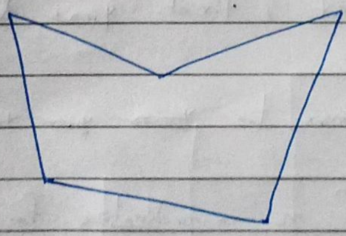
Date: 9th October 2021

No. of pages: 12

[filename ⇒ <roll no>.pdf]

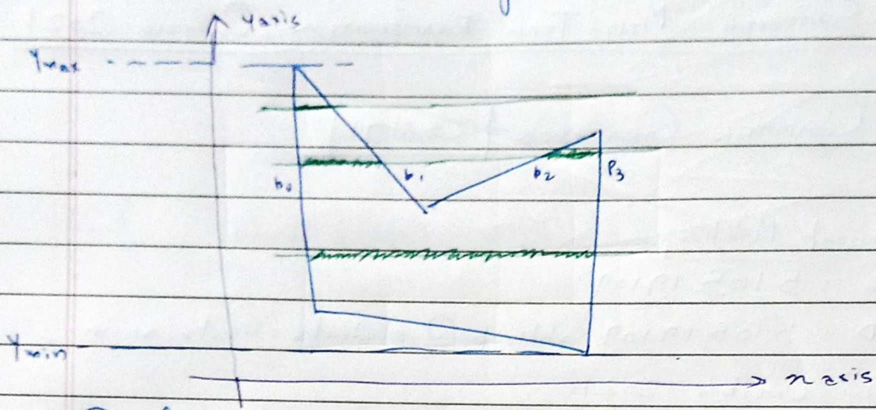
Q2) a) Scan Line Polygon fill

consider following shape [given only boundary points]



→ our objective is to fill this polygon using scanline polygon fill

→ To do scanline polygon fill, we do the following



~~1 find~~

(i) assuming the set of boundary points are in some ~~list~~ list/array, we first sort the array according to following rules

(a) sort according y co-ordinate (decresing)

(b) if y co-ordinate equal
sort according to x-co-ordinate (incresing)

(ii) using sorted list, find y_{max} & y_{min}

(iii) now go from y_{max} to ~~ymax~~ ~~yin~~ y_{min}

→ ~~fill~~ if no. of x-co-ordinates for a given y even, there is no corner, we fill between two points alternatively

Eg fill between (p_0 & p_1), and (p_2 & p_3)

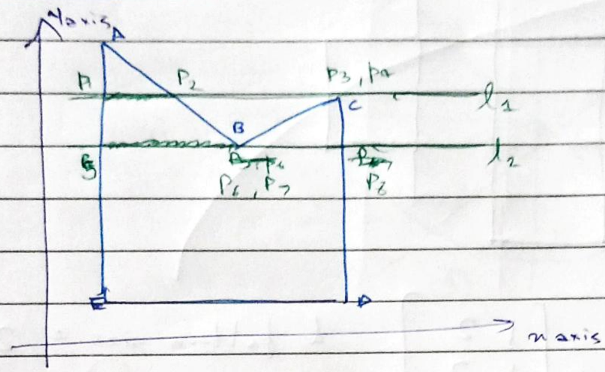
→ if no. of x-coordinate of given y odd, there is a corner.

→ detect the corner using the fact that corners have their neighbouring vertices on same side of scan line, then assume corner as two points

→ Then ~~color~~ fill alternatively

(iv) in the end, entirety of the polygon will be filled.

Corner detection



→ Property of corner → neighbouring vertices same side of scan line.

Eg for l_1, neighbouring vertex of C (~~p3, p4~~) is (~~p5, p6~~) B & D, down of l_1

Eg for B, neighbouring vertex A, B, up of l_2

→ If we detect the point, we assume corner as two point

Eg C → assume both p_3, p_4

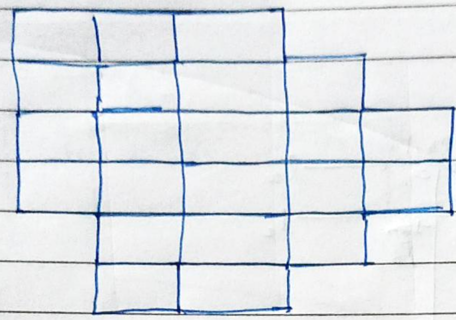
B → assume both ~~p5, p6~~ p_6, p_7

then we follow usual coloring

Eg l_1 → fill $p_1 \to p_2$, fill $p_3 \to p_4$

l_2 → fill ~~p4~~ $p_5 \to p_6$, fill $p_7 \to p_8$

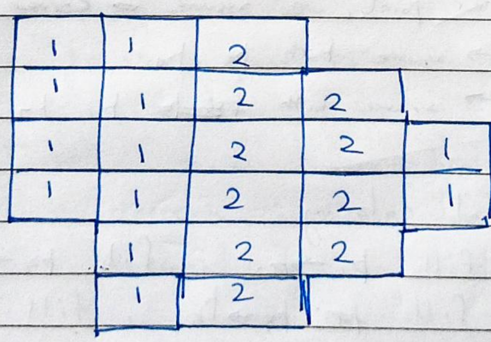
(Q2) b) let the pattern be



given pattern $= \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$ & pattern size = 2 pixel

$\therefore$ resized pattern $= \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix}$

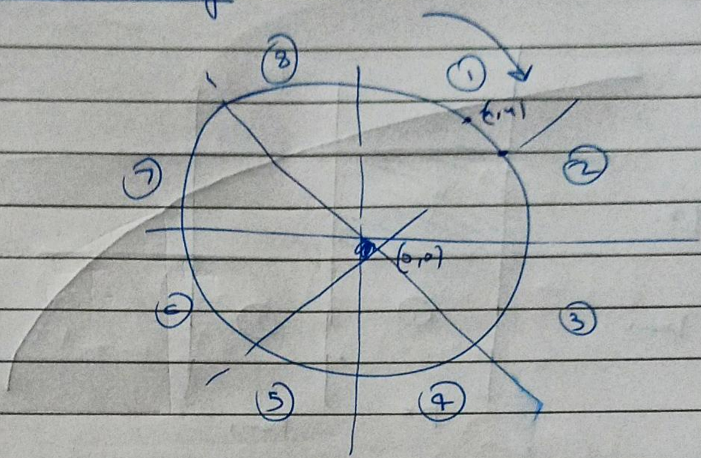
$\therefore$ filled pattern



a

Q1.a) Mid Point Circle Drawing

Idea



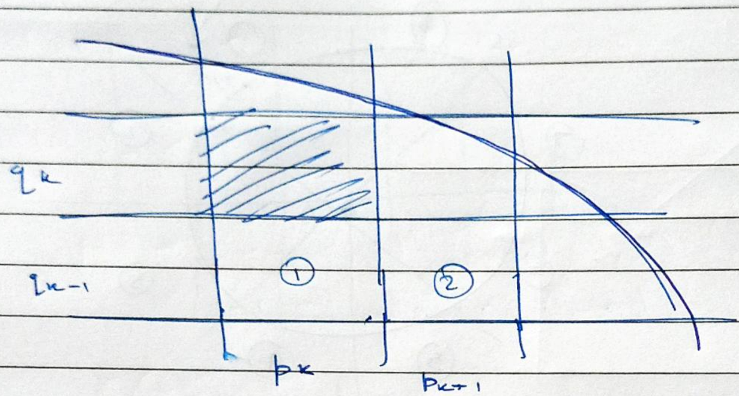
→ if we some how get point to be plotted (x, y) for region 1, we can use symmetry and also draw

② → (y, x)
③ → $(-y, x)$
④ → $(-x, y)$
⑤ $(-x, -y)$
⑥ → $(-y, -x)$
⑦ $(y, -x)$
⑧ $(x, -y)$

→ so our idea will be to only find point's in region ① & use symmetry to draw other octants.

→ we will be moving clockwise in region ①, i.e from ~~0~~, $(0, r)$ to ~~(...)~~ (x, y) where $x = y$

→ now conside we already ~~know the~~ drawn q_k ~~→ p_k~~ (p_k, q_k)



→ now we have two choice → draw pixel ① or ②,

→ to choose between ① & ②, we do the following

→ circle eq → $x^2+y^2-r^2=0$ → let $f(x,y) = x^2+y^2-r^2$

∴ if for (a,b) → $f(a,b)=0$ → (a,b) on circle
→ $f(a,b)<0$ → (a,b) inside circle
$f(a,b)>0$ → (a,b) outside circle.

→ idea is that, we check $f(x_i - 1, y_i + \frac{1}{2}) = P_i$

if $P_i < 0$ → we can plot (x_i+1, y_i-1)
$P_i > 0$ → we can't plot (x_i+1, y_i-1), we have to plot (x_i+1, y_i)

$$\therefore P_{i+1} - P_i = [(x+1)+1]^2 + [y_{i+1} - \tfrac{1}{2}]^2 - [x_i+1]^2 - [y_i - \tfrac{1}{2}]^2$$

$$= 2(x_i+1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$

$$\therefore P_{i+1} = P_i + 2(x_i+1) + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$

if P_i was chosen $\rightarrow P_i < 0$, $y_{i+1} = y_i$

P_i not chosen $\rightarrow P_i \geq 0$, $y_{i+1} = y_i - 1$

$$P_{i+1} = \begin{cases} P_i + 2(x_i+1) + 1 & P_i < 0 \\ P_i + 2(x_i+1) + 1 - 2(y_i-1) & P_i \geq 0 \end{cases}$$

$$= \begin{cases} P_i + 2x_i + 3 & P_i < 0 \\ P_i + 2(x_i - y_i) + 5 & P_i \geq 0 \end{cases}$$

initial $P_1 \longrightarrow x_i = 0 \quad y_i = r$

$$P_1 = (0+1)^2 + (r - \tfrac{1}{2})^2 - r^2$$

$$= \frac{5}{4} - r \approx 1 - r$$

Algorithm

initial → $x = 0$, $y = r$, $P = 1 - r$

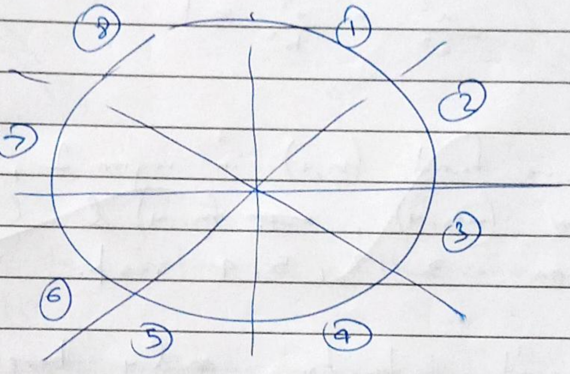
```
while (x ≤ y) {

    plot → (x,y), (y,x), (-x,y), (-y,x), (-x,-y),
           (-y,-x), (x,-y), (y,-x)

    if (P < 0)
        p = p + 2x + 3
    else {
        p = p + 2(x - y) + 5
        y = y - 1
    }
    x++;
}
```

1) b) "Midpoint circle drawing algorithm is applicable for one octant of the circle only, but for a midpoint ellipse drawing algorithm is applicable for one quadrant of the ellipse"

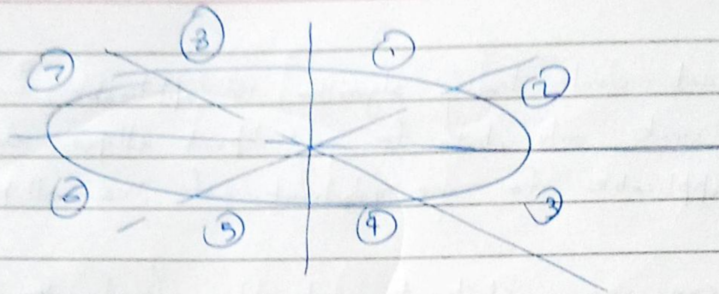
→ We can use mid point circle drawing algorithm with octant because of full symmetry of the circle.



$x^2 + y^2 = r^2$

→ if we draw (x, y) in 1, using symmetry, we can also draw symmetric point (y, x), $(-x, y)$, (y, x), $(x, -y)$, $(y, -x)$, $(-x, -y)$, $(-y, -x)$

→ for ellipse this is not possible



ellipse equation

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

→ given we found (x,y) in region 1, we can only draw $(-x,y)$, $(x,-y)$ & $(-x,-y)$ in region 8, 5, 4 resp.

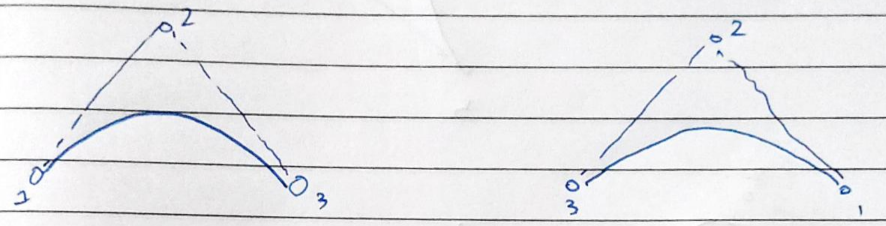
→ we can't interchange x & y here because it will violate ellipse equation

$$\frac{y^2}{a^2} + \frac{x^2}{b^2} \neq 1 \text{ always.}$$

→ because of this we have to find all points in a quadrant in mid point ellipse

Q3) Properties of Bezier Curve

(i) given a ~~bezier~~ Bezier curve, reversing the sequence of control points does not modify Bezeir curve



(ii) ~~B~~ Bezier curve lies entirely inside the convex hull of the control points.

~~→ we can quickly check if two be →~~

(iii) Bezier curve invariant under translation, rotation, shearing & scaling.

(iv) changing even one control point modifies entire Bezier curve

(v) Partition of Unity

$$\sum_{i=0}^{n} {}^{n}C_i \, (1-u)^{n-i} \, u^i = 1$$

(vi) Bezier Curve can be represented in matrix form.

Q3) b)