CSE-303: COMPUTER GRAPHICS

PROFESSOR DR. SANJIT KUMAR SAHA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING JAHANGIRNAGAR UNIVERSITY,

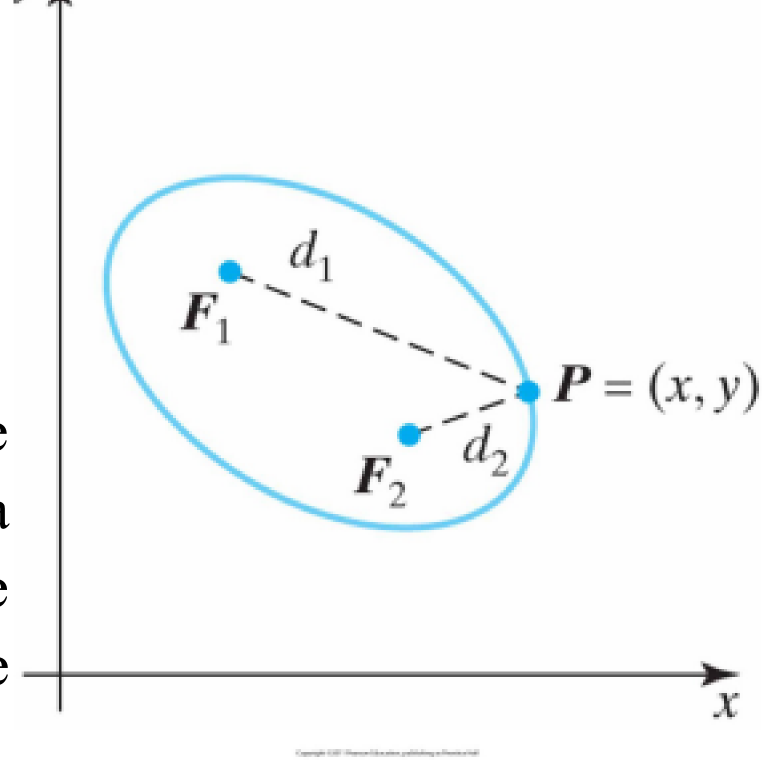
SAVAR, DHAKA

SCAN

CONVERSION III

ELLIPSE

- What is an ellipse?
- An elongated circle
- Describe as a modified circle whose radius varies from a maximum value in one direction to a minimum value in the perpendicular direction
- Major and minor axis
- Two foci



generated about foci F_1 and F_2

AN

ELLIPSE (CONT.)

- Properties of ellipse:

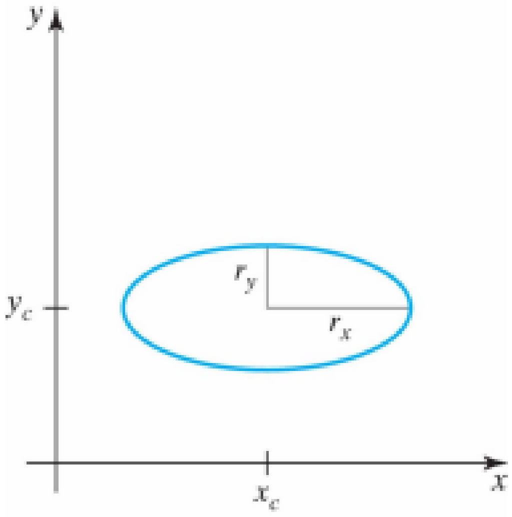
  $d_1 + d_2 = constant$

- In terms of focal coordinates $\boldsymbol{F_1} = (x_1, y_1)$ and $\boldsymbol{F_2} = (x_2, y_2)$

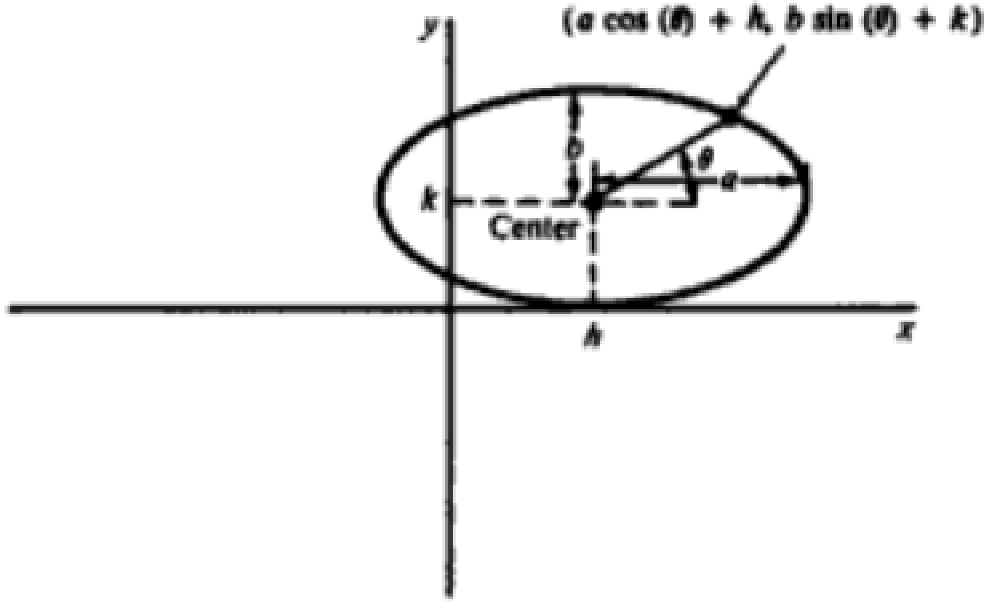
$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2}=constant$$

- Squaring, isolating remaining radical, and squaring again → General ellipse equation

  $Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$



y
r_y
y_c
r_x
x_c
x



Polynomial method:

$((x-x_c)/r_x)^2+((y-y_c)/r_y)^2=1$ where

(x,y) = current coordinates (x_c,y_c) = ellipse center

r_x = length of major axis

r_y = length of minor axis

$y = r_y * \text{sqrt}(1-((x-x_c)/r_x)^2)+y_c$ •

x is incremented from x_c to r_x

Trigonometric method: $x=x_c+r_x\cos\theta$

$y=y_c+r_x\sin\theta$

where

(x,y) = current coordinates (x_c,y_c) = ellipse center r_x = length of major axis r_y = length of minor axis θ = current angle

MIDPOINT ELLIPSE ALGORITHM

- General procedure (same like circle):

    1. Determine curve positions for an ellipse with center at *(0, 0)* (origin)

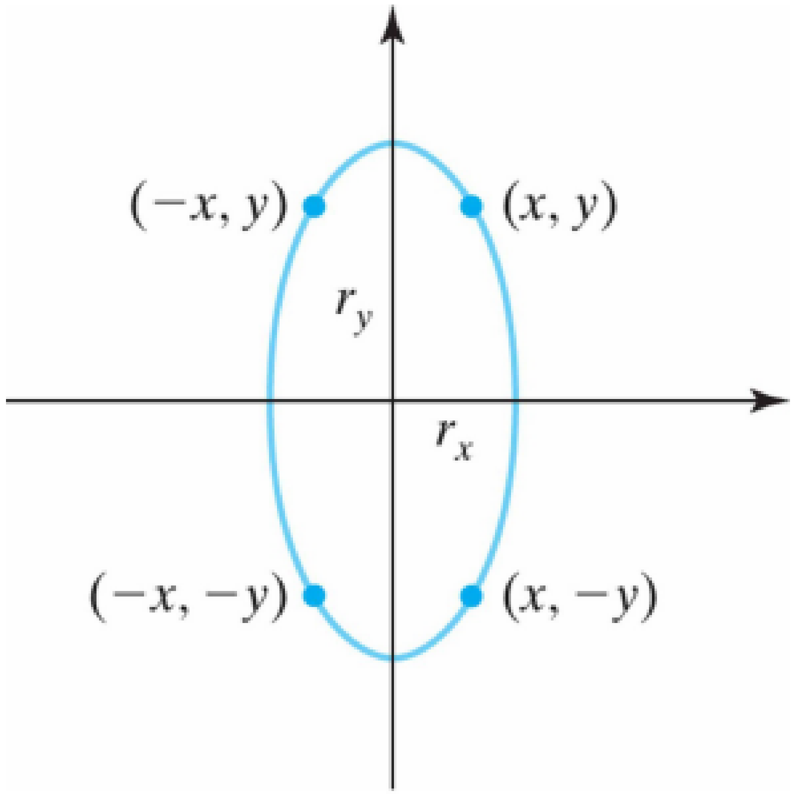
    2. Move to proper position, i.e. add x_c and y_c

    3. If required perform rotation

    4. Use decision parameter to determine closest pixel

    5. For other 3 quadrants use symmetry

SYMMETRY OF AN ELLIPSE



• Calculation of a point (*x, y*) in one quadrant yields the ellipse points shown for the other three quadrants.

MIDPOINT ELLIPSE ALGORITHM (CONT.)

- Ellipse function:

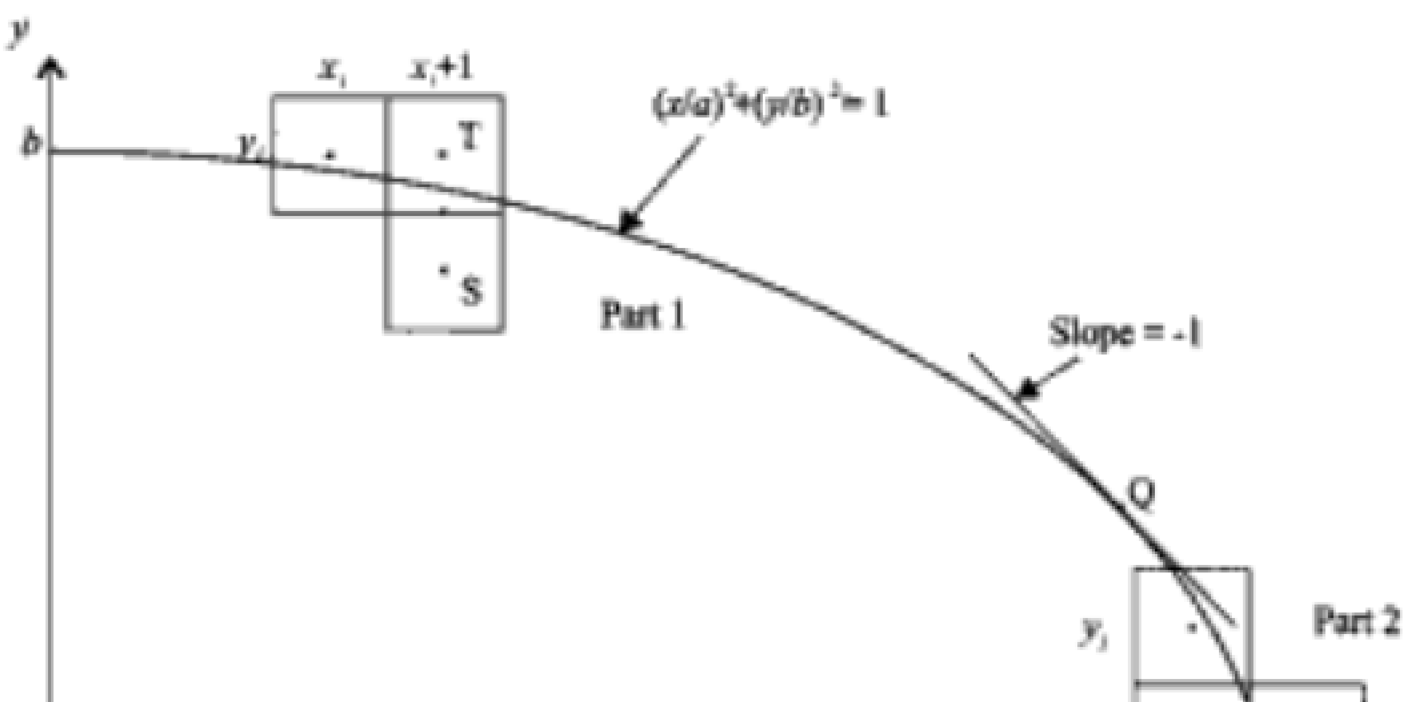
$$f_{ellipse}(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

r If any point *(x,y)* is inside the ellipse,

$f_{ellipse}(x,y)<0$ If any point (x,y) is on the ellipse,

$f_{ellipse}(x,y)=0$ If any point (x,y) is outside the

ellipse, $f_{ellipse}(x,y)>0$



POINT

r_y

r_x

- Divide the elliptical curve from $(0,r_y)$ to $(r_x,0)$ into two parts at point Q where the slope of the curve is -1.

(CONT.)

2 2 2 2 2 2 (,)

$_{ellipse\ y\ x\ x\ y} f\ x\ y = r\ x + r\ y - r\ r$

- The slope of the curve defined by �(�, �) is

$$��/�� = - ��/��$$

where �� and �� are partial derivatives of �(�, �) with respect to � and � respectively.

- Therefore, $�� = 2�_!^{"}�$ and $�� = 2�_\#^{"}�$

$$��/�� = -2�_!^{"}�/2�_\#^{"}�$$

- At the boundary between region 1 and region 2 $��/�� = -1.0$

$$���^{�}� = ���^{�}�$$

REGION 1

To determine the next position

along the ellipse path by evaluating the decision parameter at this mid point

$$p1_k = f_{ellipse}(x_{k+1}, y_k - 1/2)$$

$$= r_y^2 (x_{k+1})^2 + r_x^2 (y_k - 1/2)^2 - r_x^2 r_{y2}$$

If $p1_k < 0$,

Midpoint is inside the ellipse

Otherwise the midpoint is outside

Next sampling position ($x_{k+1}+1 = x_k+2$) the decision parameter for region 1 is calculated as

$$p1_{k+1} = f_{ellipse}(x_{k+1} + 1, y_{k+1} - ½)$$

$$= r_y^2 [(x_{k+1}) + 1]^2 + r_x^2 (y_{k+1} - ½)^2 - r_x^2 r_{y2}$$

OR $p1_{k+1} = p1_k + 2 r_y^2 (x_{k+1}) + r_y^2 + r_x^2 [(y_{k+1} - ½)^2 - (y_k - ½)^2]$

REGION 1…

- We can find a recursive expression for the next decision parameter as:

$$\text{�}1_{!"\#} = \text{�}1_{!} + 2\text{�}_{\$}{}^{\%}(\text{�}_{!} + 1) + \text{�}_{\$}{}^{\%} + \text{�}_{\&}{}^{\%}\,(\text{�}_{!"\#} - \tfrac{1}{2})^{\%} - (\text{�}_{!} - \tfrac{1}{2})^{\%}$$

- We can obtain the initial decision parameter as:

$$\text{�}1_{'} = \text{�}_{\$}{}^{\%} - \text{�}_{\&}{}^{\%}\text{�}_{\$} + \tfrac{1}{4}\text{�}_{\&}{}^{\%}$$

$$p1_{k+1} = \begin{cases} \text{�}1_{\text{�}} + 2\text{�}_{\$}{}^{\%}\text{�}_{!"\#} + \text{�}_{\$}{}^{\%}, & \text{if } \text{�}1_{!} < 0 \\ \text{�}1_{\text{�}} + 2\text{�}_{\$}{}^{\%}\text{�}_{!"\#} + \text{�}_{\$}{}^{\%} - 2\text{�}_{\&}{}^{\%}\text{�}_{!"\#}, & \text{if } \text{�}1_{!} \geq 0 \end{cases}$$

REGION 2

- For this region, the decision parameter is evaluated as

$$p2_k = f_{ellipse}(x_k + ½ , y_k - 1)$$

$$= r_y^{2} (x_k + \tfrac{1}{2})^2 + r_x^{2} (y_k - 1)^2 - r_x^{2} r_y2$$ •

$$p2_{k+1} = f_{ellipse}(x_{k+1} + \tfrac{1}{2}, y_{k+1} - 1)$$

$$= r_y^{2}(x_{k+1} + \tfrac{1}{2})^{2} + r_x^{2} [(y_{k+1} - 1) - 1]^2 - r_x^{2} r_y2$$ Or

$$p2_{k+1} = p2_k - 2 r_x^{2}(y_{k-1}) + r_x^{2} + r_y^{2} [(x_{k+1} + \tfrac{1}{2})^2 - (x_k + \tfrac{1}{2})^2]$$

13

DECISION

PARAMETER

- We can find a recursive expression for the next decision parameter as:

�2_!"# = �2_! − 2�_$^%(�_! − 1) + �_$^% + �_&^% (�_!"# + 1 2)^% − (�_!

+ 1 2)^%

- We can obtain the initial decision parameter as: �2_' = �_$^%(�_' + 1 2)^% + �_&^%(�_' − 1)^% − �_&^%�_$%

*p2*_{*k+1*}= -�2_� + 2�_$^%�_!"# − 2�_&^%�_!"# + �_&^%, if �2_! < 0 �2_� − 2�_&^%�_!"# + �_&^%, if �2_! ≥ 0

MIDPOINT ELLIPSE ALGORITHM

(CONT.): REGION 1-REGION 2?

- When to move from region 1 to region 2?
- Answer: decide on slope
- At the boundary $dy / dx = -1.0$
- If $2r_y^2 x \ge 2r_x^2 y$ move from region 1 to region 2

MIDPOINT ELLIPSE ALGORITHM (CONT.)

1. Input radii r_x, r_y and ellipse center (x_c, y_c), and obtain the first point on the origin as $(x_0, y_0) = (0, r_y)$.

2. Calculate the initial value of the decision parameter in region 1 as

�1' = �$ % − �& % �$ + 1 4 �& %

3. At each x_k in region 1, from $k=0$, perform the following test: • If $p1_k<0$, next point to plot along the ellipse centered on $(0,0)$ is (x_{k+1}, y_k) 2 1 k 1 1 k 2 y k y p = p + r x + r + +

and 1

2

• Otherwise, next point to plot is $(x_k+ 1, y_k - 1)$

and

2 1 k 1 1 k 2 y k 2 x k y p = p + r x − r y + r + + +

2 2

1 1

2 2 2 2 , 2 2 2 y k y k y x k x k X r x = r x + r r y = r y − r + +

With

2 2 2

1

2 2

1

$^{2}\,^{2}\,2 \geq 2$

And continue until

$r\ x\ r\ y\ _{y\ x}$

16

MIDPOINT ELLIPSE ALGORITHM

parameter in

�2,

=

�$% �' + 1 2% + �&% � − 1 % + �&% �$%

(x_0, y_0) is last calculated point from region 1

5. At each y_k position in region 2, starting at $k=0$, perform the following test: • If *$p2_k>0$*, next point to plot along the ellipse centered on *(0,0)* is *(x_k, y_k-1)* 2 2_k

1 2_k 2 y k y p = p + r x + r + +

1

and

2

• Otherwise, next point to plot is *(x_k+1, y_k-1)*

and

2 2_k 1 2_k 2 y k 2 x k x p = p + r x − r y + r + + +

2 2

1 1

And continue until $y = 0$

MIDPOINT ELLIPSE ALGORITHM

(CONT.)

6. Determine symmetry points in the other three quadrants.

7. Move each calculated pixel position *(x, y)* onto the elliptical path centered at (x_c, y_c) and plot the coordinate values: $x = x + x_c$, $y = y + y_c$

EXAMPLE: MIDPOINT ELLIPSE

DRAWING

19

EXAMPLE: MIDPOINT ELLIPSE

DRAWING (CONT.)

20

EXAMPLE: MIDPOINT ELLIPSE

DRAWING (CONT.)

DRAWING (CONT.)

Pixel positions along an elliptical path centered on the origin with $r_x = 8$ and $r_y = 6$, using the midpoint algorithm to calculate locations within the first quadrant

FILL-AREA ALGORITHMS

• Standard output primitives – solid, pattern, hollow • Fill primitive – solid circle, rectangle, triangle, … • Two ways of filling:

- Find where each scanline overlaps area (scan-line fill)
- Start from interior position and paint outward until boundary is reached

• Used in general purpose packages and paint programs.

AREA

FILLING

- General idea:

  1. Determine boundary intersection

  2. Fill interior

SCAN-LINE POLYGON-FILL ALGORITHM

- For convex polygons.
  - Determine the intersection positions of the boundaries of the fill region with the screen scan lines.

B E D

y

F

C

25

A

SCAN-LINE POLYGON-FILL

ALGORITHM (CONT.)

- For convex polygons.
  - Pixel positions between pairs of intersections between scan line and edges are filled with color, including the intersection pixels.

B

y C

F

E D

A

Scanline Polygon fill Algorithm (cont.)

- For concave polygons.
  - Scan line may intersect more than once:
    - Intersects an even number of edges
    - Even number of intersection vertices yields to pairs of intersections, which can be filled as previously

G C

B A D

y

E

F

Scan-line Polygon fill Algorithm (cont.)

- For concave polygons.
  - Scan line may intersect more than once:
    - Intersects an even number of edges
    - Even number of intersection vertices yields to pairs of intersections, which can be filled as previously

A

y

G

F E

C

B

D

Scan-line Polygon fill Algorithm (cont.)

- For concave polygons.
  - Scan line may intersect more than once:
    - Intersects an **odd** number of edges
    - Not all pairs are interior: (3,4) is not interior

C B

A

y 1 2 3 4 5 G

D

F

E

Scan line Polygon fill Algorithm

(cont.)

Interior pixels along a scan line passing through a polygon fill area.

Scan line Polygon fill Algorithm

(cont.)

Intersection points along scan lines that intersect polygon vertices. Scan line *y* generates an odd number of intersections, but scan line *y’* generates an even number of intersections that can be paired to identify correctly the interior pixel spans.

Scan-line Polygon fill Algorithm (cont.)

- For concave polygons.
  - Generate 2 intersections when at a local minimum, else generate only one intersection.
  - Algorithm to determine whether to generate one intersection or 2 intersections.
    - If the y-coordinate is monotonically increasing or decreasing, decrease the number of vertices by shortening the edge.
    - If it is not monotonically increasing or decreasing, leave the number of vertices as it is.

Scan-line Polygon-fill Algorithm (cont.)

scan line

y+1

y

1

The y-coordinate of the upper
endpoint of the next edge
is decreased by 1.

dinate of the upper
of the current edge
creased by 1.
y-1 y decreasing:
decrease by

Scan-line Polygon fill Algorithm (cont.)

- In previous slide edges are shortened and vertices are counted only once because edges are adjusted

- But how to do it?

- Answer: using coherence properties

Scan-line Polygon fill Algorithm (cont.)

- Coherence properties: certain properties of one part of the scene related to properties of other parts, e.G. Slope
- Sequential fill algorithm with incremental coordinate calculations

USING COHERENCE

PROPERTIES

Two successive scan lines intersecting a polygon boundary. At both positions same slopes

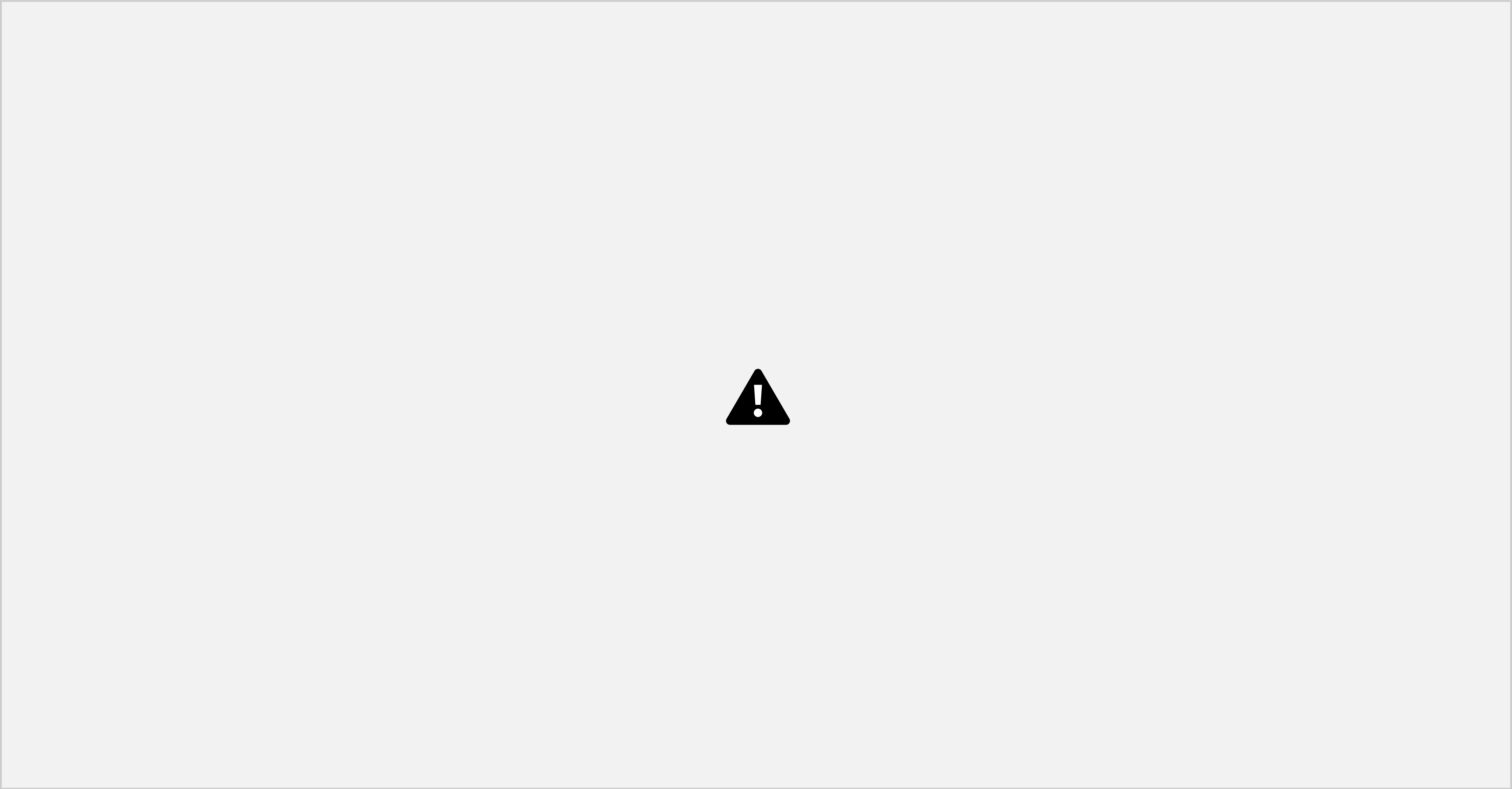
FILL-AREA ALGORITHMS

- Polygon fill-in algorithm:
    1. Store the edges in a sorted edge table where each entry corresponds to a scan line (sorted on the smallest y value on each edge)

       For sorting, e.g. Bucket sort

    2. Shorten the edges that have vertex-intersection issues by processing scan lines from bottom of polygon to top (active edge list) 3. For each scan line, fill-in the pixel spans for each pair of x intercepts.

EXAMPLE



A polygon and its sorted edge table, with edge DC shortened by one unit in the *y* direction.

OTHER FILL-AREA ALGORITHMS

- For areas with irregular boundaries
  - Boundary-fill algorithm

    start at an inside position and paint color point by point until reaching the boundary (of different color):

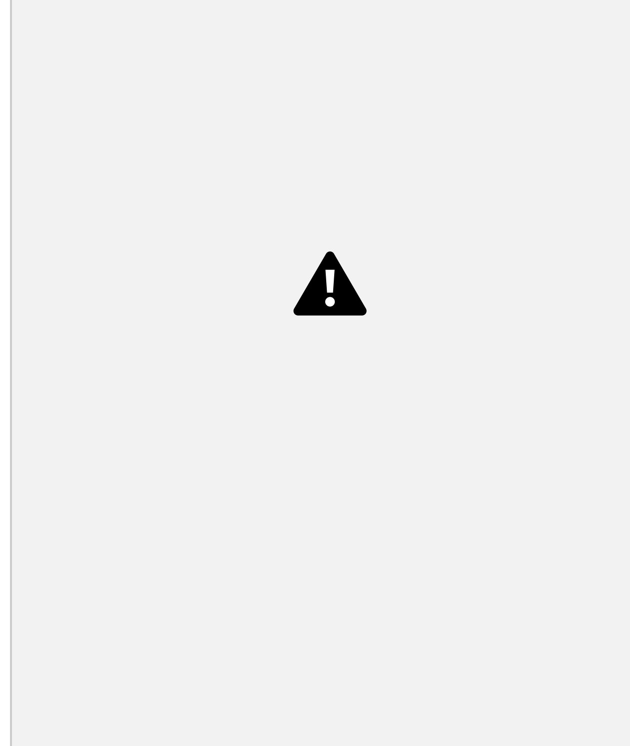
```
void boundaryFill4 (int x, int y, int fillColor, int borderColor)
{
int interiorColor;
/* Set current color to fillColor, then perform following oprations. */
getPixel (x, y, interiorColor);
if ((interiorColor != borderColor) && (interiorColor != fillColor)) {
setPixel (x, y); // Set color of pixel to fillColor.
boundaryFill4 (x + 1, y , fillColor, borderColor);
boundaryFill4 (x - 1, y , fillColor, borderColor);
boundaryFill4 (x , y + 1, fillColor, borderColor);
boundaryFill4 (x , y - 1, fillColor, borderColor)
}
}
```

BOUNDARY-FILL ALGORITHM

- General idea:

  1. Start at position *(x, y)*
  2. Test color of neighboring pixels
  3. If neighbor pixel’s color is not boundary color, change color
  4. Proceed until all pixels processed

NEIGHBORS?



(a) 4-connected area

(b) 8-connected area

Hollow circles represent pixels to be tested

from the current test position, shown as a

solid color.

BOUNDARY FILL: 4-CONNECTED VS.

8- CONNECTED

✦ *Start point*

• 4-connected • 8-connected

Boundary fill across pixel spans for a 4-connected area:

(a) Initial scan line with a filled pixel span, showing the position of the initial point (hollow) and the stacked positions for pixel spans on adjacent scan lines.

(b) Filled pixel span on the first scan line above the initial scan line and the current contents of the stack.

(c) Filled pixel spans on the first two scan lines above the initial scan line and the current contents of the stack.

(d) Completed pixel spans for the upper-right portion of the defined region and the remaining stacked positions to be processed. 43

OTHER FILL-AREA ALGORITHMS

(CONT.)

- For areas with irregular boundaries
  - Flood-fill algorithm

start at an inside position and reassign all pixel values currently set to a given interior color with the desired fill color.

```
void floodFill4 (int x, int y, int fillColor, int interiorColor)
{
int color;
/* Set current color to fillColor, then perform following operations. */
getPixel (x, y, color);
if (color = interiorColor) {
setPixel (x, y); // Set color of pixel to fillColor.
floodFill4 (x + 1, y, fillColor, interiorColor);
floodFill4 (x - 1, y, fillColor, interiorColor);
floodFill4 (x, y + 1, fillColor, interiorColor);
floodFill4 (x, y - 1, fillColor, interiorColor)
}
}
```

- *Boundary color*
- *Interior point (x, y)*
- *Fill color*

45

IMPLEMENTATION OF ANTIALIASING

- (Remember) aliasing: ınformation loss due to low-frequency sampling (undersampling)
- How to avoid it?
- Answer: using nyquist sampling frequency/rate
- Or
- Answer: using nyquist sampling interval

NYQUIST?

- Harry Nyquist: 7. February 1889 in nilsby, sweden; † 4. April 1976 in harlingen, texas
- Claude Elwood Shannon: 30. April 1916 in petoskey, michigan; † 24. Februar 2001 in medford, massachusetts
- Nyquist-Shannon-sample theorem: set the sampling frequency to at least twice that of the highest frequency occurring in the object
- Nyquist sampling frequency: $f_s = 2f_{max}$
- Nyquist sampling interval: $\Delta x = \Delta x_{cycle} / 2$

SAMPLING

Sampling the periodic shape in (a) at the indicated positions produces the aliased lower-frequency representation in (b).

ANTIALIASING METHODS

1. Supersampling (postfiltering): sample at high resolution (at subpixel level) but show on lower resolution

2. Area sampling (prefiltering): antialiasing by computing overlap areas

3. Pixel phasing (for raster objects): shift display location of pixel areas

SUPERSAMPLING

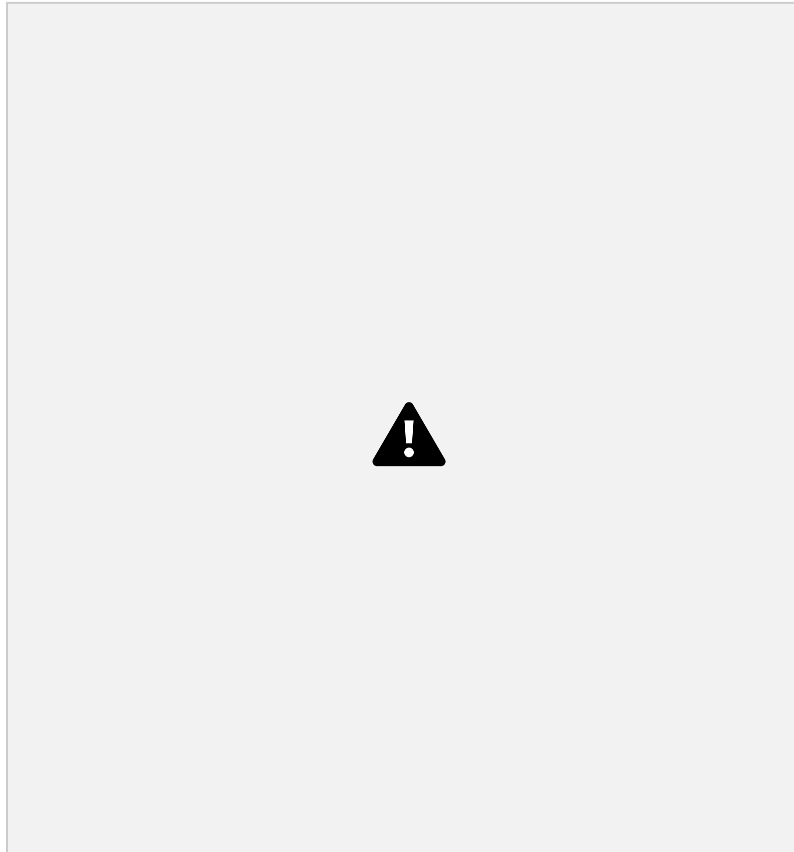
- General idea (for straight line segments):

  1. Divide each pixel into a number subpixels

  2. Count number of subpixels overlapping the line path

  3. Set intensity of each pixel to a value proportional to subpixel count

EXAMPLE: SUPERSAMPLING



Supersampling subpixel positions along a straight-line segment whose left endpoint is at screen coordinates (10, 20).

WEIGHTING SUBPIXELS

• Multiply each pixel with a weighted mask

1 2 1

2

2 4

1 2 1

Subpixel

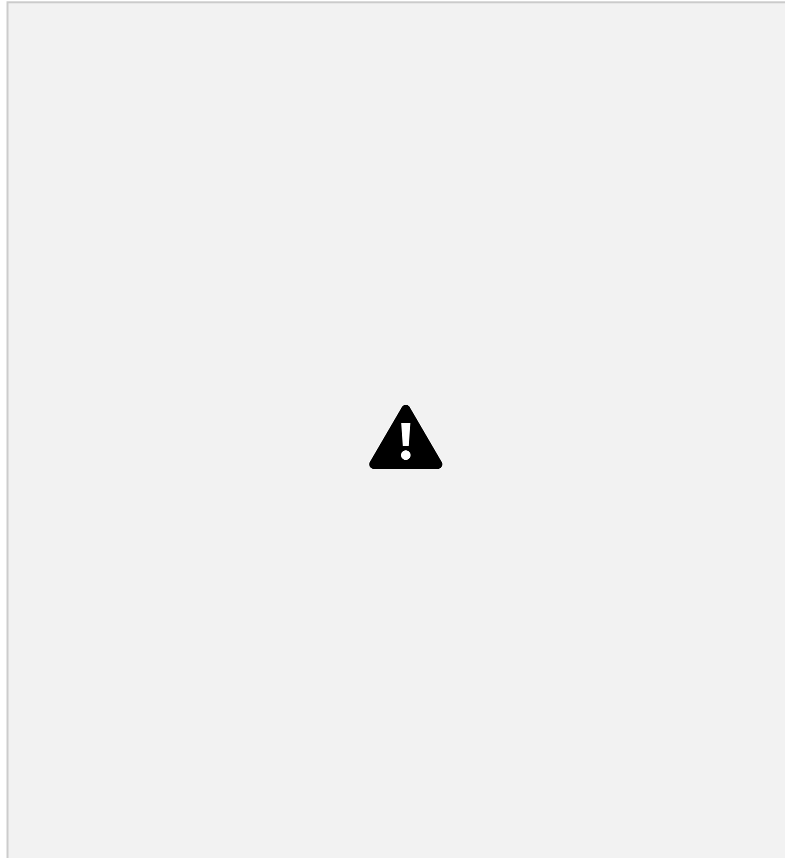
weighting

mask

AREA SAMPLING

- General idea (for straight line segments):
  - Set pixel intensities proportional to the area of overlap of the pixel with the finite-width line

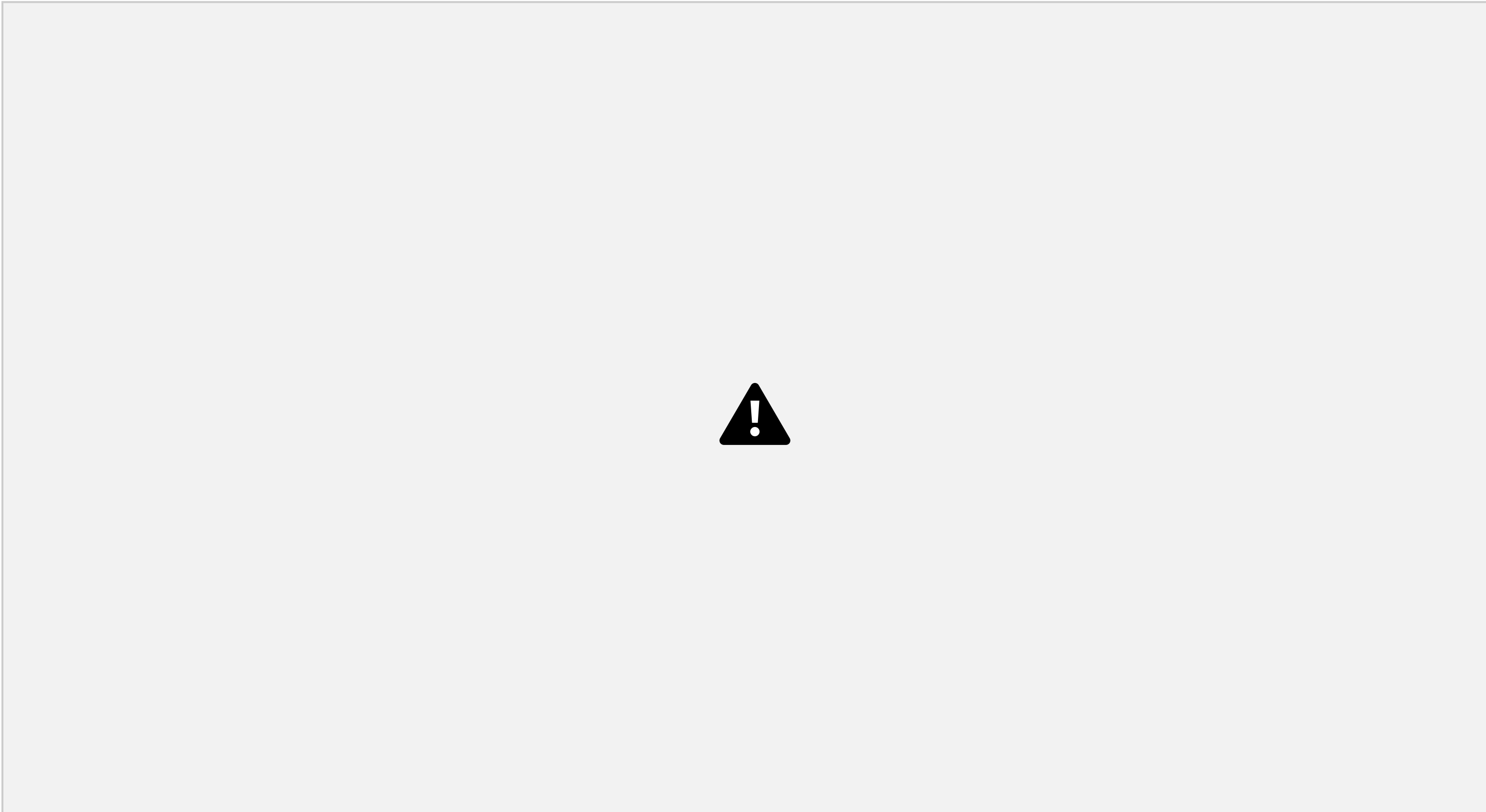
EXAMPLE: AREAS SAMPLING



Supersampling subpixel positions in relation to the interior of a line of finite width.

FILTERING TECHNIQUES

- Similar to weighting but here continuous weighting surface (filter function)
- Multiply each pixel with the function
- What kind of function?
- Answer: next slide



Filters used to antialias line paths. The volume of each filter is normalized to 1.0, and the height gives the relative weight at any subpixel position.