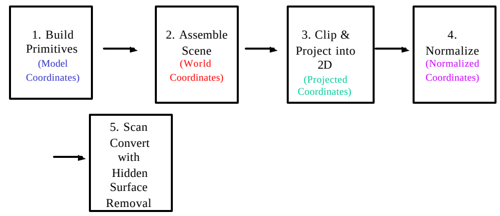
Computer Graphics

Z Sweedyk
Lecture 9

Graphics Pipeline 5



Outline

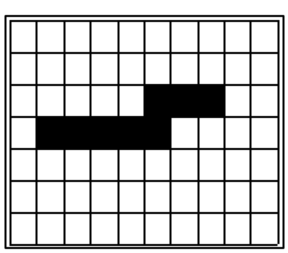
- Today : Scan conversion
  - Lines
  - Polygons
  - Filled polygons
- Next time: Hidden surface removal

Line Segments

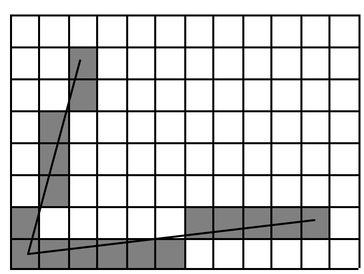
- **Scan converting line segments**
  - Naïve algorithm
  - Midpoint algorithm
  - Bresenham's algorithm

Scan Converting Line Segments

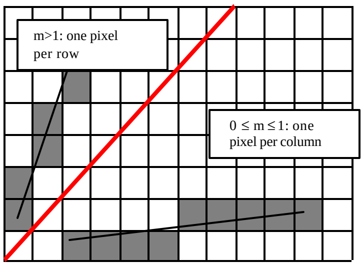
Which pixels should be on?



1-pixel wide lines

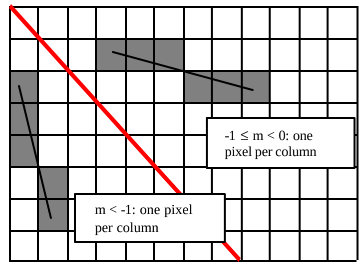


1-pixel wide lines: m≥0



10/16/01 CS-155 Graphics 7

1-pixel wide lines: m<0



10/16/01 CS-155 Graphics 8

Scan Conversion

- Input: Endpoint Pixels
- Output: Pixels to turn on for a 1-pixel wide line segment

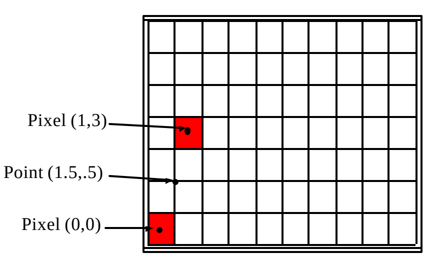
10/16/01 CS-155 Graphics 9

For today...

- A pixel is a little square!

10/16/01 CS-155 Graphics 10

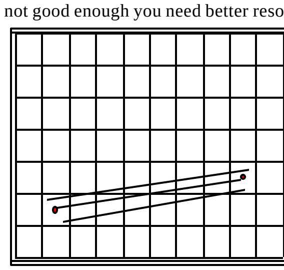
Points and Pixels



10/16/01 CS-155 Graphics 11

Equivalent Ideal Lines Segments

If that's not good enough you need better resolution!!!!!



10/16/01 CS-155 Graphics 12

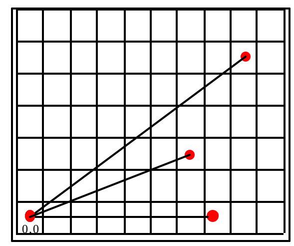
Claim

All we have to do is solve the scan conversion problem for the special case where

1. $0 \le m \le 1$
2. $x_0 = y_0 = 0$

We'll prove this later … first we'll devise an algorithm for the special case.

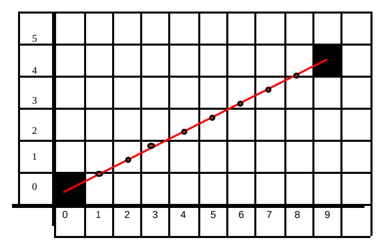
Ideal lines we'll scan convert



Line Segments

- **Scan converting line segments**
  - **Naïve algorithm (special case)**
  - Midpoint algorithm
  - Bresenham's algorithm

Endpoints: (0,0) and (9,4) $y=mx$, $m = 4/9$



Naïve Algorithm
(Case: $x_0=y_0=0$, $0 \le m \le 1$)

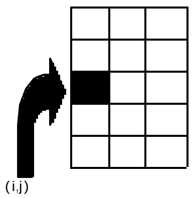
```
SpecialCaseNaive(int x1 , int y1 )
  int current x=0
  float current_y=0,
  float m = (float) y1 / (float) x1
  while (current_x ≤ x1 )
      DrawPixel(current_x,round(current_y))
      current_x += 1, current_y += m
```

Floating Point

Line Segments

- **Scan converting line segments**
  - Naïve algorithm
  - **Midpoint algorithm**
  - Bresenham's algorithm
- Clipping line segments
  - Scissoring
  - Analytical clipping
- Antialiasing

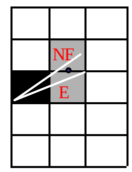
Let's try to do better!



•Suppose we've just drawn the (i,j) pixel. How do we choose the next pixel?

How do we choose next pixel?

$y=mx$, $0 \le m \le 1$



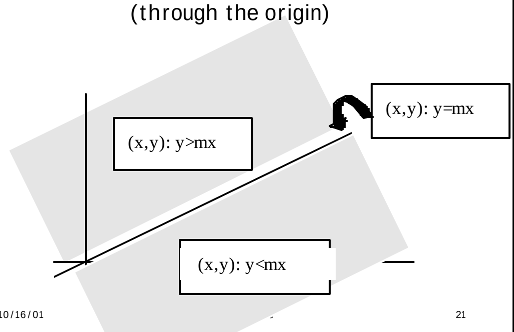
•The options are E and NE

•If midpoint lies below the ideal line: go NE

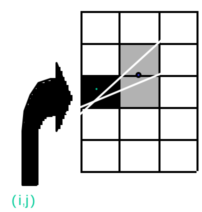
•Else: go E

Line in the plane

(through the origin)



The Test



$u=i+1$

$v=j+1/2$

Ideal line: $y=mx$½

If $v < mu$ then go NE

Else go E

Using Midpoint Test

(Case: $x_0=y_0=0$ and $0 \le m \le 1$)

```
m = y1/x1
i=0, j=0
while i<x1
    write-pixel(i,j)
    if j+1/2 ½< m·(i+1)
        i+=1, j+=1     // go NE
    else i+=1          // go E
```

Floating Point

Modified Test

Is $j + 1/2½ < m(i+1)$?

Is $j + 1/2½ < (y_1 / x_1) \cdot (i+1)$?

Is $x_1(2j + 1) < 2y_1(i+1)$?

Important: x_1 and y_1 are integers!

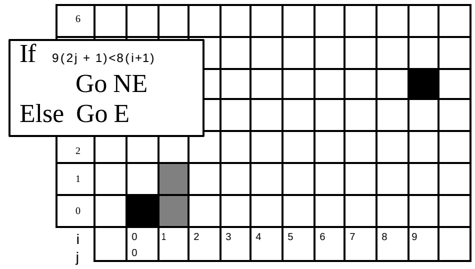
Midpoint Algorithm

(Case: $x_0=y_0=0$ and $0 \le m \le 1$)

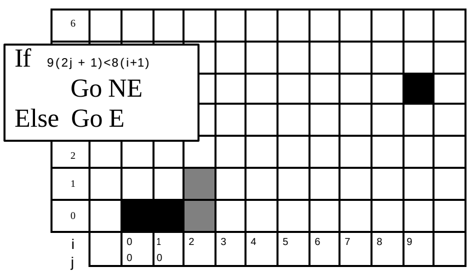
```
SpecialCaseMidpoint(x1,y1)

  i=0, j=0
  while current_i<x1
       writepixel(I,j)
       if x1(2j+1)< 2y1(i+1)
               i+=1, j+=1     // go NE
       else i+=1              // go E
```


Endpoints (0,0) & (9,4):



Endpoints (0,0) & (9,4):



What wrong now?

```
SpecialCaseMidpoint(x1,y1)

  i=0, j=0
  while current_i<x1
       writepixel(I,j)
       if x1(2j+1)< 2y1(i+1)
               i+=1, j+=1     // go NE
       else i+=1              // go E
```

Multiplication

Line Segments

- **Scan converting line segments**
  - Naïve algorithm
  - Midpoint algorithm
  - **Bresenham's algorithm**
- Clipping line segments
  - Scissoring
  - Analytical clipping
- Antialiasing

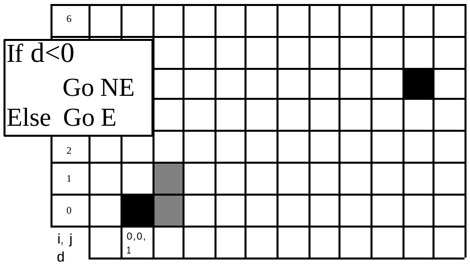
Sleight of hand

```
if x1(2j+1)< 2y1(i+1)
   go NE
else go E
```

```
d = x1(2j+1)-2y1(i+1)    if d<0
   go NE
else go E
```

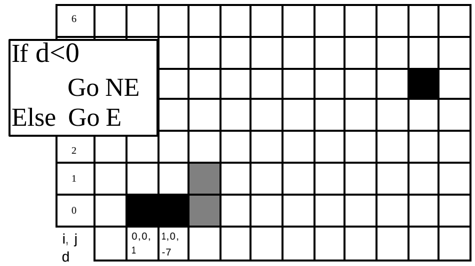
Endpoints (0,0) & (9,4):

$d=9(2j+1)-8(i+1)$



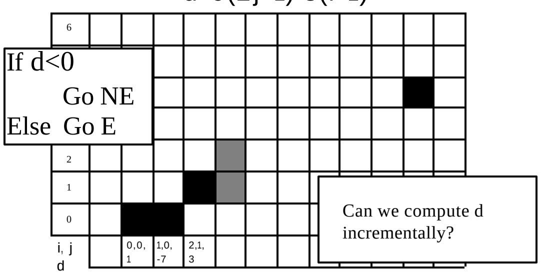
Endpoints (0,0) & (9,4):

$d=9(2j+1)-8(i+1)$



Endpoints (0,0) & (9,4):

$d=9(2j+1)-8(i+1)$



Special Case Bresenham's

($x_0=y_0=0$ and $0 \le m \le 1$)

```
SpecialCaseBresenham's(x1,y1)

  i=0, j=0
  d = x1-2y1
  while i<x1
      write-pixel(I,j)
      if d<0
              i+=1, j+=1, d+=2(x1-y1)
      else
              i+=1, d-=2y1
```

Do as addition

Yay!

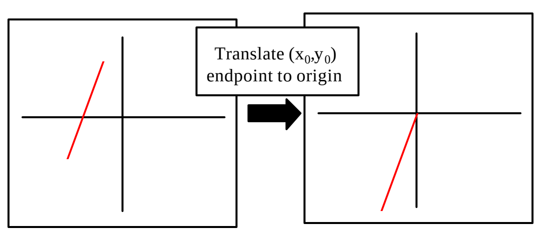
Claim

All we have to do is solve the problem for the special case where

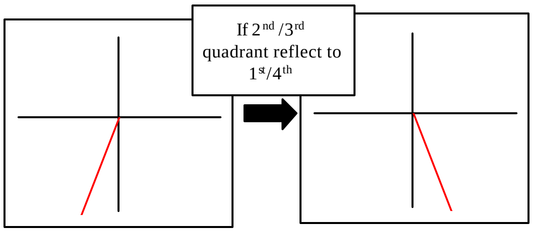
1. $0 \le m < 1$
2. $x_0=y_0=0$

Now we'll prove this claim

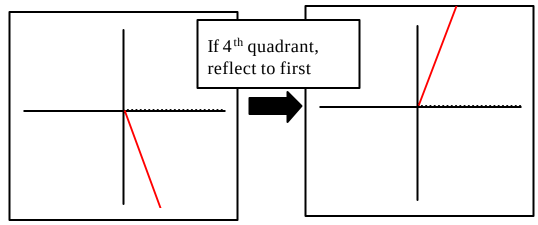
To solve general case:



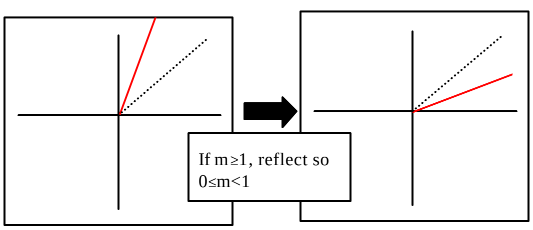
To solve general case:



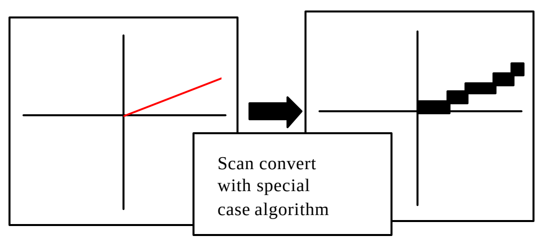
To solve general case:



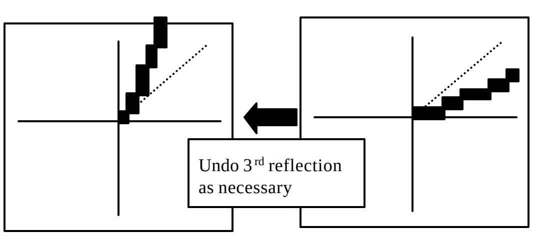
To solve general case:



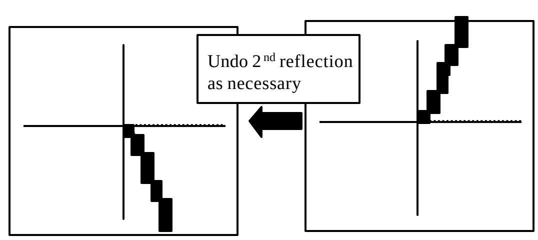
To solve general case:



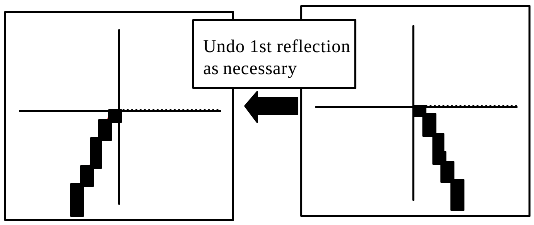
To solve general case:



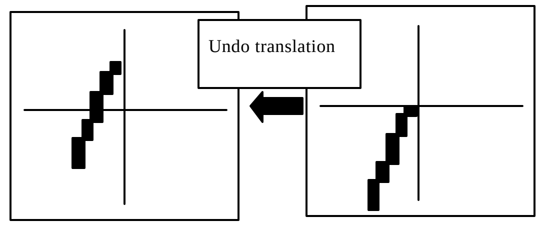
To solve general case:



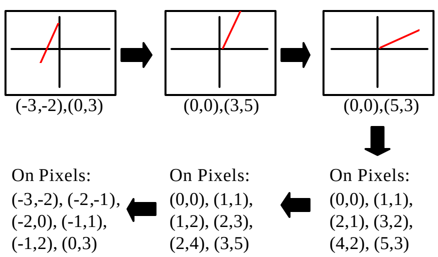
To solve general case:



To solve general case:



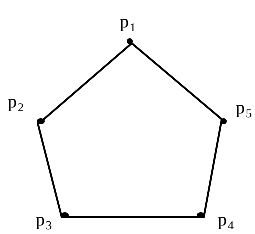
Example



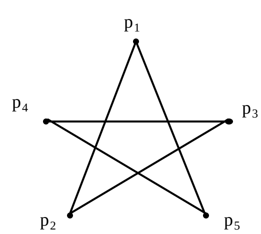
Outline

- Today : Scan conversion
  - Lines
  - Polygons
  - Filled polygons
- Next time: Hidden surface removal

Polygon: p_1,p_2,p_3,p_4,p_5



Polygon: p_1,p_3,p_5,p_2,p_4



Order
Matters

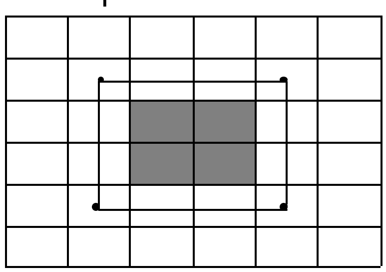
Polygon: Scan Conversion

Polygon(p_1, …, p_n)
For i=1 to n-1
DrawLine(p_i,p_{i+1})
DrawLine(p_n,p_1)

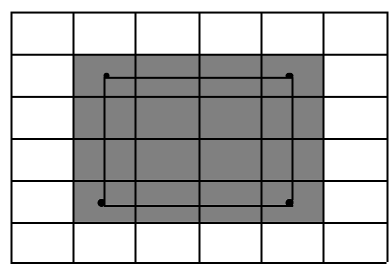
Outline

- Today : Scan conversion
  - Lines
  - Polygons
  - Filled polygons
- Next time: Hidden surface removal

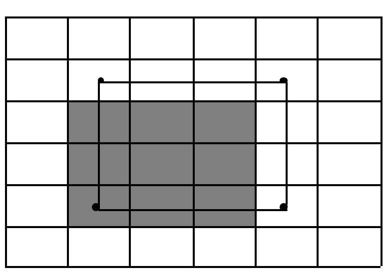
Filled Polygon
Which pixels should be on?



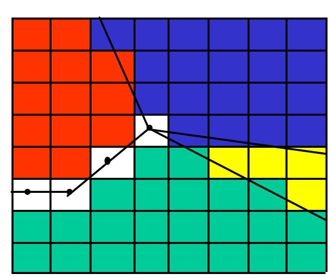
Filled Polygon
Which pixels should be on?



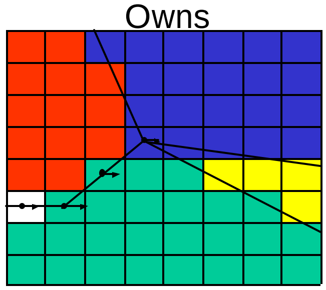
Here we get the same size!



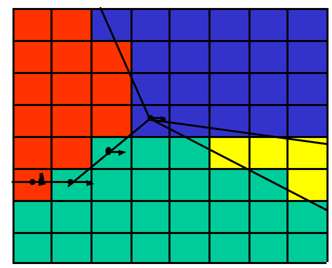
Center Claims



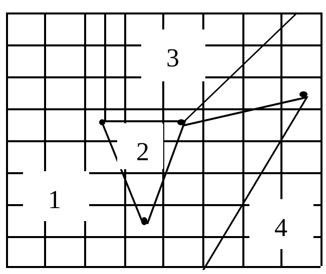
Tie Breaker 1: Entering Owns



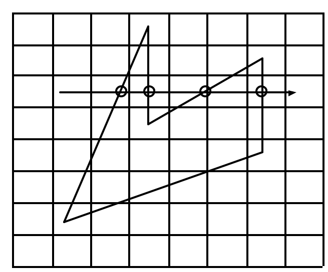
Tie Breaker 2: Up wins



Exercise

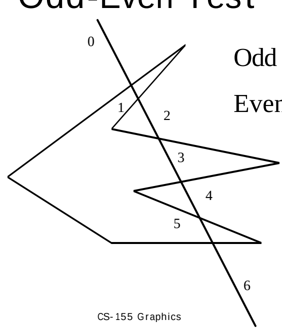


Scan Line Algorithm



1. Compute intersections
2. Order by x-coordinate
3. Use odd-even test to turn on pixels

Odd-Even Test

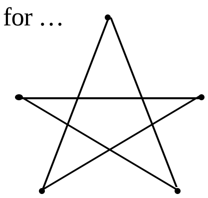


Odd = in

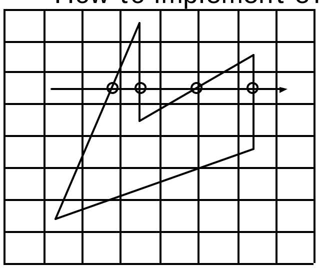
Even = out

Odd-Even Test

May not be what you're looking for … but it's easy to implement



Scan Line Algorithm
How to implement efficiently?



1. Compute intersections
2. Order by x-coordinate
3. Use odd-even test to turn on pixels

Key ideas

- Let S be the set of line segments that intersect scan line i. The set of lines that intersect scan line i+1 is:
  S+new-done
- Suppose that line L=(m,b) intersects scan line i at (x,i). If L intersects scan line i+1 it does so at:
  (x+1/m,i+1)

Data Structures

- Edge Table
- Active Edge Table

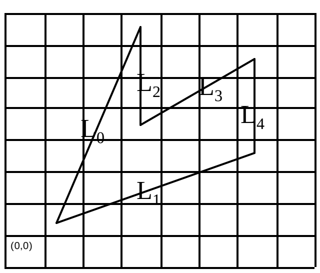
Edge Table (ET)

Yval	Lines
6	
5	
4	
3	
2	
1	
0	

Line Segments "beginning" at scan line 3

Example: Edge Table

Yval	Lines
6	-
5	-
4	L_2,L_3
3	L_4
2	-
1	L_0,L_1
0	-



Example: Edge Table

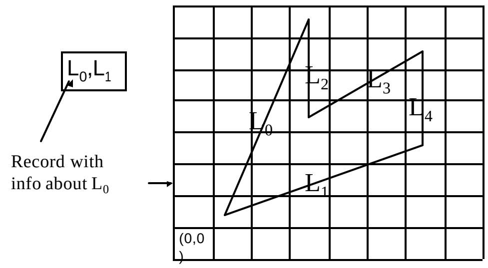
Yval	Lines
6	-
5	-
4	L_2,L_3
3	L_4
2	-
1	L_0,L_1
0	-

Record with info about L_2

Active Edge Table (AET)

List of the Line segments intersecting current scan line

Example: Active Edge Table



Scan Line Algorithm

Build ET
Yval=-1
Initialize AET=φ
Repeat until ET and AET are empty:
- Yval ++
- Update info on line segments
- Add ET[Yval] to AET
- Remove lines from AET that are "done"
- Sort lines in AET by x-intercept at y=Yval
- Choose pixels based on odd-even test

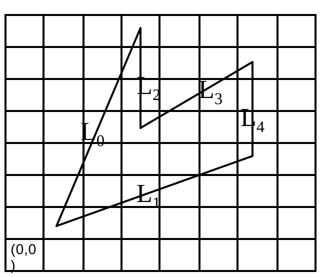
Scan Line Algorithm

Build ET
Yval=-1
Initialize AET=φ
Repeat until ET and AET are empty:
- Yval ++
- Update info on line segments
- Add ET[Yval] to AET
- **Remove lines from AET that are "done"**
- Sort lines in AET by x-intercept at y=Yval
- Choose pixels based on odd-even test

Line Record

Field 1: ymax

line	ymax
L_0	7
L_1	3
L_2	7
L_3	6
L_4	6



Scan Line Algorithm

Build ET
Yval=-1
Initialize AET=φ
Repeat until ET and AET are empty:
- Yval ++
- Update info on line segments
- Add ET[Yval] to AET
- **Remove lines from AET when Yval=ymax**
- Sort lines in AET by x-intercept at y=Yval
- Choose pixels based on odd-even test

Scan Line Algorithm

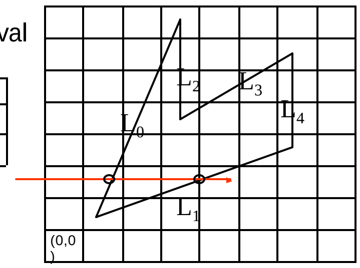
Build ET
Yval=-1
Initialize AET=φ
Repeat until ET and AET are empty:
- Yval ++
- Update info on line segments
- Add ET[Yval] to AET
- Remove lines from AET when Yval=ymax
- **Sort lines in AET by x-intercept at y=Yval**
- Choose pixels based on odd-even test

10/16/01 CS-155 Graphics 73

Line Record: CurrentYval=2

Field 2. Xval

line	xval
L_0	4/3
L_1	7/2



10/16/01 CS-155 Graphics 74

Scan Line Algorithm

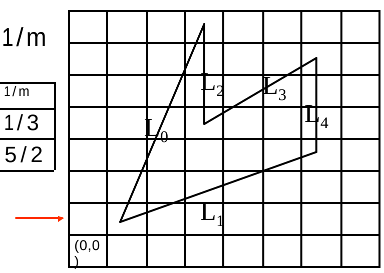
Build ET
Yval=-1
Initialize AET=φ
Repeat until ET and AET are empty:
- Yval ++
- **Update info on line segments**
- Add ET[Yval] to AET
- Remove lines from AET when Yval=ymax
- **Sort lines in AET by x-intercept at y=Yval**
- Choose pixels based on odd-even test

10/16/01 CS-155 Graphics 75

Line Record: CurrentYval=2

Field 3. 1/m

line	xval	1/m
L_0	1	1/3
L_1	1	5/2

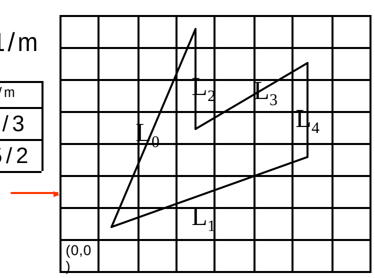


10/16/01 CS-155 Graphics 76

Line Record: CurrentYval=2

Field 3. 1/m

line	xval	1/m
L_0	4/3	1/3
L_1	7/2	5/2



10/16/01 CS-155 Graphics 77

Scan Line Algorithm

Build ET
Yval=-1
Initialize AET=φ
Repeat until ET and AET are empty:
- Yval ++
- **Increment xval by 1/m for each line in AET**
- Add ET[Yval] to AET
- Remove lines from AET when Yval=ymax
- Sort lines in AET by x-intercept at y=Yval
- Choose pixels based on odd-even test

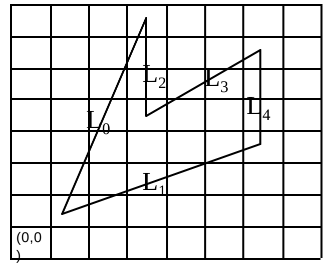
10/16/01 CS-155 Graphics 78

Scan Line Algorithm

Build ET
Yval=-1
Initialize AET=φ
Repeat until ET and AET are empty:
- Yval ++
- Increment xval by 1/m for each line in AET
- Add ET[Yval] to AET
- Remove lines from AET when Yval=ymax
- Sort lines in AET by x-intercept at y=Yval
- **Choose pixels based on odd-even test**

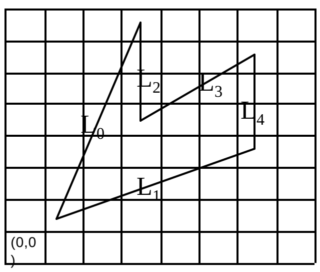
Initialize Line Records

line	ymax	x	1/m
L_0	7	1	1/3
L_1	3	1	5/2
L_2	7	3	0
L_3	6	3	3/2
L_4	6	6	0



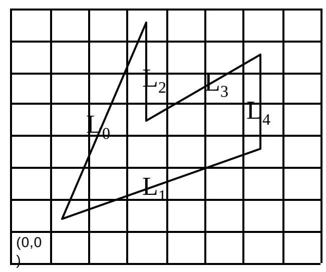
Initialize Edge Table

Yval	Lines
6	-
5	-
4	$L_2 \rightarrow L_3$
3	L_4
2	-
1	$L_0 \rightarrow L_1$
0	-



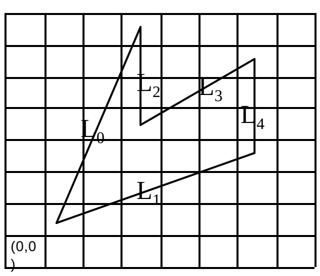
Initialize Active Edge Table

φ



Yval=0

φ

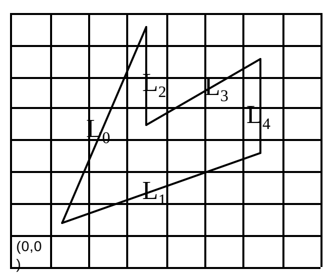


Yval=1

(sorted) AET

line	ymax	x	1/m
L_0	7	1	1/3
L_1	3	1	5/2

Turn on (i,1)
where: $1 \le i < 1$

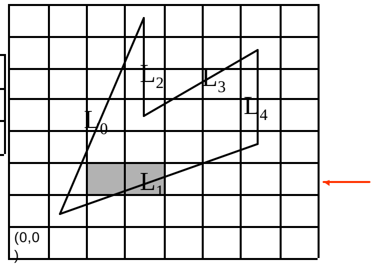


Yval=2

(sorted) AET

line	ymax	x	1/m
L_0	7	4/3	1/3
L1	3	7/2	5/2

Turn on (i,2)
where 4/3≤ i<7/2

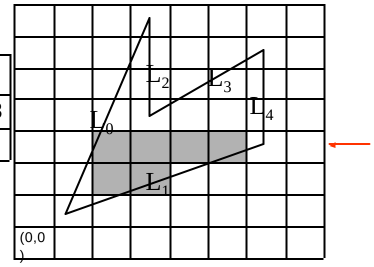


Yval=3

(sorted) AET

line	ymax	x	1/m
L_0	7	5/3	1/3
L_4	6	6	0

Turn on (i,3)
where 5/3 ≤ i<6

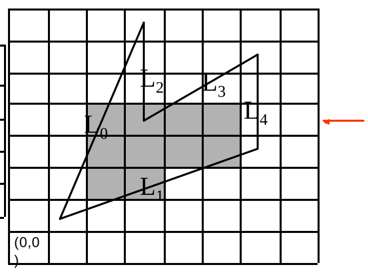


Yval=4

(sorted) AET

line	ymax	x	1/m
L_0	7	2	1/3
L_2	7	3	0
L_3	6	3	3/2
L_4	6	6	0

Turn on (i,4) where 2 ≤ i<3 or 3 ≤i<6

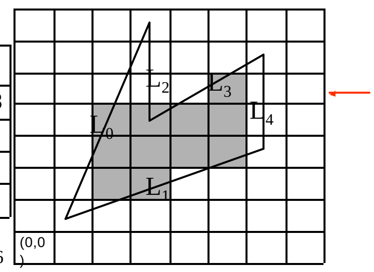


Yval=5

(sorted) AET

line	ymax	x	1/m
L_0	7	7/3	1/3
L_2	7	3	0
L_3	6	9/2	3/2
L_4	6	6	0

Turn on (i,5) where
7/3 ≤ i<3 and 9/2≤i<6

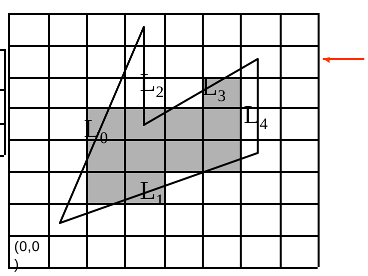


Yval=6

(sorted) AET

line	ymax	x	1/m
L_0	7	8/3	1/3
L_2	7	3	0

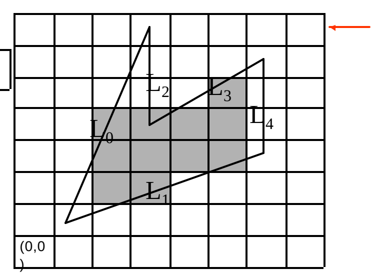
Turn on (i,6) where
8/3 ≤ i<3



Yval=6

(sorted) AET

line	ymax	x	1/m



Claims

- AET always contains an even number of lines
- The algorithm implements the correct tie-breaking rules