Rellenado de polígonos

Introducción

- La tarea de rellenar un polígono puede ser:
  - Decidir que pixels pintar para rellenar un polígono
  - Decidir con que valor pintar el pixel

- Decidir que pixels pintar consiste barrer líneas sucesivas que interceptan a la primitiva rellenando los pixels en bloques que están dentro del polígono de izquierda a derecha

Rectángulos

- Para pintar un rectángulo de un solo color
  - Se pinta una línea de barredura de izquierda a derecha con un mismo valor de *xmin* a *xmax*
  - Las primitivas exploran la coherencia espacial
    - No hay alteracion en las primitivas de un pixl para otro dentro de un mismo bloque
    - De una linea para la siguiente
    - Coherencia de linea (Son iguales de una para la otra)
    - Coherencia de aristas (Para los lados del poligono)

Rectángulos

- Algoritmo para pintar rectangulos

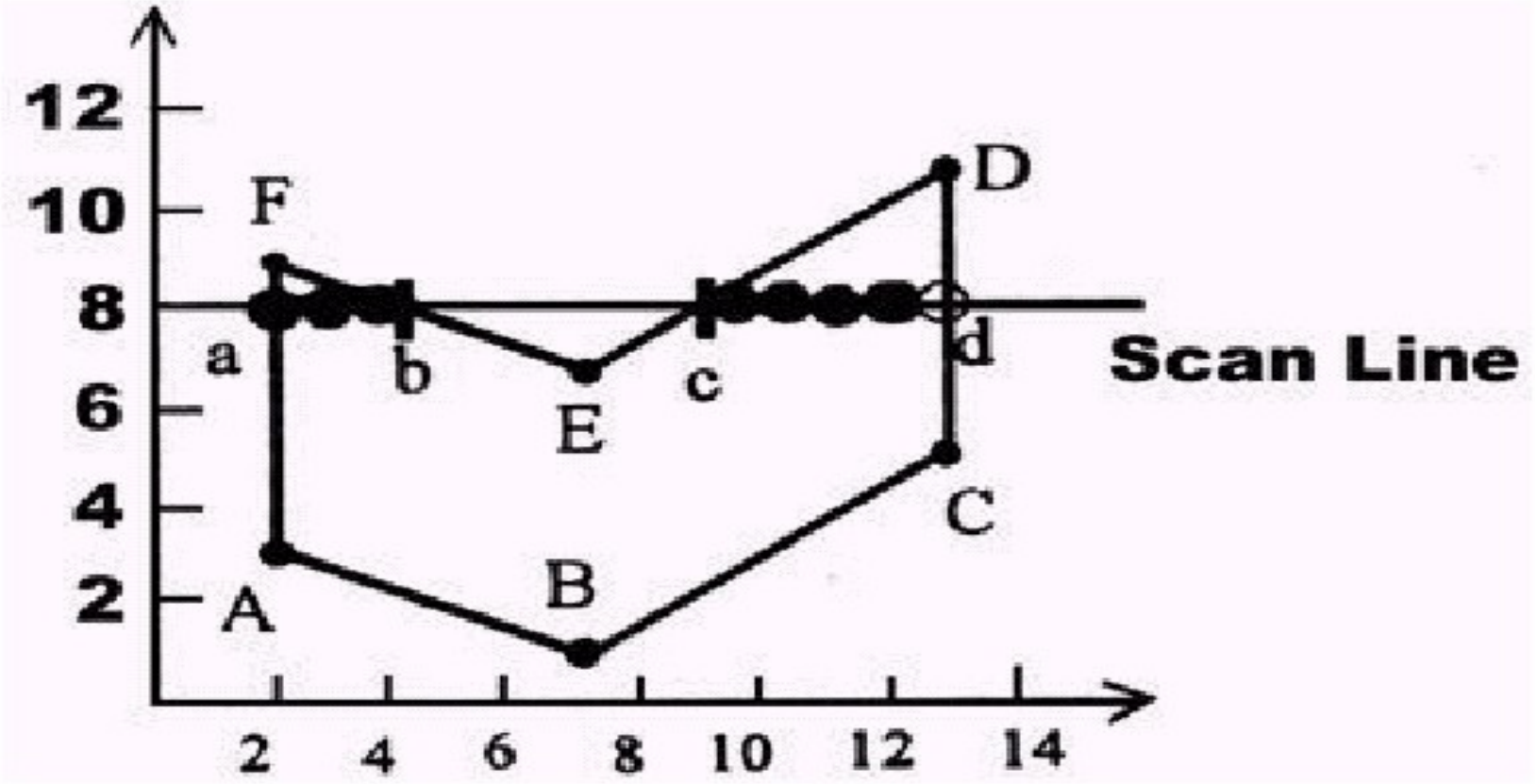
```
Para y = ymin até ymax do retângulo { Por linha de varredura }
   Para x = xmin até xmax { Cada pixel dentro do bloco }
      write_pixel (x,y,valor)
```

- Problemas exiten con rectangulos que comparten lados:
  - Los pixels que estan en la arista izquierda e inferior pertencen al poligono y se pintan

Rectángulos

- Consideraciones para graficar rectangulos
  - La regla se aplica de la misma manera a poligonos arbitrarios y no solamente a rectangulos
  - El vertice del canto inferior izquierdo aun sigue siendo pintado dos veces
  - La regla tambien puede ser usada para poligonos no pintados
  - La Regla hace que en cada bloque este faltando su pixel mas a la derecha y superior

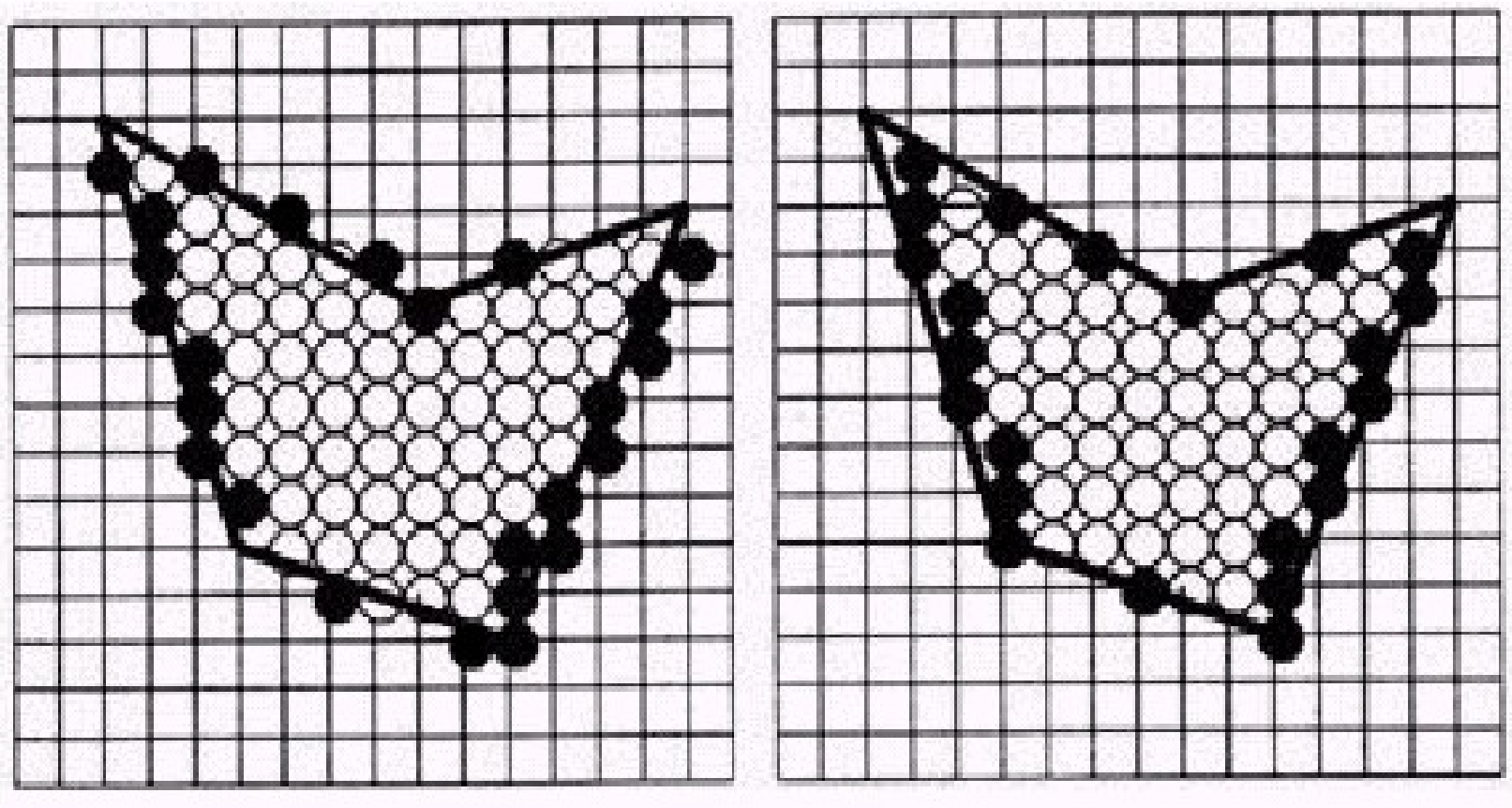
Rellenado de poligonos



Rellenado de poligonos

- Debemos darnos cuenta de los puntos extremos del poligono
  - Puede ser usando el algoritmo de punto medio
  - Este algoritmo pinta los pixels del lado mas proximo de la línea
  - No pintar pixels que no estan dentro del poligono no importando que este mas proximo de la arista real.

Rellenado de poligonos



Rellenado de poligonos

- El proceso de rellenado de poligono pues ser dividido en 3 pasos
  - Obtener la interseccion de linea de barredura con todos los lados del poligono
  - Ordenar los puntos de intersección (creciente en x)
  - Pintar los pixels entre pares de puntos de interseccion del poligono que son internos a el. Para determinar cuales son los pixels internos a un poligono podemos usar el bit de pariedad la pariedad inicialmente es par, y a cada interseccion encontrada el bit d epariedad es invertido, el pixel es pintado cuando el bit de pariedad es impar y no cuando es par.

Preguntas a responder para realizar el paso 3

Colocar todos los pixeles entre pares de intersecciones que se encuentren dentro del polígono, utilizando la regla de paridad impar

3.1 Dada una intersección con un valor ***x*** arbitrario y fraccionario, ¿cómo determinamos cuál de los dos pixeles a cada lado de la intersección es el interior?

3.2 ¿Cómo tratamos el caso especial de las intersecciones en coordenadas enteras de los pixeles?

3.3 ¿Cómo tratamos el caso especial del paso 3.2 para vértices compartidos?

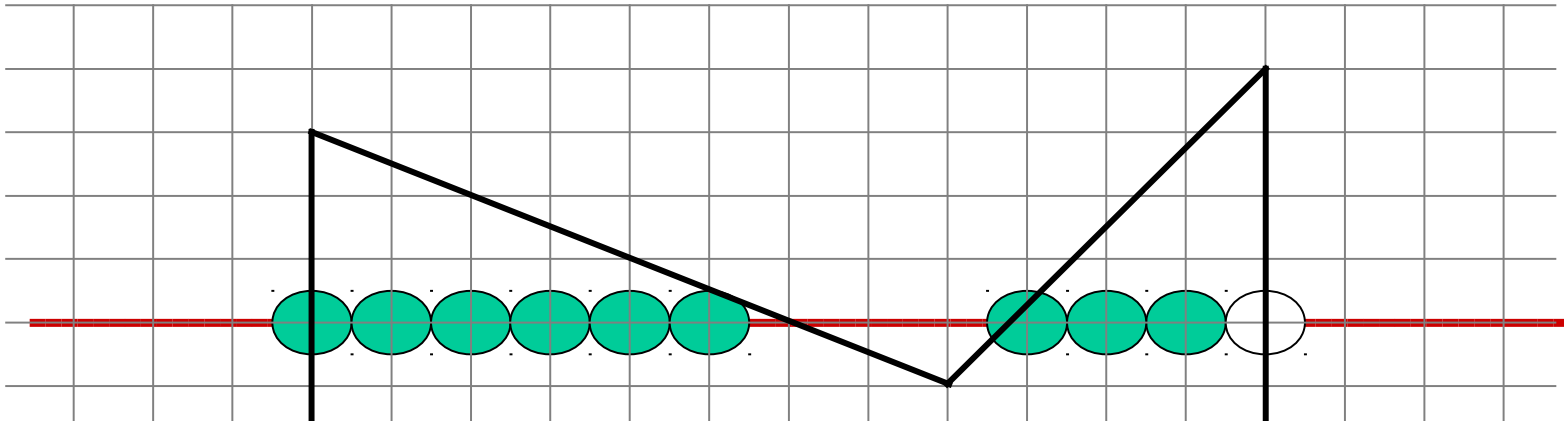
3.4 ¿Cómo tratamos el caso especial del paso 3.2 si los vértices definen una arista horizontal?

Preguntas a responder para realizar el paso 3

3.1 Dada una intersección con un valor ***x*** arbitrario y fraccionario, ¿cómo determinamos cuál de los dos pixeles a cada lado de la intersección es el interior?

Solución:

Si nos movemos hacia la derecha a una intersección fraccionaria, y estamos dentro del polígono, redondeamos hacia abajo en la coordenada x de la intersección (definiendo el pixel interior). Si estamos fuera, redondeamos hacia arriba la coordenada x (definiendo también un pixel interior).

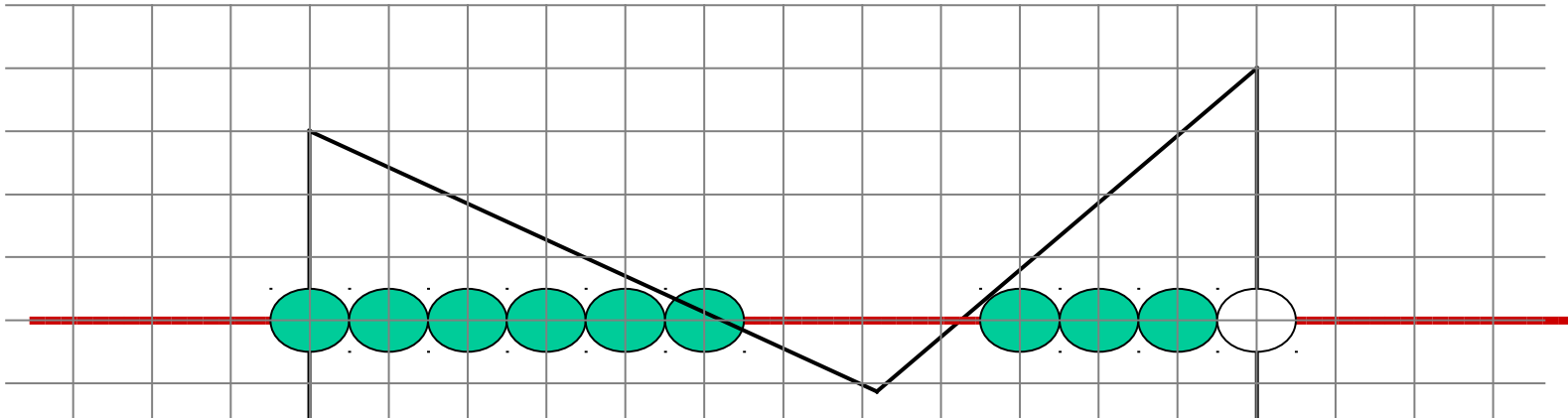


Preguntas a responder para realizar el paso 3

3.2 ¿Cómo tratamos el caso especial de las intersecciones en coordenadas enteras de los pixeles?

Solución:

Si el pixel del extremo izquierdo de un tramo tiene coordenada x entera, lo definimos como interior. Si el pixel de extremo derecho tiene coordenada x entera, lo definimos como exterior.

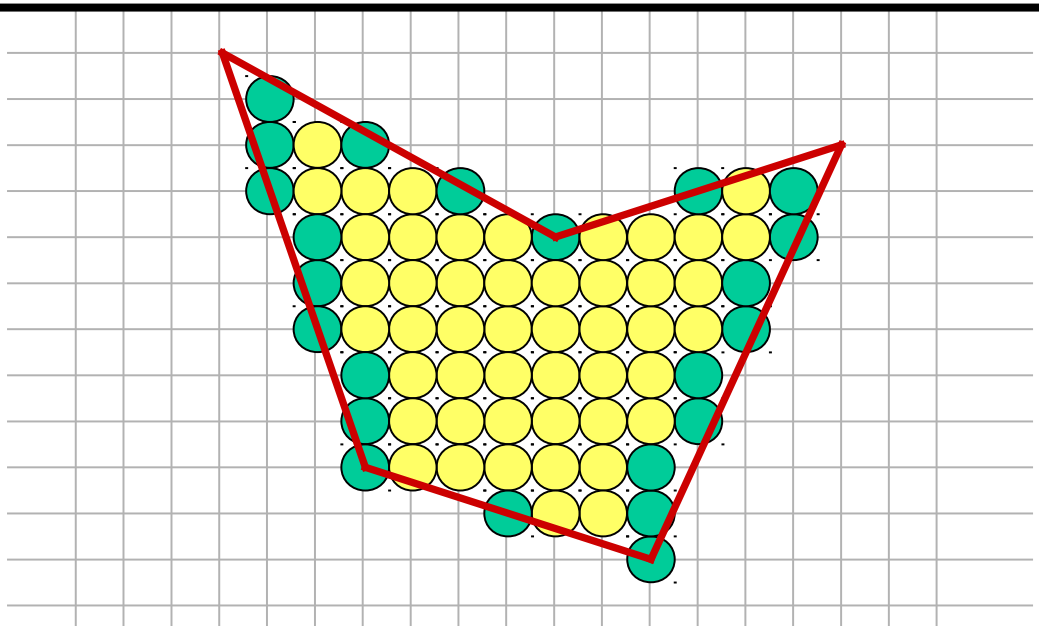


Preguntas a responder para realizar el paso 3

3.3 ¿Cómo tratamos el caso especial del paso 3.2 para vértices compartidos?

Solución:

En el cálculo de paridad, contamos solo los vértices y_{min} de las aristas y no los vertices y_{max}.

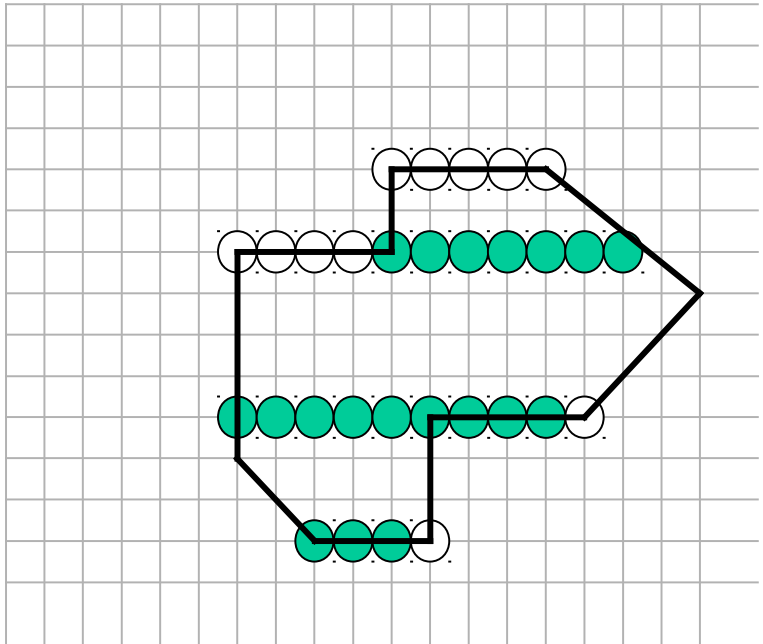


Preguntas a responder para realizar el paso 3

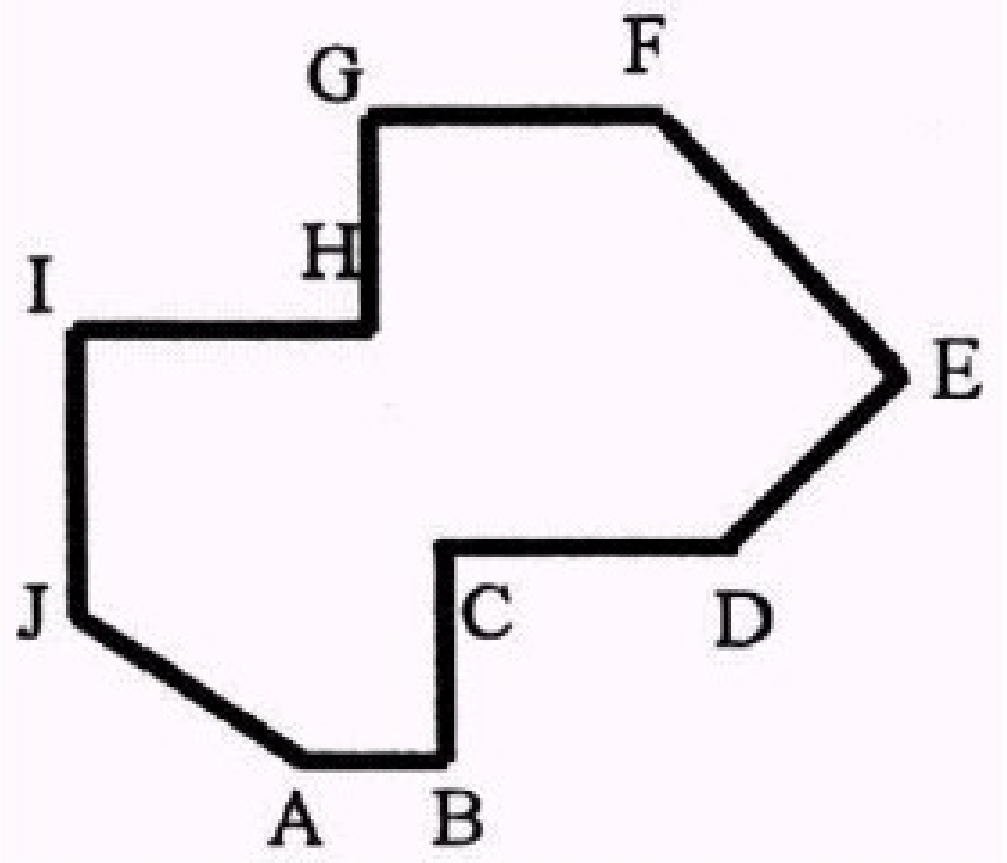
3.4 ¿Cómo tratamos el caso especial del paso 3.2 si los vértices definen una arista horizontal

Solución:

No se cuentan los vértices de las aristas. No son ni y_{min} ni y_{max}.



Trasos Horizontales



Observaciones

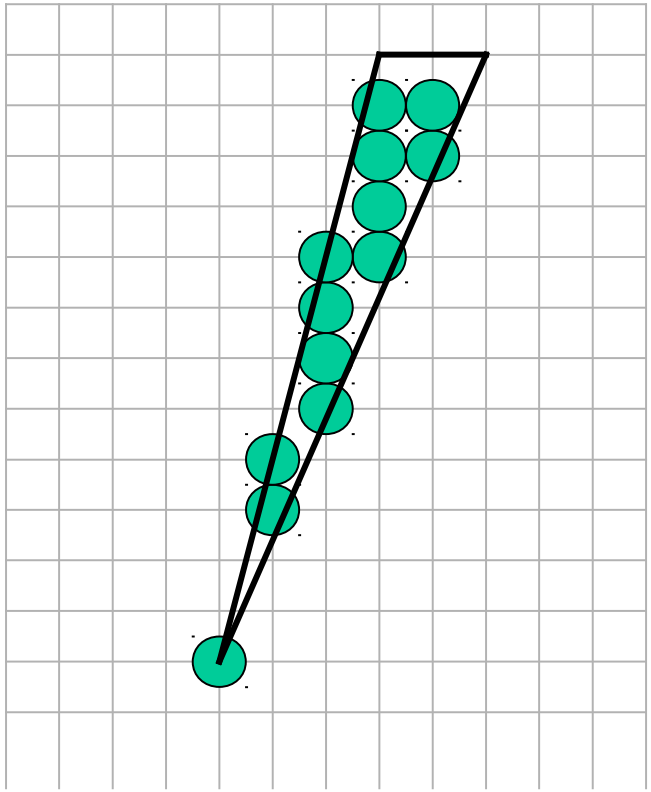
- El Algoritmo no considera los ppixels superiores ni los mas a la derecha
- No dibuja los pixels sobre el tope interno de un poligono concavo
- No dibuja los pixels que son maximos locales
- Algunos pixels correctos no los imprime

Astillas

Cuando hay polígonos con aristas tan cercanas que crean una astilla.

Pueden haber líneas de rastreo sin pintar.

Se puede solucionar si en lugar de 2 valores posibles para el píxel, se permiten valores deintensidad que varien como función de la distancia entre el centro del pixel y la primitiva.



Algoritmos para calculo de la aristas

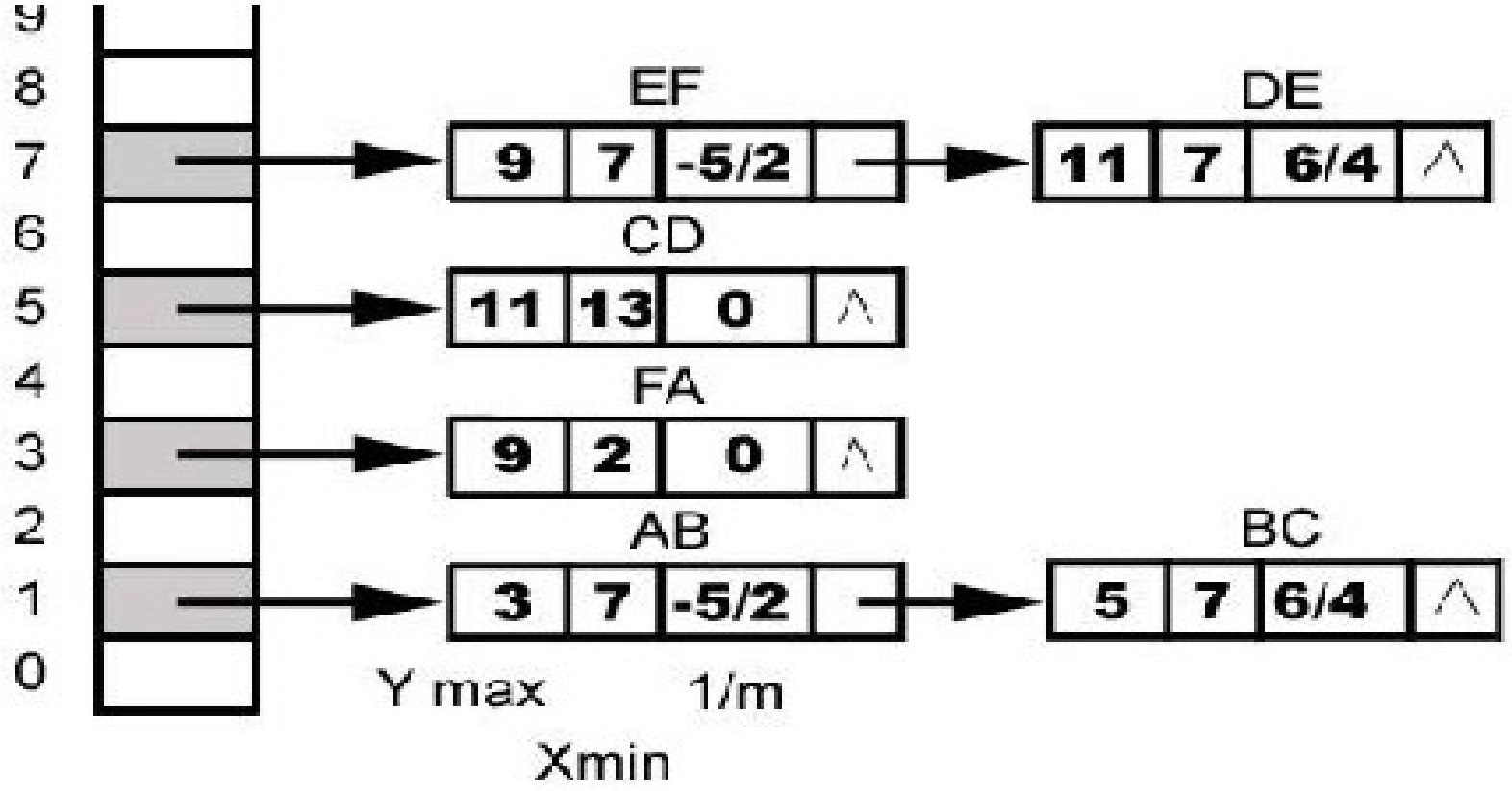
- Al movernos de una linea i hacia a otra i + 1 podemos calcularlo usando el algoritmo del punto medio

$$x_{i+1} = x_i + \frac{1}{m}$$

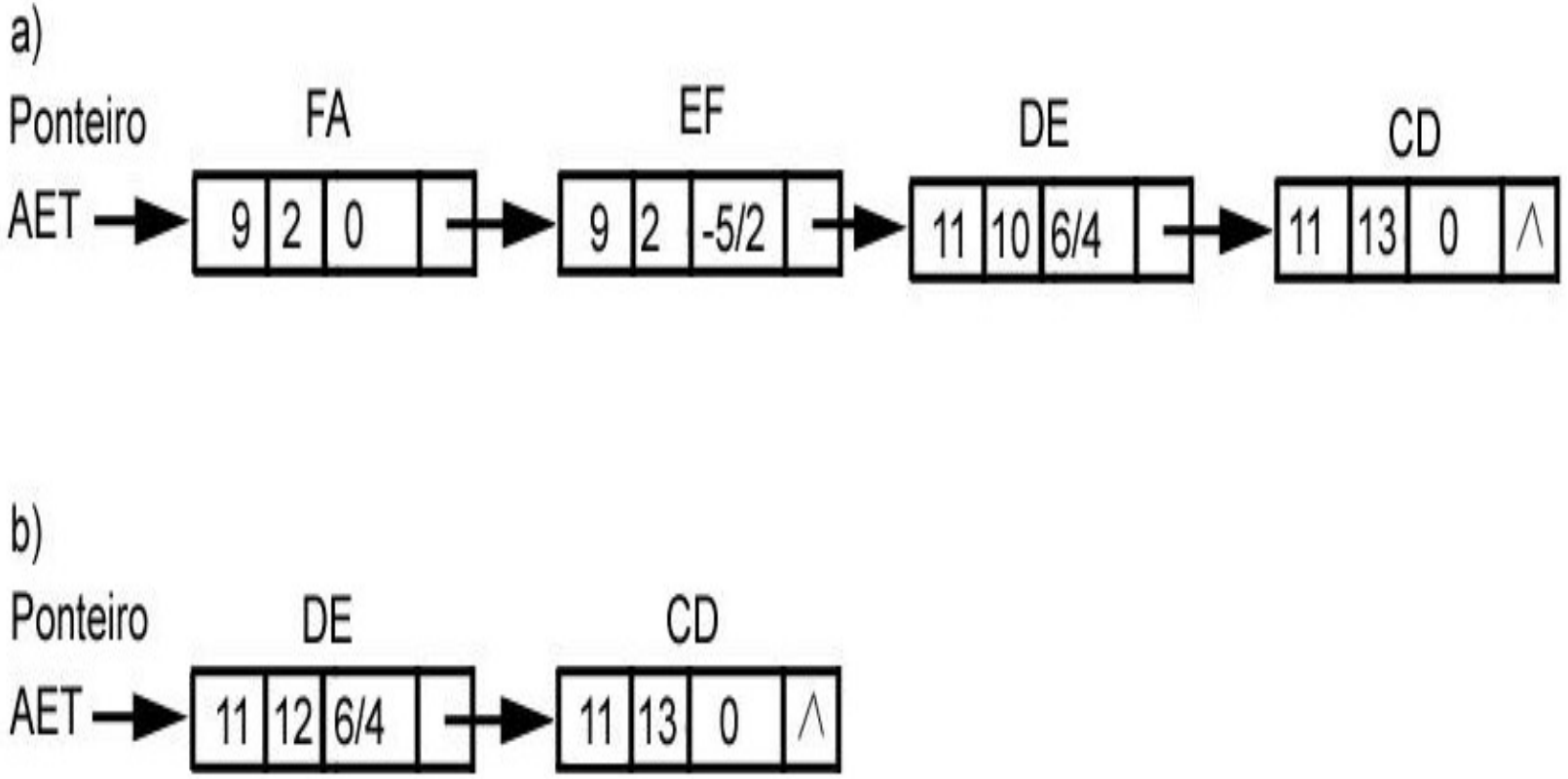
$$m = \frac{(ymax - ymin)}{(xmax - xmin)}$$

```
void LeftEdgeScan (int xmin, int ymin, int xmax, int ymax, int valor){
   int x, y;

   x = xmin;
   y = ymin;
   numerador = xmax - xmin;
   denominador = ymax - ymin;
   incremento = denominador;

   for (y = ymin; y <= ymax; y++){

      PintaPixel (x,y,valor);
      incremento+=numerador;

      if (incremento > denominador){
         /* Overflow, Arredonda o próximo pixel e decrementa o incremento */
         x++;
         incremento-=denominador;
      }/* end if */
   }/* end for */
}/* end LeftEdgeScan */
```

Edge table *ET*



Active edge table *EAT*



1. Obtém a menor coordenada y armazenada na ET; ou seja, o valor de y do primeiro "cesto" não vazio.

2. Inicializa a AET como vazia.

3. Repita até que a ET e a AET estejam vazias:

   3.1. Transfere do cesto y na ET para a AET as arestas cujo $ymin = y$ (lados que estão começando a serem varridos), mantendo a AET ordenada em x,

   3.2. Retira os lados que possuem $y = ymax$ (lados não mais envolvidos nesta linha de varredura,

   3.3. Desenhe os pixels do bloco na linha de varredura y usando pares de coordenadas x da AET.

   3.4. Incremente y de 1 (coordenada da próxima linha de varredura).

   3.5. Para cada aresta não vertical que permanece na AET, atualiza x para o novo y.

   3.6. Como o passo anterior pode ter desordenado a AET, reordena a AET.