

## 3. Raster Algorithms

### 3.1 Raster Displays

(see Chapter 1 of the Notes)

### 3.2 Monitor Intensities & Gamma

- Values of intensity calculated by an illumination model must be converted to one of the allowable intensity levels for the particular graphics system in use

How should intensity levels be spaced?

- logarithmically, not linearly.
- Eye is sensitive to ratios of intensity levels rather than to their absolute values
- ratio of successive intensities should be constant to get equal perceived brightness

For each component of the RGB model:

$I_0$ : lowest attainable intensity (0.005 - 0.025)

$n + 1$ : number of intensity levels with equal perceived brightness

We should have

$$\frac{I_1}{I_0} = \frac{I_2}{I_1} = \dots = \frac{I_n}{I_{n-1}} = r$$

with

$$r = \left(\frac{1}{I_0}\right)^{1/n}$$

Example:

$$I_0 = 1/8, \quad n = 3, \quad r = 2, \quad I = 1/8, \quad 1/4, \quad 1/2, \quad 1$$

$$I_0 = 1/8, \quad n = 255, \quad r = 1.0182, \quad I = 0.0100, \\ 0.0100, \quad \dots$$

How to display a desired **intensity**  $I$ ?

1. Determine the nearest  $I_j$

$$j = \text{ROUND}\left(\log_r \left(\frac{I}{I_0}\right)\right)$$

2. Calculate

$$I_j = r^j I_0$$

3. Determine the pixel value

$$V_j = \text{ROUND}\left(\left(\frac{I_j}{M}\right)^{1/\gamma}\right)$$

( **Gamma** correction of intensity) Why?

4. If the raster display has no look-up table, then  $V_j$  is placed in the appropriate pixel. Otherwise,  $j$  is placed in the pixel and  $V_j$  is placed in entry  $j$  of the table.

Here is "Why?"

Relationship between intensity of light output (displayed intensity) and the number of electrons in the beam:

$$I \propto N^\gamma, \quad 2.2 \leq \gamma \leq 2.5$$

Relationship between the number of electrons in the beam and intensity value specified for the pixel (input voltage):

$$N \propto V$$

Hence, to display a particular intensity value  $I$  for the pixel, the correct voltage value to produce this intensity is:

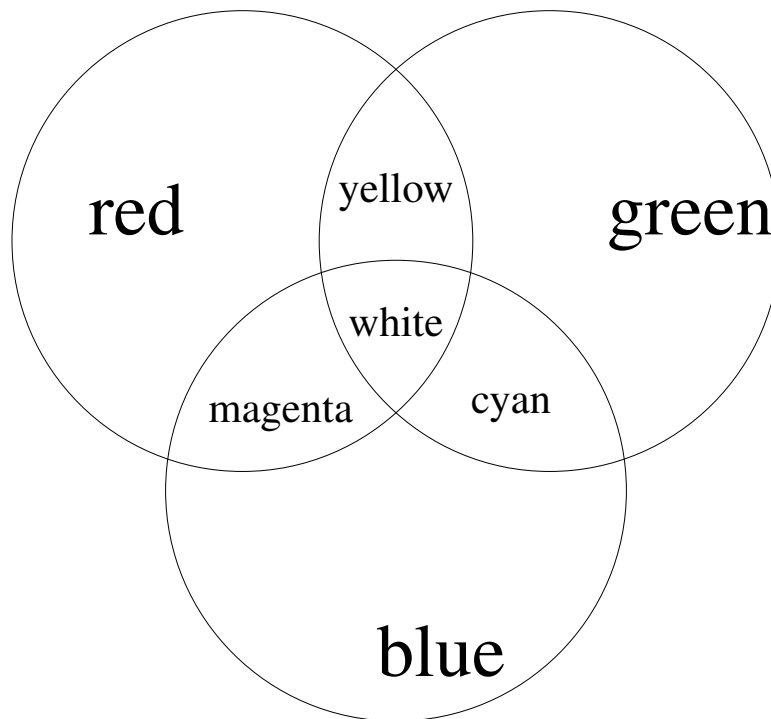
$$V = \left( \frac{I}{M} \right)^{1/\gamma}$$

where  $M$  is the maximum intensity.

### 3.3 RGB Color

- color (light) is displayed by three primary lights: **red**, **green**, **blue**, in an *additive* manner

(on the other hand, paints and crayons are generated using *subtractive* color mixing, with primaries: **red**, **yellow**, **blue**)

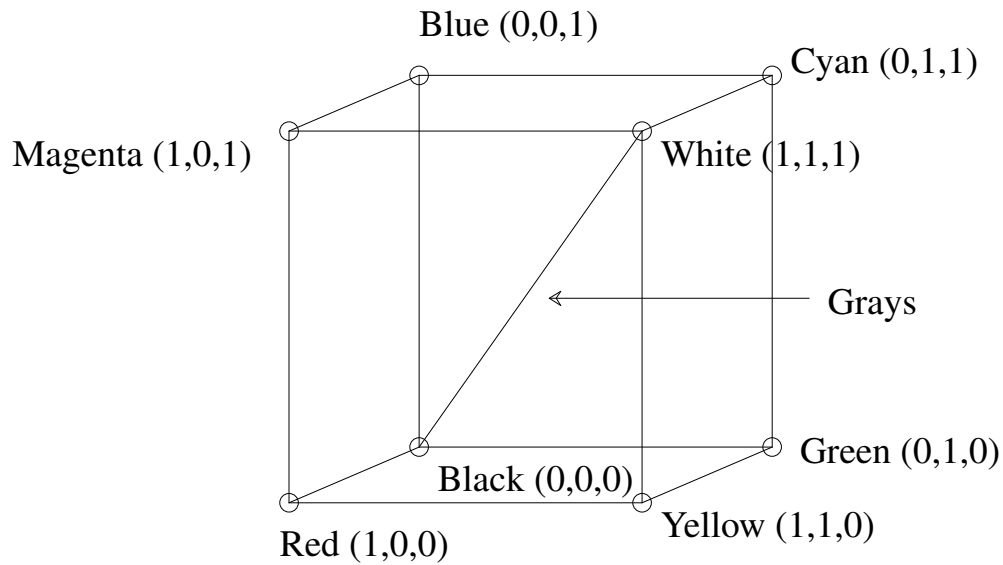


$$\text{red} + \text{green} = \text{yellow}$$

$$\text{green} + \text{blue} = \text{cyan}$$

$$\text{blue} + \text{red} = \text{magenta (purple)}$$

$$\text{red} + \text{green} + \text{blue} = \text{white}$$

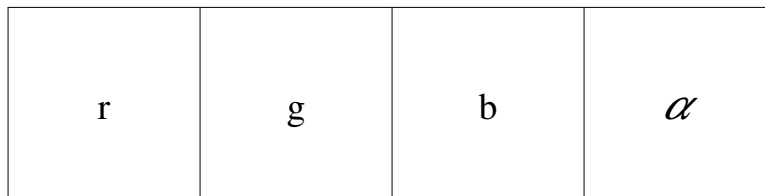


## Notes:

- Colors are specified as follows:
  - Color =  $(r,g,b)$ ,  $0 \leq r, g, b \leq 1$
- Individual contributions of each primary are added together to yield the result
- Most popular for CRT monitors

## 3.4 The Alpha Channel

- How to partially overwrite the contents of a pixel, such as in *compositing*
- Compositing is the process of combining separate image layers into a single picture
- a multi-bit gray-scale mask (called *alpha*) is maintained as a fourth *alpha channel* in addition to (r, g, b) color channels
- *alpha* channel is used to hold an **opacity factor** (or, the fraction of the pixel covered by an opaque surface) for each pixel



To composite a foreground color  $\mathbf{c}_f$  over background color  $\mathbf{c}_b$ , and the fraction of the pixel covered by foreground is  $\alpha$ , us the formula

$$\mathbf{c} = \alpha \mathbf{c}_f + (1 - \alpha) \mathbf{c}_b$$

## 3.5 Scan Converting Lines

**Task:** Given a line segment defined by  $(x_{start}, y_{start})$  and  $(x_{end}, y_{end})$ , with slope  $0 < m \leq 1$  ( $(x_{start}, y_{start})$  is to the left of  $(x_{end}, y_{end})$ ), find algorithms that can rapidly generate the **pixels** comprising the line in a very accurate way

**Basic Idea:**

---

```
 $y = y_{end}$   
for  $x = x_{start}$  to  $x_{end}$  do  
    draw  $(x, y)$   
    if (some condition) then  
         $y = y + 1$ 
```

---

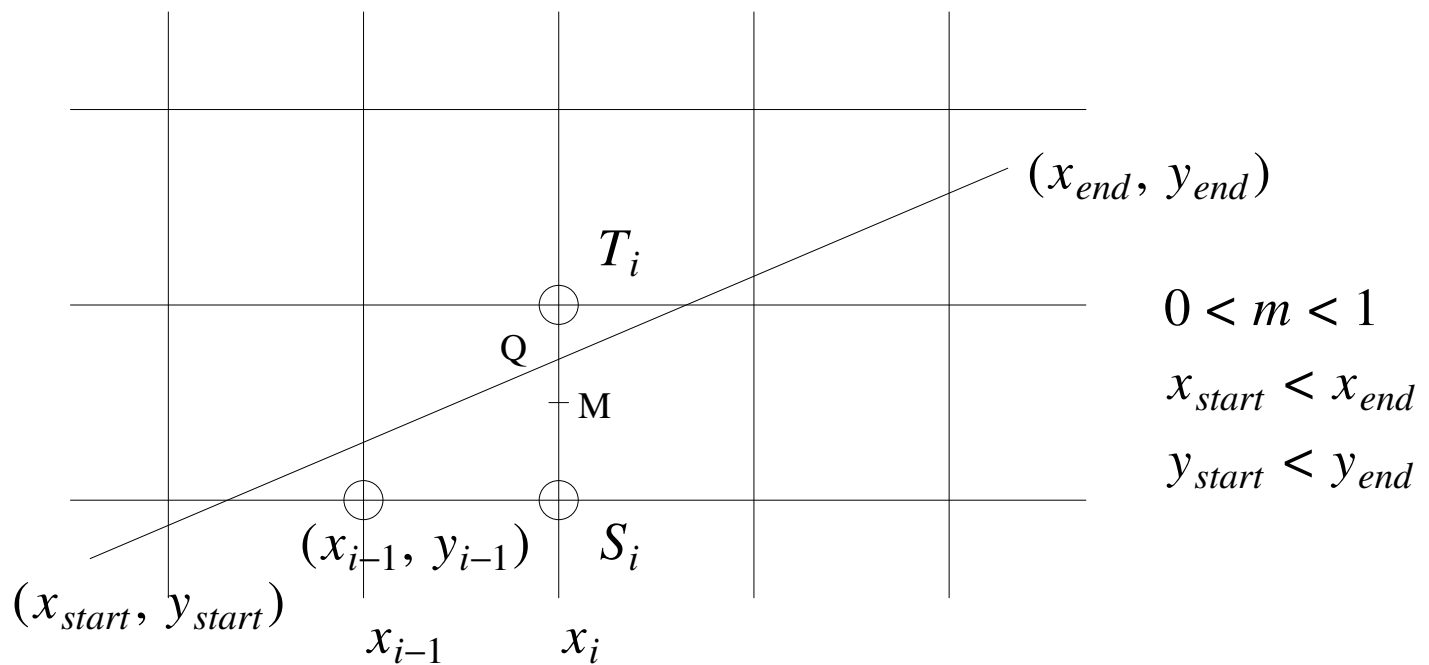
## Using Implicit Line Equations

Implicit line equation:  $F(x, y) \equiv ax + by + c = 0$

$$a = \Delta_y \equiv y_{start} - y_{end}$$

$$b = -\Delta_x \equiv -(x_{start} - x_{end})$$

$$c = x_{start}y_{end} - x_{end}y_{start}$$



$(x_{i-1}, y_{i-1})$ : pixel plotted for line  $x = x_{i-1}$

$T_i, S_i$ : only possible pixels to be chosen from for line  $x = x_i (= x_{i-1} + 1)$

**Question:** which pixel,  $T_i$  or  $S_i$ , should be selected then?

$$\text{Answer} :: \begin{cases} T_i & \text{if } Q_i \text{ is above } M_i \\ S_i & \text{if } Q_i \text{ is below } M_i \end{cases}$$

## How to make efficient decision?

Consider the sign of

$$e_i \equiv F(M_i) = F(x_i, y_{i-1} + 0.5)$$

Then

$$\begin{cases} T_i & \text{is plotted if } e_i \geq 0 \\ S_i & \text{is plotted if } e_i < 0 \end{cases}$$

or, consider the sign of

$$\begin{aligned} E_i &\equiv 2F(M_i) = 2F(x_i, y_{i-1} + 0.5) \\ &= 2\Delta_y(x_i) - \Delta_x(2y_{i-1} + 1) + 2(x_{start}y_{end} - x_{end}y_{start}) \end{aligned}$$

To compute  $E_i$ , note that

$$\begin{aligned} E_{i+1} &= 2F( x_{i+1}, y_i + 0.5 ) \\ &= 2\Delta_y x_{i+1} - \Delta_x(2y_i + 1 ) + 2( x_{start} y_{end} - x_{end} y_{start} ) \end{aligned}$$

Therefore

$$E_{i+1} - E_i = 2\Delta_y - 2\Delta_x(y_i - y_{i-1})$$

Since

$$y_i = \begin{cases} y_{i-1} + 1, & \text{if } E_i \geq 0 \\ y_{i-1}, & \text{if } E_i < 0 \end{cases}$$

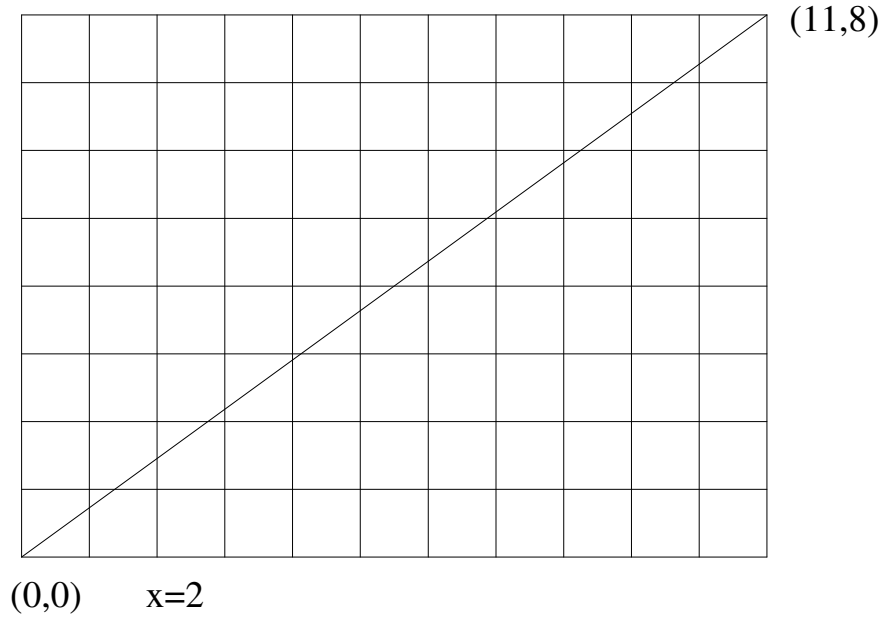
we have the following recurrence formula

$$E_{i+1} = \begin{cases} E_i + 2(\Delta_y - \Delta_x), & E_i \geq 0 \\ E_i + 2\Delta_y, & E_i < 0 \end{cases}$$

with the initial condition

$$E_1 = 2\Delta_y - \Delta_x$$

Example:



Which pixel should be plotted for the line  $x = 2$ ,  $x = 6$ ?

## Using Parametric Line Equations

Parametric line equation:

$$\mathbf{P}(t) = \mathbf{P}_{start} + t(\mathbf{p}_{end} - \mathbf{p}_{start}), \quad t \in [0, 1];$$

$$\mathbf{p}(t) = ( x(t), y(t) ),$$

$$\mathbf{p}_{start} = ( x_{start}, y_{start} ),$$

$$\mathbf{p}_{end} = ( x_{end}, y_{end} )$$

Assuming slope  $m \in [-1, 1]$ .

By computing  $t$  as a function of  $x$ , we have

---

```
for  $x = x_{start}$  to  $s_{end}$  do  
   $t = (x - x_{start}) / (x_{end} - x_{start})$   
   $y = y_{start} + t(y_{end} - y_{start})$   
  draw( $x$ , round( $y$ ))
```

---

Can be made incremental using the fact that  $t$  and  $y$  change by a constant amount in each iteration.

---

$$\Delta y = (y_{end} - y_{start}) / (x_{end} - x_{start})$$

$$y = y_{start}$$

**for**  $x = x_{start}$  **to**  $x_{end}$  **do**

**draw**( $x$ , **round**( $y$ ))

$$y = y + \Delta y$$

---

If the the colors at the endpoints are  $c_{start}$  and  $c_{end}$ , respectively, we can change the color smoothly along the line as follows:

---

$$\Delta y = (y_{end} - y_{start}) / (x_{end} - x_{start})$$

$$\Delta r = (r_{end} - r_{start}) / (x_{end} - x_{start})$$

$$\Delta g = (g_{end} - g_{start}) / (x_{end} - x_{start})$$

$$\Delta b = (b_{end} - b_{start}) / (x_{end} - x_{start})$$

$$y = y_{start} ; r = r_{start} ; g = g_{start} ; b = b_{start}$$

**for**  $x = x_{start}$  **to**  $x_{end}$  **do**

**draw**(  $x$  , **round**(  $y$  ) ,  $r$  ,  $g$  ,  $b$  )

$$y = y + \Delta y$$

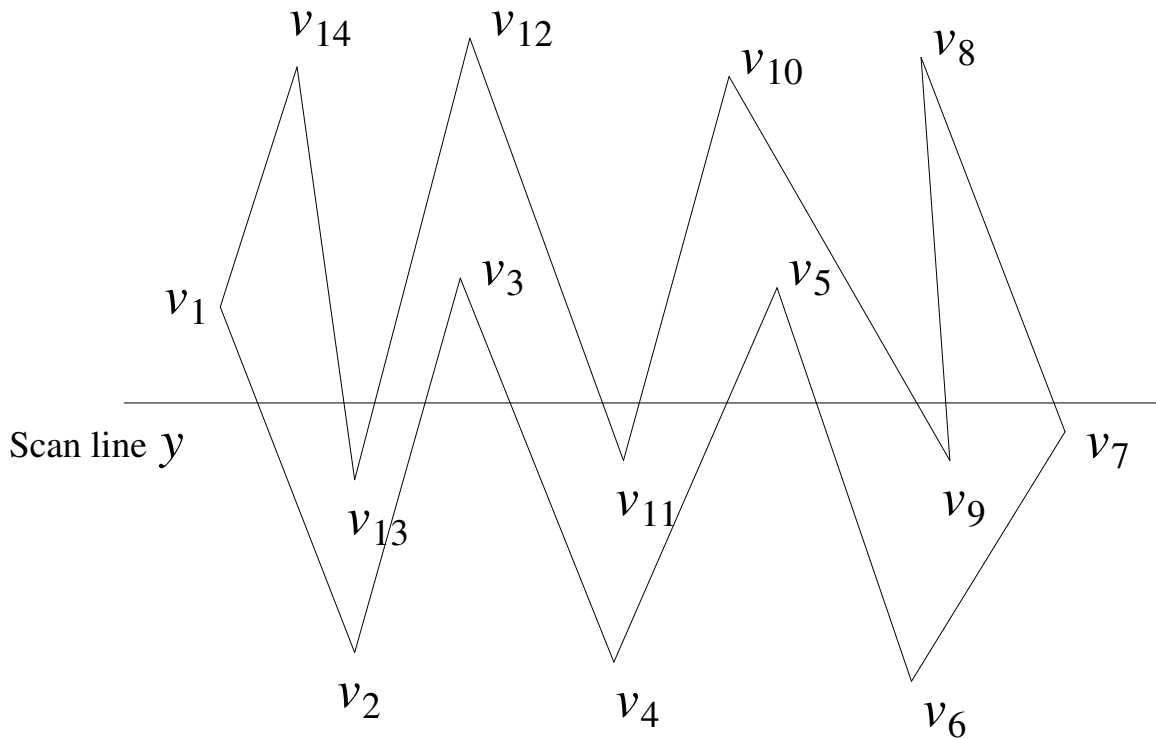
$$r = r + \Delta r$$

$$g = g + \Delta g$$

$$b = b + \Delta b$$

---

## 3.6 Scan Converting Polygons



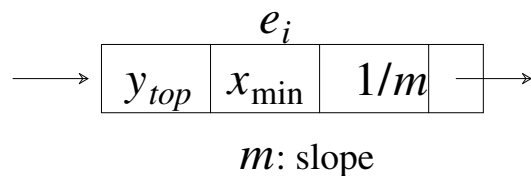
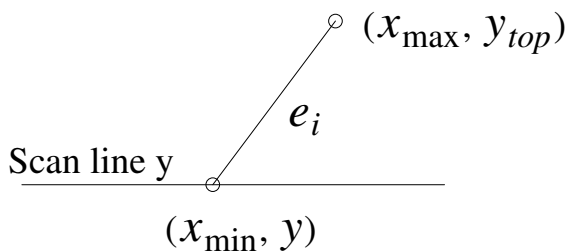
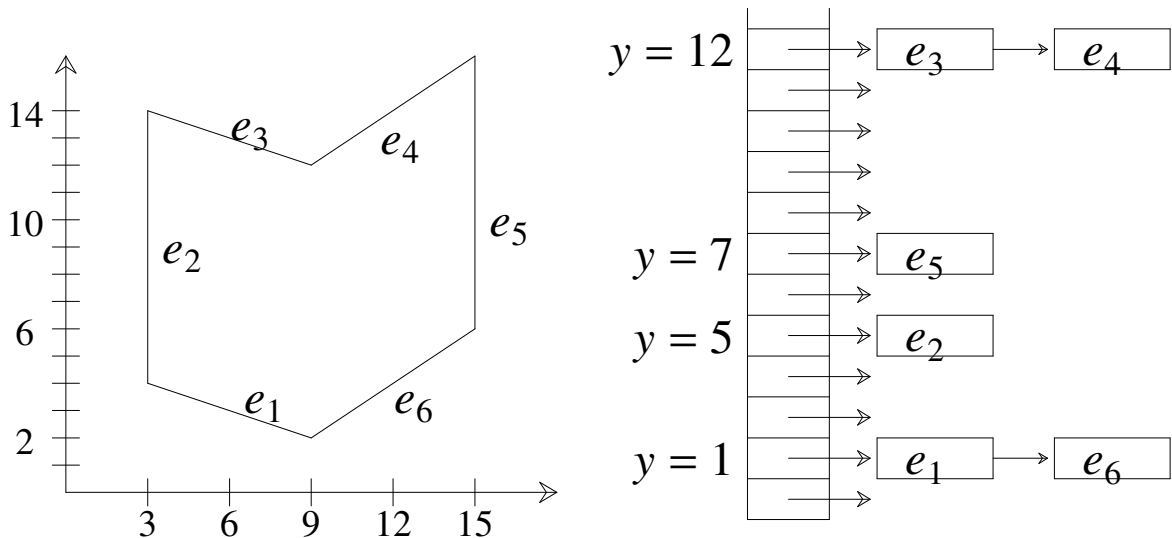
### Basic idea:

1. Compute  $x$  coordinates of intersection points of current scan line with all edges
2. Sort intersection points by increasing  $x$  values
3. Group intersection points by pairs
4. Fill in the pixels between each pair of intersection points on the current scan line

Need to create a bucket-sorted edge table (ET) first:

- To determine which edges intersect current scan line
- To efficiently compute intersection points of these edges with the current scan line
- Vertical edges need to be shortened by 1 in y direction

Example:



Also need to maintain an **active-edge table (AET)**

**Purpose:** keep track of the edges the current scan line intersects

**How:** when we move to a new scan line (bottom to top), new edges intersecting the new scan line are added into the AET, edges in AET which are no longer active (not intersected by the new scan line) are deleted

**Example:**

In the previous example, when  $y = 2$ ,

*AET* →

when  $y = 4$ ,

*AET* →

when  $y = 8$ ,

*AET* →

## Algorithms:

1. Set  $y$  to the  $y$ -coordinate of the first non-empty bucket
2. Set AET to empty
3. **Repeat until** the AET and ET are both empty
  - 3.1 **Merge** edges from ET bucket  $y$  with edges in AET, maintaining AET sort order on  $x$
  - 3.2 **Fill in** pixels on scan line  $y$  bounded by pairs of  $x$ -coordinates from edges in AET
  - 3.3 **Remove** from AET those edges for which  $y = y_{top}$
  - 3.4 For each edge remaining in AET, **replace**  $x$  with  $x + 1/m$
  - 3.5 **Increment**  $y$  by 1, to the coordinate of the next scan line