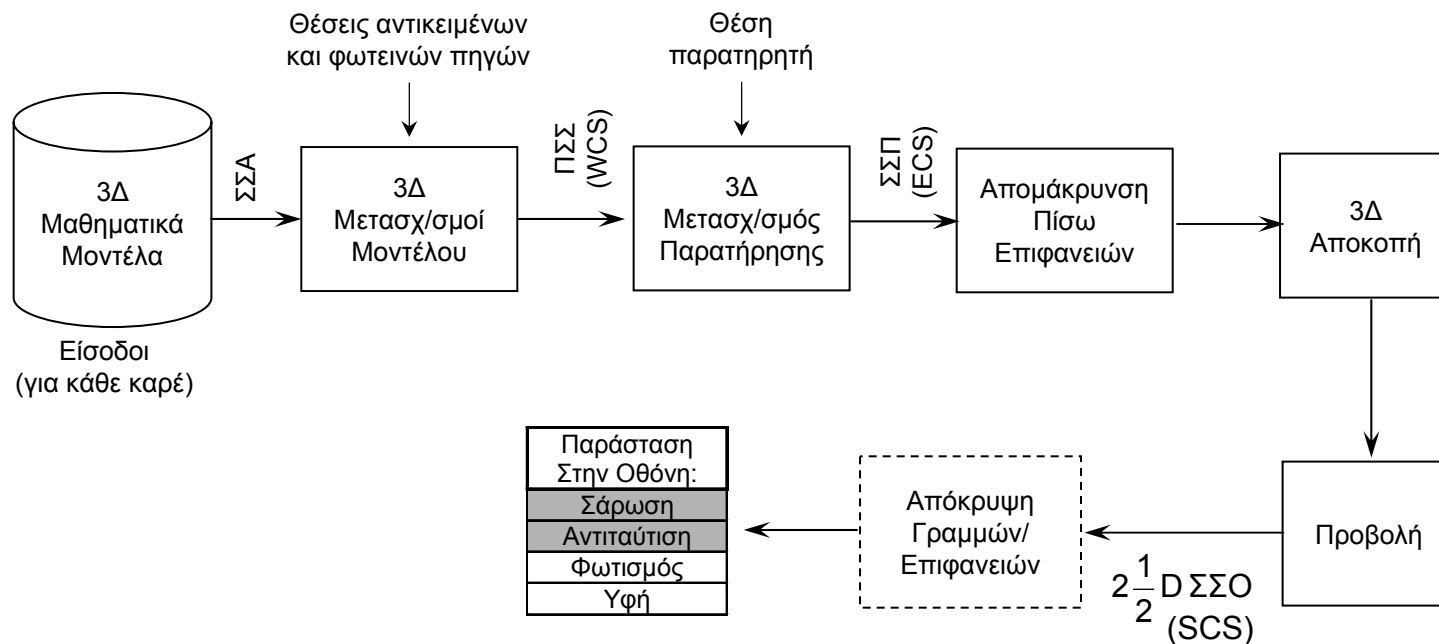


## Αλγόριθμοι Παράστασης Βασικών Σχημάτων

- Προσέγγιση μαθηματικών σχημάτων από διακριτά pixels:
  - Ευθύγραμμο τμήμα, κύκλος, κωνικές τομές, πολύγωνο.
  - S/W ή H/W.



## Ευθύγραμμο Τμήμα: Αλγόριθμος 1

- Κριτήρια καλού αλγόριθμου ευθύγραμμου τμήματος:
  - Σταθερό πάχος ανεξάρτητο κλίσης, όχι κενά (συνεκτική).
  - Pixels όσο το δυνατόν πλησιέστερα στη μαθηματική πορεία της.
  - Ταχύτητα.
- Εστω ευθύγραμμο τμήμα μεταξύ  $P_1(x_1, y_1)$  και  $P_n(x_n, y_n)$  1<sup>ου</sup> οκταμορίου:
  - Για κάθε σημείο  $P(x, y)$  του ευθύγραμμου τμήματος ισχύει:

$$y = s \cdot x + b \quad \text{με} \quad s = \frac{y_n - y_1}{x_n - x_1} = \frac{\Delta y}{\Delta x} \quad \text{και} \quad b = \frac{y_1 x_n - y_n x_1}{x_n - x_1}$$

```
line1(x1, y1, xn, yn, colour)
int x1, y1, xn, yn, colour;
/*colour η τιμή του χρώματος του ευθύγραμμου τμήματος*/
{float s, b, y; int x;
  s = (yn - y1) / (xn - x1);
  b = (y1 * xn - yn * x1) / (xn - x1);
  for (x = x1; x <= xn; x++)
    {y = s * x + b;
     setpixel(x, round(y), colour); }
}
```

## Ευθύγραμμο Τμήμα: Αλγόριθμος 2

- Πολλαπλασιασμός μέσα στο βρόχο μπορεί να αποφευχθεί:
  - Αντικαθίσταται από πρόσθεση αφού ισχύει:

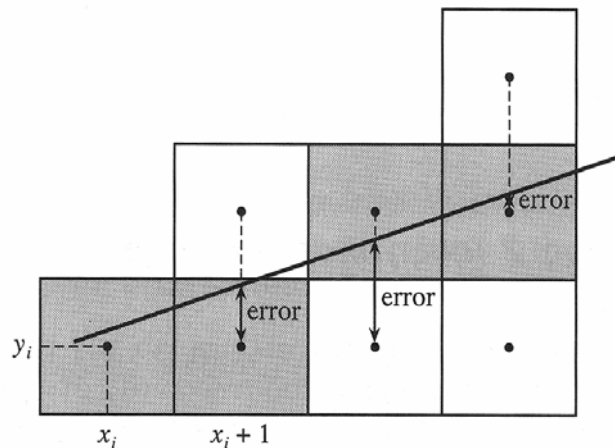
$$x_{i+1} = x_i + 1$$

$$y_{i+1} = sx_{i+1} + b = sx_i + b + s = y_i + s$$

```
line2 (x1, y1, xn, yn, colour)
int x1, y1, xn, yn, colour;
{float s, y; int x;
  s=(yn-y1)/(xn-x1);
  y=y1;
  for (x=x1; x<=xn; x++)
    {setpixel(x, round(y), colour);
     y=y+s;
    }
}
```

### Ευθύγραμμο Τμήμα: Αλγόριθμος 3

- Στρογγύλευση ( $y$ ) μέσα στο βρόχο μπορεί να αποφευχθεί:
  - Διαχωρισμός  $y$  σε ακέραιο ( $y$ ) και δεκαδικό ( $error$ ) μέρος.
  - $error$  είναι η απόσταση του pixel  $(x_i+1, y_i)$  από ιδεατή ευθεία:



```
line3 (x1, y1, xn, yn, colour)
int x1, y1, xn, yn, colour;
{float s, error; int x, y;
 s=(yn-y1)/(xn-x1);
 y=y1;
 error=0;
 for (x=x1; x<=xn; x++)
 {setpixel(x, y, colour);
 error=error+s;
 if (error>=0.5){y++; error--}}
}
```

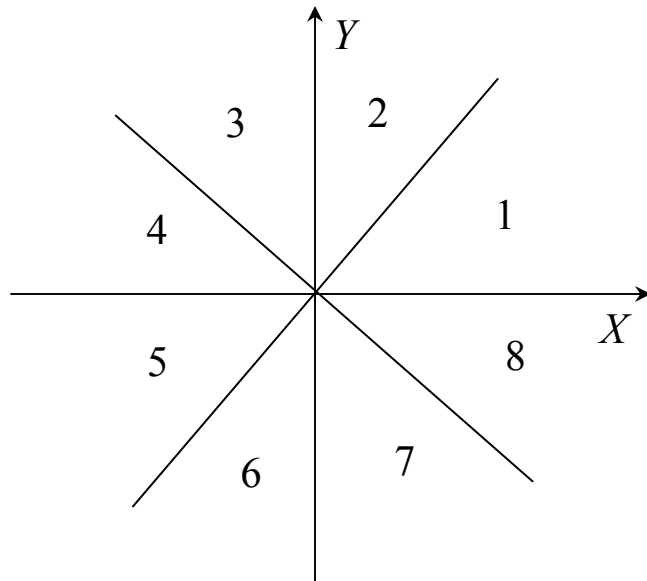
## Ευθύγραμμο Τμήμα: Αλγόριθμος 4 (Bresenham)

- Αντικατάσταση πραγματικών μεταβλητών από ακέραιες με κατάλληλη κλιμάκωση  $s$ ,  $error$  & συνθήκης επιλογής:
  - πολλαπλασιάζουμε με  $dx = x_n - x_1$ ,  $s \rightarrow dy$ ,  $error$  ακέραιο.
  - συνθήκη  $error \geq dx/2 \Leftrightarrow error \geq \lfloor dx/2 \rfloor \Leftrightarrow error \geq 0$  & αρχική αφαίρεση  $\lfloor dx/2 \rfloor$  από  $error$  ( /2 με ολίσθηση).

```
line4 (x1,y1,xn,yn,colour)
int x1,y1,xn,yn,colour;
    {int error,x,y,dx,dy;
      dx=xn-x1; dy=yn-y1;
      error=-dx/2; y=y1;
      for (x=x1; x<=xn; x++)
        {setpixel(x,y,colour);
          error=error+dy;
          if (error>=0){y++; error=error-dx}
        }
    }
```

## Ευθύγραμμο Τμήμα: Αλγόριθμος Bresenham

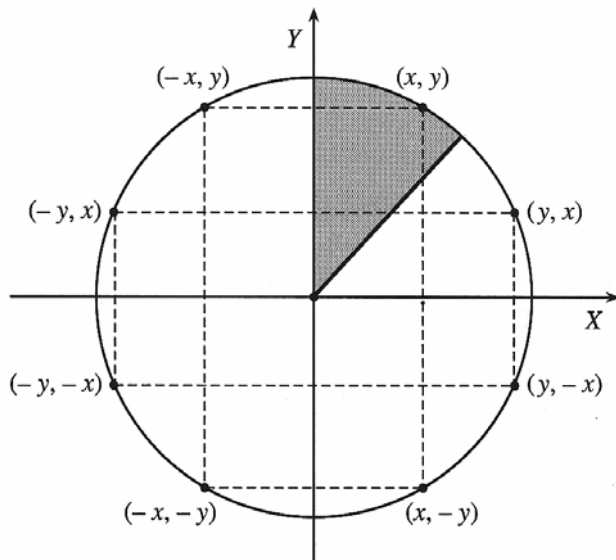
- Παραπάνω λειτουργεί μόνο στο 1<sup>ο</sup> οκταμόριο (αλλά συμμετρικά):
  - μεταφορά  $(x_1, y_1)$  ώστε να συμπίσει με αρχή αξόνων.



Οκταμόριο	Άξονας ταχυτ. Κίνησης	Άλλος άξονας
1	$x$	Αυξάνεται
2	$y$	»
3	$y$	Μειώνεται
4	$x$	»
5	$x$	Αυξάνεται
6	$y$	»
7	$y$	Μειώνεται
8	$x$	»

## Κύκλος

- 8-πλή συμμετρία, δημιουργούμε ένα οκταμόριο (έστω  $2^\circ$ )



```
circle_symmetry (x, y, colour)
int x, y, colour;
{setpixel(x, y, colour);
 setpixel(y, x, colour);
 setpixel(y, -x, colour);
 setpixel(x, -y, colour);
 setpixel(-x, -y, colour);
 setpixel(-y, -x, colour);
 setpixel(-y, x, colour);
 setpixel(-x, y, colour);
}
```

## Κύκλος: Αλγόριθμος Bresenham

- Εστω  $(x_i, y_i)$  επελέγη. Επόμενο βήμα  $(x_i+1, y_i)$  ή  $(x_i+1, y_i-1)$

- Μεταβλητή απόφασης:

$e_i = d_1 - d_2$  όπου  $d_1 = y_i^2 - y^2$  και  $d_2 = y^2 - (y_i - 1)^2$

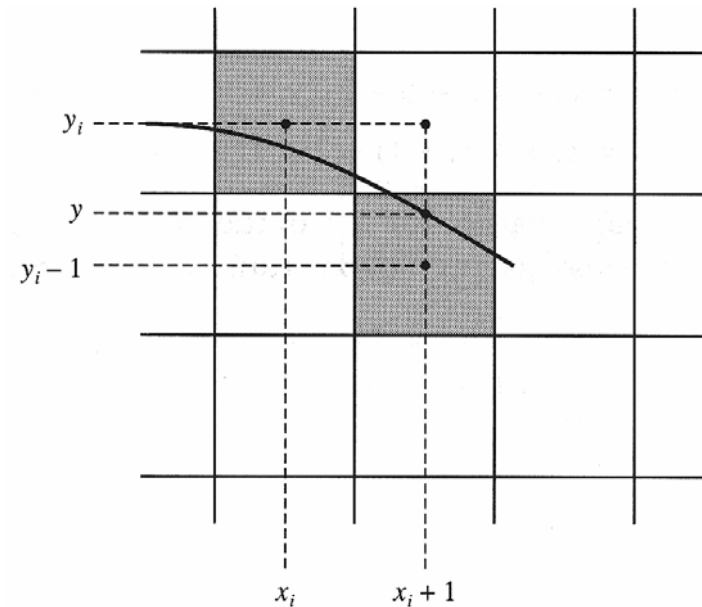
Αν  $e_i \geq 0$  επιλέγεται το σημείο  $(x_i+1, y_i-1)$

διαφορετικά » » »  $(x_i+1, y_i)$

Επειδή για  $x=x_i+1$  ισχύει  $y^2 = r^2 - (x_i+1)^2$  έχουμε:

$$e_i = y_i^2 - r^2 + (x_i + 1)^2 + (y_i - 1)^2 - r^2 + (x_i + 1)^2$$

$$= 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$$



```
circle (r, colour)
int r, colour;
{int x, y, e;
 x=0; y=r;
 e=3-2*r;
 while (x<=y)
 {circle_symmetry(x, y, colour);
 x++;
 if (e>=0) {y--; e=e-4*y}
 e=e+4*x+2;}
}
```



## Κύκλος: Αλγόριθμος Bresenham

Η τιμή  $e_{i+1}$  υπολογίζεται επαναληπτικά ως εξής:

$$\begin{aligned}e_{i+1} &= 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2 \\ &= 2(x_i + 2)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2 \\ &= 2x_i^2 + 8x_i + 8 + y_{i+1}^2 + y_{i+1}^2 - 2y_{i+1} + 1 - 2r^2 \\ &= 2(x_i + 1)^2 + 4x_i + 6 + 2y_{i+1}^2 - 2y_{i+1} + 1 - 2r^2 \\ &= e_i - y_i^2 - y_i^2 + 2y_i - 1 + 4x_i + 6 + 2y_{i+1}^2 - 2y_{i+1} + 1 \\ &= e_i + 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i)\end{aligned}$$

Για τον υπολογισμό του  $e_{i+1}$  χρησιμοποιείται το εξής τέχνασμα:

$$\text{Αν } e_i < 0 \Rightarrow y_{i+1} = y_i \Rightarrow e_{i+1} = e_i + 4(x_i + 1) + 2$$

$$\text{Αν } e_i \geq 0 \Rightarrow y_{i+1} = y_i - 1$$

$$\Rightarrow e_{i+1} = e_i + 4x_i + 6 + 2((y_i - 1)^2 - y_i^2) - 2(y_i - 1 - y_i)$$

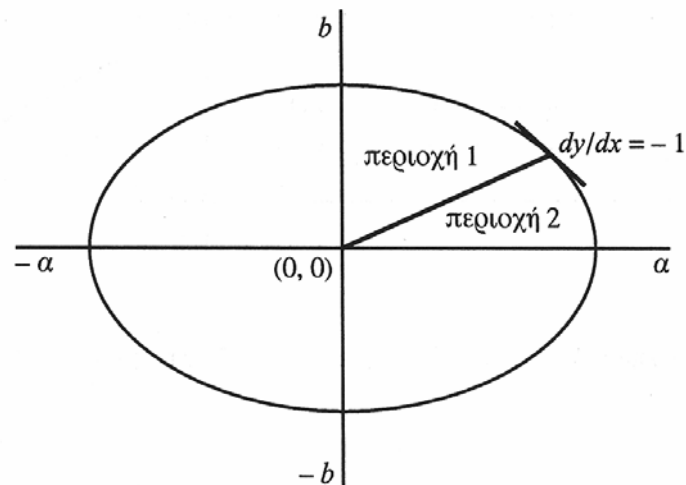
$$\text{ή } e_{i+1} = e_i + 4(x_i + 1) + 2 - 4(y_i - 1)$$

Θεωρώντας σαν πρώτο σημείο του 2ου οκταμορίου το σημείο  $(x,y)=(0,r)$

$$e_1 = 2 + r^2 + (r - 1)^2 - 2r^2 = 3 - 2r$$

## Ελλειψη

- Πολλοί αλγόριθμοι για κωνικές τομές:
  - Εδώ αλγόριθμος Αγάθου-Θεοχάρη-Μπεμ (1998).
  - Γρήγορος, μικρή απαίτηση ακρίβειας ακεραίων, σωστή μετάβασης περιοχής.
- Εξίσωση έλλειψης με κέντρο  $(0,0)$ :  $x^2/a^2 + y^2/b^2 = 1$ 
  - Τετραπλή συμμετρία: δημιουργούμε μόνο περιοχές 1, 2



## Ελλειψη

- Περιοχή 1: άξονας κύριας κίνησης ο  $X$ 
  - Εκκίνηση από  $(0, b)$
  - Τέλος περιοχής όταν  $dy/dx = -1$

Ορίζουμε:

$$d1 = y_i^2 - y^2$$

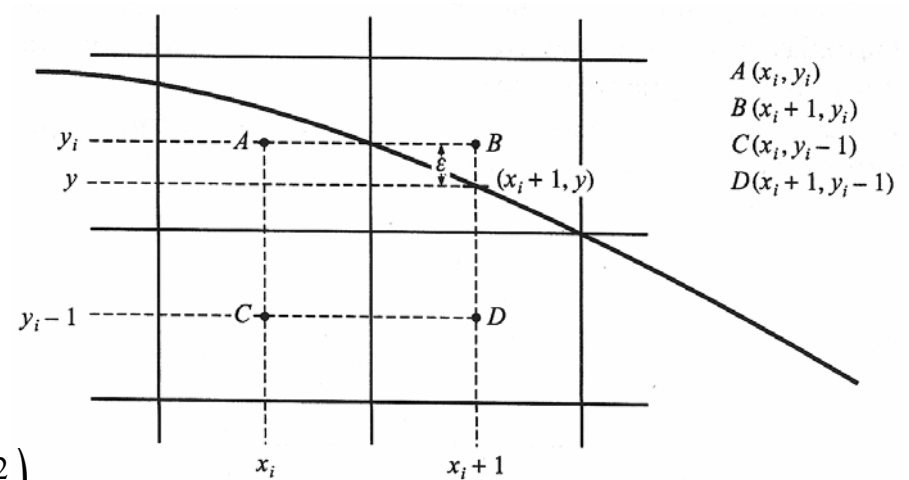
$$d2 = y^2 - (y_i - 1)^2$$

και θέτουμε:

$$d = d1 - d2 = (y_i^2 - y^2) - (y^2 - (y_i - 1)^2)$$

- Θέτουμε  $e = y_i - y$  οπότε:

$$d(\varepsilon) = -2\varepsilon^2 + 4y_i\varepsilon + 1 - 2y_i$$



## Ελλειψη

- Τιμή απόφασης:  $d(1/2) = 1/2$  δηλαδή για  $\varepsilon = 1/2$   
Αν  $d \leq 1/2$  επιλέγουμε pixel B  
διαφορετικά επιλέγουμε pixel D
- Διευκολύνουμε περαιτέρω αυξητικό υπολογισμό παίρνοντας  $d = a^2 (d_1 - d_2)$

$$d(\varepsilon) = -2a^2\varepsilon^2 + 4a^2y_i\varepsilon + a^2 - 2a^2y_i \quad (2.3) \text{ οπότε:}$$

$$\text{Αν } d \leq a^2/2 \text{ επιλέγουμε pixel B} \quad (2.4)$$

διαφορετικά επιλέγουμε pixel D

## Ελλειψη

- Αυξητικός υπολογισμός  $d$

$$\begin{aligned}d_{1,i} &= a^2(d1 - d2) \\ &= a^2 y_i^2 + a^2 (y_i - 1)^2 - 2a^2 y_i^2\end{aligned}\quad (2.5)$$

Ομως από την εξίσωση της έλλειψης για το σημείο  $(x_{i+1}, y)$  έχουμε  $a^2 y^2 = a^2 b^2 - b^2 (x_{i+1})^2$  οπότε:

$$d_{1,i} = -2a^2 b^2 + 2b^2 (x_i + 1)^2 + a^2 y_i^2 + a^2 (y_i - 1)^2\quad (2.6)$$

Στη συνέχεια ορίζουμε το  $d_{1,i+1}$  ως προς το  $d_{1,i}$ :

$$\begin{aligned}d_{1,i+1} &= -2a^2 b^2 + 2b^2 (x_{i+1} + 1)^2 + a^2 y_{i+1}^2 + a^2 (y_{i+1} - 1)^2 \\ &= -2a^2 b^2 + 2b^2 ((x_i + 1) + 1)^2 + a^2 y_{i+1}^2 + a^2 (y_{i+1} - 1)^2 \{x_{i+1} = x_i + 1\} \\ &= -2a^2 b^2 + 2b^2 (x_i + 1)^2 + 2b^2 + 4b^2 (x_i + 1) + a^2 y_{i+1}^2 + a^2 (y_{i+1} - 1)^2\end{aligned}$$

## Ελλειψη

Αλλά από την (2.6) έχουμε  $-2a^2b^2 + 2b^2(x_i + 1)^2 = d_{1,i} - a^2y_i^2 - a^2(y_i - 1)^2$ ,  
οπότε:

$$d_{1,i+1} = d_{1,i} + a^2y_{i+1}^2 + a^2(y_{i+1} - 1)^2 - a^2y_i^2 - a^2(y_i - 1)^2 + 2b^2 + 4b^2(x_i + 1)$$

Αν  $d_{1,i} > a^2/2$  τότε  $y_{i+1} = y_i - 1$  από την (2.4), οπότε:

$$d_{1,i+1} = d_{1,i} + 2b^2 + 4b^2(x_i + 1) - 4a^2(y_i - 1)$$

Αν  $d_{1,i} \leq a^2/2$  τότε  $y_{i+1} = y_i$  από την (2.4), οπότε:

$$d_{1,i+1} = d_{1,i} + 2b^2 + 4b^2(x_i + 1)$$

Η αρχική τιμή  $d_{1,0}$  βρίσκεται αντικαθιστώντας τις συντεταγμένες του πρώτου pixel της περιοχής 1  $(0, b)$  για τα  $(x_i, y_i)$  στην (2.6):

$$d_{1,0} = 2b^2 + a^2(1 - 2b)$$

## Ελλειψη

- Μετάβαση περιοχής:
  - Στηρίζεται στην τιμή του  $d$  για το σημείο  $(x_i+1, y_i-3/2)$
  - Αν πραγματική έλλειψη περνάει κάτω από αυτό  $\Rightarrow$  αλλαγή περιοχής
- Θέτοντας  $\varepsilon = 3/2$  στην (2.3)

$$d\left(\frac{3}{2}\right) = -2a^2\left(\frac{3}{2}\right)^2 + 4a^2y_i\left(\frac{3}{2}\right) + a^2 - 2a^2y_i = 4a^2(y_i - 1) + \frac{a^2}{2}$$

Αν  $d \leq 4a^2(y_i - 1) + \frac{a^2}{2}$  τότε παραμένουμε στην περιοχή 1  
διαφορετικά μεταβαίνουμε στην περιοχή 2.

- Παρομοίως βρίσκουμε μεταβλητή απόφαση περιοχής 2  $d_{2,i}$
- Αρχική τιμή μεταβλητής απόφασης περιοχής 2:  $d_{2,i}$  λαμβάνεται από τελική τιμή  $d_{1,i}$ :  $d_{2,i} = d_{1,i} + (d_{2,i} - d_{1,i})$   
 $d_{2,i} = d_{1,i} - a^2(2y_i - 1) - b^2(2x_i + 1)$

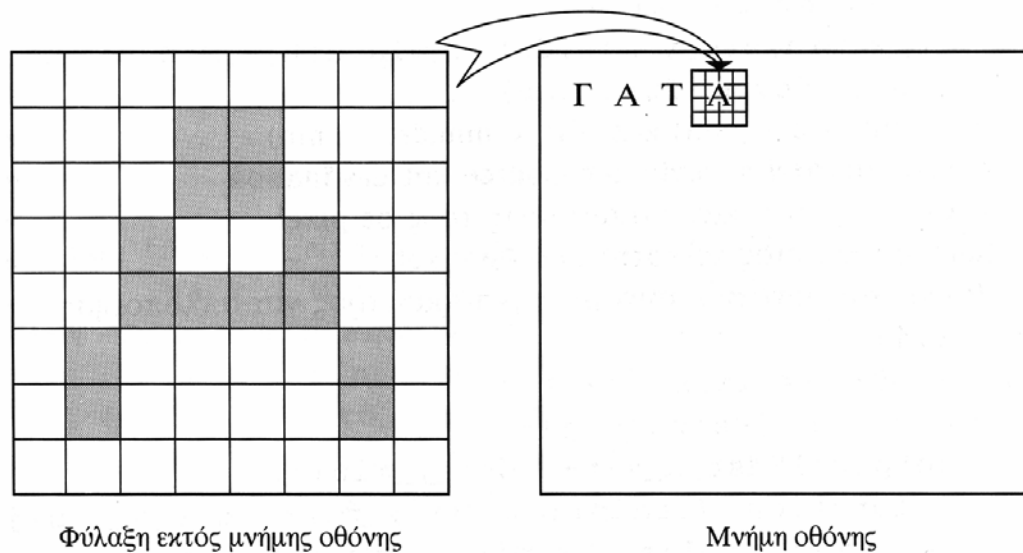
## Ελλειψη

```
Ellipse (a,b,colour)
int a,b,colour;
{int a_sqr,b_sqr,a22,b22,a42,b42,x_slope,y_slope,d,mida,midb,x,y;
  x=0; y=b;
  a_sqr=a*a; b_sqr=b*b;
  a22=a_sqr+a_sqr; b22=b_sqr+b_sqr;
  a42=a22+a22; b42=b22+b22;
  x_slope=b42; /*x_slope==(4*b^2)*(x+1) πάντα */
  y_slope=a42*(y-1); /*y_slope==(4*a^2)*(y-1) πάντα */
  mida=a_sqr>>1;
  midb=b_sqr>>1;
  d=b22-a_sqr-(y_slope>>1)-mida; /* αφαιρούμε a^2/2 για βελτιστοποίηση */
  /* περιοχή 1 */
  while (d<=y_slope)
    {setpixel(x,y,colour);
     if (d>0)
       {d=d-y_slope;
        y--;
        y_slope=y_slope-a42;}
     d=d+b22+x_slope;
     x++;
     x_slope=x_slope+b42;}
  /*Αλλαγή περιοχής*/
  d=d-(x_slope+y_slope)>>1+(b_sqr-a_sqr)+(mida-midb);
  /* περιοχή 2 */
  while (y>=0)
    {setpixel (x,y,colour);
     if (d<=0)
       {d=d+x_slope;
        x++;
        x_slope=x_slope+b42;}
     d=d+a22-y_slope; y--;
     y_slope=y_slope-a42;}
  }
```



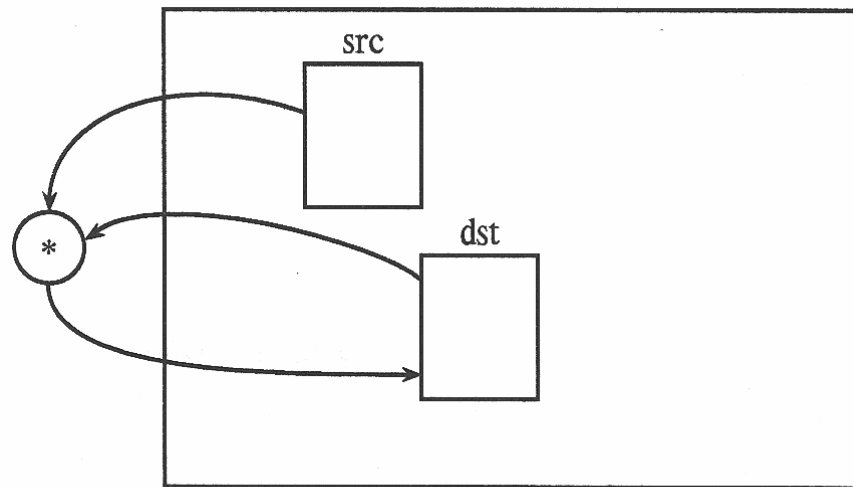
## RasterOp

- Πλεγματική οθόνη: εικόνα παριστάνεται στην μνήμη οθόνης
  - Πολλές λειτουργίες υλοποιούνται με μετακινήσεις ή/και συνδυασμούς στην μνήμη οθόνης.
  - π.χ. Επεξεργαστής κειμένου, παραθυρικές εφαρμογές:



## RasterOp

- RasterOp: γενική ρουτίνα μετακίνησης/συνδυασμού ορθογώνιων παραλληλόγραμμων τμημάτων εικόνας, ταχέως υλοποιημένη



## RasterOp

```
RasterOp(src_x_min, src_y_min, dst_x_min, dst_y_min, sizex, sizey, function)
int src_x_min, src_y_min, dst_x_min, dst_y_min, sizex, sizey;
    ftype function)
/* (src_x_min, src_y_min) και (dst_x_min, dst_y_min)          */
/* ή κάτω αριστερή γωνία των source και destination        */
/* sizex, sizey οι x και y διαστάσεις σε pixel              */
/* function κάποιος τελεστής μεταξύ pixel                  */
/* (θεωρείται δυνατή η συνάρτηση – παράμετρος για απλοποίηση) */
{int i, j;
  for (i=0; i<sizex; i++)
    for (j=0; j<sizey; j++)
      set_pixel(dst_x_min+i, dst_y_min+j,
        function(read_pixel(dst_x_min+i, dst_y_min+j)
          read_pixel(src_x_min+i, src_y_min+j)));
}
```

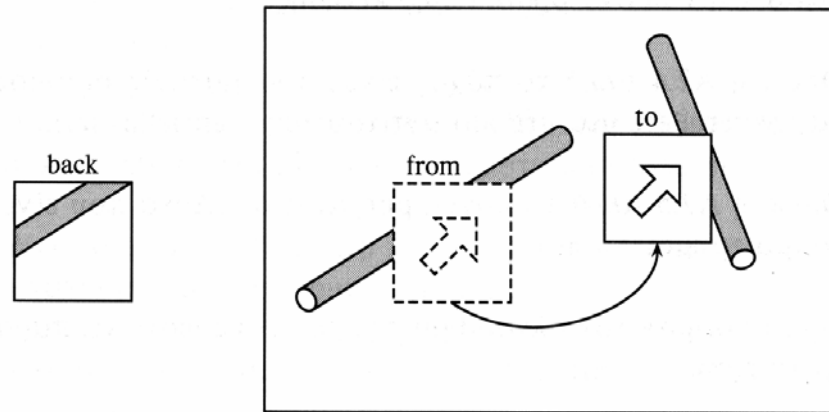
## RasterOp

- Ασπρόμαυρες εικόνες: function από τα 16 δυνατά
  - XOR ιδιαίτερα χρήσιμη για αντιμετάθεση source & destination χωρίς χρήση βοηθητικού χώρου:

```
exchange (src_x_min,src_y_min,dst_x_min,dst_y_min,sizex,sizey)
int src_x_min,src_y_min,dst_x_min,dst_y_min,sizex,sizey;
{RasterOp (src_x_min,src_y_min,dst_x_min,
           dst_y_min,sizex,sizey,XOR);
  RasterOp (dst_x_min,dst_y_min,src_x_min,
           src_y_min,sizex,sizey,XOR);
  RasterOp (src_x_min,src_y_min,dst_x_min,
           dst_y_min,sizex,sizey,XOR);
}
```

## RasterOp

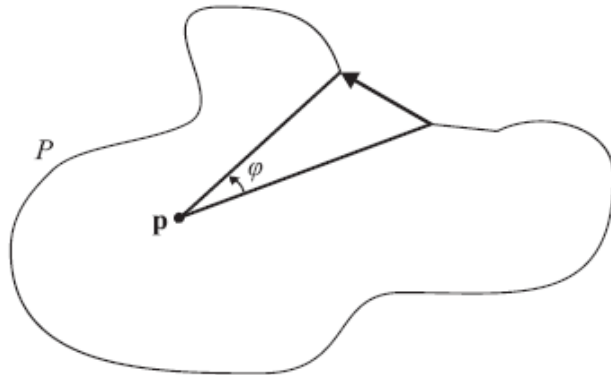
- `exchange` ιδιαίτερη χρήσιμη για διαδοχικές μετακινήσεις αντικειμένου (π.χ. Cursor)



```
move (from_x_min,from_y_min,to_x_min,to_y_min,  
      back_x_min,back_y_min,size_x,size_y)  
int from_x_min,from_y_min,to_x_min,to_y_min,  
    back_x_min,back_y_min,size_x,size_y;  
{exchange (to_x_min,to_y_min,  
           back_x_min,back_y_min,  
           size_x,size_y);  
 {exchange (from_x_min,from_y_min,  
           to_x_min,to_y_min,  
           size_x,size_y);  
}
```

## Σημείο Εντός Πολυγώνου

- 1. Αριθμός τομών ημιευθείας με περίμετρο
- 2. Winding number



- 3. Έλεγχος προσήμου (κυρτό πολύγωνο)

