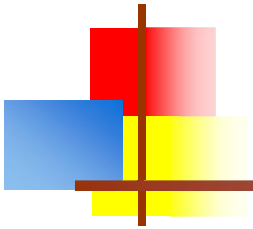
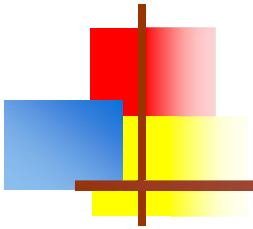
Algorithmes Graphiques de Base



PLAN



I) Tracé de Droites

- **Tracé d'un Pixel**

  void WritePixel (int X, int Y, unsigned char couleur);

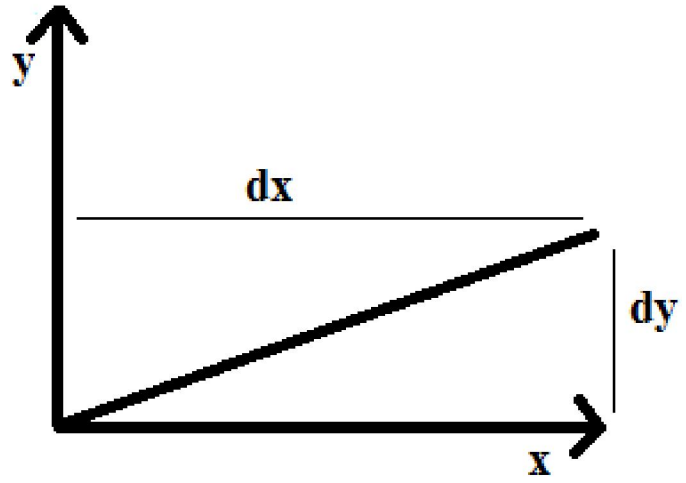
- **Discrétisation de segment de droite**

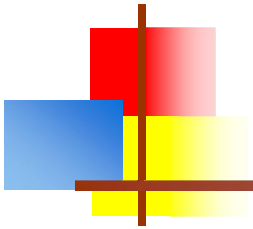
  - On cherche à dessiner un segment de droite d'un point **A (x_0, y_0)** à un point **B (x_1, y_1)** ou **B (x_0+dx, y_0+dy)**
  - On suppose que les segments ont une pente **|m| <= 1**, avec **m=Δy / Δx** (les segments de pente |m| > 1 sont obtenus avec de légères modifications).

I) Tracé de Droites

- **Algorithme incrémental fondamental**
- Equation de droite **AB: Y = m X + b**,

  tel que: { **|m|<1, dx > 0, dy > 0**}
- La méthode consiste à calculer pour chaque valeur de X, la valeur de Y par une opération d'agrandissement sachant que **Y = mX+b**, et **m = dy/dx**.

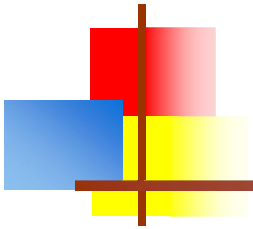




I) Tracé de Droites

- **void Dessin_Ligne (int x0, int y0, int x1, int y1, unsigned char couleur)**

```
{ /* on suppose -1<=m<=1, et x0 < x1*/
  /* x est incrémenté de x0 à x1 par pas d'une unité */
 int  x;    float dy, dx, y, m;
 dy= y1-y0;     dx= x1-x0;
 m = dy/dx;     y = y0;
```



I) Tracé de Droites

- **for (x=x0; x<=x1; x++)**

  { **WritePixel** (x, **round(y)**, couleur);

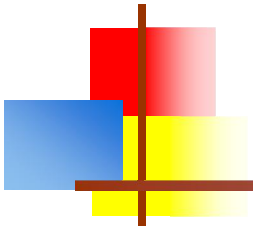
  y = y + m; /* avancer d'un pas de m */

  }

}

- **int round** (float val) /* fonction arrondi */

  { return (int) (val+0.5); }



I) Tracé de Droites

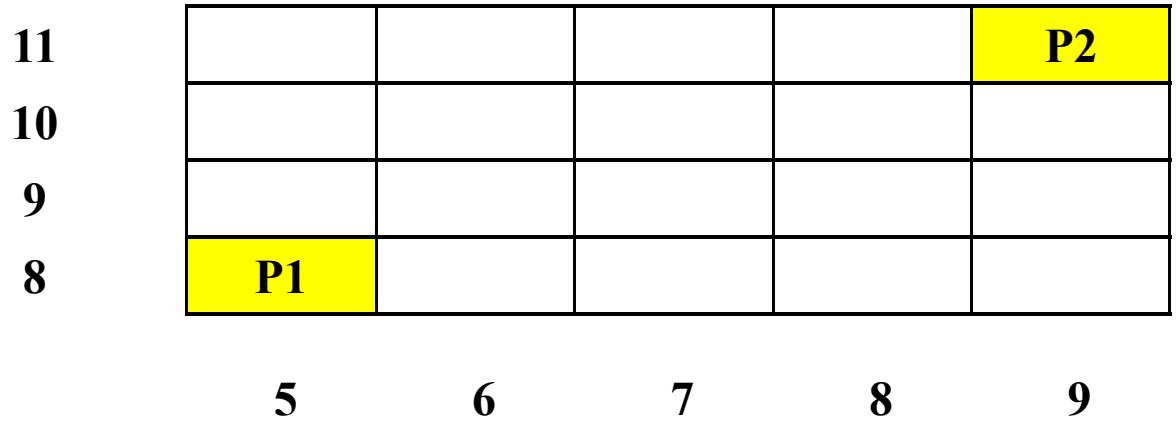
- **Remarques :**
- **Si |m| > 1,** un pas de **x** va générer un pas de y plus grand que **1**.
- Par conséquent, on doit inverser les rôles de **x** et **y**:
  - **en assignant** un pas d'une unité à **y** (mettre **y** dans la boucle **for**),
  - et **en incrémentant** x par **x + Δx**, avec **:**

$$\Delta x = \Delta y / m = 1 / m.$$

I) Tracé de Droites

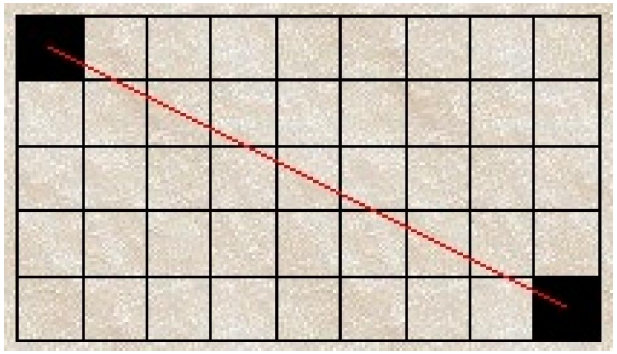
- **Exercice (application) :**

- Appliquer l'algorithme de tracé de droite entre le point (5, 8) et le point (9 , 11)

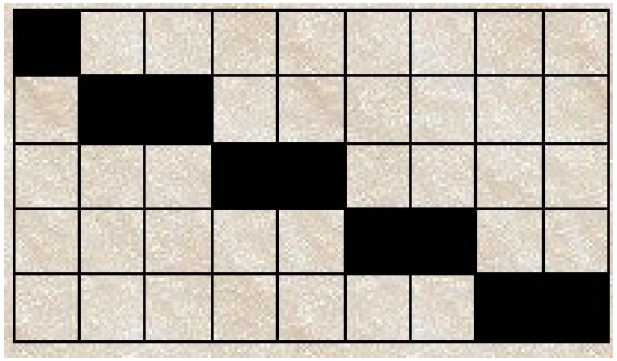


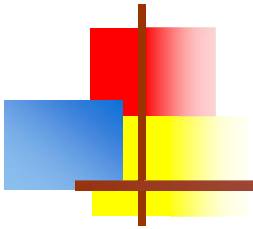
I) Tracé de Droites

- **Illustration :**
- On allume les 2 pixels extrémités de la diagonale
- La ligne rouge représente la diagonale mathématique



- On allume les pixels les plus proches de la ligne rouge





I) Tracé de Droites

- **En java :**

- On exploite la classe **Graphics** **(Graphics g)**
- Pour tracer une droite: **g.drawLine (x1, y1, x2, y2);**
- Pour tracer un point (x,y), on utilise pour le moment: g.drawLine (x, y, x, y); une autre méthode permet d'exploiter la classe **Point**
- Pour tracer un rectangle qui commence au point (x , y) avec une largeur w et une hauteur h: **g.drawRect (x, y, w, h)**

I) Tracé de Droites (**avec Applet**)

- **1) Code java avec la classe Graphics (et Applet):**

```
import java.applet.Applet;
import java.awt.*;
public class graphe extends Applet
{    void dessin (Graphics g)
      { for (int i=0; i<200; i=i+2)
            g.drawLine (1, i, 200, i); }
      public void paint (Graphics g)
       { dessin (g); } //la méthode paint existe dans Applet
} // fin du programme
```

I) Tracé de Droites (**avec JFrame**)

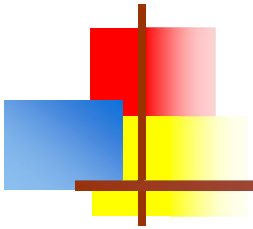
- **2è) Code java avec Graphics (et JFrame):**

```
import javax.swing.*;
import java.awt.*;
 public class graphe2 extends JFrame {
 public graphe2() {
      super (“Exemple de dessin dans un JFrame");
     setSize(480, 200);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLocationRelativeTo (null);
}
```

I) Tracé de Droites (**avec JFrame**)

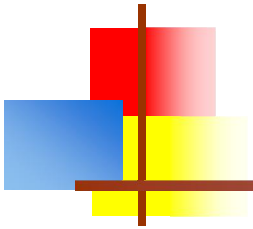
- **2è) Code java avec Graphics (et JFrame) :**

```
void dessin(Graphics g) {
        for (int i=0; i<200; i=i+2)
            g.drawLine (1, i, 200, i);     }
public void paint(Graphics g) {
        super.paint(g);  // paint existe dans JFrame
        dessin (g);    }
public static void main(String[] args) {
        new graphe2().setVisible(true);
 };     }
```



II) Tracé de Cercles

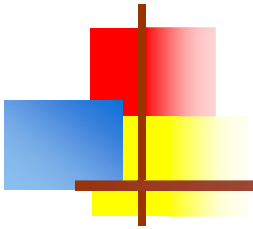
- **Principe :**
- **$x = r.\cos(\alpha)$, $y = r.\sin(\alpha)$**
- On trace le cercle en parcourant l'angle **α** (rotation de 0° à 360 degrés, ou de 0 à 2 π radians)
- Accroissement de l'angle **α** par pas de 1 degré environ.



II) Tracé de Cercles

- **void Dessin_Cercle (int ox, int oy, int r, unsigned char couleur)**

```
float x, y, angle;              angle = 0.0;
{ /* on suppose qu'on travaille avec les radians */
while (angle < 2*3.14159)
  {  x = r * cos (angle);
     y = r * sin (angle);
     angle = angle + 0.02;
   WritePixel (ox + round (x), oy + round (y), couleur);
}      }   // fin de la fonction
```



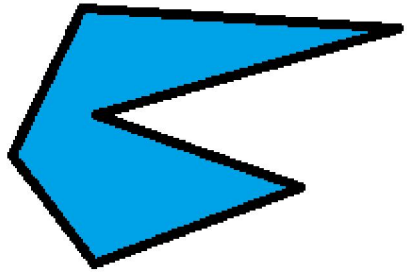
II) Tracé de Cercles

- **En java :**

- La fonction **drawOval (x, y, width, height)** de la classe **Graphics g** permet de tracer un cercle ou une ellipse délimité par une zone rectangulaire qui commence au point (x, y) avec la largeur **width** et la hauteur **height**
- Exemple: **g.drawOval (10, 20, 60, 80);**

III) Remplissage de Polygones

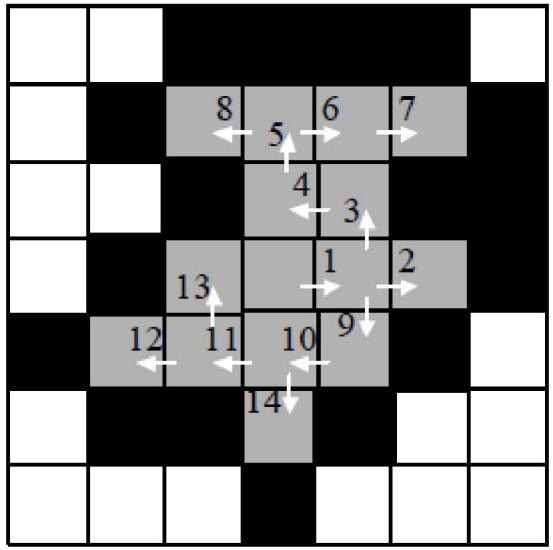
- **Méthode récursive de remplissage :**
- La fonction **RemplirPolygon** a comme paramètres un **point P** supposé être à l'intérieur du polygone et le **paramètre couleur**.
- **L'initialisation** commence par un point connu à l'intérieur du polygone.
- On utilise :
  - Une fonction **Frontiere (x , y)** qui indique **si** le point (x , y) à remplir est une frontière du polygone ou pas.
  - Une fonction **Rempli (x , y, couleur)** qui indique **si** le point a été déjà rempli par la couleur spécifiée

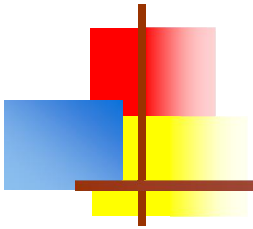


III) Remplissage de Polygones

void RemplirPolygone (int x, int y, int couleur)

```
{  if  ( (x < 0) || ( x > x_max) ) return;
   if y ( (y<0) || ( y > y_max) ) return;
if ( ! Frontiere (x , y) &&  ! Rempli (x , y, couleur) )
 {  WritePixel (x , y, couleur);
    RemplirPolygone (x-1 , y);
    RemplirPolygone (x+1 , y);
    RemplirPolygone (x , y-1);
    RemplirPolygone (x , y+1);
} }
```





III) Remplissage de Polygones

- **En java :**
- Dans la classe **Graphics g** :
- La fonction **g.drawPolygon (int [] xp, int [] yp, int nb_points)** permet de tracer le polygone avec comme paramètres un tableau de points **xp**, un tableau de points **yp** et le nombre de points **nb_points**.
- La fonction **g.fillPolygon (int [] xp, int [] yp, int nb_points)** permet de remplir le polygone avec les mêmes paramètres de **g.drawPolygon**.
- La couleur doit être déjà spécifiée par **g.setColor**