

---

## Chapitre 9

# Dessiner avec Java

16/03/04

Dessiner avec Java

page 1

---

## Redessiner

- Les méthodes précédentes sont appelées :
  - lorsque le système le juge nécessaire
    - le composant est rendu visible
    - le composant a changé de taille
    - un autre composant a bougé et a rendu visible une portion du composant
  - => *TestAutomaticRepaint.java*
  - Lorsque l'application le juge nécessaire via un appel à une méthode *repaint*

16/03/04

Dessiner avec Java

page 5

---

## Plan

- Introduction
- Classe Graphics
- Classe Graphics2D

16/03/04

Dessiner avec Java

page 2

---

## Méthodes repaint

- `repaint()`
  - redessine tout le composant
- `repaint(long milliseconds)`
  - demande que le composant soit redessiné avant `milliseconds`
- `repaint (int x, int y, int width, int height)`
  - redessine la portion du composant
- ...

16/03/04

Dessiner avec Java

page 6

---

## Introduction

16/03/04

Dessiner avec Java

page 3

---

## paint vs paintComponent

- Swing en fait implémente `paint` pour tout objet `JComponent` en faisant appel à :
  - `paintComponent`
  - `paintBorder`
  - `paintChildren`
- Il ne faut pas chercher (en général) à redéfinir `paintBorder` et `paintChildren`.

16/03/04

Dessiner avec Java

page 7

---

## Dessiner dans un composant

- Pour tout composant, il est possible de définir comment celui-ci doit être dessiné en redéfinissant une méthode.
- AWT
  - `public void paint(Graphics g)`
- Swing
  - `public void paintComponent(Graphics g)`

16/03/04

Dessiner avec Java

page 4

---

## paintComponent

```
public class MyPanel extends JPanel {
    protected void paintComponent(Graphics g) {
        // Let UI delegate paint first including
        // background filling, if I'm opaque)
        super.paintComponent(g);
        // paint my contents next....
    }
}
```

16/03/04

Dessiner avec Java

page 8

## Dessiner incrémentalement

- Redessiner (systématiquement) tout un composant
  - perte de temps
  - scintillement possible
- Il est préférable :
  - d'utiliser la méthode `repaint(x,y,width,height)`
  - de récupérer un contrôleur graphique et de l'utiliser
  - d'utiliser un timer

16/03/04

Dessiner avec Java

page 9

## Exemple

- Pour redessiner toutes les 20ms un composant donné, on écrira :  
Timer t = **new** Timer(20, **new** ActionListener()  
{  
    **public void** actionPerformed(ActionEvent e)  
    {  
        composant.repaint();  
    }  
});

16/03/04

Dessiner avec Java

page 13

## Récupérer un contrôleur graphique

- Il suffit d'appeler la méthode `getGraphics` (définie dans la classe `Component`).
- On peut alors utiliser ce contrôleur.
- Toutefois, les modifications apportées au graphisme sont perdues (à moins qu'elles ne soient gérées) lors d'un nouveau paint
- => *TestControleur.java*

16/03/04

Dessiner avec Java

page 10

## Classe Graphics

16/03/04

Dessiner avec Java

page 14

## Relâcher un contrôleur graphique

- Un contrôleur graphique peut être obtenu :
  - automatiquement : argument des méthodes `paint` ou `paintComponent`
  - À la demande : méthode `getGraphics`
- Dans le premier cas, il est relâché automatiquement à la fin de la méthode
- Dans le second cas, il faut le relâcher avec la méthode `dispose`.

16/03/04

Dessiner avec Java

page 11

## Classe Graphics

- Propose un ensemble de fonctionnalités simple mais limité.
- Les propriétés du contrôleur sont définies par :
  - `void setColor(Color c)`
  - `void setFont(Font font)`
  - `void setPaintMode()`
  - `void setXORMode(Color c1)`
  - `void setClip(int x, int y, int width, int height)`
  - ...

16/03/04

Dessiner avec Java

page 15

## Utiliser un Timer

- Un Timer permet d'effectuer une action à intervalle régulier.
- Il suffit d'enregistrer un `ActionListener` auprès de lui.
- On peut donc redessiner tout un composant de façon régulière.

16/03/04

Dessiner avec Java

page 12

## Modes de dessin

- `PaintMode` : mode (par défaut) qui permet de remplacer les pixels existants par ceux qui sont tracés.
- `XORMode` : mode qui permet de combiner les pixels existants avec ceux qui sont tracés.
  - on utilise en général la couleur de fond à l'appel
  - un second tracé efface toujours le précédent  
=> *TestXORMode.java*

16/03/04

Dessiner avec Java

page 16

## Clipping

- Toute opération de dessin en dehors de la clipping area est sans effet.
- A comparer avec la méthode repaint prenant les mêmes arguments.
- *TestClipping.java*

16/03/04

Dessiner avec Java

page 17

## Obtenir un contrôleur Graphics2D

- De façon automatique :

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2D = (Graphics2D)g;
    ... // dessiner avec g2D
```
- A la demande :

```
Graphics2D g2D = (Graphics2D)g.getGraphics();
```

16/03/04

Dessiner avec Java

page 21

## Fonctions de dessin

- drawRectangle
- drawRoundRect
- draw3DRect
- drawPolygon
- drawOval
- drawArc
- drawLine
- drawImage
- fillRectangle
- fillRoundRect
- fillDRect
- fillPolygon
- fillOval
- fillArc
- drawChars
- drawString

16/03/04

Dessiner avec Java

page 18

## Fonctions principales

- void draw(Shape s)
  - Dessiner une forme
- void fill(Shape s)
  - Remplir une forme
- Et shape ?

16/03/04

Dessiner avec Java

page 22

## Classe Graphics2D

16/03/04

Dessiner avec Java

page 19

## Shape

- Une classe générale qui représente toute forme.
- Cette classe abstraite possède de nombreuses sous-classes.
- Chaque sous-classe représente un type de formes dont certaines pouvaient être dessinées directement avec Graphics.

16/03/04

Dessiner avec Java

page 23

## Utiliser Graphics2D

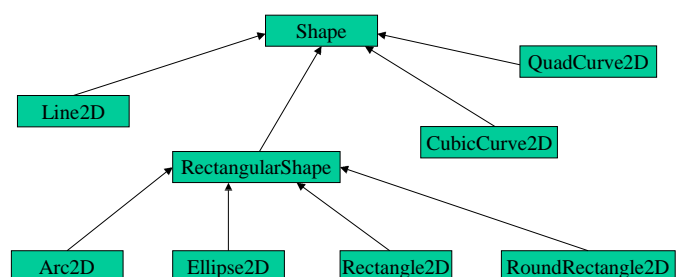
- Il est possible de manipuler des objets Graphics2D pour toute version de Java compatible Java 2D.
- Pour utiliser ces objets, il suffit d'effectuer un transtypage car de nombreuses fonctions de Java manipulent des références Graphics.
- Or Graphics2D est une sous-classe de Graphics et pour toute version Java compatible Java 2D, les objets retournés sont issus de cette sous-classe.

16/03/04

Dessiner avec Java

page 20

## Formes élémentaires



16/03/04

Dessiner avec Java

page 24

## Formes 2D à virgule flottante

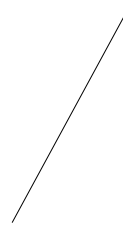
- Avec Java 2D, il est possible effectuer des conversions entre une mesure naturelle (en mètres par exemple) et une mesure en pixels.
- Pour cette raison, les coordonnées sont données en virgule flottante, c'est à dire en float ou double.

16/03/04

Dessiner avec Java

page 25

## Lignes



```
Graphics2D g2 = (Graphics2D)g;
Point2D p1 = Point2D.Double(100,0);
Point2D p2 = Point2D.Double(0,300);
Line2D line = new Line2D.Double(p1,p2);
g2.draw(line);
```

16/03/04

Dessiner avec Java

page 29

## Conséquence

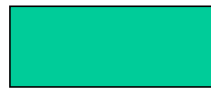
- La plupart des classes représentant des formes possèdent deux sous-classes statiques.
- L'une permet d'utiliser des coordonnées de type float
- L'autre permet d'utiliser des coordonnées de type double.

16/03/04

Dessiner avec Java

page 26

## Rectangles



```
g2.setPaint(Color.black);
g2.setStroke(new BasicStroke(2.0f));
Rectangle2D r = new Rectangle2D.Double(0,0,
200,50);
g2.draw(r);
g2.setPaint(Color.green);
g2.fill(r);
```



```
g2.setPaint(Color.black);
float[] dash = new Float(10.0f,10.0f);
g2.setStroke(new BasicStroke(1.0f,
BasicStroke.CAP_ROUND,
BasicStroke.JOIN_MITER,10.0f,dash,0.0f));
RoundRectangle2D r = new
RoundRectangle2D.Double(0,0, 180,50,5,5);
g2.draw(r);
```

16/03/04

Dessiner avec Java

page 30

## Point2D

- Point2D qui n'est pas sous-classe de Shape utilise aussi cette technique.
- Pour créer un point, il faut écrire :  
`Point2D p = new Point2D.Float(5.0f,10.0f);`  
ou encore :  
`Point2D p = new Point2D.Double(5.0,10.0);`

16/03/04

Dessiner avec Java

page 27

## Stroke

- Il s'agit du trait utilisé.
- Les caractéristiques les plus intéressantes sont :
  - la largeur du trait employé (premier argument)
  - le motif du trait (cinquième argument)
- Le motif du trait correspond à un tableau donnant alternativement la largeur du tiret puis la largeur d'un espace etc...

16/03/04

Dessiner avec Java

page 31

## Line2D

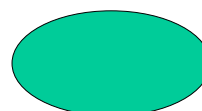
- Pour Line2D sous-classe de Shape, on écrira :  
`Line2D line = new Line2D.Float(p1,p2);`  
ou encore :  
`Line2D line = new Line2D.Double(p1,p2);`  
où p1 et p2 représentent deux points.
- De même pour les autres sous-classes.

16/03/04

Dessiner avec Java

page 28

## Ellipses et arcs



```
g2.setPaint(Color.black);
g2.setStroke(new BasicStroke(1.0f));
Ellipse2D e = new Ellipse2D.Double(0,0,
200,50);
g2.draw(e);
g2.setPaint(Color.green);
g2.fill(e);
```



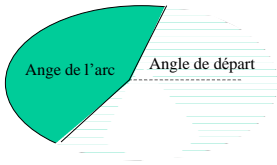
```
g2.setPaint(Color.black);
g2.setStroke(new BasicStroke(8.0f));
Arc2D a = new Arc2D.Double(0,0,
180,50,90,135,ARC2D.OPEN);
g2.draw(a);
```

16/03/04

Dessiner avec Java

page 32

## Arcs



- Lors de la construction; on précise :
  - l'angle de départ
  - l'angle de l'arc
  - le type de fermeture

16/03/04

Dessiner avec Java

page 33

## Exemple 1



```
int xs[] = {x, x+width, x, x+width};
int ys[] = {y, y+height, y+height, y};
GeneralPath polygon = new
GeneralPath(GeneralPath.WIND_EVEN_ODD);
polygon.moveTo(xs[0], ys[0]);
for (int i = 1; i < xs.length; i++)
    polygon.lineTo(xs[i], ys[i]);
polygon.closePath();
g2.setPaint(Color.red);
g2.fill(polygon);
```

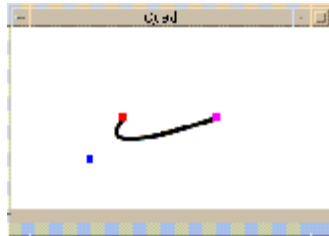
16/03/04

Dessiner avec Java

page 37

## Courbe quadratique

```
g2.setPaint(Color.black);
g2.setStroke(new BasicStroke(4.0f));
QuadCurve2D q = new QuadCurve2D.Double(startX,startY,
controlX, controlY, stopX, stopY);
g2.draw(q);
```



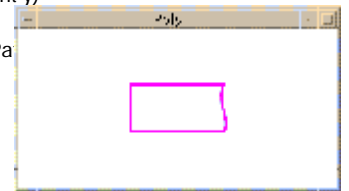
16/03/04

Dessiner avec Java

page 34

## Exemple 2

```
public GeneralPath createPath(int x, int y)
{
    GeneralPath path = new GeneralPa
    x2 = x; y2 = y;
    path.moveTo(x, y);
    x -= 100; path.lineTo(x, y);
    y += 50; path.lineTo(x, y);
    x += 100; path.lineTo(x, y);
    x += 10; y -= 10; x1 = x - 20; y1 = y - 20;
    path.curveTo(x, y, x1, y1, x2, y2);
    return path;
}
```



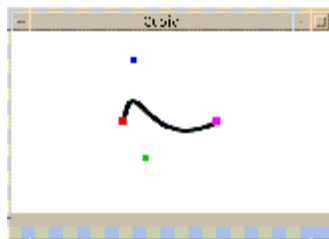
16/03/04

Dessiner avec Java

page 38

## Courbe cubique

```
g2.setPaint(Color.black);
g2.setStroke(new BasicStroke(4.0f));
CubicCurve2D q = new CubicCurve2D.Double(startX,startY,
control1X, control1Y, control2X, control2Y, stopX, stopY);
g2.draw(q);
```



16/03/04

Dessiner avec Java

page 35

## Remplissage

- Lorsqu'on remplit une forme, il est possible d'utiliser :
  - un objet Color
    - g2.setPaint(Color.pink);
  - un objet GradientPaint
    - g2.setPaint(new GradientPaint(p1,Color.red,p2,Color.blue));
  - un objet TexturePaint
    - g2.setPaint(new TexturePaint(bufferedImagep1,anchor));

16/03/04

Dessiner avec Java

page 39

## GeneralPath

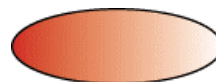
- GeneralPath, sous-classe de Shape, permet d'effectuer un tracé général.
- On utilise alors les méthodes :
  - moveTo
  - lineTo
  - quadTo
  - curveTo
  - closePath

16/03/04

Dessiner avec Java

page 36

## Exemple



```
Ellipse2D e = new Ellipse2D.Double(0,0,
200,50);
GradientPaint gp = new
GradientPaint(p1,Color.red,p2,Color.white);
g2.setPaint(gp);
g2.fill(e);
```

16/03/04

Dessiner avec Java

page 40