

Introduction à la programmation graphique

Le cas de Java2D

Mountaz Hascoët, Univ. Montpellier II

1

Plan

1. Introduction
 - Chaîne de rendu
2. Objets graphiques
 - Formes
 - Images bitmap
 - Texte
3. Contexte graphique
 - De `Graphics` à `Graphics2D`
4. Redéfinition du rendu pour un composant

Mountaz Hascoët, Univ. Montpellier II

2

Chaîne de rendu

- Le rendu peut être décomposé en plusieurs étapes
1. Définir la scène :
 - les objets graphiques qui la composent
 - Formes
 - Texte
 - Images
 2. Appliquer opérations – modifier le contexte graphique
 - transformations géométriques, clipping, composition – pour le calcul des couleurs.
 3. Effectuer le rendu
 - À l'écran
 - Sur un buffer hors-écran
 - À l'imprimante

Mountaz Hascoët, Univ. Montpellier II

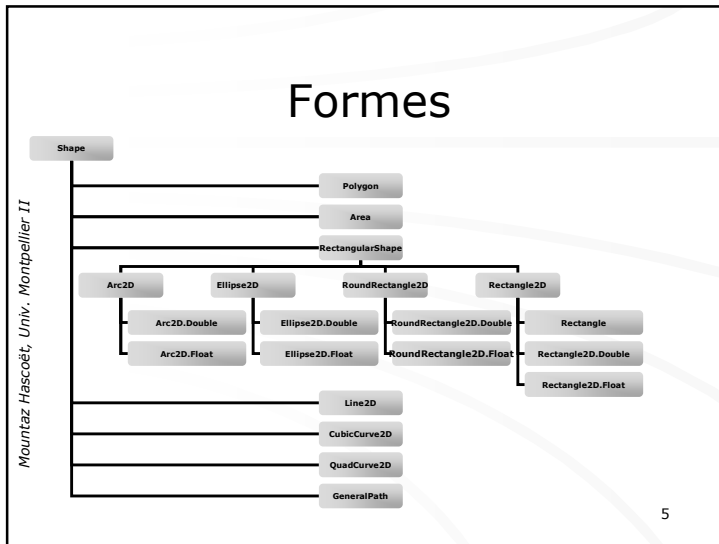
3

1. Les Objets Graphiques

Formes
Texte
Images

Mountaz Hascoët, Univ. Montpellier II

4



- ## Formes de bases
- Héritent de la classe RectangularShape
 - Rectangles, ellipses, arc.
 - Plus évolué
 - Polygone
 - Forme quelconque : GeneralPath
 - Courbes paramétriques:
 - Courbes de Béziérs
 - Courbes cubiques, quadratiques
- 6

- ## Texte: Rappel
- Font (Fonte):
 - ensemble de formes de caractère d'un même type (~ police de caractère)
 - Exemple: Georgia, Times New Roman, etc...
 - Glyph:
 - forme représentant un (ou plusieurs) caractères.
 - Ligature -> 1 glyph pour 2 caractères
 - fi
 - True Type (TT) et Postscript Type 1 (PS1)
 - Standards de définition des fontes concurrents
 - Description vectorielle des fontes
 - PS1 développées par Adobe et le plus ancien.
 - TT développées par Apple puis Microsoft.
- 7

- ## Texte
- Quelques caractéristiques standards sur les fontes
 - Famille:
 - Monospace (courier), sans-serif (helvetica) ou serif (Times, Georgia), etc...
 - Style:
 - regular, bold, italic, bold-italic
 - Fontes du système et fontes de l'application
 - Pour connaître les fontes installées sur votre système
 - Pour utiliser des fontes spécifiques pour une application:

```
GraphicsEnvironment.getLocalGraphicsEnvironment().getAllFonts();
java -Djava.awt.fonts=/Foo/Bar MonAppli.java
```
- 8

Texte: Rendu simple ou avancé

- Rendu simple (hérité de `Graphics`)


```
g2D.setFont(new Font(...));
g2D.drawString(« coucou », 40,50);
```
- Rendu plus élaboré


```
TextLayout, LineBreakMeasurer
```
- Rendu très élaboré


```
drawGlyphVector(GlyphVector g, float x,
float y)
```

9

Images

- Modèles
 - Le modèle « push »
 - Les images sont chargées par la classe `java.awt.Toolkit` et leur données sont traitées par une instance de `ImageConsumer` au fur et à mesure qu'elles arrivent.
 - Utile pour des applets ou application qui chargent les images sur internet par exemple
 - Le modèle « mode immédiat »
 - S'appuie sur la classe `awt.image.BufferedImage` issue de `Java2D`.
 - Les données de l'image sont immédiatement accessibles et l'on peut traiter ensuite.
 - Le modèle « pull »
 - Issu de `Java Advanced Imaging API`
 - Permet entre autres de ne pas traiter une image globalement mais par morceau pour les images qui peuvent ne pas tenir en entier en mémoire.

10

BufferedImage: représentation interne des images

- Principe
 - `BufferedImage` (extends `Image`)
 - Objectifs:
 - Représenter les couleurs de chaque pixel de l'image
 - Moyens
 - `Raster`
 - » Stockage de l'ensemble des valeurs des pixels
 - `ColorModel`
 - » Permet l'interprétation de la valeur des pixels stockées dans `Raster`
- Construction
 - Création explicite par les constructeurs de `BufferedImage`
 - Création par chargement d'un fichier image
 - `java.awt.Toolkit` et `java.awt.MediaTracker`
 - `com.sun.image.codec.jpeg`
 - et éventuellement `com.sun.glf.util`

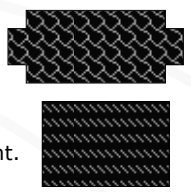
11

2. Contexte graphique

12

Le rôle du contexte graphique

- Principe général
 - Encapsule les informations utiles au rendu des objets graphiques.
 - Exemple: couleur de fond, style de traits
 - Informations variables d'une toolkit à l'autre
- Deux types de contextes graphiques dans Java: Graphics et Graphics2D
 - Graphics
 - contextes graphiques simples
 - « vieux » modèle issu de l'AWT
 - Graphics2D
 - contextes graphiques de Java2D
 - Plus récent et surtout plus intéressant.
 - Graphics2D extends Graphics
 - Ce cours concerne Graphics2D (PAS Graphics?..)



Contexte Graphique: Exemple

```

• Exemple:
Graphics2D g2D = (Graphics2D)
g;
g2D.setPaint(new
Color(102,0,204));
g2D.translate(70,100);
g2D.fill(new
Rectangle(200,100));

Ellipse2D.Double ellipse = new
Ellipse2D.Double(0,0,220,120);
g2D.setPaint(new Color
(255,51,204));
g2D.translate(70,60);
g2D.fill(ellipse);

g2D.setPaint(new Color
(255,204,102));
g2D.setStroke(new
BasicStroke(4));
g2D.draw(ellipse);

```



Contexte Graphique: caractéristiques

- 7 attributs sont définis dans un contexte graphique de type Graphics2D
- Certains servent pour tout:
 - Composite
 - Transform
 - Hint
 - Clip
- Certains ne servent que pour les formes
 - Paint
 - Stroke
- Un attribut ne sert qu'au texte
 - Font

Contexte Graphique: composite

- Composite
 - Détermine comment l'objet à dessiner doit se mélanger à ce qui est déjà dessiné. Utile pour rendre les objets transparents, faire des mélanges de couleurs.

```

• Exemple:
Graphics2D g2D = (Graphics2D) g;
rectangle = ...; ellipse = ...;
g2D.setPaint(new Color(102,0,204));
g2D.translate(70,100);
g2D.fill(rectangle);
g2D.setComposite(AlphaComposite.getInstance(
AlphaComposite.SRC_OVER,0.5f));
g2D.setPaint(new Color (255,51,204));
g2D.translate(70,60);
g2D.fill(ellipse);

g2D.setPaint(new Color (255,204,102));
g2D.setStroke(new BasicStroke(4));
g2D.draw(ellipse);

```



Contexte Graphique: composite

- Pas de mélange de couleur:
 - `g2D.setPaintMode();`
 - `g2D.setComposite(new AlphaComposite.SrcOver)`
- Mélange de couleurs
 - Principe:
 - Dessin du Pixel A sur le pixel B
 - Pixel A de couleur (RA, VA, BA, AA)
 - Pixel B de couleur (RB, VB, BB, AB)
 - Une règle de composition est appliquée pour donner un Pixel P de couleur:
 - $RP = RA \times CA + RB \times CB$
->(même calcul pour VR, BR, AR)
 - Où CA et CB sont des constantes définies par le modèle de composition
 - Il y a 8 modèles de composition définis dans AlphaComposite

Mountaz Hascoët, Univ. Montpellier II

17

Transparence: un cas particulier de mélange de couleur

- Modèle de composition à utiliser
 - AlphaComposite.SRC_OVER pour lequel
 - $CA = 1$
 - $CB = 1 - AB$
 - => $RP = RA + (1-AB)RB$ (avec les notations du transparent préc.)
 - Exemple:

```
g2D.setComposite(AlphaComposite.getInstance(
    AlphaComposite.SRC_OVER,Alpha));
```
 - A retenir:
 - transparence moyenne si $AB = 0.5$
 - Plus AB (valeur alpha du pixel B) est petit plus la transparence est grande

Mountaz Hascoët, Univ. Montpellier II

18

Contexte Graphique: Clip

- Objectifs
 - Déterminer quelle est la région visible de l'écran.
 - Par défaut, la région est toute la zone de dessin mais on peut la restreindre à n'importe quelle forme définie dans les formes.
 - Tout ce qui apparaît en dehors de la zone de clipping n'apparaît pas
- Principe
 - Définition d'une forme
 - Déclaration de cette forme comme zone de clipping

Mountaz Hascoët, Univ. Montpellier II

19

Clipping: Exemple

```
...
g2D.setClip(rectangle);
g2D.setPaint(new Color (255,51,204));
g2D.translate(70,60);
g2D.fill(ellipse);
...
```



Mountaz Hascoët, Univ. Montpellier II

20

Contexte Graphique: Transform

- Systèmes de coordonnées
 - Coordonnées utilisateurs
 - Coordonnées systèmes
 - Transformations explicites
 - Utilise des méthodes de la classe AffineTransform
 - Ou des raccourcis par la classe Graphics2D
`scale, rotate, translate, shear`
 - Composition des transformations
 - Matrices 3x3 et coordonnées homogènes
 - `T1.concatenate(T2)`
> $T1 = T1.T2$
 - `T1.preConcatenate(T2)`
> $T1 = T2.T1$
- NB: les méthodes `scale`, `rotate`, `translate` et `shear` font de la concaténation
> la dernière citée est la première appliquée.

21

Mountaz Hascoët, Univ. Montpellier II

Contexte Graphique: Hint

- Objectifs
 - Contrôle de divers paramètres du rendu qui ont un effet sur
 - La rapidité du rendu
 - Sa qualité
- Principe
 - Class `RenderingHints`
 - Les paramètres modifiables sont encapsulés par cette classe
 - Par exemple, elle permet d'activer ou non l'antialiasing
`g2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON)`

22

Mountaz Hascoët, Univ. Montpellier II

Contexte Graphique: Paint

- Modèles de remplissage des objets
 - 3 modèles de remplissage
 - `Color`, `GradientPaint`, `TexturePaint`
- Dégradé
 - Un dégradé est défini par 5 paramètres
 - Deux points: `D1`, `D2`
 - Deux couleurs: `C1` et `C2`
 - Une stratégie de parcours (optionnel)
 - Class `GradientPaint`
 - C'est elle qui encapsule ces informations.

23

Mountaz Hascoët, Univ. Montpellier II

3. Effectuer le rendu

- Modèles de propagation du dessin
 - Diffèrent légèrement entre les composants lourds (AWT) et les composants légers.
 - Pour les composants swing:
 - La méthode `Paint()` appelle par défaut
 - `paintComponent()`
 - `paintBorder()`
 - `paintChildren()`
 - Pour redéfinir le rendu du composant il faut le faire dans
 - `paintComponent()`
- > **c'est la bonne méthode à redéfinir pour les composants swing**
- La première chose à faire dans `paintComponent()` est d'appeler `super.paintComponent()`

24

Mountaz Hascoët, Univ. Montpellier II

Effectuer le rendu (suite)

- Pour appeler un rafraîchissement à la demande:
 - repaint()
 - repaint(Rectangle r) ne rafraîchit que la zone couverte par le rectangle
- Pour faire du rendu incrémental
 - Redéfinir Update()
 - Attention, il est très rare d'en avoir besoin et ça ne marche que sur les composants AWT
- Quelques recommandations
 - NE PAS appeler paint() directement
 - NE PAS appeler paintImmediately() directement
 - NE PAS redéfinir update (sauf à vouloir faire du dessin incrémental)

Récapitulons...

