Programmation graphique

Le cas de Java2D

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

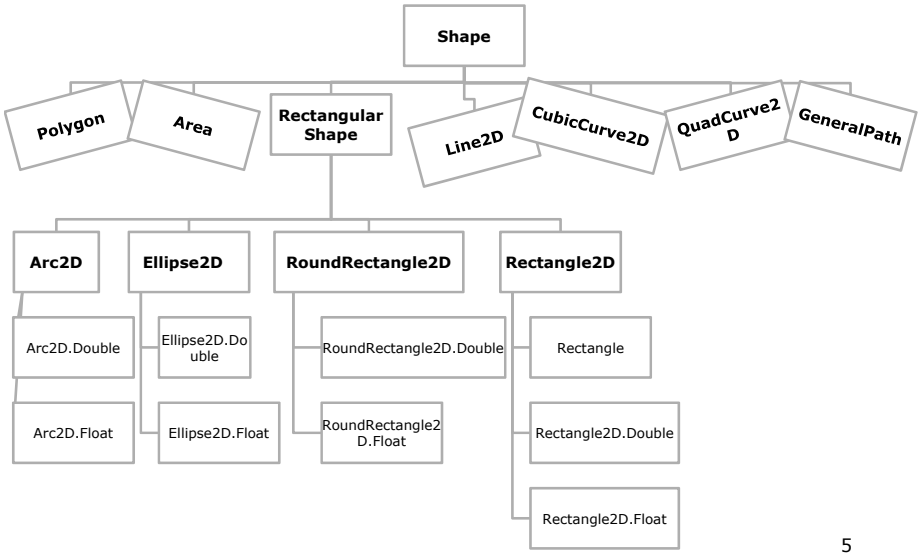
Chaîne de rendu

- L'affichage (rendu) peut être décomposé en plusieurs étapes
  1. Créer des objets graphiques
     - À partir de primitives graphiques de base
       - Formes simples
       - Texte
       - Images
  2. Appliquer des opérations de « décoration »
     = 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

1. Les Objets Graphiques

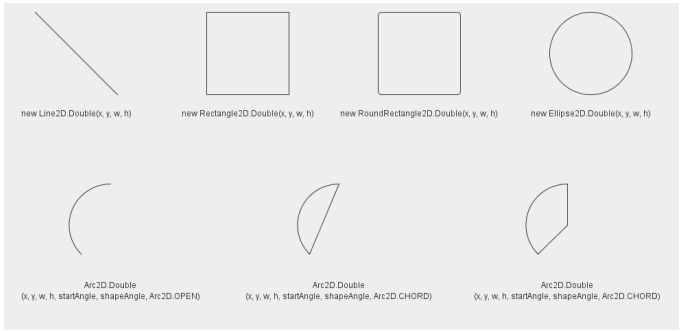
a) Formes
b) Texte
c) Images

Formes



Formes Simples

- Rectangles, RoundRectangle, Ellipses, Arc.
  - Héritent de la classe RectangularShape
- Line2D

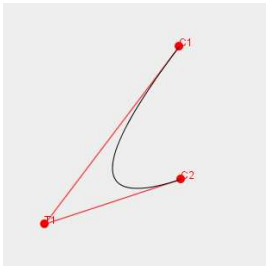


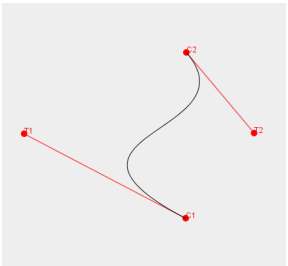
Formes élaborées (1/3)

- Courbes paramétriques
  « Une courbe paramétrique est une courbe définie par une fonction paramétrique sur un intervalle donné »
- Exemples
  - Cas particuliers simples
    - (x,y) = (cos(t), sin(t)) pour t dans [0,2𝕀]
    - (x,y) = (a + t * b, c + t * d) pour t dans [0,1]
  - Cas des courbes polynomiales
    - Courbes de Bézier
    - Hermite
    - B-Splines

Courbes de Bézier

- Existe à tous les degrés
  - Degré 2 – quadratiques – 3 pts de ctrl
  - Degré 3 – cubiques – 4 pts de ctrl





Formes élaborées (2/3)

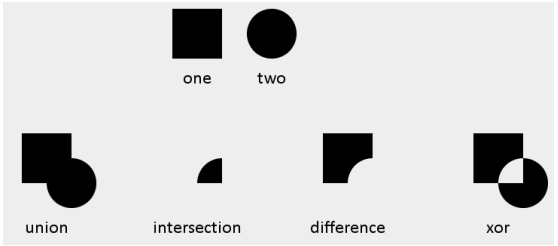
- Concaténation de
  - Forme simples
  - Lignes
  - Courbes de Bézier cubiques ou quadratiques

  > classe `GeneralPath`

Formes élaborées (3/3)

- Géométrie constructive (CAG, CSG)
  - Un objet est obtenu par composition d'opérations géométriques sur des objets plus simples

  Union (OR), Intersection (AND), Soustraction, XOR (difference symétrique)



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 Adode et le plus ancien.
  - TT développées par Apple puis Microsoft.

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 systèmes et fontes de l'application
  - Pour connaître les fontes installées sur votre système

```
GraphicsEnvironment.getLocalGraphicsEnvironment().getAllFonts();
```

  - Pour utiliser des fontes spécifiques pour une application:

```
java –Djava.awt.fonts=/Foo/Bar MonAppli.java
```

Texte

- Rendu simple (hérité de `Graphics`)

```
g2D.setFont(new Font(…));
g2D.drawString(« coucou », 40,50);
```

- Rendu multi-lignes

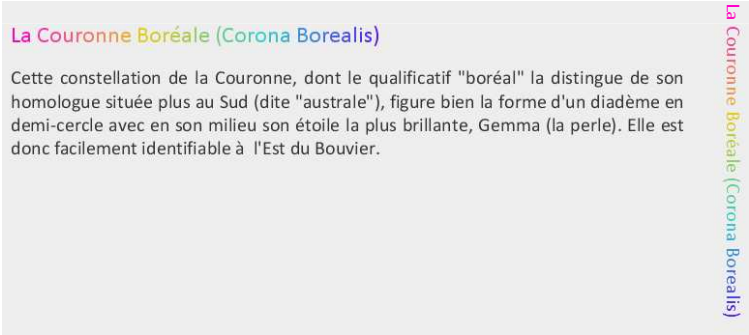
  `TextLayout, LineBreakMeasurer`

- Rendu plus élaboré

  `GlyphVector`

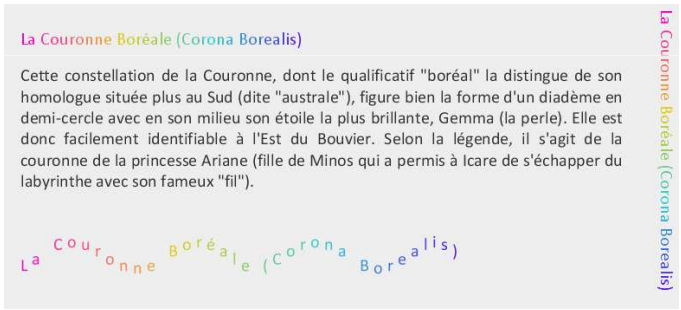
Texte – Exemple 1

- `AttributedString`
- `LineBreakMeasurer et TextLayout`



Texte – Exemple 2

- `GlyphVector`



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.

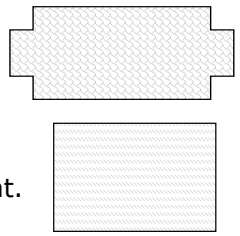
BufferedImage: représentation interne des images

- 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`

2. Contexte graphique

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…)



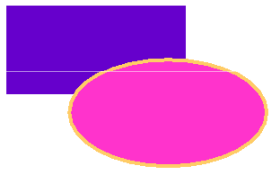
Contexte Graphique: Exemple

- 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

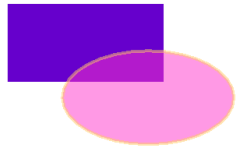
Contexte Graphique: composite

- 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.getIns
tance(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 x CA + RB x 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

Transparence:
un cas particulier de mélange de couleur

- Modèle de composition à utiliser
  - AlphaComposite.SRC_OVER pour lequel
    - CA = AB
    - CB = 1 - AB

    => RP = AB * 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

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

Clipping: Exemple

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

Contexte Graphique: Transform

- Systèmes de coordonnées
  - Coordonnées utilisateurs
  - Coordonnées écran
- 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.

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_ANTIALIASI
        NG,RenderingHints.VALUE_ANTIALIAS_ON)
      ```

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.

3. Faire l'affichage

- 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 l'affichage du composant
    - Redéfinir paintComponent()
      -> c'est La bonne méthode à redéfinir pour les composants swing
    - Dans paintComponent() on peut commencer par rappeler super.paintComponent()

Affichage (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…