Cours 3b :
Dessin 2D et Java2D

jgarcia@ircam.fr

(présentation basée sur des transparents de Fanis Tsandilas)

Dessin 2D

Dessin vectoriel

Utilisation de primitives géométriques : points lignes, courbes …

Primitives crées avec des équations

Peut être zoomé, déplacé, transformé sans perte de qualité

Dessin Bitmap

Utilisation de pixels

Les opérations d'échelles affectent la qualité

Stockage dans des fichiers d'image

Java 2D

Contexte de dessin graphique 2D avec texte et images

- Large gamme de primitives géométriques
- Les mécanismes de détection de collisions de formes, texte, images
- Couleur et transparence
- Transformations
- Impression
- Le contrôle de la qualité du rendu

Java2D : classes de base

La classe *Graphics* : classe abstraite pour tous les contextes graphiques qui permet aux applications de dessiner dans des composants.

```
public class RectWidget extends JPanel {
	private int posx, posy, w, h;
	private Color color;

	public RectWidget(int x, int y, int w, int h, Color color){
		this.posx = x;
		this.posy = y;
		this.w = w;
		this.h = h;
		this.color = color;
	}

	public void paint(Graphics g) {
		g.setColor(color);
		g.drawRect(x, y, w, h);
	}
}
```

Java2D : classes de base

Méthodes de la classe JComponent :

public paint(Graphics g)
protected paintComponent(Graphics g)
protected paintBorder(Graphics g)
protected paintChildren(Graphics g)

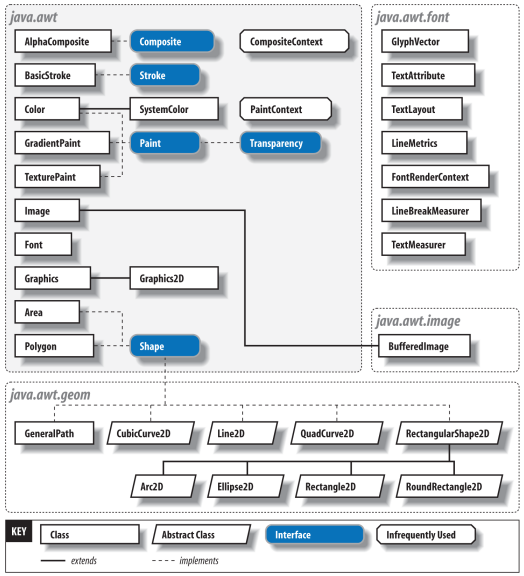
public print(Graphics g)
protected printComponent(Graphics g)
protected printBorder(Graphics g)
protected printChildren(Graphics g)

Java2D : classes de base

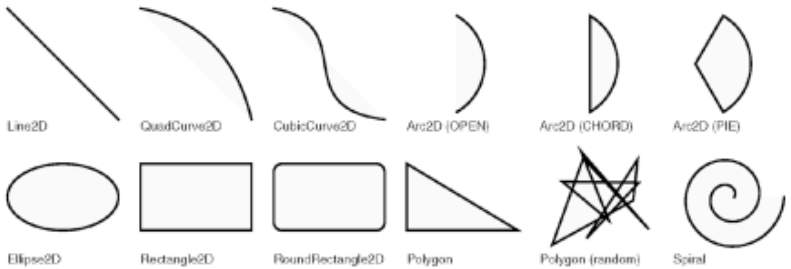
La classe *Graphics2D* étends la classe *Graphics* avec des contrôles plus sophistiqués sur la géométrie, les transformations …

```
private double x, y, w, h;
 ...
public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D)g;
    g2.draw(new Rectangle2D.Double(x , y, w, h));
}
```

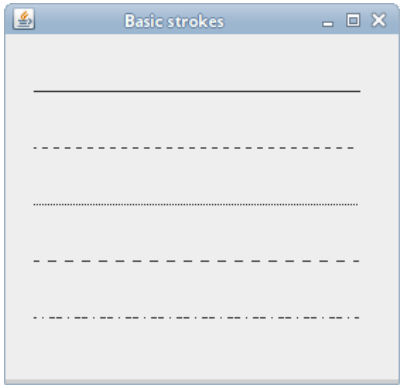
Java2D



Java2D : Shapes

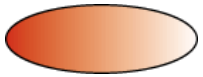


Java2D : Traits

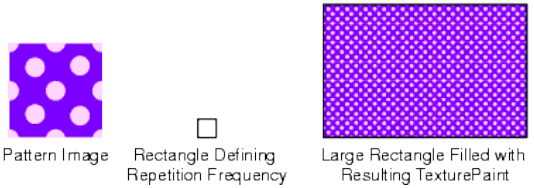



```
float[] dash1 = { 2f, 0f, 2f };
float[] dash2 = { 1f, 1f, 1f };
float[] dash3 = { 4f, 0f, 2f };
float[] dash4 = { 4f, 4f, 1f };

BasicStroke bs1 = new BasicStroke(1, BasicStroke.CAP_BUTT,
    BasicStroke.JOIN_ROUND, 1.0f, dash1, 2f );

g2d.setStroke(bs1);

g2d.drawLine(20, 80, 250, 80);
…
```

Java2D : Remplissage



```
// fill Ellipse2D.Double
redtowhite = new GradientPaint(0,0,color.RED,100, 0,color.WHITE);
g2.setPaint(redtowhite);
g2.fill (new Ellipse2D.Double(0, 0, 100, 50));
```



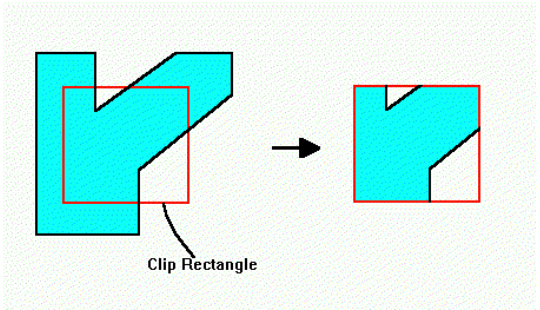
Options de rendu et anti aliasing

```
public void paint (Graphics g){
      Graphics2D g2 = (Graphics2D)g;
      RenderingHints rh = new RenderingHints(RenderingHints.KEY_TEXT_ANTIALIASING,
      RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
      g2.setRenderingHints(rh);
…
}
```

Clipping

Définir une portion à dessiner.

```
rect.setRect(x + marginx, y + marginy, w, h);
g2.clip(rect);
g2.drawImage(image, x, y, null);
```



Transformations

rotate, scale, translate, shear

g2.translate(100, 200);

La classe *AffineTransform*

atransf = new AffineTransform();

// rotate 90°

atransf.rotate(Math.PI/2);

g2.transform(atransf);

Transformations Affines