Interfaces graphiques - JavaFX

responsable : Wiesław Zielonka
zielonka@liafa.univ-paris-diderot.fr
http://liafa.univ-paris-diderot.fr/~zielonka

March 8, 2016

Définir un filtre ou handler pour le clavier

```
EventHandler<KeyEvent> keyEventHandler
                    = new EventHandler<KeyEvent>() {
    public void handle(final KeyEvent keyEvent) {
        //recuperer le caractere tape sur le clavier
        String s = event.getCharacter();
        if (event.isControlDown()) {
                System.out.println("CONTROL_DOWN");
        }
    } };
node.addEventHandler(KeyEvent.KEY_TYPED, keyEventHandler);
```

Si l'évènement est de type `keyEvent.KEY_TYPED` alors la méthode `KeyEvent.getCharacter()` retourne un String contenant le caractère tapé par l'utilisateur sur le clavier.

Pour les évènements `KEY_PRESSED` et `KEY_RELEASED` cette méthode retourne toujours `CHAR_UNDEFINED`, donc elle est inutile pour ces deux évènements.

La classe KeyEvent possède les méthodes booléennes `isControlDown()`, `isAltDown()`, `isShiftDown()`, `isMetaDown()` qui permettent de tester si on a appuyé sur une de ces touches.

`KeyEvent` possède aussi la méthode

```
KeyCode getCode()
```

qui retourne le code de la touche tapé par l'utilisateur.

Mais KeyCode retourné par `getCode()` est utile uniquement pour les événements `KeyEvent.KEY_PRESSED` et `KeyEvent.KEY_RELEASED`.
Pour l'évènement `KeyEvent.KEY_TYPED` `getCode()` retourne `KeyCode.UNDEFINED`.

Transformations

La classe `Transform` possède les sous classes suivantes :

- Translation `Translate`
- Rotation `Rotate`
- Scalling `Scale`
- Shearing `Shear`

La liste de transformations d'un noeud

```
Rectangle rec = new Rectangle(150, 80);
rec.setY(30);
Translate tra = new Translate(200, 0, 0);
Rotate rot = new Rotate(45, 0, 0, 0, new Point3D(0, 0, 1));
Translate trb = new Translate(300, 0, 0);
rec.getTransforms().add(tra);
rec.getTransforms().add(rot);
rec.getTransforms().addAll(trb);
//rec.getTransforms().addAll(tra, rot, trb);
Line line1 = new Line(0, 40, 400, 40);
Line line2 = new Line(0, 80, 400, 80);
Group group = new Group();
group.getChildren().addAll(line1, line2, rec);
```

La méthode getTransforms() appliquée à un noeud retourne la liste de transformation. Toutes les transformations sur cette liste sont exécutées avant que le noeud soit dessiné sur l'écran.

Translate

Constructeurs

```
Translate(double x, double y, double z)
Translate(double x, double y)
```

Le vecteur (x, y, z) donne la direction de la translation.

Rotate

Constructeurs :

```
Rotate(double angle,
       double pivotX, double pivotY, double pivotZ,
       Point3D axis)
Rotate(double angle, double pivotX, double pivotY)
```

L'angle de la rotation en dégrées.

L'algorithme de la rotation :

(a) le noeud est déplacé pour que le point pivot se retrouve dans $(0, 0, 0)$,

(b) on fixe l'axe de la rotation qui passe par le point $(0, 0, 0)$ et le point `axis`,

(c) on effectue la rotation,

(d) on déplace le noeud pour que le pivot retrouve son ancienne position.

QuadCurve

`QuadCurve` - la courbe quadratique de Bézier. Constructeur :

```
QuadCurve(double startX, double startY,
          double controlX, double controlY,
          double endX, double endY)
```

Lier les trois points de la courbe aux trois cercles

```
QuadCurve quadCurve = new QuadCurve();
{
    quadCurve.setStrokeWidth(2);/* largeur de courbe*/
    quadCurve.setFill(null);/* pas de remplisage */
    quadCurve.setStroke(Color.BROWN);/* couleur*/
}
Circle debut = new Circle();
Circle control = new Circle();
Circle end = new Circle();
{
    debut.setRadius(RADIUS);
    control.setRadius(RADIUS);
    end.setRadius(RADIUS);
}
```

Lier les trois points de la courbe aux trois cercles (bind)

```
quadCurve.startXProperty().bind(debut.centerXProperty());
quadCurve.startYProperty().bind(debut.centerYProperty());

quadCurve.controlXProperty().bind(control.centerXProperty());
quadCurve.controlYProperty().bind(control.centerYProperty());

quadCurve.endXProperty().bind(end.centerXProperty());
quadCurve.endYProperty().bind(end.centerYProperty());
```

Lier les trois points de la courbe ... (installer handlers)

```
EventHandler<MouseEvent> handler =
  new EventHandler<MouseEvent>() {

        @Override
        public void handle(MouseEvent event) {
            ((Circle) event.getSource())
                        .setCenterX(event.getX());
            ((Circle) event.getSource())
                        .setCenterY(event.getY());
        }
};

debut.setOnMouseDragged(handler);
control.setOnMouseDragged(handler);
end.setOnMouseDragged(handler);
```

Transformations sur un groupe de noeuds

```
Group group = new Group();
Rotate rotate = new Rotate(0, 0, 0, 0, new Point3D(0, 0, 1));
Transform precedent = null;
final Translate translate = new Translate(0,0,0);

group.getTransforms().addAll(translate, rotate);
```

L'idée – dans la liste de transformation de `group` :

- en cas de translation, au lieu d'ajouter une nouvelle translation modifier la translation `translate` qui se trouve sur la liste de transformations,
- en cas de la rotation : ajouter la nouvelle rotation dans la liste.

Transformations sur un groupe de noeuds (suite)

Mettre `group` dans un `panel`. Installer un handler sur le panel pour l'événement `MOUSE_CLIKED` :

```
pane.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
     /* clic donne la postion du pivot de la nouvelle rotation
      p - la position de clic dans les coordonnees de group
*/
     Point2D p = group.parentToLocal(
                    new Point2D(event.getX(), event.getY()));
     //nouvelle rotation
     rotate = new Rotate(0, p.getX(), p.getY(), 0,
                         new Point3D(0, 0, 1));
     //ajouter la nouvelle rotation de 0 degree
     group.getTransforms().add(rotate);
```

KeyHandler pour les translation et rotations

```
pane.addEventFilter(KeyEvent.KEY_PRESSED,
                    new EventHandler<KeyEvent>() {
    @Override
    public void handle(KeyEvent event) {
      final double delta = 2;

      KeyCode code = event.getCode();

      if (code == KeyCode.LEFT) {
         translate.setX(translate.getTx() - delta);
      } else if (code == KeyCode.RIGHT) {
         translate.setX(translate.getTx() + delta);
      } else if (code == KeyCode.UP) {
         translate.setY(translate.getTy() - delta);
      } else if (code == KeyCode.DOWN) {
         translate.setY(translate.getTy() + delta);
      } else if (code == KeyCode.A) {
         rotate.setAngle(rotate.getAngle() - delta);
      } else if (code == KeyCode.Z) {
         rotate.setAngle(rotate.getAngle() + delta);
      }
```

KeyHandler pour les translation et rotations

Touches A et Z pour les rotations et les flèches pour les translations.

Le problème : si on fait beaucoup de rotations alors la liste de transformations devient de plus en plus longue.

Le remède : modifier le code dans le handler pour le clic de la souris comme suit:

Recoller des transformations

```
Point2D p = group.parentToLocal(new Point2D(event.getX(),
                                            event.getY()));
/* le code suivant c'est juste pour avoir
    la liste de rotation plus courte */
/* precedent et rotation - les references
   sur les deux dernieres rotations */
if(precedent != null){
  /* supprimer les deux  dernieres rotations */
  group.getTransforms().removeAll(rotate, precedent);
  /* et les recoller ensemble */
  precedent = precedent.createConcatenation(rotate);
  /* remettre le resultat sur la liste de transformations */
  group.getTransforms().add(precedent);
}else{
  precedent = rotate ;
}
rotate = new Rotate(0, p.getX(), p.getY());
group.getTransforms().add(rotate);
```

Comment trouver le noeud sélectionné à l'aide de la souris?

Pane contient plusieurs noeud.
Chaque noeud peut subir une rotation.
Les rotation doivent être gérées à l'aide du clavier par le `pane` qui contient les noeuds.
Le pivot de la rotation se trouve toujours à l'intérieur du noeud qui subit la rotation.

Solution : installer deux filtres (ou deux handlers) pour gérer la rotation, les deux attachés au panel :

- un filtre pour `MouseEvent.MOUSE_CLICKED` pour mémoriser noeud à tourner,
- un filtre pour `keyEvent.KEY_PRESSED` qui effectue la rotation du noeud sélectionné par le premier filtre.

Comment trouver le noeud sélectionné à l'aide de la souris?

```
Node pickedNode ; /*noeud a tourner*/
EventHandler<MouseEvent> pickedFilter
                    = new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent e) {

    /* recuperer un noed enfant source de clic */
    pickedNode = e.getPickResult().getIntersectedNode();

    pane.requestFocus();
    }
};
pane.addEventFilter(MouseEvent.MOUSE_CLICKED, pickedFilter);
```

`e.getPickResult().getIntersectedNode()`

retourne le noeud enfant qui est la source du clic.

Tourner un noued

On suppose que chaque noeud à pivoter possède déjà une rotation comme le premier élément de sa liste de transformations. Il faut juste ajouter ou soustraire un delta à cet angle.

Les noeuds tourne autour leurs centres.

Tourner un noeud

```
EventHandler<KeyEvent> rotateFilter
                    = new EventHandler<KeyEvent>() {
  @Override
  public void handle(KeyEvent event) {
  final double delta = 2;
  if (pickedNode == null ||
       pickedNode.getTransforms().isEmpty()) {
    return;
  }
  KeyCode code = event.getCode();
  Rotate rot = (Rotate) pickedNode.getTransforms().get(0);
  if (code == KeyCode.LEFT) {
     rot.setAngle(rot.getAngle() + delta);
  } else if (code == KeyCode.RIGHT) {
     rot.setAngle(rot.getAngle() - delta);
  }
  Bounds bounds = pickedNode.getBoundsInLocal();
  double x = (bounds.getMaxX() + bounds.getMinX()) / 2;
  double y = (bounds.getMaxY() + bounds.getMinY()) / 2;
  rot.setPivotX(x);
  rot.setPivotY(y);
  }
```