Java Avance

Java FX

Emmanuel ADAM

Université Polytechnique des Hauts-De-France

UPHF/INSA HdF

Les concepts de JAVA FX

Java FX : applications graphiques évoluées

- Java FX, initialement créé par SUN, a été intégré à Java depuis la version 8.
- Java FX propose :
  - une API (des classes java) pour la gestion de graphiques, fenêtres
  - un logiciel de création de fenêtres de dialogue *'Scene Builder'*, générant des fichiers FXML
  - un composant de lecture de page web (WebView)
  - un support '*MultiTouch*'
  - utilisation de 'PRISM' pour un accès direct à la carte graphique
  - des composants pour la conception d'images 3D
  - des liens vers des composants '*Swing*' (ancien mode de création d'interfaces graphiques)
  - ...

Java FX : Théâtre, Scènes et Acteurs

Java FX : un pièce de théâtre

- Une Application JavaFX étend la classe '*javafx.application.Application*'
- Elle repose sur un théâtre ('*javafx.stage.Stage*')
- sur lequel se déroule des scènes ('*javafx.scene.Scene*')
- qui contiennent des groupes d'acteurs ('*javafx.scene.Group*')
- ces acteurs sont des composants ('shape')
  - cercle ('*javafx.scene.shape.Circle*')
  - Rectangle, Ligne, arcs de cercles, elipse, ...
- qui peuvent s'animer selon une chronologie ('*javafx.animation.Timeline*')

JAVA FX : Suivi des Acteurs

Java FX : suivi et affichage automatique des changements

- Lorsqu'un élément du groupe présent sur la scène évolue (changement de position, de couleur, ...), l'événement est capturé par la scène et la fenêtre est mise à jour
- donc *gain important sur la rapidité d'affichage* : seuls les éléments modifiés sont ré-affichés ! ! !

JAVA FX : Création d'une application

Java FX : Création d'une fenêtre application

```
public class FenetreApplication extends Application {
  public void start(Stage primaryStage) {
    //definir la troupe d'acteurs
    Group root = new Group();
    //definir la scene (largeur, hauteur, fond)
    Scene scene = new Scene(root, 800, 600, Color.BLACK);
    primaryStage.setTitle("Un titre de fenetre...");
    primaryStage.setScene(scene);
    //definir un rectangle a peu pres centre
    Rectangle rectangle = new Rectangle(400, 300, 40, 30);
    rectangle.setFill(Color.CHARTREUSE);
    root.getChildren().add(cercle);
    //afficher le theatre
    primaryStage.show();
  }
  public static void main(String[] args) {
        launch(args);
  }
}
```

JAVA FX : Animer la scène

Java FX : Animations diverses

- JavaFX possède son moteur permettant **d'animer** la scène selon une chronologie (**timeline**)
- possibilité d'animer : le déplacement, le changement de couleur, d'opacité, ...
- une transition
  - porte sur un élément graphique, possède une durée,
  - change une variable d'une valeur d'origine vers une valeur de destination,
  - peut être répétée, se dérouler en sens inverse, ...

JAVA FX Transition : Disparition

Transition : Transparence

- Ajouter un cycle 'infini' disparition-apparition du rectangle après sa définition :

```
...
rectangle.setFill(Color.CHARTREUSE);
//transition d'opacite sur 2 secondes
FadeTransition ft = new FadeTransition(Duration.millis
    (2000), rectangle);
//partir de 100% opaque vers 1% d'opacite
ft.setFromValue(1.0); ft.setToValue(0.01);
//a la fin de l'animation, retour de 0.01 vers 1.0
ft.setAutoReverse(true);
//repeter l'animation
ft.setCycleCount(Timeline.INDEFINITE);
ft.play();
```

JAVA FX Transition : Déplacement

Transition : Déplacement

- Possibilité d'ajout de déplacements rectilignes, ou selon un arc, de sauts
- Ces déplacement sont posés dans un chemin (PathTransition)

```
...
Path path = new Path();
path.getElements().add(new MoveTo(80, 60));
path.getElements().add(new LineTo(720, 240);
path.getElements().add(new LineTo(80, 540));
PathTransition pathTransition = new PathTransition(
    Duration.millis(4000), path, rectangle);
pathTransition.setCycleCount(Timeline.INDEFINITE);
pathTransition.setAutoReverse(true);
pathTransition.play();
```

JAVA FX : Translation, Rotation, Redimensionnement

Transition : Translation, Rotation, Redimensionnement

- **TranslateTransition** : permet d'effectuer des transition sur les coordonnées X, Y ou Z
- **RotateTransition** : permet une animation correspondant à une rotation
- **ScaleTransition** : permet une animation modifiant la taille des objets

```
...
RotateTransition rt = new RotateTransition(Duration.millis
    (3000), rectangle);
rt.setByAngle(180);
rt.setCycleCount(4);
rt.setAutoReverse(true);
rt.play();
```

JAVA FX : Ensemble de Transitions

Transitions en parallèle

- lancer une suite d'animation permet de les effectuer en les superposant
- pour les lancer en même temps et en parallèle, il faut les placer dans une transition parallèle (**ParallelTransition**)

```
...
// en reprenant les transition definies precedemment, sans
    les lancer
ParallelTransition parallelTransition = new
    ParallelTransition();
parallelTransition.getChildren().addAll( ft, rt,
    pathTransition);
parallelTransition.setCycleCount(Timeline.INDEFINITE);
parallelTransition.play();
```

Java FX Transition : Séquence de Transitions

Séquence de Transitions

- il est possible de lancer une **série** d'animations (**SequentialTransition**)
- ces animations doivent avoir une durée limitée (pas de répétition à l' "infini")

```
...
// en reprenant les transition definies precedemment, sans les lancer
SequentialTransition sequentialTransition = new SequentialTransition();
sequentialTransition.getChildren().addAll( ft, rt, pathTransition);
sequentialTransition.setCycleCount(2);
sequentialTransition.setAutoReverse(true);
```

JAVA FX : Liste des transitions

Les transitions

- **FadeTransition** : joue sur l'opacité.
- **FillTransition** : joue sur la couleur de remplissage
- **StrokeTransition** : joue sur la couleur de contour
- **PathTransition** : joue sur le déplacement selon un chemin
- **TranslateTransition** : joue sur le déplacement par transition
- **RotateTransition** : joue sur la rotation
- **ScaleTransition** : joue sur la dimension
- **ParallelTransition** : lance des transitions en parallèle
- **SequentialTransition** : lance des transitions en séquence

Java FX Transition : attention aux valeurs

Java FX Animation : fin de transitions, notation lourde

Attention

- Une transition ne change pas la valeur des propriété de l'objet.
- Son image est modifiée, pas ses attributs (position, ...)
- Il faut confirmer la modification des paramètres en fin de transition

Java FX : effectuer une action en fin de transitions (1/2)

Java FX Animation : fin de transitions, notation lourde

essai

- `void setOnFinished(EventHandler < ActionEvent >)` : permet d'ajouter un gestionnaire d'événements qui sera déclenché en fin de transition.
- Classiquement on peut écrire :

```
RotateTransition rt = new RotateTransition(Duration.millis
    (3000), rectangle);
...
rt.setOnFinished(new EventHandler<ActionEvent>() {
  @Override
  public void handle(ActionEvent actionEvent) {
    System.out.println("Fin de rotation !");
  }
});
```

Java FX : effectuer une action en fin de transitions (2/2)

Java FX Animation : fin de transitions, notation allégée

- Par la notation fonctionnelle, le code est simplifié ! !

```
RotateTransition rt = new RotateTransition(Duration.millis
    (3000), rectangle);
...
rt.setOnFinished(actionEvent -> {
    System.out.println("Fin de rotation !");
  });
```

Java FX : Chronologie/TimeLine

Java FX Animation : Animations temporisée

- une **Timeline** permet des animations par modification de données définies en tant que propriétés réinscritptibles (**WritableValue**)
- une **Timeline** est défini par des fenêtre de temps (**KeyFrame**) qui s'exécutent séquentiellement selon leurs durée (KeyFrame.time)
- Les propriétés modifiées sont des valeurs de type **KeyValue**

```
Timeline chronologie = new Timeline();
chronologie.setCycleCount(1);
//la valeur cle a modifier est la position x du rectangle
KeyValue kv = new KeyValue(rectangle.xProperty(), 300);
//duree de l'animation :  1.5 s
KeyFrame kf = new KeyFrame(Duration.millis(1500), kv);
chronologie.getKeyFrames().add(kf);
chronologie.play();
```

JAVA FX : Interpolation

Java FX Animation : Interpoler les points

- l'Interpolation permet de définir les étapes intermédiaires entre la source et la destination
- Par défaut, l'interpolation est linéaire
- Il est possible d'utiliser des interpolateur existant ou d'en créer
- **Interpolator.EASE_BOTH** permet de démarrer lentement une transition, de l'accélérer et de la terminer lentement. Il existe également EASE_IN, EASE_OUT, DISCRETE, ...

```
Timeline chronologie = new Timeline();
chronologie.setCycleCount(1);
//la valeur cle a modifier est la position x du rectangle
KeyValue kv = new KeyValue(rectangle.xProperty(), 0,
    Interpolator.EASE_BOTH);
//duree de l'animation :  1.5 s
KeyFrame kf = new KeyFrame(Duration.millis(1500), kv);
chronologie.getKeyFrames().add(kf);
chronologie.play();
```

JAVA FX : Créer un cycle

Java FX Animation : création de cycles

- Une animation peut répondre à des événements,
- après une durée, un événement est déclenché, il peut être rattrapé
- si l'animation est 'infinie', l'animation sera cyclique

```
Timeline chronologie = new Timeline(new KeyFrame(Duration.
    millis(tempo),
  new EventHandler<ActionEvent>() {
    public void handle(ActionEvent event) {
      //a chaque top, faire quelquechose
      lancerActions();
    }
  }));
littleCycle.setCycleCount(Timeline.INDEFINITE);
littleCycle.play();
```

JAVA FX : Cycles au plus rapide

Java FX Animation : utilisation d'un Timer

- la classe **AnimationTimer** permet de créer un timer dont le cycle est le plus bref pris pouvant être pris en charge par l'application (de l'ordre de 60 images par seconde),
- il est nécessaire de surcharger la fonction **handle(long now)**

```
AnimationTimer monTimer = new AnimationTimer() {
  @Override
  public void handle(long now) {
    rectangle.setTranslateX(rectangle.getBounds().getMinX()
        + 10);
  }
};
monTimer.start();
```

Java FX : Capture d'événements

Java FX & événements

- chaque shape peut réagir au événements souris ou clavier ou tactile
- pour réagir à un clic souris, il suffit d'associer un objet de type *EventHandler* < *MouseEvent* > à l'objet
- il est nécessaire de surcharger la fonction **handle(MouseEvent event)**

```
rectangle.setOnMouseClicked(new EventHandler<MouseEvent>() {
@Override
public void handle(MouseEvent event) {
  System.out.println("mouse␣click␣detected!␣"+event.
      getSource());
  System.out.println("click␣on␣␣" + event.getX() + "," +
      event.getY());
}
});
```

Java FX : Capture d'événements et lambda

Java FX & événements

- En java 8, il est possible d'utiliser une notation issue de la programmation fonctionnelle afin d'alléger le code :

```
rectangle.setOnMouseClicked(event-> {
  System.out.println("mouse␣click␣detected!␣"+event.
      getSource());
  System.out.println("click␣on␣␣" + event.getX() + "," +
      event.getY());
});
```