S. Var R2.02

TD 4 – JAVAFX : UN PEU D'ANIMATION

Durée : 2h00.

Matériel nécessaire : aucun.

Objectif : Créer des animations en JavaFX.

Pré-requis : Eclipse, Java 8 et JavaFX disponibles sur votre poste (optionnel la javadoc).

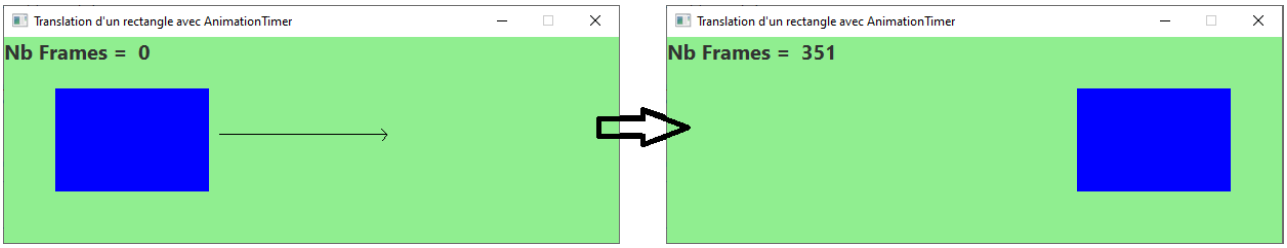
PREMIERE PARTIE : **Prise en main de l'API Animation avec un rectangle.**

Dans cette partie, on va animer un rectangle en utilisant la classe **AnimationTimer**, puis les **Transitions.**

Pour cela, créer une classe nommée **AnimateRectangleTimerApp**.java (package unilim.info.ihm.td4.exo1) héritant de javafx.application.Application.

On va utiliser un Layout Pane racine de type **Group** permettant ainsi de positionner les éléments via leurs coordonnées (X, Y). La taille de la fenêtre sera de 600px par 200px, et elle sera non redimensionnable.

Créer un rectangle bleu ayant pour coordonnées (50, 50) et pour dimensions 150 px par 100 px.



Ensuite, instancier une classe de type AnimationTimer (cf. cours) et redéfinir sa méthode *handle* (qui sera déclenchée à chaque frame, à raison d'environ 60 frames/seconde) : `public void handle(long now)`

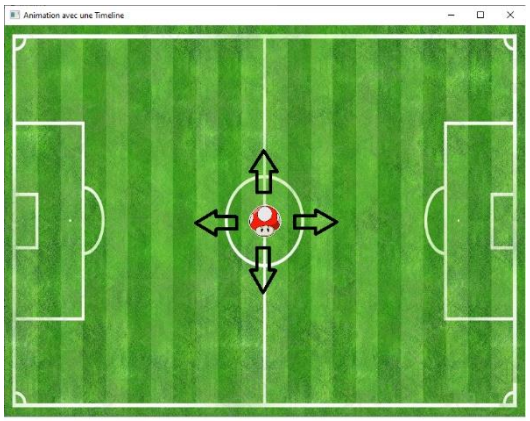
- Dans cette méthode, on va incrémenter de 1 l'abscisse du rectangle (propriété *`rectangle.xProperty()`*).
- Démarrer le timer (après que la fenêtre ait été affichée) puis observer l'animation obtenue.
- Modifier le code pour que la translation s'arrête lorsque x atteint la valeur 400 (arrêter alors le timer).
- Comptabiliser le nombre de frames et afficher celui-ci en haut de votre interface via un **Label**.
- A présent, on va réaliser une petite variante : au lieu de déplacer le rectangle, on va faire disparaître en jouant avec son opacité (valeurs allant de 1 à 0, par pas de 0.01).

Créer une nouvelle classe **AnimateRectangleTransitionApp**.java (même package). Dans celle-ci, reproduire le même rectangle et réaliser les mêmes animations (translation et fondu) à l'aide, cette fois-ci :

- de la classe **TranslateTransition** (translation horizontale vers la droite sur une durée de 4 secondes).
- de la classe **FadeTransition** (disparition progressive sur une durée de 2 secondes).
- observer les impacts sur l'animation en faisant varier la durée, tester aussi avec les options *`cycleCount`*, *`autoReverse`* et *`interpolator`* (avec ou sans l'option, avec différentes valeurs).

Implémenter un écouteur d'évènement (sous la forme d'une classe imbriquée) permettant d'afficher une boite de dialogue indiquant la fin de l'animation. Rattacher cet écouteur à l'animation via la « convenient méthode » *`setOnFinished`*`(EventHandler<ActionEvent>)`.

DEUXIEME PARTIE : **Animer un personnage avec une Timeline.**



Créer une nouvelle classe nommée **AnimatePixelTimelineApp**.java dans le package unilim.info.ihm.td4.exo2.

On va utiliser un Layout Pane racine de type **StackPane**, celui-ci permet d'empiler les composants, les uns au-dessus des autres.

La taille de la fenêtre sera de 800px par 600px, non redimensionnable.

- Charger une image de fond en utilisant la classe **BackgroundImage** ; ajuster sa taille à celle de la fenêtre.
- Charger l'image de votre personnage (taille environ 60x60) au centre de votre interface.
- Réaliser une animation horizontale, à l'aide des classes **KeyValue**, **KeyFrame** et **Timeline**, se basant sur la propriété JavaFX `translateXProperty` (par exemple, décaler l'image vers la droite de 300px).
  Faites varier la durée, tester les options `cycleCount` et `autoReverse` et `interpolator`.

TROISIEME PARTIE : **Déplacer le personnage au clavier.**

A présent, on va gérer **le déplacement de notre personnage au clavier**.

Créer une classe nommée MovePixelController (même package, pour rester simple) qui implémente l'interface `EventHandler<KeyEvent>` (écouteur d'évènements liés au clavier).

Cet classe « écouteur » doit déclencher une animation quand on presse une touche du clavier (`keyPressed`) :

- Si c'est une touche directionnelle, il faut déplacer votre personnage horizontalement ou verticalement (pas d'action si c'est une autre touche). La Javadoc de la classe KeyEvent vous aidera.

  https://docs.oracle.com/javase/8/docs/api/java/awt/event/KeyEvent.html

- Utiliser une animation **Timeline** qui définira une translation de 30px (horizontale ou verticale), en avant ou en arrière selon la touche pressée. Sa durée sera de 250ms.

  https://docs.oracle.com/javase/8/javafx/api/javafx/animation/Timeline.html

  https://docs.oracle.com/javase/8/javafx/api/javafx/animation/KeyFrame.html

  https://docs.oracle.com/javase/8/javafx/api/javafx/animation/KeyValue.html

Bonus si vous avez le temps :

Quand on presse sur la touche « Espace », il faudra à présent gérer un saut en arc de cercle à partir d'un **PathTransition**, le **Path** sera composé des éléments **MoveTo** et **ArcTo**.

Bien prendre soin de lire la documentation associée à ces classes :

https://docs.oracle.com/javase/8/javafx/api/javafx/animation/PathTransition.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/ArcTo.html