Interaction Humain-Machine

TP 2 : Gestion des événements

Objectifs

— dessiner dans un Canvas et gérer les événements de la souris

Pré-requis

— Maîtriser le cours 2 [1] et le cours 3 [2].

1 Retour sur le dernier TP

Consultez la correction du dernier TP disponible sur Moodle et posez des questions à votre enseignant, si vous en avez.

2 Préambule : rappels du cours

Il est possible de dessiner avec JavaFX en utilisant la classe Canvas [3]. Le code ci-dessous donne un exemple de dessin d'un carré :

```
public class HelloPaint extends Application {

        public void start(Stage stage) {
                VBox root = new VBox();
                Canvas canvas = new Canvas (300, 300);
                GraphicsContext gc = canvas.getGraphicsContext2D();
                gc.setFill(Color.ORANGE);
                gc.fillRect(40, 100, 20, 20);
                gc.setStroke(Color.BLACK);
                gc.strokeRect(40, 100, 20, 20);
                root.getChildren().add(canvas);

                Scene scene = new Scene(root);
                stage.setTitle("Hello Paint");
                stage.setScene(scene);
                stage.show();
        }

        public static void main(String[] args) {
                Application.launch(args);
        }
}
```

Les différents événements souris sont détaillés figure 1.

3 Dessiner et interagir avec JavaFX

Le but de cet exercice est de dessiner des carrés en cliquant sur un bouton de la souris pour faire apparaître un carré à la position du pointeur de la souris (centre du carré = position du pointeur de la souris) puis de les faire disparaître en cliquant dessus tout en appuyant sur la touche SHIFT du clavier et enfin les déplacer en cliquant dessus et en les faisant glisser tout en maintenant un bouton de la souris enfoncé.

1. https://www.iut-info.univ-lille.fr/~gery.casiez/R2.02/Cours/R2.02-Cours2.pdf
2. https://www.iut-info.univ-lille.fr/~gery.casiez/R2.02/Cours/R2.02-Cours3.pdf
3. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/Canvas.html

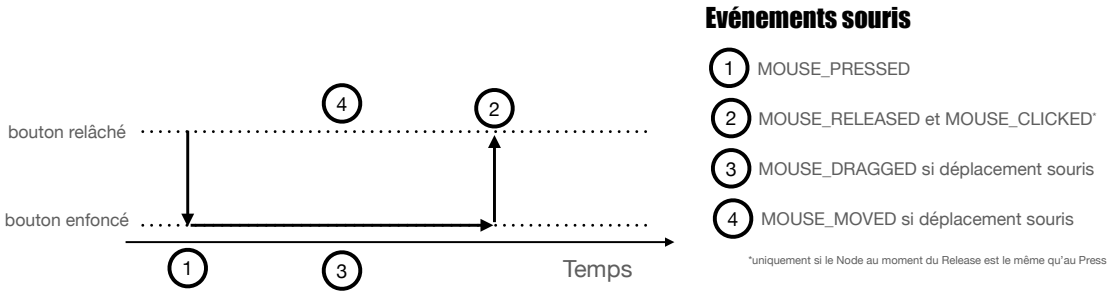

FIGURE 1 – Illustration des différents événements produits au cours du temps, suivant l'état du bouton et des déplacements de la souris.

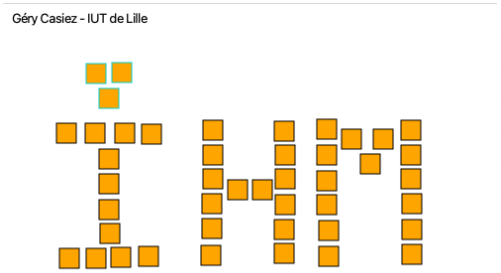

FIGURE 2 – Capture d'écran de l'interface à réaliser.

3.1 Dessiner et effacer des carrés

Q 1. Créez une interface similaire à la figure 2. Ajoutez un EventHandler pour gérer l'état "clicked" des MouseEvent. Vous mettrez à profit la classe Rectangle [4] pour enregistrer chacun des carrés à afficher (sous forme de Vector ou ArrayList). La classe `Rectangle` est utilisée pour stocker les différentes informations d'un carré : les coordonnées du coin supérieur gauche et ses dimensions. Le dessin du carré se fera "à la main" en utilisant ces informations. Le nettoyage de l'écran se fait avec la méthode clearRect [5].

Votre code pourra partir de l'exemple minimal suivant qui permet d'avoir une séparation stricte entre le code de gestion des événements et celui de mise à jour de l'affichage :

```
public class MinimalExampleSquare extends Application {
        // Structure de données pour stocker les carrés
        // avec les informations associées
        Vector<MyRectangle> rectangles = new Vector<MyRectangle>();

        // Contexte graphique du canvas, nécessaire pour dessiner
        GraphicsContext gc;
        Canvas canvas;

        class MyRectangle {
                public Rectangle rect;
                // Autres infos
        }


        public void start(Stage stage) {
                VBox root = new VBox();
                canvas = new Canvas (500, 500);
                gc = canvas.getGraphicsContext2D();
```

4. Rectangle.java
5. http://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/GraphicsContext.html#clearRect-double-double-double-double-

```
                canvas.setOnMousePressed(e -> {
                        // ....

                        // A chaque modification de l'état de l'application
                        // on redessine tout
                        repaint();
                });

                root.getChildren().add(canvas);

                Scene scene = new Scene(root);
                stage.setTitle("Hello Paint");
                stage.setScene(scene);
                stage.show();
        }

        // Méthode qui sert à redessiner tout l'écran
        // tout le code lié au dessin dans le canvas doit
        // se trouver dans cette méthode et pas ailleurs.
        // repaint() est appelée ailleurs dans le code quand
        // il est nécessaire de redessiner.
        public void repaint() {
                // On efface tout
                gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());
                // et on redessine tout
                gc.setFill(Color.BLACK);
                for (MyRectangle r: rectangles) {
                        // Dessiner chaque carré
                }
        }

        public static void main(String[] args) {
                Application.launch(args);
        }
}
```

Q 2. En utilisant la méthode isShiftDown [6], gérez maintenant la touche modificatrice (*modifier* en anglais) SHIFT pour effacer un carré quand on clique dessus. Pour cela vous utiliserez à bon escient la méthode `contains` sur vos objets Rectangle pour déterminer si les coordonnées du pointeur de la souris appartiennent à un des carrés affichés.

3.2 Déplacer des carrés

Q 3. Gérez maintenant l'état "dragged" de la souris pour déplacer un carré quand on le fait glisser avec le pointeur de la souris. Attention : veillez à faire correctement les choses pour ne pas perdre le carré quand vous déplacez rapidement le pointeur de la souris !

3.3 Ajout de retour d'information

Q 4. Réalisez maintenant un retour d'information pour suggérer à l'utilisateur qu'il est possible d'interagir avec les carrés. Pour cela, augmentez le contraste de la couleur de remplissage des carrés quand la souris les survole ("moved").

3.4 Sélection multiple, alignement

Q 5. Gérez maintenant la sélection multiple de carrés en utilisant la touche CTRL du clavier tout en cliquant sur les carrés à sélectionner. Vous prendrez le soin de fournir du retour d'information sur les carrés sélectionnés en modifiant par exemple la couleur du bord des carrés sélectionnés.

Q 6. (Pour les plus rapides) Gérez le déplacement simultané des carrés sélectionnés.

6. https://docs.oracle.com/javafx/2/api/javafx/scene/input/MouseEvent.html#isShiftDown()

4 BONUS : Créez votre potentiomètre

L'objectif est de créer votre propre potentiomètre en dessinant un rectangle pour la piste et un autre pour la poignée, de manière à obtenir un résultat proche de ce qui est représenté figure 3. L'objectif est de gérer les événements souris pour que le comportement soit le plus proche possible d'un potentiomètre de toolkit.

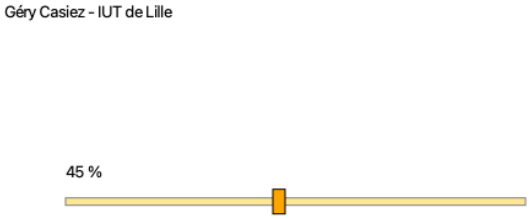

FIGURE 3 – Capture d'écran de l'interface à réaliser.