Programmieren 1 – Schmiedecke

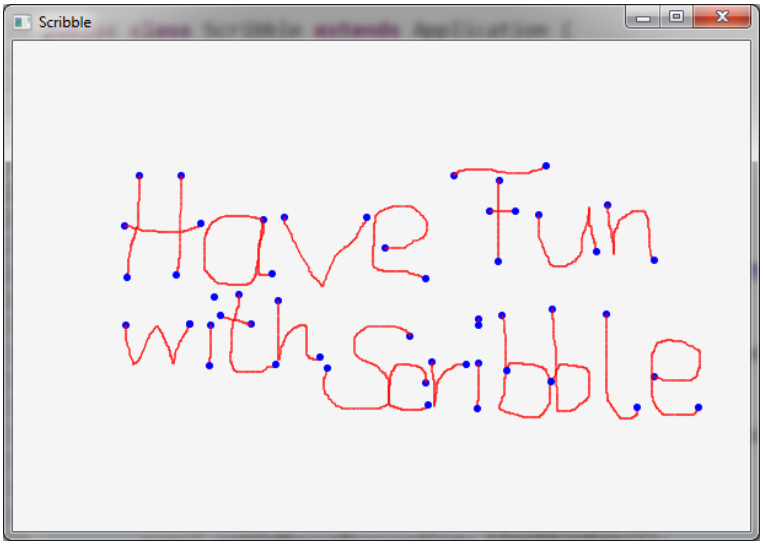
Laborexperiment 7 – Scribble

Zum Schluss noch eine kleine Aufgabe in JavaFX:

Wir haben in der Vorlesung ein primitives Scribble realisiert Machen Sie etwas draus! Und beantworten Sie bitte die Vorbereitungsfragen in Ihrem Protokoll!

Den Anfangscode finden Sie zum Abtippen am Ende des Aufgabenblatts.

Diese Aufgabbe ist klein und in einer Woche zu schaffen. Aber Sie müssen bei der Abnahme auch zu dem vorgegebenen Anfangscode Rede und Antort stehen.



Lernziele: GUIs mit JavaFX, Events und Eventhandler, Scope

Vorbereitungsfragen:

1. Zeichnen Sie den Scene Graph für das Scribble-Programm bei Programmstart, also ohne Zeichnung.
2. Aus wievielen Objekten besteht das oben dargestellte Scribble-Bild?
3. Wodurch wird ein Linienzug unterbrochen und wodurch ein neuer begonnen?
4. Woher weiß der EventHandler DragPainter, dass er auf ein MouseDragged-Event reagiert? Könnte man ihn auch für ein anderes Ereignis verwenden, z.B. ein ActionEvent?
5. Wenn man Linien kreuzt, verdeckt dann die obere die untere? Welches ist die obere?
6. Was passiert, wenn das Fenster größer oder kleiner gezogen wird? Warum?
7. Was passiert, wenn man über den Rand malt? Warum?

Laborexperimente:

Scribble ist ganz nett, aber mehr nicht. Znächst sollten sie die blauen Start- und Endpunkte entfernen. Wählen Sie dann **zwei der folgenden Feature-Vorschläge** und peppen Sie Ihr Scribble damit auf. **Legen Sie für die Effekte Knöpfe am linken Rand an**. Nutzen Sie Panes (BorderPane, Vbox, Hbox, etc...), um dafür ein schönes Layout zu gestalten.

1. **Linien konfigurieren:** Schaffen Sie die Möglichkeit, die Farbe und Linienstärke für jeden einzelnen Linienzug neu zu bestimmen.

2. **Stricharten**:Ermöglichen Sie eine Auswahl an Stricharten, z.B. gestrichelt, doppel, schattiert, aus Quadraten oder Kreisen zusammengesetzt...

3. **Bild löschen:** Erstellen Sie einen Löschknopf, der die gesamte Zeichnung löscht.

4. **Undo:** Realisieren Sie einen Rückgängig-Knopf, mit dem Sie die Linienzüge schrittweise rückbauen können.

5. **Hintergrund**:Ermöglichen Sie verschiedene Hintergrundfarben und –effekte

6. **Hintergrundbild**: Schaffen Sie die Möglichkeit, ein Hintergrundbild darzustellen (feste oder freie Auswahl) und darauf zu zeichnen..

Für Fortgeschrittene :

7. **Linienzug-Radierer:** Speichern Sie alle Linien eines Lineinzugs in einer Collection und ermöglichen Sie das Entfernen eines ganzen Linienzugs in einem Schritt. (Alternativ können Sie für jeden Linienzug ein neues tranparentes Rectangle anlegen.) Sie können die Zeichenreihenfolge rückwärts nuntzen oder den Linienzug auswählbar oder anklickbar machen.

8. **Linienzug-Bearbeitung:** Ermöglichen Sie Farb- und Strichstärkeänderungen für ganze Linienzüge (Auswal s. 7.)

9. **Linienzug-Verschiebung:** Ermöglichen Sie das Verschieben ganzer Linienzüge per Drag.

Bitte erläutern Sie im Protokoll jeweils Ihre Strategie! Erläutern Sie ggf. auch, was Sie nicht bedacht hatte und nachträglich anpassen mussten, oder was gar nicht geklappt hat, und warum.

Achtung: Alle Importe außer java.util.List aus javafx-Paketen!

```
public class Scribble extends Application {

    @Override
    public void start(Stage primaryStage) {
        root = new Group();
        Scene scene = new Scene(root, 600,400);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Scribble");

        Rectangle panel = new Rectangle(600,400,Color.WHITESMOKE);
        root.getChildren().add(panel);

        panel.setOnMousePressed(new LineStarter());
        panel.setOnMouseDragged(new DragPainter());
        panel.setOnMouseReleased(new LineEnder());

        primaryStage.show();
    }

    private Group root;
    private double x, y, lastX, lastY;

    private class LineStarter implements EventHandler<MouseEvent> {
        public void handle(MouseEvent event) {
            x=event.getX(); y=event.getY();
            root.getChildren().add(new Circle(x,y,3,Color.BLUE));
        }
    }
    private class DragPainter implements EventHandler<MouseEvent> {
        public void handle(MouseEvent event) {
            lastX=x; lastY=y;
            x=event.getX();  y=event.getY();
            Line line = new Line(x,y,lastX,lastY);
            line.setStroke(Color.RED);
            root.getChildren().add(line);
        }
    }
    private class LineEnder implements EventHandler<MouseEvent> {
        public void handle(MouseEvent event) {
            root.getChildren().add(new Circle(lastX, lastY,3,Color.BLUE));
        }
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```