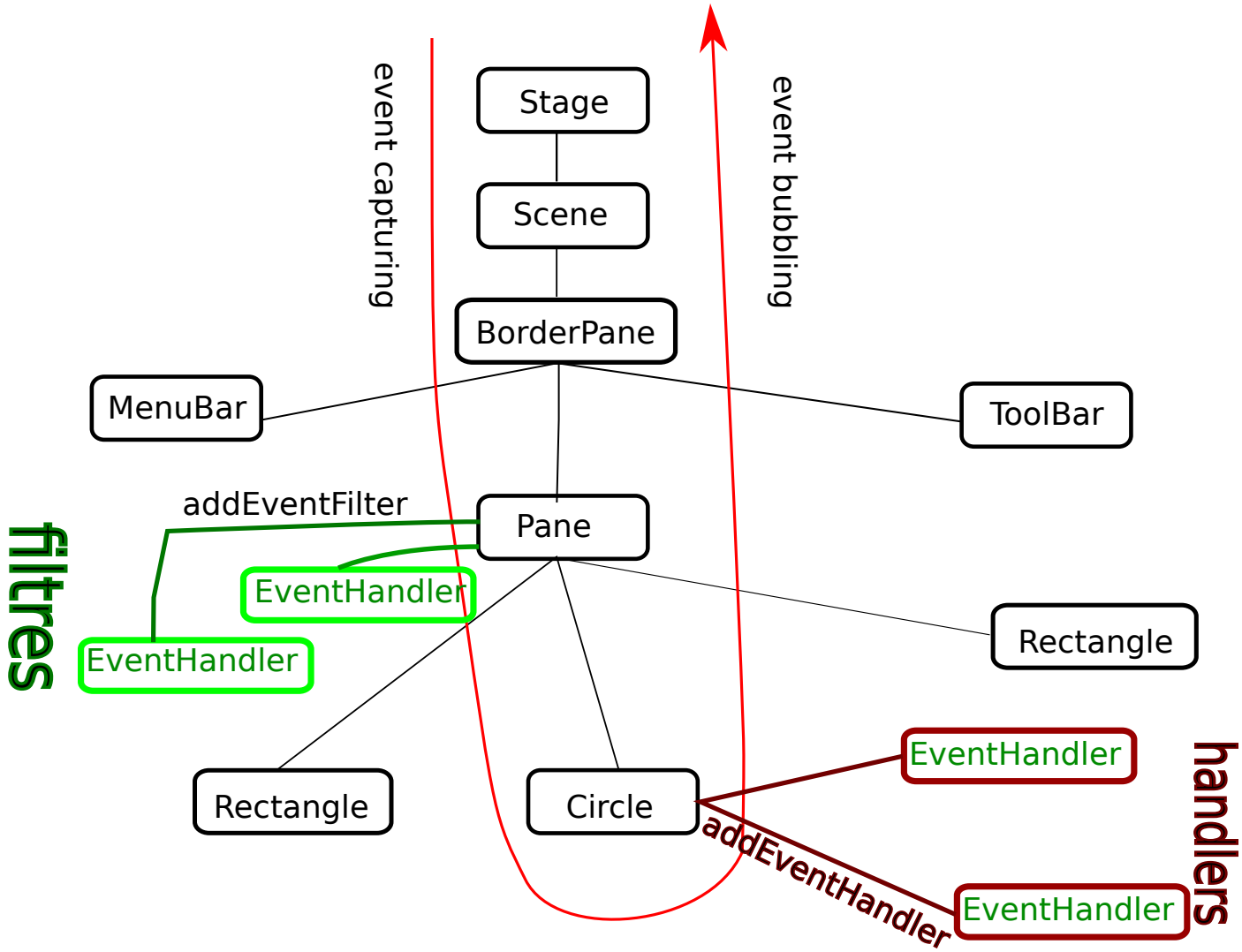
Interfaces graphiques - JavaFX

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

March 2, 2016

Filtres et handlers



Construire, enregistrer et désenregistrer `EventFilters`

```
EventHandler<MouseEvent> filtre =
    new EventHandler<MouseEvent>(){
        public void handle(MouseEvent event){
            //implementer le filtre
            //event.consume();
        }
};
//enregistrer
node.addEventFilter(MouseEvent.MOUSE_CLICKED, filtre);

//desenregistrer
node.remove(MouseEvent.MOUSE_CLICKED, filtre);
```

Construire, enregistrer/désenregistrer EventHandlers

```
EventHandler<KeyEvent> handler =
    new EventHandler<KeyEvent>(){
        public void handle(KeyEvent event){
            //implementer le filtre
            //event.consume();
        }
};

//enregistrer
node.addEventHadler(KeyEvent.KEY_TAPED, handler);

//desenregistrer
node.removeEventHadler(KeyEvent.KEY_TAPED, handler);
```

Méthodes de commodité pour enregistrer un (seul) EventHadler

Au lieu d'enregistrer un handler avec

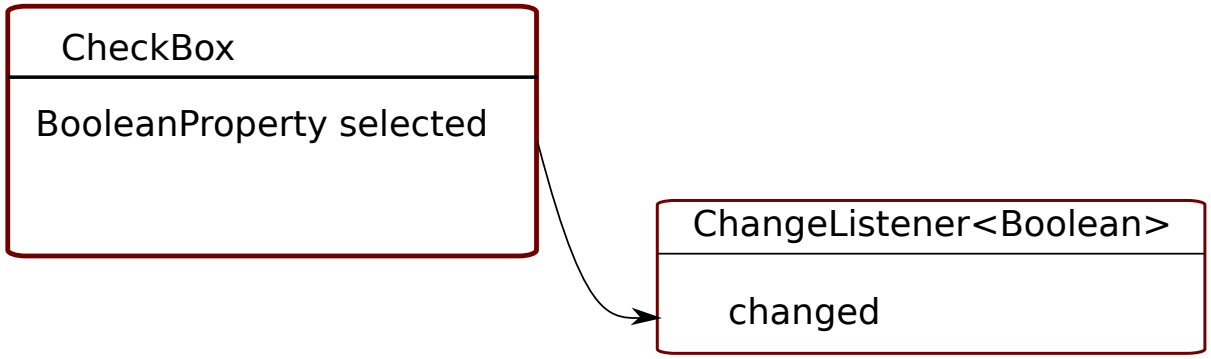
```
node.addEventHandler(MouseEvent.MOUSE_PRESSED,
        new EventHandler<MousePressed>(){

          public void handle(MouseEvent event){ ... }
});
```

on peut utiliser une méthode de commodité :

```
node.setOnMousePressed( new EventHandler<MousePressed>(){
          public void handle(MouseEvent event){ ... }
});
```

Les méthodes de commodité ont le nom `setOn`*TypeEvenement*.

Propriétés et ChangeListeners




```
checkBox.selectedProperty().addListener(
  new ChangeListener<Boolean>(){
      @Override
      public void changed(
          ObservableValue<? extends Boolean> observable,
          Boolean old_val,
          Boolean new_val) {
      .......
      }
});
```

La liste d'enfant d'un noeud

Si on a un noeud (par exemple `Pane`) de type Parent on lui ajoute un enfant avec add appliqué à la liste d'enfants :

```
pane.getChildren().add( enfant );
```

`ObservableList<Node> getChildren()` retourne la liste d'enfants d'un noeud. La méthode `add()` ajoute `enfant` à la fin de la liste.

Quand le parent est rendu sur l'écran les enfants sont rendus dans l'ordre de leur apparition sur la liste des enfants, le premier celui en tête de la liste jusqu'au dernier, celui à la fin de la liste.

La liste d'enfants

La liste obtenue par l'appel `getChidren()` est une vraie liste, toutes les opérations habituelles sur les listes sont disponibles.
Par exemple : supprimer le dernier élément de la liste d'enfants :

```
List l = pane.getChildren();
l.remove(l.size() - 1);
```

L'ordre d'affichage d'enfants

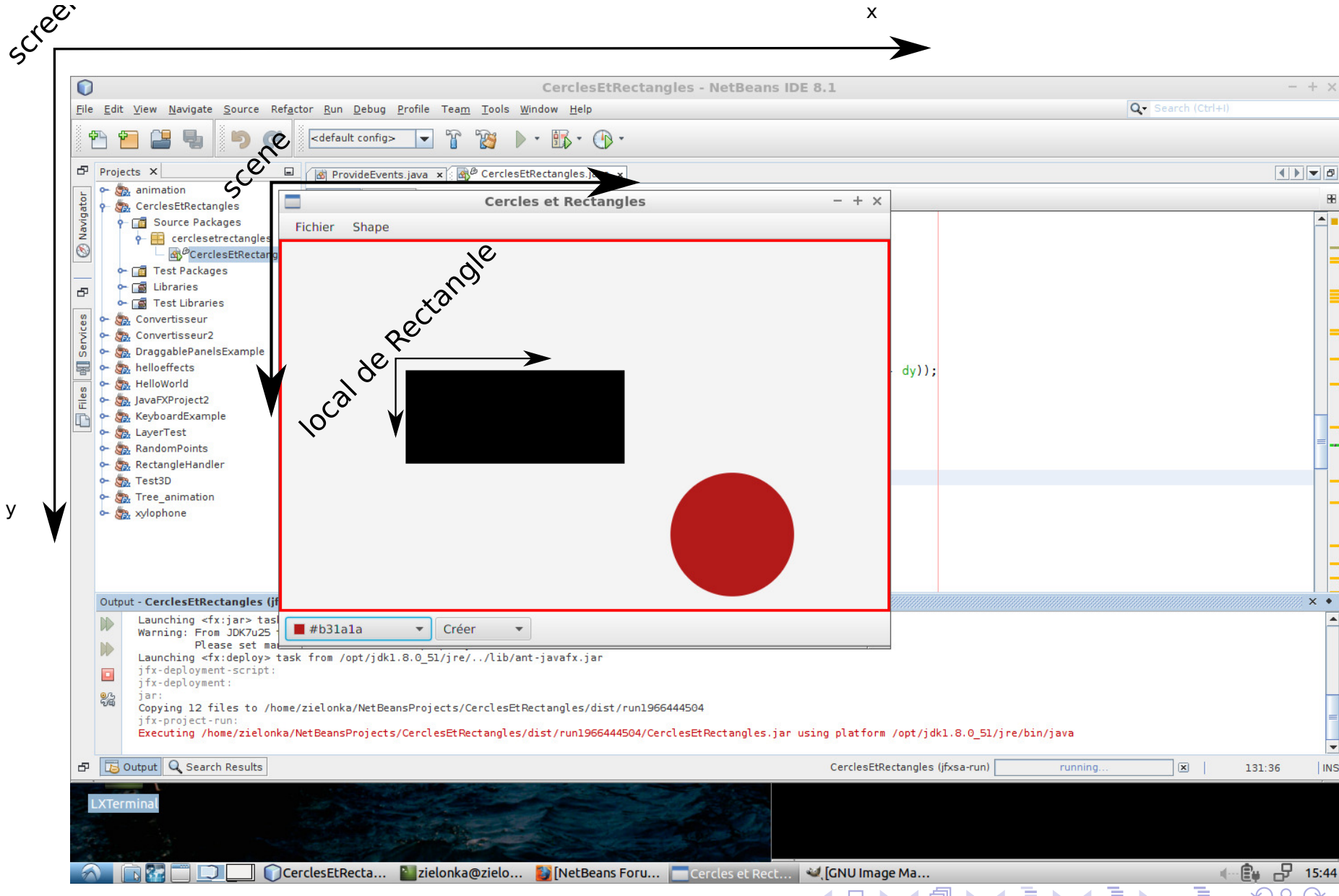
Quand le parent est rendu sur l'écran les enfants sont rendus dans l'ordre de leur apparition sur la liste des enfants, le premier celui en tête de la liste jusqu'au dernier, celui à la fin de la liste.

Systèmes de coordonnées en JavaFX

Trois types de coordonnées :

- locales par rapport au noeud,
- scene - coordonnées par rapport à la scène,
- screen – coordonnées par rapport à l'écran.

Systèmes de coordoonnées



Les systèmes de coordonnées

`MouseEvent` possède les méthodes

- `double getX()`, `double getY()`,
- `double getSceneX()`, `double getSceneY()`,
- `double getScreenX()`, `double getScreenY()`,

qui donnent les coordonnées de l'évènement dans

- le système de coordonnées locales d'un noeud source de l'événement,
- par rapport au système de coordonnées de la scène,
- par rapport au système de coordonnées de l'écran.

La classe `Node` possède les méthodes

```
localToParent()
localToScene()
localToScreen()
```

qui permettent d'obtenir les coordonnées dans le système parent, scène, l'écran à partir de coordonnées locaux d'un point.

Fabriquer un noeud "draggable"

```
private Node makeDraggable(Node node) {
  class T {
     double initialTranslateX, initialTranslateY,
            anchorX, anchorY;
  }

  final T t = new T();

  node.setOnMousePressed(new EventHandler<MouseEvent>() {
    @Override
    public void handle(final MouseEvent event) {
      t.initialTranslateX = node.getTranslateX();
      t.initialTranslateY = node.getTranslateY();
      Point2D point = node.localToParent(event.getX(),
                                         event.getY());
      t.anchorX = point.getX();
      t.anchorY = point.getY();
    }
  });
```

Fabriquer un noeud "draggable" (suite)

```
node.setOnMouseDragged(new EventHandler<MouseEvent>() {
    @Override
    public void handle(final MouseEvent event) {
        Point2D point = node.localToParent(event.getX(),
                                           event.getY());
        node.setTranslateX(t.initialTranslateX -
                           t.anchorX + point.getX());
        node.setTranslateY(t.initialTranslateY -
                           t.anchorY + point.getY());
    }
});
return node;
}
```

Propriété de translation

Propriétés de transltion d'un noeud dans les coordonnées locales :

```
DoubleProperty translateX, translateY, translateZ
```

que son parent utilise pour déterminer où dessiner le noeud.

Utiliser ColorPicker

```
Color shapeColor = Color.BLACK;

ColorPicker picker = new ColorPicker(shapeColor);
picker.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent e) {
            Color c = picker.getValue();
            if (c != null) {
                shapeColor = c;
            }
    }
});
```

Faire un bord autour de Pane

```
pane.setBorder(new Border(new BorderStroke(
        Color.RED,
        BorderStrokeStyle.SOLID,
        CornerRadii.EMPTY,
new BorderWidths(3))));
```

Clipping région

Un Rectangle mis dans Pane est dessiné même s'il sort de ce Pane !!!

Solution : définir "clip" région dans Pane en utilisant la propriété :

```
ObjectProperty<Node> clip
```

Par exemple :

```
pane.setPrefSize(400, 400);
Rectangle clip = new Rectangle(400,400);
pane.setClip(clip);
```

Clipping région (suite)

Problème : on veut que la taille de clipping région soit toujours la même que la taille de Pane, c'est-à-dire les propriétés `height` et `width` du Rectangle soit toujours égales aux mêmes propriétés de Pane.

Première solution : utiliser ChangeListener attachés aux propriétés width et height de Pane et modifie width et height de Rectangle.

Mais on peu bien mieux avec bind:

```
clip.heightProperty().bind(pane.heightProperty());
clip.widthProperty().bind(pane.widthProperty());
```

Les changements de propriétés de clip suivent automatiquement les changements de propriétés de pane.

Créer un Text

```
Text t = new Text();
t.setFont(Font.font("Verdana", 20));
t.setFill(shapeColor); //color
t.setX(cx);   t.setY(cy); //position
t.setText(string);
pane.getChildren().add(t);
```

Récupérer KeyEvents

```
//installer un filtre (ou un handler)
pane.addEventFilter(KeyEvent.KEY_TYPED, keyFilter);
pane.setFocusTraversable(true);
```

Le contrôle qui possède un filtre ou handler pour l'évènement `KeyEvent` doit posséder le focus pour recevoir ces évènements. Mais Pane ne peut pas avoir de focus sauf si on le rend "focus traversable" par

```
pane.setFocusTraversable(true);
```

Une fois pane est "focus traversable" on peut demander le focus avec

```
pane.requestFocus()
```

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.err.println("CONTROL_DOWN");
        }
    } };
```

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 cela ne concerne que les évènements `KeyEvent.KEY_PRESSED` et `KeyEvent.KEY_RELEASED`. Pour l'évènement `KeyEvent.KEY_TYPED` cette méthode retourne toujours `KeyCode.UNDEFINED`.

les effets visuels

- Blend Effect
- Bloom Effect
- Blur Effects
- Drop Shadow Effect
- Inner Shadow Effect
- Reflection
- Lighting Effect
- Perspective Effect

Mis en oeuvre

```
Text t = new Text();
t.setText("Blurry Text!");
t.setFill(Color.RED);
t.setFont(Font.font("null", FontWeight.BOLD, 36));
//preparer l'effet
BoxBlur bb = new BoxBlur();
bb.setWidth(5);
bb.setHeight(5);
bb.setIterations(3);
//setEffect
t.setEffect(bb);
```