

Cours R2.02

Introduction à l'Interaction Humain-Machine

Cours 2 : programmation événementielle

Plan du cours en 9 semaines

2

1. Introduction à l'interaction, placement
- 2. Programmation événementielle**
3. Widgets et événements (1/2)
4. Widgets et événements (2/2)
5. Conception et prototypage (1/2)
6. Conception et prototypage (2/2)
7. Heuristiques et recommandations
8. Modèles et théories
9. Méthodes d'évaluation des IHM

Différences entre la programmation procédurale et événementielle

Programmation événementielle

4

Programmation « procédurale »

Le déroulement est contrôlé par une séquence d'instructions écrites

Le programmeur écrit la boucle principale

Programme principal

initialisations

répéter

lire une commande

traiter une commande

jusqu'à la commande finir

Programmation événementielle

5

Le déroulement est contrôlé par la survenue d'événements

Pas de boucle principale

Fonctions (réaction aux événements)

Programme principal

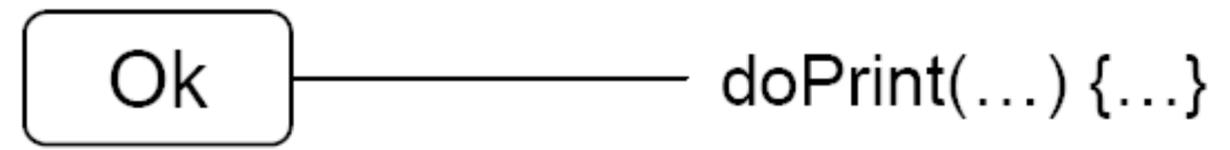
initialisations

Fonctions de rappel

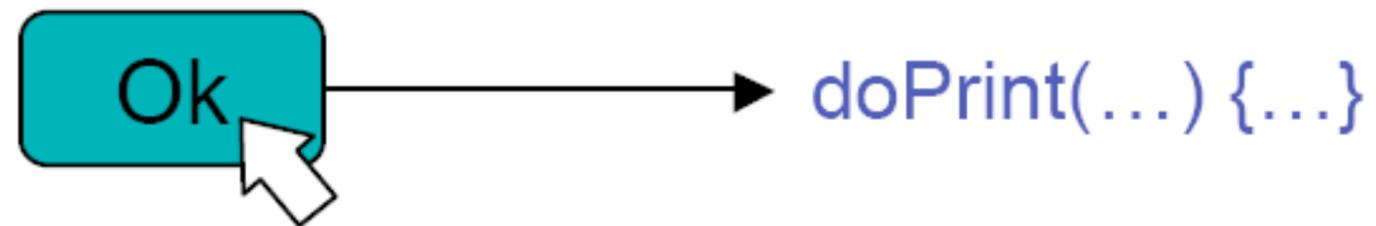
6

Fonctions de rappel (callbacks)

Enregistrées dans le widget à sa création (abonnement)



Appelées lorsqu'un événement sur le widget survient (notification)



Comment les événements sont-ils créés ?

Événements

8

Événements liés aux périphériques

Entrée/sortie du curseur dans une fenêtre

Utilisation d'un des boutons

Frappe au clavier

Gestes multitouch

Sélection d'un item dans une liste déroulante

...

Événements liés aux applications

Création/destruction de fenêtres

Ré-affichage

...

Création des événements dans la toolkit

9

Boucle principale de gestion des événements (enfouie dans la librairie)

Initialisation

```
while () {  
    attendre événement suivant E  
    traiter événement E  
}
```

Création des événements dans la toolkit

10

Exemple: JRE Windows

```
// Run the window message loop.  
while (GetMessage(&msg, NULL, 0, 0)) {  
    DispatchMessage(&msg);  
}  
return 0;
```

```
/*  
 * Dispatch messages for this window class--general component  
 */  
LRESULT AwtComponent::WindowProc(UINT message, WPARAM wParam,  
LPARAM lParam)  
{  
...  
    switch (switchMessage) {  
...  
        case WM_MOUSEMOVE:  
        case WM_MOUSEWHEEL:  
        case WM_AWT_MOUSEENTER:  
        case WM_AWT_MOUSEEXIT:  
        {  
            DWORD curPos = ::GetMessagePos();  
            POINT myPos;  
            myPos.x = GET_X_LPARAM(curPos);  
            myPos.y = GET_Y_LPARAM(curPos);  
            ::ScreenToClient(GetHwnd(), &myPos);  
            switch(switchMessage) {  
                case WM_AWT_MOUSEENTER:  
  
                    mr = WmMouseEnter(static_cast<UINT>(wParam),  
myPos.x, myPos.y); break;  
                case WM_LBUTTONDOWN:  
                case WM_LBUTTONDBLCLK:  
                    mr = WmMouseDown(static_cast<UINT>(wParam),  
myPos.x, myPos.y,  
LEFT_BUTTON); break;  
                case WM_LBUTTONUP:  
                    mr = WmMouseUp(static_cast<UINT>(wParam),  
myPos.x, myPos.y,  
LEFT_BUTTON); break;  
                case WM_MOUSEMOVE:  
                    mr = WmMouseMove(static_cast<UINT>(wParam),  
myPos.x, myPos.y); break;  
                case WM_MBUTTONDOWN:
```

Création des événements dans la toolkit

11

```
void AwtComponent::SendMouseEvent(jint id, jlong when, jint x,
jint y,
                                jint modifiers, jint
clickCount,
                                jboolean popupTrigger, jint
button,
                                MSG *pMsg)
{
    JNIEnv *env = (JNIEnv *)JNU_GetEnv(jvm, JNI_VERSION_1_2);
    CriticalSection::Lock l(GetLock());
    if (GetPeer(env) == NULL) {
        /* event received during termination. */
        return;
    }

    static jclass mouseEventCls;
    if (mouseEventCls == NULL) {
        jclass mouseEventClsLocal =
            env->FindClass("java/awt/event/MouseEvent");
        if (!mouseEventClsLocal) {
            /* exception already thrown */
            return;
        }
        mouseEventCls = (jclass)env-
>NewGlobalRef(mouseEventClsLocal);
        env->DeleteLocalRef(mouseEventClsLocal);
    }
    RECT insets;
    GetInsets(&insets);

    static jmethodID mouseEventConst;
    if (mouseEventConst == NULL) {
        mouseEventConst =
            env->GetMethodID(mouseEventCls, "<init>",
                "(Ljava/awt/Component;IJJIIIZI)V");
        DASSERT(mouseEventConst);
    }
    if (env->EnsureLocalCapacity(2) < 0) {
        return;
    }
    jobject target = GetTarget(env);
    jobject mouseEvent = env->NewObject(mouseEventCls,
mouseEventConst,
                                target,
                                id, when, modifiers,
                                x+insets.left, y+insets.top,
                                clickCount, popupTrigger, button);
}
```

Création des événements dans la toolkit

12

Les événements sont placés dans une queue (FIFO)

La boucle de gestion des événements prend les événements dans la queue et les traite

Mouse Move (10,15)

Mouse Down Left (3,40)

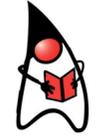
Mouse Up Left (10, 50)

```
JRE ->      /* Post event to the system EventQueue. */
              JNU_CallMethodByName(env, NULL, GetPeer(env), "postEvent",
              "(Ljava/awt/AWTEvent;)V", event);

              EventQueue.java
JDK ->      public void postEvent(AWTEvent theEvent) {
              SunToolkit.flushPendingEvents();
              postEventPrivate(theEvent);
              }
```

Propriétés des événements

Les événements avec JavaFX



Event
EventTarget
EventType

14

Un événement est un objet qui hérite de la classe **javafx.event.Event**

Tout événement a 3 propriétés :

une source : event source (propriété de Event)

une cible : event target (interface EventTarget)

un type : event type (classe EventType)

Classes d'événements

15

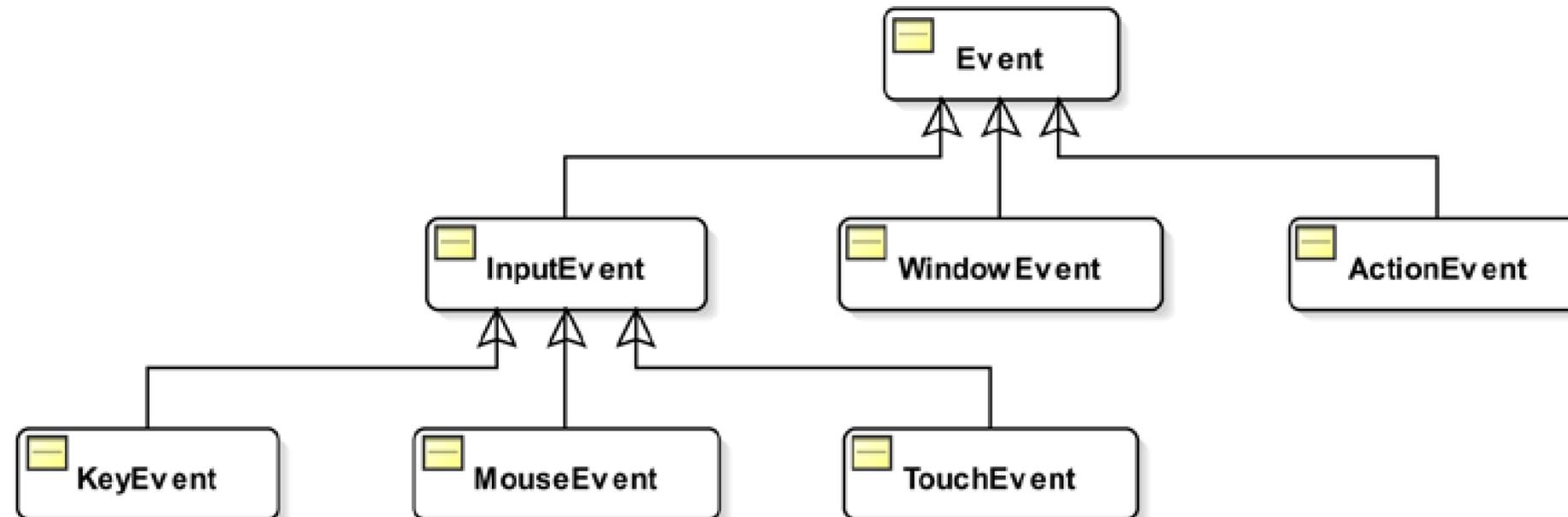
L'événement généré dépend de la nature physique de l'interaction et surtout de sa signification logique : différentes classes d'événements pour une même action physique

Interaction utilisateur	Événement émis
Clic sur un bouton	ActionEvent
Clic sur un rectangle	MouseEvent
Clic sur le bouton de fermeture d'une fenêtre	WindowEvent

Classes d'événements

16

Représentation partielle



javafx.event

Class Event

java.lang.Object
java.util.EventObject
javafx.event.Event

All Implemented Interfaces:

Serializable, Cloneable

Direct Known Subclasses:

ActionEvent, CheckBoxTreeItem.TreeModificationEvent, DialogEvent, InputEvent, ListView.EditEvent, MediaErrorEvent, ScrollToEvent, SortEvent, TableColumn.CellEditEvent, TransformChangedEvent, TreeItem.TreeModificationEvent, TreeTableColumn.CellEditEvent, TreeTableView.EditEvent, TreeView.EditEvent, WebErrorEvent, WebEvent, WindowEvent, WorkerStateEvent

Réagir aux événements avec les eventHandlers

Abonneurs / abonnés

18

Deux catégories d'entités :

Un abonné, la source d'information, le lieu où sont gérés les événements

Un **ou des** abonnés, utilisant l'information produite

Résumé des différentes étapes :

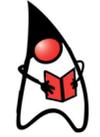
Abonnement de l'abonné auprès de l'abonné

Un événement se produit au niveau de l'abonné

L'abonné notifie l'ensemble des abonnés en leur transmettant un événement

Chaque abonné peut traiter l'événement

Abonnement / notification



EventHandler

19

L'abonnement à un événement dépend du type d'événement à gérer

D'une manière générale, utilisation de la méthode
addEventHandler(EventType<T> eventType, EventHandler<? super T> eventHandler)

eventType correspond au type d'événement géré

EventHandler<? super T> est un objet qui implémente **EventHandler** dont la méthode **handle** qui sera appelée quand l'événement sera déclenché

? super T signifie "la classe T ou toute superclasse de la classe T"

Classes génériques en Java

20

Comment gérer des objets de n'importe quel type ?

1ère solution :

```
public class Box {  
    private Object object;  
  
    public void set(Object object) { this.object = object; }  
    public Object get() { return object; }  
}
```

Classes génériques en Java

21

```
public class Box<T> {  
    // T stands for "Type"  
    private T t;  
  
    public void set(T t) { this.t = t; }  
    public T get() { return t; }  
}
```

```
Box<Integer> integerBox = new Box<Integer>();  
Box<String> integerBox = new Box<String>();
```

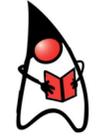
Interface EventHandler

22

```
package javafx.event;

import java.util.EventListener;

// PENDING_DOC_REVIEW
/**
 * Handler for events of a specific class / type.
 *
 * @param <T> the event class this handler can handle
 * @since JavaFX 2.0
 */
@FunctionalInterface
public interface EventHandler<T extends Event> extends EventListener {
    /**
     * Invoked when a specific event of the type for which this handler is
     * registered happens.
     *
     * @param event the event which occurred
     */
    void handle(T event);
}
```



Exemple : clic sur un bouton

23

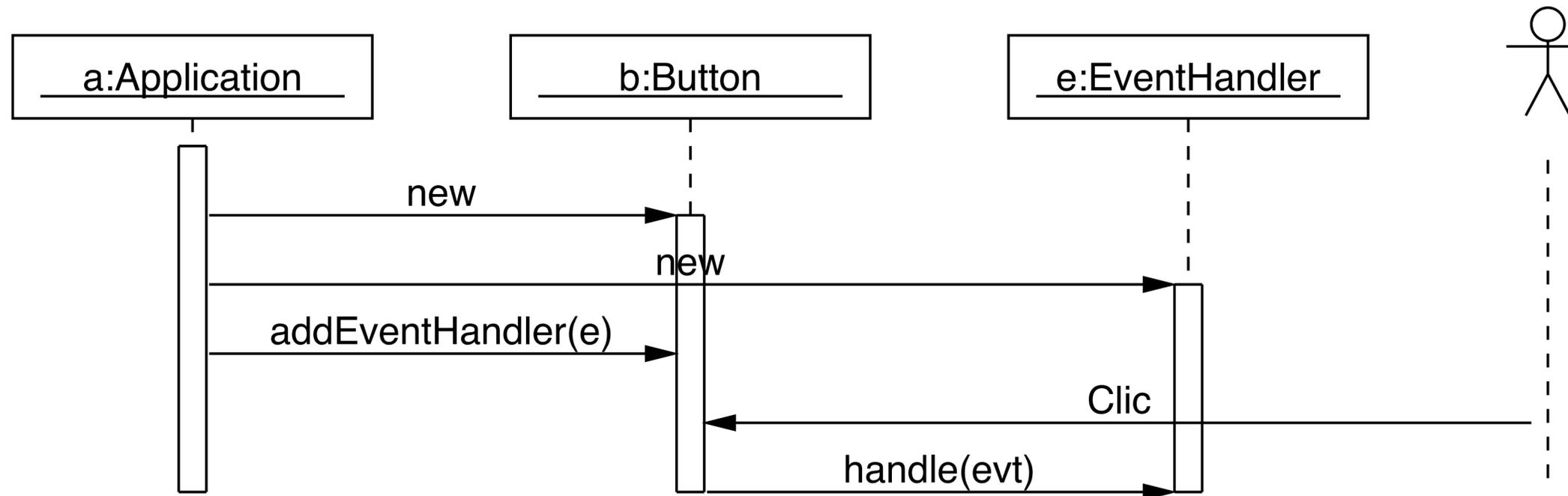
Le clic sur un bouton correspond à la classe d'événement **ActionEvent**

L'abonnement se fait par la méthode **addEventHandler** de **Button** (abonneur)

La notification se fait par l'appel de la méthode **handle** du **EventHandler<ActionEvent>** créé

Exemple : clic sur un bouton

24



Avec du code

cours2/HelloButton.java

25

```
public class HelloButton extends Application {  
  
    public void start(Stage stage) {  
        VBox root = new VBox();  
        Button b = new Button("Button");  
        b.addEventHandler(ActionEvent.ACTION, new MyClicHandler());  
        //...
```



```
public class MyClicHandler implements EventHandler<ActionEvent> {  
  
    public void handle(ActionEvent event) {  
        System.out.println("Clic sur bouton");  
    }  
}
```

ActionEvent vs MouseEvent

26

MouseEvent

Événement de la souris

ActionEvent

Événement de la souris

Événement touch d'une surface tactile

Appui touche espace du clavier

Cheminement des événements

Routes d'événements

28

Quand un événement est généré, les actions suivantes sont réalisées :

- sélection de la cible (event target)

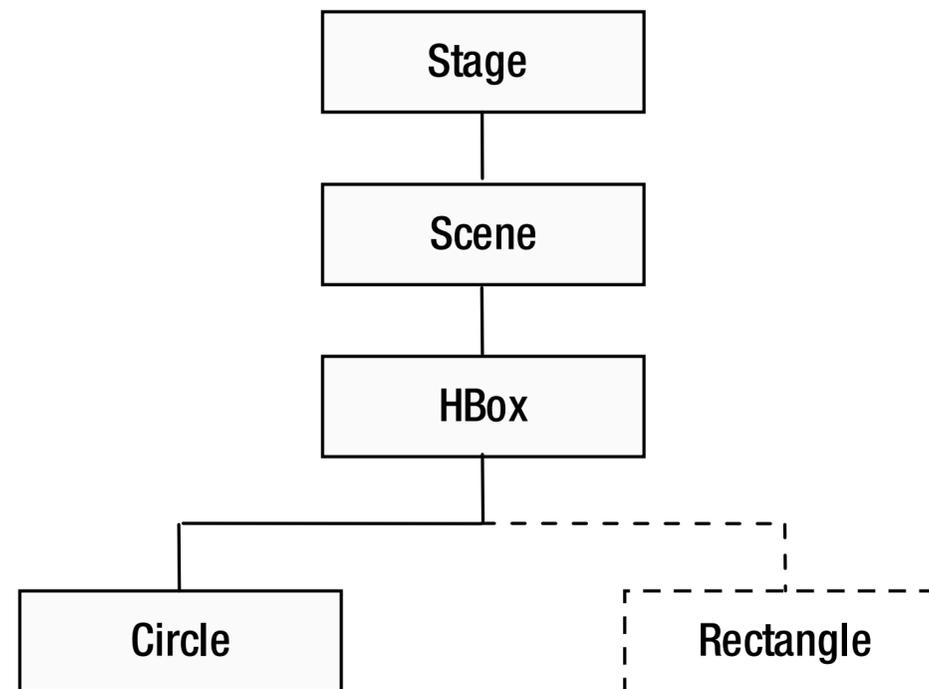
- construction de la route

- traversée de la route

Exemple de route

29

Construction de la route après un clic sur le cercle

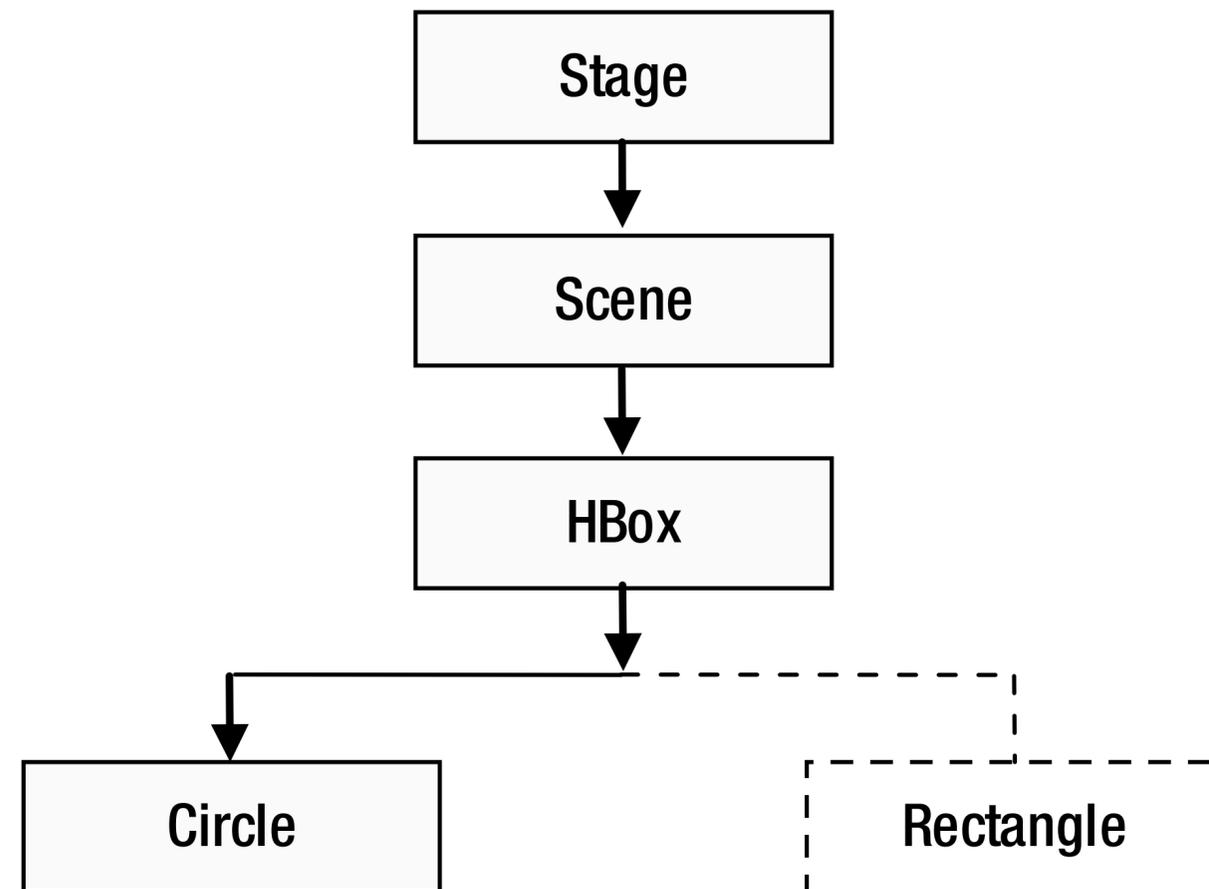


Event capture phase

30

1ère phase de traversée de la tête à la queue

Appel des event filters

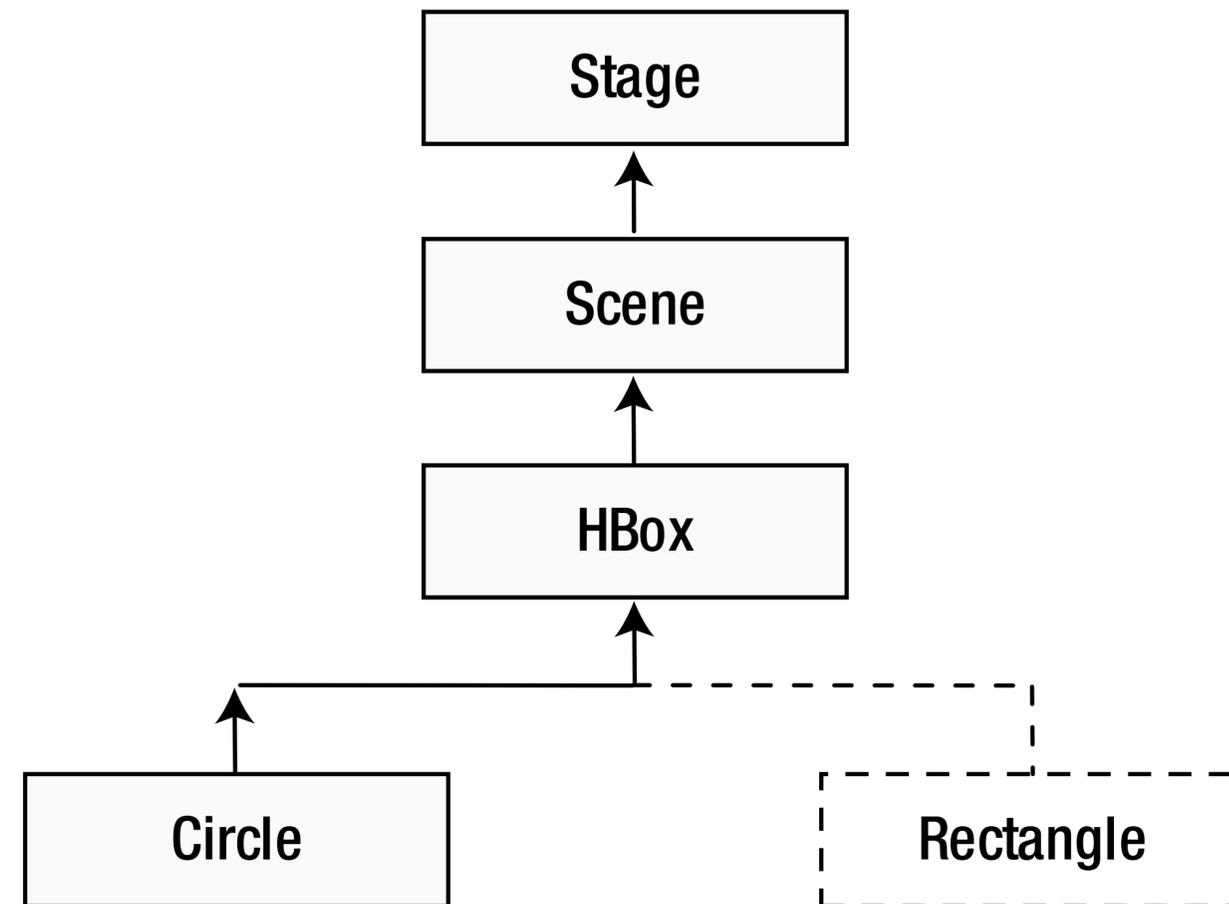


Event bubbling phase

31

2e phase de traversée de la queue à la tête

Appel des event handlers



En résumé

32

- gestion d'événements
- différents types d'événements
- mécanisme d'abonnement et de notification

Événement / Évènement

33

XMLitté

ÉVÉNEMENT

(é-vè-ne-man) s. m. **1°** Tout ce qui arrive.

Beaucoup d'événements ont démenti leurs causes, ROTROU, Antig. I, 2.

Conseils marqués par le doigt de Dieu, dont l'empreinte est si vive et si manifeste dans les événements que j'ai à traiter, qu'on ne peut résister à cette lumière, BOSSUET, Reine d'Anglet..

Le monde cependant se rit de mes excuses, Croit que, pour m'inspirer sur chaque événement, Apollon doit venir au premier mandement, BOILEAU, Épît. VI.

Les choses de dehors qu'on appelle les événements sont quelquefois plus fortes que la raison et que la nature, LA BRUY. VI.

Le grand Larousse

ÉVÉNEMENT ou **ÉVÈNEMENT** n.m. (du lat. *evenire*, se produire). **1.** Ce qui se produit, arrive ou apparaît : *Se tenir au courant des événements heure par heure.* **2.** Fait important, marquant : *Une envoyée spéciale couvre l'événement.* **3.** (En appos.). Qui suscite un très vif intérêt et fera date : *Un livre événement.* **4.** MATH. Partie d'un univers Ω réalisée quand l'une des éventualités la composant se réalise. ■ **Attendre un heureux événement**, être enceinte. ♦ n.m. pl. **1.** Ensemble des faits qui créent une situation ; conjoncture : *Il est dépassé par les événements.* **2.** Ensemble de faits marquants : *Les événements de mai 68.*