

# Programming of Interactive Systems

Anastasia.Bezerianos@Iri.fr  
(Nolwenn,Maudet@Iri.fr)

Based on Slides from Caroline Appert

## Week 6: a. State machines

[anastasia.Bezerianos@Iri.fr](mailto:anastasia.Bezerianos@Iri.fr)

Based on Slides from Caroline Appert

## Finite State Machines

**States** represent the state of your system:  
current window, active widgets, switching window...

**Transitions** are triggered by events:  
User events (mouse click, key press, ...)  
System events (timeout, incoming packet, ...)  
Custom events (gesture recognition, ...)

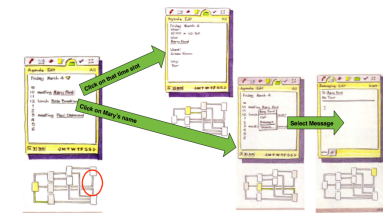
3

## Finite State Machines

Finite state machines (FSM)

Can help you think of the system behavior and possible states before coding

Work at different levels (remember interaction storyboards?)



4

# Describing **detailed** interactions

Finite state machines (FSM):

**States** represent interaction states:

Idling, dragging, drawing, ...

**Transitions** are triggered by events:

User events (mouse click, key press, ...)

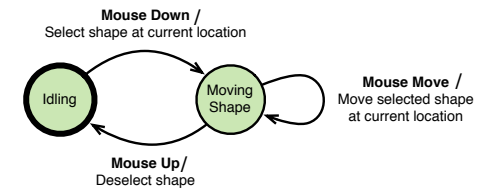
System events (timeout, incoming packet, ...)

Custom events (gesture recognition, ...)

5

# Example

Dragging a shape:

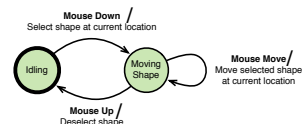


6

# Example

Implementing this with callbacks/listeners:

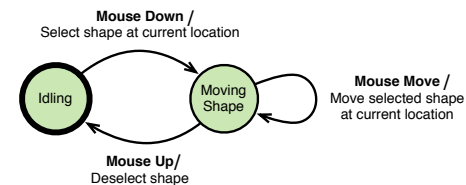
```
Shape dragged = null;
new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        // dragged is initialized, could call a function MouseDownState
    }
    public void mouseReleased(MouseEvent e) {
        // dragged is set back to null, could call a function IdleState
    }
}
new MouseMotionAdapter() {
    public void mouseDragged(MouseEvent e) {
        // dragged is translated, could call a function MovingShapeState
    }
}
```



7

# Example

This is okay for simple state machines.

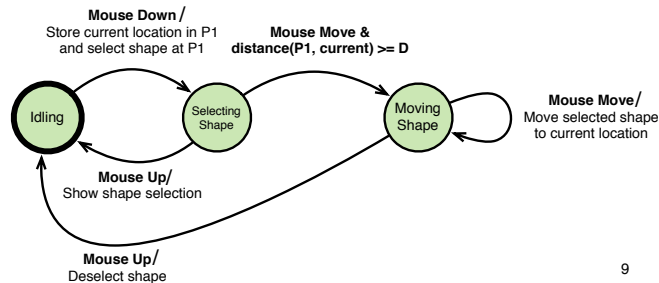


8

## Example

This is okay for simple state machines.

Let's consider the ability to select and drag an object:



9

## Example

Implementing the FSM with callbacks:

```

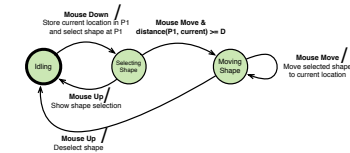
Shape dragged = null;
boolean dragging = false;

new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        // dragged is initialized
    }
    public void mouseReleased(MouseEvent e) {
        // dragged is set back to null
        // shape is selected if not dragged
    }
}

new MouseMotionAdapter() {
    public void mouseDragged(MouseEvent e) {
        if (!dragging) {
            // check if dragging occurs
        } else {
            // drag shape
        }
    }
}
    
```

State variables

Separate listeners



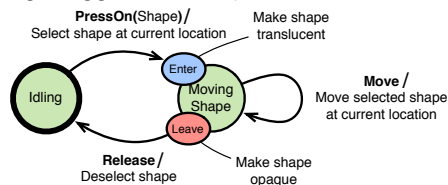
10

## Enter / Leave

In general, the user should know in which state the system is. To that end, **actions** can be triggered when **entering** or **leaving** a state to express this change.

Example:

When being dragged, the shape becomes translucent:



11

## Enter/Leave

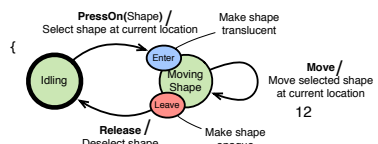
Implementing this with callbacks/listeners:

```

Shape dragged = null;

new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        // dragged is initialized, could do this in a separate function
        dragged = findShapeAt( e.getPoint() );
        dragged.setTransparent(true);
    }
    public void mouseReleased(MouseEvent e) {
        // dragged is set back to null, could do this in a separate function
        dragged = null;
        dragged.setTransparent(true);
    }
}

new MouseMotionAdapter() {
    public void mouseDragged(MouseEvent e) {
        // dragged is translated
    }
}
    
```



12

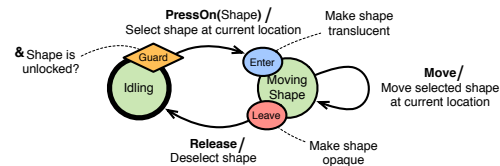
# Guard

Transitions can be moderated by a **guard** (use & symbol in transition).

If the **boolean** it returns is true, the transition will happen.

Example:

Only **unlocked** shapes can be moved:



13

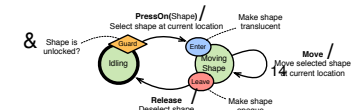
# Guard

Implementing this with callbacks/listeners:

```

Shape dragged = null;
new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        // dragged is initialized, could do this in separate function
        dragged = PressOn( e.getPoint() );
        if ( dragged.unlocked )
            dragged.setTransparent(true);
        else
            dragged = null;
    }
    public void mouseReleased(MouseEvent e) {
        // dragged is set back to null, could do this in a separate function
        dragged = null;
        dragged.setTransparent(true);
    }
}
new MouseMotionAdapter() {

```



# Calling Order

When a transition from a state A to a state B occurs, the following order should be followed:

```

Transition.guard()
StateA.leave()
Transition.action()
StateB.enter()

```



15

# Transitions

Transitions define two event **properties**:

- the **type** (press, release, move, etc.)
- the optional **target** (element type, group, widget, etc.).

A transition can have no specific target, meaning it occurs solely based on the nature of the event.

**Key events** and **custom events** are often target-less.

**Move** events should be target-less (you should know the target already from a previous event)

16

## Example Transitions

### Target-less

Click  
Press  
Release  
Drag  
Move  
- Enter  
- Leave  
KeyPress  
KeyRelease  
KeyType  
TimeOut

### To check for on shape or widget

ClickOnShape  
PressOnShape  
ReleaseOnShape  
DragOnShape  
MoveOnShape  
EnterOnShape  
LeaveOnShape

↑  
Shape events relate to specific shapes/items

17

## State Machines can

help break down complex tasks:

looking at entire program, or widgets as state machines

organize code based on states:

easier to debug

help communicate behavior to others

graphically before writing code

Often we draw state machines when  
expecting complex interactions and state  
transitions

18

## State Machines

Several UI programming libraries have

Finite State Machine extensions

e.g., SwingStates for Java Swing

<http://swingstates.sourceforge.net/>

19

## example

20

## reminder

### State machine reminder:

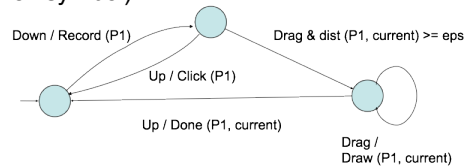
State (circle) = interaction state NOT location of the application

Transition (arc/link) = input events (Up, Down, Move, Drag, ...)

### State machine

actions associated with transitions (after the “/” symbol)

guard conditions (boolean checks) associated with transitions (after the “&” symbol)

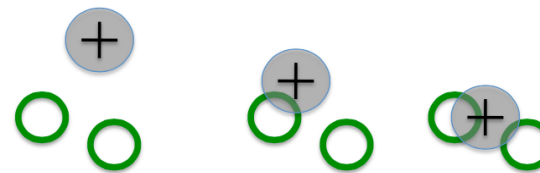


21

## example problem

Create a state machine for a technique:

Area cursor: area around cursor, can click on targets when inside (in first image a click selects nothing, in the others it selects the left target)



22

## example problem

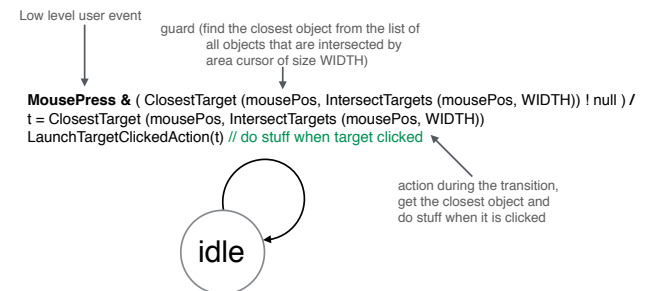
Can use:

List = IntersectTargets (mousePos, WIDTH)

Target = ClosestTarget (mousePos, List)

23

## example solution

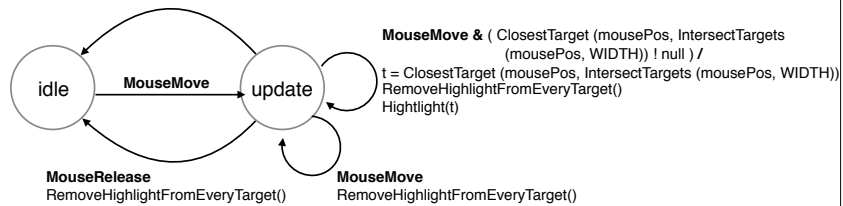


24

## example solution 2

(that highlights closest target)

```
MousePress & ( ClosestTarget (mousePos, IntersectTargets (mousePos, WIDTH)) != null ) /  
t = ClosestTarget (mousePos, IntersectTargets (mousePos, WIDTH))  
RemoveHighlightFromEveryTarget()  
Highlight(t)  
LaunchTargetClickedAction(t) // do stuff when target clicked
```



### Note:

RemoveHighlightFromEveryTarget() resets the colour of every target item  
Highlight (t) highlights the item that can be selected by the area cursor, e.g. changing it's border to be thicker

25