

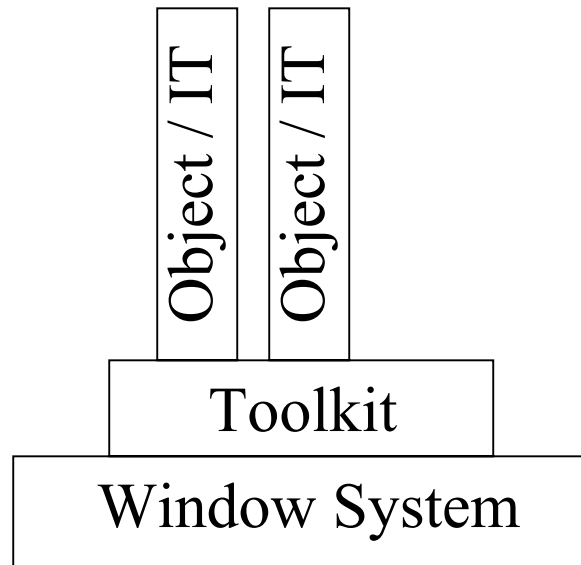
# **Interaction Techniques**



**Doug Bowman**

**April 13, 1998**

# What are ITs?



- **Methods for communicating with interactive objects**

- **widgets, interactors, controls, components**

- **Assumes simulated world**

# What are ITs?



- **Interaction technique includes:**
  - **input device(s)**
  - **mapping of input signals to semantics**
  - **feedback**
  - **output device(s)**
- **Addresses complete cycle of execution and evaluation**

# **HCI review - IT design guidelines**



- **Affordances**

- **Feedback**

- **Constraints**

- **Mechanics**

- **designing for mental models**

- **designing for novices vs. experts**

# **Why discuss design of ITs?**

---

- **Common ITs are standardized:**
  - ▮ **buttons, sliders, pop-up menus**
- **but, low-level implementation is non-standard – details can be important!**
  - ▮ **Windows vs. Mac menus, scrollbars**
- **ITs for future computing paradigms are not standardized**
  - ▮ **3D, VEs, hand-held, AR, ...**

# Implementation example

---

- **Line drawing**
- **Simple: click on 2 pts.**
  - **No affordances or feedback**
- **Affordance: change cursor shape**
- **Feedback: rubberbanding**

```
Accept press event for P1
P2 = P1
draw line P1-P2
Repeat
    new_P = current_pos()
    erase line P1-P2
    draw line P1-new_P
    P2 = new_P
Until release event
Act on line input
```

# Implementation example

- **Mechanics:**
  - **the loop must run at least 5 times/sec, preferably 10–30 times/sec**
- **Requires undraw/erase**
- **Problem with this implementation?**

```
Accept press event for P1
P2 = P1
draw line P1-P2
Repeat
    new_P = current_pos()
    erase line P1-P2
    draw line P1-new_P
    P2 = new_P
Until release event
Act on line input
```

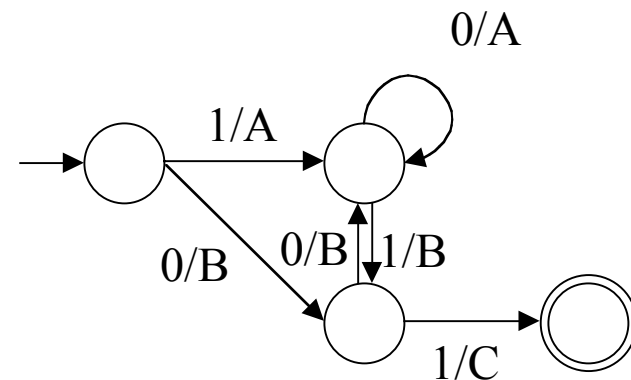
# **Event-driven IT framework**



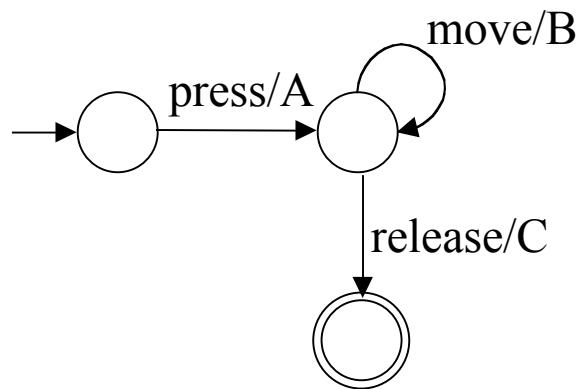
- **Take sets of actions from code and write as methods/modules**
- **Redistribute these methods in an event-driven framework (invoke them at proper time based on input events)**
- **We have to maintain state**
  - **moded implementation at low level**
  - **input sequence looks like a**

# FSM IT implementation

- Finite state machines
- States represent current interaction state or mode
- Transitions represent input events
- Actions on



# Line drawing with FSMs

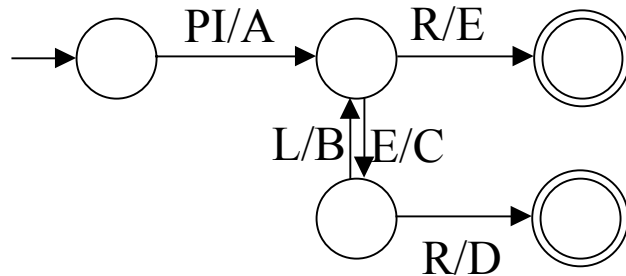


Accept press event for P1

A {  
P2 = P1  
draw line P1-P2  
Repeat  
    new\_P = current\_pos()  
    erase line P1-P2  
    draw line P1-new\_P  
    P2 = new\_P  
Until release event

C { Act on line input

# Buttons with FSMs



PI: Press inside

L: Leave

E: Enter

R: Release

- Press inside: highlight (A)
- Move in/out: change highlight (B, C)
- Release inside: act (E)
- Release outside: do nothing (D)

# FSM issues



- **Events may be complex (e.g. press-inside)**
  - **requires intelligent higher-level dispatcher**
  - **may require read/write of global variables**
- **FSMs allow analysis of ITs:**
  - **Are all possible inputs (incl. errors) handled?**
  - **What are the set of legal next inputs?**

# FSM implementation

```
state = start_state
while (1){
    switch(state){
        case 0:
            switch(event){
                case loc_move:
                    <code>
                    break;
                <etc>
            }
            <etc>
        }
    }
}
```

- **Just a big switch statement that takes the correct transition and action from the current state and input**
- **Similar to event dispatch process**

# Table-driven FSMs

```
state = start_state
while (1){
    event = waitforevent();
    for each transition tr
    out of trans_tab[state]{
        if tr.match(event){
            tr.action();
            state=tr.tostate();
            break;
        }
    }
}
```

- If events are a simple set, the FSM can be driven from a table
- This allows reuse of code
- Can afford features like analysis

# **FSM drawbacks**



- **The natural graphical notation doesn't scale well to complex interactions**
- **FSMs are by nature moded**
  - **fine for small ITs (user doesn't perceive modes)**
  - **not good for specifying full dialogs**

# Standard interaction tasks



- Positioning
  - Selecting / Picking
  - Text entry
  - Quantification
  - Command entry
- 
- 3D tasks

# Standard ITs

---

- Positioning
- Selecting / Picking
- Text entry
- Quantification
- Command entry
- 3D tasks
- Mouse / cursor
- Mouse & buttons
- Keyboard / Text fields
- Slider
- Menus / buttons
- ???

# Original Macintosh 7 ITs



- Button
  - Slider
  - Pulldown menu
  - Check box
  - Radio buttons
  - Text entry fields
  - File pick / save
- Also used:
    - window close
    - window resize
    - dragging icons & folders
    - open icons & folders

# Picking



- Really a naming mechanism
- Always implemented with pointing
- Recognition vs. recall
- How to pick given cursor position?
  - ┆ Clipping algorithm w/ small region
  - ┆ Recursive pick traversal

# Pick ambiguity



- Hierarchical
- Spatial
  
- Solutions:
  - strong typing
  - allow user to choose

# Positioning & dragging



- **Examples:**

- **move an icon**

- **resize a window**

- **Affordances?**

- **Feedback?**

- **Typically uses a mouse, but other devices can be used as well**

# Choice of devices



- **Absolute locators: one-to-one mapping from device to cursor**
  - e.g. tablet
  - faster & easier
  - fixed range therefore less accurate
- **Relative locators: map device movement to rate of change of cursor**
  - e.g. joystick
  - harder motor skills
  - range is infinite

# What about the mouse?



- **Absolute?**
- **Relative?**
  
- **Third type: “clutched absolute”**
  - **absolute mapping within a range, but device can then be clutched and the range moved**
  - **good compromise**
  - **trackballs also in this category**