Wearable Computing

Holger Kenn

Universität Bremen

WS 05/06
Abstract UIs

Wearable UIs
Task Trees

- Task Trees
  - Formal specification of user interaction
  - Can be used to support development
- ConcurTaskTrees
  - Temporal Operators
  - Examples
Wearable UIs

- Supporting a primary task, i.e. UI driven by external task
- Context-dependent (primary task is one context source)
- Non-”point-and-click”, i.e. No WIMP-based UI
- Sometimes no graphical UI at all
- Rich set of in- and output devices
- Question: How to write (and reuse) code for “generic” wearable computer?
Characterizing Wearable UIs

- Displaying information and changing state (like CTTs)
- Additionally: Context information
  - Context-dependent presentation
  - Context includes input and output modes and devices available
  - Context change triggers information display / state change

- Idea:
  - specify abstract UI using CTTs
  - use context change triggers like input in CTTs
  - decide context-dependent presentation during runtime
Example: a web browser with two presentation modes
- Desktop mode: Like firefox
- Mobile mode: like opera “small screen rendering”

Specification of UI (= html document, links) the same

“Rendering” of UI different:
- Compress graphics, change positions, use different fonts
- Change interaction: no mouse click, but chose links via cursor keys
Simple Example: Write Aircraft Repair Report
- Input text of repair report
- Indicate that the repair report entered is complete
- i.e. use CTT to specify abstract model
- Web browser equivalent: Form
  - Text input field
  - “submit” button
PDA: Java 1.2 (AWT)

1. `Panel p = new Panel();`
2. `p.add(new Label("Enter Report"));`
3. `TextField tf = new TextField("Your Report Here", 256);`
4. `p.add(tf);`
5. `Button b = new Button("Save");`
6. `p.add(b);`
Swing implementation

- Desktop: Java 5 (Swing)

```java
1 JPanel p = new JPanel();
2 p.add(new JLabel("Enter
Report");
3 JTextField tf = new JTextField("Your
Report
Here", 256);
4 p.add(tf);
5 JButton b = new JButton("Save");
6 p.add(b);
```
QT implementation

1. `QLabel *reportLabel = new QLabel(tr("Enter report"));`
2. `QTextEdit *reportEdit = new QTextEdit;`
3. `QPushButton *saveButton = new QPushButton(tr("Save"));`
4. `myLayout = new QHBoxLayout;`
5. `myLayout->addWidget(reportLabel);`
6. `myLayout->addWidget(reportEdit);`
7. `myLayout->addWidget(saveButton);`
Abstract to concrete

- How to get from abstract to concrete?
  - Idea 1: Use an expert programmer, give him the spec, let him program, use result
- How about different devices?
  - Idea 1a: Use expert for every possible device, send to expert programmer, let them work together.
- How about different contexts?
  - Idea 1b: Use domain expert to describe contexts, send to device expert to design context-dependent optimal display for specific device, send to programmer, program
- Only viable for small number of devices and huge sales. i.e. mobile phone games
Abstract to concrete (2)

- Can we do without all these experts?
- Idea 2: Divide the application program in two parts: The abstract UI and the renderer
- How about different devices?
  - The renderer can be device-specific: It knows best how to use UI elements of the target device
- How about different contexts?
  - The renderer itself can use context information in a device-specific way
- The abstract UI can choose from a number of available renderers. This choice can be based on device availability, user preference, context.
AbstractUI implementation

- AbstractUI

```java
mSave = new TriggerItem2(
    new TextData("Save"), false, this);
mComment = new TextInputItem2(
    new TextData("Comment"),
    20, "Your text here", this);
mComment.setNext(mSave);
mRoot = new GroupItem2(
    new TextData("Write Repair Report"),
    this);
mRoot.setSub(mComment);
```
Open questions

- **Fundamental question: What can the AbstractUI express?**
  - Speech-driven UI?
  - How to deal with non-renderable objects? (picture on audio-UI)

- **Technical question: How can we implement it?**
  - How can we specify an AbstractUI Model? XML?
  - How can the renderer decide what subtree of the CTT it renders? on-demand query mechanism?
Wearable UI Metaphor

- **Output Mechanism**
  - Visual: HMD
  - Audio

- **Input Mechanism**
  - Keys: Keyboard, Twiddler
  - Hands: gestures, direct manipulation
  - Speech

- **Interaction Methods**
  - menu selection, direct manipulation, form fill-in
  - command language, natural Speech
Winspect GUI

- Java Implementation
- Uses HMD and “hands-free interaction”
- GUI elements optimized for wearable use
  - Colors, font sizes, highlighting
- Interaction based on dataglove
  - Direct Manipulation: Motion, Turn
  - Gesture for selection
Winspect UI HMD

Image from T. Nicolai
Winspect Direct Manipulation

Image from T. Nicolai
WearableUI

- Renderer for AbstractUI
- Uses HMD and “hands-free interaction”
- GUI elements optimized for wearable use
  - Colors, font sizes, highlighting
  - Few elements displayed
  - Shows in the area of visual focus
- Interaction based on dataglove
  - Hand gestures to navigate and select
  - Additional keyboard for text entry
Wearable UI Gesture

Image from H. Witt
Wearable UI Glove

Image from H. Witt
Wearable UI HMD

Image from H. Witt
Summary

- AbstractUI
  - Device-independent
  - Context-aware

- WearableUI
  - Uses AbstractUI
  - Wearable interaction mode