Informing Automatic Generation of Remote Control Interfaces with Human Designs

Jeffrey Nichols • jeffreyn@cs.cmu.edu • Carnegie Mellon University • Human Computer Interaction Institute • http://www.cs.cmu.edu/~pebbles/puc/

Personal Universal Controller

Problem

How can we *automatically* generate remote control interfaces for everyday appliances on a handheld computer?

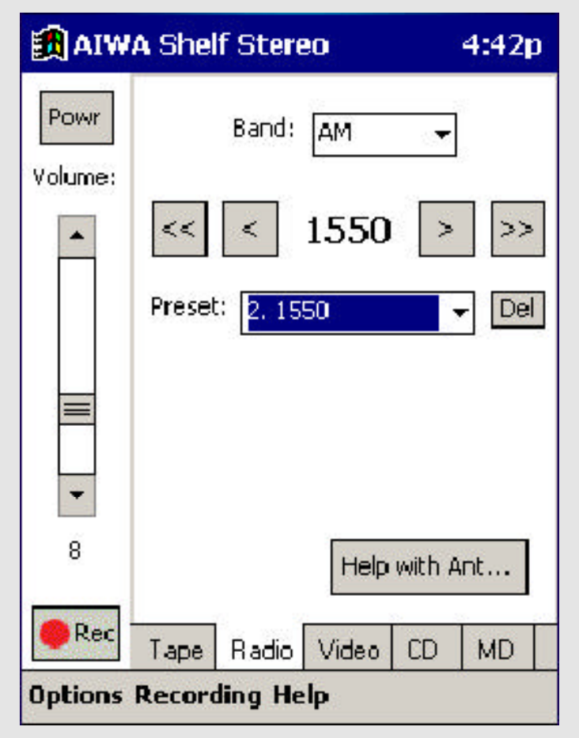
Process

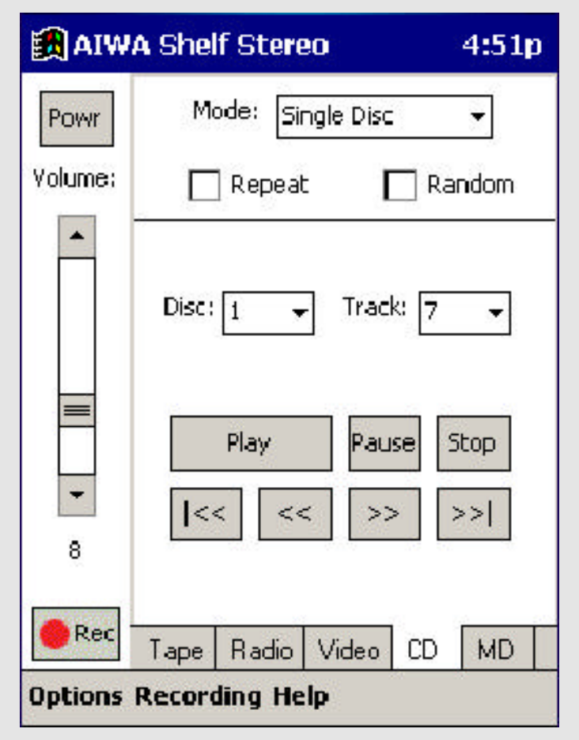
1. Manually construct interfaces for several appliances by hand and evaluate them with users.
2. Analyze the interfaces to understand what appliance information was needed to construct them.
3. Design a language for describing an appliance that includes all information necessary for creating an interface.
4. Build an interface generator that reads a file written in the language and automatically generates a remote control interface.

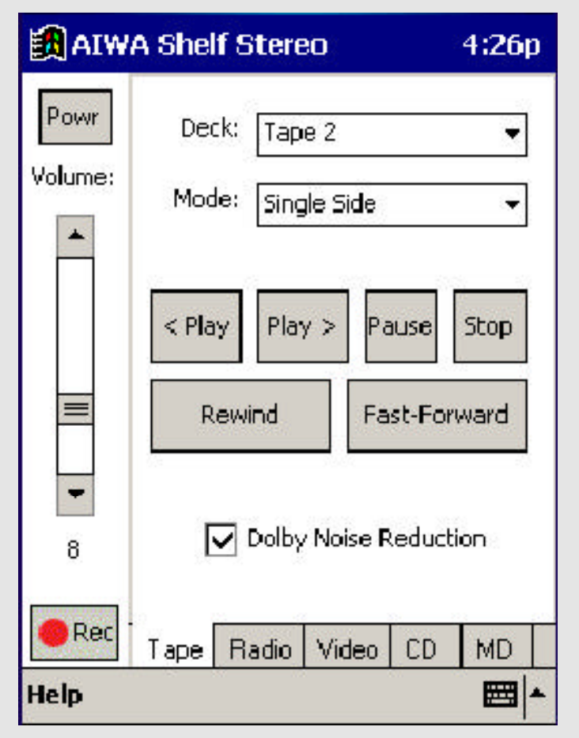
Acknowledgements

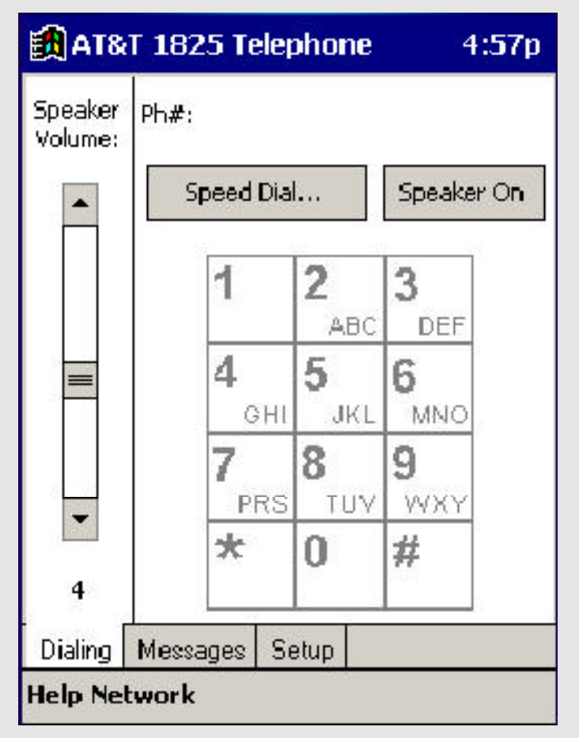
This work was conducted as part of the Pebbles project, directed by Brad A. Myers, in collaboration with Michael Higgins, Joe Hughes, and Peter Lucas of MAYA Design. It was funded by grants from NSF, Microsoft, and the Pittsburgh Digital Greenhouse, and equipment grants from Symbol Technologies, Hewlett-Packard, and Lucent. The National Science Foundation funded this work through a Graduate Research Fellowship and under Grant No. IIS-0117658. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect those of the National Science Foundation.

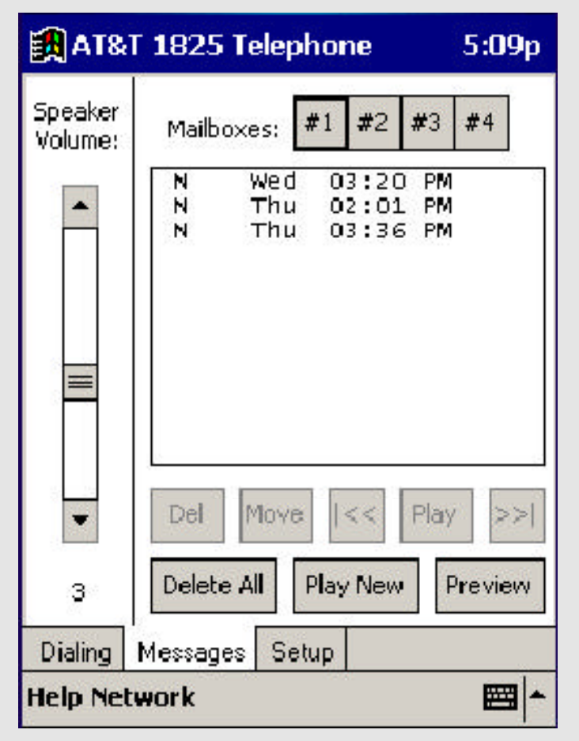
human designs











Analysis of Human Designs

We created interfaces for an AIWA CX-NMT70 shelf-stereo and an AT&T 1825 office telephone on a Microsoft PocketPC. We then analyzed these interfaces to understand the functional information that was needed for their creation. Our specification language includes the information that we found was needed to create remote control interfaces.

State Variables and Commands

Any manipulable appliance element can be represented by either a state variable or a command. Each state variable has a type that tells the interface generator how it can be manipulated. Some elements, such as the seek button on a radio, must be represented as commands, because their result cannot described as a deterministic change to a variable. Today's remote controls could be generated by a specification that contained only commands.

Type Information

The type information of each state variable tells the interface generator how the user may manipulate that variable. Boolean, enumerated, numeric, and string types are currently available.

Labeling Information

Each component must be labeled so that users can distinguish elements. Different form factors require different kinds of labels. Each semantic label for a variable or command in our specification language contains several strings that may be used based on available space.

Group Tree

Most interfaces can be described by a tree, where each leaf is a component and each branch is a panel. Our language uses a similar structure to suggest to the interface generator which variables and commands are most related to others.

Dependency Information

Enabling and disabling components is very useful for guiding users to available functionality. Our language contains formulas that specify when a variable or command is available based upon the values of other state variables. Equal-to, greater-than, and less-than relations are supported, and multiple relations may be combined with the logical AND and OR operations. We have also discovered that this information can be useful for determining the panel structure of an interface.

specification language

```
<?xml version="1.0" encoding="UTF-8"?>

<spec name="Audiophase 5 CD Stereo">
  <groupings>
    <group>
      <state name="PowerState" priority="10">
        <type name="OnOffType">
          <valueSpace>
            <boolean/>
          </valueSpace>
          <valueLabels>
            <map index="false">
              <label>Off</label>
            </map>
            <map index="true">
              <label>On</label>
            </map>
          </valueLabels>
        </type>

        <labels>
          <label>Stereo Power</label>
          <label>Power</label>
          <label>Powr</label>
          <label>Pwr</label>
        </labels>
      </state>

      <group>
        <active-if>
         <equals state="PowerState">true</equals>
        </active-if>

        <labels>
          <label>Volume</label>
         <label>Vol</label>
        </labels>

        <command name="VolumeUp" priority="10">
          <labels>
            <label>Volume Up</label>
            <label>Vol. Up</label>
           <label>^</label>
          </labels>
        </command>

        <command name="VolumeDn" priority="10">
          <labels>
           <label>Volume Down</label>
           <label>Vol. Down</label>
           <label>Down</label>
           <label>v</label>
          </labels>
        </command>
      </group>

      <group>
        <active-if>
         <equals state="PowerState">true</equals>
        </active-if>

        <state name="ModeState">
          <type>
            <valueSpace>
              <enumerated>
                <items>4</items>
              </enumerated>
            </valueSpace>
            <valueLabels>
              <map index="1">
                <label>Tape</label>
              </map>
              <map index="2">
                <label>CD</label>
              </map>
              <map index="3">
                <label>AUX</label>
              </map>
              <map index="4">
                <label>Tuner</label>
              </map>
            </valueLabels>
          </type>

          <labels>
            <label>Output Mode</label>
            <label>Mode</label>
          </labels>
        </state>
```

Interface Generation

We created an interface generator for the Compaq iPAQ using the PersonalJava API and Insignia's Jeode VM. This generator takes a file written with our specification language and creates interfaces that can be used to control actual appliances.

Panel Structure

Components may be placed across different panels, depending on how closely they are grouped and what they depend upon. Two sets of components may be placed on overlapping panels if dependency information shows that the sets are never active at the same time. This happens if the appliance has modes.

Column and Row Layout

Within a panel, components are typically placed in a column, but more complex arrangements may be created if particular patterns are found in the group tree. For example, if only two components are found in a group, then those components will be placed in a row together.

Component Choice

The interface component for a state variable or command is chosen by a decision tree. Commands are always represented by buttons, but state variables are represented differently based upon type, writeability, and other features.

Label Choice

Each semantic label in the specification file may be represented by multiple strings. The interface generator currently uses the longest string available that will fit in the space allotted by the rest of the generation process.

Future Work

We are currently working to extend our specification language and improve our automatically generated layouts.

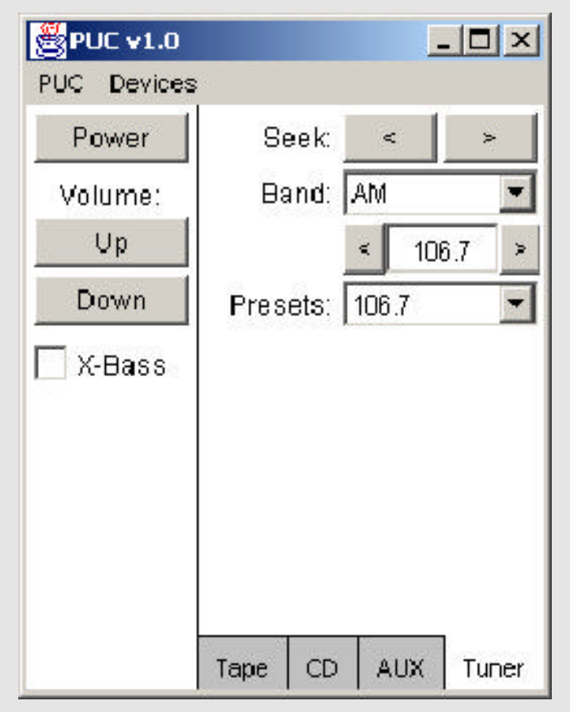
Specification Language

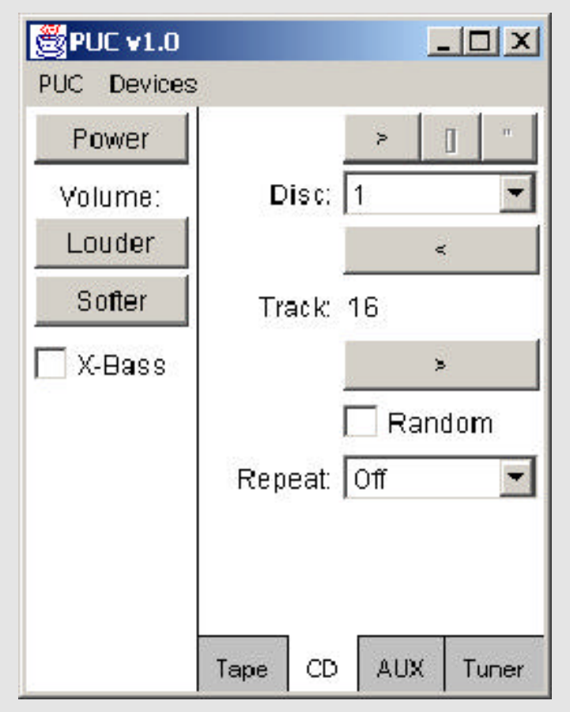
- The specification language does not currently support lists, which are used by many appliances.
- Information must be added to the specification to support generating speech interfaces.

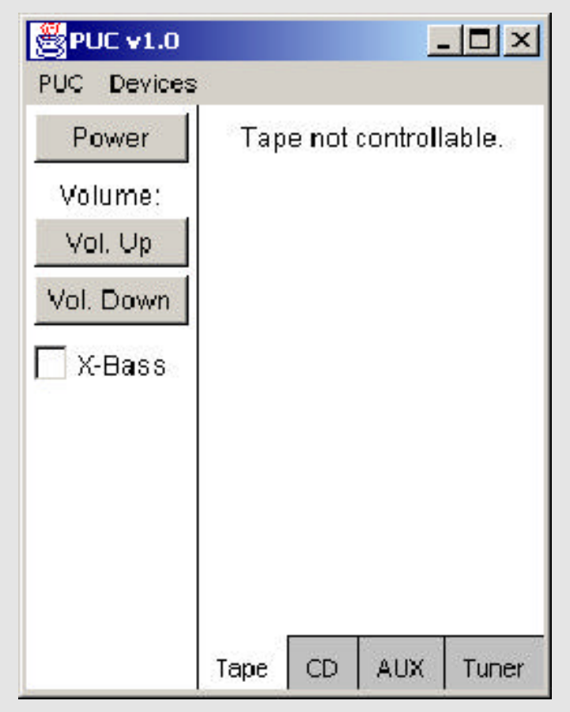
Interface Layout

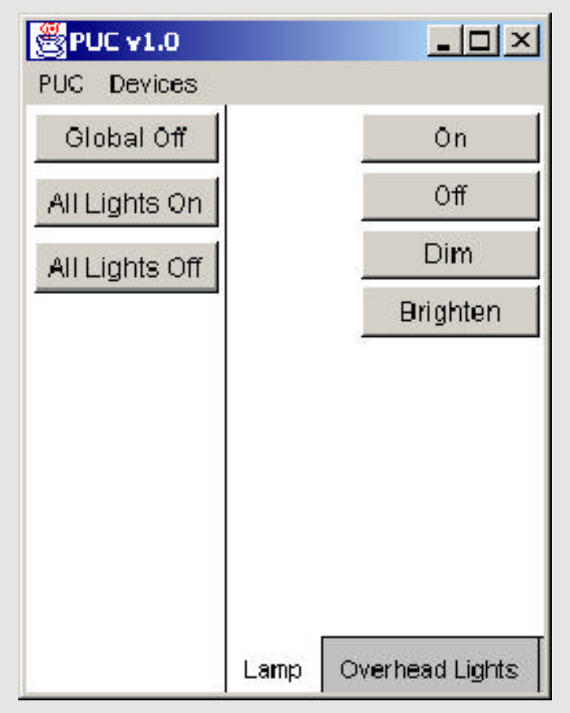
- Our interface layout algorithms need some knowledge of standard component arrangements. We need better methods of including this information in our language
- Our layout algorithm must deal appropriately with situations where there is not enough space. We are considering using backtracking, where previous decisions are re-thought in the context of newly discovered constraints.

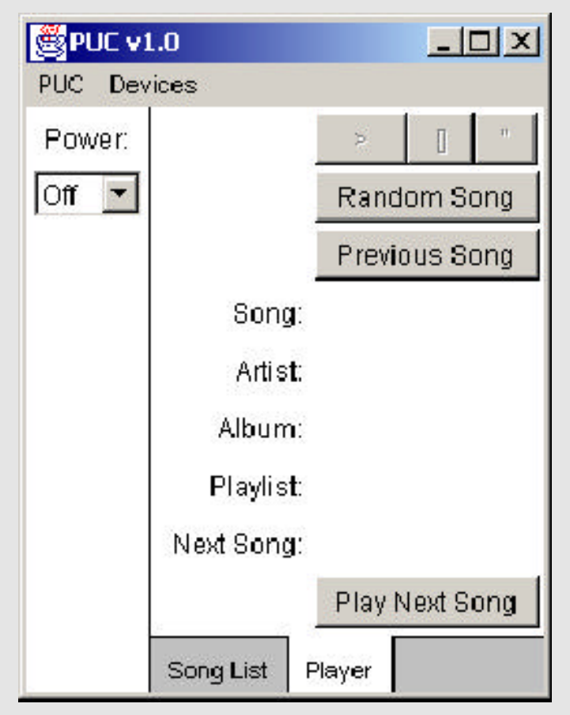
automatic designs


PUC v1.0
PUC Devices
Power
Volume:
Up
Down
X-Bass
Seek:
Band: AM
Presets:
Tape CD AUX Tuner


PUC v1.0
PUC Devices
Power
Volume:
Louder
Softer
X-Bass
Disc:
Track: 16
Random
Repeat: Off
Tape CD AUX Tuner


PUC v1.0
PUC Devices
Power
Volume:
Vol. Up
Vol. Down
X-Bass
Tape not controllable.
Tape CD AUX Tuner


PUC v1.0
PUC Devices
Global Off
All Lights On
All Lights Off
On
Off
Dim
Brighten
Lamp Overhead Lights


PUC v1.0
PUC Devices
Power:
Off
Random Song
Previous Song
Song:
Artist:
Album:
Playlist:
Next Song:
Play Next Song
Song List Player