Lecture 24: Accessibility

Today's Topics

- Kinds of impairments
- Assistive technology
- Accessibility guidelines

Today's lecture is about **accessibility**, which generally means making it possible for users with impairments to use a graphical user interface. We'll talk about the kinds of impairments we'll be concerned with; the technology (both software and hardware) that help users deal with them; and some guidelines for designing UIs for accessibility.

Diversity of Ability

- Visual impairments
  - Color perception
  - Acuity ("legal blindness")
  - Total blindness
- Hearing impairments
  - Often varies with frequency
- Motor disabilities
  - Tremor and spasms
  - Muscle weakness and fatigue
  - Paralysis

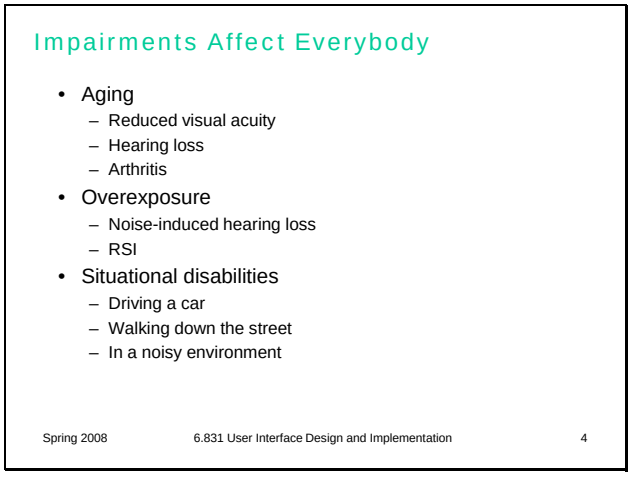
We'll focus on physical impairments for this lecture, specifically problems in vision, hearing, or motor control, because that's how "accessibility" is generally understood. But note that there are other impairments that are relevant to making a user interface usable by a wide spectrum of people. Some have cognitive disabilities, like difficulty learning or paying attention. Others have difficulty reading, either because they never learned or because they read a language different from yours. (We'll talk about this last one in the internationalization lecture coming up.)

We've talked about one form of **vision impairment** already: color blindness. Even more common than that, of course, is impaired visual acuity, i.e. inability to focus clearly. For most people, visual acuity problems can be corrected with glasses or contact lenses, but some have

uncorrectably bad vision. Roughly a million Americans are legally blind (unable to read even the biggest letter on an eye chart, even with corrective lenses). Perhaps 10% of them are totally blind, unable to sense light at all.

Hearing impairments affect the ability to sense sound intensity, and range in a spectrum from reduced sensitivity to complete loss. Hearing impairments often depend on sound frequency; a person may hear lower frequencies well, but not high frequencies.

Motor disabilities come in many different forms, and have many different causes. Sufferers of cerebral palsy experience uncontrollable tremors and spasms, making it difficult to make fine motor movements. Muscular dystrophy and multiple sclerosis can make muscles weak, and sufferers may tire easily when doing repeated or large muscle movements. Neural damage can cause complete paralysis of limbs.



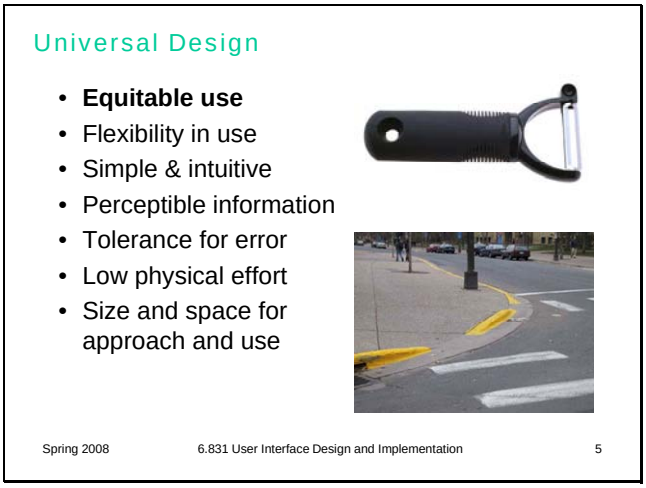
But physical impairments, or their effects, aren't limited to people with congenital diseases or trauma. **Aging** causes all three kinds of impairments. We've already discussed some of the impacts of aging on color vision. Older adults may also have reduced acuity, reduced hearing, and reduced mobility (specifically arthritis, which involves tremors, pain, and fatigue).

Overuse can also cause impairment to younger people, as if aging prematurely. Most people don't blind themselves by staring at the sun, but some lose their hearing prematurely by working in extremely loud environments (or listening to iPods?). Repetitive stress injury (RSI) is a motor impairment caused by excessive computer use (among other activities), with symptoms including pain, numbness, and weakness.

Finally, all of us can experience **situational disabilities**: temporary conditions of ourselves or our environment that effectively cause impairment. For example, when you're driving a car, your hands

and eyes are occupied with the driving task, so with respect to an in-dashboard computer, you're experiencing visual and motor impairments. Similarly, when you're walking down the street, your visual abilities are diminished (because you have to watch where you're going), and your ability to do fine motor control is reduced as well (because every step jars your entire body). In a noisy environment (say, the deck of an aircraft carrier), you can't hear. When the sunlight is glaring on your laptop screen, you can't see.

The take-away message from this is that impairments affect everybody, and vision, hearing, and motor lie on a spectrum of ability that varies widely between users and over time. So we should take them into account when we're designing.



Universal design is a school of thought that takes this fact explicitly to heart, by seeking to design for *all* users, across as much of the spectrum of capability as possible. Contrast this with the attitude that is implicit in this class, and in most actual design, where we mainly design for the *typical* user, and then (8 weeks into the course?) discuss how to make it "accessible" to everybody else. Universal design challenges us to think about supporting a wide range of capability from the start.

The proponents of Universal Design (http://www.design.ncsu.edu/cud/) have put forth seven guiding principles, listed here. Several are already familiar to us (simplicity, learnability, visibility, errors), and several are more relevant mainly to physical design (effort, size, space). But the first principle is the heart of the universal design philosophy: **equitable use**. As much as possible, all users should have the *same* interface, so that groups with differing abilities are not stigmatized. (If the identical interface isn't possible, then provide equivalent interfaces.)

Good universal designs are not dumbed down to make them universal; you shouldn't sacrifice efficiency or flexibility for typical users in order to enable users with reduced ability. Instead, a good universal design has features that make the design better for everyone. Classic examples are kitchen

tools with fat, textured handles (like the vegetable peeler shown here); not only are they easier for arthritis sufferers to grip, they're more comfortable and less error-prone for typical users too. Similarly, a sidewalk curb cut not only enables wheelchair users, but also parents with strollers, travellers with luggage, and people pushing carts. Even walkers may find the ramp more convenient than a step.

It's not always clear how to find a universal design, but it's a goal worth striving toward.

Assistive Technology

- Output
  - Screen magnifier
  - Screen reader
  - Braille display
  - Screen flashing on sound
- Pointing
  - Eye or head tracker
  - Puff-and-sip
  - Mouse keys
- Typing
  - Onscreen keyboards
  - Sticky keys
  - Speech recognition

For using computers, users with physical impairments use a variety of assistive technology, some hardware, some software.

Screen magnifier software magnifies part of the display to make it easier to read, which helps users who have reduced acuity but are not totally blind. **Screen readers** help the totally blind, by reading the contents of the display aloud as speech. For totally blind users who know Braille, a screen reader can be connected to a Braille display, which lets them read the screen privately (and quietly) and probably faster as well.

For hearing-impaired users, graphical user interfaces pose fewer problems, because far less information is conveyed by auditory cues. System sounds (like beeps) may be translated into a screen flash; videos may include closed captioning.

On the input side, there are alternative pointing devices. Eye gaze or head pose tracking can move the mouse cursor around the screen without the use of the hands. Puff-and-sip devices (in which the user blows or sucks a tube) can be used to click a button, often in combination with a mouth-driven joystick. Users with less extreme motor impairments may use touchpads or trackballs, which are less tiring than mice because they require smaller movements. The mouse cursor can also be moved around by keyboard keys; Windows and Mac both have this feature built-in.

Note that users of screen readers are not likely to use a pointing device at all, because they can't see a

mouse cursor or targets on the screen. So totally blind users typically use a keyboard exclusively.

For keyboard input by severely motor-impaired users, a pointing device can be combined with an onscreen keyboard. Keyboard driver software can often be adjusted to make it easier to use, e.g. turning off autorepeat so that keys don't have to be released quickly, or making modifier keys (like Shift and Control) "sticky" so that the user doesn't have to hold down multiple keys at once.

Speech recognition offers another way to give both command and text input without using the hands.

Accessibility Guidelines

- Section 508
- W3C Accessibility Initiative

Now let's discuss some specific guidelines for creating accessible interfaces. Most of these guidelines are targeted at making your interface amenable to assistive technology, e.g. helping a screen reader do a better job of translating the display into text.

The guidelines that follow are summarized from two sources. **Section 508** is an accessibility standard for web sites and software created by US government agencies or government contractors. Anybody who wants to sell software to the US government must follow the Section 508 rules, which cover both desktop software and web sites. The **W3C Accessibility Initiative** is a group in the World Wide Web Consortium that has produced a list of (voluntary) accessibility guidelines for web sites.

Section 508 rules:

http://www.section508.gov/index.cfm?FuseAction=Content&ID=12#Software

W3C guidelines:

http://www.w3.org/TR/WAI-WEBCONTENT/

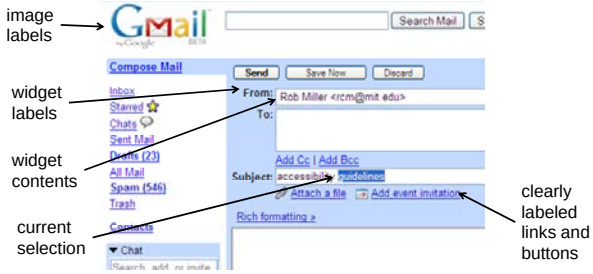
Support Keyboard Access

- Pointing interactions should have keyboard alternatives
  - Menus should be controllable by the keyboard
  - Forms and links should be navigable by keyboard
  - Needed by motor-impaired and vision-impaired

Since some users will be less able or unable to use a pointing device (e.g. users with screen readers), an accessible interface should support keyboard alternatives for all interactions. Menus should be controllable by the keyboard, either using accelerators or by allowing navigation around the menu. Similarly, the user should be able to move the keyboard focus around a form or activate a link in a web page by keyboard alone.

Provide Text to Screen Readers

- Screen readers need to automatically transform the display into spoken language



An accessible interface should be amenable to screen reading. All visual content should have textual alternatives; i.e., images should have captions or labels describing or naming their contents, so that screen reading software can articulate it.

Widgets, like textboxes and checkboxes, should have labels associated with them. This association can't be merely visual (i.e. "From:" happens to be next to the textbox), but programmatically available to the screen reader, so that the screen reader can ask for the label of that textbox and get "From address" or something similar back. The interface that screen readers use to access this information is called an "accessibility API"; we'll talk about the APIs for Java and HTML later in this lecture.

Screen readers also need to find out: the current value of a textbox or other widget; the widget with the keyboard focus; and the location of the text selection in a textbox.

For web pages with hyperlinks, the links should be clearly labeled with the identity of the target page, not something vague like "click here". This is because users of screen readers don't necessarily read the entire display linearly, from start to end. This would be painfully inefficient. Instead, they skip through, scanning the page aurally much as users with normal vision would scan it visually. A screen reader can be directed to read all the links, skipping over other text, so the links should be self-descriptive.

Don't Rely on Sound Alone

- Flash as well as beep
- Closed captioning for videos

For the sake of hearing-impaired users, don't rely on sound as the only channel by which some bit of information is delivered. To get the user's attention, don't just beep; briefly flash a window or the screen as well. Videos should include closed-captioning information.

User Control Over Colors and Fonts

- Allow user to choose high-contrast colors
- Allow user to enlarge fonts
- Don't rely on color alone

Users with impaired vision may prefer to use high-contrast colors, so allow the user to change the color scheme if necessary. Similarly, allow the user to enlarge the font size for easier readability.

We've already discussed not relying on color as a sole indicator for conveying information, because of color blindness. Use secondary cues too.

Accessibility APIs

- javax.accessibility
  - Swing widgets implement Accessible
  - getAccessibleContext() returns an object with labels, descriptions, content, selections for the widget
  - Platform-specific accessibility APIs are similar
- HTML accessibility features
  - alt and title attributes
  - <label> element
  - accesskey attribute
  - aural CSS

Java and HTML both offer built-in ways to make your interface more accessible.

The Java Accessibility API provides an interface for screen readers to inspect a Swing interface. All built-in Swing widgets implement it, so by using widgets, you get accessibility to screen readers for free. The API has one method, getAccessibleContext(), which returns an object containing information about the widget, such as a label for it, description, its current value, its current text selection, etc. Major desktop systems (Windows, Mac, Gnome, KDE) have their own equivalents for this API, and Java has a bridge that allows your Swing interface to be inspected through the platform-specific API.

In HTML, probably the most well-known accessibility feature is the **alt** attribute on images, which specifies a caption or description of the

image for the sake of a screen reader. Other elements (like frames) have a title attribute for the same purpose. Textboxes, checkboxes, and other form controls can be programmatically labeled by the <label> element.

For keyboard operation of a web page, HTML offers the accesskey attribute, which can be added to links and form controls among other elements. For example, accesskey="c" specifies that Alt-C (or some other browser-specific modifier) should navigate to or invoke the element. Unfortunately it's difficult to avoid conflicts between accesskeys specified by the web page and shortcuts used by the browser itself, or by the user's screen reader. Needless to say, screen reader users depend heavily on these shortcuts, and a web page that overrides them will create serious, painful mode errors. Some experts deprecate the accesskey attribute, favoring other forms of keyboard navigation around a page instead (see Jukka Korpela, "Using accesskey attribute in HTML forms and links", http://www.cs.tut.fi/~jkorpela/forms/accesskey.html; also "Using Accesskeys - Is it worth it?", http://www.wats.ca/show.php?contentid=32).

Finally, CSS allows specification of audio styles for a screen reader, so that volume and pacing and pitch can be designed by a web page author.

Summary

- Accessibility is a universal problem
- Support alternative input (e.g. keyboards)
- Support alternative output (e.g. screen readers)