Making Eclipse Accessible to People of all Abilities

Kip Harris
hkip@us.ibm.com
IBM Accessibility Center
Agenda

- Why accessibility?
- Assistive technology demonstration
- Accessibility in Eclipse
- Supporting Accessibility when contributing to Eclipse
- Testing
- Challenges for the future
Why accessibility?

Making software usable, regardless of ability or disability

- Text to speech or Braille
- Large fonts, high contrast, and screen magnification
- Accommodation for limited or no use of hands or fingers
- Accommodation for limited range, speed, and strength
- Visual indication of sound
- Captioning
- Adjustable white space between words and lines
- Foreground and background color selection, to improve readability
Accessibility makes sense for many reasons

Legislation
- Federal Rehabilitation Act of 1998 (Section 508)
  - Covers electronic and information technology, effective June 2001
  - Affects all federal agencies and states receiving funds
  - Opportunity / loss of sales due to accessibility
- Americans with Disabilities Act (1992)
- Telecommunications Act (Section 255)
- Others

750 million world wide are disabled
- Over 10% of the market has a disability
- Increasing because of aging population
Need enlargeable fonts and high contrast settings

Font Size

Larger font size

Even larger font size

Large fonts and high contrast

Screen magnifier required when user needs go beyond OS capabilities
**User Needs – Color Blind**

**Need more than color differences to communicate information**

Green: server online  
Red: server offline

Color blind user sees:

Okay to use color to enhance as long as it is not the only means to convey information
User Needs – Blind

**Must use a screen reader and the keyboard**

User presses "alt" key to access menu

- "File submenu press F"
- User presses “right” arrow key
- "Edit submenu press E"

Menu has to be coded in a standard way so screen reader understands and can convey it to the user.
Deaf users need captions and visual equivalents for audio alerts
Hard of hearing users need to increase the volume

Applications can provide volume control  OR  Support the system volume control
Users with limited or no use of their hands need keyboard accessibility features and alternative input methods

<table>
<thead>
<tr>
<th>User Needs – Mobility</th>
</tr>
</thead>
</table>

### Keyboard Accessibility Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mouse Keys</td>
<td>Arrow keys control mouse pointer</td>
</tr>
<tr>
<td>Sticky Keys</td>
<td>[Ctrl] then [F] activates [Ctrl-F] shortcut</td>
</tr>
<tr>
<td>Slow Keys</td>
<td>Ignore short keystrokes</td>
</tr>
<tr>
<td>Repeat Keys</td>
<td>Turn off keystroke repeat</td>
</tr>
<tr>
<td></td>
<td>Adjust delay before repeat begins</td>
</tr>
<tr>
<td></td>
<td>Adjust delay between repeats</td>
</tr>
</tbody>
</table>

### Alternative input hardware devices

- Joy Sticks
- Keyboards
- Switches
- Mouth Sticks

Speech Recognition Software

"Sales for the month of..."
Need speech synthesis, speech input, word prediction, highlighting tools, etc.

Speech Recognition Software

Sales for the month of...

"Sales for the month of ..."

Speech Synthesis

"Please prepare this month's..."

Word Prediction

To: snow-Weaver Andi

Highlighting Tools

Please provide this month's sales report for the team meeting on Tuesday.
Demonstration

Accessible Eclipse application

+ Assistive Technology

(GW Micro “Window-Eyes” screen reader)
Flexible I/O methods and AT interoperability are critical

- Accessibility: Attribute of Information Technology that allows it to be used by people with varying abilities
- Assistive Technology: Specialized IT that allows a user with a particular disability to access Information Technology

### Inaccessible IT
- Static font & color
- Requires mouse
- Graphics only
- Hard to reach controls & latches

### Assistive Technology
- Screen readers
- Magnifiers
- Speech recognition
- Special keyboards & switches

### Accessible IT
- Font & color settings
- Mouse is optional
- Speech recognition
- Text with graphics
- Easy to reach latches & controls

### Assistive Technology
- Screen readers
- Magnifiers
- Speech recognition
- Special keyboards & switches

Standards and APIs: MSAA, JAAPI, standard windows controls
Accessibility features and services are unique to platform

Platform accessibility architecture
- Defines programming interfaces, services, contract between app, OS, AT
- Provides another “view” into the GUI widgets that make up application
- Accessibility architectures are unique to platform (analogous to widget sets)
- Platform also provide accessories for end user

Windows: Microsoft Active Accessibility
- One of the earliest frameworks
- Most mature; Stable code base
- Relatively less expressive
- Future “Longhorn” project introduces new architecture

UNIX / Gnome: Gnome Accessibility Project (GAP)
- More expressive
- Less mature
Eclipse SWT accessibility package provides platform independent support for accessibility.

Platform **independent**
Eclipse plug-in or app
Built using SWT
swt.accessibility package
Platform specific runtime

Platform specific
Assistive technology
e.g.
JAWS for Windows, or
GNOME Gnopernicus

Platform routes msg to app

Platform accessibility libraries and run-times (platform specific)
e.g.
Windows / MSAA or GNOME / GAP

AT sends msg to inspects app
Progression of Eclipse Accessibility features

- Eclipse 2.0, 2.1 – Accessibility on Microsoft Windows
  - Bridge created between SWT and Windows MSAA platform support
  - Eclipse base made accessible
  - Accessibility features added into Eclipse for end user
  - org.eclipse.swt.accessibility package for programmers

- Eclipse 3.0 – Accessibility expanded for UNIX / Gnome
  - Bridge created between SWT and Gnome Accessibility Project interfaces
  - Text specific interfaces added to .swt.accessibility package

- Plug-in developer must still be aware of accessibility guidelines to ensure accessible solution
  - Usually straightforward to design / implement accessible Eclipse contributions
  - Easy to design / implement inaccessible software if working in ignorance
Building accessible Eclipse contributions

- **Keyboard access**
  - Ensure ability to navigate to all user interface elements via keyboard
  - Shortcut keys to improve usability
  - Keyboard equivalents for all functions
    - *Tip:* operate new feature without mouse
  - Do not interfere with existing keyboard functions
  - Eclipse keyboard accessibility features

- **Object information**
  - Visual indicator of current focus
  - Semantic information and text equivalents
    - *May need to use* `.swt.accessibility.AccessibleListener`
  - Associate labels with controls
    - *Tip:* Tab ordering affects label associations
Building accessible Eclipse contributions (con’d)

Display
- Use color as an enhancement, not as only way to convey information
- Custom colors, if used, must be configurable
- Ensure fonts are configurable through either application or system
  - Tip: Use system fonts and colors
- Use accessibility text API’s if writing custom controls
Verifying Accessibility of Eclipse contribution

- Operate app with mouse unplugged
- Display settings
  - Large fonts
  - High contrast
- Screen reader
  - “Window-Eyes” from GW-Micro. “JAWS” from Freedom Scientific
- Platform tools
  - Diagnostic tools provide direct inspection of objects without Assistive Technology
  - “inspect32.exe” on Windows
  - “atpoke” on Gnome
Accessibility challenges

- Challenges for today
  - Application <-> assistive technology interoperability glitches
  - “Scripting” may be needed in some scenarios for truly usable solution

- Challenges for today and tomorrow
  - Accessible presentation of graphical models
  - Fundamental richness / expressiveness of accessibility architectures
Resources

  - www.eclipse.org/articles/index.html
  - A “must read” for anyone tasked with writing a plug-in with a user interface
  - Provides good roadmap and sample code

- IBM Accessibility Center, Developer Guidelines
  - www.ibm.com/able/guidelines.html
  - Programming practices that enable an application for interoperability with an assistive technology
  - Testing criteria and methodology is well documented

- Eclipse org.eclipse.swt.accessibility package documentation
  - Eclipse SDK help, Platform Plug-in Developer Guide > Reference > API Reference > Workbench

- MSDN home page for Microsoft Active Accessibility.
  - MSAA version 1.3 support is generally the interoperability framework for assistive technologies on Windows

- Gnome Accessibility Project
  - developer.gnome.org/projects/gap/