Where does accessibility plug into the graphical desktop stack?

Samuel Thibault
Slides & stuff on http://brl.thefreecat.org/
http://liberte0.org/
Outline

- Introduction to accessibility
- Story of an 'a'
- Input side
- Output side
Color blindness: 8% male, 0.5% female
What is accessibility?

AKA a11y

Usable by people with specific needs

- Blind
- Low vision
- Deaf
- Colorblind
- One-handed
- Cognition (dyslexia, attention disorder, memory, ...)
- Motor disability (Parkinson, ...)
- Elderly

See Accessibility HOWTOs

- You

“Handicap” depends on the situation and is not necessarily permanent
Why making GUI accessible?

(when textmode seems so easier to make accessible)

- A lot of stuff is not available in textmode
  - e.g. real javascript support
- Business applications
- Non-tech people need to get help from non-tech people around
Dedicated software?

- e.g. edbrowse, a blind-oriented editor/browser
- Generally a bad idea!
  - Oriented to just one disability
  - Lack of manpower
    - e.g. Web browser
      - javascript/flash/table/CSS support?
    - e.g. An office suite
      - MSOffice/OpenOffice compatibility?
  - Disabled & non-disabled working together
    - Better use the same software

➔ Better make existing applications accessible
Design principles

• Same software, made accessible
  – Understand each other, get help, etc.

• Synchronized work
  – Just alternate input/output
  – Being able to work together

• Pervasive
  – Shouldn't have to ask for software installation / configuration
Status in a few words

- Text mode is generally quite well accessible
  - But not so well suited to beginners
- Gnome quite accessible
  - Gnome 3 was however almost a restart-from-scratch
- We're late compared to the Windows world
  - We started less than a dozen years ago
  - They started a couple of decades ago
- We're Stone Age compared to the Apple world
  - Really good and integrated support
Story of an 'a'
Input

Still a keycode
i.e. physical position

X client

KeyPress(38)

38(=30+8)

X server

input-evdev

event(KEY_A)

X client

KEY_A=30

input

X server

core

kernel

evdev

atkbd

scancode 0x1e

keyboard
XKB handles turning into keysym, i.e. keyboard cap

Widget eventually has some behavior, e.g. append to text
Not necessarily a screen, actually...

Pixmap very early!
Accessibility in input
Versatility FTW!

Some people can only use

- A keyboard
  - Keyboard shortcuts, move mouse with it, ...
- A joystick
  - Use it as a mouse
- A mouse or a button
  - Use it on a virtual keyboard
- ...
Keyboard layouts

• One-hand?
  - Would need to move the hand a lot
  - Toggle to “mirror” the keyboard layout
  - Not sure where to implement it, and layout details
Basically fine-tuning

- StickyKeys: modifiers get sticky
- MouseKeys: turn keyboard into mouse
- SlowKeys: require key pressed for some time
- RepeatKeys: slow down repeat
- ToggleKeys: audio alert for toggles
- BounceKeys: delay between strokes
  - E.g. Parkinson

Implemented in XKB in X server & X client
Virtual keyboard
Virtual keyboard

XTest injection
Some braille devices have a classical PC keyboard

- No problem
Braille keyboards

Others have a braille keyboard

• 8 keys for the 8 braille dots → 256 patterns

• Only a-z are world-standard, rest:
  – Depends on the language
    • ':' is not the same in English and in French!
  – Depends on the country
    • fr_BE vs fr_CA vs fr_FR
  – Depends on usage
    • French braille revisited several times.
    • VisioBraille devices have their own table.
    • ...
Braille keyboards

But now we have a keysym, not a keycode

- Have to backtranslate...

Typing 'A'

- Find case modifier

Typing 'ô'

- Find dead or combining accent

Remap hack, eww
Typing braille with the PC keyboard

- Turn into dots
- Then turn into text
PC Braille keyboard

Mere XKB layout + imLcFlt + Xcompose

![Diagram showing the flow of input with XKB layout and Braille support]
Braille abbreviations

- “Grade 0” ~= integral ~= litteral
  - One cell for each character
  - 8bit charsets: a mere bijection
    - A → ⠢, B → ⠣, C → ⠥, “ → ⠊, …
  - Unicode and several languages: ambiguity

- “Grade 1/2” ~= abbreviated ~= contracted
  - Common language parts expressed with few cells
    - e.g. “ation” is ⠊⠝
  - Ambiguity
    - “ation” is the same as “N”
PC Braille keyboard

Ibus daemon

![Diagram](image-url)

- `input_signal(b)`
- `KeySym(XK_b)`
- `KeyPress(41)`
- `KeyPress(40)`
- `X client`
How about wayland?

- Is it passing keycodes, keysyms, something else?
- Ideally should allow synthesizing all of them.
- Opportunity to fix all of this?
Accessibility in output
Tinkering with the rendering

- Tweak DPI to get bigger icons & fonts & such
- Xrandr panning support for basic zoom
- Gamma tuning & color inversion
- Screen mirror (!)
- TODO: Gtk3 “perfect” magnification
  - Widget requested to render in a bigger pixmap
But for blind people?

And a **lot** other accessibility possibilities

- Don't try to patch rendering,
- Make applications expose their semantics instead
X accessibility, Mercator 1.0

Diagram:

- X server
  - text
  - Mercator
  - text
  - xedit
Generic methodology

- Application
  - Abstract representation
    - Visual Rendering
  - Screen reader
    - Accessibility bus
      - Registry
- Accessibility device
Story of an 'a', continued
But screen reader also needs reading

I.e. browse the application content

- Get text
- Get parent, children
- ...
Abstract representation

- **Window**
  - **Vertical container**
    - **Menu bar**
      - File Menu
        - Open Menu Item
        - ...
      - ...
    - **Horizontal container**
      - Text area
      - Ok button
Technically speaking

• A lot of applications are already technically accessible
  – Console
  – GTK
  – KDE-Qt4/5 ("Real Soon Now")
  – Acrobat Reader

• A lot are not
  – KDE-Qt3
  – Xt
  – Self-drawn (e.g. xpdf)
In practice

• A lot of technically-accessible applications actually aren't really usable
  – A visually-organized mess of widgets...

<table>
<thead>
<tr>
<th>First name:</th>
<th>Foo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last name:</td>
<td>Bar</td>
</tr>
<tr>
<td>Password:</td>
<td>baz</td>
</tr>
</tbody>
</table>
In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...

  First column
  - Label First Name
  - Label Last Name
  - Label Password

  Second column
  - Text Foo
  - Text Bar
  - Text baz
In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...
    - Label First Name for Text Foo
    - Label Last Name for Text Bar
    - Label Password for Text baz
In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...
    - First column
      - Label First Name
      - Label Last Name
      - Label Password
    - Second column
      - Text Foo
      - Text Bar
      - Text baz
In practice

- A lot of technically-accessible applications actually aren't really usable

  - A visually-organized mess of widgets...

    First column
    - Label First Name
    - Label Last Name
    - Label Password

    Second column
    - Text Foo
    - Text Bar
    - Text baz

  ➔ Screen reader “Script” for each application
Don't try to make applications accessible, just make accessible applications

Quite often just a matter of common sense from the start

Not a reason for not fixing your existing apps of course, it will just be a bit harder :)

Graphical applications

- Design your application **without** gui in mind first
  - Logical order, just like CSS 😊
- Use standard widgets
  - e.g. *labeled* text fields
  - Avoid homemade widgets, or else implement atk yourself for them
  - Always provide alternative textual content for visual content
- Keep it simple!
  - Not only to make screen reading easier, but to make life easier for all users too!
Some pitfalls and advices

(from the accessibility howtos)

- Shouldn't have to use the mouse for anything
- Care of contrasts, configurable colors
- Avoid timing-based actions, or make them configurable
- No 2D organization, logical organization
- Keep it simple and obvious
- ...
Accerciser

Check that the tree of widgets looks sane and is complete

Test it yourself! (GUIs)
• Accessibility HOWTOs
  – Quite old, but still very useful advices
• Gnome Accessibility devel guide
  – For GTK applications
Conclusion

- Accessibility has very diverse X needs
  - Plug at various levels
  - Needs various tweaks
  ➔ We need no regression there!

- Accessibility needs the semantics, not just the rendering
  - Separate form from content