Building an Embedded Linux Prototype

Devin Carraway, Chuck Groom

Blue Mug, Inc.

Overview

- About Blue Mug, Inc.

Overview

- About Blue Mug, Inc.
- Project

Overview

- About Blue Mug, Inc.
- Project
- Hardware Selection

Overview

- About Blue Mug, Inc.
- Project
- Hardware Selection
- Low-level

Overview

- About Blue Mug, Inc.
- Project
- Hardware Selection
- Low-level
- User Interface

Overview

- About Blue Mug, Inc.
- Project
- Hardware Selection
- Low-level
- User Interface
  - UI Design

Overview

- About Blue Mug, Inc.
- Project
- Hardware Selection
- Low-level
- User Interface
  - UI Design
  - Embeddable Linux GUIs

Overview

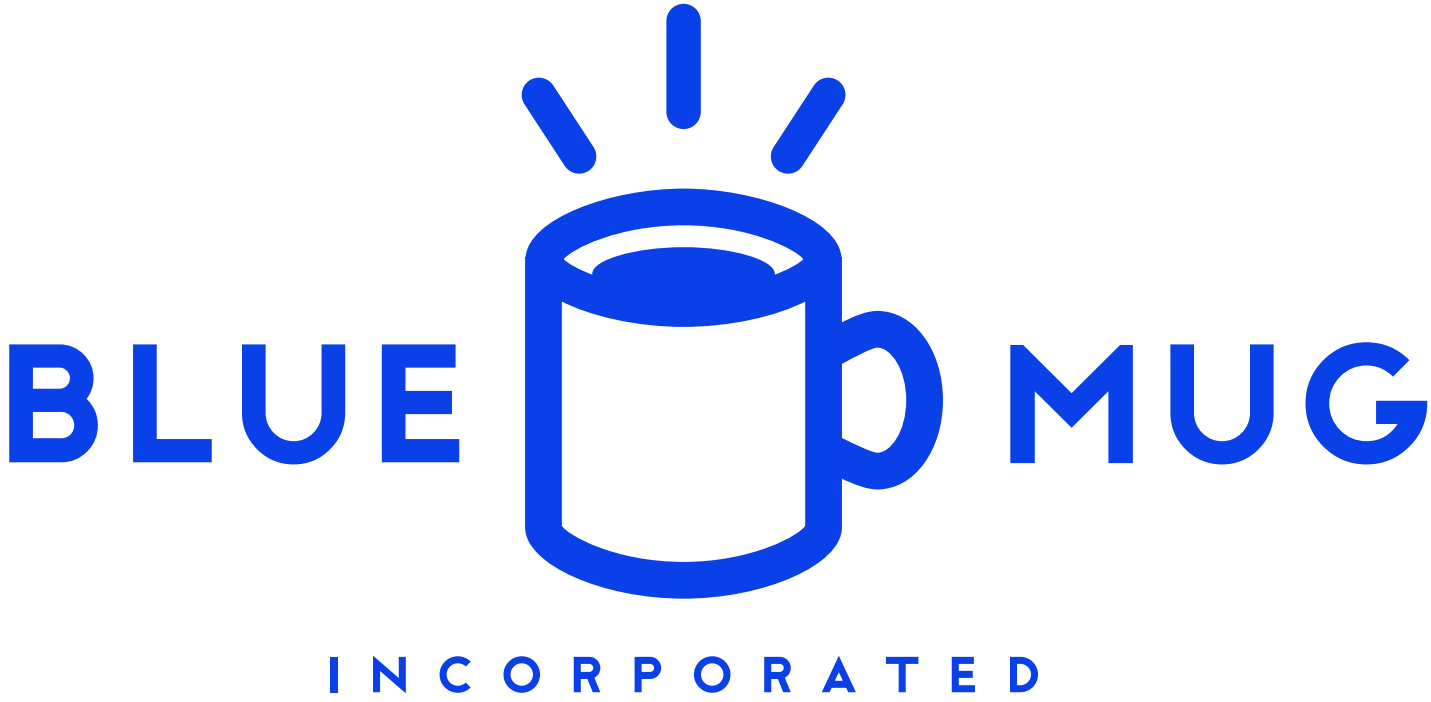
- About Blue Mug, Inc.
- Project
- Hardware Selection
- Low-level
- User Interface
  - UI Design
  - Embeddable Linux GUIs
  - Modifying Gtk+

About Us

Devin and Chuck are project engineers

- We write specs, design products, and write code
- Devin is... Linux master
- Chuck is “that user interface guy”

About Blue Mug, Inc.

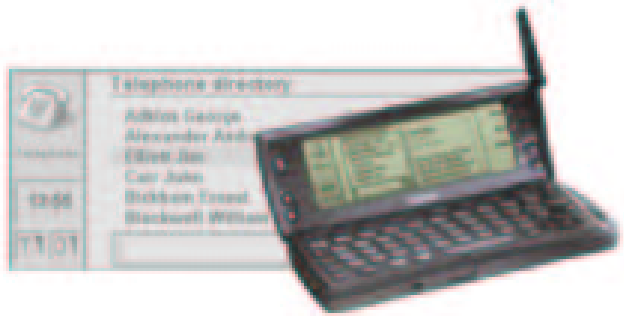


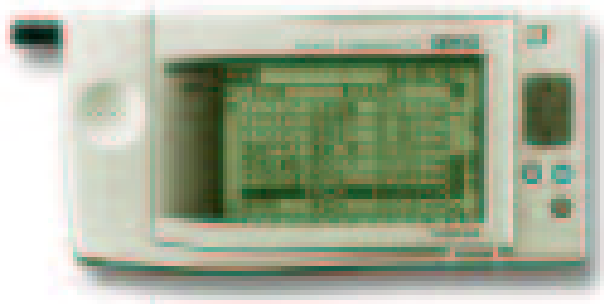
About Blue Mug, Inc.

Blue Mug creates software for mobile devices

About Blue Mug, Inc.

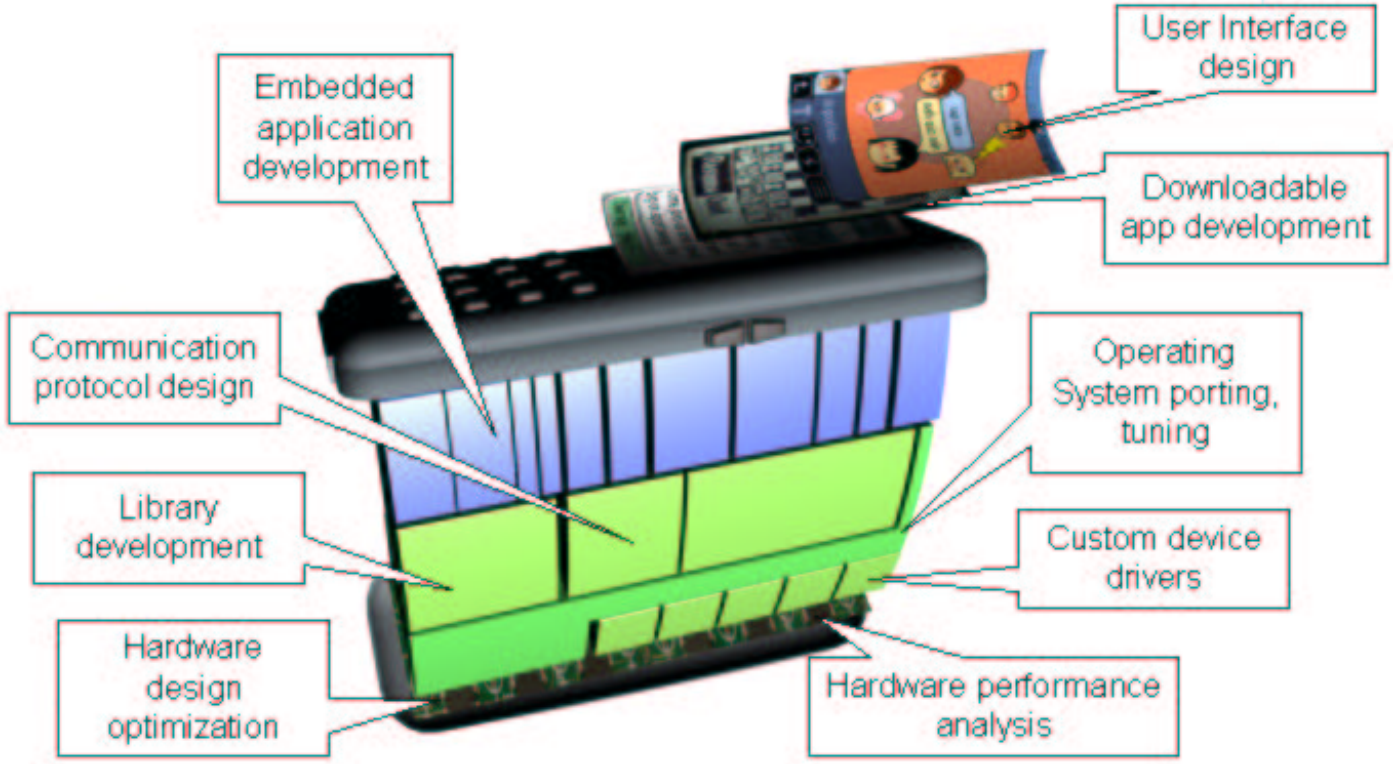
Blue Mug creates software for mobile devices





About Blue Mug, Inc.

Blue Mug creates software for mobile devices



About Blue Mug, Inc.

Blue Mug creates software for mobile devices

- Located in Berkeley
- About 18 employees, 90% engineers
- Founded in 1999 (from Geoworks' Mobile OS Group)

About Blue Mug, Inc.

Blue Mug creates software for mobile devices

We like Linux and embedded Linux, from several points of view:

- Business: free, not a dead-end technology
- Developer: sane platform
- Users: stable, doesn’t suck

About Blue Mug, Inc.

Blue Mug creates software for mobile devices

But we’re not a Linux-only company

- GEOS-SC OS
- PalmOS
- RTOS
- J2ME, BREW
- Small embedded projects

The Project

Our client asked us to create a prototype for a device:

- Low-cost (<$100)

The Project

Our client asked us to create a prototype for a device:

- Low-cost (<$100)
- Soft-key input

The Project

Our client asked us to create a prototype for a device:

- Low-cost (<$100)
- Soft-key input
- Small gray-scale screen

The Project

Our client asked us to create a prototype for a device:

- Low-cost (<$100)
- Soft-key input
- Small gray-scale screen
- Palm-like battery life (22hrs)

The Project

Our client asked us to create a prototype for a device:

- Low-cost (<$100)
- Soft-key input
- Small gray-scale screen
- Palm-like battery life (22hrs)
- Can run simultaneous apps

The Project

Our client asked us to create a prototype for a device:

- Low-cost (<$100)
- Soft-key input
- Small gray-scale screen
- Palm-like battery life (22hrs)
- Can run simultaneous apps
- Multiple access points (modem, PCMCIA for Ethernet, Bluetooth, etc.)

Project Example Mockup

Hardware Selection

Which embeddable system-on-a-chip to use?

Hardware Selection

Which embeddable system-on-a-chip to use?

Considerations:

- Performance
- Price
- Power consumption

Hardware Selection

Which embeddable system-on-a-chip to use?

StrongARM, PPC use too much power, cost too much

Hardware Selection

Which embeddable system-on-a-chip to use?

MIPS, SH are struggling

Hardware Selection

Which embeddable system-on-a-chip to use?

ARM is cheap, low-power, reasonably fast.
We choose the Cirrus Logic EP7211 board.

Hardware Selection

Which embeddable system-on-a-chip to use?

ARM is cheap, low-power, reasonably fast. We choose the Cirrus Logic EP7211 board.

- 75Mhz ARM7

Hardware Selection

Which embeddable system-on-a-chip to use?

ARM is cheap, low-power, reasonably fast. We choose the Cirrus Logic EP7211 board.

- 75Mhz ARM7
- 16Mb Flash, 16Mb RAM

Hardware Selection

Which embeddable system-on-a-chip to use?

ARM is cheap, low-power, reasonably fast. We choose the Cirrus Logic EP7211 board.

- 75Mhz ARM7
- 16Mb Flash, 16Mb RAM
- Low-power (170mw)

Hardware Selection

Which embeddable system-on-a-chip to use?

ARM is cheap, low-power, reasonably fast. We choose the Cirrus Logic EP7211 board.

- 75Mhz ARM7
- 16Mb Flash, 16Mb RAM
- Low-power (170mw)
- Successor to PS7110 used in Psion Series 5, for which there is a Linux port.

System Overview

- Two 8Mb banks of Flash
  - Kernel in one bank
  - Root file system in other bank (mounted read-only)
- /tmp in RAM
- User files, add-on apps in RAM
- No swap!

Size/RAM issues

- Size: 8Mb for all libraries, GUI, windowing system, and apps

Size/RAM issues

- Size: 8Mb for all libraries, GUI, windowing system, and apps
  - JFFS2 and cramfs (compressed file systems) weren’t ready at the time

Size/RAM issues

- Size: 8Mb for all libraries, GUI, windowing system, and apps
  - JFFS2 and cramfs (compressed file systems) weren’t ready at the time
  - Could compile in Thumb (16-bit) instruction set

Size/RAM issues

- Size: 8Mb for all libraries, GUI, windowing system, and apps
  - JFFS2 and cramfs (compressed file systems) weren’t ready at the time
  - Could compile in Thumb (16-bit) instruction set
    - Size-for-speed
    - Tricky; dynamic linking, c library...

Size/RAM issues

- Size: 8Mb for all libraries, GUI, windowing system, and apps
  - JFFS2 and cramfs (compressed file systems) weren’t ready at the time
  - Could compile in Thumb (16-bit) instruction set
    - Size-for-speed
    - Tricky; dynamic linking, c library...
- RAM usage

Size/RAM issues

- Size: 8Mb for all libraries, GUI, windowing system, and apps
  - JFFS2 and cramfs (compressed file systems) weren’t ready at the time
  - Could compile in Thumb (16-bit) instruction set
    - Size-for-speed
    - Tricky; dynamic linking, c library...
- RAM usage
  - Could do XIP from RAM

Size/RAM issues

- Size: 8Mb for all libraries, GUI, windowing system, and apps
  - JFFS2 and cramfs (compressed file systems) weren’t ready at the time
  - Could compile in Thumb (16-bit) instruction set
    - Size-for-speed
    - Tricky; dynamic linking, c library...
- RAM usage
  - Could do XIP from RAM
  - Never ran out of RAM in testing

Out of Memory – What to do?

Low-Level: OOM

Out of Memory – What to do?

Difficult on desktop. Linux kills processes based on CPU usage, run time, and access to privileged I/O resources.

Low-Level: OOM

Out of Memory – What to do?

Difficult on desktop. Linux kills processes based on CPU usage, run time, and access to privileged I/O resources.

Easier on embedded system

Low-Level: OOM

Out of Memory – What to do?

Difficult on desktop. Linux kills processes based on CPU usage, run time, and access to privileged I/O resources.

Easier on embedded system

- Known set of processes (eg. BeOS' "kill the browser" approach)
- Tie into UI to display warning or errors
- Require apps to be aware of low-memory situations

Low-Level: Memory Mapping

EP7211 memory is non-contiguous

Low-Level: Memory Mapping

EP7211 memory is non-contiguous

- Use kernel macros to map between actual and linear presentation of memory

Low-Level: Which C Library?

C library is almost as big as kernel. Which C library to use?

- glibc: GNU C library, the standard

Low-Level: Which C Library?

C library is almost as big as kernel. Which C library to use?

- glibc: GNU C library, the standard
- sglibc: Patched glibc

Low-Level: Which C Library?

C library is almost as big as kernel. Which C library to use?

- glibc: GNU C library, the standard
- sglibc: Patched glibc
- μCLibc: Reduced-size, standard API

Low-Level: Which C Library?

C library is almost as big as kernel. Which C library to use?

- glibc: GNU C library, the standard
- sglibc: Patched glibc
- μCLibc: Reduced-size, standard API
- Diet libc: Reduced-size, breaks API

Low-Level: Which C Library?

C library is almost as big as kernel. Which C library to use?

- glibc: GNU C library, the standard
- sglibc: Patched glibc
- μCLibc: Reduced-size, standard API
- Diet libc: Reduced-size, breaks API

Start with GLibC, move to sglibc.

User Interface: Design Principles

- The user interface (UI) can mean the success or failure of a consumer device

User Interface: Design Principles

- The user interface (UI) can mean the success or failure of a consumer device
- You can’t have a general-purpose mobile device GUI; it must fit device particulars

User Interface: Design Principles

- The user interface (UI) can mean the success or failure of a consumer device
- You can’t have a general-purpose mobile device GUI; it must fit device particulars

Good: Palm UI fits small-screen, stylus-central organizer

User Interface: Design Principles

- The user interface (UI) can mean the success or failure of a consumer device
- You can’t have a general-purpose mobile device GUI; it must fit device particulars

Good: Palm UI fits small-screen, stylus-central organizer

Bad: WinCE UI presents entire desktop interface on small screen

User Interface: User Goals

Always keep the user's goals in mind.

- Mobility = urgency

User Interface: User Goals

Always keep the user's goals in mind.

- Mobility = urgency
- Objective: get job done

User Interface: User Goals

Always keep the user's goals in mind.

- Mobility = urgency
- Objective: get job done
- Technology: avoid unless necessary

User Interface: User Goals

Always keep the user's goals in mind.

- Mobility = urgency
- Objective: get job done
- Technology: avoid unless necessary
- Status notification: don't alert unless problem

User Interface: User Goals

Always keep the user's goals in mind.

- Mobility = urgency
- Objective: get job done
- Technology: avoid unless necessary
- Status notification: don't alert unless problem
- Rich feature set: device feels unpredictable

User Interface: Givens

- Instant response to user interaction

User Interface: Givens

- Instant response to user interaction
- Always-on app model

User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control

User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen

User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen
- Cheap screen

User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen
- Cheap screen
  - Small

User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen
- Cheap screen
  - Small
  - Grays cost power

User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen
- Cheap screen
  - Small
  - Grays cost power
  - Low-contrast

User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen
- Cheap screen
  - Small
  - Grays cost power
  - Low-contrast
- “Walk up and use” interface

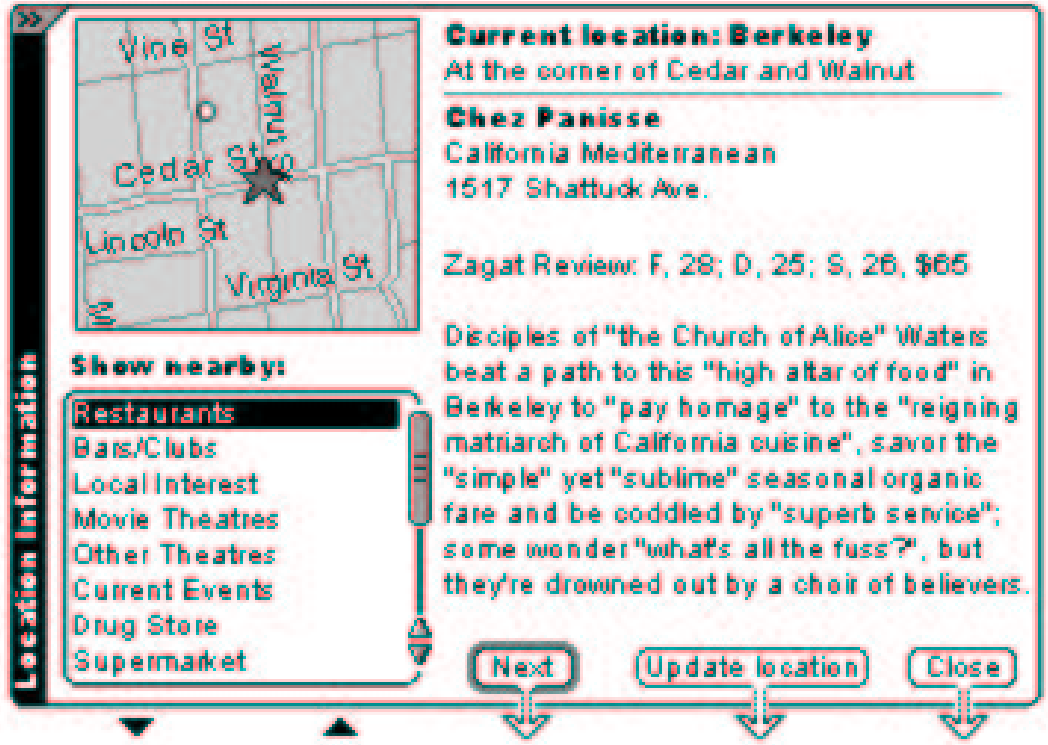
User Interface: Givens

- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen
- Cheap screen
  - Small
  - Grays cost power
  - Low-contrast
- “Walk up and use” interface
  - Borrow desktop elements as needed

User Interface: Givens

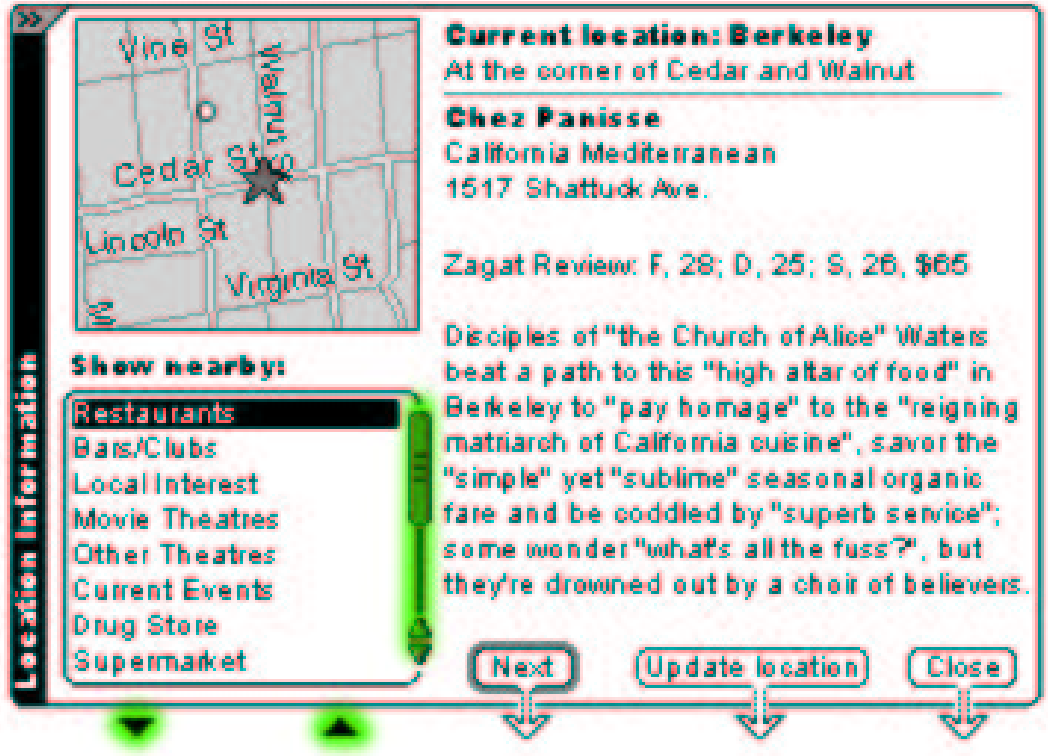
- Instant response to user interaction
- Always-on app model
- Primarily softkey control
- No touchscreen
- Cheap screen
  - Small
  - Grays cost power
  - Low-contrast
- “Walk up and use” interface
  - Borrow desktop elements as needed
  - Limit choices

User Interface: Our Design



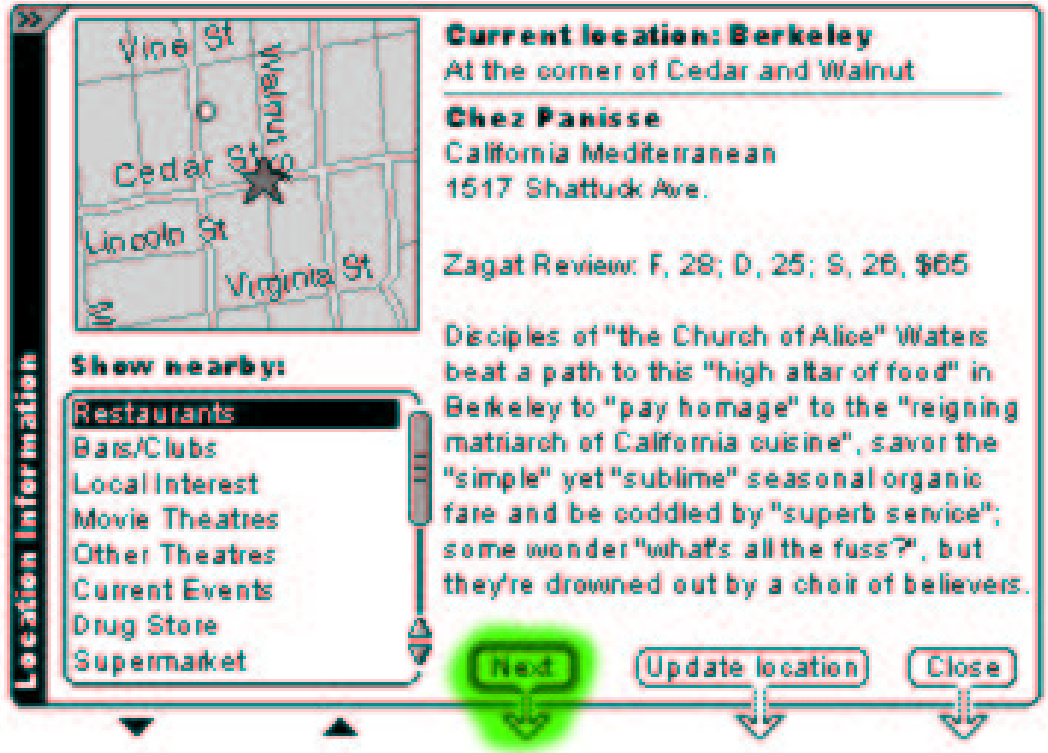
(This is a conceptual mockup)

User Interface: Our Design



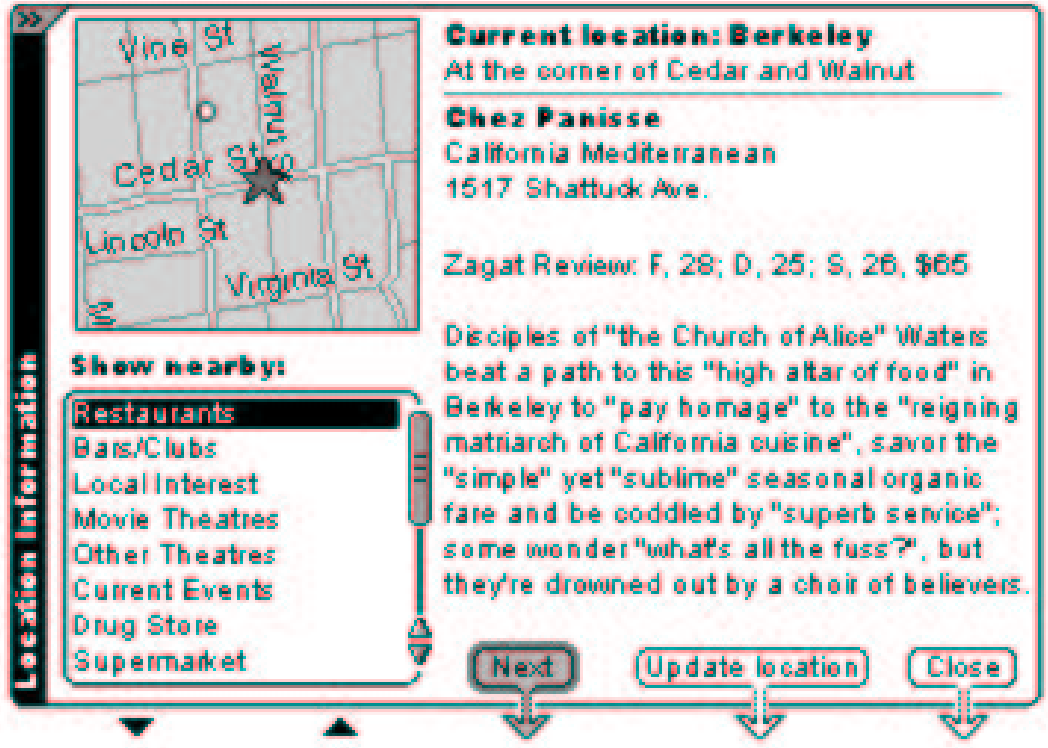
Use desktop GUI widgets with softkey control

User Interface: Our Design



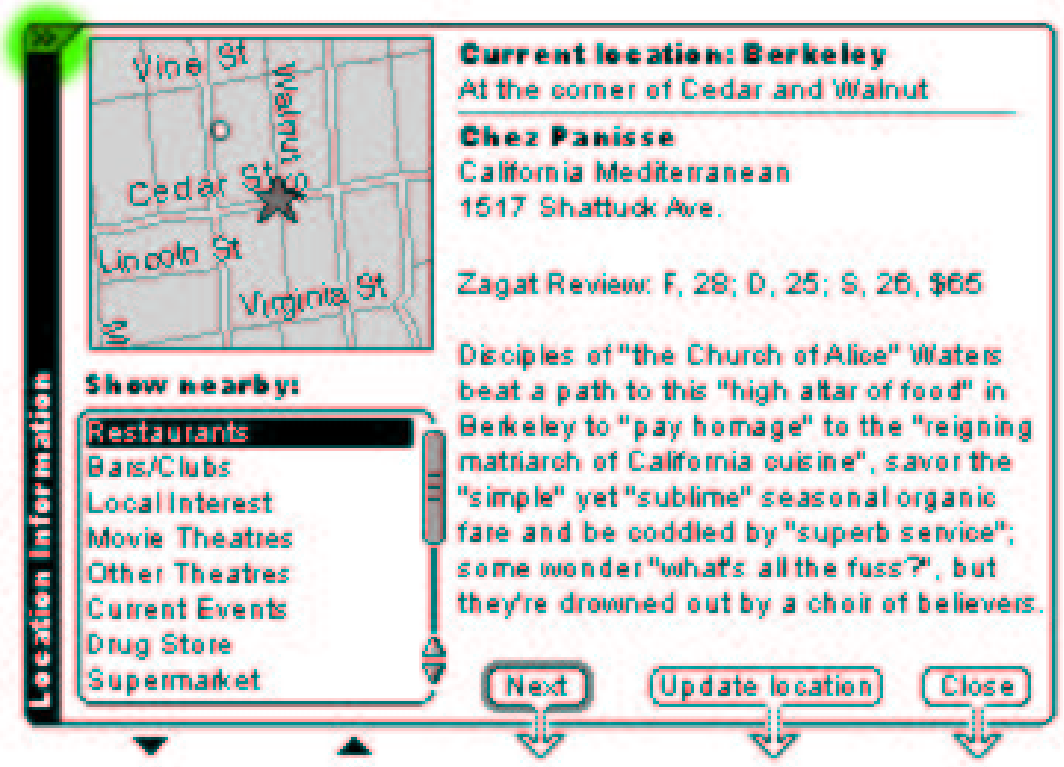
Use desktop GUI widgets with softkey control

User Interface: Our Design



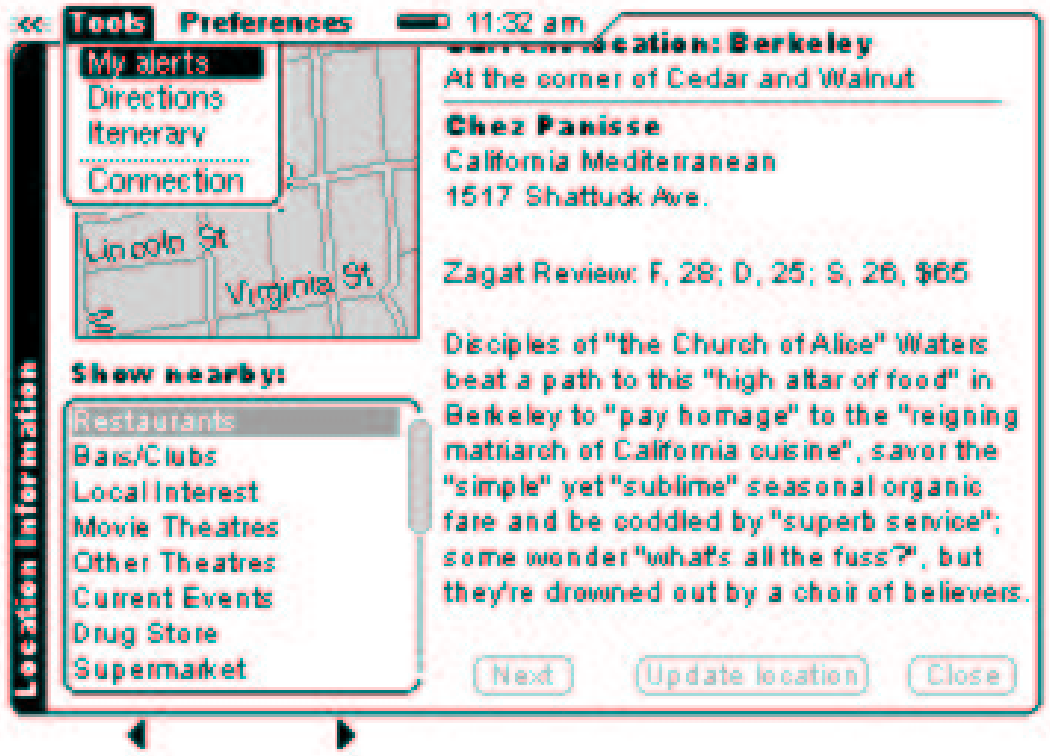
Use desktop GUI widgets with softkey control

User Interface: Our Design



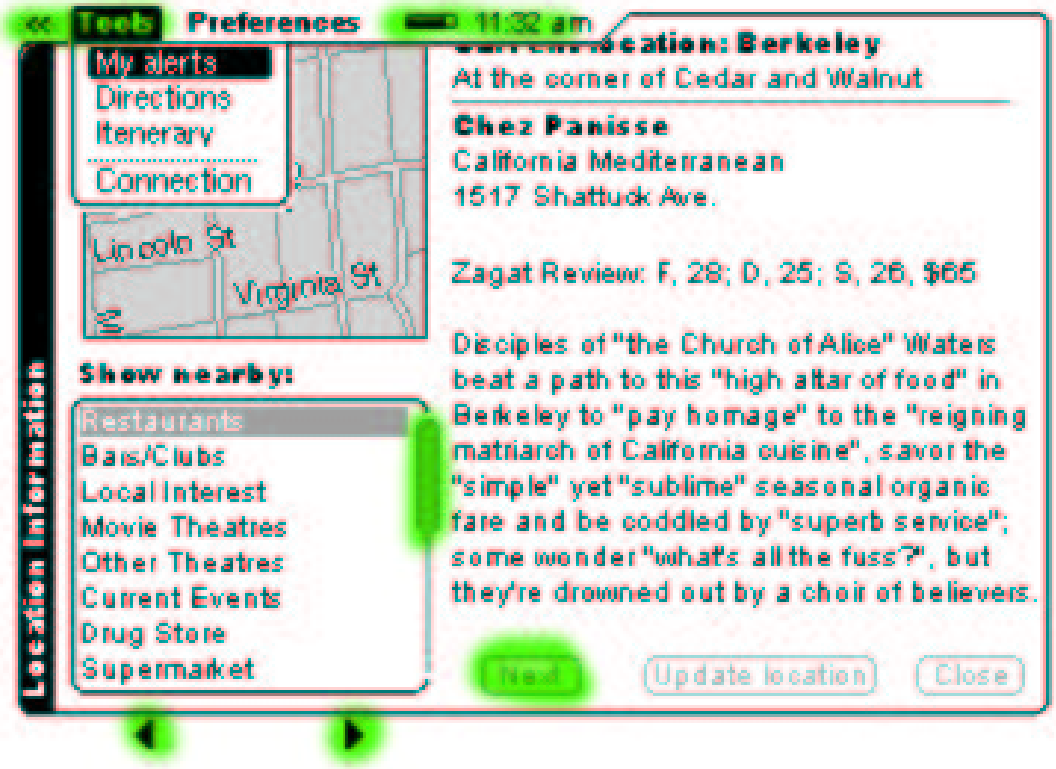
Place options in menu. Hide menu to save screen space, but indicate existence.

User Interface: Our Design



Menu bar includes time and battery.

User Interface: Our Design



Menu is modal and takes control of softkey bar. Other widgets are inactive.

User Interface: Other Elements

Other misc. design elements...

- Softkeys vs. buttons.

User Interface: Other Elements

Other misc. design elements...

- Softkeys vs. buttons.
  - Softkeys can be stand-alone

User Interface: Other Elements

Other misc. design elements...

- Softkeys vs. buttons.
  - Softkeys can be stand-alone
  - Use buttons when action affects pane.

User Interface: Other Elements

Other misc. design elements...

- Softkeys vs. buttons.
  - Softkeys can be stand-alone
  - Use buttons when action affects pane.
- Added “indeterminate” state to radio buttons, check boxes

User Interface: Other Elements

Other misc. design elements...

- Softkeys vs. buttons.
  - Softkeys can be stand-alone
  - Use buttons when action affects pane.
- Added “indeterminate” state to radio buttons, check boxes
- Dialogs

User Interface: Other Elements

Other misc. design elements...

- Softkeys vs. buttons.
  - Softkeys can be stand-alone
  - Use buttons when action affects pane.
- Added “indeterminate” state to radio buttons, check boxes
- Dialogs
- When we launch an app, display “zoomy rectangle”

User Interface: Embeddable Linux GUIs

How do we implement this interface?

User Interface: Embeddable Linux GUIs

How do we implement this interface?

$\rightarrow$ Tweak existing UI. There are many possible Linux UIs to pick between.

User Interface: Embeddable Linux GUIs

Criteria:

- Completeness
- Size
- Multiple apps can access framebuffer
- Language (C, C++)
- License

User Interface: Embeddable Linux GUIs

- Gtk+
- Qt/e
- FLTK
- OpenGUI
- MiniGUI
- PicoGUI
- Microwindows
- ...

User Interface: Embeddable Linux GUIs

Narrowed to Gtk+ or Qt/e

User Interface: Embeddable Linux GUIs

Qt

- KDE Desktop
- Developed by TrollTech
- C++ framework
- Qt/E is reduced, runs on framebuffer
- QTopia app infrastructure
- Pain to compile
- Dual-license

User Interface: Embeddable Linux GUIs

Gtk+

- GNOME Desktop
- Open source project
- C
- Developed on X; also Gtk+/fb
- LGPL

User Interface: Embeddable Linux GUIs

Decided on Gtk+ running on X

User Interface: Embeddable Linux GUIs

Decided on Gtk+ running on X

X Windows! Eek!

- Client-server windowing system
- Network-transparent
- 20 years old
- Widely regarded as bloated and archaic

User Interface: Embeddable Linux GUIs

Decided on Gtk+ running on X

We like X

- X is stable
- Network-transparency is helpful
- TinyX

User Interface: Embeddable Linux GUIs

Decided on Gtk+ running on X

Modified AEWM window manager

- Vertical title bars
- Inter-app communication
- Application-level awareness of modal dialogs

User Interface: Modifying Gtk+

- Trim unnecessary widgets (eg. file dialog, color selection)

User Interface: Modifying Gtk+

- Trim unnecessary widgets (eg. file dialog, color selection)
- Widget sizing

User Interface: Modifying Gtk+

- Trim unnecessary widgets (eg. file dialog, color selection)
- Widget sizing
- Widget drawing

User Interface: Modifying Gtk+

- Trim unnecessary widgets (eg. file dialog, color selection)
- Widget sizing
- Widget drawing
- GtkWindow (*)

User Interface: Modifying Gtk+

- Trim unnecessary widgets (eg. file dialog, color selection)
- Widget sizing
- Widget drawing
- GtkWindow (*)
- Font management (*)

User Interface: Modifying Gtk+

- Trim unnecessary widgets (eg. file dialog, color selection)
- Widget sizing
- Widget drawing
- GtkWindow (*)
- Font management (*)

Changes in-place, not sub-classed

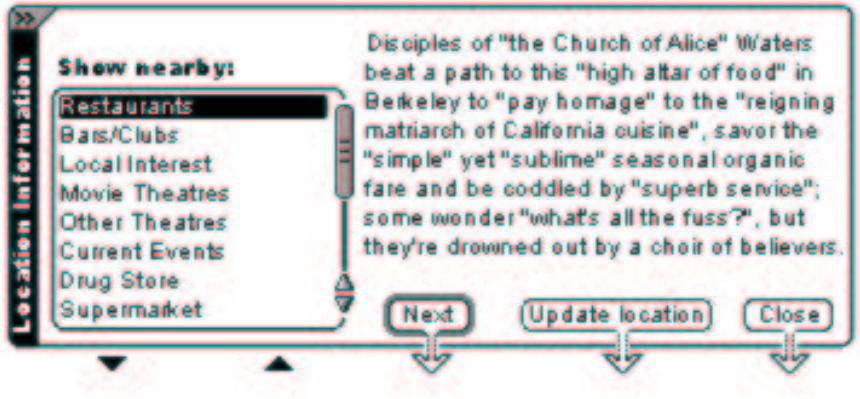
User Interface: Modifying Gtk+

- Trim unnecessary widgets (eg. file dialog, color selection)
- Widget sizing
- Widget drawing
- GtkWindow (*)
- Font management (*)

2.9Mb footprint for Gtk+/X; this could be reduced to 2.4Mb.

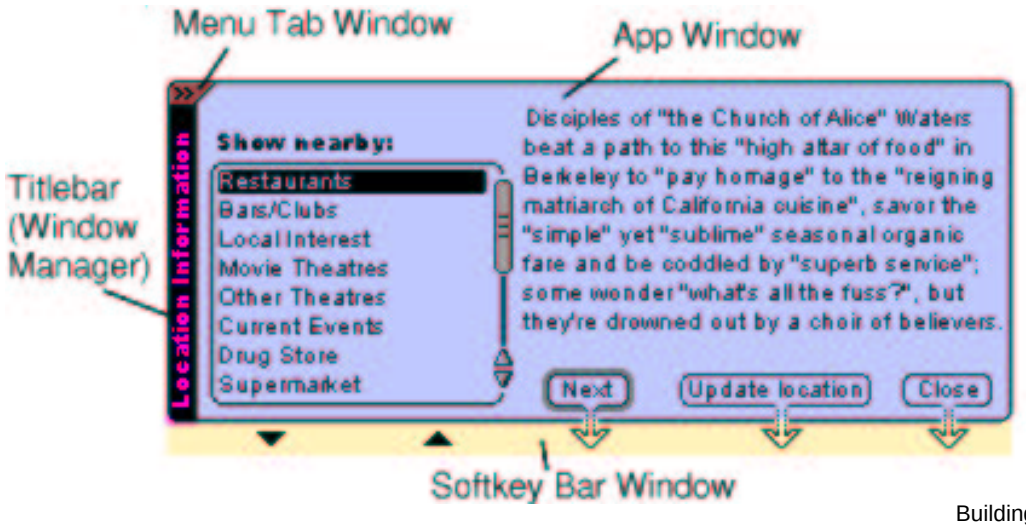
User Interface: GtkWindow

- Application window talks to window manager



User Interface: GtkWindow

- Application window talks to window manager
- Application window has-a softkey bar

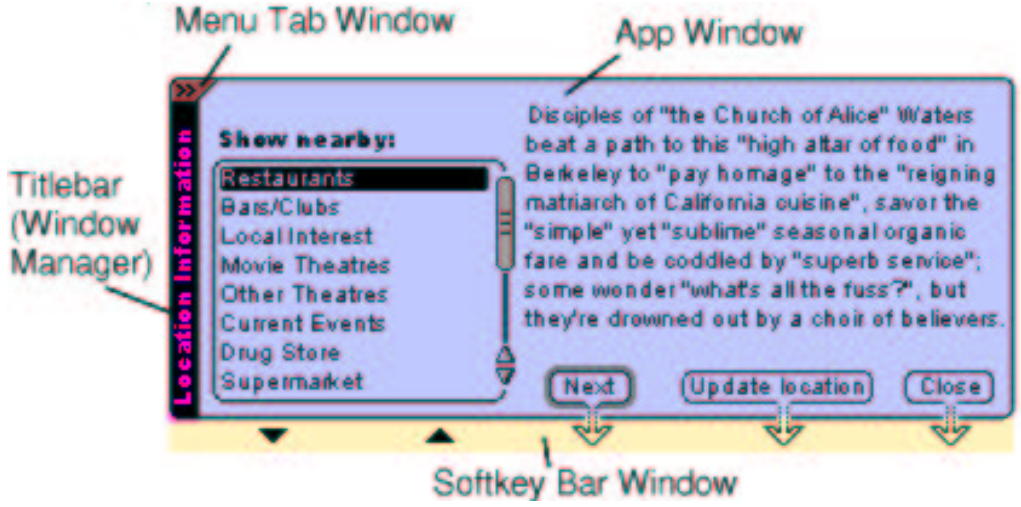


User Interface: GtkWindow

- Application window talks to window manager
- Application window has-a softkey bar
  - Not nested within widget

User Interface: GtkWindow

- Application window talks to window manager
- Application window has-a softkey bar
  - Not nested within widget
- API to register softkeys on application window



User Interface: GtkWindow

- Application window talks to window manager
- Application window has-a softkey bar
  - Not nested within widget
- API to register softkeys on application window
- Scrolling full-screen window

User Interface: GtkWindow

- Application window talks to window manager
- Application window has-a softkey bar
  - Not nested within widget
- API to register softkeys on application window
- Scrolling full-screen window

User Interface: Font

To change a font in stock Gtk+:

- Clone widget's `GtkStyle`
- Load a new X font, such as `-adobe-helvetica-bold-r-normal-12-*-*-*-p-*-iso8859-1`

User Interface: Font

To change a font in stock Gtk+:

- Clone widget's `GtkStyle`
- Load a new X font, such as `-adobe-helvetica-bold-r-normal-12-*-*-*-p-*-iso8859-1`

GtkStyle is fairly big, so this is expensive. And the developer has to know the specific font name.

User Interface: Font

We wrote API for requesting fonts by attribute relative to the base font.

User Interface: Font

We wrote API for requesting fonts by attribute relative to the base font.

```
gtk_widget_set_font_bold (widget, TRUE);
gtk_widget_set_font_enlarge (widget, 1);
```

User Interface: Font

We wrote API for requesting fonts by attribute relative to the base font.

```
gtk_widget_set_font_bold (widget, TRUE);
gtk_widget_set_font_enlarge (widget, 1);
```

We added a `GdkFont * font` to `GtkWidget`. Use `widget->font` if possible, otherwise use `widget->style->font`

User Interface: Font

We wrote API for requesting fonts by attribute relative to the base font.

```
gtk_widget_set_font_bold (widget, TRUE);
gtk_widget_set_font_enlarge (widget, 1);
```

We added a `GdkFont * font` to `GtkWidget`. Use `widget->font` if possible, otherwise use `widget->style->font`

You can request font changes even before Gtk+ knows the base font.

User Interface: Performance

- Slow launch times

User Interface: Performance

- Slow launch times
  - 2.4 seconds for most complicated app
  - Memory bandwidth bottleneck
  - For now, display eye candy when app is launched
  - In future, predictively launch applications

User Interface: Performance

- Slow launch times
- Loading pixmaps

User Interface: Performance

- Slow launch times
- Loading pixmaps
  - XPM format is bulky
  - Gtk+’s XPM parser sucks
  - Hack parser
  - Hand post-rendered pixmaps to X server

User Interface: Performance

- Slow launch times
- Loading pixmaps
- Floating point calculations

User Interface: Performance

- Slow launch times
- Loading pixmaps
- Floating point calculations
  - Floating point calculations are expensive on ARM
  - Gtk+ uses floating points for widget positioning
  - Integer math positioning gives a 3-12% speedup

Conclusion

- We’re happy with our choice of Gtk+/X

Conclusion

- We’re happy with our choice of Gtk+/X
- OSS made this project possible

Conclusion

- We’re happy with our choice of Gtk+/X
- OSS made this project possible
- Demo!

Conclusion

- We’re happy with our choice of Gtk+/X
- OSS made this project possible
- Demo!
- Questions?