eekboard
a virtual keyboard for GNOME

Daiki Ueno
ueno@unixuser.org
Red Hat, i18n team
Virtual keyboards (vkbds)

- Caribou
- matchbox-keyboard
- onBoard
- fvkbd
- GOK
- iok
- Florence
- CellWriter
- xvkbd
- kvkbd
Why everyone writes his own vkbd, while physical keyboard is merely a switch array?
Why?

• Geeks are all keyboard maniac 😊
• A wide variety of demands on vkbds
• Vkbds work closely with other desktop technologies, whose functions sometimes overlap each other
  → Many design decisions depend on how vkbds interact with those technologies
Vkbd use cases

- Kiosk
- Tablet PC
- Mobile phone without keyboard
- Typing tutor
- Unicode character input
Vkbd related desktop technologies

- GNOME Accessibility
  - Convert UI events to ones helpful for disabled users

- Input methods
  - Convert UI events to (typically multilingual) text

- Keyboard layout configuration
  - Convert physical key events to logical symbols, based on users' preferences
    - e.g. using generic US layout keyboard as Dvorak layout
Their functional territories sometimes conflict.
Vkbd related desktop technologies: contradictions

• Who activates vkbd?
  • GNOME Accessibility - Caribou
  • Input methods - ibus-input-pad, scim-panel-vkb-gtk

• How to deliver key events?
  • send them as X events
  • send translated symbols directly to input methods

• How can a vkbd intercept key events?
  • If vkbd should react to physical key events, how to capture them?
Ideas behind eekboard

• Throw away the idea of creating a *single mighty vkbd* that meets all the requirements
  • Instead, start from a GUI library to create keyboard-like user interfaces

• Decouple the GUI from accessibility, input methods, and keyboard layout configuration
eekboard

- libeek
  - easy embedded keyboard
  - A library to create keyboard like UI
  - GUI toolkit agnostic API
  - Can read keyboard layout configuration from various sources

- eekboard
  - A sample vkbd implemented with libeek
A keyboard is a tree of elements.
/* Create a GTK+ keyboard. */
keyboard = eek_gtk_keyboard_new();

/* Create a section in the keyboard. */
section = eek_keyboard_create_section (keyboard);

/* Add a row in the section. */
eek_section_add_row (section, 10, ...);

/* Create keys in the section. */
key1 = eek_section_create_key (section, 0, 0);

/* Obtain actual GTK+ widget. */
widget = eek_gtk_keyboard_get_widget (keyboard);
Supported GUI widgets

- ClutterActor
- GtkDrawingArea
  - Code borrowed from libgnomemekbd
- GTK+ button
Keyboard layout configuration

/* Create a keyboard layout configuration using XKB. */

layout = eek_xkb_layout_new();
eek_xkb_layout_set_names_full (layout,
    "symbols", "pc+us+in(ben)",
    "geometry", "kinesis",
    -1);

/* Apply the layout to the keyboard. This will populate sections/keys in keyboard. */
eek_keyboard_set_layout (keyboard, layout);
Supported keyboard layout configuration

- **XKB**
  - Consists of 3 components
    - Keycodes – physical key IDs
    - Symbols – mapping from keycodes to logical symbols
    - Geometry – appearances of keyboard
  - libxklavier wrapper makes it easier to customize by
    - Model
    - Country
    - Language

- **XML layout files**
Put it all together

/* Create a keyboard layout configuration using libxklavier. */
layout = eek_xkl_layout_new();

/* Create a keyboard element implemented as ClutterActor. */
keyboard = eek_clutter_keyboard_new();

/* Apply the layout to the keyboard. */
eek_keyboard_set_layout(keyboard, layout);

/* Convert keyboard into ClutterActor. */
clutter_group_add(CLUTTER_GROUP(stage),
eek_clutter_keyboard_get_actor(EEK_CLUTTER_KEYBOARD(keyboard)));
So how about event handling?

/* Find a key element in the logical keyboard. */
key = eek_keyboard_find_key_by_keycode (keyboard, 0x38);
g_signal_connect (key, "pressed", on_a_pressed);
How about modifiers?

Each key is assigned a matrix of symbols

```
# あ
3 あ
```

Not limited to 2x2

```c
/* Assign symbol matrix to a key. */
eek_key_set_keysyms (key, keysyms,
    num_groups, num_levels);

/* Set group/level of the entire keyboard. */
eek_keyboard_set_keysym_index (keyboard, group, level);
```
eekboard: a sample vkbd

• Startup
  • “tap” on any editable widget via a11y, or
  • invoke the command directly
• Layout can be changed from menu
• Typing monitor
  • Trap all key events and act as a typing monitor
eekboard: demo

Standalone

Popup
Things to come...

- CSS based theme support
- Flick input
- Multi touch
- Rewrite eekboard in Vala
  - Currently it is written in C
  - libeek Vala binding is already available
Questions or Comments?

http://ueno.github.com/eekboard/

“magic mushrooms” on page #2 is © lovejanine, cc-by-nc 2.0
“Cat Fight!” on page #7 is © privatenobby, cc-by-nc-sa 2.0
The rest of the slide materials including “thai typewriter” photo are © Daiki Ueno, cc-by-nc-sa 2.0