Input Methods in Plasma 5

— Modern and global text input —

Eike Hein • 2017
Introduction
About me

- KDE user since ~1999
- KDE developer since ~2005
- Core Plasma Desktop developer
  - Taskbar, desktop icons, folder widgets, launcher menus, ...
- Apps shenanigans: Konversation, Yakuake, ...
- Based in Seoul, South Korea
Talk outline

- Intro to Input Methods
- Demos
- Technology
- Plasma Desktop
- Plasma Mobile
- Q & A
Input Methods
What are Input Methods?

- Conversion of input events into text
  - Input events: Key presses, screen taps, gestures, more complex interactions
  - Divorcing output from input
- Stateful: Composing output in multiple steps
- Rich UI feedback
Who uses Input Methods?

- Use cases: Essential to communication vs. assistive / enhancing

- >20% of the world's population require an IM to write their native language: Chinese, Japanese, Korean, ...
  - Increasing % of IM users among computer users as more come online
  - Underrepresented among users of open systems: Growth opportunity

- Completions, word suggestions, emoji palettes are IMs
  - So far few users on the desktop

- It's a chicken and egg thing on both accounts
Demos
Writing Korean

- Alphabet, but grouping letters into character-wide syllabic blocks
- Stateful multi-step composition of output
- Complex interaction between client application and IM
  - Replacing existing text with each keypress: 
    ᵒ (h) → ᵃ (ha) → ᵃ (han)
  - Formatting hints provided by IM
Writing Chinese

- Logographic writing means thousands of characters
  - Problem becomes tractable with IM

- Output candidate selection
  - Interactive UI provided to applications externally, system service

- The many ways to write Chinese
  - Alternatives to Pinyin; handwriting
Typing Emoji

- Language-agnostic
  - Seem familiar? Logographic writing, too
Word completions & spell-checking

- Language-agnostic
- Learns with use
- Staple of mobile
  - Overcoming a handicap through IM
Technology
Input Method architecture (Desktop)

- Common case: Many actors, lots of IPC
  - Hardware $\rightarrow$ Kernel $\rightarrow$ Windowing System $\rightarrow$ Application
  - Application $\rightarrow$ IM daemon $\rightarrow$ IM plugin
  - IM plugin $\leftrightarrow$ UI frontend ("panel")
  - IM $\rightarrow$ Application $\rightarrow$ UI control

- IM daemon: Manages IM plugins, provides config & marshalling infra

- IM panel: Provided by desktop environment (e.g. Plasma)
Naming the players (Desktop)

- IM daemons: ibus, fcitx, scim
  - Plasma supports all of them

- Qt: Input Context plugins
  - Interface with IM daemon vs. in-process IM
  - In-tree vs. out of tree
  - QInputMethodEvent

- Mobile is its own party
Input Method community

- Wide problem domain
  - Many writing systems to cover, some not yet or poorly supported
  - Existing Input Methods are not "done" yet

- Manpower shortages
  - Few developers, in some cases we rely on single experts
  - Yet we make them duplicate their efforts
  - Community onboarding: Language barrier can be a big problem

- KDE.org is part of the community: Qt, fcitx
Plasma Desktop
Input Method Panel widget

Korean (ibus-hangul)

Chinese (ibus-pinyin)
Input Method Panel widget

- Provides access to Input Method configuration
  - Both daemon and active plugin
- Displays active Input Method controls
- Provides plugins with host UI (popups)
- Multi-backend: Supports ibus, fcitx and scim
Recent work

- Moved Input Method Panel from addons to core (Plasma 5.6)
  - Allowed us to start auto-adding it to the panel by locale on first logon

- Improved ibus support (Plasma 5.10)
  - Auto-starting panel after daemon launches

- KDE Neon Korean Developer Edition (Fall 2016)
  - Made to coincide with KDE's 20th birthday party in Seoul, South Korea
  - Serves as integration test case, based on Dev Edition git stable
KDE Neon Korean Edition
KDE's 20th Birthday in Seoul
Pain points

- Expert knowledge required for initial setup
  - For both installation and configuration

- Config UI is hard to discover and not where users expect to find it
  - And it's not ours

- System Settings both incomplete and redundant
  - Keyboard layouts are but a part of the whole
  - Settings stop working correctly when IM is used

- Input Method Panel competes with System Tray
System Settings: Keyboard Layouts

![Keyboard Hardware and Layout settings](image)

- **Layout Indicator**: Options include Show layout indicator, Show for single layout, Show flag, Show label, Show label on flag.
- **Switching Policy**: Options include Global, Desktop, Application, Window.
- **Shortcuts for Switching Layout**: Main shortcuts: None, 2nd level shortcuts: None, Alternative shortcut: Crtl+Alt+K.

**Keyboard Layouts**

<table>
<thead>
<tr>
<th>Layout</th>
<th>Variant</th>
<th>Label</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>kr</td>
<td>Korean</td>
<td>Korean (ISO/104 key compatible)</td>
<td>kr</td>
</tr>
</tbody>
</table>

**Options**

- Add, Remove, Move Up, Move Down, Preview.
Pain points

- Expert knowledge required for initial setup
  - For both installation and configuration

- Config UI is hard to discover and not where users expect to find it
  - And it's not ours

- System Settings both incomplete and redundant
  - Keyboard layouts are but a part of the whole
  - Settings stop working correctly when IM is used

- Input Method Panel competes with System Tray
System Tray vs. Input Method Panel

System Tray

Input Method Panel
Solutions (read: todos)

- Input Methods always-on
  - Force distros' hands, guarantee availability
  - Can take different forms (build time deps vs. runtime deps)

- Revamp config UI
  - From managing keyboard layouts to managing input languages

- Unify keyboard layout indicator and Input Method Panel

- Integrate Input Method Panel with System Tray
  - Fix theming (icons)
Keyboard Layout Indicator

kr - Korean (101/104 key compatible)
us - English (US)
Configure Layouts...

kr Seoul
Solutions (read: todos)

- Input Methods always-on
  - Force distros' hands, guarantee availability
  - Can take different forms (build time deps vs. runtime deps)

- Revamp config UI
  - From managing keyboard layouts to managing input languages

- Unify keyboard layout indicator and Input Method Panel

- Integrate Input Method Panel with System Tray
Plasma Mobile
Mobile stacks

- The screen as the keyboard
  - Additional interaction patterns: Swipe gestures, etc.
- Oldschool: On-screen keyboards as key event generators
- Maliit
- Qt Virtual Keyboard
  - In-process: Input context plugin
Mobile stacks

- The screen as the keyboard
  - Additional interaction patterns: Swipe gestures, etc.

- Oldschool: On-screen keyboards as key event generators

- Maliit

- Qt Virtual Keyboard
  - In-process: Input context plugin
Qt Virtual Keyboard

00:20
Thursday, 11 May 2017

Password
LOGIN

We're from Barcelona.
Thanks for using.

British English
Pain points

- Duplication of effort vs. desktop stacks
  - Not achieving feature parity (both ways)
  - API explosion: Spreading the community even thinner
  - Bad reasons we don't share: Licensing, ...

- Challenges to convergence
  - Jarring behavior differences between input devices
  - No state synchronization
  - Poor UI integration and no consistency
Solutions (more todos)

• Extend Qt Virtual Keyboard?
  • Wrap existing IM system
  • Maybe fcixt

• Reuse Input Method Panel widget in keyboard tray
And finally ...
Why we should care

- Inclusivity is one of our core values
  - Says https://manifesto.kde.org

- Expanding our reach helps us achieve our goals

- Engineering is about facilitating culture and building civilization
  - Communication is essential to culture and civilization
  - Language is essential to communication