Nested Window Management and Task-Oriented User Interfaces

Kevin Vella and Alan Zammit

QDWM is an experimental window manager for X11 [2] which distinguishes itself with the ability to run in a resizeable window managed by an existing X11 window manager, several times over. This abstract touches upon its implementation, related work and emerging opportunities for research.

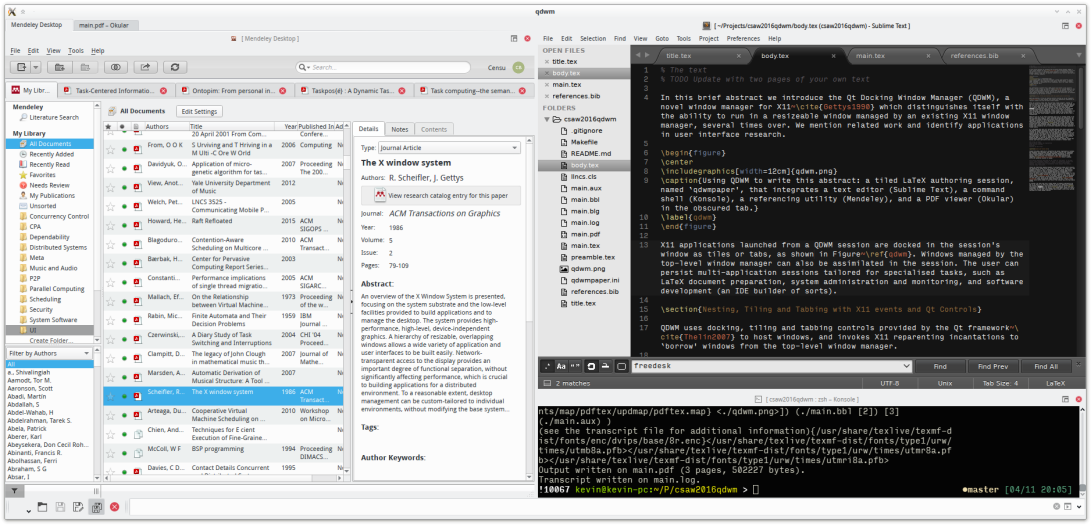

Fig. 1. Using QDWM to write this abstract in KDE Plasma Desktop: a tiled LaTeX session window that integrates a text editor (Sublime Text), a command shell (Konsole), a referencing tool (Mendeley), and a PDF viewer (Okular) in the obscured tab.

X11 windows belonging to different applications can be docked in a QDWM session window as tiles or tabs, as shown in Figure 1. Moreover, multiple QDWM sessions can coexist, each in its own resizeable window. The user can persist multi-application sessions tailored for specialised tasks, such as LaTeX document preparation, system administration and monitoring, and software development (an IDE builder of sorts).

Tiling and Tabbing Windows with X11 and Qt

QDWM uses the Qt [6] framework's resizeable and draggable docking, tiling and tabbing controls, and invokes X11 reparenting incantations to 'borrow' windows from the top-level window manager. Reparenting hides the target window's border decorations and desktop task list entries, and permits it to assume the shape and visibility of its host Qt control.

While any existing window under the top-level window manager's control can be docked to or undocked from the QDWM session through a drop-down selector, the user can also launch new applications to dock through an integrated launcher bar. As windows are subsumed by the top-level window manager upon creation, QDWM snoops on its window list using X.org and freedesktop.org protocols until all the new windows to be assimilated have been identified and reparented[1].

QDWM also acts as an X11 event forwarding proxy for all windows docked in the session. Some event types are simply forwarded between the outside world and one or more of the session's docked windows, while others require particular attention. For instance, coordinates in

[1] Owing to a wide range of possible behaviours (multiple child processes, splash screens, etc.) this procedure can time out for certain applications, in which case the new window remains under the top-level window manager's control and the user can dock it interactively.

DnD (drag-and-drop) protocol events received by the QDWM session window are used to determine the ultimate target window, and routed with adjusted coordinates[2]. At times state needs to be persisted across proxy invocations so as to correctly transform and forward sequences of related events.

It is straightforward to build QDWM sessions that combine application and document windows pertaining to specific user tasks. Since a user task's lifetime can extend indefinitely, but X11 window IDs are transient, session persistence is automated through INI files that describe the tiling and tabbing geometry, the command line for relaunching each application, and additional information that could help in identifying and docking the windows created by each application. Even though it is possible to tweak these parameters as desired by editing the session's INI file, a more sophisticated mechanism is saught to reliably capture, persist and migrate full user-level application state such as document or browser tabs.

Related and Future Work

QDWM aims to streamline the organisation of windows and documents into persistent user tasks through familiar metaphors and lightweight mechanisms. While task-oriented computing has been studied in various forms [1, 4, 3], nested window management could facilitate the seamless integration of intelligent user-task classifiers within existing open source desktop environments.

Though the desktop metaphor of overlapping windows still prevails, the popularity of window tiling is rising[3]. Surprisingly, the two modalities have seldom been considered in combination. QDWM uses nesting to provide the combined functionality in a largely desktop-agnostic X11 implementation, but it could be incorporated into every top-level window manager instead. In this spirit, BeOS's successor Haiku has integrated stacking (tabbing) and tiling into its native window manager [7], but its equivalent is absent on other platforms. WindowScape [5] groups miniaturized window thumbnails by task. Some environments allocate an entire desktop for each user task, which can be cumbersome. Macros which attempt to keep tiled windows aligned have also been shared online.

In closing, one notes that the X Window System is gradually approaching the end of its reign as it nears four decades of service on UNIX/Linux desktops. The contenders in the ensuing succession race are Wayland (from the freedesktop.org community) and Mir (from Canonical Ltd.), both under intensive development. Either way, design decisions that would impact the ability to reparent windows present a non-trivial porting challenge for the nested window manager.

References

1. M. S. Bernstein, J. Shrager, and T. Winograd. Taskposé: exploring fluid boundaries in an associative window visualization. *Symposium on User Interface Software and Technology*, page 3, 2008.
2. J. Gettys, P. L. Karlton, and S. McGregor. The X window system, version 11. *Software: Practice and Experience*, 20(S2):S35—-S67, 1990.
3. M. Kersten and G. C. Murphy. Reducing friction for knowledge workers with task context. *AI Magazine*, 36(2):33–42, 2015.
4. J. Shen et al. Detecting and correcting user activity switches: Algorithms and interfaces. *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 117–126, 2009.
5. C. Tashman and W. K. Edwards. WindowScape: Lessons learned from a task-centric window manager. In *ACM Transactions on Computer-Human Interaction*, volume 19, pages 8–33. ACM - Association for Computing Machinery, 2012.
6. J. Thelin. *Foundations of Qt Development*. Apress, Berkeley, CA, 2007.
7. C. Zeidler, C. Lutteroth, and G. Weber. An evaluation of stacking and tiling features within the traditional desktop metaphor. In *Human-Computer Interaction INTERACT 2013*, pages 702–719. 2013.

[2] Chrome and Firefox were observed to use atypical drag-and-drop event sequences, preventing individual browser tabs from being dropped into a docked browser window.

[3] Vide dwm, i3, xmonad and many others on Linux, and split-screen features in Android N and iOS 9/10.