Clustering Windows into Related Tasks

Addendum for proposed CPSC 490 Project
Matthew Blackshaw
Advisor: Holly Rushmeier

After my first week of research I encountered a major problem with the data mining/grouping branch of my proposal. It is very hard to parse the contents of a window from an external application no matter which operating system/windowing environment is used. I considered the following solutions to this problem:

- Extension of current window environment libraries to enable support for retrieving document contents (which in turn requires applications to be rewritten).
- Write support/plug-ins for each application, but this is also extremely cumbersome as it requires intricate knowledge of the source code of each application.
- Finally I considered the possibility of taking static pictures of each window then using OCR software to parse contents; this could work in the general sense, but is a particularly ugly solution that detracts from the essence of the project.

Given the inability to parse window contents, I would have to group windows solely based on window title and temporal access to windows. This is in essence exactly the same approach taken by Microsoft’s SWISH prototype. Therefore, I propose an addendum to my project proposal.

Addendum
Focus on the implementation/presentation half of my project. Instead of using existing windows management solutions, such as workspaces in Linux, I could build my own window grouping implementation with the following characteristics:

- Windows can be dragged into a 'stack' which are representative of related tasks
- 'Stacks' can be created or destroyed (closing all windows within the stack)
- Mouse over of a stack expands view to all windows within the stack, one of which can then be clicked on for focus
- What happens when a window is placed in a stack - is it minimized? What happens when it is maximized? Does it leave the stack?
- How should stacks be represented (how can we make it easy for the user to distinguish between different stacks)?
- Where stacks should be placed on the screen?
What is the expanded view of a stack?

While researching, I was far more excited by this branch of the project and it aligns nicely with my interests in UI design and user experience. The focus on implementation also aligns itself with Professor Rushmeier’s graphics experience. Finally, many of these characteristics could break standard window management paradigms raising very interesting possibilities.

Research
This week I investigated manipulation of windows in Mac OS X, Windows Vista and using Compiz Fusion in Linux. I discovered that despite Mac OS X having some of the better window management solutions, no public APIs exist to achieve functionality similar to that already in the operating system. Therefore, I focused my research on Compiz Fusion and Windows Vista.

Windows Vista
Windows Vista includes well documented APIs for its Desktop Window Manager (dwm.exe). Dwm.exe is built upon Direct3D for handling of its graphics processing. It is also possible to obtain a live thumbnail stream of a window which can be manipulated through C# code. I have found the following DWM documentation:


I believe that if I were to implement using DWM/Vista, I would make the window switcher a separate application that manipulates windows.

Compiz Fusion (Linux)
Linux has a very advanced, community supported window manager that allows easy manipulation of windows through C code. Compiz-Fusion uses OpenGL to perform its windowing trickery and given Professor Rushermeier’s experience, this may be preferable. It has a nice plug-in infrastructure, which could be used to implement my ideas. In addition, I could utilize the functionality of existing plug-ins to build my own. I have found the following Compiz Fusion resources:

- [http://www.compiz-fusion.org](http://www.compiz-fusion.org)
Revised Milestones

1. **Research**: Decide on which implementation platform to use (Vista or Compiz Fusion). Extend knowledge of graphics libraries for chosen platform. Learn how to manipulate windows (hiding, scaling and acquiring thumbnails of them).

2. **Design**: Formalize a specification detailing exact functionality of the ‘stack’ concept. Specification should also detail what libraries are used to achieve this functionality.

3. **Implementation**: Implement design according to specification on chosen platform.