An Evaluation of Stacking and Tiling Features within the Traditional Desktop Metaphor

Clemens Zeidler[1], Christof Lutteroth[1] and Gerald Weber[1]

[1] University of Auckland, 38 Princes Street, Auckland 1010, New Zealand

czei002@aucklanduni.ac.nz, {lutteroth,g.weber}@cs.auckland.ac.nz

Abstract. Having many open windows on the desktop can lead to various usability problems. Window content may get occluded by other windows and working with multiple windows may get cumbersome. In this paper, we evaluate the idea to integrate stacking and tiling features into the traditional desktop metaphor. For this purpose we introduce the Stack & Tile window manager, which allows users to stack and tile arbitrary windows into groups that can be moved and resized similar to single windows. To evaluate if stacking and tiling can improve productivity, we conducted an experimental evaluation. We found that participants were able to perform various multi-window tasks and switch between tasks significantly faster using Stack & Tile. Furthermore, we found that the time to set up a Stack & Tile window group is reasonably low. Stack & Tile is open-source and has been used for over two years now. To evaluate its usefulness in practice, we conducted a web-based survey that reveals how people are actually using the new stacking and tiling features.

Keywords: window manager, tabbing, usability, evaluation

1 Introduction

In the traditional desktop metaphor, windows can float freely on the desktop and are allowed to overlap each other. One problem that arises when having multiple windows open at the same time is that the content of some windows may get partially or fully occluded. To make the content visible again the user has to bring the occluded window to the front or move other windows aside. This can make the management of overlapping windows tedious and time consuming [7] and task management can become another task [18].

An alternative approach to overlapping windows is a tiling window manager where windows are tiled beside of each other. This approach can be superior to the overlapping approach [6, 16]. There is also the idea of stacking windows on top of each other [3]. Tabbed interfaces became very popular in web browsers, but are widely unused in window management. Despite these alternatives and extensions, all main desktop

operating systems are still primarily using the traditional overlapping window approach.

In this paper we investigate the benefits of stacking and tiling features in a traditional overlapping window manager. To analyze such features, we first present Stack & Tile, which is an extension of a traditional window manager. Stack & Tile allows users to stack windows on top of one another and tile windows beside of each other (Figure 1). In this way the user is able to create window groups (Stack & Tile groups) consisting of arbitrary windows from different applications, which can be controlled similar to single windows. Window groups can still overlap other windows or window groups, so Stack & Tile integrates seamlessly into the traditional desktop.

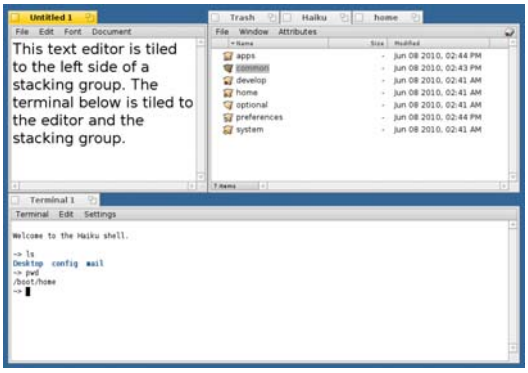

Fig. 1. Stack & Tile group. On the right side three windows are stacked into a stacking group. Tiled to this group are on the left side a text editor and at the bottom a terminal.

To answer the main question in this study, whether stacking and tiling within an overlapping window manager bring any benefits to the user, we conducted a controlled experiment to determine in which use-cases Stack & Tile performs better than a traditional window manager. We looked at use-cases where the user is working with documents of the same or of different applications, and a use-case where data is exchanged between documents of the same application, and designed experimental tasks accordingly. Another task measured the time needed to switch between different groups of windows. We found that for all tasks Stack & Tile performed significantly faster. Furthermore, the setup time used to create a Stack & Tile group is acceptably low. Most participants stated that Stack & Tile makes window management easier and more enjoyable.

Stack & Tile is already integrated in the open-source operating system Haiku[1], and thus is already exposed to a large group of developers and users. This allowed us to target another interesting question: How are stacking and tiling features used and accepted by real users? In a web-based survey we asked the Haiku community about their opinions and experiences of Stack & Tile. From 146 responses we got a detailed insight into how, how often and for what applications Stack & Tile is used.

[1] Haiku Operating System, www.haiku-os.org

1.1 Contribution

In this work, we analyze the potential of stacking and tiling features in the traditional desktop metaphor. In particular, we present:

1. A concept for integrating stacking and tiling unobtrusively into a traditional window manager.
2. A controlled experiment that shows how Stack & Tile performs for different use-cases.
3. A web-based survey that gives insight into how Stack & Tile is used by real users.

Section 2 introduces the Stack & Tile window manager. Section 3 describes how Stack & Tile is different from other work on window managers. In Section 4 the experimental evaluation is presented. Section 5 presents the web survey of how Stack & Tile is used. The paper closes with a conclusion in Section 6.

2 The Stack & Tile Window Manager

In Stack & Tile, users can stack and tile windows together to create window groups according to their needs. Groups behave like single windows, and can be used together with other windows as usual. Stack & Tile combines the advantages of stacked and tiled windows with the freedom of overlapping windows.

This combination can help users to manage their windows more effectively. Tiled windows in an active Stack & Tile group are always visible and not occluded by other windows. As shown later, this makes it easier to exchange data between windows. A window stack can be used to group windows together whose content does not need to be visible at the same time. Grouping windows used for a certain task together in a Stack & Tile group can result in a cleaner desktop and facilitate switching between tasks involving multiple windows.

Internally linear constraints are used to describe a Stack & Tile group, employing the tabstop and area system of the Auckland Layout Model (ALM) [17]. Areas are simply the tiles where windows can be placed, and tabstops are their borders. For example, two stacked windows are sharing the same tabstops and are therefore always getting the same size. The minimum size of a window is specified using a hard constraint while the maximum and current window sizes are specified using soft constraints. Window operations modify the constraint specification that describes all Stack & Tile groups, and the window manager solves the specification and re-renders the windows.

Currently, there is a fully working Stack & Tile implementation available, which is well-known and appreciated in the Haiku OS community. It is showcased regularly at large open-source conferences, and has been integrated into the default Haiku OS user interface.

2.1 Stack & Tile Operations

Stack & Tile offers two simple operations that can be used to connect windows: First, stacking, which makes use of the tab-like appearance of the Haiku OS window title bars; secondly, tiling of windows, which means that windows are arranged beside each other. A Stack & Tile operation can be triggered by holding down the Stack & Tile key, which is by default the Windows key, and dragging a window near to another window (see Figure 2). Releasing the Stack & Tile key or dropping the window finally executes a Stack & Tile operation. The dragged window is called the *candidate window*, and the window that it is dragged to is called the *parent window*. In this manner, *Stack & Tile groups* can be created.

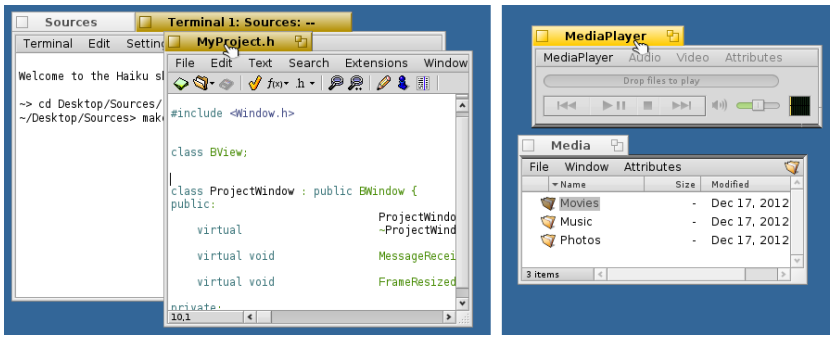

Fig. 2. Moving a window while holding the Stack & Tile key initiates stacking or tiling. Affected window tabs and borders are highlighted in gray. Left: The editor window is going to be stacked on top of the Terminal. Right: MediaPlayer is going to be tiled to the Media folder.

Stacking: Briefly, a stacking operation is triggered by moving the title tab of the candidate window onto the tile tab of a parent window while holding the Stack & Tile key. To be more precise, the candidate window has to be dragged by the title tab so that the upper edge of the candidate window tab is on the parent window tab, and the *x*-position of the mouse cursor is in the *x*-range of the parent title tab. When a valid stacking candidate-parent pair is found, the window title tabs of both windows are highlighted (left of Figure 2).

After stacking windows on top of one another, the stacked windows have the same position and size. The title tabs are automatically arranged beside each other, so that the stacked windows are accessible over a tab interface at the top of the stack. The result is comparable with a tab bar, e.g. in a tabbed web browser, with a similar functionality for reordering tabs.

Tiling: If the Stack & Tile key is pressed, dragging and dropping a candidate window border close to one or more parent window borders triggers the tiling operation (right of Figure 2). Tiled windows always share a border position. For example, when tiling two windows horizontally, the right border of the left window always has the same position as the left border of the right window. Furthermore, the top and bottom of both windows are aligned with each other. Windows can be tiled to any free rec-

tangular region of an existing group, and can span the width and height of several other windows in the group. Figure 1 shows an example of a tiled window group.

Removing from a Group: A window can be removed from a Stack & Tile group by holding down the Stack & Tile key and dragging the window away from the group. After removing the window from the group, the window behaves just like an ordinary window in the desktop metaphor. In case the removed window was the only connection between other windows in the group, the group is split into several independent groups.

2.2 Traditional Window Management Operations

When interacting with a Stack & Tile group, the semantics of the traditional window management operations change slightly. Stack & Tile applies window management operations to multiple windows, which has already been considered to be helpful in Elastic Windows [16].

Activating one window in a Stack & Tile group raises all windows in the Stack & Tile group. Only the window that triggered the group operation gets the input focus.

Moving a window by a certain offset also moves all other group windows by the same offset. This means windows in a Stack & Tile group keep their relative position to each other.

Resizing one window in a Stack & Tile group leaves all windows in the group aligned to each other. For example, windows that are tiled to a resized window are moved or resized accordingly. This is done by temporarily setting a high priority for the size constraint of the resized window. In this way, the resized window gets its new size and the other windows adapt according to the solution of the constraint system.

Hiding or **showing** a window also hides or shows all other windows in the group. Thus, all windows in a Stack & Tile group are either hidden or shown.

3 Related Work

Novel techniques for overlapping windows such as snapping windows to a master window or organizing them in tabs have already been proposed in 2001 [3]. However, they have not been evaluated for traditional window managers.

A comparison between overlapping, stacked and "piled" windows found that tabbed interfaces are doing well when it comes to finding a document [14]. The study only looked at windows of the same application, while our study also looks at tasks involving windows of different applications (in contrast to windows of the same application, they are not grouped in the taskbar). Furthermore, they did not consider the presence of windows that were not part of the task at hand.

The Google Chrome[2] browser can stack web pages running in different processes, and extends the usage of the tab interface to stack different instances of the same ap-

[2] Google Chrome, www.google.com/chrome

plication. Stack & Tile is more general and is working not only with windows of a particular application, but with arbitrary windows of arbitrary applications.

Roughly one year after the first release of Stack & Tile, a similar stacking feature was implemented in the KDE desktop environment[3]. Although KDE has an optional Notion-like tiling window manager (see below), a combination of stacking and tiling in the standard KDE desktop is not possible.

Tiling window managers allow users to tile windows beside each other. Windows are arranged so that they do not overlap each other and are aligned without gaps and fragmentation. In general, windows are arranged automatically by the window manager, using the whole screen, but the user has the ability to rearrange the window tiling using different layouts [4, 10, 11, 16, 19]. For example, the tiling window manager Notion[4] allows the user to rearrange the tile layout manually, and multiple windows can be stacked into one tile. By comparison, Stack & Tile integrates seamlessly with the traditional desktop metaphor. In Windows 7, two windows can be tiled horizontally using the whole screen space. However, more complex tiling layouts with more than two windows, as in Stack & Tile, are not possible.

Tiling windows can help the user to organize their windows better. Already in old studies it has been shown that tiled windows can be superior to overlapping windows in certain use-cases. For example, if all window content fits into the allocated tiles, the tiling approach leads to shorter task completion times [6]. Another study found that the completion time is lower for tiled windows when comparing information sources in multiple windows and scanning through windows of a certain window group [16]. Stack & Tile leverages the advantages of tiled windows, but integrates tiling with the traditional overlapping window management.

It is quite common that users work on different task in parallel and switch back and forth between different windows [20]. Typically windows from different tasks are overlapping each other, and task switching involves the manipulation of many windows. One approach to address this is to let the user group windows by task [16, 21]. Another approach is to analyze the user activities and assign windows to tasks automatically [5, 13, 18, 21, 23]. For example, WindowScape [21] creates a timeline of previous window configurations, and also allows users to assign windows to tasks manually. This is similar to an activity centered desktop [2] where applications can be assigned to activities. Also Stack & Tile can be used to group windows by task. Additionally, Stack & Tile helps users to optimize the layout of window groups.

To reduce the probability of overlapping existing windows on the screen and to group windows by task, the concept of multiple virtual workspaces can be used [15]. Another option is to increase the physical workspace by attaching multiple monitors to a computer [12]. However, often windows overlap other windows, and some approaches use alternative methods to make overlapped windows temporary available without losing the focus on the active window [8, 9]. For example, an occluded window can be made visible by folding the overlapping window back like a piece of paper [9].

---

[3] KDE Desktop Environment, www.kde.org

[4] Notion Window Manager, http://notion.sourceforge.net

The Scwm window manager [1] gives the user the opportunity to specify relations and constraints for each window. For example, it is possible to set a relative distance between two windows or set the minimum or maximum size of a window. With this approach it is also possible to tile windows beside each other. However, because constraints in Scwm are on a lower level of abstraction, tiling is not as easily accessible as in other tiling window managers and Stack & Tile.

4 Experimental Evaluation

In this section we compare Stack & Tile with a traditional window manager in a controlled experiment, using tasks that are representative of certain use-cases that involve multiple windows (*multi-window tasks*). Stack & Tile makes it possible to group windows with direct manipulation operations, and offers window management operations that affect all the windows in a group. This can be used to manage occlusion and quickly switch between groups of windows related to different tasks. An interesting question is how much time can be saved using Stack & Tile, and if the time taken to set up a Stack & Tile group is reasonable compared to the time saved. We formulate the following hypotheses about Stack & Tile for this experiment:

- **H1** The task completion time for multi-window tasks is lower with Stack & Tile than with a traditional window manager.
- **H2** Switching between tasks that each involve multiple windows is faster with Stack & Tile than with a traditional window manager.
- **H3** The time necessary for setting up Stack & Tile groups is acceptable compared to the time that can be saved by using Stack & Tile.
- **H4** Users prefer Stack & Tile over a traditional window manager.

4.1 Methodology

We conducted a within-subjects study, i.e. each participant performed two runs of tasks: one run with Stack & Tile and one run without Stack & Tile. To avoid order bias, the order of the runs was alternated between participants. Furthermore, there were two different sets of similar tasks for each run. Also the order of the sets of tasks was permuted. Each run included a window setup phase followed by three question tasks and a group switching task.

To start with, each participant got an introduction to the Haiku window manager, its taskbar and Stack & Tile. After that the participant had time to get familiar with the system. Then each participant performed a training run, which was similar to the main runs but with shorter tasks. Participants were allowed to use different methods to switch between windows, e.g. Alt+tab or the taskbar.

Each run was guided by instructions shown in a small instruction window that was placed at the bottom-right screen corner on top of all other windows (see Figure 3). The size of the instruction window was chosen to be as small as possible to not interfere with other windows on the screen. At the beginning of each task, the instruction window was in fullscreen mode, so that the participant was not distracted by other

windows on the screen. After the participant had internalized the instruction, she had to press the start button to start a timer. This resized the instruction window back to its smaller size at the bottom-right. For the question tasks, the instruction window also contained a question with a three-point multiple choice answer field. When the participant had completed the task, she had to press a finish button to stop the timer. Pressing the finish button adds a small and fairly constant offset to the task completion time, which does not affect the outcome of our comparison. Afterwards, the instruction window was shown in fullscreen mode again with the next instructions.

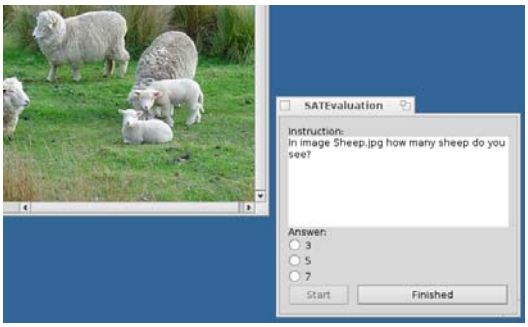

Fig. 3. The instruction window (right) with a question about a picture window (left).

Each task involved multiple windows, and all windows of all tasks (in total 12 windows) were open all the time. We think this is a realistic number of windows, but it is not unusual that users have even more windows open at a time [22]. We were interested in the efficiency of managing multiple windows and tasks, so we tried to minimize the time spent on activities unrelated to window management. Thus, the question tasks were designed to be very easy, but the answers were not guessable. The three question tasks were targeting H1, with each task addressing a different use-case.

Setup Phase The first step of a run was to open all windows for all tasks. For each task, the participant got a description in the instruction window about what windows were involved in the task. Then the windows were opened automatically, so that the participant could get familiar with them before starting the task. When using Stack & Tile, there was an extra step where the participant had to stack or tile the windows, depending on the task. Here the *setup time* taken to create a certain Stack & Tile group was measured.

Task I The first question task was about finding a picture in a set of five pictures and answering an easy question about it. Each picture was opened in a single window and there was one question for each picture (e.g. Figure 3). This targeted the use-case of working with multiple windows of the same application, where no data needs to be exchanged between the windows. It simulates typical lookup tasks, where a user needs to activate the window that contains a particular piece of information.

When it comes to finding a certain window, it can be helpful if all windows of interest for the task have been stacked previously. In this case, activating one window of the stack brings the whole group to the front, and the right window can be chosen

through the tab interface. When using Stack & Tile, the participant was asked to stack all five pictures on top of one another to take advantage of this behavior. Without Stack & Tile, the participant had to search for the right window on the desktop, or use the taskbar. Because the taskbar groups windows by application, it provides a similar grouping facility as a window stack, and it is unclear if Stack & Tile is advantageous.

Task II The second question task was similar to Task I. Here, three windows were involved: a web page, a PDF document and a text document. This targeted the use-case of working with windows of different applications, where no data needs to be exchanged between the windows. As before, there was one question for each document, e.g. "What is the first word of the second paragraph in the web page?" In the Stack & Tile condition, the three windows had to be stacked first.

There are two reasons why it is expected that navigation in this more heterogeneous group is more difficult than in Task I. First, because windows are opened with different applications, they are not grouped together in the taskbar anymore. Secondly, because documents of different types also differ visually, users cannot rely as much on visual similarity as in Task I when associating a window with the task.

Task III The third question task targeted the use-case of working with windows of the same application, where data needs to be exchanged between the windows. To simulate this use-case, a simple coordinate treasure map game was chosen. A treasure map is a 7x5 table, with each cell containing either a coordinate pointing to a cell in another map or a treasure. For example, the coordinate M2(D,5) points to map 2, cell (D,5). Starting from an initial coordinate, the participant had to visually follow the path through four different maps to find a "treasure". The treasure was always reached after three steps, and the path crossed all four maps in random order. When the treasure was found, the kind of treasure had to be selected in the instruction window. This had to be repeated three times, each time starting from a different coordinate in the first treasure map.

While this task may seem artificial, it does simulate real tasks where related information has to be collated from multiple sources. For example, a real task of that kind would be looking up the location of an appointment mentioned in an email from a calendar, and then checking a booking for that location. Many such tasks involve tabular information, as simulated by the treasure maps.

The four treasure maps were each opened in a text editor window. When using Stack & Tile, the participant had to tile the four editor windows beside each other in a 2x2 layout. In this way, it was possible to display all four maps without occlusion on the screen, i.e. all maps were completely visible.

Task IV: Group switching This task evaluated the efficiency of switching between different tasks (H2). All windows from the previous tasks were used, and the user was asked to bring the windows of each task to the front, one task at a time. Without Stack & Tile, this can be done by directly clicking the windows on the desktop, or by activating them from the taskbar. When using Stack & Tile, only one window has to be activated to bring the whole group to the front. We expected Stack & Tile to be clearly faster here; this task was included to shed light on how much faster it actually is. For the non-Stack & Tile condition , we expected a correlation between group size (between 3 and 5 windows per group) and activation time.

Questionnaire After finishing both runs, the participant was asked to fill in a Likert-scale questionnaire and make some general comments about what they liked and disliked about Stack & Tile. Furthermore, the participant had to estimate their usage in percentages of the following window management techniques: taskbar, short cuts, direct window access, virtual desktops and other.

4.2 Results and Discussion

There were 30 participants with an average age of 31 ($\sigma = 6$). Six of them were female. Most of them (~ 70%) were software developers or students of Computer Science or Software Engineering. 15 of the participants were from the Haiku community and 7 of them had used Stack & Tile before. We found that users who had used Stack & Tile before performed only slightly better than those who had not, therefore we do not differentiate between these groups in the following.

For the analysis of the task completion times, the difference between the task completion time with Stack & Tile $t_{S\&T}$ and the task completion time without Stack & Tile $t_{noS\&T}$ was calculated: $\Delta t = t_{noS\&T} - t_{S\&T}$. A pairwise Wilcoxon rank-sum test was used to calculate the probability p_{wrs} for the null-hypothesis. The Wilcoxon rank-sum test was chosen because the task completion times were only roughly Gaussian-distributed.

For each question task, three answer options were given, with only one correct option. In only ~ 3% of all cases a questions was answered wrongly. Furthermore, there was no difference between the error rates with and without Stack & Tile. Thus we can say that it was easy for the participants to answer the questions. From our observations, the participants followed the instructions carefully. Even when they chose the wrong answer, they applied the necessary window operations. Because we are not primarily interested in how accurate a task was performed but in how Stack & Tile affects window management, wrong answers were not removed from the result set.

Task I Results (5 Pictures) Table 1 shows the average task completion time $t_{S\&T}$ under the Stack & Tile condition, the average time difference Δt, their standard deviation σ and the Wilcoxon rank-sum test p-value. For all five questions, Stack & Tile had significantly shorter task completion times. Thus we can accept H1 for Task I.

Answering the first question had the longest completion time. This can be explained by an additional step: first the Stack & Tile picture group had to be found and activated. After that participants were able to just select the next picture from the tab bar and the task completion time stayed roughly constant. The time difference Δt decreases from Picture 1 to Picture 5. A possible explanation is that for the non-Stack & Tile conditions, all pictures were moved to the front after some time and it became easier to find the right window. Another possible explanation is that the participants got into the routine of selecting the right picture from the taskbar.

Table 1. Task I completion times in [s].

	$t_{S\&T}$	σ	Δt	σ	p_{wrs}
Picture 1	12	5.7	5	11.8	0.01**
Picture 2	8	2.8	5	5.7	< 0.01**
Picture 3	8	4.3	3	6.7	< 0.01**
Picture 4	8	6.5	3	8.8	< 0.01**
Picture 5	7	3.2	2	4.5	< 0.01**

Table 2. Task II completion times in [s].

	$t_{S\&T}$	σ	Δt	σ	p_{wrs}
Web Page	17	7.7	-1	11.5	0.65
PDF	10	13.9	5	18.2	< 0.01**
Text file	9	4.85	8	10.9	< 0.01**

Task II Results (3 Documents in 3 Apps) The results are shown in Table 2. For the web page question there was no significant completion time difference. However, for the two following questions, Stack & Tile had a significantly better performance.

So why did participants have problems with the web page question while they were fine with the first picture question in Task I? A possible explanation is that the window with the web page was harder to activate than the windows with the pictures. When using direct activation by clicking on a window, all a participant had to do to select the Stack & Tile group of picture windows was to click any window with a picture on the screen, since there was only one group with pictures in it. Once the group was activated, the tab interface could be used to raise the right picture. However, there were two window groups with textual documents in it, so activating the window group containing the web page was not a as simple as clicking any window containing text. When using the taskbar to activate a window, for the window group with the pictures there was exactly one group of windows listed in the taskbar, as all the pictures were shown using the same application. However, the documents for Task II were all opened with different applications and hence there was a different entry in the taskbar for each of the windows. The taskbar gave no indication that the windows were grouped together, as for the five pictures. In fact, the taskbar obfuscated the Stack & Tile grouping of the windows, by grouping them by application together with other windows belonging to different Stack & Tile groups.

Hypothesis H1 cannot be accepted for the initial question of Task II. It is reasonable to assume that once a desired window group is activated, the advantages of Stack & Tile show more clearly. To facilitate this, as a future work, the taskbar could be changed to group windows by their Stack & Tile groups.

Task III Results (Treasure Maps) The results of Task III are shown in Table 3. When using Stack & Tile, finding the first treasure required a lot less time (21s). This time difference decreased to 7s and 5s for the second and third treasures. We can clearly accept hypothesis H1 for Task III.

This can be explained from the observations of the participants during the task. Without Stack & Tile, many participants first tried to position all treasure maps in a 2x2 grid to make them visible at the same time. Thus, finding the following treasures became easier for them.

However, finding the second and third treasures was still significantly faster using Stack & Tile. A reason for that is that many participants did not manage to align the

treasure maps precisely without occlusion, and thus had to rearrange the windows later on. Moreover, some participants accidentally activated unrelated windows, which made more window operations necessary. This is consistent with the findings from Elastic Windows [16], were they found that setting up and working with overlapping windows becomes more difficult for an increasing number of windows.

Task IV Results (Activate Groups) As expected, activating all windows of a task is much faster using Stack & Tile (Table 4). This clearly supports hypothesis H2. However, no significant correlation between the number of windows of a task and the activation time could be detected. Here, the heterogeneous Group II (Task II) resulted in the largest $t_{S\&T}$. Activating a group containing only windows of the same application seems to be easier than activating a group containing windows of different applications. This is consistent with the results of Task II.

Table 3. Task III completion times in [s].

	$t_{S\&T}$	σ	Δt	σ	p_{wrs}
Treasure 1	23	15.6	21	22.7	< 0.01**
Treasure 2	19	8.6	7	14.5	0.01**
Treasure 3	15	5.1	5	8.1	< 0.01**

Table 4. Task IV: Time in [s] to activate the window groups of the Tasks I-III.

	$t_{S\&T}$	σ	Δt	σ	p_{wrs}
Group I	6	2.5	12	5.5	< 0.01**
Group II	8	9.0	9	12	< 0.01**
Group III	4	1.3	7	4.7	< 0.01**

Stack & Tile Setup Time Results To assess whether the time needed for setting up Stack & Tile groups is acceptable (H3), we compared the average setup time t_{setup} with the average saved time t_{saved} for each task (Table 5). t_{saved} is the sum of Δt for all questions of a particular task. The experimental tasks were artificial hence one could argue that such a comparison is not meaningful. However, all experimental tasks were quite short compared to real world tasks, so these numbers serve to indicate that t_{saved} is reasonably likely to outweigh t_{setup} when working with Stack & Tile groups a bit longer. From our observations, participants still had problems to set up Stack & Tile groups in an optimal manner after the training tasks. It is reasonable to assume that once users get more practice with Stack & Tile, the setup times will drop. A non-significant indication ($p < 0.25$) for this is that users who had used Stack & Tile before had a slightly shorter setup time (on average 2-3 seconds for each of the three tasks). Note that the setup time does not include the time users needed to decide how a Stack & Tile group should look like, because the layouts for the groups were given in the experiment.

Table 5. Setup times in [s] for the window groups of Tasks I-III.

	Task I	Task II	Task III
t_{setup} (σ)	29 (15.7)	14 (11.1)	22 (10.9)
t_{saved}	18	12	33

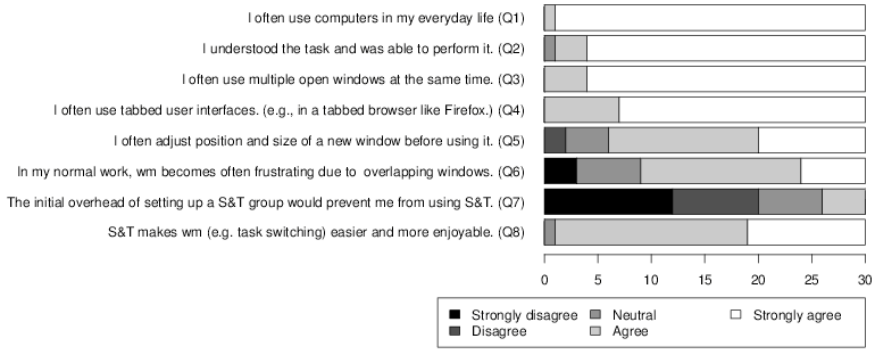

Fig. 4. Results for the Likert-scale questions.

Questionnaire Results Figure 4 shows the results from the Likert-scale questions. All participants agreed that they often use tabbed user interfaces (Q4), which shows that the stacking feature is not a fundamentally new concept to them.

There was an overall agreement that they often have multiple windows open (Q3), and over 2/3 of them agreed that window management often becomes frustrating when working with overlapping windows (Q6). This indicates that there is a need for better window management.

Only 3 of the 30 participants agreed that the initial overhead to setup a Stack & Tile group would prevent them from using Stack & Tile (Q7), and 24 of them agreed that they often adjust the position and size of a window before using it (Q5). Both supports H3 (see also Section 4.2).

Lastly, there was a general agreement that Stack & Tile makes window management easier and more enjoyable (Q8). This is also supported by many comments. As an example, one participant said: “Much easier to use. Better grouping, while still having more freedom than MDI apps”. This supports H4.

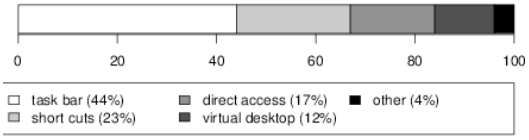

Fig. 5. Used techniques for window management.

The results for the question about the techniques participants use to manage windows are depicted in Figure 5. The results are consistent with the empirical findings in [20]. Participants said that they use direct window access only 17% of the time. This is another indication that direct window activation is not perceived as optimal.

5 Web-based Survey

Stack & Tile has been available in the Haiku OS for over two years now. An interesting question is how people are actually using Stack & Tile. To answer this question we conducted a web-based survey in the Haiku community. The questions were mainly a mix of Likert-scale questions and open questions.

The survey began with some demographic and general questions:

S1: I often use computers in my daily life.
S2: I heard about S&T before.
S3: I think S&T can be useful.
S4: I don't think there is any need for S&T.
S5: Have you ever tried S&T? (Denying this question ended the survey.)

Afterwards the participants were asked to upload screenshots of how they use S&T most frequently and describe them, followed by questions about S&T:

S6: How often do you use S&T?
S7: I think the stacking feature is more useful than the tiling feature.
S8: S&T helps with resizing window groups.
S9: Estimate the percentage of how often you use stacking and how often you use tiling.
S10: When you use S&T, how many S&T groups do you use on average at the same time?
S11: I exchange information between single windows (not in a S&T group).
S12: I exchange information within a single S&T group.
S13: I exchange information between multiple S&T groups.
S14: I exchange information between S&T groups and other single windows.
S15: Are there certain tasks where you use S&T? For example, programming or browsing.
S16: In what other situations are you using S&T?

Finally, participants had the opportunity to make suggestions and comments.

5.1 Result

During a period of two month we got 146 responses. The average age of all subjects was 32 ($\sigma = 10$). There was only one female participant. The majority of the participants was working in or had a degree in an IT-related field.

There were two groups of participants, distinguished by the general questions at the beginning: the group that had not tried Stack & Tile before (left of Figure 6) and the group that had tried Stack & Tile before (right of Figure 6). Generally both groups were heavy computer users. For the group that had not tried Stack & Tile before, almost half of them had heard about Stack & Tile. Most participants agreed that Stack & Tile is useful and disagreed that there is no need for Stack & Tile. This general opinion was much stronger for the group that had tried Stack & Tile before, indicating that Stack & Tile had made a positive impression.

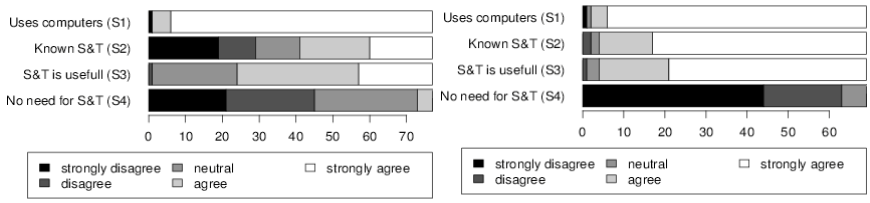

Fig. 6. Answers for S1 - S4. Left: users who had not tried Stack & Tile. Right: users who had tried Stack & Tile.

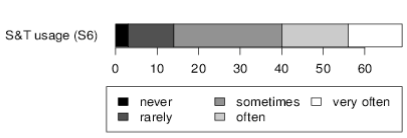

Fig. 7. Answers for S6.

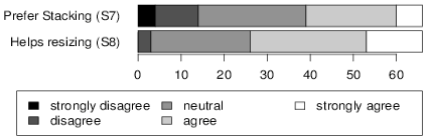

Fig. 8. Answers for S7 and S8.

Figure 7 shows the results for question S6. Here we can see that most people who had tried Stack & Tile before were still using it.

Question S7 asked for preferences for the stacking or the tiling feature. There is a trend that the participants feel stacking is more useful than tiling (see Figure 8). S9 asked for the usage of the stacking and tiling features. The average estimated percentage of tiling is 45% (σ = 29%) and for stacking 55% (σ = 29%). Stacking is slightly more used than tiling with p_{wrs} = 0.06. This indicates that participants think that stacking is more useful (S7) and consequently use it more (S9). However, keeping in mind that stacking is much more common in today's applications (e.g. tabbed browsing), it is still interesting that the tiling feature is reportedly used that much.

S8 targeted the feature that windows in a group stay aligned when resizing one window in the group. Most of the participants judged this to be helpful (see Figure 8).

Figure 9 shows the average number of groups when Stack & Tile was used. Most participants used between one and four groups at a time, with the majority using more than one. This is quite interesting because it means they not only used it sporadically for a single task, but seem to have used it for different tasks in parallel.

We already analyzed the advantages of Stack & Tile when exchanging data between windows or Stack & Tile groups in the controlled experiment. S11 - S14 targeted the question in how far participants encounter such use-cases in practice (Figure 10). The results are quite similar for S11, S12 and S14. Most people are exchanging data between groups/windows, which indicates that the results from the controlled experiment are relevant for real users. The least frequent data exchange was between Stack & Tile groups (S13), which indicates that Stack & Tile groups are used to group windows of different tasks.

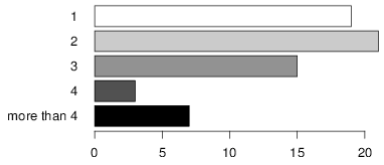

Fig. 9. Answer for S10: Number of S&T groups used at a time.

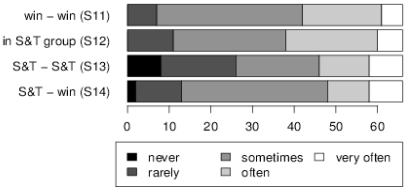

Fig. 10. Answers for S11 - S14: Exchanging data between windows.

5.2 Use-Cases for Stack & Tile

We asked the participants to upload screenshots of how they use Stack & Tile most frequently, and we got 23 screenshots in response. Furthermore, one user sent a link to a video with a demonstration of how he uses Stack & Tile. S15 and S16, targeting the questions for what tasks and in what situations Stack & Tile is used, received 47 and 23 responses, respectively.

The results indicate that there are three main use-cases where the participants used Stack & Tile: programming (28 participants), web browsing (17 participants) and file management (11 participants). For example, C++ source and header files were tiled beside of each other, or all source files were stacked to one stack and header files to another stack. Many users reported that they use Stack & Tile to group windows by task, e.g. group web browser and chat window together. Other examples were grouping a music directory and a media player, a picture directory and an image viewer, or creating an ad-hoc development environment by grouping source files, source directory and terminal together. One user tiled a web browser and a text editor together to copy information from the browser to the editor. An unexpected use-case was the creation of a Stack & Tile group to move multiple windows across different virtual desktops more easily.

From the screenshots we observed that the Stack & Tile groups had mostly moderate complexity. There were only a few use-cases where more than two windows were tiled together. However, the stacked window groups had usually more than two windows in them.

There were 34 responses for the open-ended comments & suggestions field. Here participants had some ideas for improvements and better integration into the desktop. For example, there was the request to show Stack & Tile groups in the taskbar, and that it should be possible to store and restore Stack & Tile groups, especially on reboot. Applications that are using a tabbed interface should use the stacking feature instead. These are points we want to target in future work. One participant stated that he already started to replace the tabbed interface of a browser with the Stack & Tile stacking feature. 11 participants explicitly said that they like Stack & Tile, while one participant did not think that Stack & Tile can improve manual window management.

6 Conclusion

In this paper we investigated the advantages of integrating stacking and tiling features into the traditional desktop metaphor. Stacking and tiling can help to manage windows more effectively, for example by grouping windows by task. We presented Stack & Tile, a window manager which integrates stacking and tiling seamlessly with traditional window management operations.

In a controlled experiment, we found that stacking and tiling features can significantly improve completion times for tasks involving several windows (of the same application as well as of different applications). Furthermore, switching between different tasks was found to be much faster when windows were grouped by task. Setting up a Stack & Tile group is an initial overhead that may prevent users from using these features. However, the potential time savings as well as questionnaire answers indicate that the advantages outweigh this overhead.

In a web survey we investigated how often and how Stack & Tile is used in practice. There was a wide agreement that Stack & Tile can be useful, especially by participants who had used Stack & Tile. The stacking feature was perceived as being slightly more useful and also estimated to be used more than the tiling feature. We found that people were using the stacking and tiling features for a multitude of different use-cases. In a field for general comments many people wrote that they like Stack & Tile and suggested further ideas to integrate it more into the desktop. These ideas include future works such as grouping of windows by their Stack & Tile group in the taskbar and Stack & Tile group persistence.

References

1. Badros, G.J., Nichols, J., Borning, A.: Scwm: An extensible Constraint-Enabled window manager. In: Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference. pp. 225–234 (2001)
2. Bardram, J., Bunde-Pedersen, J., Soegaard, M.: Support for activity-based computing in a personal computing operating system. In: Proc. SIGCHI Conference on Human Factors in Computing Systems. pp. 211–220. CHI ’06, ACM, New York (2006)
3. Beaudouin-Lafon, M.: Novel interaction techniques for overlapping windows. In: Proc. 14th annual ACM symposium on User interface software and technology. p. 154 (2001)
4. Bell, B.A., Feiner, S.K.: Dynamic space management for user interfaces. In: Proc. 13th annual ACM symposium on User interface software and technology. p. 248 (2000)
5. Bernstein, M.S., Shrager, J., Winograd, T.: Taskposé: exploring fluid boundaries in an associative window visualization. In: Proc. 21st ACM symposium on User interface software and technology. pp. 231–234. UIST ’08, ACM, New York (2008)
6. Bly, S.A., Rosenberg, J.K.: A comparison of tiled and overlapping windows. ACM SIGCHI Bulletin 17(4), 106 (1986)
7. Bury, K.F., Darnell, M.J.: Window management in interactive computer systems. ACM SIGCHI Bulletin 18(2), 65–66 (1986)
8. Chapuis, O., Roussel, N.: Metisse is not a 3D desktop! In: Proc. 18th ACM UIST. p. 22 (2005)

9. Chapuis, O., Roussel, N.: Copy-and-paste between overlapping windows. In: Proc. CHI. p. 210 (2007)
10. Funke, D.J., Neal, J.G., Paul, R.D.: An approach to intelligent automated window management. International Journal of Man-Machine Studies 38(6), 983 (1993)
11. Haraty, M., Nobarany, S., DiPaola, S., Fisher, B.: AdWiL: adaptive windows layout manager. In: Proc. 27th international conference extended abstracts on Human factors in computing systems. pp. 4177–4182 (2009)
12. Hutchings, D.R., Smith, G., Meyers, B., Czerwinski, M., Robertson, G.: Display space usage and window management operation comparisons between single monitor and multiple monitor users. In: Proc. of the working conference on Advanced visual interfaces. pp. 32–39 (2004)
13. Ishak, E.W., Feiner, S.: Content-aware layout. In: CHI '07 Extended Abstracts on Human Factors in Computing Systems. pp. 2459–2464. CHI EA '07, ACM (2007)
14. Jakobsen, M.R., Hornbæk, K.: Piles, tabs and overlaps in navigation among documents. In: Proc. 6th Nordic Conference on HCI: Extending Boundaries. pp. 246–255. NordiCHI '10, ACM (2010)
15. Jr, D.A.H., Card, S.: Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. ACM Transactions on Graphics (TOG) 5(3), 211–243 (1986)
16. Kandogan, E., Shneiderman, B.: Elastic windows: evaluation of multi-window operations. In: Proc. of the SIGCHI conference on Human factors in computing systems. pp. 250–257 (1997)
17. Lutteroth, C., Strandh, R., Weber, G.: Domain specific High-Level constraints for user interface layout. Constraints 13(3) (2008)
18. Oliver, N., Smith, G., Thakkar, C., Surendran, A.C.: SWISH: semantic analysis of window titles and switching history. In: Proc. 11th international conference on Intelligent user interfaces. p. 201 (2006)
19. Stewart, D., Sjanssen, S.: Xmonad. In: Proc. of the ACM SIGPLAN workshop on Haskell workshop. pp. 119–119. ACM (2007)
20. Tak, S., Cockburn, A., Humm, K., Ahlstrm, D., Gutwin, C., Scarr, J.: Improving window switching interfaces. In: Human-Computer Interaction – INTERACT 2009, Lecture Notes in Computer Science, vol. 5727, pp. 187–200. Springer Berlin Heidelberg (2009)
21. Tashman, C., Edwards, W.K.: Windowscape: Lessons learned from a task-centric window manager. ACM Trans. Comput.-Hum. Interact. 19(1), 8:1–8:33 (May 2012)
22. Wagner, J., Mackay, W.E., Huot, S.: Left-over Windows Cause Window Clutter... But What Causes Left-over Windows? In: Ergo'IHM 2012 - 24th French Speaking Conference on Human-Computer Interaction. International Conference Proceedings Series, AFIHM, ACM, Biarritz, France (Oct 2012)
23. Xu, Q., Casiez, G.: Push-and-pull switching: window switching based on window overlapping. In: Proc. SIGCHI Conference on Human Factors in Computing Systems. pp. 1335–1338. CHI '10, ACM, New York (2010)