StencilMaps and EphemeralMaps: spatially stable interfaces that highlight command subsets

Joey Scarr, Carl Gutwin, Andy Cockburn & Andrea Bunt

To cite this article: Joey Scarr, Carl Gutwin, Andy Cockburn & Andrea Bunt (2015) StencilMaps and EphemeralMaps: spatially stable interfaces that highlight command subsets, Behaviour & Information Technology, 34:11, 1092-1106, DOI: 10.1080/0144929X.2015.1046927

To link to this article: http://dx.doi.org/10.1080/0144929X.2015.1046927

Published online: 29 May 2015.
StencilMaps and EphemeralMaps: spatially stable interfaces that highlight command subsets

Joey Scarra∗, Carl Gutwinb, Andy Cockburna and Andrea Buntc

aComputer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand; bDepartment of Computer Science, University of Saskatchewan, Saskatoon, Canada; cDepartment of Computer Science, University of Manitoba, Winnipeg, Canada

(Received 30 June 2014; accepted 8 April 2015)

Identifying a target command can be difficult and time-consuming when the user is unfamiliar with a software system. One technique for assisting command identification is to provide a subset interface that contains only a limited set of the system’s capabilities. We examine the design of subset interfaces, showing that subsets can be presented separately to the full user interface (UI) (e.g. in a palette) or in place, with in-place methods using either static or dynamic methods to identify the subset. We introduce the StencilMap and EphemeralMap as in-place subset UIs that, respectively, use static and dynamic highlighting. Both StencilMaps and EphemeralMaps make all of an application’s commands concurrently available for selection within a grid. To highlight subset items StencilMaps use a static dark semi-transparent ‘stencil’ overlay to de-emphasise all but the subset items; EphemeralMaps, in contrast, use a short delay, with subset items shown immediately, and other items gradually faded in. A first experiment compares user performance with the in-place presentation of StencilMaps against that of the separate presentation of a subset palette. Results confirm the predicted spatial memory benefits for StencilMaps. A second experiment analyses the performance impact of three approaches to highlighting: none, static highlighting in StencilMaps, and dynamic highlighting in EphemeralMaps. Results show an interesting trade-off – while highlighting can offer benefits in assisting rapid target identification (particularly when the user is unfamiliar with the interface layout), there can also be longer-term performance benefits when highlighting is absent because the increased difficulty of visual search promotes the use and formation of spatial memory.

Keywords: stencils; spatial memory; subset interfaces; highlighting

1. Introduction

Many computer applications have dozens or hundreds of commands that provide specific and diverse functionalities for varied tasks and purposes. However, visually searching for and identifying a specific target command can be difficult when unfamiliar with the user interface (UI). As a result, several techniques have been proposed for reducing the number of commands that the user needs to deal with—that is, for providing a subset view of the interface to assist in novice interaction.

These subsets are typically predefined by the designer (Carroll and Carrithers 1984), although they may also be manually adapted or automatically adaptive (Greenberg and Witten 1985; Gajos et al. 2006; Lavie and Meyer 2010). For example, Figure 1 shows the AdaptableGIMP (Lafreriére et al. 2011), which allows community members to create static command subsets for common image-processing tasks, such as red-eye reduction or adding a sepia effect. When selected, task-based subsets replace the standard toolbox, bringing subset commands within easy reach—Figure 1(a) shows the full AdaptableGIMP UI, and Figure 1(b) shows the ‘Simple Photo Editing’ command subset palette.

Previous research with reduced-functionality interfaces (Carroll and Carrithers 1984; Findlater and McGrenere 2007; McGrenere et al. 2007) shows that working with subsets can improve both user performance and satisfaction. However, the merits of different methods for presenting command subsets remain relatively unclear. Studies presented in this paper examine some of these issues through two new UI designs, called StencilMap and EphemeralMap.

Subset interfaces can vary in the degree of separation between the presentation of the subset interface and the presentation of the standard interface. We broadly categorise these approaches as using separate or in-place presentation. For example, when a user selects a task subset in AdaptableGIMP, the subset is presented within a separate new palette frame that has no spatial relationship to the full UI. This separation between the normal location of commands within the full UI and their location within the subset palette creates a potential problem

*Corresponding author. Email: joey@cosc.canterbury.ac.nz

© 2015 Taylor & Francis
for the user, because selections from within the subset do not assist in learning of the full UI. The user’s efficiency with the subset, therefore, is dependent on the desired target item being present in the subset; and when the target item is absent, users must search the full UI, which may be unfamiliar if most selections are made in the subset. Some subset presentation techniques avoid this problem by presenting subset items in place, using the same location as the full interface (Lee and Yoon 2004; Fischer and Schwan 2008; Findlater et al. 2009). We developed two new in-place visual subset presentation techniques, called StencilMap and EphemeralMap. Both are an extension of CommandMaps, which maximise command visibility by using the entire window to make all commands available at once (Scarr et al. 2012), using a control key to display the grid of commands. The grid of commands is removed when the command key is released. StencilMaps and EphemeralMaps use different methods to emphasise the subset – StencilMaps place a static dark semi-transparent ‘stencil’ overlay over the full interface (see Figure 2), visually attenuating all but the subset items; EphemeralMaps use a dynamic technique in which subset items are immediately displayed and the remaining items gradually fade in (based on previous positive results in Lee and Yoon 2004; Findlater et al. 2009). The design intention of both systems is twofold: first, in-place presentation will facilitate development of spatial memory for command location and therefore improve selection time due to reduced need for visual search; second, command identification will be faster because subset items are more visually salient than non-subset items.

A first experiment focuses on the first of these design intentions. It compares user performance between the in-place approach of StencilMaps with that of the separate approach of palettes. As intended, results showed that spatial learning of the full UI was stronger with the StencilMap.

A second experiment examines user performance with different forms of subset highlighting: static highlighting with StencilMaps, dynamic highlighting with EphemeralMaps, and no highlighting with CommandMaps. Results showed that selections with StencilMaps and EphemeralMaps were, respectively, 42% and 19% faster than CommandMaps. However, recall of item locations had the reverse order, with CommandMaps and EphemeralMaps faster than StencilMaps. This finding emphasises a subtle and important interface design trade-off that influences spatial learning with UIs – interface methods that make selections fast, efficient, and effortless can impede spatial learning because positive mental effort improves learning outcomes. Opportunities for further work are discussed, including how StencilMaps and EphemeralMaps might be applied to real-world systems.
2. Related work

This research explores factors influencing the effectiveness of interfaces that present subsets of items to users. Several areas of prior work are relevant to these objectives, briefly reviewed in the following subsections.

2.1. Reduced-functionality interfaces

One way to help users with complex UIs is to provide them with reduced-functionality interfaces containing a subset of the application’s available commands. These subsets are usually either predetermined (Carroll and Carrithers 1984), explicitly user created (McGrenere et al. 2007; Lafreniere et al. 2011), or dynamically adapted to the user’s actions (e.g. split menus Sears and Shneiderman 1994; Cockburn, Gutwin, and Greenberg 2007; Cockburn et al. 2007).

Researchers have proposed a number of ways to present these subsets, with most involving the removal of commands not found in the subset (Shneiderman 2003; Christiernin et al. 2004; McGrenere et al. 2007; Lafreniere et al. 2011). For example, with McGrenere et al.’s (2007) two-interface model, users maintain a single subset interface that contains all the commands they are likely to use, with the option to switch to the full interface for any missing commands. Using a related approach, AdaptableGIMP (Lafreniere et al. 2011) partitions the command set according to task, providing users with a separate tool palette for each task. Commands in each task palette are ordered according to their anticipated role in the task, and there is therefore no relationship to the command’s location in the full interface.

Other systems identify a subset by disabling certain commands, such as Training Wheels interfaces (Carroll and Carrithers 1984) and ‘Multi-Layer interfaces’ (Shneiderman 2003), and others still embellish commands outside the subset (Findlater and McGrenere 2007). Finally, Stuerzlinger et al.’s (2006) Interface Facades allow users to selectively reconfigure commands within the interface according to their needs.

While studies have explored the benefits of subsets in comparison to the full interface (McGrenere et al. 2007), there are few direct comparisons of different subset presentation techniques for reduced-functionality interfaces. Findlater and McGrenere (2007) compared a control ‘full’ menu-based interface with two subset variants: one where unneeded commands are completely removed, condensing the interface (the ‘minimal’ UI) and one where they are present but marked with a small ‘x’ (the ‘marked’ UI). They found that users were more efficient with the minimal interface than the control interface, but that there was no significant difference between subset variants either when working with the subsets or on a transfer task to the full interface. In a related menu study, Fischer and Schwan (2008) found that spatial stability in menu item placement facilitated rapid selection of subset items – subset menu item selections were slower when the subset of menu items was compacted to remove vacant space left by the removal of non-subset items, even though doing so reduced target pointing distance.

Finally, Findlater and McGrenere (2010) showed that adaptive personalisation interfaces are subject to trade-offs between improvements in core task performance and impairments in the users’ ability to learn about new features.

2.2. Adaptive subset interfaces

Like reduced-functionality UIs, adaptive subset interfaces promote certain commands, but do so by dynamically altering the UI (typically after every command selection). There is extensive prior research on adaptive UIs, covering topics including studies of the algorithms underlying their predictions (Gajos et al. 2008; Fitchett and Cockburn 2012), studies of subtle external factors influencing their success such as task engagement and user age (Lavie and Meyer 2010), and interface methods for presenting adaptations to the user (briefly summarised in the following paragraphs).

One way to adaptively promote a subset is through spatial repositioning (Mitchell and Shneiderman 1989; Findlater and McGrenere 2004; Gajos et al. 2006). Examples of spatial repositioning include split menus, where likely commands are placed at the top of the menu (Findlater and McGrenere 2004; Sears and Shneiderman 1994); frequency menus, where items are ordered according to frequency of use (Greenberg and Witten 1985; Mitchell and Shneiderman 1989); and supplemental tool palettes containing the most recently and frequently used commands (Gajos et al. 2006). By presenting the subset separately from the full UI, users do not learn the location of items in the full UI when using the subset. This raises two potential problems. First, although the subset may assist in identifying core functions within the subset, it may impede the user’s development of spatial memory, which allows rapid command identification within the full UI. Second, if the subset is dynamically populated with items that the user is predicted to need, then any inaccuracy of the prediction algorithm is likely to impair the user’s performance: the user is likely to first check the subset interface, determine that the desired item is missing, and then search the full UI. If, instead, the user immediately searches the full UI, then the inaccurate subset interface is likely to fall into disuse and needlessly consume display space. The important role that algorithm accuracy serves in adaptive UIs has been demonstrated in several studies (Tsandilas and Schraefel 2005; Gajos et al. 2006, 2008).

The problems of separate subset presentation have motivated the development of various in-place UI methods for presenting subsets within the full UI. The objective of these approaches is to emphasise subset items, with two variant approaches doing so using either static or dynamic
techniques, where dynamic methods achieve their emphasis through some form of display variance over time, while static methods do not.

Gajos et al. (2006) examined a ‘visual pop-out’ interface that used static visual highlighting to emphasise the current subset by colouring items pink. In contrast, Lee and Yoon (2004) examined a dynamic menu design, where commands in the current subset appear immediately when a menu is opened (abrupt onset) and those outside of the subset appear after a delay – Findlater et al. (2009) described a similar technique, naming it ‘ephemeral adaptation’. Findlater et al.’s studies showed that both static and dynamic highlighting (ephemeral adaptation) improved performance over a control condition with no highlighting, and that ephemeral adaptation allowed faster selections than static highlighting. In Gajos et al.’s studies, the static pink highlighting was outperformed by two forms of adaptive toolbars that either copied or moved items into a toolbar palette. These results suggest that, in certain conditions, adaptive palettes and ephemeral adaptation’s dynamic highlighting allow faster performance than static highlighting interfaces. The studies presented in this paper extend these findings, establishing certain conditions in which these findings are reversed.

Other studies have confirmed and modelled the performance costs stemming from separate display of subsets, including the proposal and evaluation of a hybrid separate and in-place technique called ‘morphing menus’, where predicted items gradually enlarge to raise their prominence and assist in selection (although the model and study showed limited performance benefit Cockburn, Gutwin, and Greenberg 2007; Cockburn et al. 2007).

2.3. Interfaces exploiting spatial memory
Spatial memory is a key human capability that some interfaces exploit to help novices learn complex interfaces (Scarr et al. 2013). The spatial consistency of interfaces such as StencilMaps and EphemeralMaps is intended to allow users to build memory of command locations, which can transfer both to different subsets and to the full interface. A number of studies have argued for and demonstrated spatial memory’s positive impact on interface performance. For example, in some of the earliest UI guidelines, Hansen (1971) argued that interface designers should optimise operations through principles of display inertia (in which the ‘size and layout of the display do not change’) and control activation through muscle memory (now more commonly referred to as motor memory), which requires consistent placement. More recently, Scarr et al. (2012) found that regular users of Microsoft Office could remember command locations in the ribbon interface to within a median of 92 pixels. Their CommandMap design, which displays all commands in a rectangular grid rather than overloading screen locations, outperformed the standard ribbon design by 25% for expert use. Kurtenbach et al. (1999) described a similar concept to CommandMaps, called the ‘Hotbox’, in which menu hierarchies were flattened and concurrently displayed for efficient access.

3. CommandMaps, StencilMaps, and EphemeralMaps
The two key goals for the design of the StencilMaps and EphemeralMaps are (1) to allow novice users to quickly find and select target commands and (2) to assist users in learning the spatial location of commands to further facilitate rapid selection.

3.1. CommandMaps
CommandMaps (Scarr et al. 2012, 2014) are designed to allow extremely fast pointer-based command selection by presenting all commands at once in a grid display (when a modifier key is pressed), making all commands selectable with a single click without need to navigate through menu hierarchies. When the modifier key is released, the CommandMap disappears, leaving the entire display available for the work surface. Figure 3 shows the CommandMap implementation of Microsoft Word. Experimental studies demonstrated the efficiency of the CommandMap technique, once users had learned the spatial location of items. However, the studies also observed the tendency for novices to be initially overwhelmed by the number of commands concurrently displayed.

3.2. StencilMaps
StencilMaps are an extension of the CommandMap concept. They are intended to ease the tendency for CommandMaps to overwhelm novice users while also allowing any spatial location memory derived from subset selections.

Figure 3. A CommandMap interface for Microsoft Word.
to transfer to the full UI. This is achieved by reducing the visual salience of more advanced controls, allowing the subset of core functionality to appear with greater emphasis. StencilMaps, therefore, use a static technique to raise the visual salience of subset items by lowering the visual salience of non-subset items (see Figure 2). This is achieved by placing a dark, semi-transparent stencil overlay on top of the entire UI, with ‘holes’ in the stencil over the subset items. Other than subset highlighting, the interface for StencilMaps is identical to that of CommandMaps.

3.3. EphemeralMaps

EphemeralMaps are also an extension of CommandMaps. They have the same design goals as StencilMaps – using a subset to reduce the tendency of CommandMaps to overwhelm novice users and permitting spatial memory of subset selections to transfer to the full UI. However, unlike StencilMaps’ static presentation, EphemeralMaps use a dynamic technique to show the subset. When the user presses the invocation key, only the subset items are immediately displayed. The remaining non-subset items gradually fade into view. Our implementation followed Findlater et al.’s basic design (Findlater et al. 2009), but adapted the method to work with icon sets rather than menus, and to work with larger subsets. Our pilot studies showed that Findlater et al.’s onset time of 500 ms was too short for our larger command subsets (Findlater et al. used subsets of three items; we used subsets of up to 16 items), because users were unable to search the entire subset in 500 ms. Instead, our pilot studies suggested a linear relationship between subset size and onset time, so our implementation uses 500 ms for 4-item subsets, 1000 ms for 8 items, and 2000 ms for 16 items.

The following evaluations of CommandMaps, StencilMaps, and EphemeralMaps use implementations that support only the basic functionality required to test the hypotheses described. However, more complete implementations are technically feasible as described by Scarr et al. (2014).

4. Experiment One, comparing subset location: StencilMaps (in place) vs. Palettes (separate)

We carried out a user study to compare the performance of StencilMaps (as an example in-place subset interface) with that of the common design of using palettes of clustered items (which present the subset in a separate location). The experiment focuses on the second design goal for StencilMaps (Section 3), which is to facilitate rapid selections by assisting users in learning the spatial location of commands. We examine differences between the approaches in terms of selection time and long-term item location learning.

4.1. Procedure

Participants first completed a demographics questionnaire and were introduced to the study system. We evaluated two interfaces: a StencilMap and a CommandMap that had a separate palette labelled ‘Useful Commands’. We chose a CommandMap as the full interface for our Palette condition (rather than a menu or ribbon interface) because we wanted a fair comparison with StencilMaps: menus and ribbons are known to be slower than CommandMaps for familiar items (Scarr et al. 2012).

Our experimental interfaces were based on two real-world applications: Microsoft Access and Microsoft Project (Figure 2 shows a 16-item StencilMap for Microsoft Project; Figure 4(a) shows a 4-item StencilMap for Microsoft Access; and Figure 4(b) shows an 8-item Palette for Microsoft Project). Commercial applications were chosen to assist ecological validity of the findings – the applications provide a realistic icon set and semantic layout. Microsoft Access and Project were used because we needed participants to be relatively unfamiliar with the location of items within the interfaces. The
Microsoft Access interface was always displayed first, but the order of the StencilMap and Palette was counterbalanced between participants. The study was composed of repeated cued selection tasks – for each trial a button was displayed in the middle of the window with the text ‘Click to begin next trial’. After the button was clicked, a stimulus consisting of a target item and its textual name was displayed on the right-hand side of the screen. The participant then had to hold down <CTRL> in order to display the StencilMap or CommandMap (in the Palette condition), and select the target item. Each trial was timed from the appearance of the stimulus until successful target selection. Incorrect selections produced an audible beep. Participants were instructed to complete tasks ‘as quickly and accurately as possible’. In the StencilMap condition, participants had to select the item from the augmented CommandMap (Figures 2 and 4(a)); in the Palette condition, participants could choose to use either the CommandMap or the always-visible palette when the palette contained the target (Figure 4(b)). Since targets in the interface used a mixture of iconic and textual labels, we displayed both the icon and the name of the command as the stimulus.

With each interface, participants first performed a familiarisation block of 10 trials (data discarded), introducing them to the interface and experimental procedure. Participants then interacted with subsets of three different sizes: 4, 8, and 16 items. Each subset size condition consisted of two blocks: Subset and Post-Subset. The Subset block comprised 25 selections from a subset interface (five selections of each of five targets), some of which were within the subset, and some of which were not (see Section 4.1.1). The Post-Subset block was designed to analyse learning effects, and it consisted of the same sequence of items but without any subset highlighting – it used a full, unmodified CommandMap. More specifically, for the StencilMap interface, the Post-Subset block removed the stencil overlay, whereas in the Palette interface the separate palette was removed. Participants performed a single selection of each target used in that interface from a full, un-augmented CommandMap.

After finishing Subset and Post-Subset blocks with all three subset sizes, participants performed an incidental awareness test using Findlater and McGrenere’s (2007) measure of ‘indirect awareness’: participants were asked to perform 20 further selections of items that were not targets in the main experiment. The incidental awareness test measures learning of items that were not part of the main experiment.

4.1.1. Subset accuracy
Subset interfaces require an underlying algorithm or design decision to determine which items are contained within the subset. For each of the user’s selections, there is therefore a probability that the target item is contained within the subset UI. As the relative effectiveness of different interface methods may vary with the accuracy of the subset content (i.e. the percentage of target items that are present in the current subset) we included Accuracy as a between-subjects factor – 12 participants were exposed to a Low accuracy (40%), and another 12 were exposed to a High accuracy (80%), with these levels based on prior studies (Findlater et al. 2009).

4.2. Participants and apparatus
Twenty-four participants were recruited from a local university for the one-hour experiment: eight male, 16 female, mean age 25 years (s.d. 4.2). The study was performed on a Windows 7 PC with a 24” screen running at 1920 × 1200 resolution. Software was written in C#, and recorded all experimental data, including selections and errors.

4.3. Design and hypotheses
We analyse selection time using a 2 × 3 × 2 × 2 mixed-design ANOVA for within-subjects factors Interface {StencilMap, Palette}, Subset Size {4, 8, or 16 items}, and Block {Subset, Post-Subset}; and between-subjects factor Accuracy {High, Low}. Order of Interface and Subset Size was balanced using a Latin square.

Our hypotheses were as follows:

H1.1. StencilMap will be faster than Palette for Post-Subset trials.

Since StencilMaps do not change item locations, we expected them to facilitate location learning and therefore enable faster selections in the Post-Subset phase than palettes.

H1.2. Palette will be faster than StencilMap for high accuracy Subset trials.

The advantage of a palette is that it displays promoted commands in a compact, easy-to-search arrangement. We therefore expect Palette to be faster than StencilMap during the subset phase when prediction accuracy is high. When prediction accuracy is low, however, many trials will require searching the palette to determine the absence of the target before subsequently searching the CommandMap, leading to an expectation of no difference between the Palette and StencilMap in low accuracy trials.

H1.3. StencilMap will be faster than Palette for the incidental awareness test.

Since the StencilMap requires users to interact with the main interface, we expect that participants will develop better overall command awareness than with the palette.

4.4. Results
Error rates were low: 4.4% for StencilMap and 5.4% for Palette (no significant difference). The following analyses
Table 1. Summary of statistical results from a $2 \times 2 \times 2$ ANOVA for factors Interface, Accuracy, and Block.

<table>
<thead>
<tr>
<th>Factors</th>
<th>DFs</th>
<th>$F$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>1, 22</td>
<td>6.791</td>
<td>0.016</td>
</tr>
<tr>
<td>Accuracy</td>
<td>1, 22</td>
<td>3.63</td>
<td>0.07</td>
</tr>
<tr>
<td>Interface $\times$ Accuracy</td>
<td>1, 22</td>
<td>1.799</td>
<td>0.193</td>
</tr>
<tr>
<td>Block</td>
<td>1, 22</td>
<td>40.238</td>
<td>0.000</td>
</tr>
<tr>
<td>Interface $\times$ Block</td>
<td>1, 22</td>
<td>34.078</td>
<td>0.000</td>
</tr>
<tr>
<td>Accuracy $\times$ Block</td>
<td>1, 22</td>
<td>30.09</td>
<td>0.000</td>
</tr>
<tr>
<td>Interface $\times$ Accuracy $\times$ Block</td>
<td>1, 22</td>
<td>7.127</td>
<td>0.014</td>
</tr>
</tbody>
</table>

Note: DF, degrees of freedom.

Figure 5. Mean selection times for StencilMap and Palette for Subset and Post-Subset blocks, across Low and High accuracies. Error bars show $\pm 1$ standard error of the mean.

Figure 6. Mean selection times across item repetition in Subset and Post-Subset blocks. Error bars: $\pm 1$ s.e.m.

Statistical outcomes of the $2 \times 2 \times 2$ ANOVA (Interface, Accuracy, Block; collapsed across Subset Size) are summarised in Table 1. The most important results are highlighted in Figure 5.

Overall, StencilMap (2.64 s, s.d. 1.29) was faster than Palette (2.90 s, s.d. 1.13), giving a significant main effect of Interface ($F_{1,22} = 6.79, p = .016, \eta^2 = .23$). There was also a significant main effect of Block ($F_{1,22} = 40.2, p < .001, \eta^2 = .65$), with Post-Subset (2.38 s, s.d. 0.87) faster than Subset (3.16 s, s.d. 1.39).

Three interaction effects were significant. Importantly, there was a significant effect of Interface $\times$ Block ($F_{1,22} = 34.1, p < .001, \eta^2 = .61$), with StencilMap showing a large advantage on Post-Subset selections (Figure 5). Planned pairwise comparisons (paired T-tests, with Bonferroni correction) confirmed that StencilMap was faster than Palette in Post-Subset blocks ($p < .001$), and that Palette was faster than StencilMap in high accuracy Subset blocks ($p < .005$). We therefore accept H1.1 and H1.2.

A significant Accuracy $\times$ Block interaction ($F_{1,22} = 30.1, p < .001, \eta^2 = .58$) is best attributed to low accuracy tasks being slower than high accuracy ones during the Subset block (as expected, accurate subset membership assisted users), whereas low accuracy items were more quickly retrieved than high accuracy ones once the subset was removed in the Post-Subset block. This is likely due to the additional difficulty of finding inaccurately predicted items causing stronger memories for the location of those items.

There was also a three-way Interface $\times$ Accuracy $\times$ Block interaction ($F_{1,22} = 7.13, p < .005, \eta^2 = .24$). This may be due to performance with the Palette interface being particularly poor in the high accuracy Post-Subset condition. In contrast, as shown in Figure 5, in the Subset block, the main difference between the Palette and StencilMap techniques is in the opposite direction for high subset accuracy levels. At low accuracy, there was very little difference between StencilMaps and Palette (3.73 s and 3.78 s, respectively).

Figure 6 shows the difference in item location learning between StencilMap and Palette during Subset and Post-Subset blocks (connected with H1.2 and H1.1, respectively). While participants were 2.3 s slower with StencilMap on the first selection of each item in the Subset block, they performed much faster (3.7 s) in the Post-Subset block.
4.4.3. Incidental awareness

For the incidental awareness test, a $2 \times 2$ ANOVA of Interface and Accuracy on selection time showed no significant effects, with mean selection times of 9.3 s (s.d. 2.7) with StencilMap and 10.3 s (s.d. 4.4) with Palette: $F_{1,22} = 1.0$, $p = .3$. We therefore reject H1.3.

5. Experiment Two, comparing subset highlighting method: none, StencilMap’s static, and EphemeralMap’s dynamic highlighting

The aim of Experiment Two was twofold: first, to determine whether subset highlighting improves performance compared to that of a non-highlighted CommandMap; and second, to compare user performance with the static highlighting approach of StencilMaps with that of the dynamic highlighting approach used by EphemeralMaps (which is based on delayed visual onset; Lee and Yoon 2004; Findlater et al. 2009).

5.1. Experimental conditions

The experiment compared item selection time when using a CommandMap (Figure 7(a)), a StencilMap (Figure 7(b)), and an EphemeralMap (Figure 7(c)). The CommandMap was included because we needed to isolate any performance benefit stemming from the layout of the CommandMap from that arising from stencil-based highlighting.

CommandMaps. The baseline technique showed all 80 commands of the interface at once when the user pressed a control key (Figure 7(a)). The commands were shown until the user selected a command with the mouse.

EphemeralMaps. As described in Section 3.3, this method was based on ephemeral adaptation (Findlater et al. 2009) in which predicted menu items are shown immediately, while unpredicted ones are faded in after a short delay. Figure 7(c) shows the EphemeralMaps implementation with four subset items shown and the remaining items partially faded into view.

StencilMaps. The StencilMap interface behaved as for Experiment One, placing a dark translucent overlay over all items except those within the subset (Figure 7(b)).

Note that the colour and transparency of the stencil overlay is a design parameter — we chose a dark and translucent overlay such that the highlighted objects were visually prominent, while also allowing non-highlighted items to be discerned. Also note that an EphemeralMap can be considered to be a StencilMap that has a function governing the temporal fade-out removal of its stencil. We have not yet empirically analysed the impact of different settings for stencil colour or its initial transparency.
experimental CommandMap-based interfaces shown in Figure 7. For each trial, a button was displayed in the middle of the experimental window with the text ‘Click to begin next trial’. After the button was clicked, a textual stimulus was displayed on the right-hand side of the screen, beginning the trial. The user had to hold down <CTRL> in order to display the CommandMap, and then select the target item. Each trial was timed from the appearance of the stimulus until successful target selection. Incorrect selections produced an audible beep. Participants were instructed to complete tasks ‘as quickly and accurately as possible’.

To reduce learning effects between interfaces (a within-subjects factor), we used three separate icon sets. Each icon set contained 80 items, and was randomly partitioned into five sub-categories. To avoid potential confounds caused by visual pop-out effects, we used grayscale icons with a uniform visual style. While Experiment One was based on commercial software systems, we used synthetic icons in Experiment Two to reduce potential confounds associated with differential visual saliency of target items. All participants saw the icon sets in the same order, but order of interface was counterbalanced to reduce risk of confounds caused by specific icon sets. Subsets and selection sequences for each icon set were determined using a one-off random process, and were the same for all participants.

Within each interface, participants performed one familiarisation block of 10 trials (data discarded), using different subsets and target items than in the main experiment. This was followed by two Subset blocks of 25 trials each, for three subset sizes (four, eight, and 16). Each Subset block contained five selections of each of five target items. Four out of five targets in each block were in the highlighted subset. This 4/5 (80%) subset accuracy level was chosen based on Findlater et al.’s (2009) results, which showed that at low accuracy (50%), ephemeral adaptation did not offer performance benefits over a full interface, likely because participants stopped attending to the highlights. Therefore, we chose to focus the study on situations where the dynamic highlighting provided by ephemeral adaptation has been shown to be successful.

Each interface was followed by a recall test, where participants performed a single selection of each of the 30 targets used in that interface from a full, un-augmented CommandMap. We included this block to examine whether earlier trials using StencilMap’s static highlighting and ephemeral adaptation’s dynamic highlighting caused measurable differences in retrieval time, which might be attributed to their support for the formation of spatial memory for item locations. Short breaks were given between blocks. Participants filled out an abbreviated questionnaire based on the NASA-TLX after each interface condition, and ranked the three interfaces at the end of the experiment.

5.3. Participants and apparatus

Eighteen participants were recruited from a local university (eight male, 10 female; mean age 26.4 years) for the one-hour experiment. The study was performed on a Windows 7 PC with a 24” screen running at 1920 × 1200 resolution. The study software was written in C#, and recorded all experimental data, including selections and errors.

5.4. Design and hypotheses

Our primary analysis used a 3 × 3 RM-ANOVA on the Subset blocks, with two within-subjects factors: Interface {CommandMap, EphemeralMap, StencilMap} and Subset Size {4, 8, 16}. Both factors were counterbalanced between participants using a Latin square. We also planned a follow-up analysis, separating trials into in-subset and out-of-subset groups and performing separate 3 × 3 RM-ANOVAs on each group. Finally, we planned a one-way ANOVA on the recall blocks, with Interface as the only factor.

Our main hypotheses were as follows:

H2.1. For in-subset selections, both highlighting techniques will be faster than CommandMap, and StencilMap will be fastest.

Since both StencilMaps and ephemeral highlighting reduce the visual search space by raising the relative visual prominence of subset items, we expect both interfaces to be faster for in-subset selections than the CommandMap. We also expect the StencilMap to outperform EphemeralMap due to its static and continuous subset presentation; ephemeral highlighting, in contrast, allows only limited time to search the subset.

H2.2. For out-of-subset selections, StencilMap will be faster than Ephemeral, but both highlighting approaches will be slower than no highlighting (CommandMap).

StencilMaps offer an advantage over EphemeralMaps when selecting out-of-subset items because ephemeral highlighting delays their appearance through gradual onset. However, this is a risky hypothesis because the dark transparent mask used by StencilMaps may impair visually search for out-of-subset items. Further, we hypothesise that out-of-subset visual search will be fastest with no highlighting (CommandMaps) due to reduced distraction when searching for non-highlighted items.

H2.3. Users will subjectively prefer StencilMap over EphemeralMap and CommandMap.

EphemeralMaps may frustrate users if they cannot find items before the end of the onset timeout, and users may find it difficult to locate items in a CommandMap with no visual pop-out. We therefore expect StencilMaps to be subjectively rated higher than CommandMaps and ephemeral highlighting.

H2.4. Performance on the recall test will be similar with the three conditions.
Although this hypothesis is unusual in seeking to accept a null hypothesis, we include it to express our anticipation that the important issue of spatial memory for item locations will be similar with all three interfaces, which share the same spatial layout.

5.5. Results

Overall error rates were low in all conditions (2.4% for StencilMap, 2.7% for CommandMap, and 3.3% for EphemeralMap). The analysis below excludes error trials.

5.5.1. Selection times overall

As shown in Figure 8 (left), overall mean selection times were faster with StencilMap (2.52 s, s.d. 0.79) than with EphemeralMap (3.51 s, s.d. 1.42) and CommandMap (4.36 s, s.d. 1.20), giving a significant main effect of Interface ($F_{2,34} = 45.7, p < .001, \eta^2 = .73$). Pairwise Bonferroni-corrected comparisons ($\alpha = .05$) showed that StencilMap was significantly faster than both CommandMap and EphemeralMap overall. Although there was a significant effect of Subset Size ($F_{2,34} = 12.78, p < .001, \eta^2 = .43$) the influence of size on performance was consistent across interfaces, leading to no significant Interface × Subset Size interaction ($F_{4,68} = 1.0, p = .42$).

5.5.2. In-subset and out-of-subset selections

The overall results reported in Section 5.5.1 are sensitive to the experimental distribution of in-subset and out-of-subset trials. For example, Figure 8 shows the different relative efficiencies of the interfaces for in-subset and out-of-subset trials, with CommandMap slowest for in-subset trials, but fastest for out-of-subset trials. In anticipation of this effect, our main hypotheses H2.1 and H2.2 concern separate analyses of in-subset and out-of-subset trials (Section 5.4).

For in-subset trials (Figure 8, middle), there was a significant main effect of Interface ($F_{2,34} = 63.6, p < .001, \eta^2 = .79$). Bonferroni-corrected comparisons showed that StencilMap (mean 2.09 s, s.d. 0.7) was faster than both EphemeralMap (3.2 s, s.d. 1.4) and CommandMap (4.6 s, s.d. 1.2). We therefore accept H2.1 – the static highlighting provided by StencilMaps outperformed the dynamic highlighting of EphemeralMap.

Out-of-subset selections involve targets that are not in the highlighted set for EphemeralMap or StencilMaps. Figure 8 (right) shows that the mean selection times for the non-highlighting CommandMap was lower than for either of the two highlighting techniques. Although there was a trend suggesting a possible effect of Interface ($F_{2,34} = 2.8, p = .07$), the effect did not reach significance. We therefore reject H2.2. However, this only infers that incorrect highlights did not have a reliably negative effect on visual search. This may be due to a lack of statistical power, and additional work should be done to determine the potential distraction effect of incorrect highlights.

5.5.3. Benefits of subset highlighting for novices users

As shown in Figure 9, the greatest benefits of both types of highlighting are on the first selection of each highlighted item, when the user is completely unfamiliar with the item’s location. This confirms the first design goal for StencilMaps and EphemeralMaps (Section 3) – that the highlighting allows novice users to quickly find and select target commands. StencilMap was faster than EphemeralMaps and CommandMap throughout the repeated trials, although selections times approached convergence on the final trial (see Figure 9, lower chart).

5.5.4. Completion time in the recall task

Recall tasks involved selecting items from a CommandMap interface that lacked any augmentation in terms of stencils or highlighting. A one-way ANOVA on overall recall
times (Figure 10) showed a significant main effect of Interface ($F_{2,34} = 4.6$, $p = .02$, $\eta^2 = .21$), with StencilMap (4.44 s, s.d. 2.00) slower than EphemeralMaps (3.60 s, s.d. 1.82) and CommandMap (3.59 s, s.d. 2.14). Pairwise Bonferroni-corrected comparisons showed no significant difference between EphemeralMap and CommandMap ($p = .96$), but a trend suggests that participants were faster at recalling items with CommandMap than with StencilMap ($p = .053$). There was also a significant difference between EphemeralMap and StencilMap, with EphemeralMap being the faster of the two techniques ($p = .01$). We therefore reject H2.4. We anticipated that the three interface techniques would be roughly equivalent in their support for forming spatial memories for item locations, yet the rejection of H2.4 indicates that this is not the case. While StencilMap was faster than CommandMap and EphemeralMap for in-subset selections during the main block of selections (Figure 8), it was slower in recall tasks. It appears that something in the StencilMap interface caused users to learn the items’ spatial locations less well than with EphemeralMap and CommandMap. We return to this issue in Section 6.

5.5.5. Subjective responses

User response to StencilMaps was overwhelmingly positive, with StencilMap scoring significantly higher on five of the six NASA-TLX measures (Table 2). Sixteen of 18 participants stated that they preferred StencilMap, 1 EphemeralMap, and 1 did not answer. We therefore accept H2.3.

5.6. Summary

We summarise Experiment Two’s results according to our hypotheses:

H2.1. For in-subset selections, both highlighting techniques will be faster than CommandMap, and StencilMap will be fastest. Supported. The correct highlighting provided by both EphemeralMap and StencilMap resulted in faster selections than CommandMap. More interestingly, the static highlighting provided by StencilMaps outperformed the dynamic highlighting provided by EphemeralMap. This latter finding seems to contradict that of...

6. Discussion

To summarise the results of both experiments, Experiment One showed that the in-place subset provided by StencilMaps allowed for better location learning than the separate subset palette approach. Furthermore, the benefits of the palette approach when selecting accurately predicted items within the subset were negated when subset predictions were inaccurate.

Experiment Two showed that users are able to locate unfamiliar items more quickly when they were highlighted, and that the static highlighting provided by StencilMaps was more effective at reducing search time than the dynamic highlighting of EphemeralMaps. Furthermore, participants strongly preferred StencilMaps. However, when the highlighting interface was removed in the recall part of Experiment Two, participants were slightly slower at selecting in-subset target items following earlier selection with the StencilMap, suggesting that its...
use induced weaker spatial memories than with CommandMaps and EphemeralMaps.

6.1. Why were recall selections slower after using the StencilMap in Experiment Two?

StencilMaps are designed to achieve two key goals — faster selection of subset items by novices and improved spatial location learning to facilitate expertise. The negative results for StencilMaps in the recall task of Experiment Two are therefore a concern.

There are two likely reasons for this result. First, the dark and light areas of the stencil create a new set of perceptual landmarks that users can exploit to remember commands in the subset — for example, a user might remember ‘the bright command at lower left’ or ‘the middle of three highlighted commands in a row’. When the stencil is removed, these artificial landmarks disappear as well, and if the user had been relying on them, it can cause difficulties as they switch back to their memory of the interface’s reference frame.

Second (and related to the idea above), the result can be explained with respect to the guidance hypothesis of skill acquisition (Schmidt 1991). The guidance hypothesis suggests that augmented feedback, which improves early task performance through additional guidance, may impair retention of the performed skill once the guidance is removed.

With respect to this hypothesis and framework, it is plausible that the effective highlighting provided by StencilMaps (e.g. the artificial landmarks of the stencil itself) induced more ‘shallow’ encoding of item locations because the in-subset items were almost immediately ready to hand and identifiable, whereas items in the CommandMap required extensive visual search, prompting participants to think more carefully about target item locations. Related observations on the role of effort in learning the spatial location of UI objects have been made in previous studies (Ehret 2002; Cockburn, Gutwin, and Greenberg 2007; Cockburn et al. 2007). Another plausible explanation is that the better recall with CommandMap was simply an artefact of the longer ‘training’ period that participants had with items, stemming from their slower acquisition times.

Interestingly, this implies that there may be an inherent trade-off between making things easier for novices and supporting long-term application learning. There are extensive opportunities for further work in examining how assistive highlighting may be gradually tailored to best facilitate the dual purposes of efficient novice interaction and early attainment of expertise (Anderson and Bischof 2013).

6.2. Generalisation and limitations

While StencilMaps performed favourably in both experiments, there are experimental limitations that could affect results generalisability, as follows.

6.2.1. Sensitivity to onset time

As discussed prior to Experiment Two, the performance of EphemeralMaps is sensitive to onset time. There is therefore a risk that the onset times we chose (500, 1000, and 2000 ms) were not optimal for the experimental task: they may have been too short, not allowing enough time for participants to search the visual subset; or too long, unnecessarily delaying participants in finding non-predicted items. We counter this concern in two ways. First, mean selection times with EphemeralMaps were higher than StencilMaps for both in-subset and out-of-subset selections. Since adjusting ephemeral onset time is necessarily a performance trade-off between in- and out-of-subset items, changing the onset time in either direction is unlikely to have resulted in an overall win for EphemeralMaps. Second, the need to tweak onset time is an inherent weakness of the ephemeral technique: the optimal onset time is dependent on the size and distribution of the subset, as well as the abilities of the user. StencilMaps, which are permanent, avoid this problem, easing real-world implementation.

6.2.2. Source of subset commands

Command subsets can be populated from many different sources. For example, an adaptive system dynamically builds subsets based on predictions about the user’s upcoming actions; and palettes in real-world interfaces can often be personalised to match the user’s evolving needs.

However, in our two experiments, we only considered static, predefined subsets. We did this for two reasons. First, we are primarily interested in interfaces that assist novice users in gaining expertise and, in this context, pre-populated ‘training wheels’ interfaces have shown promising results (Carroll and Carrithers 1984). We envisage a system where the StencilMap subset can be populated with relevant commands for specific tasks or workflows, similar to those used in AdaptableGIMP. Novice users can turn on the stencil as a complexity-reducing tool while they learn how to perform a new task; they can then turn off the stencil once it is no longer needed, keeping their newly found spatial knowledge intact. Second, we discovered in pilot testing that subset interfaces have many subtle attributes that can affect performance; we therefore wanted to test the simplest possible case.

6.3. Demonstrating StencilMaps in the real world

To investigate how StencilMaps could be deployed in a real application, we developed a prototype version of the GIMP image editor that incorporates both CommandMaps and StencilMaps. This prototype uses Metisse (Chapuis and Roussel 2005) and the Linux accessibility application programming interface to provide a CommandMap interface to the application’s menus and toolbars. As seen in
Figure 11, StencilMap subsets can be placed on top of the command map (e.g. task subsets as with AdaptableGIMP, or a training wheels subset). This prototype shows that existing applications can be adapted to CommandMaps and StencilMaps, even if they are not initially designed for the techniques.

A second issue to be considered in deploying the stencil approach in real-world systems is whether the technique can work with interfaces that do not make the entire subset visible – e.g. some commands may be hidden behind menus or ribbons. Stencil subsets can still be used in these interfaces: the stencil simply highlights the titles of any menus or ribbons that contain items of the subset, and the user must open the hierarchical control to find those items.

6.4. Divergence from prior findings

Section 2.2 summarised a previous result from Gajos et al. (2006) which showed that target selection performance was slower with a visual pop-out interface (a form of static highlighting) than with different forms of adaptive toolbars (largely equivalent to our palettes). The results from our Experiment Two extend this result, confirming that an adaptive palette can improve immediate item selection by placing items ready to hand, but also demonstrating that this efficiency improvement requires a high degree of predictive accuracy. Furthermore, recall task results demonstrate an expected limitation of palettes in their weaker support for knowledge transfer to the full UI, but also a performance advantage for ephemeral’s dynamic highlighting over static visual highlighting. Their results also showed that incorrect highlighting slowed selections compared to no highlighting, and that participants subjectively preferred static highlighting over ephemeral. The results from our Experiment Two confirm the performance benefit that highlighting offers over non-highlighting interfaces when selecting predicted items. They also confirm the performance detriment caused by incorrect highlighting and the subjective preference for static highlighting over dynamic highlighting and no highlighting. However, our results on the relative performance of ephemeral highlighting and static highlighting contradict those of Findlater et al – StencilMaps (static) outperformed EphemeralMaps (dynamic).

One key difference between our experiment and Findlater et al.’s is that we controlled certain levels of subset size (4, 8, and 16), whereas they used a fixed subset size of 3 items. Prior to the experiment we anticipated that ephemeral highlighting might become less effective with increasing subset sizes (based on psychology research on abrupt visual onset; Yantis and Jonides 1984); however, the results shown in Figure 10 indicate a consistent performance benefit for StencilMaps’ static highlighting across all levels of subset size. An alternative explanation concerns the total size of the search space (or the ratio between subset size and total size), which was 16 items in Findlater et al.’s study, but 80 in ours. The difference in the difficulty of visual search in the two experiments is reflected in the mean task times, which was $\sim 1.25$ s in Findlater et al., but $\sim 2.75$ s in ours (more than twice the duration). We suspect this attribute of the study may have caused the
results’ divergence, but further study is needed. Another possible cause for the divergent results, also requiring further study, is differences in the saliency of the static highlighting conditions—our experiment used a static stencil to lower the saliency of all but subset items, whereas Findlater et al. used a light purple background for subset items.

7. Conclusions and future work
Subset interfaces can make it easier for novices to interact with complex interfaces. However, subset interfaces that spatially rearrange controls can hinder novices’ transition to the full UI, because the user must learn new command locations in the full interface. To address this problem, we presented two new types of interface called StencilMaps and EphemeralMaps, which combine a spatially stable CommandMap interface with subset highlighting that is either static (StencilMaps, which use a stencil overlap) or dynamic (EphemeralMaps, which use fade-in effects).

Our first experiment compared the performance of StencilMaps to palette-based subsets in a realistic Microsoft Office interface. Results show that users were slightly faster with palettes than StencilMaps in the subsets, but only when the accuracy of the subset prediction was high. As expected, StencilMaps led to much faster performance once the subset was removed.

We then evaluated the compared command selection performance with three different versions of subset highlighting: CommandMaps, which provide no highlighting; StencilMaps, which use static highlighting; and the dynamic highlighting of EphemeralMaps. Results confirmed that subset highlighting improves selection times. The static highlighting of StencilMaps was faster than EphemeralMaps for in-subset selections, with no significant difference for out-of-subset selections. Users expressed a strong preference for StencilMaps.

Results from the two studies indicate that the spatial stability of StencilMaps assists faster command identification in the full UI than palettes, and that StencilMaps’ static highlighting outperforms the dynamic highlighting of EphemeralMaps.

Our experiences with StencilMaps indicate two main directions for future work. First, we will refine the StencilMap capabilities of the existing CommandMap version of the GIMP editor. We can then experiment with stencil-based presentation of subsets (both task-based and training-wheels-based) in real usage scenarios, and carry out longer-term comparisons of palette-based and stencil-based approaches. We also plan to release this version of the GIMP for wider public evaluation. Second, we will develop a stencil version of a menu-based interface, and examine the performance and learning effects of stencils in hierarchical interface organisations.

Conflict of interest disclosure statement
No potential conflict of interest was reported by the authors.

References


