IconHK: Using Toolbar Button Icons to Communicate Keyboard Shortcuts

Emmanouil Giannisakis[1] **Gilles Bailly**[1,2] **Sylvain Malacria**[3] **Fanny Chevalier**[3]
[1]LTCI, CNRS, Telecom ParisTech, Université Paris-Saclay, Paris France
[2]Sorbonne Universités, UPMC Univ Paris 06, CNRS, ISIR, Paris, France
[3]Inria, France
firstname.lastname@{[1]telecom-paristech.fr,[2]upmc.fr,[3]inria.fr}

ABSTRACT
We propose a novel perspective on the design of toolbar buttons that aims to increase keyboard shortcut accessibility. IconHK implements this perspective by blending visual cues that convey keyboard shortcut information into toolbar buttons without denaturing the pictorial representation of their command. We introduce three design strategies to embed the hotkey, a visual encoding to convey the modifiers, and a magnification factor that determines the blending ratio between the pictogram of the button and the visual representation of the keyboard shortcut. Two studies examine the benefits of IconHK for end-users and provide insights from professional designers on the practicality of our approach for creating iconsets. Building on these insights, we develop a tool to assist designers in applying the IconHK design principle.

Author Keywords
Icons; keyboard shortcuts; hotkeys; GUI design.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
Toolbar buttons are frequently-used widgets for selecting commands. They are designed to occupy little screen real-estate, yet they convey a lot of information to users: the icon is directly tied to the meaning of the command, the color of the button informs whether the command is available or not, and the overall shape and shadow effect together afford a point & click interaction to execute the command. Because they are concise and convenient, toolbar buttons have become flagship widgets in graphical user interfaces (GUIs).

Most commands can also be selected by using an associated keyboard shortcut. Keyboard shortcuts enable users to reach higher performance than selecting a command through pointing and clicking, especially for frequent actions such as repeated "Copy/Paste" operations. Despite these advantages, many experienced users continue to use toolbar buttons over keyboard shortcuts [20] which has serious performance implications [9]. Several reasons can explain this behavior [9]. Users might not be aware of this modality, they might not foresee the gains in efficiency, or they might not be ready to make the extra effort to learn it. Even when users are eager to learn a keyboard shortcut, they currently have to navigate in a hierarchical menu to expose the key combination and to explicitly memorize it for future use. In other words, selecting a command through its keyboard shortcut is not as accessible as by pointing and clicking on a toolbar button, which might also contribute to shortcuts' underuse.

In this work, we aim to increase shortcut accessibility by reinforcing the relation between toolbar button icons and keyboard shortcuts. While it is generally assumed that keyboard shortcuts rely on recall, we demonstrate that they can also rely on *recognition*. Thus, we advocate for an integrated approach to icon design that encourages designers to consider the possibility of visually encoding keyboard shortcuts within icons, so that toolbar buttons inform about this modality.

We propose such an approach, IconHK, that blends visual cues conveying keyboard shortcuts within the toolbar buttons to enable visual recognition of the shortcut while not denaturing the pictorial representation of the command (Figure 2)[1]. IconHK aims to fulfill the following design challenges: 1) convey the shortcut key combination; 2) convey the meaning of the command in the icon; 3) maximize shortcut exposure duration; 4) minimize the visual space used to convey shortcuts; and 5) maintain the overall aesthetic appeal of the application. Our primary contribution is thus to offer a novel perspective on the icon design of toolbar buttons.

Our second contribution lies in a better understanding of the design space of IconHK, as well as an assessment of the potential, limitations, and challenges of our approach for both end-users and designers. A theoretical analysis of the dimensions of IconHK results in a set of three design strategies to embed the hotkey (or letter symbol) that can be displayed in an *empty space*, derived from the *positive space* of the pictograph (i.e. its silhouette or salient features), or derived from the icon's *negative space*. We also introduce a space-efficient visual encoding of the modifiers in the corners of the button. IconHK builds on a notion that we define as *magnification*, i.e. the factor that determines the blending ratio between the traditional pictograph of a button and the most explicit and legible representation of the keyboard shortcut symbol.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CHI 2017, May 06-11, 2017, Denver, CO, USA
©2017 ACM. ISBN 978-1-4503-4655-9/17/05
$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025595

[1]Figures 2, 4 and 5 are dynamic/interactive with Adobe Acrobat.

We conducted studies involving the two target populations of IconHK. A first study examines the benefits of IconHK for *end-users* by assessing the hotkey retrieval effectiveness of the different strategies. Results suggest that positive and negative space strategies are useful mnemonic aids. A second study provides insights from professional *designers* on the practicality of the IconHK approach. Results indicate that the empty space strategy is preferred for its simplicity and suitability for minimalistic styles, and point to design challenges, especially to maintain iconset consistency, for the two other strategies. Based on our findings on the design process, we implement *IconHKMaker*, a tool to help designers find inspiration to augment icons with the IconHK principle. Given an icon and a letter, the system suggests the best affine transformation to embed the hotkey. The system can also suggest candidate hotkey symbols and corresponding transformations.

DESIGN CHALLENGES AND RELATED WORK

The concept of *affordance* [12, 13] refers to all the possible actions suggested by the environment to an actor. It has been referred to as a design guideline for GUIs by Norman, who stated that computer applications should provide “strong visual cues to the operation of things” [32]. Affordance is well illustrated by a toolbar button: not only does its icon suggest the meaning of the associated command, its aspect also conveys that it can be activated by clicking on it. What a toolbar button does not afford, though, is the alternative modality to execute the command, i.e. the keyboard shortcut — a combination of keys, typically a *modifier* key (e.g. ctrl, alt) and a *hotkey* (usually a letter). With IconHK, we aim to augment toolbar buttons with the capacity to suggest this other modality through a better exposure of the keyboard shortcuts. We identify five design challenges related to this goal.

Challenge 1: Convey the keyboard shortcuts

The first challenge lies in effectively conveying the keyboard shortcuts, that is, explicitly expose their existence to users. The advantage is to allow users who do not know or recall the shortcuts to easily retrieve them.

Two main approaches have been adopted in most applications to communicate keyboard shortcuts. One displays key combinations along with menu items. In this case, users are exposed to shortcuts only after navigating the hierarchy of menus, which demands extra effort compared to clicking on a toolbar button. The other augments the toolbar with a tooltip mechanism that prompts the shortcut after hovering over a button, but at the cost of waiting until the tooltip is revealed—if it exists, which users have no means to know unless they try and wait—and requiring that users repeat this action several times to inspect multiple shortcuts.

Both of the above approaches rely on *feedforward*, which consists in presenting users with the information before they execute the command. Additional examples include Cheat Sheet [1] that extracts all shortcuts and corresponding commands in an application and presents them in a list, and ExposeHK [24], that exposes keyboard shortcuts in menus, toolbars or ribbons as soon as a modifier key is pressed.

Other works proposed a *feedback* strategy, where the information about the shortcut is communicated upon execution of a command, playing the role of a recommendation for the next time. The Hotkey-Eve application [3] is one example: each time a menu item is selected using the mouse, the corresponding keyboard shortcut is indicated in the top right corner of the display. HotkeySkillometer [25] goes a step further, and displays information regarding overall user performance along with key combinations, encouraging their use. Grossman et al. provide feedback through audio, after the selection of a menu item to expose users to shortcuts [15].

In this latter work, Grossman et al. also introduced a *cost-based* approach, which disables the menu in order to force users to use shortcuts. HotkeyCoach [19] combines the feedback and cost-based approaches: after every mouse-based command selection, a pop-up window appears to show the corresponding hotkey. Users cannot continue until they execute the keyboard shortcut or close the pop-up window.

IconHK encompasses feedforward and feedback strategies to expose keyboard shortcuts with minimum effort of the users.

Challenge 2: Maximize shortcuts exposure duration

The second design challenge lies in maximizing the shortcuts exposure duration. Ideally, the visual aid to recall shortcuts should always be visible, but in practice, such aid is usually transient to save screen real-estate.

Transient methods generally display the visual aid on demand. It is typically bound to contextual feedforward approaches such as tooltips or menu aids. Approaches that provide global aid, off-context, usually require users to perform explicit operations to show/hide information. This is the case with ExposeHK [24] and Cheat Sheet [1], where the visual aid is displayed as long as a modifier key is held.

A few *permanent* methods which continuously display command-shortcut mappings have also been proposed. For instance, Hopper Disassembler [2] uses the extreme approach of only displaying the keyboard shortcuts on the toolbar buttons themselves, to the detriment of graphical icons for which there is no longer room to be shown. This solution is acceptable in this particular case because the shortcuts always involve the first letter of the command name—which is rarely possible to achieve in feature-rich applications.

Too many operations to access visual aids and/or hide them slow down the interaction and break workflow. This can yield a performance dip [35] that discourages shortcut use and traps users in pointer-based “beginner mode” [9]. ExposeHK [24] overcomes this issue as it does not require extra operations to show shortcuts. On the other hand, these effortful and time-consuming processes can sometimes be perceived as an incentive to motivate users to learn shortcuts [19].

IconHK aims at maximizing exposure duration while striking a balance between communicating shortcuts effectively, not being disruptive and minimizing effort. To this end, it supports both transient and permanent exposure, offering different tradeoffs between legibility and exposure duration.

Challenge 3: Minimize visual space to convey shortcuts
Displaying key combinations requires visual space, which is not always possible nor desirable. Limited screen real-estate is the main reason why the above techniques were developed.

Several approaches explicitly make up space to fit shortcuts. For instance, menus are widened to append visual aids to menu items, but more complex combinations tend to result in too large menus. Appert et al. [4] use strokes as command shortcuts and display gesture cues aside menu items, which is more space-efficient, since each stroke has a fixed size[2].

Another approach consists in temporarily covering up part of the GUI to display the shortcut, as do tooltips or ExposeHK. The case of ExposeHK [24] is interesting, as it successfully defines a mechanism to pop-up relevant information, but fails to display the information without masking a part of the GUI (either the icon or another control in the vicinity). IconHK improves and compliments prior work by proposing different strategies to blend keyboard shortcuts *within* the pictograph.

Several approaches in the literature propose to blend information with visual elements already in place without obfuscating them. For instance, MiME [18] suggests mid-air static hand postures by highlighting a shape similar to the posture within the command names or icons. Animated icons have also been designed to more explicitly convey the meaning of a command by previewing its result [5], and to demonstrate how to perform a complex task [8]. FatFont [30], though less related, is a worth mentioning clever example of space-optimized font (digits only), where the amount of dark pixels in a numeral character is proportional to the number it represents and where multi digits numbers are nested so that every number occupies the same visual space as a single digit.

IconHK builds on the latter category of approaches consisting in blending new information without perturbing the layout.

Challenge 4: Convey the meaning of the commands
IconHK shares the primary goal of existing computer icons: communicating the meaning of commands. To this end, an icon should be 1) semantically related to the meaning of the command, 2) be comprehensible and 3) be distinguishable from other icons. In other words, users should still be able to perceive the characteristics of the icon and to interpret its meaning even after encompassing the shortcut information.

Many taxonomies [22, 23, 27, 31] aim at characterizing the similarity between the graphical representation of an icon and the *meaning* of the associated command. For instance, Lodding [22] distinguishes three types of semantic relations: representational, abstract and arbitrary. Representational icons rely on typical and intuitive objects to represent the command meaning. Abstract icons are composed of geometric shapes, whereas arbitrary icons do not have intuitive connection between the icon and the meaning of the command.

[2]Note that Apple has introduced special symbols associated to modifier keys, allowing for a more concise display (e.g. ⌥⇧⌘S for Alt+Shift+Cmd+S), that never exceeds 5 characters.

The impact of icon characteristics on visual search has also been investigated [14, 26]. These characteristics include concreteness, complexity, distinctiveness, size and shape which are tied to comprehensibility and distinguishability. In particular, the visual complexity of icons increases search time even after many trials, because “it requires extra processing time to bind all of the features in the icon together to form a percept” [26, 27]. The size of the icon also has a strong effect on visual search, especially if the icon is smaller than 0.7° of visual angle [21].

IconHK aims at embedding both the information about the keyboard shortcut and the pictorial representation of the command within the same toolbar button. The main challenge with this goal is to guarantee that the toolbar icons remain comprehensible and distinguishable.

Challenge 5: Maintain the aesthetic appeal of the icons
Besides communicating the meaning of the commands involved in a software, toolbar icons contribute to the overall aesthetic of the interface, which is a critical factor of user experience [23]. Aesthetic appeal is related to many criteria and is dependent on users personal perception: two users might have different opinions on how to improve the design of an icon [23]. The familiarity and the complexity of the icon can easily impact the aesthetic appeal of the icon [28]. While the aesthetics decisions related to the visual aspect of the pictorial representation of the command are left to the designer, IconHK sets out to be as undestructive as possible so as to preserve as high aesthetic appeal as the original visuals.

THE ICONHK APPROACH
We address the above challenges with a novel approach, IconHK, that blends visual cues conveying shortcut information with the traditional toolbar buttons without denaturing the pictorial representation of the command. This section first presents the overall design basics for creating enriched icons. In a second part, we introduce the notion of *magnification* as the factor that determines the blending ratio between the traditional pictograph of a button and the most explicit and legible representation of the keyboard shortcut symbol.

In the following, we refer to *button* as an interactive control to execute a command, whose *icon* occupies a bounded physical space of a button (typically a toolbar button); *hotkey* as the character of a keyboard shortcut (which is generally a letter), and *modifiers* as its modifier key combination (respectively S and ctrl for the ctrl+S keyboard shortcut of Save) of a command; *pictograph* as the pictorial element of a button's icon that conveys the meaning of the command (e.g. the floppy disk for a Save command button); and *symbol* as the embedded hotkey letter (e.g. S for the ctrl+S keyboard shortcut of the Save command).

Embedding keyboard shortcuts into toolbar buttons
Our initial goal consists in embedding visual cues conveying the hotkey and modifiers in a button, which means embedding the symbol and indicators about the modifiers while preserving the aesthetic and legibility of the pictograph.

Conveying the hotkey

Inspired by logo design practices [36], we propose three strategies to embed a symbol in a button's icon:

EMPTY SPACE consists in leveraging the blank space in or around the pictograph to display the symbol. For instance, only 23% of the pixels of the icon are devoted to the pictograph, which leaves enough room at the top-left of the button to integrate the P letter. The icon has a large empty space within its pictograph which is sufficient to insert the symbol C. The strategy is limited to icons with few painted pixels, or containing large uniform areas. Several icons such as or do not afford large enough of a space to incorporate a symbol as is.

POSITIVE SPACE consists in revealing the symbol from the silhouette or the most salient features (i.e. edges) of the pictograph. This strategy is often employed by logo designers to seamlessly blend text and images in a single visual or use objects of particular shapes to evoke letters [36]. Transposed to icons, this strategy is best illustrated by the scissor icon of the Cut command whose pictograph's shape resembles the X letter of the corresponding shortcut. It is worth noting that, in many applications, communicating the hotkey does not seem to be a primary objective. Perhaps for aesthetic reasons, different orientations of scissors are common, e.g. , which makes the perception of the hotkey more difficult. Ideally, the symbol derived from the pictograph's overall shape or edges should be as straight as possible to ensure that the augmented icon best communicates both the meaning and the hotkey. The icon is another example where the hotkey Q or O can easily be derived from but, as with empty space, not all icons qualify for this approach. Some pictographs do not easily support an encoded symbol through edge accentuation, such as .

NEGATIVE SPACE consists in exploiting the open space around an object to disguise a letter. This visual effect—popular in logo design [36]—builds on figure-ground ambiguity that creates a visual that affords two alternative viewpoints, a common illusion technique that stems from Gestalt's principles [37]. A well-renowned example is the Rubin's vase [34] where the white positive space forms a vase, while the black negative space forms two faces about to kiss. Transposed to icons, closed letters such as D or O can be easily dissimulated in the negative space of an icon whose pictograph resembles a round shape. In contrast, the illusion becomes more difficult to achieve with open letters (e.g. 'J', 'E'), since pictographs are usually not confounded with icons borders (hence the negative space cannot result in an open letter). Another cognitive mechanism can help in these cases: the Gestalt principle of closure, that refers to our mind's tendency to perceive complete forms even if a picture is incomplete. This mechanism is well illustrated by the H letter that the icon can evoke. While it does not exactly correspond to the pictograph's negative space, 'H' can be deduced from the global pictograph's shape through closure.

These three approaches offer a wide range of possibilities to embed the hotkey symbol into existing icons. They can also be leveraged to guide the design of novel icons in an application, and decide on the best keyboard shortcuts.

Embedding Modifiers

Embedding the sole hotkey symbol in an icon is sufficient when the keyboard shortcuts do not involve modifier keys (e.g, switching tools in Adobe Photoshop, or Final Cut) or when the shortcuts consistently involve the same modifier, typically ctrl or ⌘. When different modifiers are involved in the same application, further indications are necessary.

Embedding a graphical or textual representation of modifier keys can dramatically increase the complexity of the icon and affects its readability. To generate possible visual encodings of modifiers, we elicited ideas from four colleagues, all HCI experts not involved in the design of IconHK. We asked them, individually, to sketch on paper *"visual encodings that can be displayed on a toolbar icon to convey the modifier keys used in its keyboard shortcut."*

One frequently proposed idea consists of mapping each modifier key to a square located at one of the four corners of the button, where a square is filled when the corresponding modifier is used by the shortcut and empty otherwise (shortcuts never involve more than four modifier keys), as illustrated in Figure 1. This design was proposed for its visual consistency, simplicity and because it capitalizes on spatial memory (i.e. the same modifier is always associated with the same corner). One of the solicited experts motivated: *"each modifier key can be stably mapped to a corner of the button, depending on its physical position on the keyboard: alt at the bottom-right, ctrl bottom-left and shift top-left."* The notion of reusing keyboard layout to foster spatial memory was recurrent in other designs not involving corners.

Depending on the look-and-feel of the buttons to augment with iconHK, different visual indicators can be envisioned. Our colleagues mostly reasoned with squares; in Figure 1, we used quarter-circle instead of squares in order to minimize the visual space occupied. This design also affords a richer status encoding, that can further help recognition and recall: when a user presses a specific modifier key, the corresponding corner is highlighted on all toolbar buttons in order to help users discover this mapping (e.g., the bottom-left corner is highlighted in blue upon ctrl press in Figure 1, right).

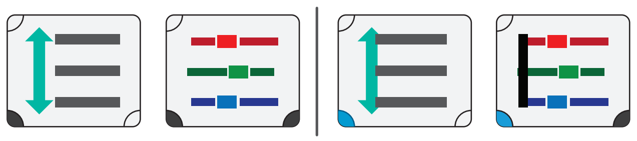

Figure 1. Example of icons augmented with iconHK that embed modifiers using the keyboard-location inspired mapping: alt at the bottom-right, Ctrl at the bottom-left, shift at the top-left. The commands *equal line space* (left) and *RGB colors* (right) respectively have Ctrl + E and Ctrl + alt + E as keyboard shortcuts. Corners are filled in dark to convey modifiers involved in the shortcuts (bottom-left corner only for *equal line space*, two bottom corners for *RGB colors*). The bottom-left corner is highlighted in blue when Ctrl is pressed to provide feedback.

The magnification continuum of IconHK

While we assume that modifiers can be displayed at all times without impacting buttons' readability, a compromise might have to be found for symbols. It is not always fortunate, nor even possible, to legibly display a symbol. Integrating a symbol can also affect aesthetic, or hinder pictograph recognition.

To overcome this issue, we introduce the notion of *magnification continuum*, as the entire spectrum of representations resulting from a progressive morphing from the pictograph representation, to the symbol only (Figure 2). At one end of the continuum, the button conveys the meaning, as found in most applications. At the other end, the button communicates the hotkey, as in Hopper Disassembler [2]. This latter strategy is seldom used and only evocative to knowledgeable users, as users who are not aware of the shortcut mechanism or key combination might be confused because they do not understand the meaning of the icon. IconHK provides static and dynamic alternatives that lie between these two extremes.

Formal definition

The magnification continuum ranges from 0 to 1, where 0 means that the button displays the pictograph only, and 1 means that the button displays the symbol only. All representations in between are a blend of the two (Figure 2).

Toolbar buttons in an application can be set to a default *static* representation from anywhere in the spectrum at a value that the designer judges offer the best tradeoff, e.g., the mid-point of the spectrum. Or, the whole or part of the continuum can be leveraged in a *dynamic* manner, e.g., a button can smoothly animate between a representation closer to the pictograph representation to one closer to the literal symbol representation of the hotkey when hovered over (we discuss different interaction design options in a later section).

To formalize, let us consider a toolbar button enriched with IconHK. We define two stable states noted $M_{\sqsubset}$ and $M_{\sqsupset}$, that correspond to the two bound states along the continuum between which the button representation can vary (e.g. pictograph as the default state, and slightly revealed symbol as the hovered over state), and let $M_{|}$ be the current state of the button representation. We have:

$$0 \leq M_{\sqsubset} \qquad M_{\sqsupset} \leq 1 \qquad M_{\sqsubset} \leq M_{|} \leq M_{\sqsupset}$$

The values for $M_{\sqsubset}$ and $M_{\sqsupset}$ are to be defined by the designer, and may be different across buttons depending on the embedding strategy and interaction design. For example, a designer may choose to set all icons enriched with empty space strategy with a $M_{\sqsubset} = M_{\sqsupset}$ value just high enough for the symbol to be visible (e.g. for the pencil command of Figure 2, $M_{\sqsubset} = 0.2$ corresponds to). For all other icons, she may set $M_{\sqsubset}$ to zero and $M_{\sqsupset}$ to one (e.g., for the expand vertically icon, $M_{\sqsubset} = 0$ corresponds to and $M_{\sqsupset} = 1$ corresponds to), betting on the users' capability to associate the pictograph with the corresponding letter while providing the opportunity to hover over the button for recall. Similarly, $M_{|}$ can vary independently for each button, depending on users' interaction. We discuss how these parameters can be leveraged to support different design scenario in a further section.

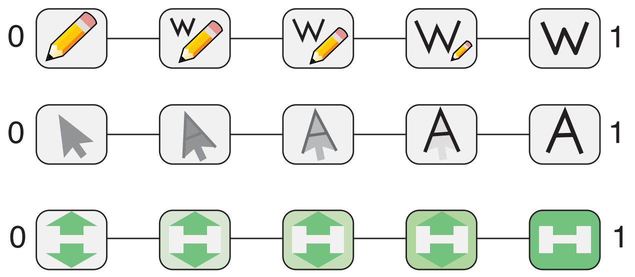

Figure 2. Variations of toolbar buttons from 0 to 1 on the IconHK continuum for the *Pencil*, *Move* and *Expand vertically* commands: *Pencil* scales down while a W scales up, exploiting the empty space; *Move* rotates and fades out the pointer while revealing a A overlaying the edges; *Expand vertically* changes the background color of the button to emphasize the H symbol in the negative space. (Animated Figure)

Magnifying along the continuum: symbol saliency

As we move along the magnification continuum, the symbol is progressively revealed until it becomes fully legible by augmenting its saliency, an effect that can be achieved with different transformations such as rotation, translation, scaling as well as manipulation of the opacity or color of the foreground and background. The approaches depend on the magnification strategy and the designer's preference.

For instance, the empty space strategy leverages pixel areas in two ways. When the symbol is to be embedded in a pixel zone unoccupied by the pictograph, augmenting its saliency can be achieved by increasing the symbol's size while reducing that of the pictograph (Figure 2, top line), resulting in sort of a swipe transition. When the empty space corresponds to an homogeneous area of the pictograph (e.g.), instead of reducing the pictograph's size, we make it grow while also increasing the symbol's size, as if zooming on the area containing the symbol until it occupies the whole area.

Both the positive and negative space strategies leverage the pictograph features to reveal a symbol. An approach consists of progressively fade in the symbol while fading out the pictograph by adapting their opacity and color. For positive space, this amounts to reducing the opacity of the pictograph except from the salient features that delineate the symbol, which, conversely, are emphasized. For negative space, the idea consists in progressively reinforcing the contrast between the positive and negative space, and further extend the positive space to create a closure (see Figure 2, bottom line). Additional affine transformation might also be used to realign the symbol. More advanced transformations might use semantic elements of the pictograph to reveal the symbol (e.g. slider thumbs translate to form the 'E' symbol in Figure 3).

IconHK dynamic behavior

IconHK affords many possibilities in terms of icon behavior. We imagine three cases when transitions from pictograph to symbol (and vice versa) could be relevant.

1. Regular reminders. An application could emphasize symbols of the whole toolbar at launch for a few seconds, before switching back to the more traditional pictographs. This gives users awareness of the special nature of the toolbar icons, and prompt hotkeys as a reminder. This operation could also be triggered every now and then to foster awareness and recall.

2. *Adaptive saliency*. The magnification level of icons could evolve depending on application usage. For instance, the $M_{\sqsubset}$ and $M_{\sqsupset}$ values can be set to vary depending on user's expertise regarding a certain command (Figure 3). Typically, the more the user selects a command using the mouse, the closer $M_{\sqsubset}$ gets to $M_{\sqsupset}$ for that command, thus encouraging hotkey usage. Conversely, the more a command is selected using its hotkey, the closer $M_{\sqsubset}$ gets to 0, since the user's behavior suggests she masters the hotkey and does not require a visual aid. Note that different strategies could be explored, for instance by increasing $M_{\sqsubset}$ regardless of the modality used for selecting the command, thus not only increasing the saliency of the symbol, but also communicating how frequently a command is used, in a similar fashion as [10]. Another strategy could be to update $M_{\sqsubset}$ or $M_{\sqsupset}$ values as a function of the frequency of the command among a group of users.

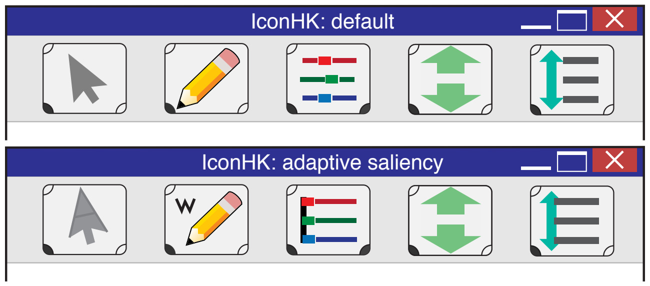

Figure 3. Example of a toolbar using IconHK. Top: by default ($M_{|}$), the magnification of all buttons is equal to $M_{\sqsubset}$: the hotkey symbol is not emphasized. Bottom: the individual $M_{|}$ of each button reflects user's command selection habits: the more frequently a command is selected by clicking, the higher the value of $M_{|}$ is.

3. *Interactive saliency*. The dynamic behavior of icons could also vary depending on user's interactions with the system at a much lower level, as for instance, when hovering over the icon or when pressing a modifier key, as described below.

Inspired by ExposeHK [24], $M_{|}$ values of all toolbar buttons can gradually increase to $M_{\sqsupset}$ when the user presses a modifier key to better emphasize the symbols associated with this modifier (see Figure 4) and remain as such until modifiers are released, triggering a saliency decrease back to $M_{\sqsubset}$. This magnification provides a feedforward mechanism allowing users to identify the hotkey when initiating a keyboard shortcut. This feedforward mechanism is less intrusive than pop-up mechanisms (like ExposeHK) in that the symbol is seamlessly embedded within the icon (preventing occlusion) and gradual transition within the icon itself is less visually distracting than appearing pop-ups.

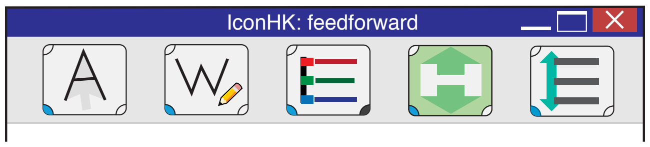

Figure 4. When the user presses a modifier key, the magnification factor of all buttons increases to $M_{\sqsupset}$ to maximize the saliency of the hotkey when the user needs them the most. (Interactive Figure)

Similar to [15, 19], IconHK can also be used to provide feedback, in complement to feedforward, when a command is activated using the mouse. For instance, every time a command is selected in such a way in the menubar, the $M_{|}$ value of its corresponding toolbar button gradually increases to $M_{\sqsupset}$ and is maintained at this value for a given time laps (e.g. 500ms)—guaranteeing a maximal saliency of the keyboard shortcut during a short period—before it is progressively decreased back to $M_{\sqsubset}$. When a command is selected using the mouse in the toolbar, $M_{|}$ immediately increases to $M_{\sqsupset}$ in order to magnify the symbol, and remains as is so long as the toolbar button is pressed, or until the command is activated (Figure 5). When activated, $M_{|}$ remains equal to $M_{\sqsupset}$ for an additional given time before decreasing back to $M_{\sqsubset}$.

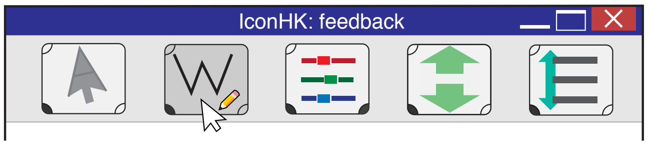

Figure 5. When the user presses on a toolbar button, its saliency instantly increases to $M_{\sqsupset}$. (Interactive Figure)

Overall, the transitions presented in this section are just one instance of possible IconHK dynamic behaviors, but other possibilities could be explored. Still, we recommend that transitions, especially when based on user's interaction, are always animated to facilitate comprehension and foster learning, as well as maintain a high visual appeal.

Mnemonic mechanisms

Widespread icons are interesting to analyse from our perspective of using icons as mnemonic aids. We discussed that the scissor (Cut) could evoke the 'X' symbol. This pictograph was inspired by traditional practice in manuscript editing whereby people would cut paragraphs from a page with scissors and paste them onto another document. It remains, however, unclear if the hotkey $\boxed{\text{X}}$ was chosen because of its similarity with a scissor silhouette, or if the icon was designed to evoke the standard hotkey, or none of the above.

It is worth noting that $\boxed{\text{X}}$, $\boxed{\text{C}}$, $\boxed{\text{V}}$ and $\boxed{\text{Z}}$ hotkeys for the most common commands Cut-Copy-Paste and Undo were chosen to be all clustered together to facilitate sequence of operations. Which one of these hotkeys was first, if any, and whether such choice was guided by a mnemonic strategy (i.e. 'X' to recall the shape of the scissor? 'C' for Copy?) also remains obscure. The important lesson to retain is that there are several factors that come into play when choosing a hotkey, be them other mnemonic mechanisms such as using the first letter of a command to reinforce associations, or constraints that limit options for icon and hotkey design.

Formatting text icons is also interesting in that the formatting effect is directly illustrated within the icon, e.g. $\boxed{\textit{I}}$ for Italics or $\boxed{\textbf{B}}$ for Bold. In that case, 'B' is both the hotkey and the first letter of the command name. We did not find any reference allowing to claim whether 'B' is used as a reminder for the command name or for the hotkey. Yet, in French applications, the corresponding pair ($\boxed{\textbf{G}}$ icon; $\boxed{\text{ctrl}}+\boxed{\text{B}}$ shortcut), the icon refers to the command name ('G' for Gras) rather than the hotkey $\boxed{\text{B}}$.

REDESIGN OF THE PHOTOSHOP PALETTE

In this section, we revisit the Photoshop CC 2015 palette (Figure 6-top) to communicate available keyboard shortcuts with IconHK. Our goal was to respect the original icon style and thus make as minimal changes as possible to not confuse users already familiar with the traditional iconset. Our proposed design is illustrated in Figure 6-middle & bottom.

Several icons were very suitable to embed a hotkey symbol, e.g., squeeze the 'G' symbol within a bucket (Paint Bucket); rotate the eyedropper to insert a vertical 'I' (Eyedropper); emphasize the edges of the arrow to reveal a 'A' (Move). Interesting design cases are the Pencil and the Pen tools that both use the empty space to embed the 'B' and 'P' symbols respectively, while leveraging the semantic of the command (i.e. the tools draw the letters).

We found that some rather obvious embedding cases could introduce confusion, e.g. the silhouette of the Dodge tool evokes a 'Q' letter whereas the shortcut's hotkey is 'O'. We flipped the pictograph so that the handle is less suggestive of the 'Q' descender.

Figure 6-bottom shows symbols with a high level of magnification. In practice, the default magnification could be less or more salient (e.g. Figure 6-middle), and icons made responsive to users' interaction as previously discussed. Our intent with this example is to demonstrate that IconHK can successfully be incorporated to already existing toolbars, though some enriched icons may not be exemplary due to the strong constraints of keeping a design style close to the original. Obviously, designing iconsets from scratch, with the IconHK principles in mind, offers more flexibility, which can result in more cohesive and more aesthetically appealing iconsets.

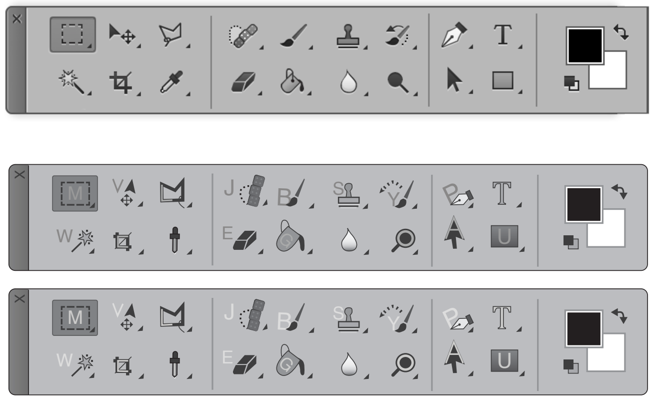

Figure 6. Possible adaption of the Adobe Photoshop CC 2015 interface using IconHK. Top: default toolbar. Middle & Bottom: the adapted toolbar integrating IconHK principles at two different levels of magnification. All buttons embed their associated hotkey, except the Blur button (water drop) which lacks a shortkey.

STUDY 1: ICONHK AS A MNEMONIC AID

The primary goal of IconHK is to provide a visual aid for prompting keyboard shortcuts, so we conducted a user study to assess its potential as a mnemonic aid. More specifically, we investigated whether a relatively short exposure to an intermediate level of magnification (i.e. $M_| = 0.5$) fosters subsequent *retrieval* from the pictograph only (i.e. $M_| = 0$).

Method

In this experiment, we compared the three IconHK strategies (*Empty*, *Positive* and *Negative* space) and a *Control* condition (where the hotkey is not embedded in the icon in any way). We used 4 commands per strategy for a total of 16 commands (see Figure 7). Every command had an icon and an associated keyboard shortcut. Each magnified icon embeds a single IconHK strategy to avoid possible confound effects. All keyboard shortcuts used the Ctrl modifier only, and a character key that was not the first or last letter of the command name [15, 4].

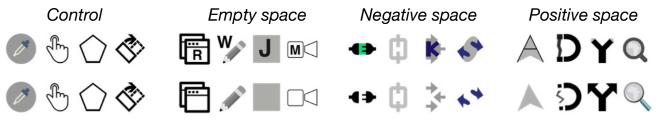

Figure 7. Icons used study 1. Top: icons used for the trainings (magnified for IconHK); Bottom: icons used for the tests (not magnified).

Twelve participants (1 female) aged 22 to 36 performed 4 repetitions of training and test phases (inspired by [6, 33]):

Training: During this phase, the interface showed the 16 commands in the toolbar. Toolbar buttons relying on IconHK were magnified with a level of $M_| = 0.5$ so that the icon conveyed both the meaning and keyboard shortcut of the command. Participants were asked to select commands as fast and accurately as possible using their keyboard shortcuts. If they did not know the shortcut, they could mouse hover over the corresponding button to reveal a tooltip displaying it.

Test: During this phase, the system displayed unmagnified icons as visual stimulus, i.e. the letter was not printed for the empty space strategy and the positive/negative space was not highlighted anymore. Participants were asked to execute the shortcut corresponding to the requested command as fast as possible. If they did not know the shortcut, they had to try to guess anyway. For Positive and Negative, this is a *retrieval* test as some visual cues remain in the icon. For Control and Empty, we did not expect participants to retrieve the shortcut from the icon, thus it can be seen as a *recall* test.

Participants then performed online post-tests, replicating the test phase 24 and 72 hours after the experiment without being re-exposed to shortcuts (that is, participants did not practice the shortcuts after the 4th block of the initial test).

In each phase, the 16 commands were presented in a randomized order, with each command appearing exactly once. In summary, each of the 12 participants performed 4 design strategies (Positive, Negative, Empty, Control) $\times 4$ commands $= 16$ trials per test phase (4 laboratory tests $+2$ online post-tests) for a total of $12 \times 4 \times 4 \times (6 + 2) = 1152$ trials.

Results

We define the *retrieval rate* as the proportion of correct answers in the test phases. The average retrieval rate across the 4 initial tests was 54% CI [43,65] for Control, 50% CI [39,62] for Empty, 88% CI [80,94] for Positive and 87% CI [80,93] for Negative. It was 73% CI [59,87] for Control, 84% CI [76,95] for Empty, and 100% for both Positive and Negative for the 2 post-tests. Figure 8 shows that learning was consistent across

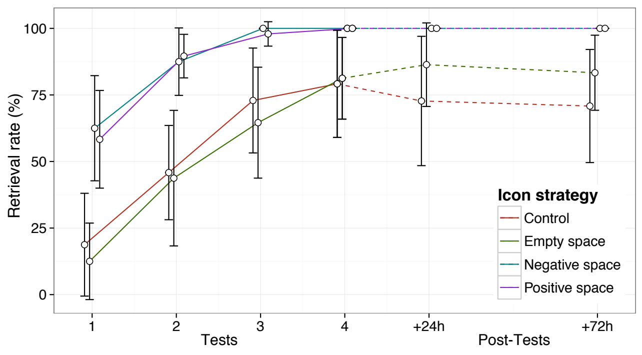

Figure 8. Percentage of recognition rates for each Icon family, for every block and retention tests (dashed)

blocks for all conditions, but quicker for Positive and Negative reaching almost 100% at the 3rd test and remaining stable until the 2nd post test, 72h after training. It also shows that Control was the only condition where the retrieval rate decreased between the last test block and the 72h retention test, changing from 79% back to 71%.

These results suggest that brief exposures were sufficient for participants to retrieve hotkeys with the Positive and Negative strategies, even 72 hours after the last training. These encouraging results motivated us to learn more about how the IconHK principle might be included in designers' practice.

STUDY 2: INSIGHTS FROM DESIGNERS

We conducted a second study with professional designers to collect feedback on their impressions about, and possible difficulties with the IconHK concept, as well as gain insight in the design process for creating iconsets building on IconHK. Three professional designers (2 females) aged 28 to 29 were recruited. We refer to them as D1, D2 and D3 in the following. All have experience in both graphic design and logo/icon design (3, 5 and 6 years respectively).

Method

Each session started with the experimenter explaining the IconHK concept, i.e. the 3 embedding strategies and the magnification continuum. Participants were then instructed to design three iconsets, one per embedding strategy given the same set of 6 commands: *Save*, *Copy*, *Print*, *Edit path*, *Plot* and *Refresh*. We chose this set of commands to have a balance between most common commands (*Save, Copy, Print*) and more specialized ones (*Edit path, Plot, Refresh*), for which a particular icon metaphor may be less standard.

For a given IconHK strategy, the task was to create an iconset implementing the embedding strategy for each of the six commands. Designers were instructed to consider good icon/shortcut design practices: for each command, (1) choose a pictograph that best represent the meaning of the command and (2) a symbol (i.e. hotkey letter) that can act as a mnemonic for the command, as well as (3) maintain consistency across the entire iconset (i.e. graphic style). We left them free, however, to weight each objective as they saw fit to make the best design compromises for the whole iconset.

Designers were free to sketch icons with their preferred tools. In the case of static sketches (pen+paper, or vector graphics), we asked them to provide at least 3 levels of magnification for each icon to illustrate the steps of the animation. We also invited them to draw inspiration from a collection of icons (6 per command) that we curated to illustrate various pictograph metaphors as well as graphical styles.

Designers were encouraged to describe and explain their design process through a think-aloud protocol. The experimenter was present during the whole session and took notes. When deemed necessary the experimenter asked the designers for clarifications. The session ended with a semi-structured interview inquiring about design decisions, impressions and the difficulties they encountered. Figure 9-left shows a selection of resulting designs.

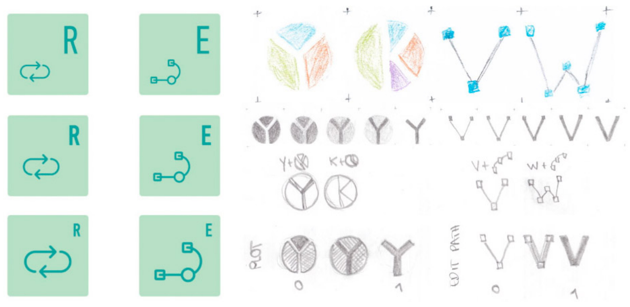

Figure 9. Left: Icons implementing the Empty space strategy from the first part of the study. Right: Positive and Negative sketches with different magnification from the follow-up study: *Plot*: A pie chart pictograph embedding either a 'Y' or a 'K'. *Edit Path*: A connected line embedding either a 'V'or a 'W'.

Results

All three designers felt more comfortable using a vector graphics software or paper+pencil over animation software or programming. Designers interrupted the session (after 3h, 2h, and 4h respectively for D1, D2, and D3) before completing all three iconsets, albeit no time constraint was imposed. We discuss here the key insights of our study:

Design process. It was interesting to observe how designers spontaneously made radical decisions regarding good icon and shortcut design practices. All of them primarily focused on consistency between icons, as they generally feel that a consistent iconset is more *"user friendly than an iconset that better communicate the information to the user,"* a mindset that contradicts IconHK's essence. How they valued consistency had an impact on the choice of the strategies: they believed that mixing the embedding strategies in the same iconset could be confusing to the user because the icons would not have the *"same identity."* After the experiment, informal discussions with HCI researchers working closely with designers confirmed that consistency is strongly anchored in designers' practice. The choice of pictographs and hotkeys as representative of commands was lower-ranking, yet designers picked meaningful ones. The designers favored simple outline or glyph designs for the pictograph, drawing inspiration from the icons we provided and also from other examples from the web focusing mostly in minimalistic designs. Finally, they consistently chose the first letter of the command as the hotkey, unless name collision occurred, in which case they favored the most frequent command and searched on the web which shortcut is generally used for the other command.

IconHK concept and feasibility. Designers thought that the concept of IconHK is interesting and potentially useful for the users but expressed reservations about the feasibility of the technique. D3 found that *"this technique can benefit the users if done correctly."* D2 reported *"I can see how this can be useful for the end user and it poses an interesting design problem as well,"* further commenting that *"creating an iconset which is coherent, minimalistic and successfully communicates the meaning and the hotkey is a time and resource demanding task. I am not sure the results would be satisfactory"*, which explains their difficulty completing the task.

Empty space. Empty space appeared to be the easiest strategy to implement. Indeed, spatially organizing the pictograph/hotkey requires less effort than creating an icon with positive or negative space. Moreover, it is the easiest strategy to maintain consistency across icons. For instance, each set of icons used the same position for the hotkey. Designers also preferred this strategy because it has a *"minimalist"* style which is a current trend in icon design.

Positive space. All designers experimented with Positive space. D1 wanted *"to see if this strategy comes natural to [her]."* However, designers quickly came to the same conclusion that it would require a considerable amount of time to produce satisfying results. The main reason is that consistency was their primary concern. They explained that it would require to design their own font, i.e. designing letters that have the same look&feel and at the same time fit the pictographs of the iconset. In their opinion this would require several workdays, but they found the exercise interesting.

Negative space. This strategy was considered as the most challenging one because the pictograph and the letter might have completely different shapes. For example D1 explained that for the *Save* command, Floppy disk and 'S' are respectively the most frequent pictograph and hotkey. However, the 'S' letter is curvy while the floppy disk is mainly made of straight lines. *"Creating a variation of either the floppy disk or the letter is a complex procedure and the result might be unsatisfactory"* (D1). This difficulty is amplified if all icons must rely on the same strategy to maintain consistency.

Follow up: Focus on Positive and Negative space
Because the designers experienced difficulty to sketch Positive space and Negative space solutions in the given time-frame, we conducted a follow-up study focusing only on these two strategies by simplifying the experimental design. Two designers, D3 and a novel designer (D4) having 6 years of experience in logo/icon design participated to this study. They were compensated 15$/h (total 2x4h=8h) for their design work. Participants were asked to create 6 icons (instead of 18): 2 positive space, 2 negative space and 2 empty space. We emphasized that consistency between icons was not mandatory, but asked for five levels of magnification for each command. All icons are attached in the supplementary materials. Figure 9-right shows some resulting sketches.

Findings: Since D3 was already familiar with the IconHK concept she did not have any new comments. D4 confirmed that the Negative space strategy was the most difficult one, especially for complex pictographs. However she also mentioned that *"choosing a letter that takes advantage of the shape of the icon can help users to memorize the hotkey, especially for visual persons."* D4 used a responsive approach for the Empty space strategy. Responsive icons dynamically change their shape and level of details according to their size and are well suited approach to be combined with Empty space. Concerning Positive space, D4 first superimposed the letter shape with the pictograph to see how well she can use the pictograph edges, otherwise, she thought about possible morphing animations but this approach is more complex.

Discussion
We learn that (1) the designers appreciated the IconHK concept, especially Empty space for its simplicity and its compatibility with minimalistic styles; (2) Positive and Negative space are more difficult to apply but raise *"interesting design challenges"*; (3) consistency is a primary quality criterion. While there are multiple debates in HCI/design about consistency vs. efficiency [16], which are not always opponents, this study confirms the current practice of ignoring keyboard shortcuts in icon design. However, our first study showed that going beyond existing icon design practices may benefit end-users. We propose that aesthetic should not always predominate over performance and advocate for design solutions favoring the transition from novice to expert behavior. This study also suggests the need for a tool that assists the designers in creating icons with the IconHK concept.

ICONHK MAKER
We implemented IconHK Maker, a tool to *assist* designers to create icons with embedded keyboard shortcuts. This proof-of-concept should neither be considered as a definite solution to the problem of IconHK design or as a means to substitute designers. Rather, we investigate how image processing algorithms can provide suggestions during the design process: the creative work still requires the designer's judgement and expertise. We present two algorithms to embed a symbol within an icon, one of which identifies the largest empty space to display the symbol, whereas the other analyses the pictograph's features to best fit the symbol in either the positive or negative space. Given an icon, a symbol and a collection of fonts, the algorithms provides a transformation (translation, rotation and scaling) indicating how to embed the symbol in the icon with one of the IconHK strategies.

Finding Empty space. Finding an empty space to display a symbol in a button can be formulated as the *maximum empty rectangle problem* [29]. Given a rectangle R and a set S of n points, the problem consists of finding a maximum area rectangle that is fully contained in R and does not contain any points of S. We implemented the algorithm described in [29], but further adapted it to ensure that the resulting rectangle is not too flat to maximize the space for the letter.

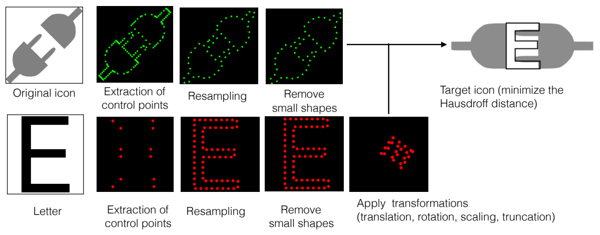
Original icon
Extraction of control points
Resampling
Remove small shapes
Target icon (minimize the Hausdroff distance)
Letter
Extraction of control points
Resampling
Remove small shapes
Apply transformations (translation, rotation, scaling, truncation)

Figure 10. The algorithm to identify an hotkey symbol in the positive and negative space.

Finding Positive and Negative space. We aim to minimize the distance between the symbol and salient features of the pictograph (e.g. its edges that can be made in common with the symbol). The algorithm searches the maximal number of edges that can be shared between these two visual elements. This optimization process comprises four steps. First, we **extract the different paths** of the pictograph and discard small shapes then we **re-sample** the paths to ensure that the distance between two consecutive points is the same. Next, we **apply the Ransac algorithm** (Random Sample Consensus) [11]. At each iteration, this algorithm generates a random transformation (see below), estimates the distance between the transformed symbol and the pictograph, and returns the transformation with the smallest distance (Hausdorff [17]).

We consider two rigid transformations: translation and rotation, as well as two non-rigid transformations: scaling and truncation, i.e. removing some paths in the letter. The idea behind this transformation is that a letter's shape is rarely fully defined within an icon. For instance, in the icons (A) and (E), the embedded letters are not complete. Our truncation transformation removes up to two edges.

Finally, we **apply the ICP algorithm** (Iterative Closest Points) [7], which aims at minimizing the distance between two sets of points. ICP is very efficient to find local optimization but does not perform well for global optimization because it is strongly sensitive to the initialization. This is why we use both Ransac (global approximation) and ICP (local optimization) to match the two sets of points.

Interface and Interaction. Our current implementation (C++/Qt) integrates the described algorithms for SVG graphics. It provides three main functionalities: (1) users can edit the SVG icons by manipulating the control points. They can also/draw sketch over the icons. (2) Once users select a letter, the system suggests the best location to embed the letter for both of our algorithms. Suggestions are updated in real time while users are editing the icon. Finally, (3) the system can also suggest a letter given an icon. The algorithm iterates on each letter and keeps the ones with the best scores.

Figure 9 illustrates the intermediate steps and final result for embedding a 'E' symbol within a plug pictograph. Our informal tests with various icons and symbols suggest that computer graphics approaches could be a valuable support for recommendation. Yet, further refinements of the algorithm are yet to be applied to reach the necessary level of robustness of a usable tool. A thorough evaluation of the algorithms performance is beyond the scope of this paper.

CONCLUSION

While graphical user interfaces provide multiple methods for selecting commands, they generally maintain many users in a local optimum of performance where they continue to point and click on toolbar buttons instead of transitioning to keyboard shortcuts. One reason might be that interface designers do not address the problem of command selection as a *whole*, i.e. the choice of the command name, icon and keyboard shortcut are not considered altogether.

With IconHK, we aim at reinforcing the relation between these components: the icon should communicate both the meaning of the command and its keyboard shortcut. However, designing icons is a complex exercise involving multiple (sometimes contradictory) objectives. Designers have to balance multiple objectives such as efficiency, consistency or aesthetics. But so far, rarely communicating the keyboard shortcut has been considered as one major objective.

This paper advocates for a more integrated approach that encourages designers to consider more often the possibility of encoding keyboard shortcuts in the icons. It presents the concept, design principles of IconHK and a proof-of-concept to assist designers. Two user studies provide initial insights in the design process of IconHK and associated challenges, as well as its potential for end-users. We hope that this novel perspective on icon design, and our initial exploration of the IconHK concept will catalize further efforts aiming at a more holistic approach to GUI design.

As IconHK addresses the difficult challenge of icon design, several questions remain opened for future work. In particular, future research should investigate whether IconHK is appropriate for real world applications, assess the impact of magnification level, icon size and modifier encoding on performance, quantify the added value of using animations or pulling methods [24] to help novice users to understand the IconHK principle, and further iterate on IconHK Maker design and evaluation.

ACKNOWLEDGEMENTS

We thank Tamy Boubeker and Stéphane Calderon and our anonymous reviewers for their valuable input on this work. We are also grateful to the participants and designers who took part in our studies.

REFERENCES

1. Cheat sheet: http://www.mediaatelier.com/cheatsheet/.
2. Hopper disassembler: http://hopperapp.com.
3. Hotkey-eve: http://www.hotkey-eve.com.
4. Appert, C., and Zhai, S. Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 2289–2298.
5. Baecker, R., Small, I., and Mander, R. Bringing icons to life. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, ACM (New York, NY, USA, 1991), 1–6.

6. Bau, O., and Mackay, W. E. Octopocus: A dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, ACM (New York, NY, USA, 2008), 37–46.

7. Besl, P. J., and McKay, N. D. Method for registration of 3-d shapes. In *Robotics-DL tentative*, International Society for Optics and Photonics (1992), 586–606.

8. Bodner, R. C., and MacKenzie, I. S. Using animated icons to present complex tasks. In *Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative research*, IBM Press (1997), 4.

9. Cockburn, A., Gutwin, C., Scarr, J., and Malacria, S. Supporting novice to expert transitions in user interfaces. *ACM Comput. Surv. 47*, 2 (Nov. 2014), 31:1–31:36.

10. Debevc, M., Meyer, B., Donlagic, D., and Svecko, R. Design and evaluation of an adaptive icon toolbar. *User Modeling and User-Adapted Interaction 6*, 1 (1996), 1–21.

11. Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

12. Gibson, J. J. The theory of affordances. *Perceiving, Acting, and Knowing: Toward an Ecological Psychology* (1977), 127–143.

13. Gibson, J. J. *The ecological approach to visual perception: classic edition*. Psychology Press, 1979.

14. Gittins, D. Icon-based human-computer interaction. *International Journal of Man-Machine Studies 24*, 6 (1986), 519–543.

15. Grossman, T., Dragicevic, P., and Balakrishnan, R. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2007), 1591–1600.

16. Grudin, J. The case against user interface consistency. *Communications of the ACM 32*, 10 (1989), 1164–1173.

17. Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 15*, 9 (1993), 850–863.

18. Ismair, S., Wagner, J., Selker, T., and Butz, A. Mime: Teaching mid-air pose-command mappings. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, ACM (New York, NY, USA, 2015), 199–206.

19. Krisler, B., and Alterman, R. Training towards mastery: Overcoming the active user paradox. In *Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges*, NordiCHI '08, ACM (New York, NY, USA, 2008), 239–248.

20. Lane, D., Napier, A., Peres, C., and Sandor, A. The Hidden Costs of Graphical User Interfaces: The Failure to Make the Transition from Menus and Icon Tool Bars to Keyboard Shortcuts. *International Journal of Human-Computer Interaction 18* (2005), 133–144.

21. Lindberg, T., and Näsänen, R. The effect of icon spacing and size on the speed of icon processing in the human visual system. *Displays 24*, 3 (2003), 111–120.

22. Lodding, K. N. Iconic interfacing. *IEEE COMP. GRAPHICS & APPLIC. 3*, 2 (1983), 11–20.

23. Ma, X., Matta, N., Cahier, J.-P., Qin, C., and Cheng, Y. From action icon to knowledge icon: Objective-oriented icon taxonomy in computer science. *Displays 39* (2015), 68–79.

24. Malacria, S., Bailly, G., Harrison, J., Cockburn, A., and Gutwin, C. Promoting hotkey use through rehearsal with exposehk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 573–582.

25. Malacria, S., Scarr, J., Cockburn, A., Gutwin, C., and Grossman, T. Skillometers: Reflective widgets that motivate and help users to improve performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, ACM (New York, NY, USA, 2013), 321–330.

26. McDougall, S., Tyrer, V., and Folkard, S. Searching for signs, symbols, and icons: effects of time of day, visual complexity, and grouping. *Journal of Experimental Psychology: Applied 12*, 2 (2006), 118.

27. McDougall, S. J., de Bruijn, O., and Curry, M. B. Exploring the effects of icon characteristics on user performance: the role of icon concreteness, complexity, and distinctiveness. *Journal of Experimental Psychology: Applied 6*, 4 (2000), 291.

28. McDougall, S. J., and Reppa, I. Why do i like it? the relationships between icon characteristics, user performance and aesthetic appeal. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 52, SAGE Publications (2008), 1257–1261.

29. Naamad, A., Lee, D., and Hsu, W.-L. On the maximum empty rectangle problem. *Discrete Applied Mathematics 8*, 3 (1984), 267–277.

30. Nacenta, M., Hinrichs, U., and Carpendale, S. Fatfonts: Combining the symbolic and visual aspects of numbers. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, ACM (New York, NY, USA, 2012), 407–414.

31. Nakamura, C., and Zeng-Treitler, Q. A taxonomy of representation strategies in iconic communication. *International journal of human-computer studies 70*, 8 (2012), 535–551.

32. Norman, D. A. *The design of everyday things: Revised and expanded edition*. Basic books, 1988.

33. Roy, Q., Malacria, S., Guiard, Y., Lecolinet, E., and Eagan, J. Augmented letters: Mnemonic gesture-based shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 2325–2328.

34. Rubin, E. Synsoplevede figurer.

35. Scarr, J., Cockburn, A., Gutwin, C., and Quinn, P. Dips and ceilings: Understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 2741–2750.

36. Silver, L. *Graphic Design that Works: Secrets for Successful Logo, Magazine, Brochure, Promotion and Identity Design*. Rockport, 2004.

37. Tuck, M. Gestalt principles applied in design, August 2010.