MenuInspector : Outil pour l'analyse des menus et cas d'étude

Gilles Bailly
Max Planck Institute for Informatics and
Saarland University
gbailly@mpi-inf.mpg.de

Sylvain Malacria
University of Canterbury
Christchurch, New Zealand
sylvain@malacria.fr

RÉSUMÉ
Les menus sont utilisés dans la plupart de nos applications pour présenter, organiser et sélectionner des commandes. Un grand nombre de techniques d'interaction a été proposé et de nombreuses évaluations expérimentales ont été conduites. Pourtant, peu d'études ont visé à analyser les menus existants. Dans cet article, nous proposons et mettons à disposition *MenuInspector*, un outil pour analyser les menus des applications existantes sous MacOSX. MenuInspector permet de récupérer de nombreuses informations comme la hiérarchie de commandes, les libellés des commandes, les raccourcis clavier, les tooltips, etc. Nous illustrons les avantages de MenuInspector à travers quatre cas d'étude : étude du recouvrement, des collisions, de la consistance entre applications ainsi que la mise en évidence d'éléments cachés.

Mots Clés
Menus; hotkeys; inspection; sélection de commandes

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

INTRODUCTION

La barre de menus est présente dans nos applications pour présenter et organiser les commandes et pour permettre à l'utilisateur de les sélectionner. L'utilisabilité de la barre de menus détermine souvent l'utilisabilité de l'application [15]. Pourtant développer un menu hiérarchique est un problème difficile pour les concepteurs qui nécessite de prendre en compte de nombreux aspects comme l'apparence, la sémantique ou l'interaction [16].

Pour guider la conception de nouvelles techniques d'interaction pour les menus ou simplement organiser les commandes dans des menus linéaires, il est primordial de comprendre les systèmes de menus existants. Nos applications fournissent une base de connaissances considérable. Chaque ordinateur personnel contient plusieurs dizaines/centaines d'exemples de barres de menus pouvant servir d'inspiration pour les chercheurs ou les industriels. Cependant, il n'est pas envisageable de capturer et d'analyser manuellement l'ensemble de ces menus.

IHM'13, November 13–15, 2013, Bordeaux, France.

Figure 1. Gauche: Les menus File, Edit, History et View de Safari se recouvrent. Droite : les données récupérées par MenuInspector permettant de quantifier le recouvrement possible entre les menus. Représentation à l'aide d'une carte de chaleur (rouge = recouvrement plus important).

Dans cet article, nous proposons MenuInspector, un outil pour étudier l'organisation des barres de menus des applications sur la plateforme MacOSX. MenuInspector est une application qui s'exécute en tâche de fond et enregistre au format XML toutes les informations concernant le système de menus dès que l'application prend le focus parmi lesquelles la hiérarchie de commandes, le libellé des commandes, les raccourcis clavier, etc. MenuInspector est accompagné d'un outil pour visualiser et explorer les menus capturés. MenuInspector est disponible à cette adresse `http://menuinspector.malacria.fr/`.

Nous démontrons l'utilité de MenuInspector à travers 4 scénarios. 1) Quantifier la quantité de recouvrement entre les sous-menus si ces derniers étaient ouverts en même temps. Les résultats ont permis de proposer une nouvelle technique de menus et d'estimer ses limitations. 2) Quantifier la quantité de collisions entre les items (nombre de commandes commençant par la même lettre). Les résultats nous ont aidé à développer une nouvelle technique d'interaction. 3) Définir une mesure de consistance entre les systèmes de menus. Les résultats ont mené à la mise au point d'une méthode d'optimisation des menus linéaires. Enfin, 4) MenuInspector a mis en évidence des éléments cachés dans les menus ainsi que des défauts d'implémentation dans des applications grand public.

Les principales contributions sont :

- L'implémentation et la mise à disposition de MenuInspector pour l'analyse des sytèmes de menus.
- 4 scénarios illustrant les avantages de MenuInspector

ÉTAT DE L'ART

La sélection de commandes est un domaine de recherche actif depuis plus de 40 ans [4]. Elle comprend la conception de nouvelles techniques d'interaction [16] comme les Marking Menus [11], l'élaboration de modèles de performance [8], des méthodes d'optimisation [15] ou des études empiriques [6]. L'étendue et le dynamisme de ce domaine de recherche justifient l'élaboration d'outils adaptés.

Des outils ont été proposés pour aider à l'analyse [10], la conception [3], l'implémentation [2] ou l'évaluation de techniques d'interaction [12]. Cependant, peu d'entre eux se sont concentrés sur les menus. Par exemple, Matejka et al. ont récemment proposé Patina [14], un outil pour collecter et visualiser la manière dont les utilisateurs sélectionnent les commandes. Alors que Patina se concentre sur l'interaction à la souris sur la barre d'outils, MenuInspector vise à l'analyse des menus et des raccourcis clavier. De plus, Patina fonctionne uniquement pour les applications implémentant l'API d'accessibilité sous Windows, un sous-ensemble réduit par rapport à l'ensemble des applications sous Windows. Enfin, il se concentre davantage sur les actions de l'utilisateur plutôt que sur le contenu de l'interface. Au contraire, certains outils se concentrent uniquement sur le contenu de l'interface comme Prefab [9] ou Sikuli [7]. Bien que ces outils aient l'avantage d'être multi-plateforme, ils ont l'inconvénient d'être difficiles à utiliser et à déployer dans la mesure où ils reposent sur de l'analyse vidéo de l'affichage de l'ordinateur.

MENU INSPECTOR

Nous proposons MenuInspector, un outil pour décrire et comprendre la composition des menus dans les applications de bureau. Il fonctionne avec tous les menus apparaissant dans la barre de menu MacOSX et permet d'accéder à de nombreuses informations : le nom des menus, des sous menus, des éléments (items) ainsi que leurs raccourcis clavier respectifs. MenuInspector permet également d'accéder à des éléments cachés du menu qui n'apparaissent que sur des actions spécifiques de l'utilisateur.

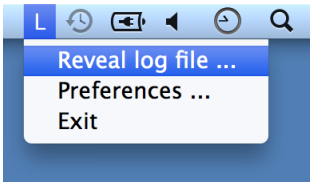

Figure 2. MenuInspector s'exécute en tache de fond et apparait dans la barre d'état. La base de données des menus inspectés au format XML est accessible via un menu.

Historique

MenuInspector génère un fichier XML (figure 3) contenant une description des menus des applications en train de s'exécuter. Il vérifie chaque seconde l'application qui a le focus. La barre de menus de cette application est ajoutée si celle-ci n'est pas dans le fichier ou si elle a été modifiée.

```
.....
<menu title="View">
        <menuitem title="as Icons" modifiers="0" hotkeyChar="1"></menuitem>
        <menuitem title="as List" modifiers="0" hotkeyChar="2"></menuitem>
        <menuitem title="as Columns" modifiers="0" hotkeyChar="3"></menuitem>
        <menuitem title="as Cover Flow" modifiers="0" hotkeyChar="4"></menuitem>
        <separator></separator>
        <menuitem title="Clean Up" modifiers="10"></menuitem>
        <menuitem title="Clean Up Selection" modifiers="8"></menuitem>
.....
```

Figure 3. Exemple d'une partie du fichier généré par MenuInspector

Utilisation

Une fois démarré, MenuInspector apparait dans la barre d'état. Pour ajouter la description d'un nouveau menu d'application, il suffit de donner le focus à l'application correspondante, par exemple en la démarrant ou en cliquant dessus dans le dock MacOSX. L'utilisateur peut ensuite utiliser le menu contextuel de MenuInspector (figure 2) pour accéder aux fichiers d'historique.

Réalisation

MenuInspector est une application pour MacOSX qui s'exécute comme un élément de la barre d'état (figure 2). Il a été réalisé en Objective-C sous le Framework Cocoa. Il repose sur l'utilisation de l'API d'accessibilité Apple [1] qui donne accès à un nombre conséquent d'informations provenant des propriétés des éléments contenus par l'interface graphique. En particulier, l'API d'accessibilité permet d'accéder à la barre des menus MacOSX et de parcourir l'intégralité de son arborescence.

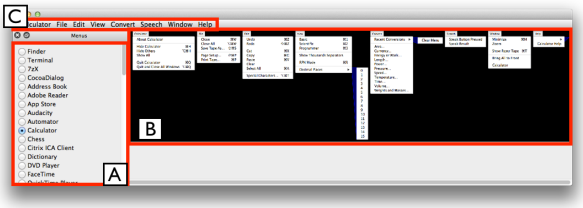

Figure 4. Outil de visualisation accompagnant MenuInspector. A) Liste des applications disponibles. B) Vue dépliée de la barre de menus de l'application sélectionnée. C) Menu avec lequel l'utilisateur peut intéragir.

Visualisation

MenuInspector est accompagné d'un outil pour visualiser et explorer les différents menus enregistrés dans le fichier XML illustré (figure 4). L'utilisateur peut choisir un système de menus et celui-ci est affiché avec une vue dépliée, sans recouvrement comme illustré. L'utilisateur peut également interagir avec le menu comme il le ferait avec une application.

Scripts

Des scripts Java ont été développés pour être utilisés avec MenuInspector et fournir des informations supplémentaires telles que le nombre de commandes par menu, la proportion d'éléments avec raccourcis clavier, etc. Ces scripts parcourent la base de données XML créée par MenuInspector et affichent ces informations dans une table. Ces informations peuvent ensuite

être exportées dans un fichier CSV pour éventuellement effectuer des opérations supplémentaires.

MenuInspector, l'outil de visualisation ainsi que les différents scripts utilisés sont disponibles à :

`http://menuinspector.malacria.fr/`.

CAS D'ÉTUDE

Dans cette section, nous montrons l'utilité de MenuInspector à travers plusieurs exemples.

Étude du recouvrement entre les menus

MenuInspector a été utilisé pour étudier la quantité de recouvrement entre les menus afin de concevoir une nouvelle technique d'interaction, ExposeHK [13], pour favoriser l'usage des raccourcis clavier.

ExposeHK [13] (figure 5) affiche tous les menus du premier niveau d'une application dès que l'utilisateur presse la touche Ctrl (ou Cmd). L'utilisateur peut alors sélectionner une commande en reconnaissant le raccourci clavier correspondant au lieu de l'avoir mémorisé. Cependant, afficher simultanément tous les menus du premier niveau génère des recouvrements (figure 1) empêchant l'utilisateur de lire certains raccourcis clavier.

MenuInspector nous a permis 1) de quantifier le nombre d'éléments dont l'intitulé du raccourci clavier associé était recouvert par d'autres menus et 2) de positionner les menus afin d'éviter ce problème de recouvrement. Sur 30 applications fréquentes sous MacOSX, le recouvrement moyen d'un item avec un raccourci clavier par un autre item avec un autre raccourci clavier est de 23,93 % (std = 10,35) et de 44,9 % (std = 18,78) avec un item sans raccourci clavier. Une application a en moyenne 8,9 menus et la largeur moyenne d'un menu est 135 pixels.

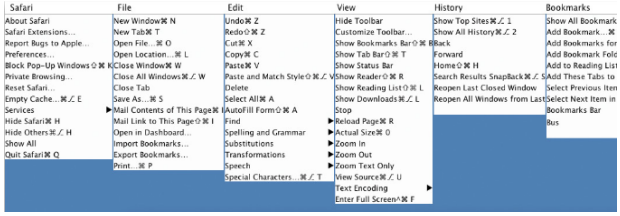

Figure 5. ExposeHK affiche les menus du premier niveau d'une application dès que l'utilisateur presse la touche Ctrl (ou Cmd) pour favoriser l'usage des raccourcis clavier.

Étude des collisions entre les différentes commandes

MenuInspector a été utilisé pour étudier les collisions de noms de commandes afin de justifier la conception d'une technique d'interaction appelée Augmented Letters [17].

Les Augmented Letters sont des raccourcis gestuels combinant la première lettre d'une commande avec un trait directionnel dans une des 8 directions. Cependant, la technique ne fonctionne plus si plus de 8 commandes commencent par la même lettre.

MenuInspector a été utilisé avec 32 applications pour compter le nombre total de commandes (N) et répertorier la première lettre de celles-ci. Grace à ces données, les auteurs ont montré que la probabilité p qu'une lettre corresponde à plus de 8 commandes augmente linéairement avec N, mettant notamment en avant que $p < .05$ pour $N < 64$, l'équation exacte étant :

$$p = 0,0019 * N - 0,07, r^2 = 0,92$$

Étude de la consistance entre systèmes de menus

MenuInspector a aussi servi à la définition d'une mesure de consistance entre des systèmes de menus afin 1) d'étendre un modèle de performance des menus et 2) d'optimiser les menus [5].

Avec l'aide de MenuInspector, nous avons construit une base de données de *co-occurences* entre les items. Le script utilisé (1) liste la totalité des paires de commandes possibles : dans notre cas, 111 859 paires de commandes qui correspondent aux 3 290 commandes collectées à partir de 68 applications. Pour chacune de ces paires, le script (2) calcule un score représentant la propension de deux commandes à être "proches" l'une de l'autre. Par exemple, le score de "Copier - Coller" est 0,99 car toujours adjacentes. Celui de "Enregistrer - Imprimer" est 0,36 car dans le même sous-menu. Enfin, celui de "Nouveau document - Annuler" est 0,0 car ces deux commandes n'ont pas d'ancêtre en commun.

La méthode d'optimisation utilise ce score pour évaluer la consistance entre les différents menus générés avec les menus des applications existantes. Le résultat pour un logiciel graphique est illustré sur la figure 6.

Étude des élements "cachés"

Enfin, l'utilisation de MenuInspector a mis en évidence des éléments cachés dans un menu. En effet, les barres de menus MacOSX contiennent des éléments qui ne sont pas toujours présents dans le menu. C'est notamment le cas du menu *Go* (ou *Aller à*) de l'application Finder qui n'affiche l'élément *Library* que lorsque l'utilisateur presse la touche Alt. D'autres éléments peuvent également être remplacés de la même manière (*Deselect All* qui remplace *Select All*). Ces éléments "cachés" sont difficiles à découvrir pour l'utilisateur, puisqu'ils reposent sur une situation relativement improbable du menu. MenuInspector met en évidence ces éléments puisqu'il a accès à l'intégralité des éléments du menus.

DISCUSSION

MenuInspector s'est révélé utile pour plusieurs projets liés à la conception de nouvelles techniques d'interaction et d'une méthode d'optimisation. Cependant, son utilisation a aussi mis en évidence quelques limitations.

Défaut d'implémentation. Certaines applications n'ont pas codé tous les raccourcis clavier de manière correcte. Par exemple, iTunes permet de "Jouer" / "Mettre en Pause" un morceau en appuyant sur la barre d'espace. Cependant, ce raccourci n'est pas codé dans le système de menu, mais directement dans le nom de l'item (en utilisant plusieurs espaces pour l'aligner sur la droite). Ces défaut d'implémentation sont capturés par MenuInspector, mais ne sont pas détectés automatiquement, ce qui peut potentiellement biaiser une analyse des raccourcis clavier.

Menus contextuels et barre d'outils. MenuInspector ne couvre actuellement pas les interacteurs comme les menus

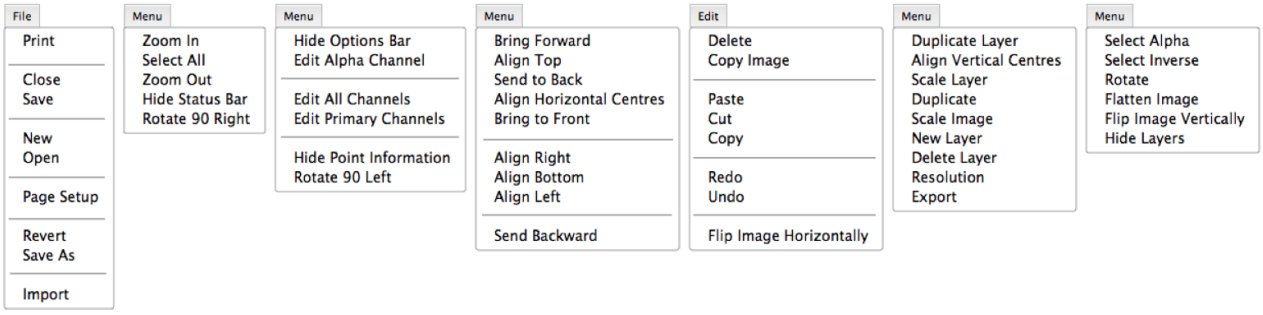

Figure 6. Optimisation de menus. MenuInspector a été utilisé pour définir un score de consistance entre un système de menus et les menus d'autres applications. Ce score a été utilisé pour générer automatiquement en 3 minutes le système de menus pour une application graphique. Par exemple, "Cut", "Copy", "Paste" sont groupés ensemble ce qui est cohérent avec les applications existantes. Cependant "Paste" est placé en haut parce qu'il est plus fréquent. Une évaluation expérimentale a montré que cet outil soulageait le travail des concepteurs [5].

contextuels ou les barres d'outils. L'API d'accessibilité ne permet pas de récupérer les menus contextuels et comporte de nombreux bugs pour les barres d'outils. Nous travaillons actuellement à outrepasser cette limitation.

Vie privée. La barre de menus ne contient généralement pas d'informations concernant l'intimité de l'utilisateur. Cependant, certains sous-menus sont générés dynamiquement comme le menu "Open Recent" qui contient le nom de certains fichiers de l'utilisateur. Nous envisageons d'identifier ces sous-menus afin de supprimer automatiquement ces informations du fichier d'historique.

CONCLUSION

Nous avons présenté MenuInspector, un outil pour l'analyse des barres de menus des applications sur la plateforme MacOSX. Nous avons également démontré son utilité à travers 4 exemples d'applications : l'étude du recouvrement, des collisions, de la consistance et des éléments cachés. Nous espérons que ces exemples d'application inspireront de futurs utilisateurs de MenuInspector pour des usages dans d'autres contextes.

REMERCIEMENTS

Cette recherche a été en partie financée par la Royal Society of New Zealand Marsden Grant 10-UOC-020. Nous remercions également Dong-Bach Vo, Simon Perrault, Quentin Roy et Mathieu Nancel.

BIBLIOGRAPHIE

1. Apple accessibility api. `https://developer.apple.com/library/mac/#documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXIntro/OSXAXintro.html`.
2. Appert, C., and Beaudouin-Lafon, M. Swingstates: adding state machines to the swing toolkit. *ACM UIST '06*, 2006, 319–322.
3. Appert, C., Beaudouin-Lafon, M., and Mackay, W. Context matters: Evaluating interaction techniques with the cis model. *Springer People and Computers XVIII — Design for Life*, 2005, 279–295.
4. Bailly, G. Techniques de menus: description, développement, évaluation. *ACM IHM '07*, 2007, 217–220.
5. Bailly, G., Oulasvirta, A., Timo, K., and Hoppe, S. Menuoptimizer: Interactive optimization of menu systems. *ACM UIST '13*, 2013, to appear.
6. Byrne, M. D., Anderson, J. R., Douglass, S., and Matessa, M. Eye tracking the visual search of click-down menus. *ACM CHI '99*, 1999, 402–409.
7. Chang, T.-H., Yeh, T., and Miller, R. C. Gui testing using computer vision. *ACM CHI '10*, 2010, 1535–1544.
8. Cockburn, A., Gutwin, C., and Greenberg, S. A predictive model of menu performance. *ACM CHI '07*, 2007, 627–636.
9. Dixon, M., and Fogarty, J. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. *ACM CHI '10*, 2010, 1525–1534.
10. Guimbretiére, F., Dixon, M., and Hinckley, K. Experiscope: an analysis tool for interaction data. *ACM CHI '07*, 2007, 1333–1342.
11. Kurtenbach, G., and Buxton, W. User learning and performance with marking menus. *ACM CHI '94*, 1994, 258–264.
12. Mackay, W. E., Appert, C., Beaudouin-Lafon, M., Chapuis, O., Du, Y., Fekete, J.-D., and Guiard, Y. Touchstone: exploratory design of experiments. *ACM CHI '07*, 2007, 1425–1434.
13. Malacria, S., Bailly, G., Harrison, J., Cockburn, A., and Gutwin, C. Promoting hotkey use through rehearsal with exposehk. *ACM CHI '13*, 2013, 573–582.
14. Matejka, J., Grossman, T., and Fitzmaurice, G. Patina: Dynamic heatmaps for visualizing application usage. *ACM CHI '13*, 2013, 3227–3236.
15. Matsui, S., and Yamada, S. Genetic algorithm can optimize hierarchical menus. *ACM CHI '08*, 2008, 1385–1388.
16. Menua. `http://gillesbailly.fr/menua/`.
17. Roy, Q., Malacria, S., Guiard, Y., Lecolinet, E., and Eagan, J. Augmented letters: mnemonic gesture-based shortcuts. *ACM CHI '13*, 2013, 2325–2328.