Immersion d'interfaces 2D dans un espace 3D

Eric Lecolinet, Matthieu Nottale

GET Télécom-Paris / CNRS - LTCI
46 rue Barrault, 75634 Paris cédex 13
elc@enst.fr

RESUME

Cet article présente *Ubit3D* une boîte à outils graphique qui permet de créer des interfaces graphiques tridimensionnelles. *Ubit3D* étend le toolkit graphique *Ubit* en apportant la possibilité de composer un nombre quelconque de surfaces 2D contenant des interacteurs classiques dans un espace de représentation 3D. Afin de pouvoir facilement déplacer et orienter ces surfaces, nous proposons également une technique de manipulation interactive basée sur l'utilisation de *Control menus*.

MOTS CLES: Interfaces 3D, visualisation d'information, gestion des documents, Ubit, Control menus.

ABSTRACT

This paper presents *Ubit3D* a toolkit for developing three-dimensional interfaces. *Ubit3D* is an extension of the *Ubit* toolkit that makes it possible to compose an arbitrary number of 2D interactive surfaces in a 3D space. An interaction technique based on *Control menus* is also proposed to move and orient these surfaces in the 3D space.

CATEGORIES AND SUBJECT DESCRIPTORS: D.2.6 [Software Engineering]: Programming Environments | Graphical environments; H.5.2 [Information Interfaces and Presentation]: User Interfaces | GUIs.

GENERAL TERMS : Algorithms, Design.

KEYWORDS : 3D Interfaces, information visualisation, document management, Ubit, Control menus.

INTRODUCTION

Dès le milieu des années 80, des chercheurs du Xerox PARC proposaient d'utiliser des représentations tridimensionnelles pour améliorer les interfaces utilisateurs [2]. Cette idée a ensuite donné lieu au développement de nombreuses techniques de visualisation de l'information [4]. Nous nous intéressons ici au cas particulier des métaphores de bureau tridimensionnelles où des interfaces graphiques 2D « classiques » sont immergées dans un espace de représentation à trois dimensions.

L'utilisation de la troisième dimension est censée apporter divers avantages que nous résumons dans la première section. Ces améliorations restent cependant assez théoriques dans la mesure où il est difficile de mettre en œuvre de tels systèmes, faute d'outils largement disponibles, et par conséquent, d'analyses et d'évaluations expérimentales.

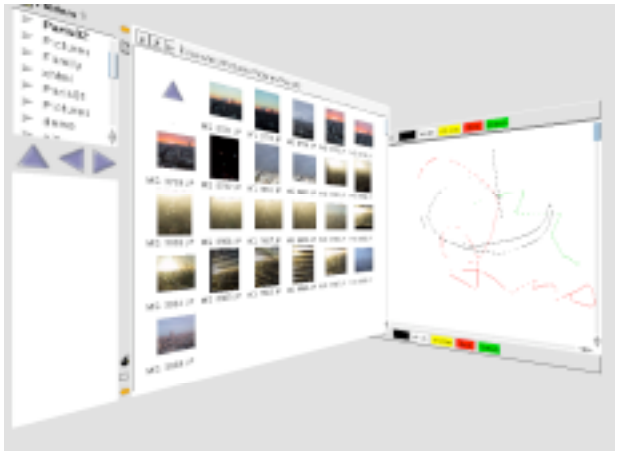

Figure 1: Immersion de deux interfaces 2D dans un espace 3D.

Le travail présenté ici tente de combler en partie cette lacune en proposant une boîte à outils qui permet de créer des interfaces graphiques tridimensionnelles. Ce toolkit, nommé ***Ubit3D*** constitue une extension de la boîte à outils 2D *Ubit* [7]. Contrairement à d'autres systèmes publiés dans des études antérieures, *Ubit3D* n'est pas un gestionnaire de fenêtre capable de positionner celles-ci dans un bureau 3D mais un toolkit graphique complet qui permet la création d'applications interactives en 3D (l'interface 3D étant composée d'un nombre quelconque de surfaces 2D contenant des interacteurs classiques, comme dans la Fig. 1). Les seconde et troisième sections résument les principes de la boîte à outils *Ubit* et l'implémentation de l'extension *Ubit3D*. Nous présentons ensuite une technique de manipulation interactive des surfaces 2D dans l'espace 3D basée sur l'utilisation de *Control menus* puis nous proposons quelques perspectives et concluons.

METAPHORES 3D

L'utilisation de la troisième dimension permet tout d'abord de représenter davantage de données à l'écran, et donc de mieux utiliser cet espace nécessairement limité en taille. Par exemple, le Web Forager [3] permet d'afficher des pages Web à taille réduite en les éloignant du point de vision (ou au contraire de les agrandir en les ramenant au premier plan). L'utilisateur peut ainsi affi-

cher plusieurs pages sur son espace de travail sans qu'elles se recouvrent. Même si certaines de ces pages ne peuvent être lues immédiatement, du fait de leur trop petite taille, elles restent néanmoins facilement identifiables et directement accessibles. L'utilisation de la 3D permet donc une certaine gradualité qui se rapproche des caractéristiques d'un bureau réel où la distance entre les documents et le lecteur est inversement proportionnelle à leur degré d'intérêt par rapport à la tâche qu'il est en train d'exécuter. En d'autres termes, la 3D permet de passer d'une métaphore du bureau « ternaire » (document ouvert, fermé, iconifié) à une métaphore « graduelle » où les documents peuvent être plus ou moins visibles, comme dans le monde réel.

La 3D permet également divers effets visuels intéressants, comme la représentation en perspective des interfaces. L'idée est la même que précédemment, fournir un niveau de gradualité correspondant au degré d'intérêt des données, sauf que cette gradualité peut intervenir à l'intérieur d'une même représentation (typiquement une fenêtre contrôlée par une application). Par exemple, les Data Mountains [5] permettent d'organiser un ensemble de signets correspondant à des pages Web en fonction de leur niveau d'intérêt (les signets étant d'autant plus proche du point de vision, et donc plus gros et plus visibles qu'il sont considérés comme intéressants par l'utilisateur). Cette étude est par ailleurs intéressante car c'est l'une des rares à avoir fait l'objet d'une étude expérimentale significative [5] qui a clairement montré l'efficacité du type de représentation proposé. Enfin les représentations 3D (ou 2D1/2) peuvent également permettre d'enrichir la dynamique de l'interaction. On citera par exemple le WebBook [3] où les pages peuvent être « feuilletées » interactivement ou le toolkit CPN2000 [1] qui permet de « plier » les fenêtres afin de faire apparaître ce qu'il y a en dessous.

Deux cas de figures peuvent se présenter sur le plan logiciel. Soit le système de représentation 3D est dédié, auquel cas les applications doivent avoir été programmées à l'aide de cet outil (par exemple [6]). C'est le cas de l'outil présenté dans suite de cet article. Notons ici pour éviter toute ambiguïté que les boîtes à outils 3D qui n'offrent pas de support spécifique pour la réalisation d'interfaces interactives (comme OpenInventor ou Java3D) n'entrent pas dans le sujet de cet article.

Soit le système de représentation est une sorte de gestionnaire de fenêtre qui permet de positionner les fenêtres d'applications 2D classiques dans un espace à trois dimensions [9,11,12]. Dans ce cas, les fenêtres sont généralement d'abord rendues dans une zone non visible de la mémoire vidéo puis affichées sous forme de textures positionnées de manière appropriée sur la surface visible de l'écran.

Le principal avantage de la seconde approche est qu'elle ne nécessite pas le développement d'applications spécifiques. C'est également sa principale limitation : comme ces applications ne peuvent « savoir » qu'elles sont représentées en trois dimensions il n'est donc pas possible d'adapter leurs techniques d'affichage et d'interaction. Par exemple, il peut être souhaitable d'afficher une application « éloignée » du point de vision, et donc de petite taille, de manière simplifiée (pour optimiser le temps d'affichage) ou différente de sa représentation à grande échelle (selon le principe du zoom sémantique). Par ailleurs l'emploi de textures peut entraîner des problèmes de lisibilité du texte si celles-ci sont de taille insuffisante ou des problèmes d'utilisation mémoire ou de performances dans le cas inverse. Ce problème est particulièrement susceptible de se produire dans le cas d'interactions incrémentales ou répétées (sélection et édition de texte, déplacement incrémentaux à l'aide d'ascenseurs, manipulation directe…).

Il est enfin nécessaire de pouvoir facilement déplacer et orienter les diverses parties des interfaces 2D dans l'espace 3D, tout en gardant les possibilités d'interaction habituelles avec l'application. Deux possibilités sont alors envisageables : soit d'utiliser des dispositifs physiques spécifiques (souris 3D, etc.) mais ceux-ci sont relativement coûteux, peu répandus et nécessitent une phase d'apprentissage ; soit d'utiliser une classique souris 2D, un dispositif aux capacités plutôt limitées par rapport aux multiples opérations que cet unique objet est alors censé contrôler. Nous proposons dans la dernière section une technique fondée sur l'utilisation de *Control menus* qui tente d'apporter une réponse à ce problème. Notons enfin que, de même que pour l'affichage, il peut être souhaitable de pouvoir adapter l'interaction au niveau des applications : un mécanisme trop général risque d'être soit inutilement complexe, soit inapproprié pour certaines applications.

LE TOOLKIT UBIT

Le toolkit **Ubit** [8, 14] présente la particularité d'être basé sur une architecture hybride, dite « moléculaire », qui combine les avantages des systèmes basés sur l'utilisation d'arbres de widgets et de graphes de scènes. La plupart des toolkits graphiques 2D sont fondés sur la notion de widgets, c'est-à-dire des composants logiciels opaques qui encapsulent un nombre relativement important de données et comportements. L'enrichissement des widgets se fait principalement par héritage, un mécanisme statique qui nécessite une phase de compilation avec la plupart des langages à objets courants. Les capacités d'enrichissement sont de plus généralement limitées en pratique : l'encapsulation de nombreuses propriétés et comportements rend difficile, voire impossible, toute modification en profondeur de ces objets. Les widgets peuvent donc être vus comme des « boîtes noires » dont

le degré d'abstraction est relativement élevé mais qui offrent assez peu de flexibilité et des capacité d'adaptation limitées. A contrario, les architectures à base de graphes de scènes, qui sont principalement utilisées dans les toolkits 3D, offrent une bien plus grande flexibilité. Elles sont basées sur la combinaison dynamique de nombreux objets de faible granularité dans le graphe d'instance (selon le principe de décoration des « Design patterns »). Mais cette flexibilité à un prix : l'obligation d'avoir à composer de nombreux objets élémentaires au lieu de disposer de composants complexes offrant un « look and feel » prédéfini. Ces toolkits ont de plus d'abord été conçus pour faciliter le rendu et n'offrent généralement que des capacités d'interaction de bas niveau.

L'architecture « moléculaire » du toolkit Ubit intègre ces deux niveaux d'abstraction. Elle comporte d'une part des *briques* élémentaires qui modélisent des comportements, des propriétés graphiques ou des objets de représentation atomiques et d'autre part des *conteneurs génériques* qui supportent la combinaison dynamique de briques élémentaires (ou d'autres conteneurs). Ainsi, un « widget » *Ubit* (par exemple un bouton, un champ textuel, etc.) n'est en réalité qu'un conteneur générique contenant diverses briques (de même qu'une molécule est composée de divers atomes). Ce modèle est très flexible dans la mesure ou n'importe quel widget peut être complètement modifié en y ajoutant de nouvelles briques (par exemple, n'importe quel interacteur de base peut être transformé dynamiquement en n'importe quel autre ou en un objet hybride combinant plusieurs comportements). Une interface graphique peut donc être vue selon deux niveaux de granularités : soit comme graphe de pseudo-widgets, soit comme un graphe de scène.

Ce modèle offre divers possibilités dont la description détaillée sort du cadre de cet article (voir [7] pour plus de détails). En particulier, n'importe quelle brique (ou conteneur, un conteneur n'étant qu'un type de brique particulier) peut être partagée par plusieurs parents. Le partage de briques élémentaire permet de synchroniser implicitement plusieurs widgets (par exemple une chaîne de caractère dont le contenu sera automatiquement mis à jour dans tous les widgets qui la contiennent). Le partage de widgets entraîne leur réplication graphique à l'écran, ceux-ci pouvant apparaître autant de fois qu'il existe de chemins entre la « racine » du graphe de scène (qui a une structure de DAG, c'est-à-dire de graphe acyclique orienté). Ces réplications graphiques sont automatiquement synchronisées mais ne sont pas nécessairement identiques visuellement, divers mécanismes de paramétrage pouvant être mis en œuvre. Elles peuvent en outre intervenir sur des écrans de machines distantes, ceci permettant la réalisation d'interfaces multi surfaces.

L'EXTENSION UBIT3D

La version courante du toolkit Ubit repose sur la librairie *X Window*. Cependant le rendu peut indifféremment être effectué en utilisant cette librairie ou *OpenGL*. Cette propriété à été à la base de l'implémentation de l'extension ***Ubit3D***. Cette extension a nécessité quelques modification relativement limitées du toolkit *Ubit*. Comme expliqué précédemment, ce toolkit repose sur l'utilisation de briques élémentaires et de conteneur génériques. La première évolution a consisté en l'ajout d'un nouveau conteneur (nommé *U3Dbox*) qui permet l'affichage et l'orientation d'une surface 2D dans l'espace 3D. Ce conteneur est compatible avec les autres objets du toolkit et peut être inséré à n'importe quel niveau du DAG de scène. Les enfants de ce conteneur sont alors affichés sur la surface qu'il définit dans l'espace 3D. Une interface 2D classique (ou n'importe quelle sous partie la constituant) peut ainsi être affichée en 3D en rajoutant un nœud (ou autant qu'il y a de sous parties orientées) dans le DAG de scène. Ce modèle est donc à la fois simple et général dans la mesure où il ne se limite pas à l'affichage des fenêtres des applications dans un espace 3D (comme le ferait un gestionnaire de fenêtres tridimensionnel). Une application peut afficher autant de sous parties orientées que nécessaire (par exemple pour représenter un mur perspectif ou permettre le feuilletage des pages d'un livre).

L'implémentation du conteneur *U3Dbox* repose sur l'utilisation des matrices de projection et du ZBuffer *d'OpenGL*. L'affichage des nœuds du DAG de scène *Ubit* est réalisé en profondeur d'abord, puis de la gauche vers la droite, les éléments élémentaires étant tous affichés en utilisant le même contexte graphique. Cet algorithme reste valable en 3D à condition de modifier la matrice de projection *d'OpenGL* de manière appropriée lorsque l'on traverse un conteneur. Il nécessite également l'utilisation du ZBuffer lorsqu'un *U3Dbox* a plusieurs enfants directs. En 2D, l'affichage de la gauche vers la droite des noeuds d'un même niveau a pour effet de faire apparaître les premiers « au dessous » des derniers. Le principe reste le même lorsque les surfaces 2D sont immergées dans l'espace 3D sauf qu'il faut gérer correctement les possibles intersections entre ces surfaces. Ce problème est résolu en affectant à chaque conteneur une coordonnée en Z qui correspond à sa profondeur dans le DAG de scène.

Les autres routines d'affichage n'ont pas nécessité de modification sauf en ce qui concerne la gestion des polices de caractères. Nous avons utilisé la librairie *texFont* [13], qui représente les caractères sous forme de textures, afin de pouvoir afficher du texte en trois dimensions. Enfin le système de gestion des événements a été adapté de manière à effectuer les conversions de coordonnées adéquates.

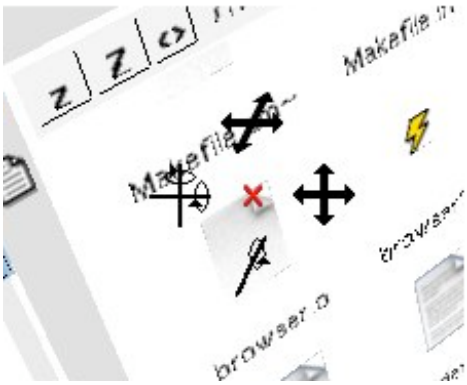

Figure 2 : Positionnement et orientation par Control menu

POSITIONNEMENT PAR CONTROL MENUS

Les représentations tridimensionnelles proposées précédemment ne sont réellement intéressantes que si elles peuvent être facilement contrôlées par l'utilisateur. Il faut de plus éviter de perturber l'interaction avec les interacteurs habituels, même s'ils sont affichés en 3D. Le problème posé nécessite donc de pouvoir facilement modifier six paramètres numériques (trois de position et trois d'orientation) tout en utilisant le plus petit nombre possible de boutons de la souris et/ou de modifieurs du clavier. Nous avons choisi d'utiliser un *Control menu* [8,10], un nouveau type d'interacteur qui offre la particularité de pouvoir sélectionner une opération puis de la contrôler interactivement en un même geste. Cette propriété semble bien adaptée à la nécessité de pouvoir contrôler 6 paramètres numériques quasi-simultanément.

De même que les Pie et Marking menus dont ils s'inspirent, les *Control menus* sont des menus circulaires. L'utilisateur n'a plus besoin de sélectionner une ligne dans une liste linéaire mais peut tout simplement déplacer le pointeur dans la bonne direction et relâcher le bouton de la souris. La particularité des *Control menus* est que le geste effectué permet également de contrôler l'opération sélectionnée interactivement. Par exemple, après avoir sélectionné la dimension « x » en déplaçant le pointeur sur le bouton correspondant du menu, l'utilisateur pourra ensuite choisir sa valeur en continuant à déplacer le pointeur. Ce type de menu offre également la possibilité d'implémenter un « mode expert » en introduisant un délai avant que le menu n'apparaisse à l'écran. L'interaction devient alors purement gestuelle dans le cas d'une utilisation experte, le geste restant identique à celui de l'interaction de base mais la rapidité de son exécution évitant que l'utilisateur ne soit distrait par l'apparition du menu.

Un Control menu peut permettre de contrôler huit paramètres différents (deux par direction horizontale, verticale et diagonales à 45 et 135°) ou davantage si des sous-menus cascadés sont utilisés. Notre implémentation préliminaire n'utilise que les directions horizontale et verticale (Fig. 2). Ceci permet d'accéder à quatre sous-menus, chacun contrôlant deux paramètres (un verticalement et un horizontalement). Ce menu permet donc de sélectionner et contrôler huit paramètres sans avoir à utiliser d'autres interacteurs. Sa manipulation s'effectue en appuyant sur le bouton du milieu de la souris. Un seul bouton (ou alternativement, un seul modifieur du clavier) suffit donc à contrôler le positionnement des surfaces 2D.

PERSPECTIVES ET CONCLUSIONS

La combinaison des techniques de représentation et d'interaction présentées précédemment nous semble offrir une voie de recherche prometteuse. Cependant, la validation de ce type d'approche nécessitera la réalisation et l'évaluation d'applications réalistes ainsi qu'une mise au point fine des techniques d'interaction proposées. Le principal apport du toolkit domaine public *Ubit3D* est de fournir les outils logiciels nécessaires pour entreprendre ce type d'étude.

BIBLIOGRAPHIE

1. Beaudouin-Lafon M. *Novel interaction techniques for overlapping windows*. UIST 2001, pp 153-154. ACM Press.
2. Card S., Robertson G., Mackinlay J. *The information visualizer*. CHI 1991, pp.181-188. ACM Press
3. Card S, Robertson G.,York W. *The WebBook and the Web Forager: An Information Workspace for the World-Wide* Web, CHI 1996 p 111, ACM Press.
4. Card S., Mackinlay, J., Scneiderman B. *Readings in Information Visualization*:. 1999, Morgan Kaufman.
5. Czerwinski M. et al., *The Contribution of Thumbnail Image, Mouse-over Text and Spatial Location Memory to Web Page Retrieval in 3D*, INTERACT 1999. IOS Press.
6. Dumas C. et al. *SpIn : a 3-D Interface for Cooperative Work* Virtual Reality Society Journal, 1999, Springer-Verlag
7. Lecolinet E., *A molecular architecture for creating advanced interfaces*, UIST, pp. 135-144. 2003. ACM Press.
8. Lecolinet E., Pook S., *Interfaces zoomables et Control menus: Techniques focus+contexte pour la navigation interactive dans les bases de données*, Cahiers du numérique. Vol.3, pp. 191-210. 2002 . Hermès.
9. Looking Glass. http://wwws.sun.com/software/looking_glass
10. Pook S., Lecolinet E., Vaysseix G., Barillot E., *Control Menus: Execution and Control in a Single Interactor*. CHI 2000. pp. 263-264. ACM Press
11. Robertson G., et al. *The task gallery: a 3D window manager*. CHI, pages 494-501, 2000. ACM Press.
12. Roussel N. VideoWorkspace : *une boîte à outils pour l'exploration de nouvelles techniques de gestion de fenêtres*. IHM 2002, pages 271-274, ACM Press.
13. Texfont : http://is6.pacific.net.hk/~edx/texfont.htm
14. Ubit et Ubit3D : www.infres.enst.fr/~elc/ubit