SYNCHRONISATION DE PLUSIEURS MODES VISUELS POUR AIDER A LA PROGRAMMATION DES UTILISATEURS NON-PROGRAMMEURS

Valérie Bellynck

CUI
Université de Genève
Groupe de recherche en programmation visuelle et génie logiciel
Rue du Général Dufour 24
CH-1211 Genève 4
SUISSE

Leibniz
Imag
Equipe EIAH - projet Cabri-géomètre
46 avenue Felix Viallet
38051 Grenoble cedex
FRANCE

Résumé *— Nous présentons un prototype dans lequel plusieurs modalités sont offertes à l'utilisateur pour manipuler les objets qu'il construit dans un micro-monde d'apprentissage. Les multimodalités d'accès visent à rendre abordables les tâches de programmation à des utilisateurs de tous profils. Par la vue graphique, l'utilisateur accède aux effets de ses choix et manipule le résultat de ses constructions. Par la vue textuelle du programme, il accède à la structure logique de la construction et donc du programme sous-jacent, qu'on aurait pu (et qu'on pourra) tout aussi bien proposer sous la forme d'un graphe. Les deux vues proposées correspondent bien à deux modalités, car elles aboutissent au même programme généré et au même résultat, et car le passage de l'une à l'autre est transparent pour l'utilisateur. La complémentarité des deux vues induit un apprentissage implicite du paradigme de programmation et permet d'aborder des concepts informatiques importants et classiques. Inversement, le langage "narratif" induit des manipulations de l'interface, et qui décrit le programme de construction en cours, permet d'imaginer une nouvelle approche pour concevoir les interfaces graphiques de certains logiciels à manipulation directe.*

1. Motivation

De nombreux logiciels offrent des interfaces dans lesquelles l'utilisateur peut concevoir son environnement de travail, et définir ses propres outils. D'autres logiciels offrent aux utilisateurs de tous profils un environnement de conception de leur propre application. La limite entre les deux points de vue devient tangible pour certains logiciels.

En particulier, dans les micro-mondes d'apprentissage, où l'approche constructiviste conduit à des activités de nature programmatoire, l'utilisateur professeur peut préparer un environnement de travail pour ses élèves, qui, au delà d'un seuil d'adaptation, peut être considéré comme une application.

Différents moyens sont proposés aux utilisateurs, allant de la programmation visuelle [12] à la programmation par démonstration ou par l'exemple [6], en passant par les recherches actuelles sur la programmation naturelle [13].

Dans la programmation visuelle, il s'agit de remplacer la nature textuelle des programmes par l'utilisation à plus ou moins grande échelle de l'image [9]. La programmation visuelle contient différentes approches graphiques permettant de spécifier des programmes.

La programmation à l'aide d'un langage visuel permet de spécifier la structure logique du programme avec des graphes, ou avec des graphiques associés à des grammaires.

La programmation par l'exemple permet de fabriquer des programmes en donnant des exemples, et ainsi de travailler sur des exemples concrets plutôt que sur la description de procédures dans l'abstrait.

La programmation par démonstration est basée sur l'apprentissage automatique des processus utilisés par l'utilisateur lors d'activités de conception sur ordinateur [7]. Le sens du mot « démonstration » est ici pris dans le sens suivant: on montre au logiciel ce qu'il doit faire, et le logiciel produit un programme à partir des manipulations de l'utilisateur.

Cependant, la place du graphique comme aide au raisonnement est toujours controversée et réapparaît aujourd'hui en informatique avec les interfaces graphiques.

Certains systèmes allient les deux modes de programmation selon le profil des utilisateurs (informaticiens ou non), et le niveau de leur intervention dans le produit à développer, comme le système DIVA qui permet une programmation hétérogène textuelle et graphique [8, 10].

Certaines recherches émergentes ne renoncent pas à la composante textuelle ou verbale, mais cherchent plutôt à ce que le système s'adapte aux concepts de nature programmatoire manipulés par des utilisateurs novices [13].

Nous proposons une autre approche, qui consiste à allier les différents modes d'accès au programme au travers d'une représentation de ses données et de sa structure logique. Il s'agit de permettre à l'utilisateur de travailler sur les différentes représentations en même temps, indifféremment, et de façon équivalente. Chaque vue du programme doit être dynamique et synchronisée avec les autres.

2. Modalité géométrique de base

Le travail présenté ici s'inscrit dans le cadre des recherches liées au projet IMAG Cabri-géomètre [11, 1, 2].

Cabri-géomètre, le Cahier de BRouillon Interactif, constitue un micro-monde d'apprentissage pour le cas de la géométrie et est aussi appliqué à d'autres domaines mathématiques. Son interface présente une feuille blanche de $1m^2$ vue au travers d'une fenêtre, et une barre d'outils, qui sont les équivalents interactifs de la règle et du compas, ainsi que des outils de mesure, et quelques compositions usuelles de ces outils (transformations géométriques, calculatrice, table).

L'utilisateur peut construire des objets non seulement géométriques, mais aussi numériques et textuels, grâce aux outils de base. La construction finale est élaborée incrémentalement comme une juxtaposition de constructions élémentaires. Pour chacune des constructions, l'utilisateur s'appuie sur le retour d'information que lui fournit l'interface et qui l'informe de la manière dont l'outil courant peut utiliser l'objet survolé par le curseur. Ainsi, chaque construction élémentaire est le résultat d'une action de l'utilisateur sur les objets déjà construits, par manipulation directe de ces objets.

L'utilisateur peut manipuler la figure construite et la déformer. Pour cela, il lui suffit de saisir un des objets de base de la figure et de le déplacer. Les propriétés de définitions restent vérifiées : ce sont donc des contraintes de la figure.

Les constructions élémentaires peuvent être regroupées pour définir de nouveaux outils qui constituent des sous-programmes de construction applicables à d'autres objets.

Ces activités ont une nature programmatoire évidente, et sont effectuées par des utilisateurs de tous profils, de tous niveaux de familiarité avec l'outil informatique (professeur / élève / chercheur, novice / expert en programmation). Leurs motivations vis à vis de l'apprentissage de concepts informatiques diffèrent.

Les difficultés rencontrées par l'utilisateur sont de deux ordres :

- les objets graphiques manipulés directement par les utilisateurs sont des instances de variables du programme de construction,
- les macros, redéfinitions, et éliminations d'objets nécessitent un accès à la structure logique du programme pour que leurs effets soient anticipables par les utilisateurs.

Les objets graphiques manipulés constituent le résultat d'une exécution sur des valeurs particulières : les valeurs des objets de base de la construction. C'est sur les représentants graphiques que l'utilisateur agit et ressent exécuter des actions. L'image obtenue (résultat graphique) ne contient pas de trace explicite des causes, c'est-à-dire des actions de l'utilisateur : on n'y trouve que les effets, c'est-à-dire les conséquences des choix. Cela rend les contraintes indiscernables des propriétés et conduit à des difficultés de mise au point des programmes. Pour remédier à cet inconvénient, il faut avoir accès, d'une manière ou d'une autre, à une vue de la structure interne du programme.

3. Nouvelle modalité textuelle

Nous avons commencé par introduire dans le logiciel une vue textuelle, à cause de sa lisibilité immédiate. Ultérieurement, nous voulons introduire aussi une forme graphique [6]. L'introduction d'une telle vue, bien que permettant la programmation textuelle, n'est pas un retour aux langages de commandes : ce n'est pas le résultat du texte qui est exécuté, mais la partie utile de ce qui a été exécuté pour construire la figure (et qui est exécuté en permanence pour la dessiner) qui est visualisé par un texte.

Les vues textuelle et géométrique sont synchronisées, avec sélection simultanée des objets dans la forme correspondant à chaque vue. On parle d'ubiquité, car un objet est matérialisé en plusieurs lieux à la fois (un dans la figure et en général plusieurs dans le texte).

L'illustration suivante montre une copie de l'écran, dans laquelle l'utilisateur fait passer le curseur au dessus du cercle C3 dans la fenêtre graphique, alors que l'outil cercle est l'outil courant. Le programme de construction se construit "en écho" dans la fenêtre textuelle. L'utilisateur peut tout aussi bien manipuler les représentants des objets dans la fenêtre textuelle avec un "écho" dans la fenêtre géométrique, ou encore passer d'une fenêtre à l'autre.

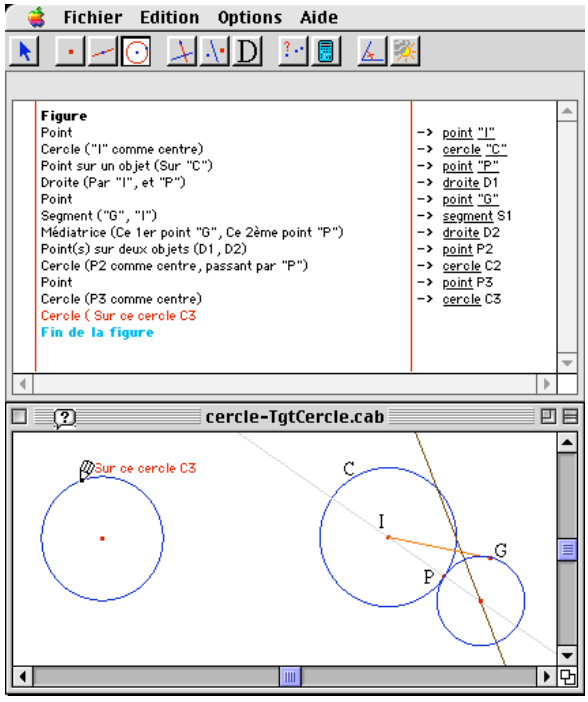

Figure 1 *— Construction depuis la fenêtre géométrique avec écho dans la fenêtre textuelle..*

La vue graphique sera ultérieurement ajoutée selon le même principe, et insérée dans la même architecture logicielle. Nous en avons proposé plusieurs formes dans [3, 4]. On peut aussi imaginer des vues dans

d'autres modalités, comme par exemple une "vue" auditive associée à des manipulations vocales.

4. Apport de la bimodalité

Cette bimodalité permet non seulement d'aider l'utilisateur de Cabri-géomètre à concevoir ses applications, mais lui offre aussi un contexte nouveau et efficace pour se familiariser avec plusieurs aspects importants en programmation :

1. Analyse d'un problème et formulation abstraite de la solution,
2. Algorithmisation au niveau d'objets informatiques abstraits (par exemple, les objets géométriques de Cabri),
3. Assimilation des outils algorithmiques (structures de contrôle et de données) du langage et de l'environnement,
4. Savoir-faire en débogage (détection de dysfonctionnement et planification de remédiation, d'ajustement de la stratégie) par passages de raffinements successifs entre les étapes 2 et 3.

En particulier, dans le micro-monde constitué par Cabri, la manipulation directe des objets abstraits et le retour d'information, donné préventivement pour guider l'utilisation des outils, minimise la distance entre les points 2 (manipulation d'objets informatiques abstraits) et 3 (assimilation d'outils algorithmiques). Cela favorise l'acquisition de compétences en débogage (étape 4).

Non seulement les programmes sont exécutés au fur et à mesure de leur édition dans la vue géométrique, mais en plus les bogues de l'utilisateur apparaissent comme des décalages entre les comportements désirés des objets qu'il a construits et ceux qui sont révélés par les animations de sa construction.

En fait, notre vue textuelle permet même de naviguer entre différents niveaux d'abstraction. Au niveau le plus bas, on peut voir toutes les informations mémorisées sur chacun des objets géométriques. Au plus haut niveau, on dispose d'une vue synthétique de toute la construction grâce à l'affichage "plié" des macro-constructions utilisées.

Grâce à la vue textuelle, les utilisateurs peuvent aussi aborder aisément la notion de procédure, de paramètre, et de programmation structurée, en examinant les macro-constructions élaborées automatiquement. En effet, la notion de macro en Cabri est très proche de celle de procédure. La vue textuelle permet de voir les macros, alors qu'elle sont impossibles à visualiser dans la forme géométrique qui ne montre que le résultat des constructions. On comprend beaucoup mieux ce que fait le logiciel quand il fabrique les macros, et on apprend ainsi "par l'exemple". De plus, la différence entre paramètre et variable est perceptible dans le programme par la forme de l'identificateur et par son identification multiple.

Les principales difficultés techniques dans notre implémentation proviennent de la nécessité que les différentes vues réagissent simultanément à chaque action associée à une manipulation de l'interface, et cela de manière adaptée aux contraintes particulières de la vue, et même de façon optimale par rapport à ses spécificités. À titre indicatif, notre implémentation représente un surcoût de 7% de code (sur 1,09 Mo) et de 22% des données (sur 107 Ko), sachant qu'on pourrait réduire ce facteur pour les données.

5. Conclusions

Nous avons donc démontré l'intérêt d'introduire la multimodalité dans un environnement au départ uniquement graphique. Pour l'instant, il s'agit de bimodalité, et l'apport essentiel se situe au niveau de l'aide à la construction des figures et à la compréhension de notions importantes en programmation.

Dans le futur, nous souhaitons introduire d'autres vues, dans les mêmes modalités et peut-être dans la modalité auditive, pour expérimenter et comparer leurs apports respectifs.

Bibliographie

[1] Franck Bellemain, "*Conception, réalisation et utilisation d'un logiciel d'aide à l'enseignement de la géométrie†: Cabri-géomètre*", Thèse, Université Joseph Fourier, Grenoble, 1992.

[2] Franck Bellemain, "*Micromonde, manipulation directe et enseignement de la géométrie†: un éclairage pour comprendre l'évolution de Cabri-géomètre vers Cabri-géomètre II*", Université d'été "Cabri-géomètre" de l'ordinateur à la calculatrice, De nouveaux outils pour l'enseignement de la géométrie, ed. IREM de Grenoble, Grenoble, 1996, p.171-202.

[3] Valérie Bellynck, "*Multimodal Visualization of Geometric Constructions*", COLING-ACL'98 Workshop on Content Representation and Multimodal Visualization, CVIR'98 , Montréal, Canada, Août 1998.

[4] Valérie Bellynck, "*Introduction d'une vue textuelle synchronisée avec la vue géométrique primaire dans le logiciel Cabri-II*", Thèse, Université Joseph Fourier, Grenoble, octobre 1999.

[5] Burnett Margaret M. and Marla J. Baker, *A Classification System for Visual Programming Languages,* Journal of Visual Languages and Computing, 287-300, September 1994.

[6] Yves Carbonneaux, *"Conception et réalisation d'un environnement informatique sur la manipulation directe d'objets mathématiques, l'exemple de Cabri-graphs*", Thèse, Université Joseph Fourier, Grenoble, janvier 1998 .

[7] Allen Cypher, and al., "*Watch What I Do: Programming by Demonstration*", publish. by Cypher A., in†MIT Press, 1993.

[8] Martin Erwig, Brend Meyer, "*Heterogeneous Visual Languages - Integrating Visual and Textual Programming*", in 11th IEEE Symp. on Visual Langages, Darmstadt, 1995, pp. 318-325.

[9] Patrick Girard, *"Environnement de programmation pour non-programmeurs et paramétrage en conception assistée par ordinateur : le système Like"* ,Thèse, Université de Poitier, France, juillet 1992.

[10] Ibrahim Bertrand, *"Diagrammatic representation of data types and data manipulations in a combined data- and control-flow language",* IEEE International Symposium on Visual Languages, Halifax, Canada, September 1998.

[11] Colette Laborde, Jean-Marie Laborde, "*Micromondes et environnements d'apprentissage*", in†: Bellissant C. (ed.) Actes des XIII Journées francophones sur l'informatique. Grenoble†: IMAG & Université de Genève, 1991, pp.157-177.

[12] Myers Brad A. *"A Brief History of Human Computer Interaction Technology"*, rapport de recherche CMU-CS-96-163 et CMU-HCII-96-103 de l'université de Carnegie Mellon, Pittsburg, décembre 1996.

[13] Pane, B.A. Myers, "Usability Issues in the Design of Novice Programming Systems," Carnegie Mellon University, School of Computer Science Technical Report CMU-CS-96-132, Pittsburgh, PA, August 1996, 85 pages.

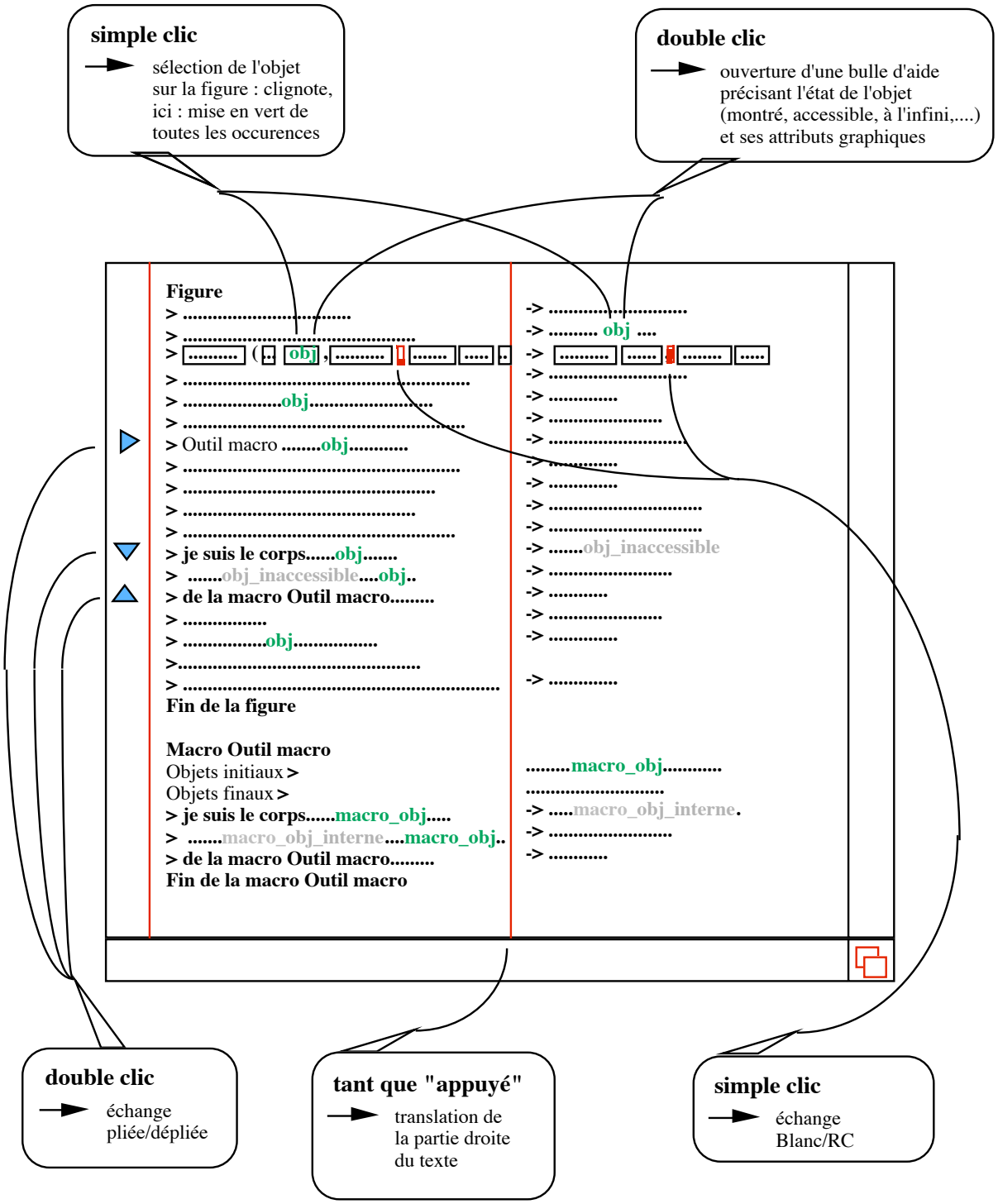

Figure 2 *— Manipulations de la fenêtre textuelle*