Cahiers GUTenberg

METAPOST, L'INTELLIGENCE GRAPHIQUE

Denis ROEGEL

Cahiers GUTenberg, nº 41 (2001), p. 5-16.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2001___41_5_0>

METAPOST, l'intelligence graphique

Denis ROEGEL

LORIA
Campus scientifique
BP 239
54506 Vandœuvre-lès-Nancy cedex
roegel@loria.fr

Résumé. Cet article examine quelques liens entre METAPOST et les dessins qu'on est amené à produire avec cet outil. Nous argumentons en faveur d'un format textuel et nous développons l'idée d'« intelligence graphique ». Nous illustrons comment cette intelligence se manifeste dans un exemple de construction de graphe.

Abstract. *In this article, we position METAPOST with respect to various types of drawings. We argue in favor of a text based format and we develop the concept of "graphical intelligence." We illustrate how this intelligence appears on a graph construction example.*

1. Introduction

METAPOST [Hobby, 1992] est un outil de programmation graphique, orienté essentiellement vers les dessins techniques. Il a été créé par John Hobby à partir du programme METAFONT, un système de création de caractères pour TEX, développé par Donald Knuth [Knuth, 1986]. La compilation d'une description METAPOST d'un dessin produit un fichier POSTSCRIPT. METAPOST n'a pas la prétention d'être un outil universel. C'est un outil qui a ses limites et qui est destiné à résoudre un type particulier de problèmes. Dans cet article, nous voulons expliciter quelque peu ce qu'est cet outil, le type de dessins qu'il peut traiter, et finalement ce qu'il demande à son utilisateur pour lui donner satisfaction.

Un premier exemple d'utilisation de METAPOST est donné en figure 1. Cet exemple montre la détermination du barycentre G d'un triangle comme intersection des médianes. La position du point G a été déterminée par METAPOST, si bien qu'un déplacement des sommets du triangle permet d'avoir un positionnement toujours correct de G. Le même dessin peut être obtenu avec un éditeur graphique, mais le résultat ne sera en général pas aussi bon, et nécessitera un repositionnement de G si le triangle est modifié. Cet exemple

permet d'illustrer que pour réaliser un dessin correct, il faut connaître les relations entre ses différentes composantes. Le dessin à réaliser ou à reproduire doit être analysé et cette analyse sera d'autant plus poussée que le dessin est complexe. Un exemple plus élaboré sera donné en fin d'article.

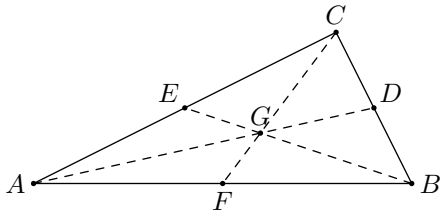

FIG. 1 – Le barycentre d'un triangle

Pour développer notre sujet, nous allons donc nous intéresser à cette tâche en apparence particulière, mais en réalité très générale, à savoir la reproduction de dessins. En fait, ce terme recouvre presque toute création artistique où un dessin est reproduit soit à partir d'une représentation préexistante, soit à partir d'une représentation mentale. Pour simplifier, nous considérons la reproduction d'un document « graphique » à l'aide d'un outil informatique.

2. Différentes représentations graphiques et différents objectifs

Le problème de la reproduction est intimement lié à la nature de la représentation graphique. Considérons pour fixer les idées quelques exemples de documents de nature « graphique » :

– un texte manuscrit ;
– le même texte imprimé, par exemple avec la présente police ;
– une tâche produite par la chute d'un encrier ;
– une photographie d'une montagne, par exemple le Cervin ;
– une peinture, par exemple les *Noces de Cana* de Véronèse ;
– un dessin technique.

Il importe de comprendre que reproduire l'un de ces documents n'a pas le même sens selon le document et selon le logiciel utilisé.

Par exemple, pour un texte manuscrit, si seul le texte est important, c.-à-d. le contenu, il suffira pour le reproduire d'employer un éditeur ou un traitement

de texte, en complétant éventuellement le texte de quelques informations non textuelles (endroits des coupures, interligne, marges, etc.) ; si le texte manuscrit doit être reproduit aussi fidèlement que possible, sans pour autant scanner le document, le texte peut être approximé par une suite de points ou de segments ; il peut être souhaitable de considérer un cas intermédiaire, celui où l'on souhaite produire du texte, comme s'il avait été écrit par une personne ; pour ce faire, on peut par exemple considérer un certain nombre de dessins de lettres (certaines éventuellement en plusieurs exemplaires) ; cette option s'apparente à la création d'une police.

La reproduction d'un texte imprimé peut consister à trouver (ou créer) la police utilisée, mais ceci peut être rendu ardu par les différentes imperfections des caractères.

Une tâche d'encre est assez difficile à reproduire fidèlement sans utiliser une technique de traçage qui en extrait les contours ; toutefois, dans certains cas, la tâche d'encre peut être simplifiée en une forme plus facile à réaliser automatiquement ; seule une bonne compréhension du dessin permet de savoir si l'on a besoin d'une tâche quelconque, ou si c'est exactement de cette tâche-là dont on a besoin.

Une photographie ou une peinture posent encore davantage la question de la fidélité de la reproduction.

Enfin, pour des dessins techniques, la tâche de reproduction est beaucoup plus simple, car les constituants du dessin sont en général élémentaires ; la reproduction peut alors viser à respecter la cohérence des traits, les jonctions des segments, les alignements, la position du texte, etc. C'est fondamentalement à ce type de dessins qu'est destiné METAPOST, en particulier si de nombreuses propriétés peuvent s'exprimer de manière mathématique.

Dans tous les cas, la reproduction doit se fixer un objectif à accomplir. Une *re-production* est une nouvelle production de l'image, et entretient avec celle-ci une certaine proximité. Une reproduction sera fidèle à une certaine composante de l'image. Il ne s'agira pas nécessairement d'une composante spatiale (comme la couleur, la position, etc.), mais peut-être d'une composante plus contextuelle (par exemple la cohérence, ou d'autres éléments plus subjectifs), etc. Le plus souvent, on ne cherchera pas à produire un fac-similé, qui pourrait se réaliser de manière satisfaisante, au moins au niveau de l'apparence, en scannant ou photographiant l'original. On cherchera en réalité le plus souvent à simplifier ou améliorer l'original.

Sans vouloir être exhaustif, on peut proposer comme exemple d'amélioration la réalisation d'un dessin plus proche de ce qu'aurait dû être le dessin initial (qui ne respectait peut-être pas les symétries, les points de fuite, etc.), éventuellement en tirant mieux profit de l'outil utilisé. On peut aussi cher-

cher à produire le même dessin, mais de manière plus abstraite, en libérant le créateur de tâches pouvant facilement être automatisées. Le même dessin peut aussi être représenté de manière plus compacte. On peut imaginer bien d'autres améliorations, mais elles fournissent d'ores et déjà une piste à la question « pourquoi améliorer ? ». Outre le fait que l'amélioration d'un dessin représente une satisfaction de l'esprit, elle a aussi des retombées techniques : l'amélioration d'un dessin peut rendre plus facile la création de variantes, peut permettre d'obtenir des dessins plus homogènes, entre eux ou avec leur environnement (voir les nouvelles figures METAPOST de *The art of computer programming* [Knuth, 1997]), etc.

3. Reproduction intelligente et compréhension

Reproduire exactement et fidèlement un dessin technique est un processus différent de l'amélioration (ou stylisation). Le second processus fait intervenir la compréhension.

Pour reproduire efficacement un dessin, il faut savoir distinguer ce qui est important de ce qui ne l'est pas ; il n'y a pas de règle générale ; dans certains cas, on voudra par exemple que la reproduction inclut tel défaut de l'original, dans d'autres cas, il faudra le supprimer.

La stylisation peut se faire plus ou moins abstraitement, et plus ou moins précisément. Elle peut être abstraite et imprécise (« un cercle »), abstraite et précise (« un cercle de rayon ... centré en ... »), peu abstraite et précise (liste de points, au lieu d'un procédé explicitement itératif), etc.

La précision se traduit souvent — mais pas toujours — sous forme textuelle ou formelle. Elle peut aussi s'exprimer graphiquement, mais souvent au prix de l'introduction de conventions peu standard (par exemple un symbole particulier pour représenter une dépendance dynamique). En particulier, la description complète d'un dessin doit expliciter le processus de construction de ce dessin, spécialement s'il contient des contraintes dynamiques.

FIG. 2 – Drapeau avec un défaut

Une approche trop automatique et pas assez intelligente de la reproduction peut conduire à des bizarreries, des artéfacts. Par exemple, si on prend le des-

sin de la figure 2, qui contient quelques tâches plus claires à l'intérieur du cercle, on pourrait imaginer faire une copie à haute résolution (bitmap), puis d'analyser les contours et déterminer où la couleur est unie. Il ne sera en général pas simple de déterminer que le contour est un cercle et qu'il délimite deux couleurs. En cas de défaut de l'original, la reconnaissance aurait du mal à se débarasser de ces défauts et pourrait nécessiter un avis humain pour indiquer qu'on est en présence d'un cercle et qu'aucune autre courbe n'apparaît à l'intérieur de ce cercle.

Quoique cette approche soit viable, il semble en fait beaucoup plus simple de reconnaître intuitivement qu'on a un cercle et deux zones colorées et que l'image contient de petits défauts, peut-être liés aux moyens de reproduction, mais qui doivent être ignorés.

Dans tous les cas, une bonne reproduction, qu'elle soit automatique, ou semi-automatique, nécessite donc la compréhension du dessin. Il s'agit d'expliciter sa structure, les relations entre ses différentes composantes, ce qui y est important, ce qui y est accessoire, ce qui peut en améliorer l'esthétique, etc. ; il s'agit aussi de déceler tout ce qui peut en faciliter la reproduction.

Et comprendre un dessin nous apprend à le voir différemment. L'utilisation de METAPOST, en ce qu'elle nous incite à comprendre, développe ce sens. On pourrait dire que ce logiciel développe « l'intelligence graphique ».

4. Formats textuels et graphiques

METAPOST fournit une aide à la réalisation de dessins techniques en permettant d'exprimer ceux-ci et leurs composantes sous forme textuelle. D'autres systèmes peuvent adopter une approche plus visuelle (voir figure 3). Avec un format textuel, la stylisation, la géométrie et la dynamicité sont prises en compte naturellement.

Une expression entièrement textuelle et de haut niveau d'un dessin présente en fait beaucoup d'avantages. Nous pouvons en particulier citer les suivants, sans prétendre à l'exhaustivité :

– le dessin dans son entier ou en partie est normalement représenté par du texte ; ceci supprime les problèmes liés à l'hétérogénéité de plusieurs parties d'un dessin ;
– des parties structurelles de dessin correspondent à des parties de texte ; par exemple, un rectangle sur un dessin pourra correspondre à une « commande rectangle » dans le texte ; cette propriété n'est pas vraie pour toutes les composantes d'un dessin ; par exemple, des droites peuvent s'intersecter sans que l'on ait demandé explicitement de construire cette intersection ;

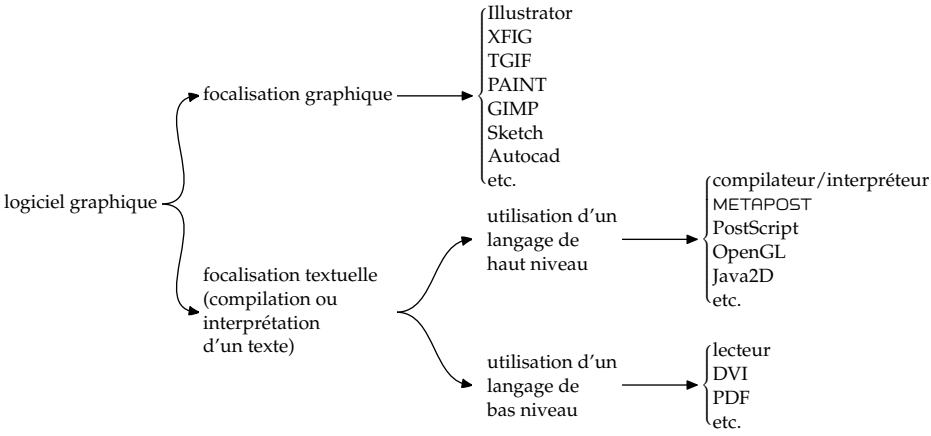

FIG. 3 – Classification schématique des outils graphiques, sans tenir compte de la nature du dessin (par exemple le fait que certains dessins soient vectoriels)

- le format textuel facilite l'échange d'information, en particulier avec d'autres outils textuels, y compris le courrier électronique ;
- la reproduction de certaines parties d'un dessin correspondra en général à du copier/coller ;
- un format textuel n'exclut pas un format graphique plus ou moins complémentaire, voire un format semi-graphique sur lequel on pourrait interagir ;
- le format textuel permet l'expression naturelle de relations formelles, car ces dernières utilisent habituellement le langage des mathématiques ;
- la précision des dessins, ainsi que leur dynamicité (c'est-à-dire les dépendances de certaines composantes par rapport à d'autres) peuvent être facilement contrôlées ;
- un format textuel de haut niveau bénéficie des avantages d'un langage de programmation, et en particulier de l'extensibilité et du paramétrage ; cela peut aussi être introduit graphiquement, mais c'est sans doute moins simple ;
- un format textuel de haut niveau est souvent plus facile à retenir et à expliquer ; il peut aussi être expliqué sans le recours au logiciel lui-même ;
- un format textuel facilite la génération automatique de dessins ;
- un format textuel permet une bonne interface avec un traitement de texte à focalisation textuelle, comme par exemple TEX.

Cette liste ne convaincra peut-être pas tout le monde, et il est vrai que ces points forts peuvent aussi se retrouver à un niveau interface, mais il faut re-

connaître que les interfaces habituelles ne fournissent pas la plupart des facilités mentionnées, ou plutôt n'incitent pas à les utiliser. Il nous semble que ce phénomène est à rapprocher de celui que l'on observe en comparant les traitements de texte à focalisation graphique (aussi dits WYSIWYG) et à focalisation textuelle. Les feuilles de style sont peu utilisées dans les traitements de texte à focalisation graphique, tout simplement parce que l'environnement graphique ne peut que dissuader leur emploi. La table suivante montre que ce qui est souhaitable (bien) est difficile (c.-à-d. non naturel, non intuitif) à réaliser avec des outils WYSIWYG. Au contraire, un outil non-WYSIWYG encourage un codage optimal, car c'est le codage que voit l'utilisateur. Ces conclusions rejoignent aussi celles de certains typographes [Taylor, 1997].

	WYSIWYG (balisage invisible et complexe)	NON-WYSIWYG (balisage visible et simple)
FACILE	tout ce qui correspond à des boutons (mal)	utilisation de commandes de haut niveau, styles (bien)
DIFFICILE	feuilles de style (bien)	utilisation de commandes de bas niveau (mal)

5. Les apports de METAPOST

METAPOST permet facilement, par sa programmabilité, d'exprimer des relations abstraites et de construire ainsi des « méta dessins », lesquels peuvent se décliner en plusieurs variantes, à l'instar des polices METAFONT, dont le source permet d'obtenir à la fois des caractères romains, italiques, gras, etc.

Un aspect fondamental de cette abstraction est le paramétrage, et celui-ci n'est pas automatique. Le concepteur de dessins doit réfléchir pour aller plus loin que la simple figure qu'il essaie de reproduire. Il doit imaginer ce qu'il aimerait y changer par la suite et doit s'efforcer de traduire sous une forme algorithmique les propriétés les plus importantes du dessin, car cette traduction en facilitera la mise à jour.

La programmabilité de METAPOST est telle qu'il est possible de définir des couches évoluées facilitant la création d'un type particulier de dessins. Nous avons par exemple récemment proposé des extensions 3D pour la géométrie dans l'espace [Roegel, 2001a], ainsi que le système `metaobj` [Roegel, 2001b], entièrement programmé en METAPOST, et qui permet de manipuler des « objets ». Un tel système permet en principe de créer un objet tel que ceux qui sont examinés dans la section suivante, et de le manipuler à sa guise. Toutefois, l'utilisateur est constamment confronté à un dilemme : il doit trouver un juste milieu entre d'une part une abstraction extrême, dont il n'aura peut-être

pas besoin, parce qu'il sait qu'il n'aura qu'un ou deux dessins d'un certain type à réaliser, et que l'abstraction est très coûteuse en temps ; et d'autre part, entre une abstraction nulle, qui risque de lui faire effectuer de nombreuses tâches répétitives ; c'est en fonction de chaque cas particulier que l'utilisateur doit évaluer jusqu'où le souci de perfection doit le mener.

6. Un exemple d'analyse

Il y a quelques années, une collègue m'avait montré un exemple des graphes avec lesquels elle travaillait. Le dessin était à peu près celui de la figure 4.

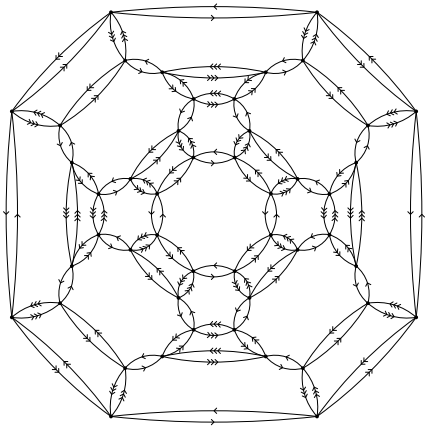

FIG. 4 – Le graphe complet (figure apparaissant dans [Strogova, 1996, p.6], mais créée par l'auteur)

Les graphes réalisés par ma collègue l'avaient été en utilisant des outils interactifs qui ne permettaient pas d'exprimer certaines propriétés géométriques ou de modifier facilement un dessin. La réalisation de ces graphes était donc inutilement longue et pénible.

Une approche algorithmique permet de simplifier considérablement la réalisation de ces graphes, en les rendant en particulier paramétrables. Un examen attentif du graphe montre qu'une première étape sera naturellement de déterminer les positions des sommets. Ensuite, ceux-ci peuvent-être regroupés et reliés simplement. Les flèches ont, elles-aussi, une organisation régulière.

Les points peuvent être regroupés en huit « rayons » et numérotés de telle sorte que cette symétrie apparaisse. C'est ce que nous avons fait dans la figure 5.

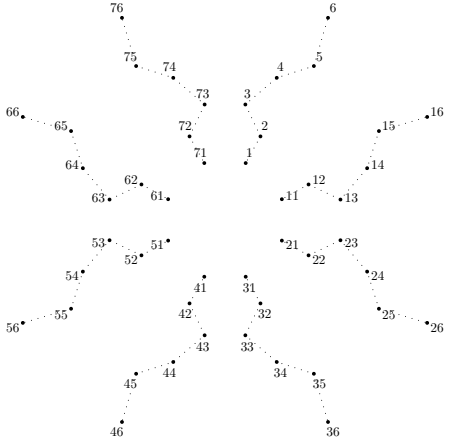

FIG. 5 – Numérotation des points du graphe

Il est facile de voir que les différents rayons se déduisent les uns des autres par symétrie. Les rayons 0 (points 1 à 6) et 7 (points 71 à 76) se déduisent par une symétrie autour d'un axe vertical, les rayons 0 (points 1 à 6) et 1 (points 11 à 16) se déduisent par une symétrie autour d'une droite à 45 degrés, etc.

Le positionnement de l'ensemble des points revient donc à fixer les positions des six points d'un rayon particulier.

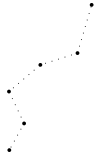

FIG. 6 – Le « rayon » 0 du graphe

Le rayon 0 (figure 6) peut être fixé par les commandes

```
numeric u;
u=1cm;
z0=origin;
```

```
z1-z0=polar(1.6u,70) ; z2-z0=polar(2.4u,67) ;
z3-z0=polar(3.1u,80) ; z4-z0=polar(4u,70) ;
z5-z0=polar(4.7u,60) ; z6-z0=polar(6u,63) ;
```

où `polar` est défini ainsi :

```
def polar(expr r,a) = r*(unitvector dir(a)) enddef;
```

Chacun des six points du rayon est donc déterminé par une distance et un angle. Il ne s'agit bien sûr pas de la seule manière de faire.

Les sept autres rayons sont obtenus très simplement. Une première symétrie détermine le rayon 1 :

```
for i:=1 upto 6:
  z[i+10]=z[i] reflectedabout(z0,(100u,100u));
endfor;
```

et les six autres rayons sont obtenus par des rotations :

```
for i:=1 upto 6:
  z[i+20]=z[i] rotatedaround(z0,-90);
  z[i+30]=z[i+10] rotatedaround(z0,-90);
  z[i+40]=z[i] rotatedaround(z0,180);
  z[i+50]=z[i+10] rotatedaround(z0,180);
  z[i+60]=z[i] rotatedaround(z0,90);
  z[i+70]=z[i+10] rotatedaround(z0,90);
endfor;
```

Il reste à connecter les points par des arcs doubles, comportant une, deux ou trois flèches. Les points étant numérotés de manière régulière, il est facile de voir que les connections peuvent se répartir en trois groupes : celles interne à un rayon, celles liant un rayon de numéro $2n+1 \pmod 8$ au rayon $2n+2 \pmod 8$ et celles liant un rayon de numéro $2n$ au rayon $2n+1 \pmod 8$. Par conséquent, étant donnée une commande `link` prenant trois arguments, le premier étant le nombre de flèches, le second le point de départ et le troisième le point d'arrivée des deux arcs, l'ensemble des connections peut être obtenu par :

```
for i:=0 upto 7:
  link(3,z[i*10+1],z[i*10+2]) ;
  link(1,z[i*10+2],z[i*10+3]) ;
  link(2,z[i*10+3],z[i*10+4]) ;
  link(1,z[i*10+4],z[i*10+5]) ;
```

```
  link(3,z[i*10+5],z[i*10+6]) ;
endfor;
for i:=0 step 2 until 6:
   link(2,z[i*10+1],z[(i+1)*10+1]) ;
   link(2,z[i*10+2],z[(i+1)*10+2]) ;
   link(2,z[i*10+5],z[(i+1)*10+5]) ;
   link(2,z[i*10+6],z[(i+1)*10+6]) ;
endfor;
for i:=1 step 2 until 7:
   link(1,z[i*10+1],z[((i+1) mod 8)*10+1]) ;
   link(3,z[i*10+3],z[((i+1) mod 8)*10+3]) ;
   link(3,z[i*10+4],z[((i+1) mod 8)*10+4]) ;
   link(1,z[i*10+6],z[((i+1) mod 8)*10+6]) ;
endfor;
```

La figure de départ était très symétrique ce qui nous a conduit à un codage très compact. Nous avons pu prendre en compte toutes les propriétés géométriques du dessin. Dans la réalité, les choses ne sont pas toujours aussi simples, mais on gagne presque toujours à abstraire, donc à comprendre le dessin. Dans notre exemple, nous pouvons facilement créer des variantes du graphe, en modifiant les points du rayon 0, et ces modifications se répercuteront automatiquement sur tous les autres rayons. Le nombre de rayons pourrait être lui-même modifié et les possibilités sont innombrables. Enfin, un tel graphe pourrait même être produit automatiquement.

7. Conclusion

Quoique aucun outil ne soit parfait, METAPOST présente des caractéristiques intéressantes, au moins pour un certain type d'applications. Outre l'avantage d'employer un format textuel, il est portable, ce qui n'est pas négligeable. Enfin, il est léger et permet d'exprimer sans lourdeurs des choses simples.

Pour des dessins uniques, ne devant pas s'harmoniser avec d'autres dessins ou avec leur environnement, la souris et un logiciel quelconque peuvent suffire. Mais si on veut faire une série de dessins, ou simplement paramétrer l'intégration du dessin dans son environnement, on peut peut-être aller plus loin avec METAPOST.

Il est important de ne pas voir ce logiciel ou le format METAPOST comme des fins en soi. Le format n'est pas un format fermé et isolé. Des évolutions de METAPOST pourront faire produire autre chose que du POSTSCRIPT, par exemple du PDF, du SVG, etc. D'autre part, certains programmes peuvent d'ores et déjà produire des sorties METAPOST. Il faut donc plutôt voir METAPOST comme un maillon d'une chaîne, et un maillon appelé à évoluer.

Certes, le logiciel a des limites. Ce n'est ainsi pas un outil très adapté pour du dessin non vectoriel, bien qu'il puisse inclure des bitmaps. On peut cependant imaginer des interfaces graphiques évoluées (comme LyX pour LaTeX), qui permettront d'intégrer METAPOST à d'autres outils qui eux gèrent mieux les aspects un peu étrangers à METAPOST. Toutefois, une telle intégration serait-elle vraiment une bonne chose ? Ne perdrait-on pas ainsi beaucoup des avantages actuels de METAPOST ?

Remerciements

L'auteur tient à remercier Jacques André et Damien Wyart pour leurs remarques judicieuses.

Bibliographie

[Hobby, 1992] John D. Hobby. A User's Manual for MetaPost. Rapport Technique 162, AT&T Bell Laboratories, Murray Hill, New Jersey, Avril 1992. `http://cm.bell-labs.com/who/hobby/MetaPost.html`. Voir la traduction française de Pierre Fournier dans ce *Cahiers GUTenberg*.

[Knuth, 1986] Donald E. Knuth. *The METAFONTbook*. Reading, MA : Addison-Wesley, 1986.

[Knuth, 1997] Donald E. Knuth. *The Art of Computer Programming, volumes 1, 2 and 3*. Addison-Wesley Publishing Company, 1997. Nouvelles éditions.

[Roegel, 2001a] Denis Roegel. La géométrie dans l'espace avec METAPOST. *Cahiers GUTenberg*, 39–40 : 107–138, 2001.

[Roegel, 2001b] Denis Roegel. *The METAOBJ tutorial and reference manual*, 2001. CTAN : `graphics/metapost/contrib/macros/metaobj`.

[Strogova, 1996] Polina Strogova. *Techniques de Réécriture pour le Traitement de Problème de Routage dans des Graphes de Cayley*. Thèse de doctorat d'université, Université Henri Poincaré — Nancy 1, 1996.

[Taylor, 1997] Conrad Taylor. Mais qu'est-ce qu'ont bien pu nous apporter les systèmes WYSIWIG ? *Cahiers GUTenberg*, 27 : 5–33, 1997.