Cahiers GUTenberg

VERS LA CONSTRUCTION DE MACROS DE MISE EN COULEUR POUR TEX

Christophe CÉRIN

Cahiers GUTenberg, nº 10-11 (1991), p. 197-207.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1991___10-11_197_0>

Vers la construction de macros de mise en couleur pour TEX

Christophe CÉRIN

Université de Paris Sud, LRI bat 490, 91405 Orsay, France
email : cerin lri.lri.fr

Résumé. Cet article présente une approche pas à pas de construction de macros de mise en couleur pour TEX.

Abstract. *Here is presented a step by step approach to putting colour in TEX documents.*

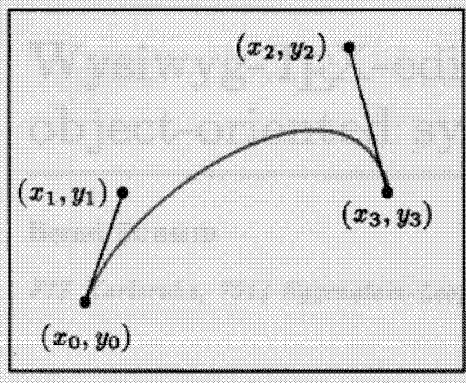
Mots clef : TEX, couleur, PostScript.

Depuis un peu plus d'une année, nous développons des jeux de macros permettant la gestion de la couleur dans le traitement de texte TEX– LATEX. Le premier jeu [Cerin et Lemaire 90] autorise le balisage de textes en couleur, le tracé de rectangles bordés de couleur et de réaliser différents dégradés sur des chaînes de caractères ...L'orsqu'il s'agit de changer de couleur, nous réalisons l'implantation au moyen d'opérateurs PostScript de gestion de la couleur si bien que les macros sont dépendantes du convertisseur utilisé, en l'occurence `dvips`.

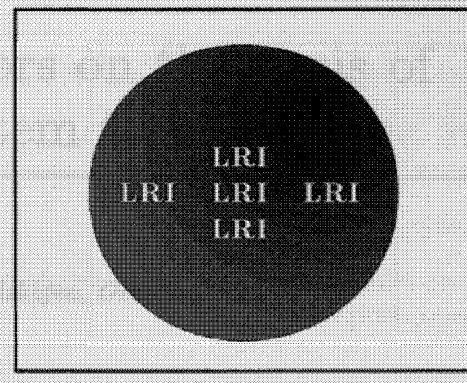
Le nouveau jeu proposé – version 2.3– permet de tracer des boites dont le fond est en couleur, des cercles concentriques dégradés, des rectangles dont le fond est constitué de dégradés horizontaux ou verticaux ainsi que de primitives graphiques notamment des courbes de Bézier en couleur ainsi que des cercles hachurés destinés à l'environnement LATEX `picture`.

Nous souhaitons, à cette occasion, commenter les règles générales de conception des macros couleur :

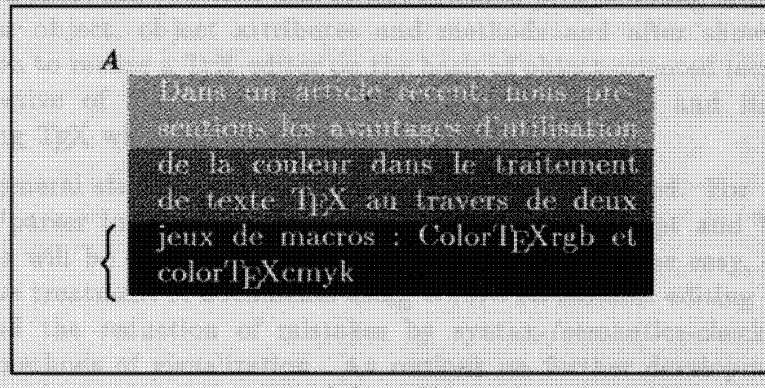
— Chaque fois que nous aurons besoin d'empiler ou de juxtaposer des boites (au sens TEX) contenant des objets graphiques ou des objets textes à colorer ; chaque fois que nous aurons besoin de connaître les dimensions de ces boites : utilisons TEX !



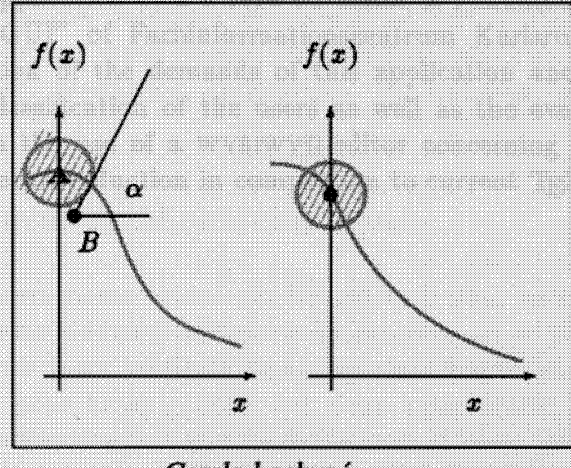
Courbe de Bézier



Fond dégradé



Fond dégradé



Cercle hachuré

— De même, chaque fois que nous devons dessiner un trait de couleur, positionner avec precision un texte, un cercle sur la page (notion de point courant) : opérons au moyen de PostScript.

Enfin, nous montrons qu'une discipline de spécification ainsi que la reprise de concepts de programmation de langages classiques conduisent à des implantations efficaces en temps et place disque.

1. Une macro couleur TEX sans notion de point courant.

Nous souhaitons une macro **\echelon** qui réalise un dégradé en couleur sur une chaîne de caractères. La première et la dernière lettre de la chaîne sont imprimées avec une valeur de couleur donnée en paramètre (couleur de début et couleur de fin). La difficulté principale consiste à répartir uniformément les couleurs.

Informellement, le problème s'ennonce de la manière suivante : mettre une chaîne de caractères en couleur consiste à imprimer le premier caractère dans la couleur de début, puis d'incrémenter la valeur de la couleur et de recommencer ce même traitement sur la chaîne initiale privée de son premier élément.

Un code intermédiaire plus proche d'une implantation réelle est : (cf [Cerin et Lemaire 90])

```
dec echelon:chaine → liste d'instructions
    echelon(c)← l;
      where l =
        if l = ∅ then ← ∅;
        else
          let l1 = (\begincolor{couleur} (car c) \endcolor) ;
          couleur ← couleur + increment ;
          if couleur > couleurmax then
            couleur ← couleurmin ;
          endif ;
          ← (cons l1 echelon (cdr c)) ;
        endif
    ;;
```

Le code TeX correspondant à la macro est commenté dans [Cerin et Lemaire 90]. Nous ne faisons appel à PostScript que pour changer la couleur des caractères. La définition et son aspect récursif sont directement implantés d'une manière récursive en TeX : nous suivons à la lettre la définition pour programmer !

```
{% debut de groupe
% rouge = bleu = vert = 20
% maxrouge = maxvert = maxbleu = 210
% chaque caractere change de couleur
\Large\bf
\initcolor{20}{20}{20}{210}{210}{210}{1}
% increrouge = increvert = increbleu = 40
\initinc{40}40{40}
\echelon{exemple}
}% fin du groupe
```

qui produit le résultat : exemple

2. Une macro couleur TEX presque totalement implantée en PostScript.

Pour tracer des courbes en couleur dans un environnement `picture`, nous proposons une implantation PostScript reposant sur l'opérateur `curveto` qui a besoin des quatre points $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$ définissant les tangentes de la courbe de la figure 1[1].

Puis, on effectue une interpolation polynomiale pour tracer la « meilleure » courbe reliant les points de coordonnées $(x_0, y_0), (x_3, y_3)$. La macro se nomme `\Bezierps` et admet comme paramètres une liste de points. Cette liste se termine, pour TEX, lorsque l'analyseur syntaxique rencontre la balise `\endbezierps` et elle constitue le premier paramètre de la macro (pour l'utilisation de macros avec arguments balisés cf [Seroul 89] page 201).

Le corps proprement dit repose directement sur une abstraction sur les listes disponible en langage PostScript – il s'agit de la primitive `forall` – qui correspond à la fonction Lisp `map`. Cette fonction applique une fonction donnée en paramètre à tous les éléments d'une liste de paramètres.

[1] Le lecteur se reportera à la planche 1 pour les figures en couleur. *ndlr.*

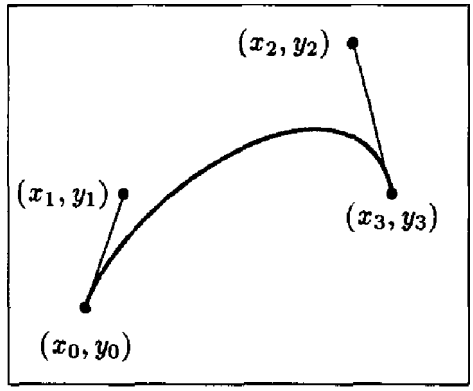


Figure 1 : *courbe de Bézier*

Dans notre cas, nous appliquons « curveto » (l'opérateur PostScript qui dessine une courbe de Bézier) à une liste finie de listes de 4 points permettant ainsi le tracé de plusieurs courbes en n'utilisant qu'une seule instruction **curveto**. Cette réalisation est plus concise qu'une écriture avec autant d'instructions **curveto** qu'il y a de courbes à dessiner. Le gain en place disque est évident dans le cas de dessins avec de nombreuses courbes de Bézier.

Le dessin de la figure 1 a été réalisé sur PC avec TEXCad l'équivalent de PicTEX sur station. Comme nous ne pouvons pas manipuler pour l'instant à la souris des courbes de Bézier sous TexCad, nous avons implicitement marqué les points $(x_0, y_0) \cdots (x_3, y_3)$ et récupéré les coordonées de ces points sous éditeur pour les passer comme arguments à la procédure **\Bezierps**. Nous obtenons le codage suivant :

```
\begin{figure}[th]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(60,50)
\put(0.00,0.00){\framebox(60,50)[t]{}}
% debut de la courbe de bezier
\put(0,0){\bezierps [ [ 10 10 15 25 45 45 50 25 ] ]
          \endbezierps{.4}{3}{2}{255}{0}{0}}
\put(10,5){\makebox(0,0)[cc]{$(x_0,y_0)$}}
```

```
\put(7,25){\makebox(0,0)[cc]{$(x_1,y_1)$}}
\put(36,45){\makebox(0,0)[cc]{$(x_2,y_2)$}}
\put(50,22){\makebox(0,0)[cc]{$(x_3,y_3)$}}
\put(10,10){\makebox(0,0)[cc]{$\bullet$}}
\put(15,25){\makebox(0,0)[cc]{$\bullet$}}
\put(45,45){\makebox(0,0)[cc]{$\bullet$}}
\put(50,25){\makebox(0,0)[cc]{$\bullet$}}
\put(10,10){\line(1,3){5}}
\put(45,45){\line(1,4){5}}
\end{picture}
\caption{{\it courbe de B\'ezier}}\label{figone}
\end{center} \end{figure}
```

3. Une macro couleur moitié implantée en TEX, moitié en PostScript.

Il s'agit d'écrire du texte sur un fond dégradé horizontalement ou verticalement. Le texte est contenu dans une boite TEX dont la hauteur et la largeur sont utilisées pour tracer le rectangle dégradé recherché. Il est alors nécessaire de récupérer (cf figure 2) les coordonnées relatives du point A par rapport au centre du rectangle et de tracer en PostScript les différentes bandes couleur à partir de ce point. L'implantation s'opère en utilisant l'ordre TEX **\special** pour l'introduction du texte PostScript. Enfin, nous avons aussi besoin de connaître le système de coordonnées (position des axes, facteur d'échelle ...) à l'intérieur d'un ordre **\special**. Après cette succession d'opérations, il n'est pas alors très difficile de déposer sur la feuille d'impression la boite contenant le texte qui vient s'inscrire au dessus du dégradé.

Le codage TEX est :

```
\setbox101=\vbox{\hsize=0.5\textwidth\bf\noindent%
\begincolor{255}{255}{255}%
Dans un article r\'ecent, nous pr\'esentions les avantages
d'utilisation de la couleur dans le traitement de texte
\TeX\ au  travers de deux jeux de macros~: Color\TeX rgb et
color\TeX cmyk \ec \par}
```

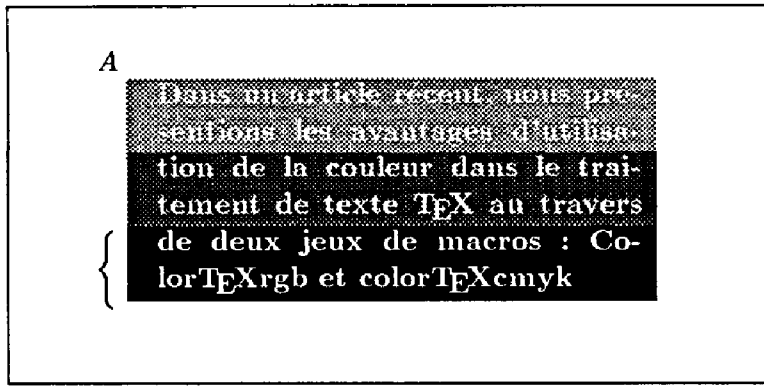


Figure 2 : *fond dégradé*

```
\horizontaltrue % degrade horizontal
\begin{figure}[th]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(100,50)
\put(0,0){\framebox(100,50)[t]{}}
\put(15,40){\makebox(0,0)[cc]{$A$}}
\put(15,15){\makebox(0,0)[cc]{
   \hbox{$\left\{\vbox to 7mm{}\right.$}}}
\put(50,25){\makebox(0,0)[cc]{
   \RectangleDegrade{\copy101}{200}{200}{0}{0}{0}{0}{3}}}
\end{picture}
\caption{{\it fond d\'egrad\'e}}\label{figtwo}
\end{center}
\end{figure}
```

4. Comment dessiner des cercles hachurés au moyen de « fenêtres » d'impression ?

La macro que nous présentons maintenant réalise le tracé d'un cercle hachuré en couleur dont la distance entre les hachures, l'angle ainsi que la forme des hachures (pointillés ou traits pleins) est paramétrable.

Nous utilisons le mécanisme PostScript du « clipping » qui consiste à définir un contour fermé où seuls les objets à l'intérieur du contour sont visibles.

Ainsi, pour réaliser la macro décrite ci-dessus, nous procédons de la manière suivante (cf figure 3) :

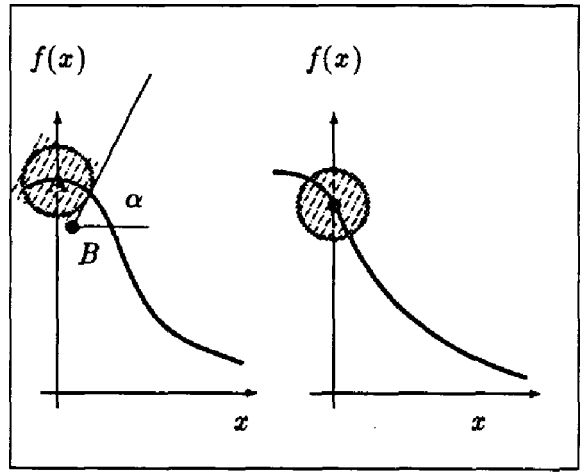


Figure 3 : *cercle hachuré*

1. Le déplacement vers le point B s'effectue à partir du point A au moyen d'un calcul faisant intervenir les cosinus et sinus de l'angle α ainsi que le rayon du cercle. Nous tracons les lignes nécessaires après un changement de repère ;

2. Nous nous placons au point A et nous tracons le cercle de rayon demandé en considérant ce cercle comme la fenêtre d'impression (les « angles » du carré disparaissent). Nous obtenons le résultat de la figure 3 ;

Pour terminer, nous remarquons que nous avons tracé le plus petit carré circonscrit dans le cercle et que nous employons la facilité du « clipping » de PostScript. Le tracé de toutes les hachures à leur dimension réelle demande au programmeur un calcul beaucoup trop dur !

5. Récupération, traitements à partir du point courant.

L'ordre TEX `\special` permet l'inclusion de textes PostScript. La syntaxe `\put(0,0){\special{"` pour le translateur `dvips` nous positionne dans le coin inférieur gauche de la figure 4. Ce point est alors le point courant à partir duquel on travaille. Si l'unité courante souhaitée est le millimètre, il nous faut maintenant instaurer cette unité d'échelle (nous travaillons en « points » à l'intérieur du `\special`) :

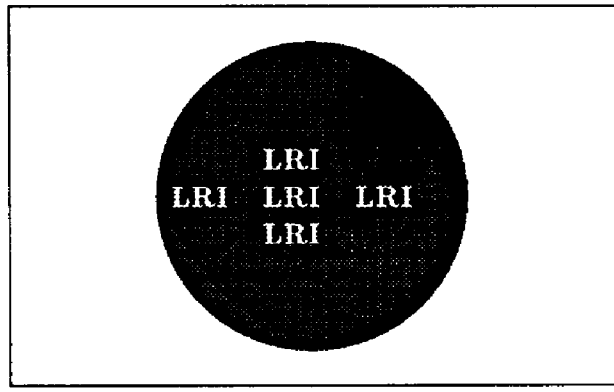


Figure 4 : *fond dégradé*

```
% la constante 2.85 = 72 / 25.6
%    ==> pour travailler en mm
\special{" gsave 2.85 2.85 scale
.
% votre code -- unite en mm
.
grestore}
```

Vous pouvez maintenant examiner plus en détail le code TEX de la macro correspondant au dessin de la figure 4. Le tracé du cercle dégradé s'effectue au moyen de l'opérateur PostScript `clip` en décomposant le problème de la manière suivante :

1. La boite TEX 1 contient le texte. Soit $r = \max\{\backslash ht1, \backslash wd1\}$; soit $xtrans, ytrans$ les variables donnant les valeurs de l'excentration du cercle ; soit $A = (x, y)$ le point du centre du cercle ;

2. On se déplace au point $A' = (x + xtrans, y + ytrans)$ et on calcule $r' = r + \sqrt{xtrans^2 + ytrans^2}$;

3. On trace ensuite un cercle de rayon r' à partir de A' en considérant le cercle de rayon r et de centre A comme contour de « clipping » . On itère enfin le procédé de tracé de cercles de plus en plus petits à partir du point A' ;

```
\setbox100=\vbox{\bf\begincolor{255}{255}{255}
\halign{\quad#&\quad#&\quad#\cr
& LRI& \cr
LRI& LRI& LRI\cr
& LRI& \cr}
\endcolor}
% rouge=(255,0,0)  noir=(0,0,0)   nb-nuances=10
\xtrans=30pt% on decale le centre de 50pt vers la droite
\ytrans=30pt% et 50pt vers le haut
\begin{figure}[th]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(80,50)
\put(0,0){\framebox(80,50)[t]{}}
\put(40,25){\makebox(0,0)[cc]{% il y a 10 nuances
\CercleGeneral{\copy100}{255}{0}{0}{0}{0}{0}{10}}}
\end{picture}
\caption{{\it fond d\'egrad\'e}}\label{figfour}
\end{center}
\end{figure}
```

6. Conclusion

Une programmation par décomposition en sous tâches ainsi que l'utilisation d'opérateurs PostScript qui offrent un important niveau d'abstraction constituent le socle de la démarche de mise en couleur.

Cette façon de faire conduit à une programmation en TEX ou en PostScript structurée. Elle permet aussi de découvrir quelques zones d'ombres dans la définition des opérateurs PostScript. Par exemple, on

imagine assez bien ce qu'est le contour d'un caractère vectoriel – la collection successive des courbes le définissant – mais qu'en est-il d'un caractère « bitmap » ? Autrement dit, peut-on récupérer en PostScript le contour d'un caractère bitmap ? Par quel moyen ?

Nous avons aussi commenté les opérations nécessaires pour réaliser l'interface TEX–PostScript au moyen de la commande `\special`.

Le jeu de macros ainsi qu'un guide complet d'utilisation est disponible auprès de l'auteur.

Références bibliographiques

[Adobe 86] Adobe Systems Incorporated, *PostScript Language Reference Manual*, Addison-Wesley, october 1986.

[Adobe 87] Adobe Systems Incorporated, *PostScript Language Tutorial and Cookbook*, Addison-Wesley, march 1987.

[Adobe 88] Adobe Systems Incorporated, *PostScript Language Program Design*, Addison-Wesley, february 1988.

[Cerin et Lemaire 90] Christophe CÉRIN et Benoît LEMAIRE, "Vers l'introduction de la couleur dans TEX", in *Cahiers GUTenberg*, no 5, pages 8-15, mai 1990.

[Knuth 86] Donald E. KNUTH, *The TEXbook*, Addison-Wesley, 1986.

[Lamport 86] Leslie LAMPORT, LATEX *user's guide & reference Manual*, Addison-Wesley, 1986.

[Seroul 89] Raymond SEROUL, *Le petit livre de TEX*, InterEdition, Paris, mai 1989.