Petite introduction au langage PostScript

Jean-Paul Davalan

1999

1 Un Aperçu du Langage PostScript

Ce texte n'est pas un manuel de PostScript. L'intention est de présenter quelques notions élémentaires suffisantes à la réalisation des programmes des 17 pavages réguliers du plan.
Pour une meilleure connaissance du PostScript on pourra se référer aux liens indiqués.

1.1 Quelques notions de PostScript

1.1.1 Le langage et la notation polonaise inverse

Un langage pour voir et imprimer des documents, textes ou images. Le langage PostScript permet à une imprimante le comprenant, d'imprimer des documents textes ou graphiques. PostScript est indépendant de l'imprimante ou de tout autre matériel qui l'utilisera, d'ailleurs un même document sera susceptible d'être traité en des lieux et à des moments différents, par des matériels divers. Un fichier PS n'est pas préparé pour une imprimante particulière et il est même préférable de ne pas donner d'ordres spécifiques à une imprimante particulière, ni même, comme c'est souvent le cas, d'indication de résolution (300 dpi) pour laisser à l'utilisateur le choix de ce qui lui convient le mieux.
Les textes et dessins à réaliser sont découpés et décrits page après page, PostScript est un langage dit de description de page. PostScript intervient dans la chaîne de réalisation de ce document, les fichier de suffixes *.ps* et *.eps* sont des fichiers PS.
La figure *(Fig. 1)* présente quelques logiciels utiles et les fichiers obtenus.
Les suffixes des fichiers sont donnés dans la table *(Tab. 1)*.

On peut utiliser le logiciel Ghostscript (gs) et ses drivers pour convertir un fichier PostScript en un autre format de fichier ou de langage, pour préparer l'impression sur une imprimante non PostScript ou pour réaliser une image à mettre sur une page Html . . .

Ghostscript permet de voir, convertir et d'imprimer les fichiers aux formats PostScript et PDF. On peut donc utiliser toute la puissance du PostScript, en particulier dans ce qui nous intéresse ici : le dessin géométrique.

Ghostscript est disponible gratuitement, il suffit de le télécharger. Une adaptation à Windows 95 NT . . .existe.
De nombreux documents PostScript sont disponibles sur le WEB, Ghostscript (gs), permet de les lire et de les imprimer.
Pour voir directement, à l'écran, sous XWindows, les fichiers PostScript, il suffit d'écrire par exemple :

```
monordi:~/jp/doc# gs monfichier.ps
```

On peut préférer utiliser Ghostview qui offre un menu permettant des changements de formats de pages, des agrandissements, de changer de fichier.
Ghostview est disponible sous Windows.

Le logiciel Mgv offre une meilleure qualité d'image mais est un peu plus lent dans le tracé des images. Il permet de lire directement les fichiers PostScript compressés par Gzip (extension de fichiers : .gz), les fichiers textes au format .ps étant relativement volumineux (1 Mbytes est courant), on a avantage à les garder compressés

```
monordi:~/jp/doc# gzip monfichier.ps
```

S'il le faut, on les décompresse en faisant

```
monordi:~/jp/doc# gunzip monfichier.ps.gz
```

Un autre logiciel Gv est rapide et offre des menus complets. Pour une bonne qualité d'image, choisir l'option antialias dans le menu *State*.

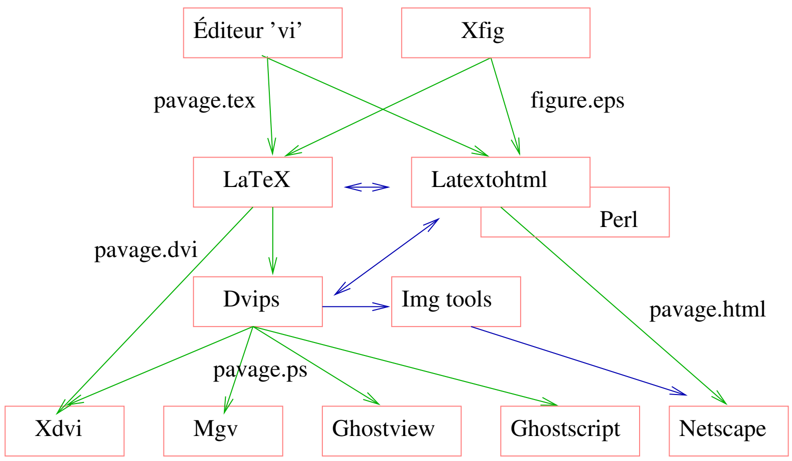

FIG. 1 – *Réalisation du document.*

suffixes	format
.tex	TeX ou LaTeX
.fig	Xfig
.ps	PostScript
.eps	Encapsulated PostScript
.html	HyperText Markup Language

TAB. 1 – *Suffixes de noms de fichiers.*

Ghostview, Mgv et Gv nécessitent l'installation préalable de Ghostscript.

Ghostscript est un produit de Aladdin Enterprises.
Adobe, Acrobat, PostScript sont des trademarks de Adobe Systems.

Un passage obligé pour l'impression. Sous Linux, lorsqu'on écrit ses documents aux formats TEX ou LATEX, ceux-ci sont d'abord transformés en un fichier au format DVI puis enfin en PS :

```
latex fichier.tex
dvips -o fichier.ps fichier.dvi
```

Les fichiers DVI produits par TEX etLATEX peuvent déjà être vus à l'aide de Xdvi, avant leur conversion en PS.

Un langage interprété. PostScript est réellement un langage de programmation, c'est un langage interprété, ce qui veut dire que les instructions sont exécutées, dès leur lecture, au fur et à mesure. Si une instruction doit être exécutée plusieurs fois, elle sera relue et déchiffrée chaque fois.

Un langage pour créer des images, des dessins géométriques. Notre objectif est de réaliser des pavages du plan en Postscript, directement, sans l'aide d'un quelconque logiciel, tout simplement car ce sera simple et efficace.

Lorqu'un dessin comporte des parties semblables, en grand nombre, l'écriture d'un programme de réalisation est nécessaire, soit en utilisant un logiciel spécialisé, soit en ayant recours à un langage général mais puissant comme PostScript.
Pour d'autres figures géométriques simples, il sera souvent plus rapide d'utiliser un logiciel comme Xfig.

La notation polonaise inverse. PostScript (en abrégé PS) utilise la notation polonaise inverse, voici un exemple dans la notation polonaise inverse du PS et le passage à l'écriture avec parenthèses correspondante est donné à la table *(Tab. 2)*. Le détail du calcul effectué est donné à la figure *(Fig. 3)*.

```
  .5      3    4       -5 mul add add 2 mul 7 div
  .5      3   (4 x  (-5))    add add 2 mul 7 div
  .5    ( 3 + (4 x  (-5)))       add 2 mul 7 div
 (.5 + ( 3 + (4 x  (-5))))           2 mul 7 div
((.5 + ( 3 + (4 x  (-5)))) x 2 )          7 div
(((.5 + ( 3 + ( 4 x (-5)))) x 2) /         7)
```

TAB. 2 – *Passage de la notation polonaise inverse à la notation parenthésée*

```
                pile     en attente
.5     3    4     -5     mul          add add 2 mul 7 div
      .5    3    -20     add          add 2 mul 7 div
           .5    -17     add          2 mul 7 div
                 -12       2          mul 7 div
           -12     2     mul          7 div
                 -24       7          div
           -24     7     mul
                -168
```

TAB. 3 – *Détail d'un calcul en PS*

Le travail de l'interpréteur est simplifié, en effet les données sont empilées au fur et à mesure de leur lecture, lorsqu'un opérateur arrive, il opère sur une, deux, trois données ou plus, les dernières arrivées étant les premières utilisées, ces données sont alors effacées et le résultat obtenu est empilé. Ainsi `.5 3 4 -5` est l'état de la pile lorsqu'arrive l'opérateur *mul* de multiplication, le produit `4 (-5) = -20` est effectué et empilé, les données utilisées pour ce calcul sont effacées, la pile est alors `.5 3 -20`. Les opérateurs *add* et *div* indiquent l'addition et la division.

En écriture polonaise directe (par opposition à inverse) les opérateurs sont écrits avant les opérandes. En écriture polonaise le même calcul s'écrirait

```
D  M  A  .5  A  3  M  4  -5  2  7
```

où la signification des opérateurs A, M et D d'addition, de multiplication et de division est évidente.

Le passage de l'écriture parenthésée à l'écriture polonaise est immédiate, son explication est simplifiée lorsque toutes les parenthèses sont écrites car ensuite on n'a plus à tenir compte des précédences des opérateurs.

```
(3 -4 + 5) (7 + 2) \ 3
( ( ( 3 + -4 )  + 5 )   x ( ( 7 + 2 ) / 3 ) )
( [ ( 3 + -4 )  + 5 ]   x ( ( 7 + 2 ) / 3 ) )
(  ( 3  + -4 )    5 add x ( ( 7 + 2 ) / 3 ) )
(    3    -4 add  5 add x ( ( 7 + 2 ) / 3 )    )
     3    -4 add  5 add   ( ( 7 + 2 ) / 3 )    mul
     3    -4 add  5 add     ( 7 + 2 )   3 div  mul
     3    -4 add  5 add       7   2 add 3 div  mul
```

TAB. 4 – *Passage des parenthèses à la notation polonaise inverse*

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 100 80
/gsave
0 80 moveto 100 0 lineto 0 1 0 setrgbcolor 1.0 setlinewidth stroke
40 20 moveto 90 70 lineto 1 0 0 setrgbcolor 1.2 setlinewidth stroke
/showpage
/grestore
```

TAB. 5 – *Premier programme PS.*

Le procédé est d'effacer deux parenthèses associées et de transférer l'opérateur correspondant à l'emplacement de la parenthèse gauche ou droite selon que la notation est directe ou inverse. Un exemple de conversion est donné à la table *Tab.(4)*. L'ordre des remplacements des parenthèses est indifférent, il peut correspondre ou non à l'ordre que l'on choisirait pour le calcul de l'expression. Pour une expression donnée avec parenthèses, dans tous les cas l'écriture obtenue sera la même.
La notation polonaise, inverse ou non, permet de s'affranchir de l'usage des parenthèses, les étapes du calcul restent parfaitement définies, on sait toujours sur quels objets et dans quel ordre, opérer.

placer en $(10; 20)$	:	10 20 moveto
relier à $(40; 70)$	:	40 70 lineto
colorier en rouge	:	1 0 0 setrgbcolor
épaisseur 1.2	:	1.2 setlinewidth
encrer	:	stroke

TAB. 6 – *Trait en PS.*

Dans l'exemple géométrique de la table *(Tab. 5)* on a la séquence

```
0 80 moveto 100 0 lineto 0 1 0 setrgbcolor 1.0 setlinewidth stroke
```

qui permet de placer la pointe du crayon au point de coordonnées $(0; 80)$, de tracer un trait jusqu'en $(100; 0)$, de choisir la couleur rouge, l'épaisseur du trait et enfin, avec *stroke* d'effectuer définitivement l'encrage du trait, (mais on pourra toujours le recouvrir ensuite, si on le désire). Ne pas oublier que l'écriture est polonaise inverse et l'interpréter de gauche à droite en empilant et en dépilant les données. L'explication du tracé du second trait est donnée à la table *(Tab. 6)*, la figure obtenue est *(Fig.2)*.

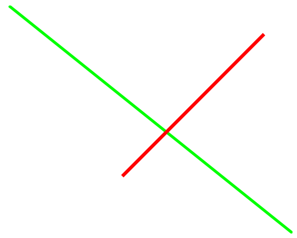

FIG. 2 – *Les deux traits.*

base 2 (binaire)	:	0, 1
base 3	:	0, 1, 2
base 8 (octal)	:	0, 1, . . .6, 7
base 10 (décimal)	:	0, 1, . . .9
base 12	:	0, 1, . . .9, A, B
base 16 (hexadécimal)	:	0, 1, . . .9, A, B, C, D, E, F
base 24	:	0, 1, . . .9, A, B, . . .M, N

TAB. 7 – *Chiffres dans les bases 2, 3, 8, 10, 12, 16 et 24.*

1.1.2 Ojets simples

Nombres Ce sont les nombres entiers ou les décimaux, dans les intervalles :

- $[-2^{31} - 1;\ 2^{31} - 1]$ pour les entiers, ($2^{31} - 1 = 2\,147\,483\,647$),
- $[10^{-38};\ 10^{38}]$ environ pour les décimaux positifs, non nuls.

Entiers Les entiers s'écrivent dans le système décimal (1999, -1234 par exemple) ou dans tout système de base b, $2 \leq b \leq 36$, les chiffres successifs étant alors 0, 1, 2, 3,. . .,8, 9, A, B, C, D, . . ., V, W, X, Y, Z, en se limitant aux b premiers chiffres :

L'écriture en base b est la suivante : `xy#mn...p`, la première partie `xy` indique la base, la deuxième `mn...p` est l'écriture dans cette base, des exemples sont donnés à la table *(Tab. 8)*.

En décimal les négatifs s'écrivent en plaçant le signe $-$ à gauche du nombre. Dans les autres bases le premier bit est 1 et la représentation est celle du complément à 2, il est nécessaire de bien connaître la

Base	Écriture PS
2	`2#11001111010000000110010001`
3	`3#10210020020101210`
8	`8#317200621`
10	`54329745`
12	`12#162409A9`
16	`16#33D0191`
24	`24#6JI2B9`

TAB. 8 – *Écritures en bases 2, 8, 10, 16 et 24.*

caractères	description
`{ }`	encadrent une procédure
`( )`	encadrent une chaîne
`[ ]`	encadrent un tableau
`<>`	encadrent une chaîne en écriture hexadécimale
`%`	débute un commentaire
`/`	amène un littéral
`\`	premier caractère d'une séquence d'échappement

TAB. 9 – *Caractères spéciaux.*

signification de ces termes pour les utiliser, on peut aussi prendre l'opposé d'un nombre positif en utilisant l'opérateur *neg* et en écrivant par exemple : 135 neg

Réels Les réels sont en réalité écrits par leur approximation décimale, d'une façon habituelle, par exemple :
-3.45E-5 pour $-3,45 \times 10^{-5}$, les lettres majuscule « E » ou minuscule « e » indiquent la puissance de 10 par laquelle on multiplie le décimal placé à gauche et obtenir la valeur désirée. Une écriture dite *scientifique* est d'ailleurs de la forme $x \times 10^n$ où $-1 \leq x \leq 1$.
Dans les écritures des nombres, il faut garder l'ensemble des chiffres groupé et ne pas insérer d'espace, ainsi `-3.45 E -5` serait incorrect.

Booléens Habituellement notés 0 et 1 dans d'autres langages, en PostScript ils sont écrits **false** et **true** (faux et vrai) et ne sont pas assimilables aux valeurs entières 0 et 1.
false et **true** sont utilisés avec les opérateurs logiques comme **and**, **or**, **not** …Ce sont aussi les résultats de comparaisons comme par exemple `a b lt` où `a` et `b` sont deux variables, réelles ou entières, et `lt` l'opérateur $<$. La comparaison est donc $(a < b)$ qui prend la valeur **true** lorsque a est plus petit que b, la valeur **false** dans le cas contraire.

Caractères Peuvent être classés en plusieurs catégories suivant le rôle qu'ils remplissent (et donc de leur emplacement dans le programme PS).

- **programmation.** Seuls les caractères dont le code ASCII est compris entre 32 et 127 sont admis.
- **commentaire.** Tous les caractères accessibles par le clavier. (les commentaires débutent par le signe % et se terminent en fin de ligne).
- **impression.** Sont ceux de la police en cours et sont au nombre de 224 (maximum).
- **spéciaux d'encadrement …** Voir ci-après.
- **échappement.** Sont décrits au paragraphe suivant.

Chaînes Ce sont les chaînes de caractères.

Les constantes S'écrivent de deux manières.

- Entre parenthèses :
  ```
  (Une ligne de PostScript.}
  ```
- En code hexadécimal entre les deux signes `<>`.
  ```
  <556E206C69676E6520646520506F73745363726970742E>
  ```

séquence	nom	effet
`\n`	newline	changement de line
`\r`	carriage return	retour à la ligne
`\t`	horizontal tab	tabulation horizontale
`\f`	form feed	
`\\`	backslash	
`\(`		parenthèse ouvrante
`\)`		parenthèse fermante
`\xyz`		le code octal d'un caractère

TAB. 10 – *Séquences d'échappement.*

Les séquences d'échappement sont :
Une chaîne constante est créée par les parenthèses ou les <> qui la délimitent.

Les chaînes de longueur variable Une chaîne variable est créée par :
- **string** `/ch n string def`
- **def** `/ch (...) def`

où `ch` est le nom de la chaîne créée.
L'opérateur **length** permet d'obtenir la longueur :

```
/ch(abcdef\tghij)def
ch length
```

devrait mettre en haut de la pile le nombre de caractères qui est de 11.

Les indices des caractères vont de 0 à $n - 1$ si la longueur est n.

Une copie de chaîne s'obtient par :
- **copy** `/ch1 ch2 copy` qui copie `/ch1` dans `/ch2`
- `(...) ch2 copy` qui copie `(...)` dans `/ch2`
- **def** `/ch (...) def` qui copie `(...)` dans `/ch`

Variables Sont définies de plusieurs façons.

def Par exemple :

```
/a 4 def /b 5 def /c 3.141592 def
```

permet de créer trois variables a, b, c et de leur affecter des valeurs, dans d'autres langages ce serait $a = 4$, $b = 5$, $c = 3.141592$.

```
/dblea a 2 mul def
```

crée une variable $dblea$ qui est double de a, $dblea = a \times 2$. Une modification de la valeur de a entraîne aussi un changement de valeur de $dblea$, il n'y a pas affectation immédiate, la variable $dblea$ n'est évaluée qu'au moment où sa valeur est réclamée dans un calcul.

```
%%% -------------- DEFINITIONS GENERALES -----------------------
/PavDict 50 dict def PavDict begin   % C'est un commentaire
/bd{bind def}bind def/ed{exch def}bd/ld{load def}bd
/cl/setrgbcolor ld/tr/translate ld/ro/rotate ld
/gs/gsave ld/gr/grestore ld         % C'est un autre commentaire
/M/moveto ld/L/lineto ld/rM/rmoveto ld/rmt/rmoveto ld
/PL{{lineto}repeat}bd
/PLM{/pln ed/pmy ed/pmx ed pmx pmy M pln 1 sub PL pmx pmy L}bd
/st/stroke ld/np/newpath ld/cp/closepath ld/lj/setlinejoin ld/
lc/setlinecap ld/ml/setmiterlimit ld/sl/setlinewidth ld
/sc/scale ld
end
```

TAB. 11 – *Exemple d'en-tête de programme en PS.*

bind def En utilisant cet opérateur, les variations ultérieures de la variable a n'affectera pas $dblea$ qui n'est évaluée qu'une fois, lors de sa création :

```
/dblea{ a 2 mul} bind def
```

Un léger gain de temps à l'exécution résulte de cette seconde façon de créer une variable. Suivant le contexte du programme on choisira l'une ou l'autre.

load def L'opérateur **load** recherche un nom dans les dictionnaires et le place sur la pile, ainsi

```
/st{stroke} load def
```

remplace *st* par *stroke* lorsqu'il rencontre ce mot dans le programme.

Dans ce cas particulier le gain escompté est le temps que l'on gagne en écrivant le programme, tout le préambule d'un fichier PS étant généralement copié d'un précédent programme, cette partie n'occasionne aucune perte de temps dans sa rédaction.
La lisibilité peut aussi être meilleure lorsqu'on est habitué à utiliser les mêmes abréviations.

Le plus souvent les fichiers sont produits par des applications, pour être lus par l'imprimante, non par une personne, le gain escompté est alors dans la rapidité de la transmission, de la lecture et de l'interprétation par l'imprimante.

À titre d'exemple, les définitions générales présentes dans plusieurs fichiers PS de réalisation de pavages sont données à la table *(Tab. 11)*.

En PS les commentaires débutent par un signe % qui n'a d'effet que jusqu'à la fin de sa ligne. La partie de programme ci-dessus comporte donc une ligne complète de commentaire et deux fins de lignes, introduites comme exemples uniquement.

On pourra remarquer que *bind def* et *load def* sont eux-mêmes simplifiés en *bd* et en *ld* et aussitôt utilisés dans cette nouvelle écriture :

```
/bd{bind def}bind def
/ld{load def}bd
```

```
/fig{
        gsave
        0 0 0 setrgbcolor
        1 setlinewidth
        8 {
            gsave
            0 0 moveto 0 20 lineto stroke
            grestore
            10 0 translate
          } repeat
        grestore
    }def
```

TAB. 12 – *Procédure en PS.*

Piles Il y a plusieurs piles :

- La pile des opérandes ou pile opérationnelle où s'accumulent les arguments avant d'être utilisés par les opérateurs, toutes les données autres que les opérateurs s'y accumulent.
- La pile des dictionnaires.

Dictionnaires Un dictionnaire regroupe les couples (nom, valeur), où le nom est celui d'une variable ou d'un opérateur et où la valeur est le code associé.
L'interpréteur PostScript va y rechercher le code associé à une variable, lorsqu'il en a besoin.
Systemdict est le dictionnaire contenant le code des opérateurs PS.
Userdict existe par défaut et sert de dictionnaire courant, sa taille est de 200 entrées au plus.
Ces deux dictionnaires ne peuvent être détruits.
Si on crée un dictionnaire, il vient ensuite et sert alors de dictionnaire courant.
Exemple :

```
50 dict
```

réserve de la place pour 50 entrées,

```
begin
... (ici du code )
end
```

L'instruction **begin** ouvre ce dictionnaire pour en faire le dictionnaire courant, alors que **end** le referme lorsqu'il n'est plus utilisé.
On pourra retrouver cette utilisation des dictionnaires dans les programmes PS de pavages.

Opérateurs Un cerain nombre des opérateurs disponibles sont répertoriés dans les tables **??**, **??**, 24,**??**. Quelques indications, sous forme abrégée, de leurs effets sont indiquées dans ces tables.

Procédures Correspondent aux fonctions d'autres langages, voir en exemple la table *(Tab. 12)*.

Dans cet exemple la procédure fig permettra, à chaque appel, de tracer 8 traits parallèle,s il suffira d'écrire `fig` pour l'invoquer.
D'autres procédures ont besoin d'arguments qu'il faut alors placer sur la pile opérationnelle, elles peuvent

```
%!PS
/mondessin{      % procédure
   ...           % de mon  dessin
}def             %

gsave
...              % dessine le cadre
mondessin        % mon dessin 1

100 100 translate
15 rotate
.75 .75  scale

mondessin        % mon dessin 2 (différent de mon dessin 1)
showpage
grestore
```

TAB. 13 – *Recentrage et replacement.*

aussi retourner des valeurs en les plaçant sur la pile.

1.1.3 Objets graphiques

Le repère La page graphique ou espace utilisateur, correspond au premier quadrant du repère cartésien habituel, l'origine est placée en bas à gauche, le premier axe est l'axe horizontal, dirigé vers la droite, le deuxième axe est dirigé vers le haut. Les unités sur les axes sont le pica, 72 pica $= 1$ pouce $= 2.54$ cm donc 1 pica ≈ 0.353 mm et 1 cm $\approx$ 28.34646 pica. Le pica est le **point** typographique.
Une largeur de 21 cm d'une page au format A4 correspond donc à 595 points et la longueur de 29.7 cm à 842 points, les valeurs utiles d'une page sont légèrement moindre.
Les coordonnées sont des nombres réels (en principe, car il y a une limitation dans leur écriture).

Pour changer d'unité du repère, on peut écrire en début de programme

```
28.34646 28.34646 scale
```

et l'unité est alors le cm. Pour que ce soit le mm on écrirait `2.834646 2.834646 scale` ou mieux `2.834646 dup scale`, et aussi bien `72 25.4 div dup scale` qui est équivalent.

Pour placer l'origine au centre de la page, l'unité étant le point et les dimensions de 560 sur 840 environ, on peut écrire `280 420 translate`.
Plus exactement, toute figure déja tracée reste identique à elle même et tout dessin à venir sera décalé sur la page

Une rotation de 15 degrés se ferait en écrivant `15 rotate`. Voir la table *(Tab. 13)*. Si les deux dessins de *(Tab. 13)* sont effectués séparément on obtient les deux schémas de *(Fig. 3)*, lorsqu'ils sont effectués sur la même page on obtient *(Fig. 4)*.

L'espace utilisateur est converti en espace matériel par le logiciel et on n'a pas à se préoccuper des caractéristiques de cet autre espace.

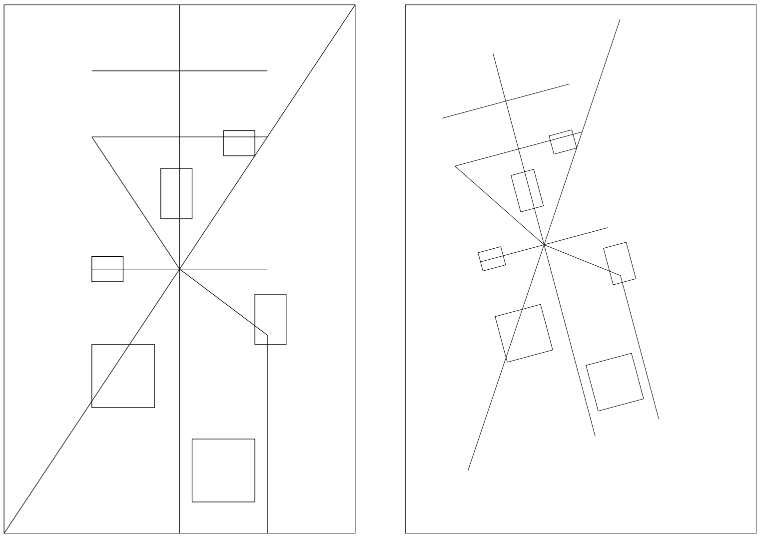

FIG. 3 – *Translation, rotation homothétie*

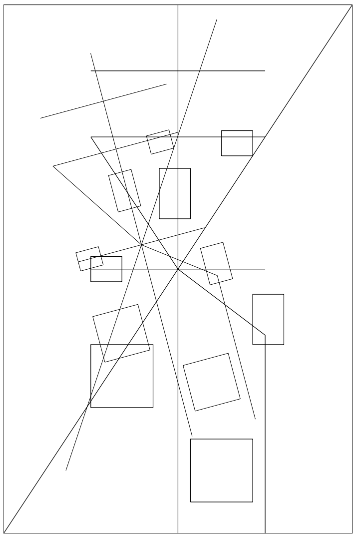

FIG. 4 – *Superposition des deux dessins.*

La CTM (Current Transformation Matrix) est une matrice qui contient les données relatives aux rotation, changement d'échelle et translation à effectuer pour passer de l'espace utilisateur à l'espace matériel.

L'état graphique Les paramètres suivants, tant qu'ils ne sont pas modifiés, s'appliquent à tout tracé ou coloriage, ce sont les paramètres courants :
position, largeur de trait, forme des pointillés, forme des fins de ligne, forme des angles de lignes, couleur (ou niveau de gris), police, taille des caractères (le corps), pochoir, matrice CTM.
Quand on se réfère à un chemin, il s'agit du chemin courant.
Pour sauvegarder un état graphique afin de le récupérer plus tard, on utilise les opérateurs `gsave` et `grestore`, comme à la table *(Tab. 14)*.
Les transformations n'ont pour champ d'application que les dessins effectués après elles et avant le premier `grestore` qui suit, (ou la fin du fichier s'il n'y a pas de `grestore`).

les modifications de l'origine, de l'échelle et de l'orientation n'ont d'effet qu'avant le prochain `grestore`

```
gsave
  100 100 translate
  15 rotate
  0.75 0.75 scale
  ...         % un dessin quelconque décalé
              % tourné de 15 degrés et diminué aux 3/4
              % de sa taille.
grestore      % la suite, elle, ne sera pas concernée.
```

TAB. 14 – *Opérateurs gsave et grestore.*

et on retrouve à partir de cet endroit l'état précédent qui avait été sauvegardé par `gsave`.

Lignes et régions

1.1.4 Impression de la page

2 Annexes

2.1 Tables des opérateurs

op.	la pile avant, l'opérateur	la pile après	
abs	n abs	a	val. absolue
add	num1 num2 add	num3	addition
aload	tab aload	a0…an-1 tab	
anchorsearch	ch sch anchorsearch	ch2 ch1 true / ch false	
and	bool1 bool2 and	bool	et logique
arc	x y r ang1 ang2 arc		dessine un arc de cercle
arcn	x y a1 a2 angn		
arcto	x1…y2 r arcto	xt1…yt2	coord pts de contact
array	n array	tab	
ashow	l h ch ashow		
astore	a0…an-1 astore	tab	
	tab a0…an-1 array astore def		
atan	n d atan	ang	num dénominateur angle
awidthshow	l1 h1 c l h ch awidthshow		

TAB. 15 – *Opérateurs A*

op.	la pile avant, l'opérateur	la pile après	
begin	dict begin		
bind	proc1 bind	proc2	
bitshift	n nb bitshift	m	
bytesavailable	fic bytesavailable	n	

TAB. 16 – *Opérateurs B*

op.	la pile avant, l'opérateur	la pile après	
ceiling	x ceiling	n	
charpath	ch bool charpath		
clear	clear	pile vide	
cleartomark	[a0. . .an cleartomark		vide jusqu'à [
clip	clip		
clippath	clippath		
closepath	closepath		ferme le chemin courant
concat	matr concat		
copy	a0. . .an-1 n copy	a0 . . .an-1 a0 . . .an-1	
copypage	copypage		
curveto	x1 y1. . .y3 curveto		crbe de Bézier du pt cr^{t} à x3 y3

TAB. 17 – *Opérateurs C*

op.	la pile avant, l'opérateur	la pile après	
def	nom valeur def		définition
div	num1 num2 div	num3	division
dup	objet dup	objet objet	duplique

TAB. 18 – *Opérateurs D*

op.	la pile avant, l'opérateur	la pile après	
end	end		retire de la pile le dernier dict
exch	val1 val2 exch	val2 val1	échange les deux valeurs

TAB. 19 – *Opérateurs E*

op.	la pile avant, l'opérateur	la pile après	
fill	fill		ferme le chemin et encre l'intérieur
findfont	nom findfont	police	met la police sur la pile
for	ini incr final proc for		boucle for

TAB. 20 – *Opérateurs F*

op.	la pile avant, l'opérateur	la pile après	
grestore	grestore		
gsave	gsave		

TAB. 21 – *Opérateurs G*

op.	la pile avant, l'opérateur	la pile après	
if	bool proc if		
ifelse	bool proc1 proc2 ifelse		
index	vn . . .v1 v0 n index	vn . . .v0 vn	

TAB. 22 – *Opérateurs I*

op.	la pile avant, l'opérateur	la pile après	
lineto	x y lineto		
moveto	x y moveto		
mul	val1 val2 mul	produit	multiplication
neg	val neg	v	opposé v = -val
newpath	newpath		efface le chemin courant
pop	val pop		efface val de la pile
print	ch print		vers fichier (pas l'imprimante)
pstack	obj1 ...objn pstack	obj1 ...objn	copie ds fichier de sortie
put	ch ind obj put		remplace ds ch (ou tabl dict)
quit	quit		quitte

TAB. 23 – *Opérateurs L - M*

op.	la pile avant, l'opérateur	la pile après	
rand	rand	n	de 0 à $2^{31} - 1$ au hasard
rcurveto	dx1...dy3 rcurveto		prolonge par crb Bézier
read	fic read	val true	lit 1 octet, (false ssi fin)
readhexstring	fic ch readhexstring	ss-ch bool	false si fin fic
readline	fic ch readline	ss-ch bool	
readstring	fic ch readstring	ss-ch bool	
repeat	n {proc} repeat		boucle repeat (n fois)
restore	état restore		restaure l'état antérieur
rlineto	dx dy rlineto		trait (relatif)
rmoveto	dx dy rmoveto		change le pt courant (relatif)
roll	an . a1 n j roll	aj . a1 an .aj+1	
rotate	ang rotate		rot ds ss des aig d'une mtre
round	m round	n	entier le plus proche
run	nomfic run		exécution de nomfic

TAB. 24 – *Opérateurs R*

op.	la pile avant, l'opérateur	la pile après	
save	save	état	sauvegarde complète
scale	sx sy scale		changment d'échelle
scalefont	police taille scalefont	police	change de taille
search	ch ct search	ch2 ct ch1 true	échge ch1 ch2 sinon : ch false
setdash	tab dec setdash		motif décalage
setflat	n setflat		
setfont	police setfont		devient la police courante
setgray	valeur setgray		val valeur de 0 à 1
sethsbcolor	h s b sethsbcolor		
setlinecap	n setlinecap		n= 0 1 2
setlinejoin	n setlinejoin		n= 0 1 2
setlinewidth	larg setlinewidth		fixe la larg du trait
setmiterlimit	n setmiterlimit		limite nb arêtes
setrgbcolor	r g b setrgbcolor		rge vert bleu de 0 à 1
setscreen	l i {proc}setscreen		
settransfer	{proc}settransfer		
show	chaîne show		trace la chaîne
showpage	showpage		imprime la page
stroke	stroke		trace la ligne du chem crt
sub	val1 val2 sub	val	soustraction val1 - val2
translate	x y translate		translation de l'origine

TAB. 25 – *Opérateurs S - T*

Liste des tableaux

Table des figures

Table des matières