Commandes METAPOST

20 janvier 2008

Ceci est un aide mémoire sur les commandes de bases METAPOST. Cette liste est non exhaustive donc elle est modifiable pour qui veut le faire. Cette aide ne dispense aucunement de la lecture de la traduction de Pierre FOURNIER et de Jean-Côme CHARPENTIER d' ***Un Manuel de*** METAPOST de John D.HOBBY.

★ LES TYPES DE DONNÉE ★

Commande	Explications	Options ou exemple
`numeric`	Ce sont des quantités numériques	`numeric a ; a=1/6432 ;`

`pair`
 	
`z<suffixe>`
 	
`xpart` et `ypart`
	Ce sont des couples (pour répérer des points)
	Abréviation pour `(x<suffixe>,y<suffixe>)`
	Permet d'extraire les composantes d'une `pair`
		`pair A ; A :=(1,2) ;`
		
		`z6=(4,3) ;`
		`z5=(x6+2,y6-3) ;`
		`q :=((ypart p)+5,(xpart p)-2) ;`

`path`	Ce sont des chemins, traits droits avec - - et courbe de Béziers avec ..
	Si on utilise .. on peut définir les tangentes au point considéré avec "une coordonnée de vecteur"
	Si on veut un chemin fermé...
		`path trait, l ;`
		`trait :=(2,3)--(3,4) ;`
		`l :=(1,2)..(2,3)..(4,5) ;`
		`l :=(2,1){2,3}..(3,5) ;`
		
		`l :=(1,2)..(2,3).. (4,5)..cycle ;`

`picture`	On stocke un dessin dans une figure	`picture mapicture ;` `mapicture := Image(<dessin>) ;`
`transform`	On stocke une transformation (plus complexe que les sept définies[1]	`transform T ; T = identity xscaled -1 rotated 90 shifted (1,1) ;`

[1] voir TRANSFORMATIONS AFFINES pour les différentes transformations possibles `[U+FFFD]` réaliser

★ LES TYPES DE DONNÉE (SUITE) ★

`pen`	Décaration d'une pointe	`pen nvstylo ; nvstylo = makepen`[2] `(fullcircle xscaled 2 yscaled 1) ;`
`[]`	Déclaration de Tableau qui évite la déclaration : `path ligne1, ligne2... ;`	`picture disque[] ; path ligne[] ;`

★ LE TRAIT ★

`draw`	Comme son nom l'indique...	`draw (4,5)--(6,7) ;` `draw mapicture ;`
`drawarrow`	Une flêche au dernier point du chemin	`drawarrow z1..z2`
`drawdblarrow`	double flêche	`drawdblarrow z1..z2`
`undraw`	"Gomme" : trace avec la couleur de fond	`undraw ligne ;`
Quelques options		

`withcolor`
 	
`dashed`
	La couleur
	
	Les motifs de traits :
	`evenly` : les pointillés réguliers
	`withdots` : les lignes de points
	`dashpattern` : on définit sur combien de point la ligne est apparente et sur combien elle disparait
		`draw ligne withcolor red ;`
		
		`draw ligne dashed evenly ;`
		`draw ligne dashed withdots ;`
		`draw dashpattern(on 15bp off 15bp) dashed evenly ;`

[2] Voir la rubrique STYLO OU PINCEAU pour l'utilisation de `makepen`

★ Le Remplissage ★

`fill`	"Colorier" : remplir avec une couleur (le chemin doit être fermé)	`fill patatoide ;`
`unfill`	"Gomme" : colorie avec la couleur de fond	`unfill petitcercle ;`
	De la même manière qu'avec `draw`, on peut utiliser l'option `withcolor`	`fill patate withcolor red ;`

★ Les chemins ★

`buildcycle`	Détermine le chemin étant l'intersection d'autres chemins Le chemin définit par 4 chemins ne se coupant deux [U+FFFD] deux qu'en 1 point	`buildcycle(moitcercle, carre) ;` `buildcycle(c1,c2,c3,c4) ;`
`intersectiontimes`	Détermine le "temps" d'intersection de deux chemins, paramétrage de chacun des chemins [U+FFFD] l'intersection	`che1 intersectiontimes che2 ;`
`intersectionpoint`	Point d'intersection de deux chemins	`M=cercle intersectionpoint droite ;`

`point<numeric>of <chemin>`
`length<chemin>`
	Le point x du chemin p
	
	Donne le "paramètre" maximum du chemin
		`point 6 of circle ;`
		
		`length cercle ;`

`subpath (t1,t2) of <path>`	Donne le chemin du paramètre t1 au paramètre t2 du chemin	`subpath (0,6) of circle ;`

★ Les équations ★

`:=`	L'affectation	`a :=3cm`
`=`	L'équation de "bases"	`a :=3 ; b=a ;`
k`[z1,z2]`	La définition d'un point d'un segment, dans le cas général, ceci est équivalent au point `z1+`k`(z2-z1)`	`z3=1/3[z1,z2] ;`
`whatever[z1,z2]`	On définit un point sur le segment `[z1,z2]` mais n'importe où	`z3=wathever[z1,z2] ;`

Tant que les systèmes d'équations sont linéaires, METAPOSTsait les résoudre.

★ Labels ★

`label`	L'expression la plus simple est `label("`*expression*`",`*pair*`) ;`	`label("A",(0,0)) ;`
Les options		
`btex ... etex`	Utilisation de TEX pour la mise en forme des étiquettes	`label(btex $M$ etex, (0,0)) ;`
	La localisation du label autour du point	
`rt`	Décalage [U+FFFD] droite	`label.rt(btex $x=y$ etex,M) ;`
`lft`	Décalage [U+FFFD] gauche	`label.lft(btex $x=y$ etex,M) ;`
`top`	Décalage en haut	`label.top(btex $x=y$ etex,M) ;`
`bot`	Décalage en bas	`label.bot(btex $x=y$ etex,M) ;`
`urt`	Décalage en haut [U+FFFD] droite	`label.urt(btex $x=y$ etex,M) ;`
`ulft`	Décalage en haut [U+FFFD] gauche	`label.ulft(btex $x=y$ etex,M) ;`
`lrt`	Décalage en bas [U+FFFD] droite	`label.lrt(btex $x=y$ etex,M) ;`
`llft`	Décalage en bas [U+FFFD] gauche	`label.llft(btex $x=y$ etex,M) ;`
`dotlabel`	Ajout d'un point au point de coordonnée indiqué	`dotlabel.urt(btex $M$ etex,M) ;`

★ Stylo ou pinceau ★

`pickup<stylo>`	Prendre le stylo défini	`pickup pencircle scaled 2cm ; draw fullcircle ;`
`withpen<stylo>`	faire... avec le stylo défini	`draw fullcircle withpen pencircle scaled 2cm ;`
`makepen`	Définir le chemin du nouveau stylo définit	`pen monstylo ; monstylo = makepen ((1,0)- -(0,.5)- -(0,0)- -cycle)) ;`

Le stylo `pencircle` est le stylo par défaut, comme son nom l'indique, il est circulaire, le stylo `pensquare` est déj[U+FFFD] défini est c'est un... carré.

★ Transformations Affines ★

(x,y) `shifted` (a,b)	Comme son nom l'indique : déplace *i.e* donne le `pair` $(x+a, y+b)$	`draw fullcircle shifted M ;`
(x,y) `rotated` θ	Effectue la rotation de θ **degrès** autour de l'origine *i.e* donne le `pair` $(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$	`M= A rotated 30 ;`
(x,y) `slanted` a	Donne le `pair` $(x+ay, y)$	
(x,y) `scaled` a (x,y)(`xscaled` ou `yscaled`)a	"Multiplie" le `pair` *i.e* donne le `pair` (ax, ay) Multiplie uniquement une coordonnée	`gcercle=fullcircle scaled 4cm ;`
(x,y) `zscaled` (a,b)	Peut-être compris comme "Multiplication" par un complexe *i.e* donne le `pair` $(ax-by, bx+ay)$	
`reflectedabout` (p,q)	La figure symétrique par rapport `[U+FFFD]` la droite définie par p et q	
`roatatedaround` (P,θ)	Effectue la rotation de θ degrès autour du point P	

On peut définir des transformations plus complexes en se créant ses propres transformations avec les varaibles de type `transform`.
Ensuite ces transformations ne sont pas valables uniquement pour les `pair`, on peut les appliquer aussi `[U+FFFD]` des `path`, des `picture`...

★ Découpage ★

`currentpicture`	picture où sont stockés tous les "dessins" réalisé au cours de la figure	
`clip`<picture> `to` <chemin fermé>	Permet d'éliminer tout ce qu'il y a (de la `picture`) en dehors du chemin	`clip current picture to (3,2)- -(4,5)- -(-4,-5)- -cycle ;`

★ Macro ★

`def`<nouvelle fonction> `(expr`<variables>`)= ... enddef ;`	Définition de nouvelles fonctions, les possibilité sont énormes	`def milieu(expr a,b) = .5[a,b] enddef ;`

★ Programmation ★

`for`<expression>`=` <numeric> `step` <numeric> `until` <numeric> `:` ...`endfor ;`	Tant de fois répéter l'action... `step` est le pas d'indentation et `until` la valeur max	`for i :=0 step 2 until 6 : draw (5i,4)-(4i,8) ; endfor ;`
`upto`	Macro permettant d'éviter d'écrire `step 1`	`for i :=0 upto 4 :...`
`downto`	Macro permettant d'éviter d'écrire `step -1`	`for i :=20 downto 9 :...`
`if :...fi ;`	Test comme en programmation	`if a=b : unfill cercle ; fi ;`
Options		
`elseif :` ou `else :`	Avec quelques bases en anglais, on comprend...	
tests	différent <>, égal =,supérieur >,inférieur <	