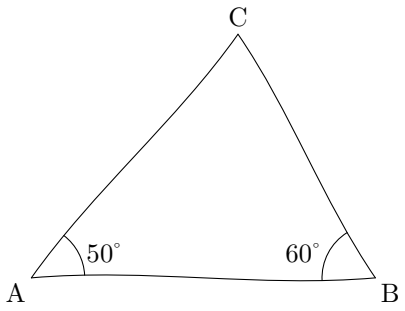
Documentation sur le fichier geometriesyr15.mp

Christophe Poulain

Version 1.0

Table des matières

Cette nouvelle version qui marque le passage de geometriesyr13.mp à geometriesyr15.mp apporte de nouvelles fonctions et surtout une modification majeure qui justifie, à elle seule, l'abandon de la version 14.
Si les nouvelles fonctions seront détaillées par la suite, un mot sur la modification majeure : **le dessin à main levée** dont voici de suite un exemple.



C'était une idée qui me trottait dans la tête depuis pas mal de temps. Mais comment faire du dessin à main levée avec un ordinateur ? Sans forcément rentrer dans les détails, j'ai utilisé des courbes de Beziers pour donner un aspect « main levée » au dessin obtenu. Plusieurs contraintes et nouvelles idées sont venues me forcer à reprendre très souvent le code jusqu'à obtenir ce que je souhaitais. J'attends bien entendu les remarques éventuelles des futurs utilisateurs.

1 Détails du fichier

Ce fichier fait appel aux fichiers :
- CONSTANTES.MP qui, comme son nom l'indique, contient les différents paramètres nécessaires aux tracés (couleurs, unités de longueur,...).
  Les constantes disponibles sont π, e, c la conversion d'un radian en degrés.
  Les couleurs disponibles sont rouge, bleu, vert, kaki, blanc, noir, orange, violet, rose, ciel, orangevif, jaune et gris. Elles s'utilisent avec ces noms francisés.
- PAPIERS2.MP qui permet de produire[1] différents types de papiers (millimétré, 5×5, isométrique, ...)

et à plusieurs fichiers de base de MetaPost.

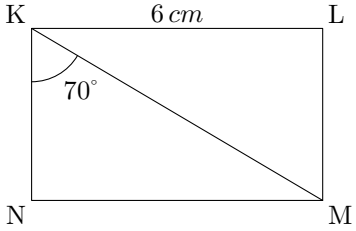
1.1 Environnement général de la figure.

- La macro `figure(xa,xb,ya,yb)` va limiter tout le schéma à l'intérieur d'un cadre dont le sommet inférieur gauche a pour coordonnées (x_a, y_a) et le sommet supérieur droit (x_b, y_b). A noter que cette macro a une indentation automatique : plusieurs figures avec pour extension .1, .2, ...peuvent être créees les unes à la suite des autres.
  Avec cette macro, le code du schéma devra se conclure par `fin;`.
  Cette macro permet un *tracé classique* des figures contrairement à la macro suivante.
- La macro `figuremainlevee(xa,xb,ya,yb)`, tout comme la précédente, va limiter tout le schéma à l'intérieur d'un cadre dont le sommet inférieur gauche a pour coordonnées (x_a, y_a) et le sommet supérieur droit (x_b, y_b).
  Avec cette macro, le code du schéma devra se conclure par `finmainlevee;` ; MetaPost repassant alors immédiatement en mode « classique ». Cela permet de mélanger les différents types de figures au sein d'un même fichier.
  Cette macro permet un ***tracé à mainlevée***.

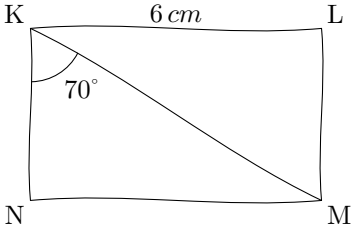
J'ai choisi cette possibilité pour obtenir des codes qui ne changent pas en fonction du tracé voulu : seuls changent les macros qui cadrent la figure elle-même. Avec le « même code », on peut obtenir deux figures différents d'aspects mais présentant les mêmes caractéristiques.

[1] Voir la page 10 pour connaître les commandes disponibles.

```
figure(0,0,5cm,5cm);
...
fin;
```




```
figuremainlevee(0,0,5cm,5cm);
...
finmainlevee;
```



1.2 Affichage.

Pour l'affichage des points, on dispose d'un paramètre `marque_p` qui peut prendre les valeurs suivantes :

rien : dans ce cas, aucun pointage (valeur par défaut) ;

plein : dans ce cas, on pointe avec un disque noir ;

creux : dans ce cas, on pointe avec un cercle ;

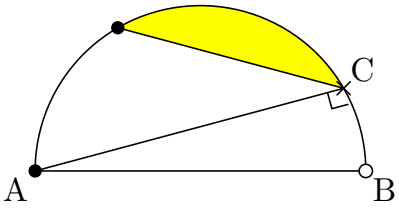
croix : dans ce cas, on pointe avec une croix.

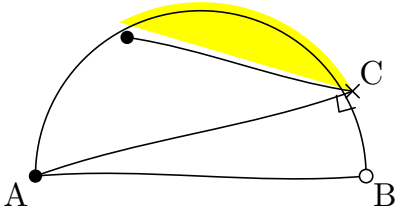
Une fois ce choix fait, on peut repérer un point de deux façons :

pointe(A,B,C,...) permet de repérer les points avec le type de marquage choisi sans les nommer,

nomme.pos(A) permet de repérer le point A avec le type de marquage choisi en le nommant à la position pos[2]

On trouvera également les macros francisées `trace` et `remplis` pour remplacer respectivement `draw` et `fill`.






```
figure(0,0,7u,6u);
pair A,B,C,D;
path cc,cd;
A=u*(1,1);
B=u*(6,1);
cc=cercledia(A,B);
cd=arccercle(B,A,iso(A,B));
C=pointarc(cc,30);
D=pointarc(cc,120);
remplis (C--arccercle(C,D,iso(A,B))--cycle)
 withcolor jaune;
trace A--B;
trace cd;
trace A--C--D;
trace codeperp(A,C,B,5);
marque_p:="plein";
nomme.llft(A);
pointe(D);
marque_p:="creux";
```

[2]Ce sont les positions classiques de MetaPost : llft,lft,ulft,top,urt,rt,lrt,bot

```
nomme.lrt(B);
marque_p:="croix";
nomme.urt(C);
fin;
```

1.3 Différents types de codage.

Voici la liste des codages disponibles, ils sont tous à utiliser avec la commande `trace` :

Angle droit : `codeperp(A,B,C,5)` produit un angle droit en B avec pour « épaisseur » 5.

longueurs égales : `codesegments(A,B,C,D,n)` code les segments $[AB]$ et $[CD]$ avec le codage d'ordre n : 1, 2, 3 pour autant de traits, 4 pour la croix et 5 pour le cercle.

Codage d'un angle : `codeangle.pos(A,B,C,2,5mm,btex nom etex)` produit un codage de l'angle $\widehat{ABC}$ donné dans le sens direct par 2 arcs de cercles dont le premier est situé à $5\,mm$ du sommet B et dont le nom est placé dans la position pos par rapport au « milieu » des arcs de cercles.

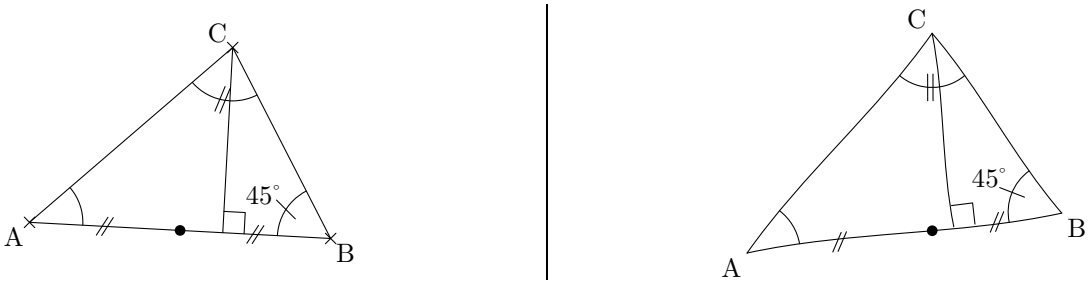
Marquage d'un angle formé par 3 points : `marqueangle(A,B,C,2)` produit un codage de l'angle $\widehat{ABC}$ donné dans le sens direct par 1 arc de cercle et 2 tirets sur cet arc. Le rayon de l'arc de cercle crée (par cette macro et les deux suivantes) est donné par `marque_a` valant 20 par défaut.

Marquage d'un angle formé par deux demi-droites : `Marqueangle(aa,bb,2)` produit un codage de l'angle formé par les demi-droites $[aa)$ et $[bb)$ donné dans le sens direct par 1 arc de cercle et 2 tirets sur cet arc.

Cette macro est apparue nécessaire pour le codage des angles formés par une bissectrice, dans le cadre d'une figure à main levée. La bissectrice étant construite de manière « aléatoire » à chaque appel de la macro, il faudra la déclarer avant utilisation.

Coloriage d'un angle formé par 3 points : `coloreangle(A,B,C)` permet de construire un chemin fermé qui devient donc coloriable.

Codage de 2 droites parallèles : `marque_para(d1,d2,0.75)` indique par deux flèches coupés par le symbole parallèle au milieu, que les droites (d_1) et (d_2) le sont ; 0.75 représentant la position de la flèche sur la première droite.




```
figure(0,0,8u,8u);
pair A,B,C,I,H;
trace triangleqcq(A,B,C);
I=milieu(A,B);
H=projection(C,A,B);
trace segment(C,H);
trace codesegments(B,I,I,A,2);
trace codeperp(C,H,B,8);
trace codeangle.ulft(C,B,A,1,btex 45\degres etex);
trace marqueangle(A,C,B,2);
trace Marqueangle(demidroite(A,B),demidroite(A,C),0);
nomme.llft(A);
nomme.lrt(B);
```

```
nomme.ulft(C);
marque_p:="plein";
pointe(I);
marque_p:="non";
fin;
```

1.4 Points.

Quelques macros pour définir des points particuliers simplement :

iso(A,B,C,...) construit l'isobarycentre des points A, B, C, ...(uniquement dans le cas classique du tracé) ;

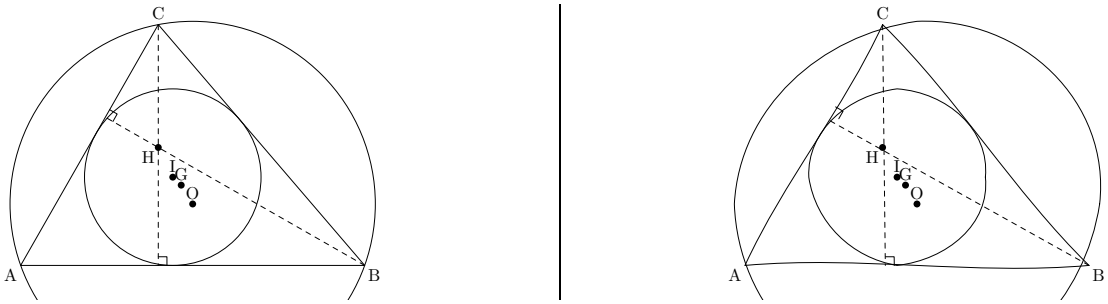
milieu(A,B) construit le milieu du segment $[AB]$. Dans le tracé à « main levée », il y a un décalage aléatoire à chaque appel de cette macro. Il faut donc définir le point précisément si on veut le réutiliser par la suite. (Voir exemple ci-dessus).

projection(M,A,B) construit le projeté orthogonal de M sur la droite (AB). Ici aussi, lors d'un tracé à main levée, il y a un décalage aléatoire à chaque appel.

CentreCercleC(A,B,C) construit le centre du cercle circonscrit au triangle ABC.

Orthocentre(A,B,C) construit l'orthocentre du triangle ABC.

CentreCercleI(A,B,C) construit le centre du cercle inscrit au triangle ABC.




```
figure(0,0,12u,12u);
pair A,B,C,O,H,G,I;
A=u*(1,1);
B=u*(11,1);
C=u*(5,8);
trace triangle(A,B,C);
O=CentreCercleC(A,B,C);
H=Orthocentre(A,B,C);
G=iso(A,B,C);
I=CentreCercleI(A,B,C);
marque_p:="plein";
nomme.top(G);
nomme.llft(H);
nomme.top(O);
nomme.top(I);
trace cercles(A,B,C);
trace C--projection(C,A,B) dashed evenly;
trace B--projection(B,A,C) dashed evenly;
trace codeperp(B,projection(B,A,C),C,5);
trace codeperp(C,projection(C,A,B),B,5);
trace cercles(I,projection(I,A,B));
marque_p:="non";
nomme.llft(A);
```

```
nomme.lrt(B);
nomme.top(C);
fin;
```

1.5 Cercles.

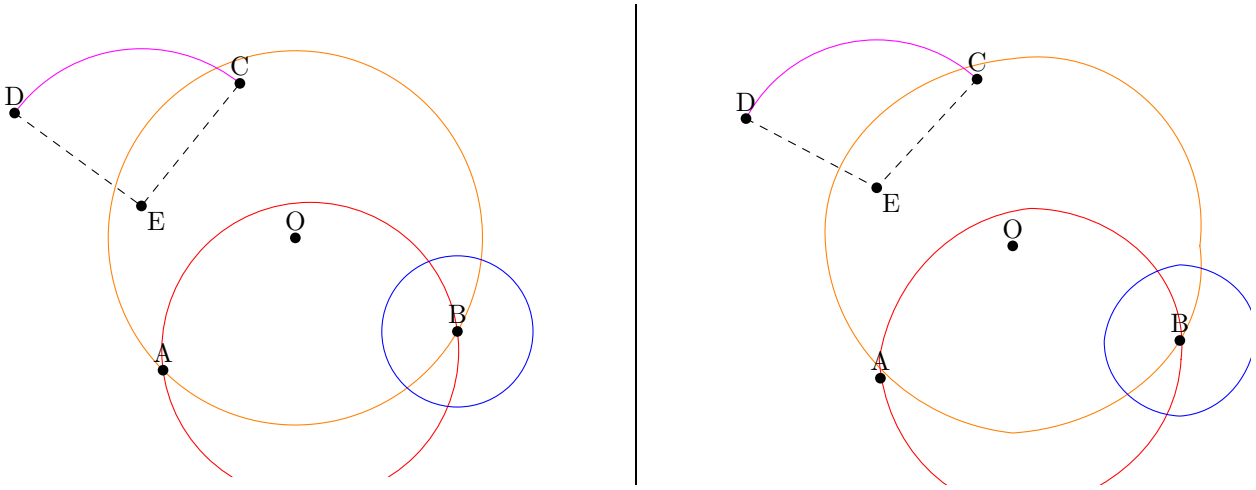
cercledia(A,B) construit le cercle de diamètre $[AB]$.

cercles(*expression*) peut construire trois types de cercles :
cercles(A,2cm) construis le cercle de centre A et de rayon $2\,cm$;
cercles(F,G) construis le cercle de centre F et passant par G.
cercles(A,G,D) construis le cercle circonscrit au triangle AGD.

pointarc(cc,45) détermine le point du cercle `cc` qui a pour angle par rapport à l'horizontale 45°.

arccercle(A,B,0) construit l'arc de cercle de centre O et allant de A à B dans le sens direct ; A et B étant sur un même cercle. A noter que quel que soit le type de tracé choisi, le dessin sera le même.

Arc de cercle intervenant dans une construction (pour le report de longueur par exemple) : `coupdecompas(A,B,10)` permet de construire un arc de cercle de centre A, passant par B, commençant 10°avant l'angle du vecteur $\overrightarrow{AB}$ (en respectant les conventions de Metapost) et s'arrêtant 10°après cet angle.




```
figure(-3u,-u,8u,10u);
pair A,B,C,D,E,O;
O=u*(3.5,3.5);
A=u*(1,1);
path cc,cd;
cc=cercles(O,A);
trace cc withcolor orange;
B=pointarc(cc,45);
trace cercledia(A,B) withcolor rouge;
trace cercles(B,1cm) withcolor jaune;
cd=cercles(A,B);
C=pointarc(cd,75);
D=pointarc(cd,120);
E=CentreCercleC(D,O,C);
trace arccercle(C,D,E) withcolor violet;
trace D--E--C dashed evenly;
marque_p:="plein";
nomme.lrt(E);
nomme.top(A);
```

```
nomme.top(B);
nomme.top(C);
nomme.top(D);
nomme.top(O);
fin;
```

1.6 Droites.

segment(A,B) construit le segment $[AB]$.

droite(A,B) construit la droite (AB).

demidroite(A,B) construit la demi-droite $[AB)$.

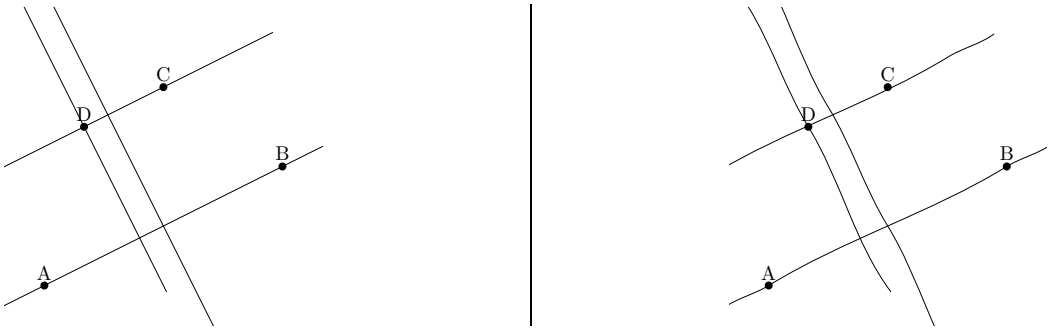
bissectrice(A,B,C) construit la bissectrice de l'angle $\widehat{ABC}$ donné dans le sens direct. Dans le cas d'un dessin à main levée, une rotation aléatoire de la bissectrice est envisagée. Il convient donc de la définir avant une réutilisation ultérieure.

mediatrice(A,B) construit la médiatrice du segment $[AB]$. Dans le cas d'une figure à main levée, comme le milieu est construit de manière « aléatoire » à chaque appel, il peut ne pas y avoir correspondance : la médiatrice ne passe pas forcément par le milieu précédemment crée. Dans ce cas, il vaut mieux passer par la macro suivante en ayant eu soin de déclarer le milieu du segment avant.

perpendiculaire(A,B,I) construit la perpendiculaire à la droite (AB) passant par I.

parallele(A,B,I) construit la parallèle à la droite (AB) passant par I.

triangleqcq(A,B,C) construit un triangle ABC (avec définition des points A, B, C et appel ultérieur possible de ces points) qui est « vraiment » quelconque au sens qu'il ne possède aucune propriété particulière.




```
figure(0,0,8u,8u);
pair A,B,C,D;
A=u*(1,1);
B=u*(7,4);
C=u*(4,6);
trace droite(A,B);
trace parallele(A,B,C);
trace mediatrice(A,B);
D=u*(2,5);
trace perpendiculaire(A,B,D);
nomme.top(A);
nomme.top(B);
nomme.top(C);
nomme.top(D);
fin;
```

1.7 Transformations.

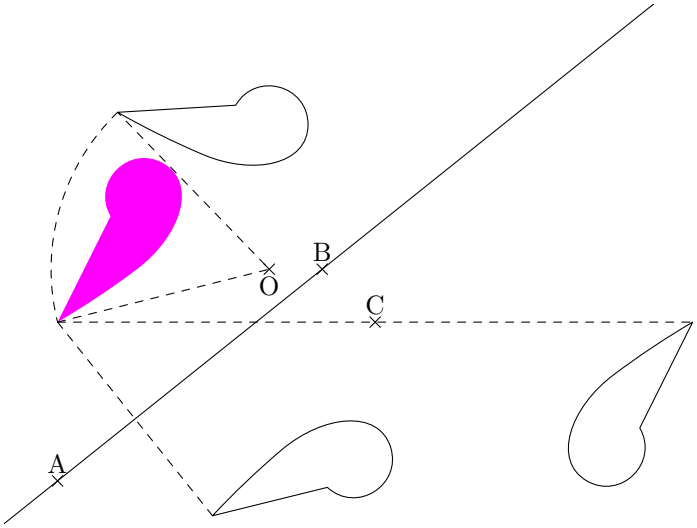
Dans ce qui suit, *objet* représente un objet connu de MetaPost (un point, un chemin, une figure,...)

rotation(*objet*,O,60) construit l'image de l'objet par la rotation de centre O et d'angle 60° dans le sens direct.

Symetrie(*expression*) permet de construire 2 types d'image par symétrie :

symetrie(*objet*,B) image de l'objet par la symétrie centrale de centre B.

symetrie(*objet*,B,C) image de l'objet par la symétrie axiale d'axe (BC).




```
figure(0,0,20u,10u);
pair A,B,C,D,O;
A=u*(1,1);
B=u*(6,5);
O=u*(5,5);
C=u*(7,4);
path ima;
pair W[];
W1=u*(1,4);
ima=u*(1,4)--u*(2,6)..u*(3,7)..u*(2.5,5)..cycle;
remplis ima withcolor violet;
trace rotation(ima,O,-60);
trace symetrie(ima,C);
trace symetrie(ima,A,B);
trace arccercle(rotation(W1,O,-60),W1,O)
 dashed evenly;
trace W1--O--rotation(W1,O,-60) dashed evenly;
trace W1--symetrie(W1,C) dashed evenly;
trace W1--symetrie(W1,A,B) dashed evenly;
trace droite(A,B);
marque_p:="croix";
nomme.top(A);
nomme.top(B);
nomme.bot(O);
nomme.top(C);
fin;
```

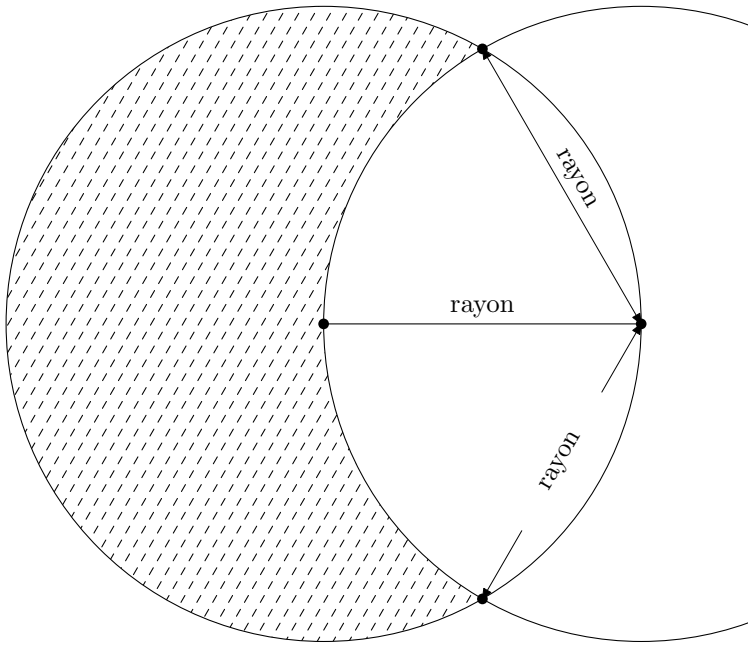
1.8 Sucres

hachurage(chemin, angle, ecart, type de hachures) permet d'hachurer *un chemin* **fermé** avec des hachures faisant un *angle* par rapport à l'horizontale, hachures espacées d'un *ecart* et avec un certain style de traçage : 0 correspond à un trait continu, 1 un trait pointillé, 2 un trait d'axe.

cotation(A,B,2mm,3mm,btex nom etex) trace une flèche de cotation pour le segment $[AB]$ située à 2mm au dessus du segment $[AB]$ et une côte *nom* située à 3mm au dessus de la flèche de cotation. On peut bien sûr placé la flèche et la côte de manière « négative ».

appelation(A,B,2mm,btex nom etex) permet de nommer la longueur du segment $[AB]$ avec *nom* situé à $2\,mm$ au dessus du segment $[AB]$.

cotationmil(A,B,2mm,20,btex nom etex) même chose que cotation sauf que la côte est placée au milieu de la flèche ; pour cela, on a laissé un ecart de 2Opt autour du milieu de la flèche.




```
figure(0,0,15u,15u);
pair A,B,C,M,N,R;
A=u*(7,7);
B=u*(13,7);
path cc,cd,ce;
cc=cercles(A,B);
cd=cercles(B,A);
M=cc intersectionpoint cd;
N=symetrie(M,A,B);
R=pointarc(cd,180);
trace B--R;
ce=arccercle(M,N,A)--reverse(arccercle(M,N,B))--cycle;
trace hachurage(ce,60,0.3,1);
trace cc;
trace cd;
marque_p:="plein";
pointe(R,M,N,B);
trace cotation(M,B,0,2mm,btex rayon etex);
trace cotationmil(N,B,0,30,btex rayon etex);
trace appelation(R,B,2mm,btex rayon etex);
fin;
```

2 Documentation sur papiers2.mp

Ce fichier, qui doit être utilisé avec geometriesyr15.mp, permet de construire différents types de papiers (millimétré, 5×5, isométrique,...)
Le papier demandé sert en fait de fond d'écran pour le fichier image : en effet, le papier s'inscrira complètement dans le cadre crée à l'occasion de l'utilisation de la commande `figure(xa,ya,xb,yb)`.

2.1 Types de papier

Ils s'obtiennent tous avec la commande `trace` (ou `draw`).

papier millimétré : `papiermillimetre` ;

papier 5×5 : `grille(0.5)` (d'autres utilisations sont possibles, papier 10×10 par exemple) ;

papier cahier (grand carreaux) : `papiercahier` ;

papier pointé : `papierpointe` ;

papier triangulaire : `papiertriangle` ;

papier isométrique : `papierisometrique` ;

papier isométrique pointé : `papierisometriquepointe` ;

pavage hexagonal : `papierhexagonal`.

Tous ces papiers génèrent des unités horizontales, verticales ou plus originales pour les papiers triangulaires et hexagonaux. Cela nous sera utile pour certaines macros.

2.2 Autres fonctions

Ce qui suit est valable essentiellement pour les papiers millimétrés et les grilles.

- On peut aisément définir une origine par `origine((xO,yO));` qui va placé l'origine d'un repère au point de coordonnées $(x_O; y_O)$ dans les unités du papier choisi. Par défaut, l'origine est placé dans le coin en bas à gauche du schéma.
- Une fois ceci fait, on peut redéfinir les unités suivants les axes par `unites(ux,uy)` où u_x et u_y sont les unités choisies par l'utilisateur.
- `placepoint(xA,yA)` va placer, dans le repère choisi par l'utilisateur (origine et unités) le point de coordonnées (x_A, y_A).
- `trace axes` permet de tracer les axes des abscisses et des ordonnées choisis par l'utilisateur.
- Pour la graduation des axes, on dispose d'un marqueur `marque_axe` qui permet de choisir une graduation minimale (repère des unités seules) en le positionnant par `marque_axe:=''simple''` (valeur par défaut) ou une graduation complète (repère de tous les multiples des unités) par `marque_axe:=''complete''`. On utilise les graduations par `graduantx` et\ou `graduanty`.
- Pour le papier triangulaire, on dispose de `placepointtri(a,b)` qui place le point dans le repère du papier triangulaire.
- Pour le papier hexagonal, on dispose de `reperehexa(col,lign)` qui repère l'hexagone situé dans la colonne `col` et à la ligne `lign`. On dispose également de `Centrehexa(col,lign)` qui permet d'inscrire ou dessiner au centre de l'hexagone situé dans la colonne `col` et à la ligne `lign`.

A Fichier geometriesyr15.mp

```
%%=============================================
%% GEOMETRIESYR.MP
%% christophe.poulain@melusine.eu.org
%% Création : 19 Février 2003
%% Dernière modification : 26 Août 2004
%%=============================================
%---------------------------------------------
% Appel fichier
%---------------------------------------------
input constantes;
input papiers2;
%---------------------------------------------
% La figure (début et fin) JMS/CP
%---------------------------------------------
path feuillet;
numeric _tfig,_nfig;
pair coinbg,coinbd,coinhd,coinhg;
_nfig:=0;

string typetrace;
typetrace="normal";

def feuille(expr xa,ya,xb,yb) =
  feuillet := (xa,ya)--(xa,yb)--(xb,yb)--(xb,ya)--cycle;
  coinbg := (xa,ya);
  coinbd := (xb,ya);
  coinhd := (xb,yb);
  coinhg := (xa,yb);
  z.so=coinbg;
  z.ne=coinhd;
  extra_endfig := "clip currentpicture to feuillet;" & extra_endfig;
enddef;
def figure(expr xa,ya,xb,yb) =
  _nfig:=_nfig+1;
    beginfig(_nfig);
    feuille(xa,ya,xb,yb);
    _tfig:= if (xb-xa)>(yb-ya): xb-xa else: yb-ya fi;
  enddef;

def figuremainlevee(expr xa,ya,xb,yb) =
  _nfig:=_nfig+1;
  typetrace:="mainlevee";
  beginfig(_nfig);
    feuille(xa,ya,xb,yb);
    _tfig:= if (xb-xa)>(yb-ya): xb-xa else: yb-ya fi;
enddef;

def fin =
  endfig;
enddef;
```

```
def finmainlevee=
endfig;
typetrace:="normal";
enddef;

%%-----------------------------------------------
%% Les marques (JMS)
%%-----------------------------------------------
string marque_p;
marque_p := "non";
marque_r := 20;
marque_a := 20;
%-----------------------------------------------
% Les tables
%-----------------------------------------------
numeric _tn;
_tn:=0;
pair _t[];
%%-----------------------------------------------
%% Procédures d'affichage
%%-----------------------------------------------
def MarquePoint(expr p)=
  %JMS
  if marque_p = "plein":
    fill fullcircle scaled (marque_r/5) shifted p;
  elseif marque_p = "creux":
    fill fullcircle scaled (marque_r/5) shifted p withcolor white;
    draw fullcircle scaled (marque_r/5) shifted p;
  %fin JMS
  elseif marque_p = "croix":
    draw (p shifted (-u/10,u/10))--(p shifted (u/10,-u/10));
    draw (p shifted (-u/10,-u/10))--(p shifted (u/10,u/10));
  fi
enddef;
%JMS
vardef pointe(text t) =
  for p_ = t: if pair p_: MarquePoint(p_); fi endfor;
enddef;
vardef nomme@#(suffix p)=
  MarquePoint(p);
  label.@#(str p,p);
enddef;
def trace expr o =
    if path o: draw o else: draw o fi
enddef;
def remplis expr o =
    if path o: fill o else: fill o fi
enddef;
vardef triangle(expr aa,bb,cc)=
  if typetrace="mainlevee":
    save $;
```

```
    picture $;
    $=image(
      trace aa{dir(angle(bb-aa)+5)}..bb{dir(angle(bb-aa)+5)};
      trace cc{dir(angle(bb-cc)+5)}..bb{dir(angle(bb-cc)+5)};
      trace aa{dir(angle(cc-aa)+5)}..cc{dir(angle(cc-aa)+5)};
      );
  else:
    save $;
    path $;
    $=aa--bb--cc--cycle;
  fi;
  $
enddef;
%fin JMS
vardef bary(expr a,b,c,d)=
  save $;
  pair $;
  numeric t[];
  t1=uniformdeviate(1);
  t2=uniformdeviate(1);
  t3=uniformdeviate(1);
  t4=uniformdeviate(1);
  $=(1/(t1+t2+t3+t4))*(t1*a+t2*b+t3*c+t4*d);
  $
enddef;
vardef triangleqcq(text t)=
  save $;
  pair pointchoisi[];
  pointchoisi1:=bary(coinbg,1/4[coinbg,coinbd],iso(coinbg,iso(coinhg,coinhd)),
iso(coinhg,coinbg));
  pointchoisi2:=bary(coinbd,3/4[coinbg,coinbd],iso(coinbd,iso(coinhg,coinhd)),
iso(coinhd,coinbd));
  test:=uniformdeviate(1);
  choix:=43+uniformdeviate(4);
  ecart:=abs(45-choix);
  relation:=60-(ecart/2)+uniformdeviate(ecart);
  if test<0.5 :
    pointchoisi3:=droite(pointchoisi1,rotation(pointchoisi2,pointchoisi1,choix))
intersectionpoint droite(pointchoisi2,rotation(pointchoisi1,pointchoisi2,
-relation));
  else :
    pointchoisi3:=droite(pointchoisi2,rotation(pointchoisi1,pointchoisi2,-choix))
intersectionpoint droite(pointchoisi1,rotation(pointchoisi2,pointchoisi1,
relation));
  fi
  j:=1;
  for p_=t:
    p_=pointchoisi[j];
    j:=j+1;
  endfor;
  if typetrace="mainlevee":
    picture $;
```

```
  else:
    path $;
  fi;
  $=polygone(pointchoisi1,pointchoisi2,pointchoisi3);
  $
enddef;

vardef polygone(text t)=
  pair aaa[];
  j:=0;
  for p_=t: if pair p_:
      j:=j+1;
      aaa[j]=p_;
    fi;
  endfor;
  aaa[j+1]:=aaa[1];
  if typetrace="mainlevee":
    save $;
    picture $;
    $=image(
      for k=1 upto j:
        trace segment(aaa[k],aaa[k+1]);
      endfor;
      );
  else:
    save $;
    path $;
    $=aaa1--
    for k=2 upto j:
      aaa[k]--
    endfor
    cycle;
  fi;
  $
enddef;

vardef chemin(text t)=
  pair aaa[];
  j:=0;
  for p_=t: if pair p_:
      j:=j+1;
      aaa[j]=p_;
    fi;
  endfor;
  if typetrace="mainlevee":
    save $;
    picture $;
    $=image(
      for k=1 upto (j-1):
        trace segment(aaa[k],aaa[k+1]);
      endfor;
      );
```

```
  else:
    save $;
    path $;
    $=aaa1
    for k=2 upto j:
      --aaa[k]
    endfor;
  fi;
  $
enddef;

%-----------------------------------------------
% Procédures de codage
%-----------------------------------------------
%Codage de l'angle droit de sommet B
vardef codeperp(expr aa,bb,cc,m)=%normalement m=5
  (bb+m*unitvector(aa-bb))--(bb+m*unitvector(aa-bb)+m*unitvector(cc-bb))
--(bb+m*unitvector(cc-bb))
enddef;

%Codage d'un milieu
vardef codemil(expr AA,BB, n) =%extrêmités-angle de codage
  save $,a,b,c,d;
  path $;
  pair a,b,c,d;
  a=1/2[AA,BB];
  b=(a+2*unitvector(BB-AA))-(a-2*unitvector(BB-AA));
  c=b rotated n shifted a;
  d=2[c,a];
  $=c--d;
  $
enddef;
%Codage de deux segments égaux
vardef codesegments(expr AA,BB,CC,DD,n)=%extrémités des segments(4)-type de codage
  save $,v,w;
  picture $;
  $=image(
    if n=5 :
      draw fullcircle scaled 0.1cm shifted (1/2[AA,BB]);
      draw fullcircle scaled 0.1cm shifted (1/2[CC,DD]);
    elseif n=4 :
      pair v,w;
      v=1/2[AA,BB];
      w=1/2[CC,DD];
      draw codemil(AA,BB,60);
      draw codemil(AA,BB,120);
      draw codemil(CC,DD,60);
      draw codemil(CC,DD,120);
    elseif n=3 :
      draw codemil(AA,BB,60);
      draw codemil(AA,BB,60) shifted (2*unitvector(AA-BB));
      draw codemil(AA,BB,60) shifted (2*unitvector(BB-AA));
```

```
      draw codemil(CC,DD,60);
      draw codemil(CC,DD,60) shifted (2*unitvector(CC-DD));
      draw codemil(CC,DD,60) shifted (2*unitvector(DD-CC));
    elseif n=2 :
      draw codemil(AA,BB,60) shifted unitvector(AA-BB);
      draw codemil(AA,BB,60) shifted unitvector(BB-AA);
      draw codemil(CC,DD,60) shifted unitvector(CC-DD);
      draw codemil(CC,DD,60) shifted unitvector(DD-CC);
    elseif n=1 :
      draw codemil(AA,BB,60);
      draw codemil(CC,DD,60);
    fi;
    );
    $
enddef;
%Codage de l'angle abc non orienté (mais donné dans le sens direct)
§% n fois avec des mesures différentes
vardef codeangle@#(expr aa,bb,cc,nb,nom)=
  save s,p,$;
  path p;
  picture $;
  $=image(
    trace marqueangle(aa,bb,cc,nb);
    label.@#(nom,w);
    );
  $
enddef;

vardef Marqueangle(expr aa,bb,mark)=%codage d'un angle formé par les
% demi-droites aa et bb dans le sens direct avec la marque mark
  save $;
  picture $;
  path conf,rr;
  pair w,tangent;
  numeric t,tt;
  conf=fullcircle scaled (2*marque_a) shifted (aa intersectionpoint bb);
  numeric te;
  te=angle((conf intersectionpoint aa)-(aa intersectionpoint bb));
  rr=(conf intersectionpoint aa){dir(90+angle((conf intersectionpoint aa)-
(aa intersectionpoint bb)))}..(conf intersectionpoint bb);
  t=length rr/2;
  w=point(t) of rr;
  tangent=unitvector(direction t of rr);
  $=image(
    trace rr;
    if mark=1:
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90);
    elseif mark=2:
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted tangent;
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(-tangent);
```

```
    elseif mark=3:
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90);
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(1.5*tangent);
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(-1.5*tangent);
    elseif mark=4:
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,45);
      trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,-45);
    fi;
    );
  $
enddef;

vardef marqueangle(expr aa,bb,cc,mark)=%codage d'un angle de sommet bb
% dans le sens direct par la marque mark.
  save $;
  picture $;
  path conf,rr;
  pair w,tangent;
  numeric t;
  if typetrace="mainlevee":
    conf=fullcircle scaled (2*marque_a) shifted bb;
    rr=(conf intersectionpoint demidroite(bb,aa)){dir(90+angle(aa-bb))}..
(conf intersectionpoint demidroite(bb,cc));
    w=rr intersectionpoint droite(bb,CentreCercleI(aa,bb,cc));
    t=length rr/2;
    tangent=unitvector(direction t of rr);
    $=image(
      trace rr;
      if mark=1:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90);
      elseif mark=2:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted tangent;
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(-tangent);
      elseif mark=3:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90);
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(1.5*tangent);
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(-1.5*tangent);
      elseif mark=4:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,45);
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,-45);
      fi;
      );
  else:
    rr=arccercle(bb+marque_a*unitvector(aa-bb),bb+marque_a*unitvector(cc-bb),bb);
    w=rr intersectionpoint droite(bb,CentreCercleI(aa,bb,cc));
    t=length rr/2;
```

```
    tangent=unitvector(direction t of rr);
    $=image(
      trace rr;
      if mark=1:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90);
      elseif mark=2:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted tangent;
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(-tangent);
      elseif mark=3:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90);
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(1.5*tangent);
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,90)
 shifted(-1.5*tangent);
      elseif mark=4:
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,45);
        trace rotation((w shifted(5*tangent))--(w shifted(-5*tangent)),w,-45);
      fi;
      );
  fi;
  $
enddef;

vardef coloreangle(expr aa,bb,cc)=arccercle(bb+marque_a*unitvector(aa-bb),
bb+marque_a*unitvector(cc-bb),bb)--bb--cycle
enddef;

vardef marque_para(expr dd,ee,pa)=
  save im;
  picture im;
  pair kk,ll,mn,mo;
  kk=point(pa*length dd) of dd;
  ll=projection(kk,point(0.25*length ee) of ee,point(0.5*length ee) of ee);
  mn=iso(kk,ll);
  mo=(mn--kk) intersectionpoint cercles(mn,3mm);
  im=image(
    drawarrow mo--kk;
    drawarrow symetrie(mo,mn)--ll;
    label(btex $//$ etex,mn);
    );
  im
enddef;

%-----------------------------------------------
% Points
%-----------------------------------------------
%JMS
vardef iso(text t) =
  save s,n; numeric n; pair s; s := (0,0) ; n := 0;
  for p_ = t: s := s + p_; n := n + 1 ; endfor;
```

```
  if n>0: (1/n)*s fi
enddef;

vardef milieu(expr AA,BB)=
  save $;
  pair $;
  if typetrace="mainlevee":
    $=point((length segment(AA,BB))*(1/2+(-1+uniformdeviate(2))/10)) of
 segment(AA,BB)
  else:
    $=iso(AA,BB)
  fi;
  $
enddef;

% -- projection de m sur (a,b)
vardef projection(expr m,a,b) =
    save h; pair h;
    h - m = whatever * (b-a) rotated 90;
    h = whatever [a,b];
    if typetrace="mainlevee":
      h:=h shifted((-2+uniformdeviate(4))*unitvector(a-b))
    fi;
    h
enddef;
% -- centre du cercle circonscrit
vardef CentreCercleC(expr a, b ,c) =
    save o; pair o;
    o - .5[a,b] = whatever * (b-a) rotated 90;
    o - .5[b,c] = whatever * (c-b) rotated 90;
    o
enddef;
% -- orthocentre
vardef Orthocentre(expr a, b, c) =
  save h; pair h;
  h - a = whatever * (c-b) rotated 90;
  h - b = whatever * (a-c) rotated 90;
  h
enddef;
%fin JMS
vardef CentreCercleI(expr aa,bb,cc)=
  save $,a,c;
  pair $;
  numeric a,c;
  a=(angle(aa-cc)-angle(bb-cc))/2;
  c=(angle(cc-bb)-angle(aa-bb))/2;
  ($-cc) rotated a shifted cc=whatever[aa,cc];
  ($-bb) rotated c shifted bb=whatever[bb,cc];
  $
enddef;
%-----------------------------------------------
% Cercles
```

```
%-----------------------------------------------
%Cercle connaissant le centre A et le rayon q
vardef cercle(expr aa, q)=fullcircle scaled (2*q) shifted aa
enddef;
%Cercle de centre A et passant par B
vardef cerclepoint(expr aa,bb)=fullcircle scaled (2*abs(aa-bb)) shifted aa
enddef;
%Cercle connaissant le diamètre [AB]
vardef cercledia(expr aa,bb)=cercles(iso(aa,bb),bb)
  %fullcircle scaled (2*abs(1/2[aa,bb]-bb)) shifted (1/2[aa,bb])
enddef;
%Cercles complets
vardef cercles(text t)=
  save $;
  save n;
  n:=0;
  for p_=t:
    if pair p_:
      n:=n+1;
      _t[n]:=p_;
    fi
    if numeric p_:
      rayon:=p_;
    fi;
  endfor;
  if typetrace="mainlevee":
    path $;
    path dep;
    pair ct;
    if n=1 : dep=fullcircle scaled (2*rayon) shifted _t[1];
      ct=_t[1];
    elseif n=2 : dep=fullcircle scaled (2*abs(_t[1]-_t[2])) shifted _t[1];
      ct=_t[1];
    elseif n=3 :
      dep=fullcircle scaled (2*abs(CentreCercleC(_t[1],_t[2],_t[3])-_t[1]))
 shifted CentreCercleC(_t[1],_t[2],_t[3]);
      ct=CentreCercleC(_t[1],_t[2],_t[3]);
    fi
    path ess;
    ess=pointarc(dep,0){dir(80+uniformdeviate(20))}..
pointarc(dep,90){dir(170+uniformdeviate(20))}
..pointarc(dep,90){dir(185+uniformdeviate(10))}..
pointarc(dep,180){dir(-90+uniformdeviate(10))}
..pointarc(dep,180){dir(-95+uniformdeviate(10))}..
pointarc(dep,270){dir(-10+uniformdeviate(10))}
..pointarc(dep,270){dir(uniformdeviate(10))}..
pointarc(dep,360){dir(80+uniformdeviate(10))};
    $=ess;
  else:
    path $;
    if n=1 : $=fullcircle scaled (2*rayon) shifted _t[1];
    elseif n=2 : $=fullcircle scaled (2*abs(_t[1]-_t[2])) shifted _t[1];
```

```
    elseif n=3 : $=cercles(CentreCercleC(_t[1],_t[2],_t[3]),_t[1]);
    fi
  fi
  $
enddef;

%Point particulier sur le cercle
vardef pointarc(expr cercla,angle)=
  point(arctime((angle/360)*arclength cercla) of cercla) of cercla
enddef;
%Arc de cercle AB de centre 0(dans le sens direct) : les points A et B
% doivent être sur le cercle.
vardef arccercle(expr aa,bb,oo)=
  path tempo;
  path arc;
  tempo=fullcircle scaled (2*abs(aa-oo)) shifted oo;
  if (angle(aa-oo)=0) or (angle(aa-oo)>0) :
    if (angle(bb-oo)=0) or (angle(bb-oo)>0):
      if (angle(aa-oo)<angle(bb-oo)):
        arc=subpath(angle(aa-oo)*(length tempo)/360,
angle(bb-oo)*(length tempo)/360) of tempo;
      else:
        arc=subpath(angle(aa-oo)*(length tempo)/360,
(length tempo)+angle(bb-oo)*(length tempo)/360) of tempo;
      fi;
    elseif (angle(bb-oo)<0):
      arc=subpath(angle(aa-oo)*(length tempo)/360,
(length tempo)+angle(bb-oo)*(length tempo)/360) of tempo;
    fi;
  elseif (angle(aa-oo)<0):
    if (angle(bb-oo)=0) or (angle(bb-oo)>0):
      arc=subpath(length tempo+angle(aa-oo)*(length tempo)/360,
length tempo+angle(bb-oo)*(length tempo)/360) of tempo;
    elseif (angle(bb-oo)<0):
      if (angle(aa-oo)=angle(bb-oo)) or (angle(aa-oo)<angle(bb-oo)):
        arc=subpath((length tempo)+angle(aa-oo)*(length tempo)/360,
(length tempo)+angle(bb-oo)*(length tempo)/360) of tempo;
      else:
        arc=subpath((length tempo)+angle(aa-oo)*(length tempo)/360,
2*(length tempo)+angle(bb-oo)*(length tempo)/360) of tempo;
      fi;
    fi;
  fi;
  arc
enddef;

vardef coupdecompas(expr ab,ac,ad)=arccercle(pointarc(cercles(ab,ac),
angle(ac-ab)-ad),pointarc(cercles(ab,ac),angle(ac-ab)+ad),ab)
enddef;

%-----------------------------------------------
% Droites
```

```
%-----------------------------------------------
vardef segment(expr aa,bb)=
  save $;
  path $;
  if typetrace="mainlevee":
    $=aa{dir(angle(bb-aa)+5)}..bb{dir(angle(bb-aa)+5)}
  else:
    $=aa--bb
  fi;
  $
enddef;

vardef droite(expr AA,BB)=
  save $;
  path $;
  if typetrace="mainlevee":
    $=(_tfig/abs(AA-BB))[BB,AA]{dir(angle(BB-AA)+10)}..segment(AA,BB)..
(_tfig/abs(AA-BB))[AA,BB]{dir(angle(BB-AA)+10)}
  else:
    $=(_tfig/abs(AA-BB))[BB,AA]--(_tfig/abs(AA-BB))[AA,BB]
  fi;
  $
enddef;
vardef demidroite(expr AA,BB)=
  save $;
  path $;
  if typetrace="mainlevee":
    $=segment(AA,BB)..(_tfig/abs(AA-BB))[AA,BB]{dir(angle(BB-AA)+10)}
  else:
    $=AA--(_tfig/abs(AA-BB))[AA,BB]
  fi;
  $
enddef;

vardef bissectrice(expr AA,BB,CC)=
  save $;
  path $;
  if typetrace="mainlevee":
    $=rotation(demidroite(BB,CentreCercleI(AA,BB,CC)),BB,-5+uniformdeviate(10))
  else:
    $=demidroite(BB,CentreCercleI(AA,BB,CC))
  fi;
  $
enddef;

vardef mediatrice(expr AA,BB)=droite(iso(AA,BB),rotation(BB,iso(AA,BB),90))
enddef;
%main levee : passer par la perpendiculaire passant par le milieu.
vardef perpendiculaire(expr AA,BB,II)=droite(iso(AA,BB),rotation(BB,
iso(AA,BB),90)) shifted (II-iso(AA,BB))
enddef;
vardef parallele(expr AA,BB,II)=droite(AA,BB) shifted (II-
```

```
(projection(II,AA,BB)))
enddef;
%-----------------------------------------------
% Transformations
%-----------------------------------------------
vardef rotation(expr p,c,a)=
  p rotatedaround(c,a)
enddef;
vardef symetrie(expr x)(text t)=
  save n;
  n:=0;
  for p_=t:
    n:=n+1;
    _t[n]:=p_;
  endfor;
  if n=1:
    rotation(x,_t[1],180)
  elseif n=2:
    x reflectedabout(_t[1],_t[2])
  fi
enddef;
%-----------------------------------------------
%Sucres
%-----------------------------------------------
vardef hachurage(expr chemin, angle, ecart, trace)=
  save $;
  picture $;
  path support;
  support=((u*(-37,0))--(u*(37,0))) rotated angle;
  if trace=1:
    drawoptions(dashed evenly);
  elseif trace=2:
    drawoptions(dashed dashpattern(on12bp off6bp on3bp off6bp));
  fi;
  $ = image(
    for j=-200 upto 200:
      if ((support shifted (ecart*j*(u,0))) intersectiontimes chemin)<>(-1,-1):
        draw support shifted (ecart*j*(u,0));
      fi
    endfor;
    );
  clip $ to chemin;
  drawoptions();
  $
enddef;
%flèche pour coter un segment [AB] (Jacques Marot)
vardef cotation(expr aa,bb,ecart,decalage,cote)=
  pair m[] ;
  save $;
  picture $;
  m3=unitvector(bb-aa) rotated 90;
  m1=aa+ecart*m3;
```

```
  m2=bb+ecart*m3;
  $=image(
    pickup pencircle scaled 0.2bp;
    drawdblarrow m1--m2 ;
    draw aa--m1 dashed evenly;
    draw bb--m2 dashed evenly;
    label(cote rotated angle(m2-m1),(m1+m2)/2+decalage*m3);
    );
  $
enddef;

vardef appelation(expr aa,bb,decalage,cote)=
  save $;
  picture $;
  pair m[];
  m3=unitvector(bb-aa) rotated 90;
  $=image(
    label(cote rotated angle(bb-aa),(bb+aa)/2+decalage*m3);
    );
  $
enddef;

vardef cotationmil(expr aa,bb,ecart,decalage,cote)= %Christophe
  pair m[] ;
  save $;
  picture $;
  m3=unitvector(bb-aa) rotated 90;
  m1=aa+ecart*m3;
  m2=bb+ecart*m3;
  $=image(
    pickup pencircle scaled 0.2bp;
    drawarrow (1/2[m1,m2]+decalage*unitvector(m1-m2))--m1;
    drawarrow (1/2[m1,m2]-decalage*unitvector(m1-m2))--m2;
    draw aa--m1 dashed evenly;
    draw bb--m2 dashed evenly;
    label(cote rotated angle(m2-m1),(m1+m2)/2);
    );
  $
enddef;

vardef cotationmilrap(expr aa,bb,ecart,decalage,cote)= %Christophe
  pair m[] ;
  save $;
  picture $;
  m3=unitvector(bb-aa) rotated 90;
  m1=aa+ecart*m3;
  m2=bb+ecart*m3;
  $=image(
    pickup pencircle scaled 0.2bp;
    draw (1/2[m1,m2]+decalage*unitvector(m1-m2))--9/10[m2,m1];
    draw (1/2[m1,m2]-decalage*unitvector(m1-m2))--m2;
    label(cote rotated angle(m2-m1),(m1+m2)/2);
```

```
    );
  $
enddef;

endinput
```