Documentation sur les fichiers geometriesyr12.mp et christ5.tex

Christophe Poulain

Version 1.0

Table des matières

1 Documentation sur christ5.tex

1.1 Quelques environnements.

Sont définis les environnements numérotés :

Proposition par `\begin{prop}...\end{prop}` pour obtenir

Proposition 1 *...*

Propriété par `\begin{prop}...\end{prop}`

Théorème par `\begin{theo}...\end{theo}`

Définition par `\begin{defi}...\end{defi}`

Lemme par `\begin{lemme}...\end{lemme}`

Corollaire par `\begin{coro}...\end{coro}`

Règle par `\begin{reg}...\end{reg}`

Conjecture par `\begin{conj}...\end{conj}`

Remarque par `\begin{remar}...\end{remar}`

Exemple par `\begin{exem}...\end{exem}`

1.2 Quelques raccourcis « classiques ».

Des commandes pour obtenir des mots particuliers soulignés : `\rema` pour Remarque, `\exe` pour Exemple, `\pre` pour Preuve, `\cas` pour Cas particulier, `\cass` pour Cas particuliers, `\Not` pour Notation, `\Si` pour Si, `\si` pour si, `\alors` pour alors et `\cons` pour Conséquence.

1.3 Quelques commandes de présentation.

encadre{*texte à encadrer*} L'encadrement se fait sur toute la largeur de la page (s'adapte au mode un colonne/deux colonnes).

titrage{*titre*}{*remarque*} Permet d'obtenir

Développements 3e

partie{*longueur du trait*}{*texte*} Permet d'obtenir[1]

Activités numériques.

exo pour obtenir une numérotation automatique des exercices. Se remet automatiquement à 1 à chaque changement de section.

myenumerate Cet environnement permet d'obtenir une liste numérotée du type suivant

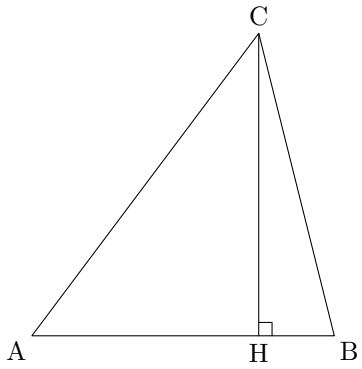
1/

2/

sur un niveau seulement.

compo{*n° de l'image*}{*nom de l'image*}{*échelle de l'image*}{*texte à positionner*} Permet de placer du texte du côté droit d'une image metapost du type « geometrie.1 ». Pour les numéros d'images supérieurs ou égaux à 10, on utilisera la variante `Compo`. **Nécessite le package graphicx**.

[1] Nécessite les packages FANCYBOX et COLOR.



Soit un triangle ABC. La perpendiculaire à la droite (AB) passant par H coupe la droite (AB) en H.

1. Exprime l'aire du triangle AHC.
2. Exprime l'aire du triangle ABC.

```
\compo{1}{docfichierschrist}{1}{Soit un triangle $ABC$. La
perpendiculaire à la droite $(AB)$ passant par $H$ coupe la droite
$(AB)$ en $H$.
\begin{enumerate}
\item Exprime l'aire du triangle $AHC$.
\item Exprime l'aire du triangle $ABC$.
\end{enumerate}}
```

1.4 Quelques commandes mathématiques.

cqfd Permet d'obtenir □

qed Même chose que `\cqfd`.

vecteur{*nom du vecteur*} permet d'obtenir $\overrightarrow{AB}$.

Eqalign Cette commande, utilisée en mode mathématique, permet d'aligner de nombreuses équations, expressions,...Avec le codage suivant,

```
$$\Eqalign{
A&=\frac{1}{3}+\frac{5}{4}\div\frac{2}{7}\kern1cm&B&=(x+2)^2-(x-3)^2\cr
\cr
C&=\sqrt{124}-\sqrt{56}\cr
&&E&=10^{12}\times10^{-8}\cr
}$$
```

on obtient

$$A = \frac{1}{3} + \frac{5}{4} \div \frac{2}{7} \qquad B = (x+2)^2 - (x-3)^2$$

$$C = \sqrt{124} - \sqrt{56} \qquad E = 10^{12} \times 10^{-8}$$

2 Documentation sur geometriesyr12.mp

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.
- PAPIERS1.MP qui permet de produire[2] différents types de papiers (millimétré, 5×5, isométrique,...)

2.1 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.

2.2 Affichage.

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

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

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

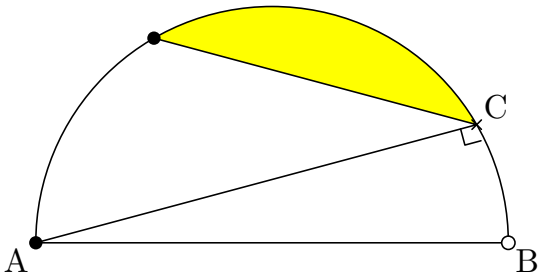
croix : dans ce cas, on pointe avec une croix.

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[3]

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

2.3 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.

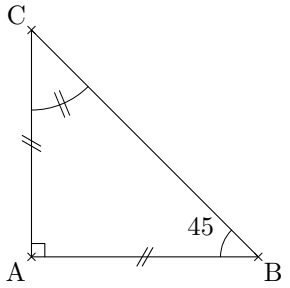
longueurs égales : `codesegments(A,B,C,D,n)` code les segments $[AB]$ et $[CD]$ avec le codage d'ordre n : 1, 2, 3 pour autant de traits, 4 pour la croix et 5 pour le cercle.

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.

[2] Voir la page 9 pour connaître les commandes disponibles.

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

Marquage d'un angle : `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.




```
figure(0,0,5u,5u);
pair A,B,C;
A=u*(1,1);
B=u*(4,1);
C=rotation(B,A,90);
trace triangle(A,B,C);
trace codeperp(B,A,C,5);
trace codesegments(B,A,A,C,2);
trace codeangle.ulft(C,B,A,1,5mm,btex 45° etex);
trace marqueangle(A,C,B,2);
nomme.llft(A);
nomme.lrt(B);
nomme.ulft(C);
fin;
```

2.4 Points.

Quelques macros pour définir des points particuliers simplement :

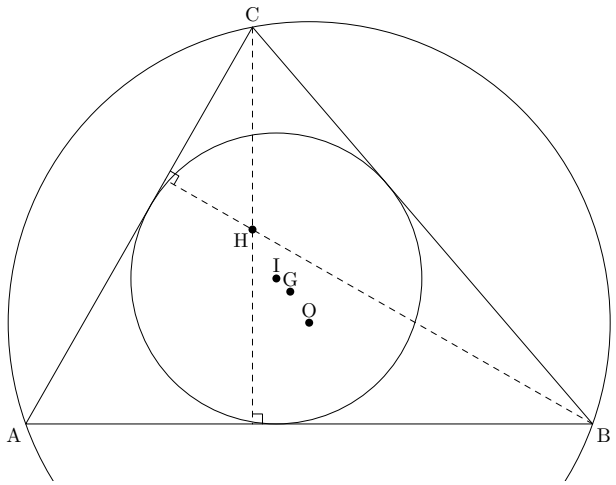
iso(A,B,C,...) construit l'isobarycentre des points A, B, C, ...

projection(M,A,B) construit le projeté orthogonal de M sur la droite (AB).

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;
```

2.5 Cercles.

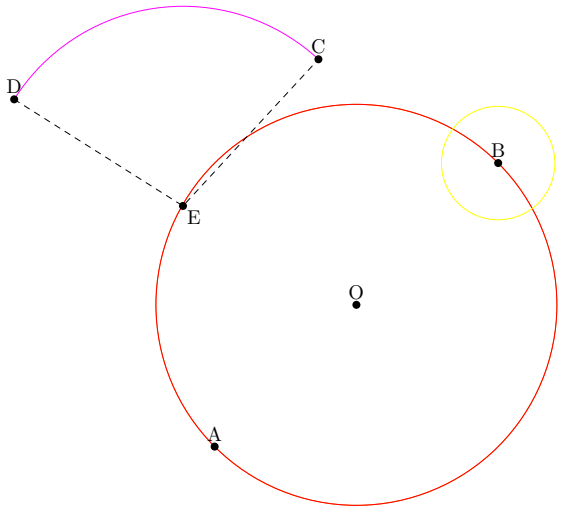
cercledia(A,B) construit le cercle de diamètre $[AB]$.

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.




```
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;
```

2.6 Droites.

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

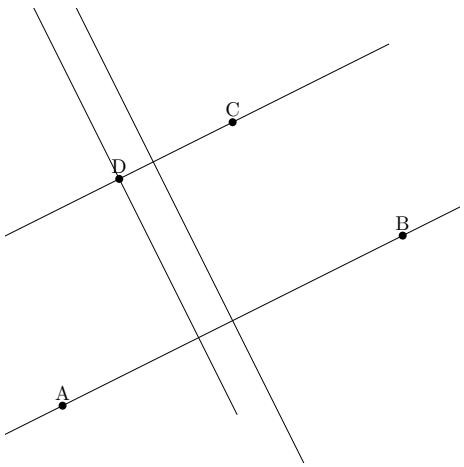
demidroite(A,B) construit la demi-droite $[AB)$;

mediatrice(A,B) construit la médiatrice du segment $[AB]$;

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;
```

2.7 Transformations.

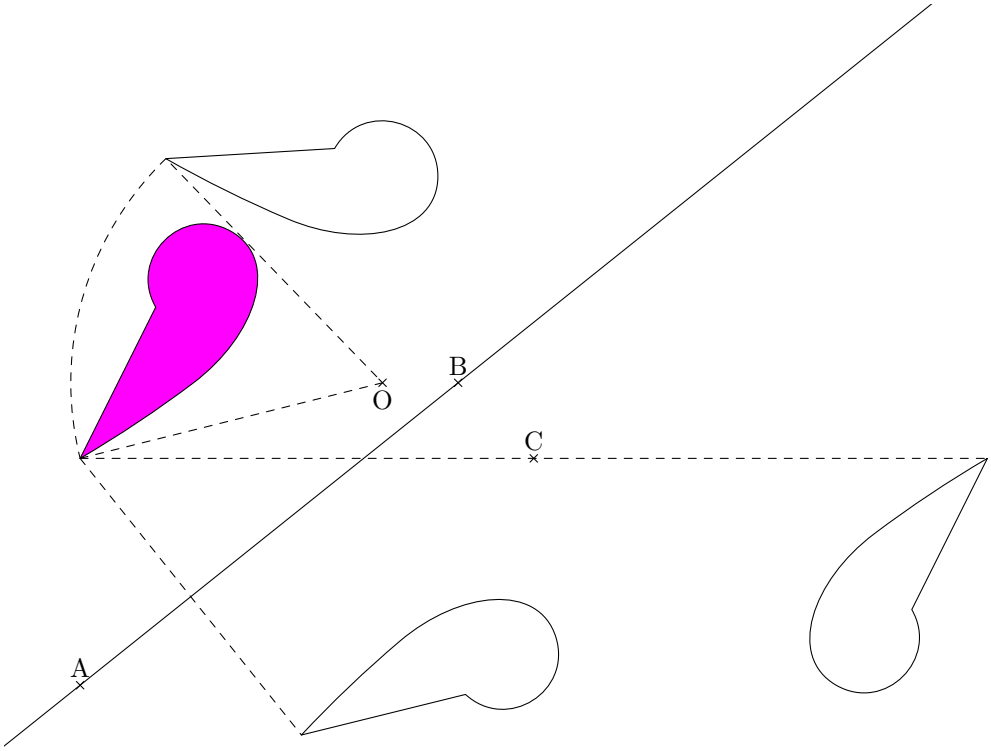
Dans ce qui suit, *objet* représente un objet connu de MetaPost (un point, un chemin, une figure,...)

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;
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;
```

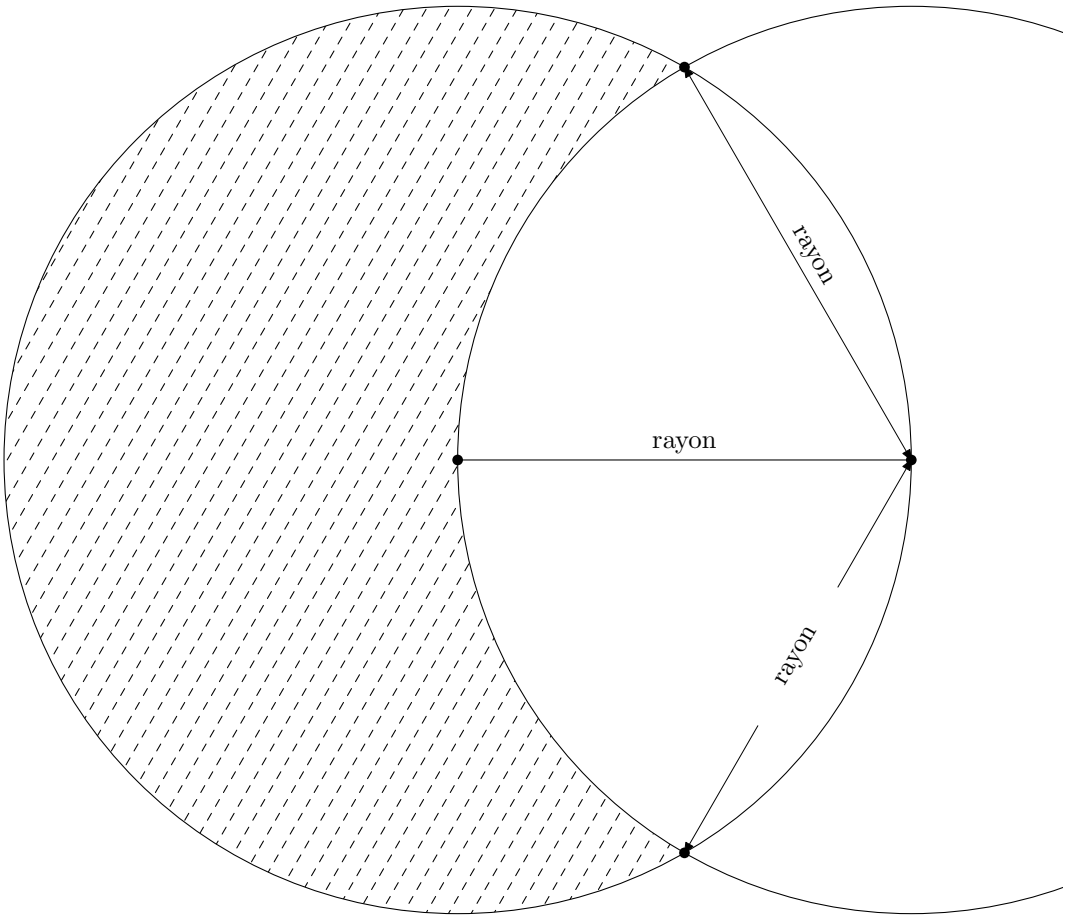
2.8 Sucres

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;
```

3 Documentation sur papiers1.mp

[à suivre]

A Fichier christ5.tex

```
%=======================================
%Macros personnelles
%christophe.poulain@melusine.eu.org
%création : 25 Septembre 1999
%dernière modification : 07 Avril 2003
%=======================================
\newtheorem{ppte}{Propriété}
\newtheorem{theo}{Théorème}
\newtheorem{defi}{Définition}
\newtheorem{lemme}{Lemme}
\newtheorem{coro}{Corollaire}
\newtheorem{prop}{Proposition}
\newtheorem{reg}{Règle}
\newtheorem{conj}{Conjecture}
\newtheorem{remar}{Remarque}
\newtheorem{exem}{Exemple}

\newcommand{\rema}{\underline{Remarque} }
\newcommand{\exe}{\underline{Exemple} }
\newcommand{\pre}{\underline{Preuve}}
\newcommand{\cas}{\underline{Cas particulier}}
\newcommand{\cass}{\underline{Cas particuliers}}
\newcommand{\Not}{\underline{Notation} }
\newcommand{\Si}{\underline{Si} }
\newcommand{\si}{\underline{si} }
\newcommand{\alors}{\underline{alors} }
\newcommand{\cons}{\underline{Conséquence}}

\def\qed{\hfill\raise -2pt\hbox{\vrule\vbox to 10pt{\hrule width4pt\vfill\hrule}\vrule}}
\def\cqfd{\hfill\unskip\kern 6pt\penalty 500\qed\par}

\catcode`\@=11
\def\Eqalign#1{\null\,\vcenter{\openup\jot\m@th\ialign{
\strut\hfil$\displaystyle{##}$&$\displaystyle{{}##}$\hfil
&&\quad\strut\hfil$\displaystyle{##}$&$\displaystyle{{}##}$
\hfil\crcr #1\crcr}}\,}
\catcode`\@=12

\newcommand{\vecteur}[1]
{\overrightarrow{#1}}

\font\tenbb=msbm10
\font\sevenbb=msbm7
\font\fivebb=msbm5
\newfam\bbfam
\textfont\bbfam=\tenbb
\scriptfont\bbfam=\sevenbb
\scriptscriptfont\bbfam=\fivebb
\def\bb{\fam\bbfam\tenbb}
\let\oldbb=\bb
\def\bb #1{{\oldbb #1}}

\def\tvi{\vrule height 12pt depth 5pt width 0pt}
\def\tvj{\vrule height 12pt depth 5pt width 1pt}
\def\hfq{\hfill\,\,}
\def\cc#1{\hfq #1\hfq}
```

```
\def\tv{\tvi\vrule}
\def\tw{\tvj\vrule}
\def\traithorizontal{\noalign{\hrule}}
\def\traithorizontale{\noalign{\hrule height 1pt}}

\newcommand{\encadre}[1]
{\begin{center}
\fbox{\begin{minipage}{\linewidth}
{#1}
\end{minipage}}
\end{center}
}

\def\pgcd{\mathop{\rm pgcd}\nolimits}
\def\ppcm{\mathop{\rm ppcm}\nolimits}

\def\cut{{}\hfill\cr \hfill{}}

\newcommand{\biindice}[3]%
{
\renewcommand{\arraystretch}{0.5}
\begin{array}[t]{c}
#1\\
{\scriptstyle #2}\\
{\scriptstyle #3}
\end{array}
\renewcommand{\arraystretch}{1}
}

\newlength{\ltxt}
\newcommand{\compo}[4]{
\setlength{\ltxt}{\linewidth}
\setbox#1=\hbox{\includegraphics[scale=#3]{#2.#1}}
\addtolength{\ltxt}{-\wd#1}
\addtolength{\ltxt}{-10pt}
\begin{minipage}{\wd#1}
\includegraphics[scale=#3]{#2.#1}
\end{minipage}
\hfill
\begin{minipage}{\ltxt}
#4
\end{minipage}
}

\newlength{\lntxt}
\newcommand{\Compo}[4]{
\setlength{\lntxt}{\linewidth}
\setbox#1=\hbox{\includegraphics[scale=#3]{#2}}
\addtolength{\lntxt}{-\wd#1}
\addtolength{\lntxt}{-10pt}
\begin{minipage}{\wd#1}
\includegraphics[scale=#3]{#2}
\end{minipage}
\hfill
\begin{minipage}{\lntxt}
#4
\end{minipage}
```

```
}

\newlength{\lnttxt}
\newcommand{\dispo}[3]{
\setlength{\lnttxt}{\linewidth}
\setbox#1=\hbox{#2}
\addtolength{\lnttxt}{-\wd#1}
\addtolength{\lnttxt}{-20pt}
\begin{minipage}{\wd#1}
#2
\end{minipage}
\hfill
\begin{minipage}{\lnttxt}
#3
\end{minipage}
}

\newcounter{num}[section]
\newcommand{\exo}{\addtocounter{num}{1}
\par
\par\underline{\bf Exercice~\thenum} }

\newcommand{\titrage}[2]{
{\Large #1}\hfill#2
\par\rule[+6pt]{\linewidth}{0.5mm}
\par
}

\newcommand{\titragedossier}[1]{
{\small #1}\hfill{\small www.melusine.eu.org/syracuse/poulecl/}
\par\rule[+6pt]{\linewidth}{0.5mm}
\par
}

\newcommand{\partie}[2]{
\begin{center}
\begin{minipage}{#1pt}
\begin{center}
\boxput*(0,0){\colorbox{white}{#2}}
{\rule{\linewidth}{0.5mm}}
\end{center}
\end{minipage}
\end{center}
\par
}

\newenvironment{myenumerate}{
\renewcommand{\theenumi}{\arabic{enumi}}
\def\labelenumi{{\bf \theenumi /}}
\begin{enumerate}}{\end{enumerate}}
```

B Fichier geometriesyr11.mp

```
%%=============================================
%% GEOMETRIESYR.MP
%% christophe.poulain@melusine.eu.org
%% Création : 19 Février 2003
```

```
%% Dernière modification : 22 Novembre 2003
%%================================================
%------------------------------------------------
% Appel fichier
%------------------------------------------------
input constantes;
input papiers1;
%------------------------------------------------
% La figure (début et fin) JMS/CP
%------------------------------------------------
path feuillet;
numeric _tfig,_nfig;
pair coinbg,coinbd,coinhd,coinhg;
_nfig:=0;
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 fin =
    endfig;
enddef;
%%------------------------------------------------
%% Les marques (JMS)
%%------------------------------------------------
string marque_p;
marque_p := "non";
marque_r := 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/20,u/20))--(p shifted (u/20,-u/20));
    draw (p shifted (-u/20,-u/20))--(p shifted (u/20,u/20));
  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)=aa--bb--cc--cycle
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 $;
  path $;
  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(
  else :
    pointchoisi3:=droite(pointchoisi2,rotation(pointchoisi1,pointchoisi2,-choix)) intersectionpoint droite
  fi
  j:=1;
  for p_=t:
    p_=pointchoisi[j];
    j:=j+1;
  endfor;
  $=pointchoisi1--pointchoisi2--pointchoisi3--cycle;
  $
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,ecart,nom)=
  save s,p,$;
  path p;
  picture $;
```

```
  $=image(
    pickup pencircle scaled 0.25bp;
    for j=0 upto (nb-1) :
      draw arccercle(((ecart+j*mm)*unitvector(aa-bb) shifted bb),((ecart+j*mm)*unitvector(cc-bb) shifted b
    endfor;
    label.@#(nom,iso((ecart+nb*mm)*unitvector(aa-bb) shifted bb,(ecart+nb*mm)*unitvector(cc-bb) shifted bb
    );
  $
enddef;

vardef marqueangle(expr aa,bb,cc,mark)=%codage d'un angle de sommet bb dans le sens direct par la marque m
  save $;
  picture $;
  path rr;
  pair w;
  pair tangent;
  numeric t;
  rr=arccercle(bb+30*unitvector(aa-bb),bb+30*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;
    );
  $
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;
% -- 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];
    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)=
  fullcircle scaled (2*abs(1/2[aa,bb]-bb)) shifted (1/2[aa,bb])
enddef;
%Cercles complets
vardef cercles(text t)=
  save $;
  path $;
  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 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
  $
  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=cercle(oo,abs(aa-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 tem
      fi;
    else :
      if (angle(aa-oo)=angle(bb-oo)) or (angle(aa-oo)>angle(bb-oo)):
        arc=subpath(angle(aa-oo)*(length tempo)/360,(length tempo)+angle(bb-oo)*(length tempo)/360) of tem
      fi;
    fi;
  else:
    if (angle(bb-oo)<0):
      if (angle(aa-oo)<=angle(bb-oo)):
        arc=subpath((length tempo)+angle(aa-oo)*(length tempo)/360,(length tempo)+angle(bb-oo)*(length tem
      else:
        arc=subpath((length tempo)+angle(aa-oo)*(length tempo)/360,2*(length tempo)+angle(bb-oo)*(length t
      fi;
    else :
      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 tem
      fi;
    fi;
  fi
  arc
enddef;
%-----------------------------------------------
% Droites
%-----------------------------------------------
vardef droite(expr AA,BB)=(_tfig/abs(AA-BB))[BB,AA]--(_tfig/abs(AA-BB))[AA,BB]
enddef;
vardef demidroite(expr AA,BB)=AA--(_tfig/abs(AA-BB))[AA,BB]
enddef;
vardef mediatrice(expr AA,BB)=droite(iso(AA,BB),rotation(BB,iso(AA,BB),90))
enddef;
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;

endinput
```