

Les commandes graphiques en L^AT_EX 2_ε

Michel GOOSSENS^a et Michèle JOUHET^b

^a*CERN, Division CN*

CH-1211 Genève 23, Suisse

Michel.Goossens@cern.ch

^b*CERN, Division AS*

CH-1211 Genève 23, Suisse

Michele.Jouhet@cern.ch

Résumé. Les opérations géométriques, comme les variations d'échelle, les rotations, ou l'insertion d'images graphiques externes ne font pas partie des fonctions de base de L^AT_EX 2_ε. C'est l'extension standard `graphics` (ou sa version généralisée `graphicx`) qui à travers une interface utilisateur unique introduit ces fonctions d'une façon transparente en se basant sur les commandes internes des pilotes d'écran et d'imprimante¹. Avec L^AT_EX 2_ε et la commande `\qbezier` on peut maintenant mieux contrôler les paramètres des courbes de Bézier à l'intérieur d'un environnement `picture`.

Abstract. L^AT_EX 2_ε provides a generalized driver-independent interface for the inclusion of external graphic material, as well as for scaling and rotation of L^AT_EX boxes. These features are not in the L^AT_EX 2_ε kernel, but are implemented in the `graphics` or `graphicx` extension packages. These packages rely on features that are not in T_EX itself, but which must be supplied by the “driver” used to print the dvi file². Inside a `picture` environment L^AT_EX 2_ε also offer better support to draw Bezier curves with the `\qbezier` command.

1. L'extension `graphics`

L'extension `graphics` (et sa généralisation `graphicx`) permet un changement d'échelle, une rotation ou une réflexion pour un élément L^AT_EX. Elle permet également d'inclure des graphiques créés par d'autres logiciels. L^AT_EX calculera la boîte qu'occupera l'élément après l'opération ou l'insertion et laissera l'espace libre pour permettre la visualisation correcte par un pilote, comme `dvips`, `xdvi`, etc. Si toutefois le pilote ne dispose pas de la fonctionnalité nécessaire pour réaliser l'opération en question, l'élément peut être montré chevauchant le texte environnant, mais la pagination globale sera quand-même correcte.

1. Malheureusement tous les pilotes n'offrent pas de fonctions identiques, et même la méthode interne d'accès à ces extensions varie entre les différents pilotes. Par conséquent, on doit spécifier le pilote désiré comme *option* avec la commande `\usepackage` ou `\documentclass`, par exemple `\usepackage[dvips]{graphics}` si l'on choisit le pilote “`dvips`”

2. Unfortunately not all drivers support the same features, and even the internal method of accessing these extensions varies between drivers. Consequently all these packages take options such as “`dvips`” to specify which driver is being used.

1.1. Modifier l'échelle d'une boîte L^AT_EX

La commande `\scalebox` permet l'agrandissement ou la réduction d'un texte ou de tout autre élément L^AT_EX par un facteur de graduation.

```
\scalebox{facteur-d'échelle}{contenu}
```

Lorsque la commande contient deux arguments (obligatoires), le premier (*facteur-d'échelle*) spécifie le facteur par lequel les deux dimensions de la boîte L^AT_EX (*contenu*) doivent être redimensionnées.

Considérons l'exemple suivant :

```
\scalebox{2}{Le texte change de dimensions} \\
      Le texte change de dimensions \\
\scalebox{0.5}{Le texte change de dimensions}
```

Le texte change de dimensions

Le texte change de dimensions

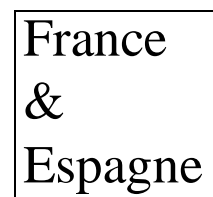
Le texte change de dimensions

Si un argument optionnel supplémentaire est présent, il sera utilisé pour spécifier un facteur d'échelle verticale différent.

```
\scalebox{échelle-h} [échelle-v] {contenu}
```

L'effet du paramètre optionnel est visible dans les exemples suivants, qui montrent comment des lignes multiples peuvent être dimensionnées en utilisant la commande L^AT_EX `\parbox`.

```
\framebox{\scalebox{2}{%
\parbox{.5in}{France \&\Espagne}}}
```



```
\framebox{\scalebox{2} [.5]{%
\parbox{.5in}{France \&\Espagne}}}
```



```
\reflectbox{contenu}
```

`\reflectbox` est une abréviation pour `\scalebox{-1}[1]{contents}`.

France ?\reflectbox{France ?}

France ?εαρντF

1.2. Redimensionner à une taille définie

Il est aussi possible de spécifier que le matériel L^AT_EX soit édité dans des dimensions horizontales et verticales fixes.

```
\resizebox*{dim-h}{dim-v}{boîte LATEX}
```

Quand les proportions du graphique doivent être maintenues, il suffit de spécifier une des dimensions en remplaçant l'autre par le signe « ! ».

```
\framebox{\resizebox{5mm}{!}{\parbox{14mm}%
  {Londres, \\\Berlin \&\\Paris}}}
```



```
\framebox{\resizebox{!}{10mm}{\parbox{14mm}%
  {Londres, \\\Berlin \&\\Paris}}}
```



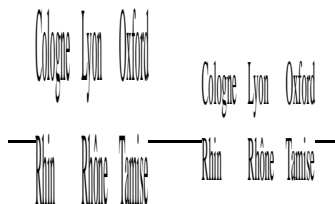
Si les deux dimensions *dim-h* et *dim-v* sont définies explicitement, alors le résultat peut être une déformation de l'élément original. Dans l'exemple ci-dessous la ligne de référence sera indiquée par une ligne horizontale créée par la commande \HR.

```
\newcommand{\HR}{\rule{1em}{0.4pt}}
\HR\begin{tabular}{@{}lll@{}}
  Cologne & Lyon & Oxford \\\ Rhin & & Rhône & Tamise
\end{tabular}\HR
\HR\resizebox{2cm}{.5cm}{%
  \begin{tabular}{@{}lll@{}}
    Cologne & Lyon & Oxford \\\ Rhin & & Rhône & Tamise
  \end{tabular}}\HR
```



La forme non étoilée de la commande \resizebox base ses calculs sur la hauteur des données L^AT_EX, tandis que la forme étoilée \resizebox* prend en compte la hauteur totale (profondeur + hauteur) de la boîte L^AT_EX. Les exemples suivants de \parbox qui ont une profondeur importante montrent la différence.

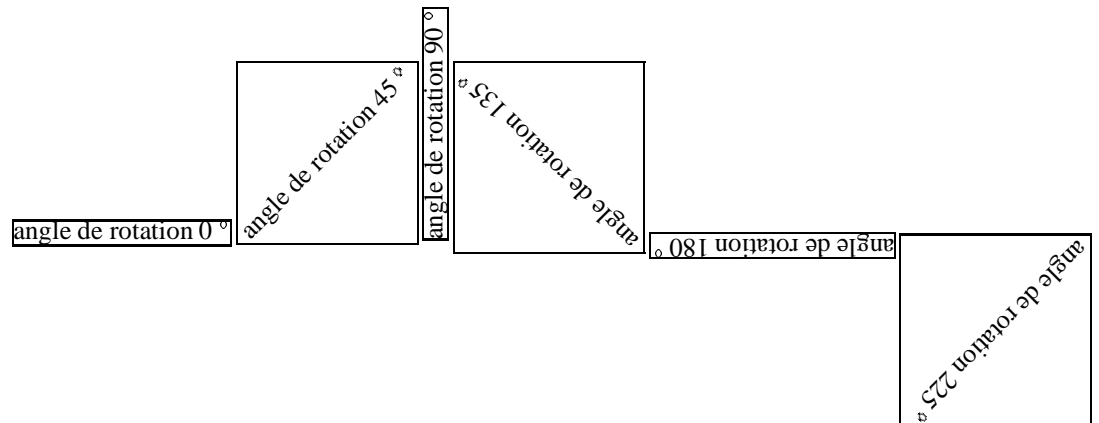
```
\HR\resizebox{15mm}{20mm}{%
  \begin{tabular}{@{}lll@{}}
    Cologne & Lyon & Oxford \\\
    Rhin & & Rhône & Tamise
  \end{tabular}}\HR
\HR\resizebox*{15mm}{20mm}{%
  \begin{tabular}{@{}lll@{}}
    Cologne & Lyon & Oxford \\\
    Rhin & & Rhône & Tamise
  \end{tabular}}\HR
```



1.3. Rotation d'une boîte L^AT_EX

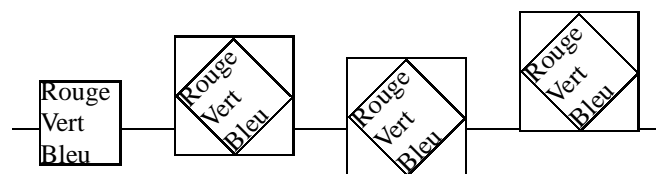
```
\rotatebox{angle}{contenu}
```

Le matériel L^AT_EX peut pivoter suivant un angle avec la commande `\rotatebox`. La «boîte» L^AT_EX 2_ε spécifiée en deuxième argument (*contenu*) sera tournée suivant un angle donné, *angle*, dans le sens inverse des aiguilles d'une horloge autour d'un point de référence. L^AT_EX laissera la place nécessaire au bon positionnement de la «boîte» résultante.



Il est important de réaliser où se trouve le point de référence ; l'exemple suivant montre qu'il peut être contrôlé en se servant du paramètre de positionnement de la commande `\parbox`.

```
\HR\framebox{\parbox{3em}{Rouge\Vert\Bleu}}\HR
\HR\framebox{\rotatebox{45}{\framebox{\parbox{3em}{Rouge\Vert\Bleu}}}}\HR
\HR\framebox{\rotatebox{45}{\framebox{\parbox[t]{3em}{Rouge\Vert\Bleu}}}}\HR
\HR\framebox{\rotatebox{45}{\framebox{\parbox[b]{3em}{Rouge\Vert\Bleu}}}}\HR
```

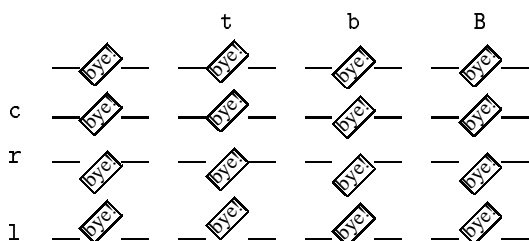
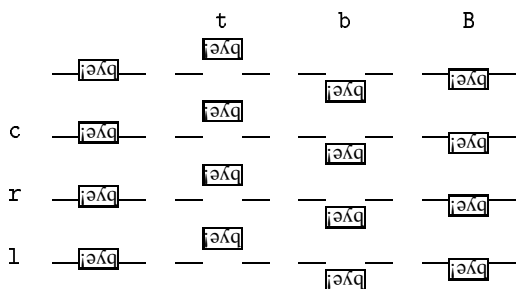


L'extension `graphicx` offre en plus la possibilité de spécifier le point autour duquel la rotation se fera en utilisant des couples *clé-valeurs*.

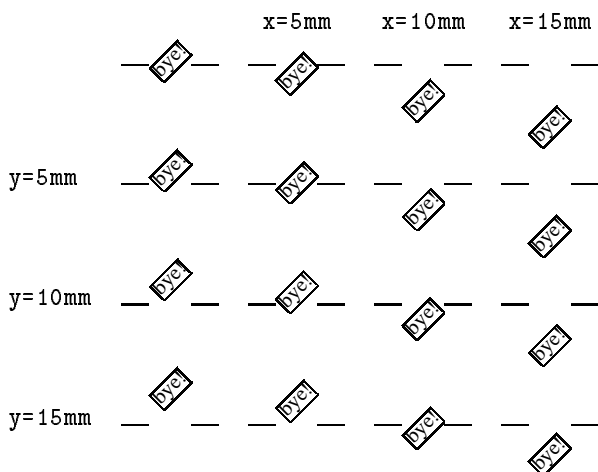
```
\rotatebox[liste clé-valeurs]{angle}{boîte LATEX}
```

Une des clés possible est `origin`, et quant aux valeurs, vous pouvez en choisir une ou deux dans la liste suivante :

<i>horizontal</i>	l	gauche	r	droite	c	centre
<i>vertical</i>	t	haut	b	bas	B	ligne de référence



Vous pouvez également spécifier les coordonnées x et y du point de référence autour duquel s'effectuera la rotation en utilisant la syntaxe : $x = dim$, $y = dim$.



Finalement, l'interprétation de l'argument *angle* de `\rotatebox` peut être contrôlée avec le mot-clé *units* dans l'argument optionnel de `\rotatebox`. Ainsi pour spécifier les angles en radians, on mettra : `units=6.283185`, alors que la direction de la rotation se fera dans le sens des aiguilles d'une montre en déclarant `units=-360`.

1.4. Rotation dans l'environnement tabular

Le contenu de l'environnement `tabular` peut faire l'objet d'une rotation. Les exemples suivants montrent comment la distance entre les colonnes et l'espace vertical de la table peuvent être contrôlés en utilisant des filets invisibles (à largeur ou hauteur zéro).

```
\begin{tabular}{rrr}
\rotatebox{45}{Colonne 1} & & 
\rotatebox{45}{Colonne 2} & & 
\rotatebox{45}{Colonne 3} \\ \hline
1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ \hline
\end{tabular}
```

Colonne 1	Colonne 2	Colonne 3
1	2	3
4	5	6
7	8	9

```
% Filet hauteur zéro, largeur un cadratin
\newcommand{\IR}[1]{\rule{1em}{0pt}}
\makebox[0cm][c]{\rotatebox{45}{\ #1}}
\begin{tabular}{rrr}
\IR{Colonne 1} & \IR{Colonne 2} & \IR{Colonne 3} \\ \hline
1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ \hline
\end{tabular}
```

Colonne 1	Colonne 2	Colonne 3
1	2	3
4	5	6
7	8	9

Une rotation à l'intérieur d'une cellule d'un tableau est également autorisée. Dans l'exemple ci-dessous, notez l'utilisation de la longueur `\depth` (profondeur), qui garantit que la boîte sera alignée correctement avec le reste de la ligne du tableau.

```
\setlength{\extrarowheight}{4pt}
\HR\rotatebox{90}{\begin{tabular}{|l@{\quad}r|} \hline
\em Mot & \raisebox{\depth}{%
\rotatebox{90}{0currences}} \\ \hline
hello & 33 \\ \hline
au revoir & 34 \\ \hline
\end{tabular}}\HR
```

Occurrences	
<i>Mot</i>	
hello	33
aurevoir	34

L'exemple suivant est un peu plus complexe. Ici un tableau complet est tourné par la commande `\rotatebox`. Notez l'utilisation d'une autre commande `rotatebox` qui génère une ligne verticale le long du texte. En précisant une profondeur et une hauteur égales à zéro pour le résultat de la commande `\raisebox`, L^AT_EX considère que la boîte tournée avec le texte «Classes des formats» n'occupe pas de place, et sera donc ignorée dans ses calculs pour composer le tableau.

Dimension de formats papiers ISO (mm)	série A	série B	Série C	
	0	841×1189	1000×1414	917×1297
	1	594×841	707×1000	648×917
	2	420×594	500×707	458×648
	3	297×420	353×500	324×458
	4	210×297	250×353	229×324
	5	148×210	176×250	162×229
	6	105×148	125×176	114×162
	7	74×105	88×125	81×114
8	52×74	62×88	57×81	
Classes des formats				

```

\renewcommand{\arraystretch}{1.1}
\setlength{\tabcolsep}{2mm}
\rotatebox{90}{%
\begin{tabular}{|l|l*3{r0{${\times}$}l}|} \hline
\multicolumn{8}{|c|}{Dimension de formats papiers ISO (mm)} \\\hline
& &\multicolumn{2}{c}{série A}
&&\multicolumn{2}{c}{série B}
&&\multicolumn{2}{c}{Série C}\\\hline
& 0& 841&1189 & 1000&1414& 917&1297 && \\\cline{2-8}
& 1& 594& 841 & 707&1000& 648& 917 && \\\cline{2-8}
& 2& 420& 594 & 500& 707& 458& 648 && \\\cline{2-8}
& 3& 297& 420 & 353& 500& 324& 458 && \\\cline{2-8}
& 4& 210& 297 & 250& 353& 229& 324 && \\\cline{2-8}
& 5& 148& 210 & 176& 250& 162& 229 && \\\cline{2-8}
& 6& 105& 148 & 125& 176& 114& 162 && \\\cline{2-8}

```

```

& 7& 74& 105 & 88& 125& 81& 114 &&&\cline{2-8}
\raisebox{8mm}[0mm][0mm]{%
  \rotatebox{90}{\hspace*{8mm}Classes des formats}}
& 8& 52& 74 & 62& 88& 57& 81 &&&\hline
\end{tabular}}

```

1.5. Placement d'éléments à l'intérieur de tableaux

Comme déjà évoqué ci-dessus, le contenu (d'une partie) d'une cellule d'un environnement `tabular` ou `array` peut pivoter à la demande. Cette fonctionnalité peut s'utiliser, par exemple, pour créer des accolades ou parenthèses (tournées), qui couvrent de multiples colonnes ou lignes.

Les blocs dans les exemples suivants et dans la table 1 sont créés par la macro `\Bpara` dont le mécanisme est expliqué ci-dessous. Le principe est de produire une parenthèse (} dans l'exemple) en mode mathématique, mais de cacher sa présence à L^AT_EX. Dans ce but nous utilisons l'environnement `picture` avec une hauteur et largeur déclarées égales à zéro. Une boîte produite par l'environnement `\makebox` avec largeur zéro (pour supprimer la dimension horizontale), contient l'élément à faire tourner. La commande `\smash` assure que L^AT_EX ignorera sa hauteur ; la commande `\rule` (largeur également zéro) contrôle la hauteur de la parenthèse en mode mathématique ; la commande `\left.` est nécessaire pour équilibrer les parenthèses. Comme déjà indiqué, des commandes similaires peuvent être définies pour construire d'autres grands symboles.

```

\setlength{\unitlength}{1mm}
% #1 coordonnée x de point de référence du symbole
% #2 coordonnée y de point de référence du symbole
% #3 angle de rotation pour le symbole (parenthese)
% #4 hauteur totale du symbole
\newcommand{\Bpara}[4]{%
  \begin{picture}(0,0)%
    \put(#1,#2){\makebox[0mm]{\rotatebox{#3}{%
      \smash{\$ \left. \mbox{\rule{0mm}{#4}} \right\} }}}}%
  \end{picture}%
}

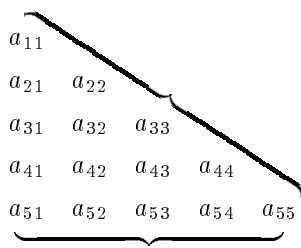
```

Un autre exemple montre deux grandes parenthèses autour d'une matrice triangulaire.

```

\[
\begin{array}{ccccc}
a_{11} & & & & \\
a_{21} & a_{22} & & & \\
a_{31} & a_{32} & a_{33} & & \\
& \Bpara{0}{2}{57}{6.7em} & & & \\
a_{41} & a_{42} & a_{43} & a_{44} & \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \\
& & \Bpara{-4}{-1}{-90}{5.6em} & & \\
& & & a_{54} & a_{55}
\end{array}
\]

```



```
\begin{tabular}{|l|l|r@{\,--\,}|l|l|l|l|l|l|}
\hline
1.4$ũ10^{-4}$& 0.02& 0.06& 0.03& 0.64& 0.935& 0.066& 0.020& & \\
2.8$ũ10^{-4}$& 0.04& 0.09& 0.07& 0.65& 0.922& 0.051& 0.030& & \\
4.5$ũ10^{-4}$& 0.04& 0.15& 0.11& 0.64& 0.865& 0.038& 0.021& & \\
& & & & & \Bpara[-1]{0}{0}{10ex} & 0.018& 0.097& & \\
6.7$ũ10^{-4}$& 0.06& 0.22& 0.15& 0.61& 0.897& 0.036& 0.015& & \\
9.0$ũ10^{-4}$& 0.15& 0.25& 0.21& 0.67& 0.894& 0.027& 0.020& & \\
1.5$ũ10^{-3}$& 0.15& 0.3 & 0.34& 0.63& 0.888& 0.012& 0.021& 0.006& 0.048& \\
3.0$ũ10^{-3}$& 0.15& 1.2 & 0.61& 0.61& 0.917& 0.009& 0.009& 0.003& 0.030& \\
\hline
\end{tabular}
```

1.410 ⁻⁴	0.02–0.06	0.03	0.64	0.935	0.066	0.020	} 0.018 0.097
2.810 ⁻⁴	0.04–0.09	0.07	0.65	0.922	0.051	0.030	
4.510 ⁻⁴	0.04–0.15	0.11	0.64	0.865	0.038	0.021	
6.710 ⁻⁴	0.06–0.22	0.15	0.61	0.897	0.036	0.015	} 0.006 0.048
9.010 ⁻⁴	0.15–0.25	0.21	0.67	0.894	0.027	0.020	
1.510 ⁻³	0.15–0.3	0.34	0.63	0.888	0.012	0.021	} 0.003 0.030
3.010 ⁻³	0.15–1.2	0.61	0.61	0.917	0.009	0.009	

TAB. 1 - Construction d'une parenthèse sur plusieurs rangs

Des constructions plus complexes sont aussi possibles.

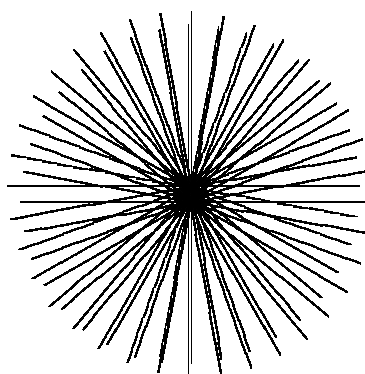
```
\begin{tabular}{|c|c|c|}
\hline
\multicolumn{2}{|c|}{} & 13 \\
\multicolumn{2}{|c|}{} & 23 \\
\multicolumn{2}{|c|}{} & 33 \\
\hline
41 & 42 & 43 \\
\hline
\end{tabular}
```

A	}	13
		23
		33
41	42	43

En utilisant l'extension `ifthen` nous pouvons montrer l'effet de rotation d'une barre verticale sur 360 degrés, avec un pas de 10 degrés. Pour ce faire nous remplaçons le symbole } dans la définition de `\Bpara` par |.

```
\renewcommand{\Bpara}[4]{%
\begin{picture}(0,0)%
\put(#1,#2){\makebox[0mm]{\rotatebox{#3}{%
\smash{\left.\mbox{\rule{0mm}{#4}}\right|}}}}%
\end{picture}%
}
\begin{center}
```

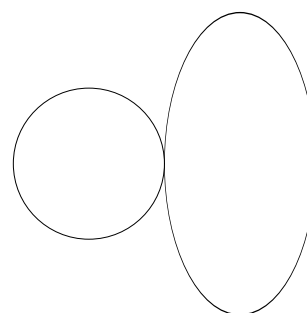
```
\newcounter{Angle}\setcounter{Angle}{0}
\whiledo{\value{Angle}<360}{\Bpara{0}{0}{\theAngle}{7em}%
\addtocounter{Angle}{10}}
\end{center}
```



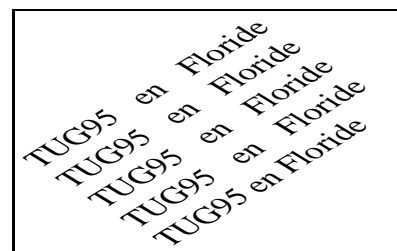
1.6. Les effets combinés

Les commandes de rotation et d'échelle peuvent être combinées et imbriquées à volonté; nous pouvons les appliquer à des objets dans l'environnement `picture` de L^AT_EX.

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,40)
\put(15,20){\circle{20}}
\put(35,20){\scalebox{1}[2]{\circle{20}}}
\end{picture}
```



```
\framebox{\resizebox{5cm}{25mm}{%
\rotatebox{45}{\parbox{3cm}{%
TUG95 en Floride TUG95 en Floride
TUG95 en Floride TUG95 en Floride
TUG95 en Florida
}}}
```



2. L'inclusion de fichiers graphiques

Pour faciliter l'inclusion des différents types de fichiers graphiques qui peuvent être manipulés par les pilotes, L^AT_EX 2_ε propose une syntaxe uniforme pour leur inclusion. Généralement, l'interprétation d'un fichier est basée sur l'extension qui identifie son type. Des fichiers de configuration pour chaque pilote définissent les types d'extensions que le pilote en question peut manipuler. L'utilisateur peut étendre cette liste en utilisant les déclarations décrites plus loin. Bien que la plupart des exemples ci-dessous utilisent des inclusions PostScript, le système permet également de traiter correctement des fichiers de type TIFF, PCX, Macdraw, etc. Il est entendu que dans ces cas, les pilotes adéquats doivent être disponibles pour chaque usage.

2.1. L'extension graphics

En utilisant l'extension `graphics` un fichier graphique *fichier* peut être inséré avec la commande :

```
\includegraphics[llx,lly][urx,ury]{fichier}
```

Sans aucun argument optionnel, la taille du graphique sera déterminé par la lecture du fichier externe *fichier* (contenant le graphique lui-même, ou une description, voir ci-dessous).

Si l'argument `[urx,ury]` est présent, il spécifiera les coordonnées du coin supérieur droit de l'image, comme une paire de dimensions T_EX. Les unités par défaut sont de grands points PostScript. Ceci veut dire que, par exemple, `[72,72]` et `[1in,1in]` sont équivalents, parce qu'un pouce (un *inch* ou `1in`) contient 72 points PostScript, qui, comme cette dernière unité est le défaut, ne doit pas être spécifiée. Si un seul argument optionnel est donné, le coin inférieur gauche de l'image est supposé être à `[0,0]`, sinon `[llx,lly]` précise les coordonnées de ce point.

La forme étoilée de la commande `\includegraphics*` sert à "détourer" la taille de l'image graphique à la taille donnée par le rectangle englobant le graphique (la *bounding box*). Avec la forme normale (sans `*`) de la commande `\includegraphics` n'importe quelle partie de l'image graphique qui tombe à l'extérieur (employant le terme anglais) de la *bounding box* surimprimera le texte environnant.

Pour se rendre compte des différents arguments, nous montrerons les effets de chacun d'entre eux. Ci-dessous le cadre correspond à la *bounding box*, et la ligne de référence est indiquée par une ligne horizontale. Nous utilisons un petit programme PostScript (le fichier `w.eps`), qui dessine la lettre W majuscule, et quelques lignes. Sa source est décrite ci-dessous. Le commentaire PostScript `%%BoundingBox` stipule que l'image commence au point avec les coordonnées `100 100` (en points PostScript) et va jusqu'à `172 172`, c.-à-d. que sa taille naturelle est un pouce (inch) sur un pouce.

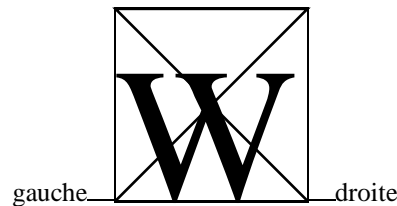
```
%! Une petite image PostScript
```

```

%%BoundingBox:100 100 172 172
100 100    translate % translation au point 100 100
  0  0     moveto   % y mettre le point courant
 72 72     rlineto  % tracer une ligne diagonale
 72 neg 0   rlineto % tracer une ligne horizontale
 72 72 neg rlineto % tracer une autre ligne diagonale
stroke     % dessiner (rendre visibles) ces lignes
  0  0     moveto   % redefinir le point courant
/Times-Roman findfont % activer la police Times-Roman
72 scalefont setfont  % la mettre à l'échelle 1 pouce
                    % et en faire la police courante
(W) show      % dessiner la lettre W majuscule
showpage     % montrer la page

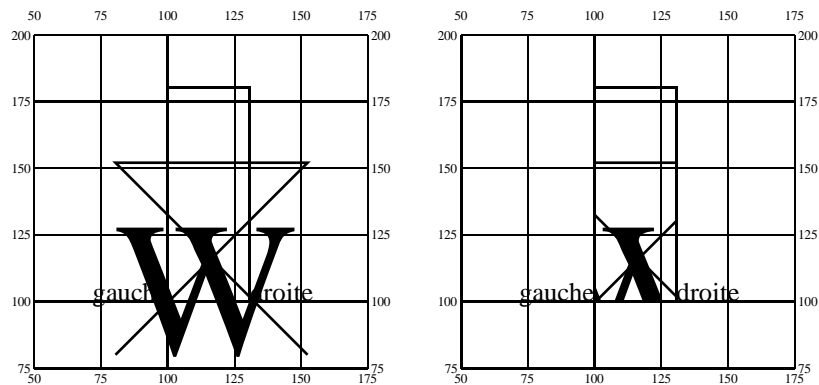
gauche\HR\framebox{\includegraphics{w.eps}}\HR droite

```



Dans ce cas l'image et sa *bounding box* coïncident parfaitement.

Maintenant nous spécifions une *bounding box* qui correspond à une partie seulement de l'image, de sorte que les parties se trouvant à l'extérieur de ses limites recouvriront le matériel entourant l'image. Si la forme étoilée (avec "détourage") est utilisée, l'image est réduite à la *bounding box*, comme le montre l'exemple de droite. Pour mieux se rendre compte de ce qui se passe, une grille est surimprimée sur la figure.

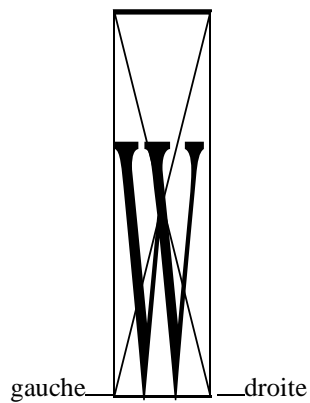


Les exemples suivants montrent comment les commandes `\scalebox` et `\resizebox` peuvent être utilisées ensemble avec `\includegraphics`. Notez également la transformation de la [bounding box] de l'image.

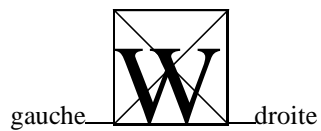
```
gauche\HR\framebox{\scalebox{.5}{\includegraphics{w.eps}}}\HR droite
```



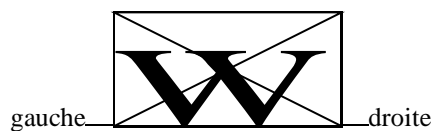
```
gauche\HR\framebox{\scalebox{.5}[2]{\includegraphics{w.eps}}}\HR droite
```



```
gauche\HR  
\framebox{\resizebox{15mm}{!}{\includegraphics{w.eps}}}%  
\HR droite
```

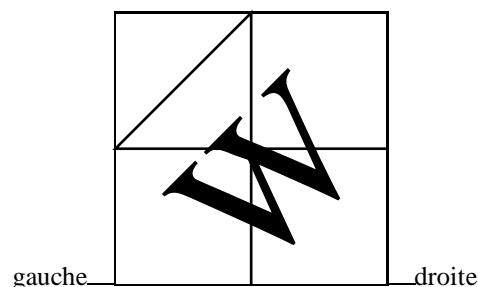


```
gauche\HR  
\framebox{\resizebox{30mm}{15mm}{\includegraphics{w.eps}}}%  
\HR droite
```



Ajouter des rotations rend les choses encore plus intéressantes.

```
gauche\HR  
\framebox{\rotatebox{45}{\includegraphics{w.eps}}}%  
\HR droite
```



2.2. L'extension graphicx

Si vous choisissez d'utiliser l'extension `graphicx` au lieu de `graphics`, la syntaxe devient quelque peu plus transparente.

```
\includegraphics*[liste clés-valeurs]{fichier}
```

La liste de toutes les clés possibles est :

<code>bb</code>	La <i>bounding box</i> de (le rectangle englobant) l'image. Il est représenté par ses 4 coordonnées, séparées par des espaces.
<code>bbllx</code>	coordonnée x du point inférieur gauche (obsolète ¹).
<code>bbllly</code>	coordonnée y du point inférieur gauche (obsolète ¹).
<code>bburx</code>	coordonnée x du point supérieur droit (obsolète ¹).
<code>bbury</code>	coordonnée y du point supérieur droit (obsolète ¹).
<code>natheight</code>	Hauteur naturelle de la figure ² .
<code>natwidth</code>	Largeur naturelle de la figure ² .
<code>angle</code>	Angle de rotation (en degrés, dans le sens inverse des aiguilles d'une horloge)
<code>width</code>	Largeur désirée.
<code>height</code>	Hauteur désirée.
<code>scale</code>	Facteur d'échelle.
<code>clip</code>	Active le détournement aux limites spécifiées par la <i>bounding box</i> (par exemple avec le paramètre <code>bb=</code>). Le paramètre <code>clip</code> est de type booléen, c.-à-d. qu'il peut avoir seulement les valeurs « <code>true</code> » (« vrai », sa valeur par défaut), ou « <code>false</code> » (« faux »).
<code>draft</code>	Localement change en mode « brouillon ». C'est une valeur booléenne, comme « <code>clip</code> ».
<code>type</code>	Spécifie le type de graphique.
<code>ext</code>	Spécifie l'extension du fichier.

1. Gardée uniquement pour raisons de compatibilité. `bbllx=a`, `bbllly=b`, `bburx=c`, `bbury=d` est équivalent à `bb = a b c d`, donc la forme `bb=` doit être utilisée

2. Ces arguments peuvent être utilisés pour définir les coordonnées du point inférieur gauche (0 0) et du point supérieur droit (`natwidth natheight`).

`read` Spécifie l'extension du fichier à lire.
`command` Spécifie une commande quelconque à appliquer au fichier.

La forme étoilée est seulement une indication pour la compatibilité avec la version standard. Elle est équivalente à la clé `clip`.

Si les dimensions sont données sans unités de mesure, l'unité sera en points PostScript.

Les sept premières clés spécifient la taille de l'image. Il est nécessaire de les spécifier quand le fichier ne peut être lu par T_EX (nous aurions alors des informations incorrectes ou illisibles), ou si l'on veut détourer l'image à un certain rectangle.

Les quatre autres clés suivantes sont des spécifications d'échelle ou de rotation qui modifient l'image. Des effets similaires peuvent être obtenus en utilisant l'extension `graphics` et la commande `\includegraphics` et en plaçant ce dernier comme argument de `\resizebox`, `\rotatebox` ou `\scalebox` (voir la section 2.1 ci-dessus).

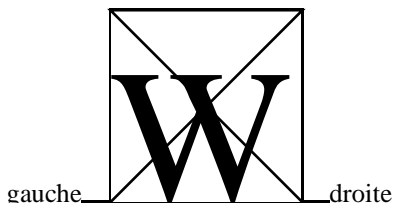
Les clés sont interprétées de gauche à droite, ainsi `[angle=90, height=2cm]` signifie que l'image sera tournée de 90 degrés, et sa hauteur sera de 2cm, tandis que `[height=2cm, angle=90]` aura pour résultat une *largeur* de 2cm.

Par défaut L^AT_EX réserve pour l'image l'espace spécifié soit dans le fichier soit par les arguments optionnels. Si une partie de l'image se trouve à l'extérieur de cette zone, par défaut le texte sera surimprimé. Si la forme étoilée, ou l'option `clip`, est utilisée, la partie de l'image à l'extérieur de la surface spécifiée ne sera pas imprimée.

Les quatre dernières clés suppriment l'analyse du nom du fichier. Quand elles sont utilisées, l'argument principal *fichier* ne doit pas avoir d'extension (voir la description de la commande `\DeclareGraphicsRule` ci-dessous).

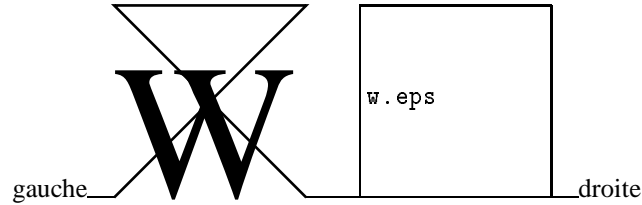
Nous répétons les exemples donnés dans la section précédente, cette fois-ci en utilisant la syntaxe de l'extension `graphicx`. Dans la plupart des cas cette forme est intuitive et plus simple à comprendre.

```
gauche\HR\framebox{\includegraphics{w.eps}}\HR droite
```



Dans cet exemple on ne change pas le rectangle limite (*bounding box*) ni les dimensions de l'image. Avec l'option `draft` seul un cadre, montrant la surface rectangulaire occupée par l'image en question, ainsi que le nom du fichier sont imprimés.

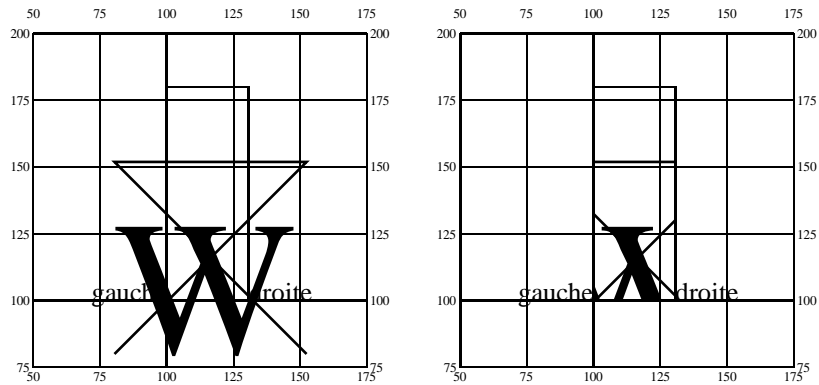
```
gauche\HR\includegraphics{w.eps}\HR
\HR\includegraphics[draft]{w.eps}\HR droite
```



Ensuite, nous considérons seulement une partie de l'image et comme certaines parties seront à l'extérieur de ces limites nous surimprimerons le texte entourant. En détournant, nous nous débarrassons de tout le "superflu" (la grille est montrée pour plus de clarté seulement).

```
\framebox{\includegraphics [bb=120 120 150 200]{w.eps}}droite}}
```

```
\framebox{\includegraphics [bb=120 120 150 200,clip]{w.eps}}droite}}
```

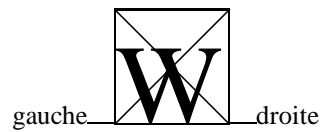


Une image peut être redimensionnée. Un même facteur d'échelle peut être appliqué globalement dans les deux directions, ou, en spécifiant à la fois les deux dimensions, une image déformée peut être engendrée.

```
gauche\HR\framebox{\includegraphics [scale=.5]{w.eps}}\HR droite
```

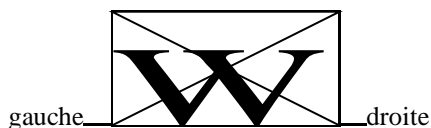


```
gauche\HR\framebox{\includegraphics [width=15mm]{w.eps}}\HR droite
```



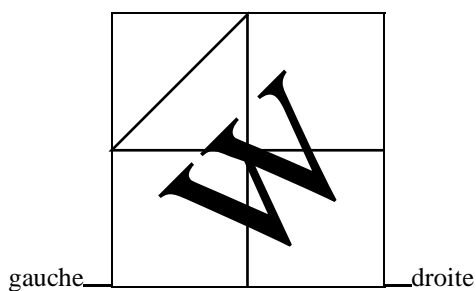
```
gauche\HR
```

```
\framebox{\includegraphics [height=15mm,width=30mm]{w.eps}}%
\HR droite
```



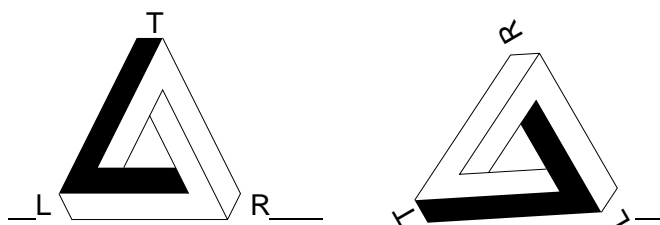
D'une manière analogue on peut faire pivoter l'information.

```
gauche\HR
\framebox{\includegraphics [angle=45]{w.eps}}%
\HR droite
```

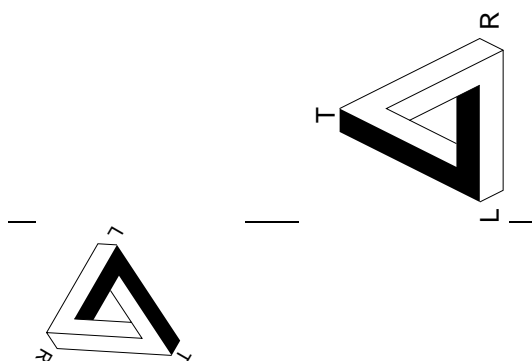


En combinant rotations et mises à l'échelle, nous pouvons obtenir tout effet voulu.

```
\HR \includegraphics [width=3cm]{Escher.eps}\HR
\HR \rotatebox{-240}{\includegraphics [width=3cm]{Escher.eps}}\HR
```



```
\HR \includegraphics [angle=240,width=27mm]{Escher.eps}\HR
\HR \includegraphics [angle=90,width=27mm]{Escher.eps}\HR
```



Comme exemple final nous montrons dans la figure 1 comment on optimise les dimensions d’une image pour tirer profit au maximum de la taille du papier (A4 ou 210cm × 297cm dans ce cas). Il s’agit d’une carte de Gdańsk, dessinée à l’italienne, dont nous voulons que la largeur (hauteur pour l’original) devienne (après rotation) égale à la largeur du texte déclarée c.-à-d. 21cm. Les décalages horizontaux et verticaux `\hoffset` et `\voffset` doivent aussi être spécifiés, parce qu’en L^AT_EX la “classe” article imprime son premier point à 40mm approximativement du point supérieur gauche de la page.

2.3. Déclaration globale de la valeur des clés

Si nous voulons spécifier une valeur globale pour un ensemble de clés, nous pouvons utiliser la commande `\setkeys` définie dans l’extension `keyval` (décrite à la section A.3 ci-dessous).

Considérons, par exemple, le cas où nous voulons dimensionner toutes nos figures à la largeur de la ligne. Nous pouvons alors préciser :

```
\setkeys{Gin}{width=\linewidth}
```

Le premier argument `Gin` de la commande `\setkeys` se réfère à la commande `\includegraphics`. Toutes les images insérées avec cette commande (quand `graphicx` est chargé) seront à la largeur désirée à l’intérieur d’un groupe ou d’un environnement.

De la même façon nous pouvons préciser n’importe lequel des arguments possibles de la commande `\rotatebox` en utilisant la spécification `Grot` :

```
\setkeys{Grot}{origin=tc}
```

```
\documentclass[a4paper]{article}
\usepackage[dvips]{graphics}
\setlength{\textwidth}{210mm}\setlength{\textheight}{297mm}
\setlength{\voffset}{-40mm}\setlength{\hoffset}{-40mm}
\pagestyle{empty}
\begin{document}
\resizebox{\textwidth}{!}{\rotatebox{90}{\includegraphics{gdansk.eps}}}
\end{document}
```

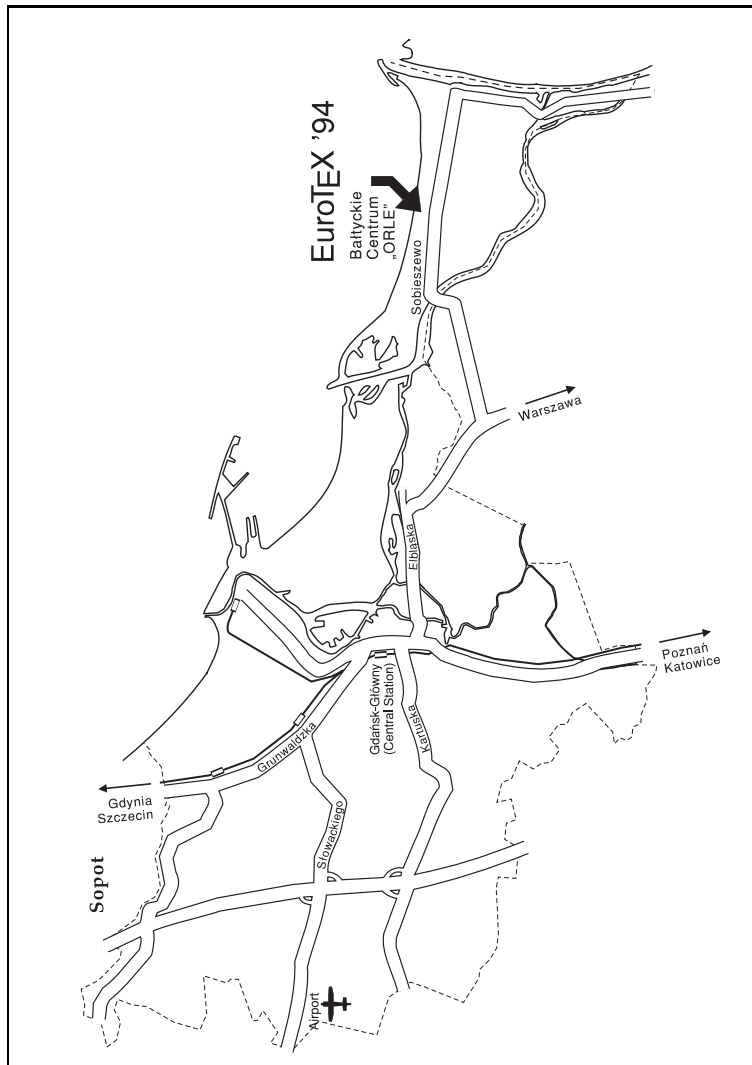


FIGURE 1 - *Optimiser les dimensions d'une image*

2.4. Commandes avancées du progiciel graphicx

Une liste de répertoires dans lesquels L^AT_EX devra chercher les fichiers graphiques peut être définie par la commande `\graphicspath`, dont la syntaxe est :

```
\graphicspath{liste-rep}
```

liste-rep est une liste de répertoires, chacun à l'intérieur de parenthèses `{}` (même s'il n'y en a qu'un dans la liste). Par exemple,

```
\graphicspath{./}{eps/}{tiff/}
```

demandera à L^AT_EX de faire une recherche dans le répertoire courant ainsi que dans ses sous-répertoires `eps` et `tiff`. Par défaut L^AT_EX cherchera les fichiers graphiques dans les mêmes répertoires où il a trouvé les fichiers T_EX.

```
\DeclareGraphicsExtensions{liste-ext}
```

La commande `\DeclareGraphicsExtensions` nous permet de spécifier le comportement du système quand aucune extension n'est donnée dans l'argument de la commande `\includegraphics`. *{liste-ext}* une liste d'extensions de fichiers, séparées par des virgules. Le nom complet d'un fichier est construit en complétant le nom spécifié tour à tour avec chacune des extensions de la liste *liste-ext*. Le premier nom complet qui correspond à un fichier présent dans un des répertoires « vus » par L^AT_EX sera pris.

Alors que l'algorithme précédent teste l'existence d'un fichier pour déterminer quelle extension utiliser, au cas où la commande `\includegraphics` est spécifiée sans extension, le fichier graphique doit exister au moment où L^AT_EX est exécuté. Cependant, si l'extension est déclarée, ex. `\includegraphics{gr.eps}` au lieu de `\includegraphics{gr}`, alors le fichier graphique n'a pas besoin d'exister lorsque L^AT_EX est exécuté¹. Toutefois, L^AT_EX a besoin de connaître la taille de l'image. Cette taille doit donc être spécifiée dans les arguments de la commande `\includegraphics`, ou dans le fichier actuellement lu par L^AT_EX (cela peut être le fichier graphique lui-même, ou un autre fichier déclaré avec l'argument `read=` ou construit à partir de la liste des fichiers extensions). Dans le dernier cas, ce fichier doit exister au moment où l'on exécute L^AT_EX.

Avec la déclaration suivante, la commande `\includegraphics` cherchera d'abord le fichier `fichier.ps` et, si aucun fichier n'existe, elle cherchera le fichier `fichier.ps.gz`.

```
\DeclareGraphicsExtensions{.ps,.ps.gz}
\includegraphics{fichier}
```

1. Par exemple, ce fichier peut être créé à la volée par l'argument *commande* déclaré dans la déclaration `\DeclareGraphicsRule`, décrite plus loin.

Pour être certain que le nom complet d'un fichier est toujours spécifié, un argument vide pour la commande `\DeclareGraphicsExtensions` doit être utilisé. Notons aussi que, dans l'exemple ci-dessous, la taille de chaque image (bitmap) est déclarée explicitement avec la commande `\includegraphics`, parce que L^AT_EX ne peut pas interpréter les fichiers (binaires) de type `pcx`.

```
\DeclareGraphicsExtensions{}
\includegraphics[1in,1in]{fichier.pcx}
\includegraphics[75pt,545pt][50pt,530pt]{fichier.pcx}
\includegraphics[bb=75 545 50 530]{fichier.pcx}
```

L'action qui doit être prise quand on se trouve en présence d'un fichier avec une extension donnée est contrôlée par la commande suivante :

```
\DeclareGraphicsRule{ext}{type}{fichier-lu}{commande}
```

Plusieurs déclarations `\DeclareGraphicsRule` peuvent être présentes. La signification de chacun des arguments est :

ext

L'extension du fichier. Elle peut être explicite ou, si l'argument de la commande `\includegraphics` n'a pas d'extension, elle peut être extraite de la liste des extensions spécifiées dans l'argument *liste-ext* de la commande `\DeclareGraphicsExtensions`.

type

Le "type" du fichier impliqué. Tous les fichiers du même type seront appelés par la même commande interne (qui doit être définie dans le "le fichier du pilote" correspondant). Par exemple, les fichiers avec l'extension `ps`, `eps`, ou `ps.gz` sont tous classés comme type `eps`.

fichier-lu

L'extension du fichier qui doit être lu pour déterminer la taille de l'image. Il peut être identique à *ext*, mais, dans le cas d'une image compressée ou binaire, il ne pourra pas être interprété directement par L^AT_EX. Ainsi l'information de la taille (*bounding box*) se trouve normalement dans un fichier séparé (par exemple, les fichiers PostScript compressés par l'utilitaire `gzip` sont habituellement caractérisés par l'extension `ps.gz` et ils ont un fichier lisible correspondant avec l'extension `ps.bb` qui ne contient que l'information *bounding box*). Si l'argument *fichier-lu* est vide {}, le système n'essayera pas de localiser un fichier externe pour déterminer la taille, et cette dernière devra être spécifiée directement dans les arguments de *includegraphics*. Si le fichier pilote spécifie une procédure pour lire les fichiers déclarant la taille pour *type*, alors la procédure en question sera utilisée, sinon on utilisera la procédure pour lire les fichiers `eps`. Par conséquent, en l'absence d'un format spécifique, la taille d'une image bitmap peut être spécifiée en utilisant la syntaxe des images PostScript, c.-à-d. avec une ligne `%%BoundingBox`.

Le système décrit ici donne quelques problèmes si l'extension *ext* ne correspond pas à l'argument *type*. Nous pouvons, par exemple, avoir une série de fichiers PostScript appelés `fichier.1`, `fichier.2`, ... Il n'est pas facile pour L^AT_EX de détecter automatiquement si ces fichiers sont en format PostScript, à moins que l'argument `type=eps` soit utilisé pour chaque fichier dans la commande `\includegraphics`. Afin de faciliter les choses on peut déclarer un type par défaut en utilisant un type `*`. En particulier, une déclaration

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

spécifie que par défaut tous les fichiers graphiques caractérisés par une extension inconnue, seront traités comme des fichiers PostScript et le fichier graphique sera lu pour localiser la ligne `%%BoundingBox`.

commande

Normalement cet argument est vide. Si *commande* n'est pas nulle, son contenu prendra la place du nom de fichier dans la commande `\special` générée. Ainsi pour les fichiers compressés on peut vouloir les décompresser avant de les inclure dans le fichier généré par le pilote (par exemple `dvips`) pour impression. Dans ce cas on peut écrire :

```
\DeclareGraphicsRule{ps.gz}{eps}{ps.bb}{'gunzip #1}
```

où l'argument `#1` est le nom complet du fichier. Dans ce cas l'argument final demandera au pilote (`dvips`) d'utiliser la commande `gunzip` pour décompresser le fichier avant de l'inclure.

La Table 2 montre diverses possibilités pour les arguments de la commande `\DeclareGraphicsRule`.

3. Les courbes de Bézier

L^AT_EX 2_ε permet de construire des courbes mathématiques assez sophistiquées, en utilisant une technique d'approximation pour des courbes de Bézier. Le langage PostScript utilise aussi des courbes de Bézier (de troisième ordre) comme base de construction de ses courbes.

`\qbezier[N](AX,AY)(BX,BY)(CX,CY)`

Cette commande définit une courbe de Bézier du deuxième degré, limitée par ses deux points extrêmes, (AX,AY) et (CX,CY) , et avec le point (BX,BY) comme point de contrôle. Le paramètre optionnel N , s'il est présent, spécifie que $N + 1$ points sont tracés pour construire la courbe. Dans le premier exemple ci-dessous A et C sont les points extrêmes

<i>ext</i>	<i>type</i>	<i>fichier-lu</i>	<i>commande</i>
<i>PostScript de base</i>			
ps	eps	ps	
eps	eps	eps	
<i>décompression dynamique</i>			
pz	eps	bb	'gunzip -c #1
ps.pz	eps	ps.bb	'gunzip -c #1
eps.pz	eps	eps.bb	'gunzip -c #1
<i>formats MS-DOS</i>			
tif	tif		
pcx	bmp		
bmp	bmp		
msp	bmp		
<i>formats Macintosh</i>			
pict	pict		
pntg	pntg		

TAB. 2 - Exemples d'arguments de la commande `\DeclareGraphicsRule`

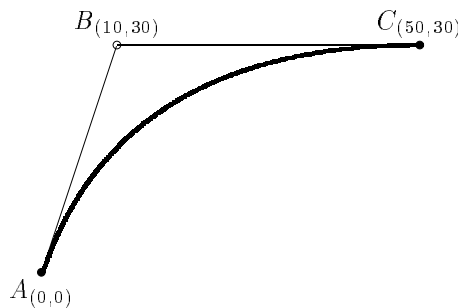
et B est le point de contrôle. Le nombre de points tracés nécessaire pour obtenir une courbe lisse est calculé automatiquement (par défaut).

La variation du nombre de points a un effet très visible. Dans le second exemple, deux courbes utilisent un nombre de points par défaut, tandis que le nombre à utiliser pour les autres courbes est explicité. La position des points de contrôle est indiquée par un cercle.

```

\setlength{\unitlength}{1mm}
\begin{picture}(55,35)(-3,-3)
  \linethickness{1pt}
  \qbezier(0,0)(10,30)(50,30)
  \thinlines
  \put(0,0){\line(1,3){10}}
  \put(50,30){\line(-1,0){40}}
  \put(0,0){\circle*{1}}
  \put(0,-1){\makebox(0,0)[t]{%
    ${A}_{(0,0)}$}}
  \put(10,30){\circle{1}}
  \put(10,31){\makebox(0,0)[b]{%
    ${B}_{(10,30)}$}}
  \put(50,30){\circle*{1}}
  \put(50,31){\makebox(0,0)[b]{%
    ${C}_{(50,30)}$}}
\end{picture}

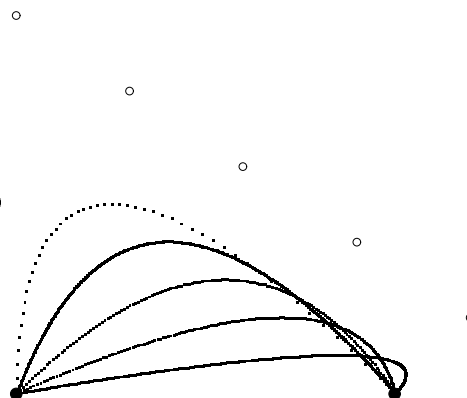
```



```

\setlength{\unitlength}{.5mm}
\begin{picture}(120,100)(-5,0)
  \linethickness{.5pt}
  \qbezier [50](0,0)(0,100)(100,0)
  \qbezier (0,0)(30,80)(100,0)
  \qbezier [150](0,0)(60,60)(100,0)
  \qbezier [200](0,0)(90,40)(100,0)
  \qbezier (0,0)(120,20)(100,0)
% marquez les points finaux
  \put(0,0){\circle*{3}}
  \put(100,0){\circle*{3}}
% marquez les point de contrôle
  \multiput(0,100)(30,-20){5}%
    {\circle{2}}
\end{picture}

```



Appendice : les autres extensions du « paquet » graphique

A.1. Les extensions epsfig et rotating

Sebastian Rahtz avait développé pour L^AT_EX 2.09 deux “styles”, epsfig et rotating qui sont devenus très populaires pour inclure des graphiques ou faire pivoter des images. Pour faciliter la transition vers L^AT_EX 2_ε deux extensions avec le même nom sont distribuées avec le « paquet » graphique. Ainsi les utilisateurs de ces anciens styles peuvent continuer à utiliser une syntaxe qui leur est familière. En plus l’extension rotating offre quelques fonctionnalités supplémentaires, comme les environnements `sidewaysstable` et `sidewaysfigure`.

A.2. L’extension trig

L’extension trig n’est pas supposée être utilisée directement dans les documents L^AT_EX. En fait, elle calcule les fonctions trigonométriques comme sinus, cosinus et tangente, qui sont nécessaires pour calculer l’espace occupé par une boîte pivotée. Cette extension est aussi utilisée par le programme d’Alan Jeffrey fontinst qui convertit les fichiers PostScript dans un format utilisable par T_EX.

A.3. L’extension keyval

À nouveau il s’agit d’une extension qui ne devrait pas directement concerner l’utilisateur *Lambda*. Elle sert essentiellement à interpréter les arguments des commandes contenant des listes de paires « *mots-clé=valeur* ». Elle est utilisée par l’extension graphicx et aussi dans pstricks de Timoty van Zandt.

A.4. L'extension `landscape`

L'extension `landscape` a besoin des même options que l'extension `graphics`. Elle définit l'environnement `landscape` qui a comme effet de tourner une page à 90 degrés c.-à-d. que la page est composée à l'italienne. L'entête et le bas de la page ne sont pas affectés et ils apparaissent dans la position standard (`portrait`).