Les modèles **Animation.mod** et **Capture.mod**

Pour TeXgraph 1.95

Table des matières

1 Animation.mod

1.1 Présentation

Le modèle *Animation.mod* permet une création simplifiée d'animations dans TeXgraph, celles-ci pouvant être enregistrées automatiquement sous forme de fichiers pgf [1] (frame1.pgf, frame2.pgf...) puis compilées dans un même fichier pdf avec une image par page, chaque page étant à la taille de l'image.

Ce fichier d'images peut être :

- converti en gif animé [c'est une option du menu du modèle], cela suppose que vous disposiez de la suite *ImageMagick* [2], car c'est son utilitaire **convert** qui est utilisé,
- converti en fichier flash [c'est une option du menu du modèle], cela suppose que vous disposiez des outils *Swftools* [3]. Deux fichiers sont créés : *<fichier>.swf* et *<fichier>.html*, le deuxième contenant le code permettant d'inclure le premier dans une page html,
- utilisé pour fabriquer un pdf animé à l'aide du package *animate* d'*Alexander* GRAHN.

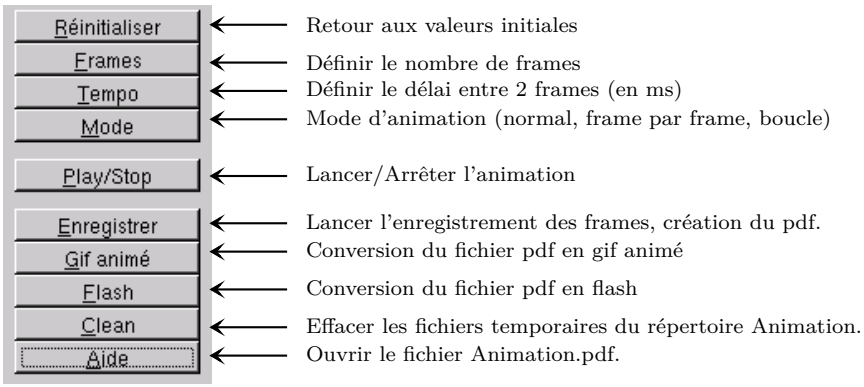


FIGURE 1 – Le menu

1.2 Le menu

Celui-ci est composé des boutons suivants :

- **Réinitialiser** : permet d'arrêter l'animation si elle est en cours, et de la replacer à son état initial.
- **Frame** : permet de définir le nombre de frames de l'animation, chaque frame donnera un fichier pgf à l'enregistrement, et une page dans le fichier pdf final.
- **Tempo** : réglage de la temporisation, c'est à dire du laps de temps entre 2 frames, celui-ci est compté en milli-secondes. En général la valeur 100 semble convenable y compris pour l'enregistrement.
- **Mode** : définition du mode d'animation.
  - Mode=0 : c'est le mode normal, l'animation se déroule automatiquement de la première à la dernière frame.
  - Mode=1 : c'est le mode frame par frame, il faut cliquer le bouton *Play/Stop* pour passer à la frame suivante.
  - Mode=2 : c'est le mode en boucle.

1. Utilisant les macros du package pgf, compilables avec pdfLaTeX, ceci parce que les lignes en pointillés et tirets sont mal rendues en Flash si on passe par la chaine latex+dvips+ps2pdf.
2. http://imagemagick.sourceforge.net/
3. http://www.swftools.org/

L'enregistrement se fait dans le mode normal.

- **Play/Stop** : permet de lancer ou stopper l'animation. Pour remettre celle-ci au point de départ alors qu'elle n'est pas terminée, il faut utiliser le bouton *Réinitialiser*.
- **Enregistrer** : permet de lancer l'animation en mode normal, chaque frame est affichée à l'écran et enregistrée dans un dossier temporaire sous forme d'un fichier pgf : frame1.pgf, frame2.pgf ... Lorsque l'animation est terminée vous devez donner un nom de fichier (avec chemin, mais sans extension), les fichiers « frames » seront alors compilés en un fichier pdf qui portera ce nom, avec une image par page, et chaque page étant à la taille de l'image.
  Mise en garde : si vous avez un grand nombre de frames et une temporisation rapide, il est possible que votre système ait du mal à suivre et que toutes les frames ne soient pas encore enregistrées sur le disque au moment où vous allez lancer la compilation, il ne sert donc à rien de se précipiter lors de cette phase !
- **Gif animé** : permet de convertir le fichier pdf créé à l'enregistrement, en gif animé. Cette option utilise le programme *convert*. Le fichier gif porte le même nom que le fichier pdf.
- **Flash** : permet de convertir le fichier pdf créé à l'enregistrement, en fichier flash (*.swf). Cette option utilise les outis *Swftools*. Un fichier html est également créé, il contient le code permettant d'inclure le fichier flash dans une page html. Ces deux fichiers portent le même nom que le fichier pdf.
- **Aide** : permet d'ouvrir le présent document dans xpdf.

1.3 Créer une animation

On commence par importer le fichier modèle (F3) *Animation.mod* (le fichier de macros *Animation.mac* est automatiquement chargé par le modèle).

• On compose le graphique qui correspond à la première frame. Il y aura en général des éléments fixes et des éléments mobiles, ces derniers vont dépendre de **paramètres** qui vont changer au cours de l'animation, on déclare ces paramètres en **variables globales** avec leur valeur d'origine (valeur par défaut).

Puis il faut complèter certaines macros qui sont déjà dans la liste des macros :

• La macro **Initialiser()** : dans celle-ci on remet les paramètres à leur valeur par défaut et on peut terminer par la commande *ReCalc()* qui aura pour effet de revenir à la première frame.

• La macro **OnBeginAnim()** : si vous avez des éléments graphiques à créer juste le temps de l'animation, ou bien si vous voulez recréer vos éléments mobiles en mode NotXor pour plus de fluidité, c'est ici qu'il faut le faire. Cette macro est exécutée juste avant le lancement de l'animation.

• La macro **OnEndAnim()** : si vous avez des éléments graphiques à détruire juste après l'animation, ou bien si vous voulez afficher en mode normal les éléments mobiles créés en mode NotXor, c'est ici qu'il faut le faire. Cette macro est exécutée juste après l'animation.

• La macro **MakeFrame()** : cette macro à un paramètre (%1 qui représente le numéro de la frame) est le coeur de l'animation, dans celle-ci vous devez donner à vos différents paramètres la valeur correspondant à la frame n° %1. On termine cette macro soit par un *ReCalc()*, soit par un *Move(...)* si vos éléments mobiles sont en mode NotXor.

• Avec le bouton *Frames*, définissez le nombre de frames de votre animation.

1.4 Exemple 1

Nous allons représenter un point M en mouvement circulaire autour de l'origine avec un rayon de 4 :

1. Charger le modèle *Animation.mod* (Shift+F2).
2. Créer les axes (Ctrl+A).
3. Nous prendrons comme paramètre l'angle $(\vec{\imath}, \overrightarrow{OM})$, appelons le **alpha**. On crée alors une variable globale (F7), on la nomme **alpha**, avec la valeur 0.
4. Créer un arc de cercle (Alt+A), appelé **trajectoire**, avec la commande
   `[4,0,4*exp(i*alpha),4]`
   et dans les attributs prendre : Width=4 et la couleur rouge.
5. Créer un point (Alt+Maj+P), appelé **M**, avec la commande `4*exp(i*alpha)`, et dans les attributs cocher l'option bigdot dans DotStyle.

Nous avons constitué la frame initiale, passons aux macros

6. Dans la macro **Initialiser()**, ajouter les instructions :

```
alpha:=0, ReCalc(M, trajectoire)
```

7. Dans la macro **MakeFrame()**, ajouter les instructions :

```
alpha:= %1*pi/18, ReCalc(M, trajectoire)
```

   cela signifie que l'on tourne de $\pi/18$ (10 degrés) à chaque frame, cela nous donne donc 36 frames.

8. Avec le bouton *Frames* définir le nombre de frames à 36. Reste à essayer !

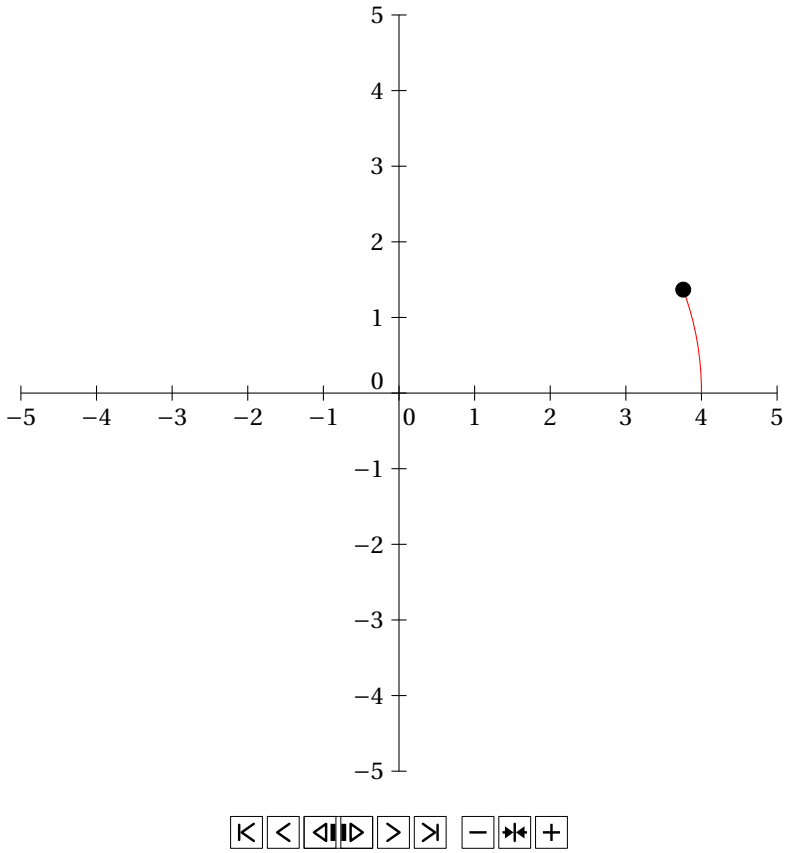
   Nous allons enregistrer l'animation

9. Cliquer le bouton *Enregistrer* et valider. On voit l'animation se dérouler, et à la fin de celle-ci une fenêtre s'ouvre demandant un nom de fichier, donner un non (avec chemin) mais sans extension, et valider. Nous supposerons que ce nom est *Exemple1*.

Pour créer le fichier Exemple1.pdf, TeXgraph a généré le fichier TEX suivant :

```
\documentclass[11pt,frenchb]{article}
\usepackage[utf8]{inputenc}
\usepackage[upright]{fourier}
\usepackage{pgf,amssymb,amsmath,amsfonts,babel}
\usepackage[a4paper,margin=0cm,pdftex]{geometry}
\usepackage[active,tightpage]{preview}
\pagestyle{empty}
\begin{document}
  \newcounter{compt}
  \setcounter{compt}{1}
  \loop
  \begin{preview}
    \input{frame\thecompt.pgf}%
  \end{preview}
  \ifnum \thecompt<36\addtocounter{compt}{1}
  \repeat
\end{document}
```

puis celui-ci a été compilé avec pdflatex.
Ce fichier peut-être utilisé pour faire une animation dans un fichier pdf, comme ici :



1.5 Exemple 2

Dans cet exemple, nous allons illustrer ce qu'est une courbe cycloïde, en montrant une roue de rayon 1 roulant sur l'axe des abscisses. Initialement la roue aura son centre sur l'axe des ordonnées et la valve sera confondue avec l'origine. Au fur et à mesure que la roue avancera il nous faudra dessiner la trajectoire décrite par la valve. Pour le mouvement de la roue, nous utiliserons le mode NotXor pour éviter de tout redessiner à chaque frame.

1. Charger le modèle *Animation.mod* (F3).
2. Modifier la fenêtre (F6) en prenant Xmin=-1, Xmax=13, Ymin=-0.5 et Ymax=3.
3. Créer les axes (Ctrl+A).
4. Créer une variable globale appellée *courbe* et initialisée à `0`. Elle contiendra la liste des positions successives de la valve, le premier étant l'origine.
5. Créer une ligne polygonale (Ctrl+L) appelée *trajectoire*, avec la commande `courbe`, puis dans les Attributs, prendre Width=8 et Color=red.
6. Créer une variable globale appellée *Theta* et initialisée à `0`, cette variable sera notre paramètre pour les éléments mobiles, elle représente l'angle $(\overrightarrow{CM}, -\vec{j})$ où C désigne le centre de la roue et M la valve.
7. Créer un élément graphique Utilisateur appelé *mobile* et avec la commande :

```
[Width:=8, Color:=blue,
$X:=Theta, $M:=X+i+exp(i*(-pi/2-Theta)),
Cercle(X+i,1), Ligne( [X+i, M], 0), DotStyle:=cross, Point(X+i),
DotStyle:=bigdot, LineStyle:=noline, LabelStyle:=framed,
LabelDot( M, "$M$", "N", 1)
]
```

Nous avons constitué la frame initiale, passons aux macros

8. Dans la macro **Initialiser()**, ajouter les instructions :

```
Theta:=0, courbe:=0, ReCalc()
```

9. Dans la macro **OnBeginAnim()**, ajouter les instructions :

```
NotXor(mobile)
```

on recrée ainsi l'élément mobile en mode NotXor.

10. Dans la macro **OnEndAnim()**, ajouter les instructions :

```
Stroke(mobile)
```

on redessine l'élément mobile en mode normal.

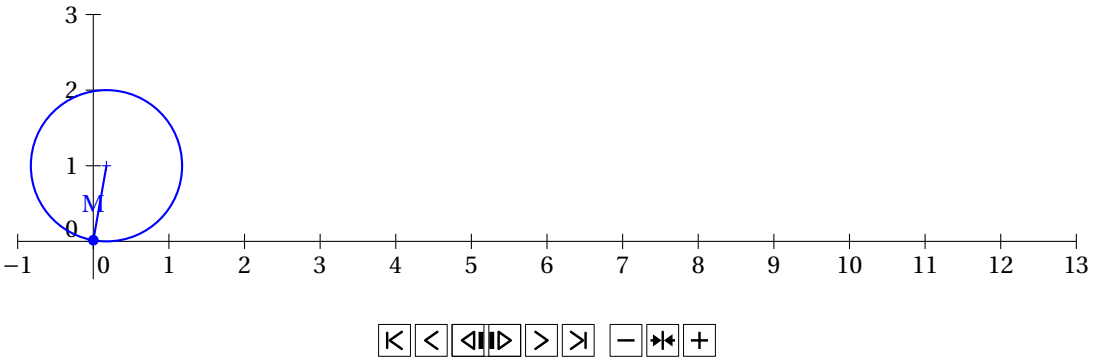
11. Dans la macro **Makeframe()**, ajouter les instructions :

```
if %1>1 then
Inc(Theta, pi/18), Insert(courbe, Theta+i+exp(i*(-pi/2-Theta)) ),
Move(mobile), Stroke(trajectoire)
fi
```

on tourne de 10 degrés, on ajoute la nouvelle position à la liste *courbe*, on déplace la roue et on redessine la trajectoire. La rotation ne commence qu'à la frame 2 de manière à ce que la figure initiale soit enregistrée comme frame 1. Il faudra en tenir compte pour le nombre de frames.

12. Avec le bouton *Frames* définir le nombre de frames à 73, ce qui correspond à 2 tours de roue (donc 2 arches de cycloïde) plus la position initiale.

Ce qui devrait donner :



2 Capture.mod

2.1 Présentation

Ce modèle est prévu pour venir se greffer sur un graphique déjà fait, et permettre à l'utilisateur de capturer des frames quand il le désire. Quand les captures sont terminées, l'utilisateur peut ensuite les compiler en un fichier pdf, et fabriquer un gif animé ou un fichier Flash, comme avec le premier modèle.

On commence par charger son graphique, puis on charge le modèle *Capture.mod* (F3).

Si vous voulez qu'à chaque ouverture de votre graphique, les macros de *Capture.mac* soient automatiquement chargées, vous devez créer une macro *TegWrite* dans votre graphique, avec la commande :

InputMac("Capture.mac")

2.2 Le menu

Celui-ci apparaît dans la liste déroulante, il faut donc cliquer le bouton OK pour activer l'option.

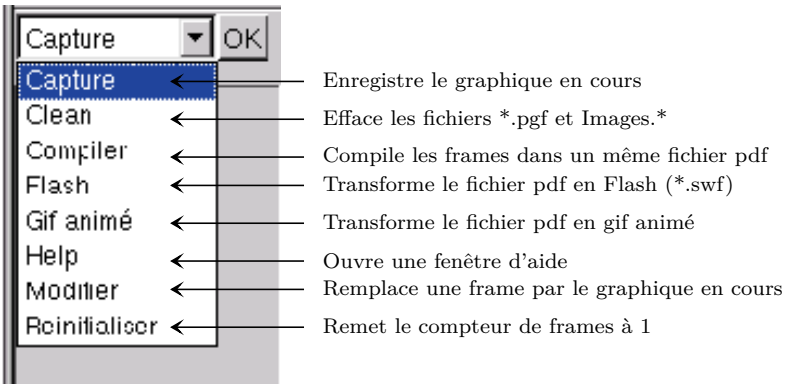


FIGURE 2 – Le menu

- **Capture** : initialement, la variable *NumFrame* a la valeur 1, cette option permet d'enregistrer le graphique en cours dans un fichier *frameXXX.pfg* où XXX désigne la valeur de *NumFrame*, puis cette valeur est incrémentée de 1. Il est possible d'automatiser la capture en appelant directement la macro *CaptureImage()* (sans paramètre).
- **Compiler** : permet de choisir un nom de fichier (avec chemin, mais sans extension), les fichiers « frames » seront alors compilés en un fichier pdf qui portera ce nom, avec une image par page, et chaque page étant à la taille de l'image. Il est possible d'automatiser l'enregistrement en appelant directement la macro à un paramètre *MakePdf(<nombre de frames>)*, le nombre de frames déjà enregistrées est **NumFrame-1**.
- **Modifier** : permet de remplacer une frame par le graphique en cours, le numéro de frame doit être compris entre 1 et le nombre de frames déjà capturées. Par contre le fichier pdf n'est pas modifié, pour cela il faut de nouveau compiler.

2.3 Exemple 3

Nous avons repris le fichier *Hanoi.teg* (que l'on trouve sur le site de TeXgraph dans la galerie) :

- charger ce fichier dans TeXgraph.
- charger le modèle *Capture.mod*.
- cliquer le bouton *Nb disques* et mettre la valeur 4.
- cliquer le bouton *Nouveau* pour mettre la pile dans son état initial.
- faire une première capture avec l'option *Capture* dans la liste déroulante (ne pas oublier de cliquer OK pour valider).
- nous allons maintenant modifier la macro *deplacer()* pour l'obliger à faire deux captures : une après l'affichage de la flèche qui indique le déplacement à effectuer, et une après le déplacement du disque. La macro initiale est la suivante :

```
 [ Si( %1=1, [Set(source, get(T1,1)-l/2+ep*i/2), Set(Tourdep, Tour1)],
       %1=2, [Set(source, get(T2,1)-l/2+ep*i/2), Set(Tourdep, Tour2)],
                 [Set(source, get(T3,1)-l/2+ep*i/2), Set(Tourdep, Tour3)] ),
 Si( %2=1, [Set(but, get(T1,1)-l/2+ep*i/2), Set(Tourbut, Tour1)],
      %2=2, [Set(but, get(T2,1)-l/2+ep*i/2), Set(Tourbut, Tour2)],
                 [Set(but, get(T3,1)-l/2+ep*i/2), Set(Tourbut, Tour3)] ),
 Set($disque, Copy(Tourdep,1,1)), Set($disque2, Copy(Tourbut,1,1)),
 Si( nil(disque2), Set(disque2,n+1)),
  Si( nil(disque)=0,
       Si( disque2<disque, Message("Déplacement impossible"),
         [ Eval(["Del(Tour",%1,",1,1)"]),
           ReCalc(deplacement),
           Eval(["Insert(Tour",%2,",",disque,",1)"]), Delay(Laps),
           Inc(compteur,1), ReCalc(Compteur,disques)
           ]
        )
     )
]
```

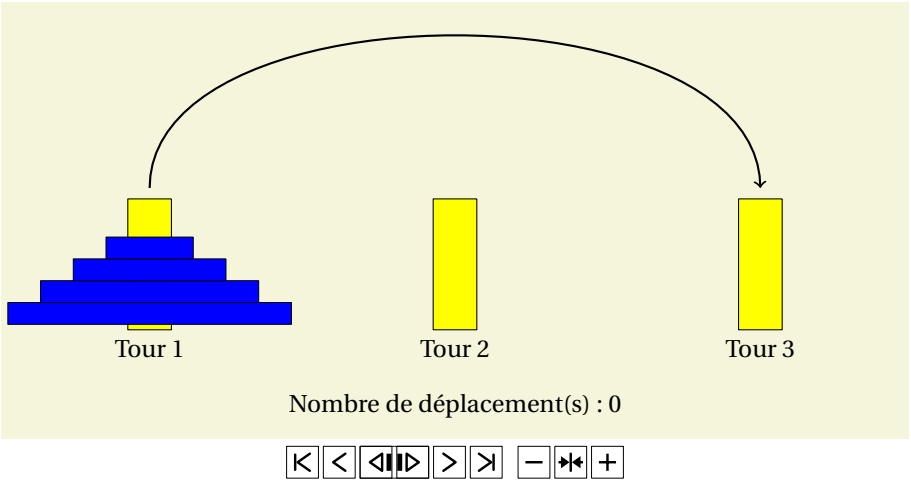
Nous plaçons une instruction *CaptureImage()* après *ReCalc(deplacement)*, et après *ReCalc(Compteur,disques)*, ce qui donne :

```
 [ Si( %1=1, [Set(source, get(T1,1)-l/2+ep*i/2), Set(Tourdep, Tour1)],
       %1=2, [Set(source, get(T2,1)-l/2+ep*i/2), Set(Tourdep, Tour2)],
                 [Set(source, get(T3,1)-l/2+ep*i/2), Set(Tourdep, Tour3)] ),
 Si( %2=1, [Set(but, get(T1,1)-l/2+ep*i/2), Set(Tourbut, Tour1)],
      %2=2, [Set(but, get(T2,1)-l/2+ep*i/2), Set(Tourbut, Tour2)],
                 [Set(but, get(T3,1)-l/2+ep*i/2), Set(Tourbut, Tour3)] ),
 Set($disque, Copy(Tourdep,1,1)), Set($disque2, Copy(Tourbut,1,1)),
 Si( nil(disque2), Set(disque2,n+1)),
  Si( nil(disque)=0,
       Si( disque2<disque, Message("Déplacement impossible"),
         [ Eval(["Del(Tour",%1,",1,1)"]),
           ReCalc(deplacement), CaptureImage(),
           Eval(["Insert(Tour",%2,",",disque,",1)"]), Delay(Laps),
           Inc(compteur,1), ReCalc(Compteur,disques), CaptureImage()
           ]
        )
     )
]
```

- Cliquer ensuite le bouton *Solution*, les frames s'enregistrent toutes seules.
- Il ne reste plus qu'à compiler le tout avec l'option *Compiler* de la liste déroulante. Le fichier obtenu (appelons-le Hanoi.pdf) peut être ensuite inclus dans un document pdf, comme celui-ci :

```
\documentclass[pdftex,12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{fourier}
\usepackage{animate}
\usepackage[margin=2cm]{geometry}
\begin{document}
  \begin{center}
    \animategraphics[loop,controls]{1}{Hanoi}{}{}
  \end{center}
%le fichier Hanoi.pdf va etre entierement inclus
%la valeur 1 indique le nombre d'images par seconde
%l'animation sera en boucle et les boutons visibles.
\end{document}
```

Ce qui donne :


Tour 1
Tour 2
Tour 3
Nombre de déplacement(s) : 0