# **Apprendre le XLogo par l'exemple.** Table des matières

Manuel d'apprentissage du XLogo
conçu et rédigé par
Bernard Gisin
juin 2006

Remarque :
Je suis un débutant en XLogo. J'ai écrit ce manuel en apprenant le langage XLogo. J'espère qu'il peut vous être utile.

Remerciements :
Merci tout particulièrement au créateur de XLogo. Merci également aux diverses personnes qui mettent à disposition des manuels et exemples sur le web.

Références :
1) http://xlogo.free.fr le site de référence du XLogo
2) http://www.algo.be/logo.html un site sur un logo similaire à XLogo, avec quelques différences de syntaxe, mais avec des manuels et exemples bien utiles.
3) http://www.perso.ch/bernard.gisin/xlogo C'est mon site sur le XLogo, avec en particulier ce manuel de XLogo par l'exemple.

# 1. Qu'est ce que " XLogo" et comment l'installer ?

XLogo, c'est quoi ?
Une réponse à cette question se trouve sur : http://xlogo.free.fr/presentation-fr.html

**http://xlogo.free.fr/** est le site officiel du développeur de XLogo.

Vous pouvez télécharger gratuitement le logiciel XLogo sur :
http://xlogo.free.fr/telechargements-fr.html

Comment installer XLogo ?
Vous trouverez une réponse à cette question à la ligne :
- Manuel d'installation : ----> Guide détaillé pour installer xlogo
sur : http://xlogo.free.fr/telechargements-fr.html
Plus directement allez à : http://xlogo.free.fr/fichiers/demarrer.pdf

A cette même adresse vous trouverez un manuel d'utilisation et un tutoriel écrit par le concepteur du logiciel XLogo.

Il est conseillé de lire le tutoriel et de se référer au manuel d'utilisation qui se trouvent à :
http://xlogo.free.fr/telechargements-fr.html

Ce qui suit donne de nombreux exemples, mais ne remplace pas les références ci-dessus.

# 2. Introduction et quelques conventions

Le but de ce qui suit est de montrer comment utiliser XLogo à travers des exemples.

Vous pouvez sauter des chapitres.
La fin de chaque chapitre résume les instructions qui ont été vues.

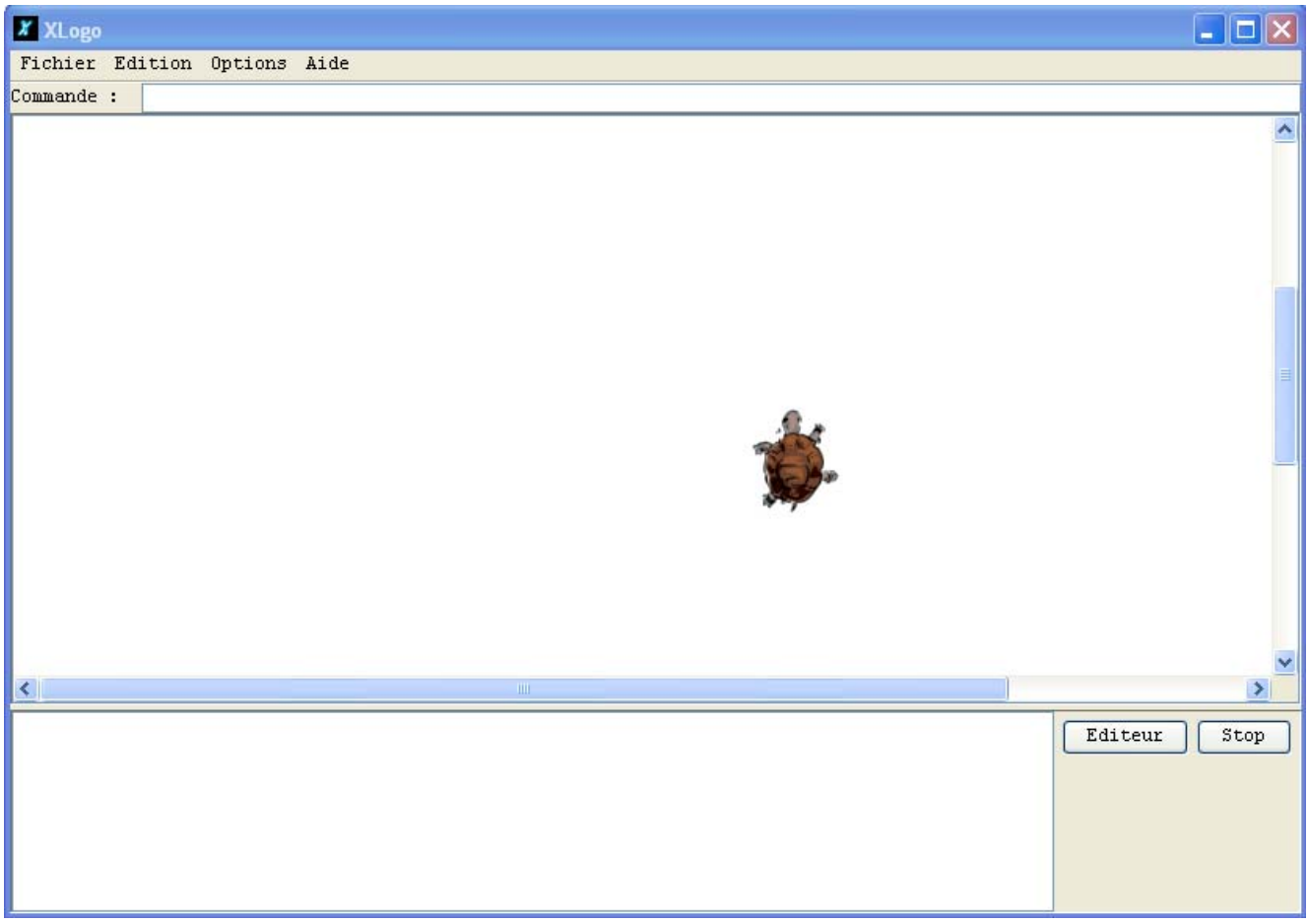
A partir du chapitre :
"4. Les programmes et sous-programmes, l'instruction : Pour ... Fin"
chaque exemple se trouvera dans un fichier que vous pourrez charger et exécuter avec XLogo. Cela vous évitera de devoir retaper les exemples.

Conventions :

Pour indiquer qu'il faut taper un nombre à un endroit donné, le mot *nombre* écrit en *italique* sera écrit à l'endroit où le nombre doit être écrit.
Exemple : **avance** *nombre*
signifie que l'instruction **avance** doit être suivie d'un *nombre* ou d'une variable contenant un nombre.

# 3. Premiers pas avec XLogo

Ici, il est supposé que vous avez lancé le programme XLogo et que vous vous trouvez face à un écran semblable à l'image suivante :




Tapez : av 100 abréviation de AVance 100
et la tortue avancera de 100 unités en traçant une ligne le long de son trajet.

Tapez : tg 90 abréviation de TourneGauche 90
et la tortue tournera à gauche de 90 degrés.

Tapez : av 200
et la tortue avancera de 200 unités en traçant une ligne le long de son trajet.

Tapez : td 26 abréviation de TourneDroite 26
et la tortue tournera à droite de 26 degrés.

Tapez : re 224 abréviation de REcule 224
et la tortue reculera de 224 unités en traçant une ligne le long de son trajet.

Ici un triangle rectangle devrait être dessiné à l'écran.

Tapez : ve abréviation de VideEcran
et le dessin est effacé, avec la tortue revenue au centre, dirigée vers le haut.

Tapez : av 150 tg 120 av 150 tg 120 av 150
et le dessin d'un triangle équilatéral devrait être dessiné à l'écran.

Tapez : lc abréviation de LeveCrayon
Tapez : av 100
et la tortue a avancé, sans tracer de ligne le long de son trajet.

Tapez : bc abréviation de BaisseCrayon
Tapez : av 100 tg 90 av 100 td 45 re 141
et le dessin d'un triangle rectangle isocèle devrait être dessiné à l'écran.

Tapez : ct abréviation de CacheTortue
et la tortue n'est plus visible. Mais elle n'a pas changé de place.
Pour le vérifier, tapez : re 50
et une ligne de longueur 50 unités sera tracée.

Tapez : mt abréviation de MontreTortue
et la tortue est de nouveau visible.

Tapez : ve
pour effacer et ramener la tortue au centre de l'écran.

Tapez : repete 6 [av 100 tg 60]
et le dessin d'un hexagone régulier devrait être dessiné à l'écran.

Pour aller plus loin.

Pour ne pas devoir retaper chaque fois une liste d'instructions, il est avantageux d'écrire les instructions dans un **programme** ou un **sous-programme** ou une **procédure**. Nous ne ferons pas de distinctions entre ces trois mots.

Le chapitre suivant traite des programmes.

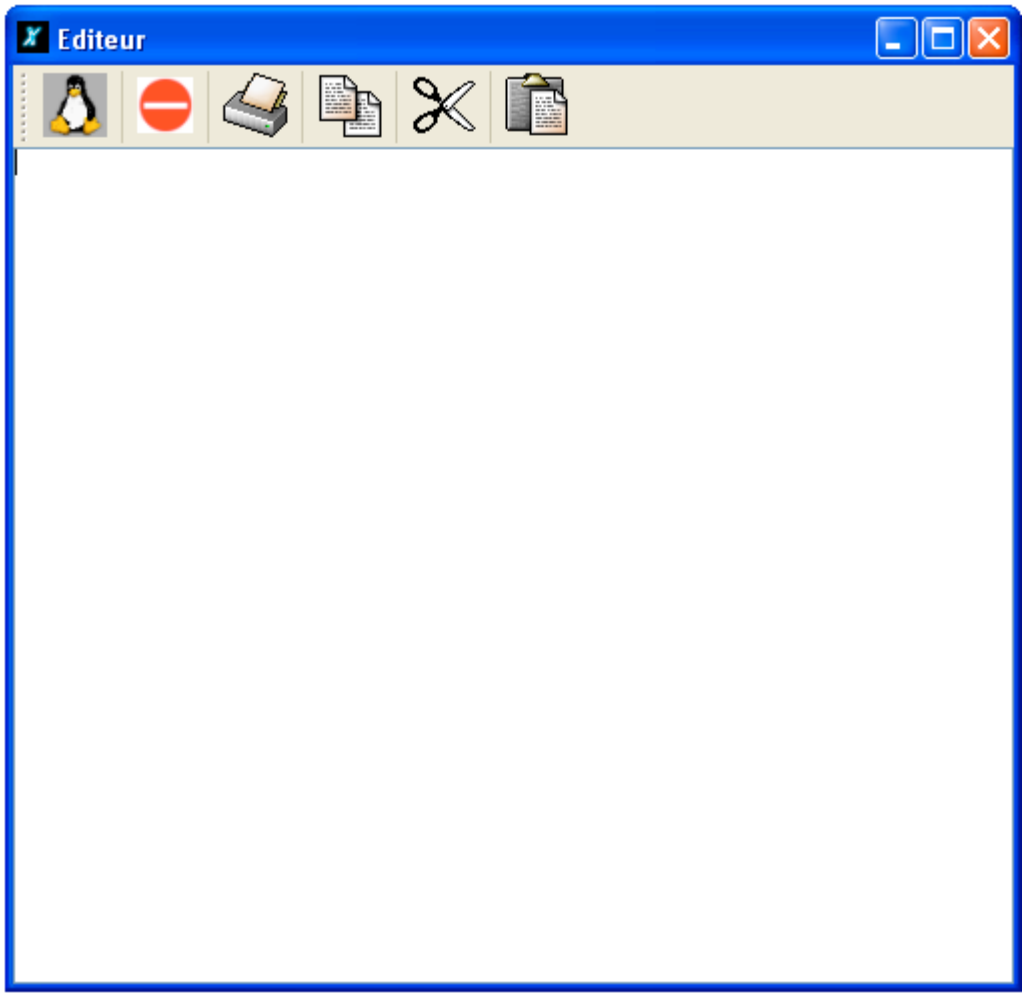
Résumé : Les instructions suivantes ont été vues dans ce chapitre :
C.F. chapitre 4.1 du manuel de XLogo.

av *nombre* abréviation de AVance *nombre*
tg *nombre* abréviation de TourneGauche *nombre*
td *nombre* abréviation de TourneDroite *nombre*
re *nombre* abréviation de REcule *nombre*
ve abréviation de VideEcran
lc abréviation de LeveCrayon
bc abréviation de BaisseCrayon
ct abréviation de CacheTortue
mt abréviation de MontreTortue
repete *nombre* [ *liste d'instructions* ] répète *nombre* fois la *liste d'instructions*.

# 4. Les programmes et procédures, l'instruction : Pour ... Fin

En bas à droite de la fenêtre du programme XLogo, se trouve un bouton nommé "Editeur".
Cliquez dessus pour ouvrir une fenêtre qui vous permettra d'écrire du texte.

Une fenêtre similaire à celle ci-dessous devrait apparaître :




Dans cette fenêtre, tapez les lignes suivantes :

```
Pour essai01
av 100
tg 30
av 100
td 60
re 100
tg 30
re 100
td 90
av 100
fin
```

Ensuite, cliquez sur le bouton  en haut à gauche. La fenêtre disparaît et le texte suivant apparaît dans la fenêtre du bas : "Vous venez de définir essai01."

Tapez : ve essai01
le dessin d'une maison apparaît.
"essai01" est le premier programme décrit dans ce manuel.

Enregistrements de programmes

Vous pouvez sauver sur le disque dur votre premier programme comme suit :
Cliquez sur "Fichier", puis sur "Enregistrer sous...".
Dans la fenêtre qui apparaît, choisissez le dossier dans lequel vous voulez enregistrer votre programme et choisissez le nom du fichier dans lequel vous voulez enregistrer votre programme.
Terminez en cliquant sur le bouton "Enregistrer".

Ouvertures de programmes
Pour ouvrir un programme XLogo qui se trouve sur le disque dur, cliquez sur "Fichier", puis sur "Ouvrir". Dans la fenêtre qui apparaît, choisissez le dossier dans lequel se trouve le fichier contenant le programme désiré, puis cliquez sur le nom du fichier désiré pour terminer en cliquant sur "Ouvrir".

Par exemple, le programme précédent se trouve dans le fichier : "ex_04_01_essai01.lgo"

Les commentaires dans les programmes
Dès que l'on écrit des programmes, il est très important d'écrire des commentaires pour expliquer ce que fait le programme et pour qu'il soit d'apparence plus jolie et plus lisible.

Ajoutons des commentaires dans le programme précédent.

```
Pour essai01
############
# Dessine une maison très simple
av 100
tg 30
av 100
td 60
re 100
tg 30
re 100
td 90
av 100
fin
```

La ligne de ####### sert uniquement à souligner le nom du programme.
La ligne suivante indique ce que fait ce programme.
Pour des petits programmes, cela n'est pas très important, mais rapidement les commentaires permettent de s'y retrouver dans des programmes.

Ce programme se trouve dans le fichier : "ex_04_02_essai01.lgo"

Tapez : ve essai01 essai01
et deux maisons seront dessinées...

Par convention, on appel **procédure** les suites d'instructions entre les mots "Pour" et "Fin".
On appel **programme** l'ensemble de toutes les procédures qui se trouvent dans un fichier.

Il est possible d'avoir plusieurs procédures dans un programme (qui peut s'enregistrer dans un fichier).
Voici un exemple.

```
pour triangle
#############
# Dessine un triangle équilatéral
av 100
tg 120
av 100
tg 120
av 100
fin

pour carre
##########
# Dessine un carré
av 100
tg 90
av 100
tg 90
av 100
tg 90
av 100
fin

pour pentagone
##############
# Dessine un pentagone régulier
av 100
tg  72
av 100
tg  72
av 100
tg  72
av 100
tg  72
av 100
fin
```

Ce programme se trouve dans le fichier : "ex_04_03_figures.lgo"
Vous pouvez soit les taper, soit les ouvrir avec "Fichier", "Ouvrir", ... comme décrit précédemment.

En, cliquant sur le bouton  en haut à gauche de la fenêtre d'éditeur, celle-ci disparaît et le texte suivant apparaît dans la fenêtre du bas : "Vous venez de définir triangle, carre, pentagone."

Vous pouvez dessiner un triangle, un carré et un pentagone, en tapant :
carre ou triangle ou pentagone

Résumé : Ce chapitre montre comment :
C.F. chapitre 4.6 du manuel de XLogo.
1) écrire des programmes, ( Pour ... Fin)
2) ajouter des commentaires dans un programme, ( # )
3) sauver des programmes dans des fichiers, ("Fichier", "Enregistrer sous...")
4) ouvrir des programmes à partir de fichiers. ("Fichier", "Ouvrir")

# 5. Les répétitions d'instructions : repete

Pour effacer les procédures en mémoire, cliquez sur "Fichier", "Nouveau".

Souvent, des instructions doivent être répétées plusieurs fois.
Simplifions nos procédures précédents.

```
pour triangle
#############
# Dessine un triangle équilatéral
repete 3 [av 100   tg 120]
fin

pour carre
##########
# Dessine un carré
repete 4 [av 100   tg 90]
fin

pour pentagone
##############
# Dessine un pentagone régulier
repete 5 [av 100   tg 72]
fin
```

Ces procédures se trouvent dans le fichier : "ex_05_01_repete.lgo"

Cela simplifie beaucoup les programmes.
Remarquez qu'ils ne font pas exactement la même chose que les procédures précédentes, car la tortue termine dans une direction différente dans ces exemples.

```
Voici comment dessiner un hexagone

pour hexagone
#############
# Dessine un hexagone régulier
repete 6 [av 100   tg 60]
fin
```

Résumé : Ce chapitre montre comment
C.F. chapitre 4.12 du manuel de XLogo.
1) Effacer les procédures en mémoire.
2) Répéter plusieurs fois une suite d'instructions. ( repete)

# 6. Les variables

Pour effacer les programmes en mémoire, cliquez sur "Fichier", "Nouveau".

Les procédures précédentes ont toutes des limitations qui les rendent peu intéressantes :
la longueur de chaque côté est fixée lors de la programmation.
La notion de variables permet de s'affranchir de cette limitation. Voici comment :

```
pour triangle :lon
##################
# Dessine un triangle équilatéral de côtés longueur :lon
repete 3 [av :lon   tg 120]
fin

pour carre :lon
###############
# Dessine un carré de côtés longueur :lon
repete 4 [av :lon   tg 90]
fin

pour pentagone :lon
###################
# Dessine un pentagone régulier de côtés longueur :lon
repete 5 [av :lon   tg 72]
fin
```

Ce programme se trouve dans le fichier : "ex_06_01_variables.lgo"

Tapez : ve triangle 200
et le dessin d'un triangle équilatéral de côtés de longueur 200 unités devrait être dessiné à l'écran.

Tapez : ve td 90 triangle 180
et l'orientation du triangle équilatéral de côtés de longueur180 unités change.

Le nombre tapé après le nom de la procédure (triangle) est stocké dans la **variable** :lon.
A l'exécution de la procédure, :lon est remplacé par le nombre stocké dans la variable :lon.

Changement de la valeur de la variable :lon durant l'exécution d'un programme.

Pour effacer les procédures en mémoire, cliquez sur "Fichier", "Nouveau".

Voici un exemple d'utilisation de changement de la valeur de variables.

```
pour triangle :lon
##################
# Dessine un triangle équilatéral de côtés longueur :lon
repete 3 [av :lon   tg 120]
fin

pour multi_triangles :nb :lon
#############################
# Dessine  :nb  triangles équilatéraux
# de côtés longueurs :lon, :lon*1.5, :lon*1.5*1.5,
# :lon*1.5*1.5*1.5, ...
repete :nb [triangle :lon   donne "lon :lon*1.5]
fin
```

Ce programme se trouve dans le fichier : "ex_06_02_variables.lgo"

Tapez : ve multi_triangles 5 100
et le dessin de 5 triangles équilatéraux devrait être dessiné à l'écran.

Cet exemple montre :
1. qu'on peut mettre plusieurs variables comme paramètres d'un programme.
2. qu'on peut utiliser un programme comme sous-programme d'un autre programme.
3. qu'on peut multiplier deux nombre entre eux avec *nombre*1 * *nombre*2.
4. que les nombres à virgule s'écrivent avec un point . à la place de la virgule.
5. qu'on peut modifier la valeur d'une variable avec l'instruction **donne** "*variable nombre*.

Quand on utilise une variable, on met : devant.
Quand on lui donne une nouvelle valeur, on met " devant.

Voici un exemple de programme qui permet de dessiner des polygones.

```
pour polygone :nbcote :lon
##########################
# Dessine un polygone régulier à  :nbcote  côtés
# de côtés longueur :lon
repete :nbcote [av :lon   tg 360/:nbcote]
fin
```

Ce programme se trouve dans le fichier : "ex_06_03_variables.lgo"

Cet exemple montre comment effectuer une division avec le symbole /.

Voici autre exemple de programme qui permet de dessiner des polygones.

```
pour polygone :nbcote :lon
##########################
# Dessine un polygone régulier à  :nbcote  côtés
# de côtés longueur :lon
repete :nbcote [av :lon   tg 360/:nbcote]
fin

pour multi_polygone :nb :nbcote :lon
####################################
# Dessine  :nb  polygones à :nbcote cotés
# de côtés longueurs :lon, :lon*2, :lon*3, :lon*4, ...
#
donnelocale "lon_var :lon  # défini une nouvelle variable
#
# dessin des polygones, presque centrés à la position de la tortue.
repete :nb [
  LeveCrayon
  re :lon_var/2
  tg 90
  re :lon_var*:nbcote/6.28
  td 90
  BaisseCrayon
  polygone :nbcote :lon_var
  attends 60
  LeveCrayon
  tg 90
  av :lon_var*:nbcote/6.28
  td 90
  av :lon_var/2
  BaisseCrayon
  donne "lon_var :lon_var + :lon
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_06_04_variables.lgo"

Cet exemple montre comment définir une nouvelle variable locale à un sous-programme.
( **donnelocale** "*nom_de_variable nombre*)
Il montre aussi comment faire une pause dans un programme. ( **attends** *nombre*)
Le nombre est exprimé en secondes / 60.  "attends 60"  attend une seconde.

Voici un exemple de définition et d'utilisation de **variable globale**.
Une variable globale est une variable qui est utilisable dans plusieurs procédures. Elle peut être définie dans une procédure et utilisée dans une autre.

```
pour carre :lon
###############
# dessine un carré, après avoir avancé d'une longueur
# fixée dans la variable globale :gl_deplace
re :gl_deplace
repete 4 [av :lon tg 90]
fin

pour demarre :nb
################
# défini une variable globale
donne "gl_deplace 5
#
# défini une variable locale
donne "cote 10
repete :nb [carre :cote   donne "cote :cote + 10]
fin
```

Ce programme se trouve dans le fichier : "ex_06_05_var_globale.lgo"

Résumé : Ce chapitre montre comment
C.F. chapitre 4.6 du manuel de XLogo.
1) utiliser des variables, ( :*nom_de_variable* )
2) utiliser des sous-programmes,
3) écrire des nombres à virgule, ( 1.5 )
4) effectuer les quatre opérations, ( + - * / )
5) ajouter une pause dans un programme, ( **attends** *nombre* )
6) donner une autre valeur à une variable, ( **donne** "*nom_de_variable nombre* )
7) définir une nouvelle variable locale à un programme. ( **donnelocale** "*nom_de_variable nombre*)
8) définir une nouvelle variable globale. ( **donne** "*nom_de_nouvelle_variable nombre*)

On peut aussi élever un nombre à une puissance, ( **puissance** *nombre*1 *nombre*2 )

Aux chapitres 11 et 12 il sera vu comment une procédure peut retourner des valeurs.

# 7. Les tests : Si *condition* [ *liste d'instructions* ]

Souvent il est utile de pouvoir tester si une condition est satisfaite. Voici un exemple.

```
pour info
#########
# information sur ce programme
ec [tapez demarre nb    ou  nb  est un nombre]
fin

pour demarre :nb
################
# Effectue un tracé en fonction d'un nombre.
# Le tracé est basé sur la décomposition en base 2 du nombre.
#
donnelocale "res reste :nb 2
#
# répete au maximum 20 fois les instructions suivantes.
repete 20 [
  si (:res = 1) [ tg 45   av 30   td 45 ]
  si (:res = 0) [ td 45   av 30   tg 45 ]
  donne "nb tronque :nb/2
  donne "res reste :nb 2
  #
  tape :res  # écrit les valeurs du reste :res
  tape mot car 32 car 32  # écrit 2 espaces
  ec :nb   # écrit les valeurs de :nb, avec retour à la ligne
  #
  si (:nb = 0) [stop] # test d'arrête pour sortir de la boucle
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_07_01_condition.lgo"

Remarquez que les instructions dans la boucle sont décalées vers la droite, uniquement pour la lisibilité et montrer que ses instructions sont dans une boucle.

Pour simplifier la gestion des procédures, il est bon de commencer par une procédure "**info**" qui fournit de l'information sur comment utiliser ce programme.

Il est aussi agréable de donner toujours le même nom au programme principal.
Par la suite j'appellerai toujours **demarre** la procédure principale.

Résumé : Ce chapitre montre comment
1. effectuer un test, ( **si** (*condition*) [*liste d'instructions*] )
2. éliminer les chiffres après la virgule, ( **tronque** *nombre* )
3. calculer le reste d'une division entière, ( **reste** *nombre*1 *nombre*2 )
4. écrire du texte dans la fenêtre du bas, ( **tape** *nombre* ) ou ( **tape** *texte* )
5. écrire du texte avec un retour à la ligne, ( **ec** *nombre* ) ou ( **ec** *texte* )
6. sortir d'une boucle. ( **stop** )

# 8. Les boucles

Nous avons déjà vu la boucle : **repete** *nombre* [ *liste d'instructions* ].
Voici la boucle : **tantque** [*condition*] [ *liste d'instructions* ].
Tant que la *condition* est vraie, répète la liste d'instructions. Il est habituel de changer la valeur d'une variable dans la *liste d'instructions* et d'effectuer un test sur cette variable dans la *condition*.
Voici un programme qui fait la même chose que le précédent, mais en utilisant l'instruction **tantque**.

```
pour info
#########
# information sur ce programme
ec [tapez demarre nb    où  nb  est un nombre]
fin

pour demarre :nb
################
# Effectue un tracé en fonction d'un nombre.
# Le tracé est basé sur la décomposition en base 2 du nombre.
#
donnelocale "res reste :nb 2
#
tantque [:nb > 0] [
  si (:res = 1) [ tg 45   av 30   td 45 ]
  si (:res = 0) [ td 45   av 30   tg 45 ]
  donne "nb tronque :nb/2
  donne "res reste :nb 2
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_08_01_boucle.lgo"

Voici un autre exemple d'utilisation de **tantque**. Tant que l'utilisateur n'intervient pas, le programme continue de dessiner.

```
pour info
#########
# information sur ce programme
ec [tapez demarre]
fin

pour demarre
############
# Effectue un tracé au hasard.
# s'arrête quand on appuie sur une touche.
#
tantque [non touche?] [
  tg hasard 360  # tg d'un angle au hasard entre 0 et 359
  av 1
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_08_02_boucle.lgo"

Résumé : Ce chapitre montre comment
C.F. chapitre 4.12 du manuel de XLogo.
1. effectuer une boucle avec une condition d'arrêt, ( **tantque** [*condition*] [*liste d'instructions*] )
2. obtenir un nombre entier au hasard, ( **hasard** *nombre* ) retourne un nombre entre 0 et *nombre*-1
3. savoir si le clavier a été pressé, ( **touche?** ) retourne vrai si on a pressé sur le clavier, faux sinon.
   ( **non touche?** ) retourne faux si on a pressé sur le clavier, vrai sinon. C.F. chapitre 11.

# 9. Les instructions de gestion des dessins

Les principales instructions de dessins ont été vues : AVance, REcule, TourneGauche, TourneDroite.
Voici d'autres instructions, qui permettent de changer la couleur des traits dessinés, leur épaisseur,
Pour une liste complète des instructions, référez-vous au manuel.

```
pour info
#########
# information sur ce programme
ec [ce programme montre comment changer la couleur de trait
ec [et leur épaisseur.]
ec [tapez demarre nb    ou  nb  est un nombre]
fin

pour polygone :nbcote :lon
##########################
# Dessine un polygone régulier à  :nbcote  côtés
# de côtés longueur :lon
repete :nbcote [av :lon   tg 360/:nbcote]
fin

pour demarre :nb
################
# Effectue un tracé en fonction d'un nombre.
# Le tracé est basé sur la décomposition en base 2 du nombre.
#
donnelocale "nb_cote 3
#
# répete au maximum 20 fois les instructions suivantes.
repete :nb [
  FixeCouleurCrayon 0  # noire
  FixeTailleCrayon :nb_cote-2
  polygone :nb_cote (50 + :nb_cote)
  #
  FixeTailleCrayon 1
  si (:nb_cote =  3) [FixeCouleurCrayon 1]  # rouge
  si (:nb_cote =  4) [FixeCouleurCrayon 2]  # vert
  si (:nb_cote =  5) [FixeCouleurCrayon 3]  # jaune
  si (:nb_cote =  6) [FixeCouleurCrayon 4]  # bleu
  si (:nb_cote =  7) [FixeCouleurCrayon 5]  # magenta
  si (:nb_cote =  8) [FixeCouleurCrayon 6]  # cyan
  si (:nb_cote >  9) [FixeCouleurCrayon 7]  # blanc
  polygone :nb_cote (55 + :nb_cote)
  donne "nb_cote :nb_cote+1
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_09_01_couleurs_taille.lgo"

`FixeCouleurCrayon` *nombre* fixe la couleur du crayon.
*nombre* est un nombre entier entre 0 et 7

`FixeCouleurCrayon` [*rouge vert bleu*] fixe la couleur du crayon.
*rouge, vert, bleu* sont 3 nombres entier entre 0 et 255 qui fixent la couleur.

`FixeTailleCrayon` *nombre* fixe la taille du crayon.

Il existe deux procédures pour remplir une zone. Voici un exemple qui les illustres.

```
pour figure
###########
# dessine deux figures de couleurs différentes.
FixeCouleurCrayon 2 # vert
lc   td 60   av 50   tg 80   bc
repete 3 [av 150 tg 120]
lc   td 80   re 50   tg 60   bc
# il est important qu'il soit dessiné en 2ème.
# sinon un trait vert couperait le trait rouge
# et rempliszone remplirait tout l'écran !
FixeCouleurCrayon 1  # rouge
repete 6 [av 100 tg 60]
fin

pour remplissage_1
##################
# dessine deux figures de couleurs différentes.
figure
# va dans la figure rouge et attend une seconde
lc   td 80   re 50   bc
attends 60
# remplis la figure rouge
FixeCouleurCrayon 1  # rouge
remplis
# reviens au départ
lc   av 50   tg 80   bc
fin

pour remplissage_2
##################
# dessine deux figures de couleurs différentes.
lc   td 90   av 200   tg 90   bc
figure
# va dans la figure rouge et attend une seconde
lc   td 80   re 50   bc
attends 60
# remplis la figure rouge
FixeCouleurCrayon 1  # rouge
rempliszone  #  remplis tout, jusqu'aux bords rouges !
# reviens au départ
lc   av 50   tg 80   bc
fin

pour demarre
############
remplissage_1  # avec  rempis
remplissage_2  # avec  rempliszone
fin
```

Ce programme se trouve dans le fichier : "ex_09_02_remplis_rempliszone.lgo"

**remplis** remplit la zone où se trouve la souris, jusqu'aux frontières délimitées par une couleur différente de celle du point de départ.
**rempliszone** remplit la zone où se trouve la souris, jusqu'aux frontières délimitées par la couleur du crayon.

Voici un autre exemple de remplissage de figures.

```
pour demi_cercle :c
###################
# trace un demi-cercle de diamètre :c
repete 180 [av  :c*tan 0.5 td 1]
av :c*tan 0.5
td 90    av :c
# Pour remplir le demi-cercle, il faut se déplacer
# à l'intérieur du demi-cercle.
lc   re :c / 2   td 90   av :c / 4   bc
# remplis le demi-cercle
remplis
# va à la position de départ
lc   re :c / 4   tg 90   av :c / 2   td 90   bc
fin

pour demarre
############
donne "rayon 400
donne "couleur 0
repete 8 [
  FixeCouleurCrayon :couleur
  demi_cercle :rayon
  donne "rayon (:rayon -50)
  donne "couleur (:couleur + 1)
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_09_03_remplis.lgo"

Résumé : Ce chapitre montre comment
C.F. chapitre 4.1 et 4.8 du manuel de XLogo.
1. fixer la couleur du crayon ( **FixeCouleurCrayon** *nombre*
2. fixer la couleur du crayon plus finement ( **FixeCouleurCrayon** [*rouge vert bleu*] )
3. fixer la taille du crayon, ( **FixeTailleCrayon** *nombre* )
4. remplir une zone d'une couleur donnée avec **remplis**.
5. remplir une zone d'une couleur donnée avec **rempliszone**.

D'autres instructions utiles de dessins sont :
origine # positionne la tortue à l'originie [0 0]
FixePOSition [*x_pos y_pos*]
fixexy [*x_pos y_pos*] # identique à FixePOSition
fixex *x_pos*
fixey *y_pos*
fixecap *angle* # fixe le cap de la tortue, c'est-à-dire sa direction relativement à la verticale.
pos # retourne une liste [*x_pos y_pos*] indiquant la position de la tortue.
cap # retourne un nombre angle indiquant le cap (la direction) de la tortue.
CouleurCrayon # retourne la couleur du crayon.
etc. # c.f. Déplacement de la tortue, gestion du crayon et des couleurs

# 10. L'affichage de textes

Voici un exemple qui illustre comment afficher du texte dans la fenêtre du bas :

```
pour demarre
############
# Exemple d'affichage de texte dans la fenetre du bas
donne "prenom [Bernard]
donne "nom "Gisin
# Les deux manières de faire ci-dessus sont équivalentes.
fixestyle [gras souligne]
ecris [Bonjours, voici un exemple d'affichage de texte.]
fixestyle [aucun]
# Dans la version 9_21f  de XLogo, tout le texte affiche est dans le
dernier style indique
# C'est une erreur qui sera corrigee, a mon avis.
tape  [Mon nom complet est :\ ]  #  "\ "  insert in espace
tape  :prenom
tape  [\ ]  # insert un espace dans le texte affiche.
ecris :nom
fin
```

Ce programme se trouve dans le fichier : "ex_10_01_ecris.lgo"

"ECris" permet d'écrire du texte dans la fenêtre du bas, avec un retour à la ligne à la fin.
"tape" est identique à "ecris", sauf qu'il n'y a pas de retour à la ligne en fin d'écriture.

Certains caractères spéciaux s'obtiennent avec le caractère \.
*espace* où *espace* est simplement un espace, permet d'insérer un espace.
\# permet d'insérer le caractère #.
\( permet d'insérer le caractère (.
\) permet d'insérer le caractère ).
\[ permet d'insérer le caractère [.
\] permet d'insérer le caractère ].
\\ permet d'insérer le caractère \.
\n permet d'insérer un retour à la ligne.

La gestion de listes, décrite au chapitre 12, permet de simplifier le programme en écrivant plusieurs arguments après les instructions "ecris" et "tape".

Voici un exemple qui illustre comment afficher du texte à la position de la souris :

```
pour demarre
############
# Exemple d'affichage de texte a la position de la souris
donne "prenom [Bernard]
donne "nom "Gisin
# Les deux manières de faire ci-dessus sont équivalentes.
LeveCrayon
CacheTortue
fixexy (-200) (100)  # se positionne dans la fenetre.
etiquette [Bonjours, voici un exemple d'affichage de texte.]
re 20  #descend
etiquette  [Mon nom complet est :\ ]  #  "\ "  insert in espace
re 20  #descend
etiquette  :prenom
#etiquette  [\ ]  # insert un espace dans le texte affiche.
re 20  #descend
etiquette :nom
fin
```

Ce programme se trouve dans le fichier : "ex_10_02_etiquette.lgo"

"etiquette" permet d'afficher du texte dans la fenêtre de la tortue, à la position de la tortue.

Voici un exemple qui permet d'afficher une fenêtre avec un message :

```
pour demarre
############
# Exemple d'affichage d'une fenetre contenant un message
message [Bonjours, voici un exemple de message.]
fin
```

Ce programme se trouve dans le fichier : "ex_10_03_message.lgo"

"message" permet d'afficher une fenêtre avec un message.

Voici un exemple qui permet d'afficher des nombres de manière formatée, c'est-à-dire en contrôlant la place qu'ils prennent à l'affichage.

```
pour longueur_du_nombre :nb
############################
# :nb  est un nombre
# retourne le nombre de chiffres avant la virgule, 1 de plus s'il
est negatif.
donne "long 1  # nombre de chiffre de :nb
si (:nb < 0) [donne "long 2  donne "nb (-:nb)]
#
tantque [ou (:nb > 10) (:nb = 10)] [donne "long :long +1  donne "nb
:nb/10]
retourne :long
fin

pour affiche :nb :long :virgule
###############################
# affiche le nombre  :nb  en prenant
# :long  caracteres avant la virgule
# :virgule chiffres aprÃ¨s la virgule
#
# affiche les espaces avant le nombre
repete (:long - (longueur_du_nombre :nb)) [tape "\ ]
#
# transforme le nombre pour changer le nombre de chiffres apres la
virgule
donne "base puissance 10 :virgule
donne "nb (arrondi :base * :nb)/:base
#
# affiche le nombre
tape :nb
fin

pour demarre :nb
################
# affiche le nombre avec 6 caracteres avant la virgule et 2 chiffres
apres.
affiche :nb 6 2
ecris "\
fin
```

Ce programme se trouve dans le fichier : "ex_10_04_ecris_formatte.lgo"

Résumé : Ce chapitre montre comment
C.F. chapitre 4.1 et 4.2 du manuel de XLogo.
1. écrire du texte dans la fenêtre du bas ( **ECris** *liste* ou **tape** *liste* )
2. fixer le style du texte ( **FixeStyle** [*divers styles*] )
   Styles possibles : aucun, gras, italique, barre, indice, exposant, souligne
   C.F. chapitre 4.2 du manuel de XLogo.
3. afficher du texte à la position de la tortue. ( **etiquette** *liste* )
4. afficher un message dans une boite de dialogue ( **message** *liste*)

# 11. Les interactions avec l'utilisateur

Voici comment interagir avec l'utilisateur à travers une boîte de dialogue.

```
pour demarre
############
# Exemple d'interaction avec l'utilisateur en posant des questions.
lis [Quel est ton age ?] "age
# :age contient alors une liste a un element,
# on extrait cet element et on le stocke dans :age
donne "age premier :age
si non (nombre? :age) [ecris [tu n'as pas ecris correctement ton
age]  stop]
#
lis [Quel est ton prénom ?] "prenom
#
tape [Bonjour\ ]
tape :prenom
tape [, ton age est :\ ]
ecris :age
# test si tu es majeur ou mineur.
si ou :age>18 :age=18 [ecris [Tu es majeur.]] [ecris [Tu es
mineur.]]
fin
```

Ce programme se trouve dans le fichier : "ex_11_01_lis.lgo"

"lis" ouvre une boîte de dialogue avec une texte dans la ligne de titre,
dans laquelle on peut écrire un texte.
"si" permet de tester une condition. Cette instruction a déjà été vue au chapitre 7.

Voici comment tester si une touche a été pressée et déterminer quelle est la touche pressée.

```
pour demarre
############
# Exemple d'interaction en lisant si une touche a ete pressee.
ecris [utilises les fleches pour deplacer la tortue, la touche ESC
pour terminer]
donne "car 28
#
tantque [non (:car = 27)] [
  si touche? [
    donne "car liscar
    si :car=-37 [tg 90]
    si :car=-39 [td 90]
    si :car=-38 [av 10]
    si :car=-40 [re 10]
    si :car=27 [stop]
    ]
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_11_02_liscar.lgo"
"touche?" est vrai si une touche a été pressée.
"liscar" retourne le code numérique de la touche pressée.

-37 = code de la flèche -> -39 = code de la flèche <-
-38 = code de la flèche ^ -40 = code de la flèche V 27 = code de la touche ESC

Autre manière de faire, plus facile à manipuler à mon avis.

```
pour demarre
############
# Exemple d'interaction en lisant si une touche a ete pressee.
ecris [utilises les fleches pour deplacer la tortue, la touche ESC
pour terminer]
donne "car 28
#
tantque [non (:car = 27)] [
  si touche? [
    donne "car liscar
    si :car=-37 [fixecap 270]
    si :car=-39 [fixecap  90]
    si :car=-38 [fixecap   0]
    si :car=-40 [fixecap 180]
    si :car=27 [stop]
    av 10
    ]
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_11_03_liscar.lgo"

"touche?" teste si une touche du clavier a été pressée.
"liscar" lit le code du caractère pressé.

Voici comment utiliser la souris.

Le programme doit encore être amélioré de ce point de vu, car on ne peut pas gérer les boutons pressé ou non, en même temps que le déplacement de souris.

```
pour demarre
############
# Si on deplace la souris, se positionner a la nouvelle position
ecris [deplacez la souris. La touche ESC termine le programme.]
donne "car 28
fixexy 0 0
#
tantque [non (:car = 27)] [
  si touche? [donne "car liscar]
  si :car=27 [stop]
  si lissouris=0 [fpos possouris]
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_11_04_lissouris.lgo"

**lissouris** rend un nombre permettant de caractériser l'événement. Voici les différents codes associés aux différents événements qu'ils représentent :
0 –> on a déplacé la souris
1 –> on a appuyé sur le bouton 1 de la souris
2 –> on a appuyé sur le bouton 2 de la souris
etc.
Les boutons sont numérotés de la gauche vers la droite (en principe...)

**possouris** : Renvoie une liste contenant les coordonnées de la souris lors du dernier événement intercepté.

**souris?** : rend vrai ou faux selon que l'on ait agi ou non sur la souris depuis le début de l'exécution du programme.

Résumé : Ce chapitre montre comment
C.F. chapitre 4.13 du manuel de XLogo.
1. acquérir du texte de l'utilisateur, depuis une boîte de dialogue ( **lis** *liste* "*variable* )
2. test si une touche a été pressée ( **touche?** )
3. lit la touche pressée ( **liscar** )
4. lit un événement souris ( **souris?** )
5. lit l'action lié à l'événement souris ( **lissouris** )
6. lit la position de la souris ( **possouris** )

Deux instructions utiles du chapitre 4.4 du manuel XLogo sont :
**unicode** "*lettre* retourne le code de la lettre.
**caractere** *nombre* retourne le caractère dont le code est *nombre*.

# 12. Les listes

La gestion de listes est importante dans le programme XLogo.
Voici un premier exemple.

```
pour demarre
############
# Exemples de traitement de mots et de listes
donne "mot1 "ceci_est_un_mot
donne "mot2 "ceci\ est\ un\ mot
# les "\ " permettent d'inserer des espaces dans les mots.
donne "mot3 "\ autre_mot
# :mot3 est un mot qui commence par un espace
donne "mot4 "espace\ parentheses\(\)\[\]_slash\\_retour\n_fin
# un mot avec beaucoup de caracteres speciaux.
# "\n" signifie retour a la ligne
#
donne "mot5 mot :mot2 :mot3
# cree un nouveau mot en collant :mot2 et :mot3
#
donne "liste1 liste :mot2 :mot3
# cree une liste contenant les deux mots :mot2 et :mot3
#
donne "liste2 [ceci est une liste de sept mots]
# cree une liste contenant les deux mots :mot2 et :mot3
#
donne "liste3 phrase :liste1 :liste2
# cree une liste en joignant les deux listes :liste1 et :liste2
#
ecris :mot1
ecris :mot2
ecris :mot3
ecris :mot4
ecris :mot5
ecris :liste1
ecris compte :liste1  # c'est une liste de 2 mots
ecris :liste2
ecris compte :liste2
ecris :liste3
ecris compte :liste3
fin
```

Ce programme se trouve dans le fichier : "ex_12_01_mots.lgo"

Un **mot** est une suite de caractères.
Une **liste** est une suite de mots. Ils sont mis entre crochet [] et séparés par des espaces.
L'instruction **mot** crée un nouveau mot à partir de deux autres.
L'instruction **liste** crée une liste à partir de deux mots.
L'instruction **phrase** crée une liste à partir de deux listes.
Le chapitre 4.4 du manuel de XLogo donne beaucoup d'instructions permettant de manipuler des listes et des mots.

En informatique il est très utile de manipuler des tableaux de nombres. Les listes de XLogo permettent de le faire. Voici un exemple, qui calcule les premiers nombres premiers.

```
pour demarre
############
# Utilise le crible d'Eratosthene pour trouver les premier nombres
premier.
donne "Table [0]  # liste de nombres
donne "nbrMaxRacine 32
donne "nbrMax :nbrMaxRacine * :nbrMaxRacine
#
# rempli le tableau des nombres de 2 à nbrMax
donne "nbr 1
repete :nbrMax-1 [
  donne "nbr :nbr+1
  donne "Table metsdernier :nbr :Table
  ]
#
donne "nbr2 2
#
# boucle pour eliminer tous les nombres non premiers de la table.
tantque [:nbr2 < :nbrMaxRacine] [
  #
  # Elimine tous les multiple de nbr2 de la table
  donne "nbr1 2 * :nbr2
  tantque [:nbr1 < :nbrMax] [
    donne "Table remplace :Table :nbr1 0
    donne "nbr1 :nbr1 + :nbr2
    ]
#  ec :Table
  # recherche le prochain nombre premier de la liste
  donne "nbr2 :nbr2 + 1
  tantque [ (item :nbr2 :Table) = 0] [donne "nbr2 :nbr2+1]
  ]
#
# ec :Table
# tous les nombres non nuls sont premiers. Ecrit-les.
donne "nbr1 1
tantque [:nbr1 < :nbrMax] [
  si (non (item :nbr1 :Table) = 0) [tape item :nbr1 :Table  tape "\
]
  donne "nbr1 :nbr1 + 1
  ]
fin
```

Ce programme se trouve dans le fichier : "ex_12_02_nombres_premiers.lgo"

Résumé : Ce chapitre montre comment
C.F. chapitre 4.4 du manuel de XLogo.
1. manipuler des mots ( **mot** :*mot1 :mot2* )
2. manipuler des listes ( **liste** :*mot1 :mot2* et **phrase** :*liste1 :liste2* )
3. ajoute un élément en fin de liste ( **metdernier** :*mot :liste* )
4. récupère un élément d'une liste ( **item** :*nombre :liste* )
5. remplace un élément d'une liste ( **remplace** *:liste* :*nombre :mot*)

# 13. La récursivité

La récursivité est une notion assez difficile, mais qui permet d'obtenir en peu d'instructions des effets assez complexes.

Une définition est dite récursive si la notion définie fait appelle à elle-même.

La récursivité est le fait de décrire un processus dépendant de données, en faisant appel à ce même processus sur d'autres données plus "simples".

Voici des exemples pour comprendre.
En informatique, *un **dossier** est défini par la propriété qu'il peut contenir des **fichiers** et des **dossiers***.
Donc le mot dossier est défini en utilisant le mot dossier ! Il semble que l'on tourne en rond.
Mais les définitions plus précises suivantes lèvent l'ambiguïté.

*Un **dossier de niveau 0** est défini par la propriété qu'il peut contenir des **fichiers***.

*Un **dossier de niveau 1** est défini par la propriété qu'il peut contenir des **fichiers et des dossiers de niveau 0***.

*Un **dossier de niveau 2** est défini par la propriété qu'il peut contenir des **fichiers, des dossiers de niveau 0 et des dossiers de niveau 1***.

*Un **dossier de niveau 3** est défini par la propriété qu'il peut contenir des **fichiers, des dossiers de niveau 0, des dossiers de niveau 1 et des dossiers de niveau 2***.

Etc. Cet exemple qu'il y a plusieurs types de dossiers, de complexité croissante.

Voici un autre exemple de définition récursive.
*La **factorielle d'un nombre** égale le nombre fois la factorielle du prédécesseur du nombre.*
La factorielle d'un nombre entier naturel $n$ s'écrit : $n!$
Sous forme de formule, la définition devient : $n! = n \cdot (n - 1)!$
Donc
$4! = 4 \cdot 3!$ $3! = 3 \cdot 2!$ $2! = 2 \cdot 1!$ $1! = 1 \cdot 0!$ quand s'arrête-t-on ?

Pour être utile, la définition doit être complétée.
Par définition, **$0! = 1$**.
Donc
$1! = 1 \cdot 0! = 1 \cdot 1 = 1$
$2! = 2 \cdot 1! = 2 \cdot 1 = 2$
$3! = 3 \cdot 2! = 3 \cdot 2 = 6$
$4! = 4 \cdot 3! = 4 \cdot 6 = 24$
$5! = 5 \cdot 4! = 5 \cdot 24 = 120$

Programmons cela avec XLogo :

```
pour factorielle :nb
####################
# calcule la factorielle du nombre :nb
si (:nb < 1) [retourne 1]
retourne :nb * (factorielle (:nb - 1))
fin

pour demarre :nb
#################
ec factorielle :nb
fin
```

Ce programme se trouve dans le fichier : "ex_13_01_factorielle.lgo"

Tapez : demarre 5 et la réponse 120 s'affiche en bas.
Tapez : demarre 4 et la réponse 24 s'affiche en bas.
Tapez : demarre 1 et la réponse 1 s'affiche en bas.

Cet exemple n'est que démonstratif, car une meilleure manière de calculer la factorielle d'un nombre est la suivante.

```
pour demarre :nb
#################
# calcule la factorielle du nombre :nb
donnelocale "res 1
tantque [:nb > 0] [
  donne "res (:nb * :res)
  donne "nb (:nb - 1)
  ]
ec :res
fin
```

Quand on peut éviter simplement une définition récursive, mieux vaut le faire.

Voici un autre exemple plus complexe de procédure récursive.
La suite de Fibonacci est : 1 1 2 3 5 8 13 21 34 55 89 ...
La suite commence par 1 1, puis chaque nombre est la somme des deux précédents.

Voici une définition récursive de la suite de Fibonacci.

```
pour fibonacci :nb
###################
# calcule le nombre de fibonacci numéro :nb
si (:nb < 2) [retourne 1]
retourne (fibonacci (:nb -1)) + (fibonacci (:nb -2))
fin

pour demarre :nb
#################
ec fibonacci :nb
fin
```

Ce programme se trouve dans le fichier : "ex_13_03_fibonacci.lgo"

Tapez : demarre 7 et la réponse 13 s'affiche en bas.
Tapez : demarre 9 et la réponse 55 s'affiche en bas.

Cet exemple peut être programmé sans récursivité, mais de manière plus compliquée.

```
pour demarre :nb
################
# calcule le nombre de fibonacci numéro :nb
donnelocale "res 1
donnelocale "pre 1
donnelocale "tmp 1
#
tantque [:nb > 1] [
  donne "tmp :res  # mémorise la valeur de :res
  donne "res (:res + :pre)
  donne "pre :tmp  # :pre prend la valeur qu'avait :res
  donne "nb (:nb -1)
  ]
ec :res
fin
```

Ce programme se trouve dans le fichier : "ex_13_04_fibonacci.lgo"

Voici un autre exemple plus complexe de procédure récursive.
Le triangle de Pascal est :

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5 10 10  5  1
1  6 15 20 15  6  1
```

Chaque nombre est la somme du nombre au-dessus de lui et au-dessus à gauche de lui.

Voici une définition récursive du nombre de la ligne :lin et colonne :col du triangle de Pascal.

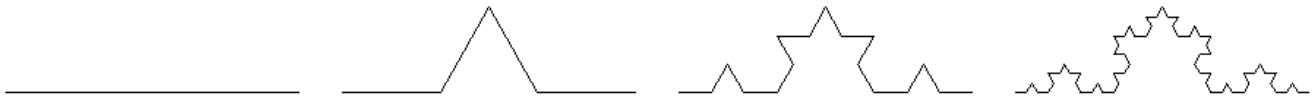
```
pour pascal :lin :col
#####################
# calcule le nombre de la ligne :lin et :col du triangle de Pascal
si (:col < 2) [retourne 1]  # les nombres de la 1ère col. égalent 1
si (:lin < 2) [retourne 0]  # les nombres de la 1ère ligne. égalent
0, sauf celui de la première colonne
retourne (pascal (:lin -1) :col) + (pascal (:lin -1) (:col - 1))
fin

pour demarre :lin :col
######################
ec pascal :lin :col
fin
```

Ce programme se trouve dans le fichier : "ex_13_05_pascal.lgo"

Tapez : demarre 7 4 et la réponse 20 s'affiche en bas.
Cette fois ci, une version non récursive est pénible à écrire, quoi que possible.
Il sera mis peut-être une fois dans "ex_13_06_pascal.lgo".

Voici une utilisation de la récursivité pour dessiner des fractales.

Voici quatre étapes d'une ligne de Koch :



Le premier segment en subdivisé en 3, celui du milieu est remplacé par un triangle équilatéral, la base étant effacée.
Chacun des 4 segments ainsi obtenu est subdivisé en 3, celui du milieu est remplacé par un triangle équilatéral, la base étant effacée.
Chacun des 16 segments ainsi obtenu est subdivisé en 3, celui du milieu est remplacé par un triangle équilatéral, la base étant effacée. etc.

```
pour koch_sub1 :ordre :lon
##########################
# Dessin d'une ligne subdivisée en lignes.
si (:ordre < 1) | (:lon < 1) [avance :lon stop]
koch_sub1 :ordre-1 :lon/3
tg 60
koch_sub1 :ordre-1 :lon/3
td 120
koch_sub1 :ordre-1 :lon/3
tg 60
koch_sub1 :ordre-1 :lon/3
fin

pour demarre :ordre
###################
# Dessine un flocon de Koch.
# demarre 1  =>  un triangle equilateral.
# demarre 2  =>  chaque cote est subdivise en 4 parties.
# demarre 7  est la limite.
si (:ordre > 7) [ donne  "ordre  7]
koch_sub1 :ordre-1 300
td 120
koch_sub1 :ordre-1 300
td 120
koch_sub1 :ordre-1 300
td 120
fin
```

Ce programme se trouve dans le fichier : "ex_13_07_Koch_triangle.lgo"

Tapez : tg 90  Koch_sub1 0 200  pour obtenir le premier segment.
Tapez : tg 90  Koch_sub1 1 200  pour obtenir la deuxième figure de 4 segments.
Tapez : tg 90  Koch_sub1 2 200  pour obtenir la troisième figure de 16 segments.
Tapez : tg 90  Koch_sub1 3 200  pour obtenir la quatrième figure de 64 segments.

Tapez : demarre 1  pour obtenir un triangle équilatéral.
Tapez : demarre 4  pour que chaque face soit décomposée en 64 segments de Koch.
On obtient un flocon de Koch.
Remarquez que Koch_sub1 est une procédure qui s'appelle 4 fois. Sans récursivité, ce même dessin est beaucoup plus difficile à obtenir.

Voici d'autres programmes utilisant de la récursivité pour dessiner des fractales.

```
pour kochc_s1 :ordre :lon
#########################
si (:ordre < 1) | (:lon < 1) [avance :lon stop]
kochC_s1 :ordre-1 :lon/3
tg 90
kochC_s1 :ordre-1 :lon/3
td 90
kochC_s1 :ordre-1 :lon/3
td 90
kochC_s1 :ordre-1 :lon/3
tg 90
kochC_s1 :ordre-1 :lon/3
fin

pour kochc :ordre
#################
# Dessine un flocon de Koch base sur un carre.
# KochC 1  =>  un carre.
# KochC 2  =>  chaque cote est subdivise en 5 parties.
# KochC 6  est la limite.
#
si (:ordre > 6) [ donne  "ordre  6]
kochC_s1 :ordre-1 200
td 90
kochC_s1 :ordre-1 200
td 90
kochC_s1 :ordre-1 200
td 90
kochC_s1 :ordre-1 200
td 90
fin
```

Ce programme se trouve dans le fichier : "ex_13_08_Koch_carre.lgo"
Il réalise la même idée que le flocon de Koch, mais sur la base d'un carré au lieu d'un triangle.

D'autres exemples se trouvent dans :
"ex_13_09_Sierpinski_carre.lgo" *demarre 4* dessine un tapis de Sierpinski.
"ex_13_10_Sierpinski_triangle.lgo" *demarre 4* dessine un tapis triangulaire de Sierpinski.
"ex_13_11_Sierpinski_carre_5.lgo" *demarre 3* dessine un tapis de Sierpinski avec moins de trous.
"ex_13_12_Penrose_pavage.lgo" *demarre 5* dessine un pavage de Penrose dans un triangle.
"eex_13_13_Peano_courbe.lgo" *demarre 4* dessine une courbe de type Peano.

Résumé : Ce chapitre montre comment
1) définir une procédure récursive, La procédure s'appelle elle-même.

Pour plus d'information sur ce sujet, cherchez "récursivité" sur le site de wikipedia :
http://fr.wikipedia.org/wiki/Accueil
Un cours sur la récursivité avec beaucoup d'exemples se trouve sur :
http://chambily.com/recursivite/

# 14. Les instructions musicales

Voici comment simplement jouer de la musique avec XLogo.
Ce premier exemple joue la gamme standard, puis sur trois octaves.
En fin de chapitre, j'indique comment simuler différents instruments de musique.
Ces exemples sont tirés et inspirés du manuel de référence de XLogo.

```
pour demarre
############
# Joue la gamme, les 12 notes, puis joue la gamme sur 3 octaves.
#
# 1.0 = des blanches   ;   0.5 = des noires   ;   0.25 = des croches
# :- = une octave plus bas   ;    :+ = une octave plus haut
# do+ = do dièse = ré- = ré bémole   ;   re+ = ré dièse = mi- = mi
bémole  ; etc.
#
efseq       # On efface la séquence actuellement en mémoire
sequence [0.5 do re mi fa sol la si :+ 1.0 do]  # octave standard.
joue   # joue la séquence en mémoire
#
attends 60 # attend une seconde
efseq       # On efface la séquence actuellement en mémoire
sequence [0.5 do do+ re re+ mi fa fa+ sol sol+ la la+ si :+ 1.0 do]
# octave standard.
joue   # joue la séquence en mémoire
#
attends 60 # attend une seconde
efseq       # On efface la séquence actuellement en mémoire
sequence [:- 0.5 do re mi fa sol la si]
sequence [   0.5 do re mi fa sol la si]  # octave standard.
sequence [:+ 0.5 do re mi fa sol la si :+ 1.0 do]
joue   # joue la séquence en mémoire
#
# Vous pouvez également changer d'instruments,
# soit à l'aide de la commande finstr soit dans le menu
# Options-Préférences-Onglet son.
# Vous trouverez la liste de tous les instruments disponibles
# avec leur numéro
# C'est en anglais, mais ça permet de se donner une idée.
# Il est possible d'avoir plus de 400 instruments disponibles!
fin
```

Ce programme se trouve dans le fichier : "ex_14_01_Gamme_3_octaves.lgo"
Tapez "demarre" pour entendre la musique.

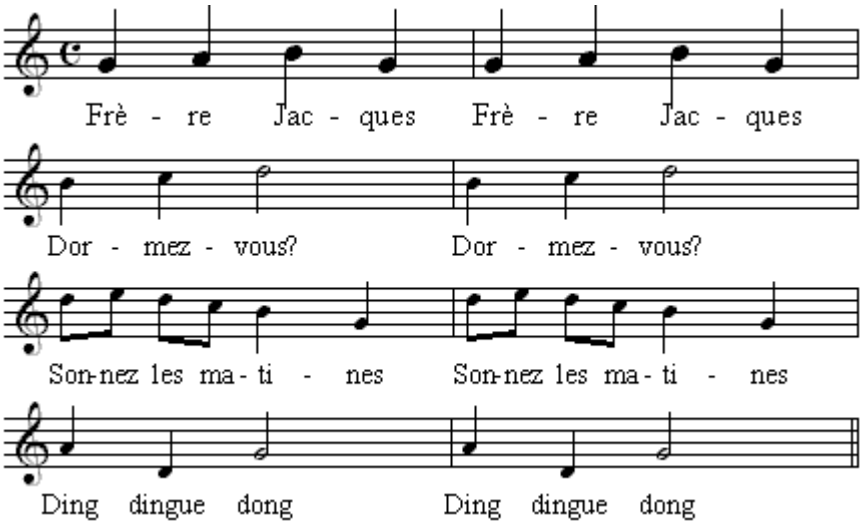
"efseq" efface la séquence en mémoire.
"sequence [liste]" met en mémoire la partition.

Dans la liste, on peut y mettre les notes de la gamme "do re mi fa sol la si"
1.0 indique que les notes qui suivent sont des blanches.
0.5 indique que les notes qui suivent sont des noires.
0.25 indique que les notes qui suivent sont des croches.
:- indique qu'on passe à l'octave en dessous.
:+ indique qu'on passe à l'octave en dessus.
Une note suivie d'un + est la note dièse.
Une note suivie d'un - est la note bémole.
Les 12 notes de la gamme sont : do do+ re re+ mi fa fa+ sol sol+ la la+ si

Ce deuxième exemple joue la musique de "Frère Jacques".

```
pour frere_jacques_1
####################
# Met en mémoire la partition
sequence [0.5 do re mi do do re mi do]
sequence [0.5 mi fa 1.0 sol 0.5 mi fa 1.0 sol]
sequence [0.25 sol la sol fa 0.5 mi do 0.25 sol la sol fa 0.5 mi do]
sequence [0.5 re :- sol :+ 1.0 do 0.5 re :- sol :+ 1.0 do]
fin

pour frere_jacques_2
####################
# Même partition, transposée quelques notes plus haut.
# Met en mémoire la partition
sequence [0.5 sol la si sol  sol la si sol]
sequence [0.5 si :+ do 1.0 re :- 0.5 si :+ do 1.0 re :-]
sequence [:+ 0.25 re mi re do :- 0.5 si sol :+ 0.25 re mi re do :-
0.5 si sol]
sequence [0.5 la re 1.0 sol 0.5 la re 1.0 sol]
fin

pour demarre :nb
#################
# Joue la mélodie : "Frère Jacques"
#
efseq       # On efface la séquence actuellement en mémoire
si (:nb = 1) [Frere_jacques_1]    # On charge la musique
si (:nb = 2) [Frere_jacques_2]    # On charge la musique, plus haut.
joue   # joue la séquence en mémoire
fin
```

Ce programme se trouve dans le fichier : "ex_14_02_Frere_Jacques.lgo"
Tapez "demarre 1" pour entendre la musique de "Frère Jacques".
Tapez "demarre 2" pour entendre la musique de "Frère Jacques", une quinte plus haut.



Ce troisième exemple joue la musique de "Frère Jacques" en canon.

```
pour frere_jacques_1
####################
# Met en mémoire la partition
sequence [0.5 do re mi do do re mi do]
sequence [0.5 mi fa 1.0 sol 0.5 mi fa 1.0 sol]
sequence [0.25 sol la sol fa 0.5 mi do 0.25 sol la sol fa 0.5 mi do]
sequence [0.5 re :- sol :+ 1.0 do 0.5 re :- sol :+ 1.0 do]
fin

pour demarre
############
# joue frère_Jacques en canon.
efseq             # On efface la séquence actuellement en mémoire
Frere_jacques_1   # On charge la musique
findseq 4         # On décale de 4 temps la mélodie
Frere_jacques_1   # On recharge la même séquence décalée de 4 temps.
joue              # joue la séquence en mémoire
fin
```

Ce programme se trouve dans le fichier : "ex_14_03_Frere_Jacques.lgo"
Tapez "demarre" pour entendre la musique.

Ce quatrième exemple joue la musique de "Frère Jacques" à deux voies.

```
pour frere_jacques_1
####################
# Met en mémoire la partition
sequence [0.5 do re mi do do re mi do]
sequence [0.5 mi fa 1.0 sol 0.5 mi fa 1.0 sol]
sequence [0.25 sol la sol fa 0.5 mi do 0.25 sol la sol fa 0.5 mi do]
sequence [0.5 re :- sol :+ 1.0 do 0.5 re :- sol :+ 1.0 do]
fin

pour frere_jacques_2
####################
# Met en mémoire la partition
sequence [0.5 sol la si sol  sol la si sol]  # 8/2 = 4 temps
sequence [0.5 si :+ do 1.0 re :- 0.5 si :+ do 1.0 re :-]  # 2/2 + 1
+ 2/2 + 1 = 4 temps
sequence [:+ 0.25 re mi re do :- 0.5 si sol :+ 0.25 re mi re do :-
0.5 si sol]  # 4/4 + 2/2 + 4/4 + 2/2 = 4 temps
sequence [0.5 la re 1.0 sol 0.5 la re 1.0 sol]
fin

pour demarre
############
# joue la mélodie sur deux voie en même temps.
efseq             # On efface la séquence actuellement en mémoire
Frere_jacques_1   # On charge la musique
findseq 0         # On revient au début
Frere_jacques_2   # On recharge la deuxième voie.
joue              # joue les deux voies en mémoire
fin
```

Ce programme se trouve dans le fichier : "ex_14_04_Frere_Jacques.lgo"
Tapez "demarre" pour entendre la musique.

Ce dernier exemple joue la musique de "J'ai du bon tabac".

```
pour tabac
##########
# Met en mémoire la partition
sequence [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :- si si 0.5
sol la si sol
          1 la 0.5 la si 1 :+ do re 2 :- sol ]
sequence [:+ 1 re 0.5 re do 1 :- si 0.5 la si 1 :+ do re 2 :- la ]
sequence [:+ 1 re 0.5 re do 1 :- si 0.5 la si 1 :+ do re 2 :- la ]
sequence [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :- si si 0.5
sol la si sol
          1 la 0.5 la si 1 :+ do re 2 :- sol ]
fin

pour demarre
############
# Joue la mélodie : "J'ai du bon tabac"
efseq       # On efface la séquence actuellement en mémoire
tabac       # On recharge la musique précédente
# findseq 2   # On replace le curseur au niveau du premier "la" noir
de la 2eme mesure
# tabac       # On recharge la même séquence mais décalée de deux
temps.
joue        # Un magnifique canon !
fin
```

Ce programme se trouve dans le fichier : "ex_14_05_Bon_Tabac.lgo"
Tapez "demarre" pour entendre la musique.

Pour simuler divers instruments de musique.
Dans "Option", "Préférences", sous l'onglet "Son", il est possible de choisir un instrument de musique, si votre installation java est installer avec les instruments de musique.
Voici comment faire pour installer les instruments de musique :
1) Télécharger le fichier compressé soundbank-mid.gm.zip de :
   http://java.sun.com/products/java-media/sound/soundbank-mid.gm.zip
   D'autres possibilités et explication en anglais se trouvent sur :
   http://java.sun.com/products/java-media/sound/soundbanks.html
   Des explications en français se trouvent sur : http://java.sun.com/j2se/1.3/l10n/fr/README
2) Décompressez le fichier que vous venez de télécharger et copiez son contenu dans
   c:\Program Files\Java\jre1.5.0_06\lib\audio
   "jre1.5.0_06" peut être un autre nom de répertoire, suivant la version de java.
3) Renommez le fichier en : soundbank.gm

Si vous trouvez cela trop compliqué, pour vous simplifier la tache, j'ai mis à disposition le fichier "soundbank.gm" à l'adresse : http://www.perso.ch/bernard.gisin/xlogo/soundbank.gm

Une méthode alternative d'installation que je pense être moins bonne est la suivante :
Copiez le fichier soundbank.gm dans le dossier où se trouve "xlogo.jar" et les instruments sont disponibles.

Résumé : Ce chapitre montre comment
1) introduire une séquence de musique pour la jouer. sequence
2) efface une séquence de musique. efsequ
3) joue une séquence de musique. joue
4) recherche une séquence de musique. findsequ
5) attend quelques soixantièmes de secondes. attends

# 15. Les animations graphiques

Voici une animation graphique que j'ai prise du chapitre 7.1 du tutoriel de XLogo.
Il indique clairement qu'il y a un effet de clignotement, qui sera amélioré dans le programme suivant.

```
pour rec :lo :la
################
si :lo=0 |:la=0[stop]
repete 2[av :lo td 90 av :la td 90]
rec :lo-1 :la-1
fin

pour chiffre :a :b :c :d :e :f :g
#################################
# On dessine le rectangle 1
si :a=1 [rec 160 40]
# On dessine le rectangle 2
si :b=1 [rec 40 160]
lc td 90 av 120 tg 90 bc
# On dessine le rectangle 3
si :c=1 [rec 160 40]
lc av 120 bc
# On dessine le rectangle 5
si :e=1 [rec 160 40]
# On dessine le rectangle 4
tg 90 lc re 40 bc
si :d=1 [rec 160 40]
# On dessine le rectangle 6
td 90 lc av 120 tg 90 bc
si :f=1 [rec 160 40]
# On dessine le rectangle 7
lc av 120 tg 90 re 40 bc
si :g=1 [rec 160 40]
fin

pour demarre
############
ve ct chiffre 0 1 1 1 1 1 1 attends 60
ve ct chiffre 1 1 1 1 1 1 1 attends 60
ve ct chiffre 0 0 1 0 1 1 0 attends 60
ve ct chiffre 1 1 1 1 0 1 1 attends 60
ve ct chiffre 0 1 1 1 0 1 1 attends 60
ve ct chiffre 0 0 1 1 1 0 1 attends 60
ve ct chiffre 0 1 1 1 1 1 0 attends 60
ve ct chiffre 1 1 0 1 1 1 0 attends 60
ve ct chiffre 0 0 1 0 1 0 0 attends 60
ve ct chiffre 1 1 1 0 1 1 1 attends 60
fin
```

Ce programme se trouve dans le fichier : "ex_15_01_animation.lgo"

Voici comment éviter le clignotement. (Repris du chapitre 7 du tutoriel de XLogo)
Les procédures "chiffre" et "rec" sont les mêmes que précdement.

```
pour demarre
############
# On passe en mode animation
animation vrai
ve ct chiffre 0 1 1 1 1 1 1 rafraichis attends 60
ve ct chiffre 1 1 1 1 1 1 1 rafraichis attends 60
ve ct chiffre 0 0 1 0 1 1 0 rafraichis attends 60
ve ct chiffre 1 1 1 1 0 1 1 rafraichis attends 60
ve ct chiffre 0 1 1 1 0 1 1 rafraichis attends 60
ve ct chiffre 0 0 1 1 1 0 1 rafraichis attends 60
ve ct chiffre 0 1 1 1 1 1 0 rafraichis attends 60
ve ct chiffre 1 1 0 1 1 1 0 rafraichis attends 60
ve ct chiffre 0 0 1 0 1 0 0 rafraichis attends 60
ve ct chiffre 1 1 1 0 1 1 1 rafraichis attends 60
# On rebascule en mode dessin classique
animation faux
fin
```

Ce programme se trouve dans le fichier : "ex_15_02_animation.lgo"

Voici un dernier exemple d'animation, d'un disque qui bouge dans l'écran.

```
pour disque :rayon
##################
# Dessine un disque noir de rayon  :rayon
cercle :rayon
remplis
fin

pour demarre
############
# animation d'une boule qui se deplace dans la fenetre
ecris [pressez sur la touche  ESC  pour terminer]
donne "touche 28
donne "posX 0
donne "dirX 1  # direction de deplacement horizontal du disque
donne "posY 0
donne "dirY 1  # direction de deplacement vertical du disque
#
# trace le cadre
levecrayon
fixexy -330 (-226)
baissecrayon
fixecouleurcrayon 0  # noir
fixexy -330 ( 226)
fixexy  330 ( 226)
fixexy  330 (-226)
fixexy -330 (-226)
levecrayon
fixexy :posX :posY
baissecrayon
#
animation vrai
#inversecrayon  # pas utile
fixecouleurcrayon 0  # noir
```

```
disque 20
#
tantque [non (:touche = 27)] [
  si touche? [donne "touche liscar]
  fixecouleurcrayon 7  # blanc
  disque 20  # efface le disque
  #
  # va a la nouvelle position
  donne "posX :posX + :dirX
  donne "posY :posY + :dirY
  levecrayon
  fixexy :posX :posY
  baissecrayon
  fixecouleurcrayon 0  # noir
  disque 20 rafraichis # dessine le disque
  #
  # Test s'il faut changer de direction
  si (:posX >  307) [donne "dirX (-1)]
  si (:posX < -307) [donne "dirX  1]
  si (:posY >  203) [donne "dirY (-1)]
  si (:posY < -203) [donne "dirY  1]
  ]
animation faux
fin
```

Ce programme se trouve dans le fichier : "ex_15_03_animation_disque.lgo"

Résumé : Ce chapitre montre comment
C.F. chapitre 4.1 du manuel de XLogo.
1) exécuter une animation, avec les instructions **animation** et **rafraichis**.

# 16. Les gestions de fichiers

On peut lire le contenu de fichier, ainsi que écrire dans des fichiers.
On peut aussi charger et afficher des images. En combinant avec l'animation précédente, cela donne le petit programme suivant.

```
pour demarre
############
# animation d'une image qui se deplace dans la fenetre
ecris [pressez sur la touche  ESC  pour terminer]
donne "touche 28
donne "posX 0
donne "dirX 1  # direction de deplacement horizontal du disque
donne "posY 0
donne "dirY 1  # direction de deplacement vertical du disque
#
animation vrai
#
tantque [non (:touche = 27)] [
  si touche? [donne "touche liscar]
  #
  # va a la nouvelle position
  donne "posX :posX + :dirX
  donne "posY :posY + :dirY
  ve ct
  levecrayon
  fixexy :posX :posY
  chargeimage "pinguin.png
  # pinguin.png doit se trouver dans le meme repertoire que le
programme
  rafraichis # dessine l'image
  #
  # Test s'il faut changer de direction
  si (:posX >  307) [donne "dirX (-1)]
  si (:posX < -307) [donne "dirX  1]
  si (:posY >  203) [donne "dirY (-1)]
  si (:posY < -203) [donne "dirY  1]
  ]
animation faux
fin
```

Ce programme se trouve dans le fichier : "ex_16_01_animation_image.lgo"

Le fichier "pinguin.png" contenant une image est chargée à différents endroits de la fenêtre, et se déplace ainsi.

Cette partie ne marche pas encore !

```
#
# On ouvre un flux vers le fichier desire. Ce flux sera repere par
le numero 2
fixerepertoire "c:\\
ouvreflux 2 "exemple
# On ecrit les lignes desirees
ecrisligneflux 2 [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
ecrisligneflux 2 [abcdefghijklmnopqrstuvwxyz]
ecrisligneflux 2 [0123456789]
# On ferme le flux pour achever l'ecriture
fermeflux 2
A present, on peut constater que l'ecriture s'est bien passee :
# On ouvre un flux vers le fichier a lire. Ce flux sera repere par
le numero 0
ouvreflux 0 "c:\\exemple
# On lit les lignes du fichiers successivement
ec lisligneflux 0
ec lisligneflux 0
ec lisligneflux 0
# On ferme le flux
fermeflux 0
Si on souhaite
fixereperto
ouvreflux 1
ajouteligne
fermeflux 1
fin
```

Ce programme se trouve dans le fichier : "ex_16_02_lecture_fichier.lgo"

Résumé : Ce chapitre montre comment

C.F. chapitre 4.7 du manuel de XLogo.

1) **chargeimage** et **ouvreflux** et **ecrisligneflux lisligneflux ajouteligne fermeflux**.

# 17. Dernières remarques

De nombreuses instructions n'ont pas été traitées dans cette approche de XLogo par l'exemple.
Il est vivement conseillé de lire également le tutoriel se trouvant à l'adresse :
http://xlogo.free.fr/fichiers/tutoriel.pdf

LA référence qu'il est indispensable d'avoir à disposition se trouve à l'adresse :
http://xlogo.free.fr/fichiers/manuel-xlogo.pdf

# Index

A Faire

Pour vraiment bien faire, il faudrait créer un index. Chose que je n'ai pas faite.