Présentation générale du logiciel 'jfMindMap' (jfMM en abrégé)

1) Avant tout signalons une exception dans les conventions de notations

Tout au long de ces explications notre programme est nommé 'jfMindMap', que nous abrégeons en jfMM, c'est à dire qu'il commence par 2 minuscules 'jf'.

Cependant, veuillez noter une exception : sa classe principale, au lieu de se nommer 'jfMindMap' est appelée : JFMindMap. Cette irrégularité est imposée par la syntaxe du langage Java qui préconise que les classes commencent par une majuscule. Elle apparaît seulement au lancement de ce logiciel.

2) Pour quels usages jfMindMap est-il conçu ?

a) Il est à la fois un traitement de texte et un logiciel de présentation

Sorte de traitement de texte il permet de :

- Lire, afficher chacun des fichiers '.lsp' constituant la substance de chacun de mes cours à la Philotechnique.
- Présenter les documents sous la forme indentée des programmes structurés : Lisp, C, Python, Java...

Plus précisément, à la manière de la présentation d'un document dans un traitement de texte, il permet d'afficher le plan d'un exposé sous la forme d'un arbre, sous la forme de N paragraphes qui se décomposent en K sous paragraphes. Ainsi le texte de l'exposé à présenter est distribué entre les nœuds de cet l'arbre et ses feuilles terminales.

jfMindMap est un logiciel de présentation, très léger et très simple

En tant que logiciel de présentation, il afficher des illustrations (photos, dessins, schémas, images) :

Pour cela, en fin de n'importe quelle ligne de l'éditeur jfMM, on écrit un lien vers une illustration. Ce lien est très visible : sur la droite des explications, il est introduit par le caractères '§' qui lui-même est entouré d'un cadre vert fluo.

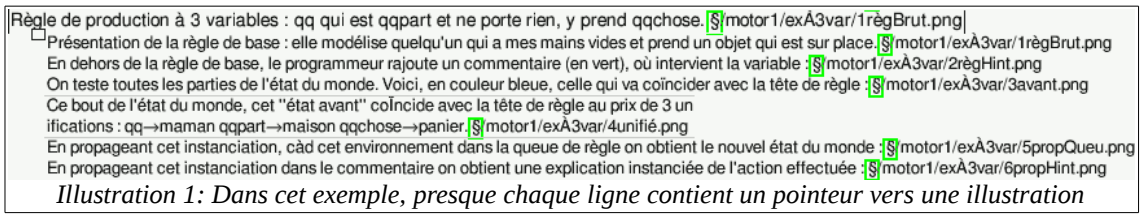


Illustration 1: Dans cet exemple, presque chaque ligne contient un pointeur vers une illustration

Quand on clique ce lien, jfMM escamote le texte arborescent et affiche l'illustration pointée par l'adresse correspondante dans le dossier arborescent nommé '§'. Voici un exemple de lien : §/jfMindMap/pŋVuGlo_56-1.png.

b) jfMM est destiné à 2 types d'utilisateurs et chacun d'eux en fait un usage spécifique

1) De retour chez lui, un auditeur de la Philotechnique peut visualiser son cours au calme. Il consulte le plan, lit les explications et clique sur les liens qui affichent les illustrations correspondantes.

2) Pour un conférencier, il permet de planifier une présentation, rédiger un texte en rapport et y lier les illustrations de son propos.

Note : ce fonctionnement en mode 'présentation' n'est pas prévu dans la prestation gratuite que je fournis, pour cette raison, cette utilisation de jfMM pour la présentation est livrée SANS support technique, SANS documentation et SANS garantie du codeur.

3) : Le problème de l'encodage des fichiers qui diffère entre Linux &Windows

A) Expliquer à quoi correspond l'encodage des fichiers (file encoding)

Au plus profond de l'ordinateur, les données traitées sont représentées en binaire (0 et 1). Puis sur la couche suivante, vient le code ASCII qui permet de représenter ces données en binaire (chiffres, lettres, textes, nombres).

Initialement l'ASCII encodait les caractères américain sur 8 bits (chiffres, lettres, ponctuation, tabulation...). Ensuite il a fallu l'étendre pour coder les caractères correspondants aux caractères spécifiques de chaque nation. Ça a donné différents encodages de fichiers sur 16 bits.

Évidemment Windows utilise l'encodage ANSI (cp1252) qui est différent de celui de l'UNICODE de Linux (UTF-8). De plus, la JVM (Java Virtual Machine) tourne exclusivement avec l'unicode UTF-8. Alors, nativement, les *.lsp que j'utilise pour les cours sont en UTF-8.

B) Là où le problème apparaît :

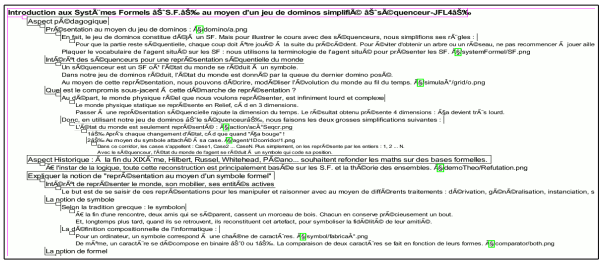
a) Position du problème :

Quand je fournis un support de cours en PDF, vous tous pouvez le lire car c'est la vocation d'un pdf d'être un outil de communication inter-plateformes.

Une précision avant de commencer : initialement, mes dossiers *.lsp fournis dans phiDe0aC, sont encodés en UTF-8. Alors, quand un utilisateur Linux travaille avec ces *.lsp, il peut les ouvrir et sauver ses modifications car Linux aussi fonctionne avec l'UTF-8.

b) Il en va de même pour un utilisateur Windows qui utilise jfMM pour éditer MES *.lsp

Ensuite, sachez que pour Windows, j'ai programmé jfMM afin qu'il puisse ouvrir mes fichiers *.lsp, les éditer et les sauvegarder (tout ça en UTF-8). Donc, sous Windows, le jfMM fonctionne avec mes fichiers *.lsp.



c) Mais il reste deux problèmes sous Windows

La console de Windows est initialement encodée en ANSI c'est-à-dire en 'cp1252'

Mais jfMM lui fournit de l'UTF-8 et à l'affichage apparaissent des signes cabalistiques : si on ne corrige pas ce problème initial, à l'écran on obtient un affichage tel que ceci :

La commande 'java' pour ordonner à jfMM de transcoder l'affichage à l'écran est "-Dfile.encoding=UTF-8". Elle doit être écrite au lancement de jfMM sous Java.

Au cas où un utilisateur veut créer SES PROPRES fichiers *.lsp

Rappel : ce type de fonctionnement n'est pas prévu dans la prestation gratuite fournie, donc n'est ni soutenu ni documenté.

Mais revenons à cet usager : pour créer ses propres fichiers il doit utiliser un éditeur capable de créer des fichiers en encodage UTF-8. Ce n'est pas un drame car ce genre d'éditeur de texte existe. Mais il reste que quelqu'un qui ne serait pas averti du fonctionnement de mes *.lsp en mode UTF-8 serait déconcerté par les problèmes qui apparaîtraient le jour où il cesserait de les utiliser pour voler de ses propres ailes. Surprise !

4) Installation du logiciel pour Windows

Note : l'installation de Java sous Linux n'est pas expliquée ici car un praticien de ce système saura toujours se débrouiller seul.

A) Avant toutes choses, il vous faut disposer de Java :

Note : sur le site, vous trouverez des explications plus détaillées à ce sujet dans le pdf : 'généSurJava-installYUsing_XX'.

Note : Java est le pendant de C# (C sharp) de Microsoft, mais Java est gratuit pour les particuliers, C# est payant.

jfMM est compilé avec Java1.5. Grâce à la compatibilité ascendante, vous devez installer au moins java1.6, alors, peu importe la version : ça peut être : java1.6, 1.7, 1.8...

Comment vérifier si vous disposez de Java ?

En mode console lancez Java, c'est à dire tapez la commande 'Java' suivie de Entrée :

Si c'est un succès si vous obtenez un long message qui commence comme ceci :

Si c'est un échec si vous recevez un message du genre : "java : commande introuvable". Alors pour installer java, reportez-vous à la doc officielle de Sun ou à mon pdf 'généSurJava-installYUsing_XX'..

```
jf@jf-PX615AA-ABF-w5067-fr:~$ java
Usage: java [-options] class [args...]
           (to execute a class)
   or  java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available
    -client       to select the "client" VM
    -server       to select the "server" VM
    -zero         to select the "zero" VM
    -jamvm        to select the "jamvm" VM
    -avian        to select the "avian" VM
    -dcevm        to select the "dcevm" VM
    -hotspot      is a synonym for the "client" VM  [deprecated]
                  The default VM is client.

    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
                  A : separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
```

Ici java est bien installé, taper 'java' affiche ce texte

Vous n'êtes pas obligés d'installer le gros JDK (Java Developpement Kit), le petit JRE (runtime) suffit à jfMM.

B) Afin de pouvoir installer jfMM,

Vous devez connaître le mot de passe 'administrateur' car nous copions des fichiers exécutables sur le disque dur.

C) jfMM est disponible sur mon site, et fonctionne pour Windows ET pour Linux

a) Comment accéder à ces pages ?

Solution 1)

- Sur la page d'accueil de "www.flucas.com" ou "http://jflucas.free.fr".
- Aux 2/5 de l'unique tableau de cette page.
- En regard de : "Télécharger fichiers ou dossiers ... " cliquez sur le lien : "... Pour jfMindMap Philotec 19-20".

Solution 2)

Ou encore directement à ce lien, la page de téléchargement de jfMM :
http://jflucas.free.fr/DownLoadLesFichiersOuDossiersPourJfMM.htm

Nouveau : les fichiers et dossiers pour jfMindMap mixte linux-windows (avril 2020)

Les 41 *.class de code Java exécutable pour jfMindMap §, illustration des cours 0 à c, mixte Linux Windows Pour Java, càd pour tous, en UTF-8 (illisibles en ANSI)	pour Linux il va à : /home/jf, pour Windows à : c:/home/jf 145 items pour §de0aC, le dossier mixte Win/lux des illustrations 13 cours *.lsp zippés dans le dossier 0'phi'De0aC
Pdf mixte (Windows-Linux) présentant 'Java' (version 1) Pdf mixte (Windows-Linux) présentant 'jfMM' (version 8)	pdf qui présente Java version 1 pdf qui présente jfMM version 8

b) Vous n'en avez que 5 fichiers à charger :

Les 2 fichiers *.pdf qui constituent des explications pour vous aider, càd :
- presenta°DeJfMM_XY.pdf : À lire en premier.

Et les 3 dossiers 'mixtes, universels' càd communs à Windows et Linux :
- phiDe0aC.zip : les 21 fichiers *.lsp des 13 cours (de 0 à 12) que j'ai déjà préparés.
- dossierDesClass.zip : qui contient le code exécutable de jfMM.
- paragDe0aC.zip : qui contient '§' l'arbre des illustrations du cours (fichiers png, bnp et jpg).

D) Vous devez seulement installer ces 3 dossiers : a, b et c

a) Le dossier '§' qui contient toutes les images illustrant les exposés (png, bmp, jpg) :

Votre intervenant (conférencier) vous a fourni ces images classées dans une arborescence, conservez ce classement car l'affichage des liens l'utilise et sera irrémédiablement perturbé si vous l'abandonnez.

b) Le dossier 'φde0àC' qui contient tous les supports de cours pour l'année scolaire 19-20 :

Pour chacune des leçons de l'année scolaire 19-20, votre intervenant vous fournit les *.lsp constituant le texte de chacune de ses interventions.

c) Le dossier 'dossierDesClass' qui constitue le code du programme jfMindMap :

Il contient 'JFMindMap.class', le fichier lanceur de jfMM. C'est un exécutable 'Java' qui constitue le point d'entrée de jfMindMap (celui qu'en tant qu'opérateur vous devez lancer pour initialiser le processus).

Il contient les autres fichiers exécutables : 40 autres fichiers avec l'extension 'class'.

Note : le programme jfMM est mixte c'est-à-dire qu'il fonctionne avec/pour les systèmes d'exploitation Window et Linux à condition que vous stockiez '§' arbre des illustrations dans le **même lieu imposé : '/home/jf'.**

E) Pour installer ce logiciel il faut copier ces 3 dossiers à leur place respective

a) Le dossier '§'

Sur le site il se nomme 'paragDe0aC.zip'

C'est à dire 'le caractère paragraphe '§' pour les leçons allant de i0 à iC (i0, i1… i9, iA, iB et iC). Il doit être impérativement copié dans le dossier dédié '/home/jf'.

Encore quelques précisions à propos de ces dossiers

Donc, si le dossier dédié correspondant à votre système d'exploitation Windows ou Linux, n'existe pas sur votre disque dur, c'est à vous de le créer pour pouvoir y copier le dossier '§'.

Attention, sur votre disque dur, après décompression de ce dossier zippé, le nom obtenu n'est pas '§' mais quelque chose comme '§de0àC', vous devez donc le renommer impérativement '§' car c'est à cette **SEULE** adresse que les liens cliqués dans jfMM vont chercher l'illustration correspondante pour l'afficher à l'écran.

b) Le dossier mixte 'φde0aC'

Sur le site il se nomme 'phiDe0aC.zip'

C'est à dire 'le caractère 'phi' ou 'φ' de l'alphabet grec pour les leçons allant de i0 à iC (i0, i1… i9, iA, iB et iC).

Même si c'est un dossier 'système' il est mixte, c'est-à-dire commun à Windows et Linux

Cette bizarrerie tient au fait qu'il contient les fichiers *.lsp correspondant au cours, lesquels sont exploités par la JVM (machine virtuelle Java) pour générer les pages de texte structuré. En effet, elle fonctionne exclusivement en Unicode, donc, pour les 2 systèmes d'exploitation, ils sont encodés uniquement en UTF-8.

Ce dossier 'φde0aC' peut être mis où bon vous semble (vous pouvez même le renommer)

Quand, pour étudier une leçon du cours, au moyen du bouton 'LoadOpen' de l'interface graphique de jfMM vous ouvrez le dossier *.lsp correspondant, c'est vous qui naviguer dans l'arborescence du disque dur, depuis la racine jusqu'à votre cible.

C'est pourquoi ce dossier 'φde0aC' peut être copié où bon vous semble (vous pouvez même le renommer) car c'est à vous d'adapter votre navigation au-travers du disque dur jusqu'à sa position.

Mais je pense que le mieux serait de le copier dans le dossier contenant '§' afin de minimiser votre navigation dans l'arborescence du disque dur !

c) Le dossier 'dossierDesClass' contient tous les exécutables Java de jfMM, donc :

Soit vous le copiez à la même hauteur/profondeur que Java.exe (c'est-à-dire au-dessus ou au-dessous)

Ces explications sont abstraites, alors voici d'autres façons de dire la même chose : Vous le copiez...
- Dans le dossier qui lui-même contient le dossier 'Java.exe'.
- C'est à dire au même niveau que le JRE (ou bien le JDK) de Java.

C'est à dire que le plus simple pour installer jfMM est de copier le dossier 'dossierDesClass' dans le dossier où se trouve 'java.exe' (le code de la JVM).

Mais où trouver 'java.exe', le code de la JVM ? Sous Linux, en ligne de commande, c'est à dire dans une console, on tape : "which java". Et Linux répond : "/usr/bin/java". Sous Windows l'information doit être visible dans le panneau de configuration (la liste des programmes déjà installés).

Expliquer à quoi correspond cette astuce d'installation :

En effet, pour lancer JFMindMap, dans une console, on commande au système : java dossierDesClass.JFMindMap

> Petite précision sur ce lancement de jfMM : en Java les classes commencent par une majuscule : c'est pourquoi le point d'entrée de jfMM est JFMindMap et pas jfMindMap. Il en découle que la syntaxe de lancement de jfMM est bien "java dossierDesClass.JFMindMap" et pas "java dossierDesClass.jfMindMap".

Alors le système cherche sur le disque dur où se trouve le fichier 'java.exe' et le lance en lui passant un paramètre : dossierDesClass.JFMindMap. Par convention ce paramètre indique à Java le point d'entrée du programme (donc ici de 'jfMindMap'). La JVM qui reçoit ce paramètre en déduit donc le point de départ de jfMM, le programme Java à exécuter pour lire le cours et voir les illustrations. Pragmatique, elle commence donc à chercher 'JFMindMap' dans le lieu le plus simple pour elle : sous ses pieds, c'est à dire dans le dossier où elle-même se trouve. CQFD.

Soit vous copiez 'dossierDesClass' là où vous le voulez…

Mais c'est à vous d'assumer, il vous reste à expliquer à votre système (Windows/Linux) où il le trouvera afin qu'il puisse l'exécuter. Débrouillez-vous !

Sur Linux et les anciens systèmes Windows

Vous devez bricoler le 'path', le 'chemin' de votre système. J'espère que vous voyez ce que je veux dire.

> À titre indicatif, sans garantie que ça marche sur votre système, voici un exemple (mais il vous faut le transposer) : path=%path;C:/program files/java/jdk1.6.0./bin;
>
> Explication de la ligne au dessus : dans l'ancien path du système, on rajoute le chemin où est installé Java (c'est à dire : C:/program files/java/jdk1.6.0./bin).

Sur les systèmes Windows plus récents il faut bricoler les variables d'environnement :

Je ne suis pas habitué à Windows que j'ai abandonné depuis longtemps, voici, sans garantie, quelques explications prise sous XP, mais Vista est à peu près pareil.

Successivement cliquer : Démarrer / Panneau de configuration / Système / avancé /variables d'environnement.

Ne pas toucher aux variables système mais dans les variables utilisateur ajouter une autre variable 'Path' et, par exemple pour 'jre7 sur XP' lui donner la valeur :
C : Program Files\Java\Jre7\bin ;

F) Travailler avec jfMM

a) Pour Linux, en mode console, c'est-à-dire en ligne de commande, il se lance :

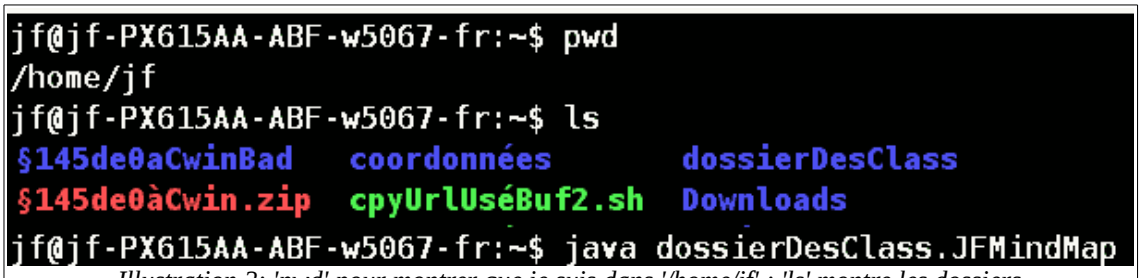


Illustration 2: 'pwd' pour montrer que je suis dans '/home/jf' ; 'ls' montre les dossiers

Pour cela, dans le dossier où vous avez copié le dossier 'dossierDesClass' (on le voit en bleu, en haut à droite), après le caractère d'invite (ici, le dollar), il faut taper la commande :

java dossierDesClass.JFMindMap

Puis, en faisant 'entrée' on obtient un écran d'accueil qui ressemble à ceci :

b) Voici quelques explications qui vous aideront à mémoriser cette commande :

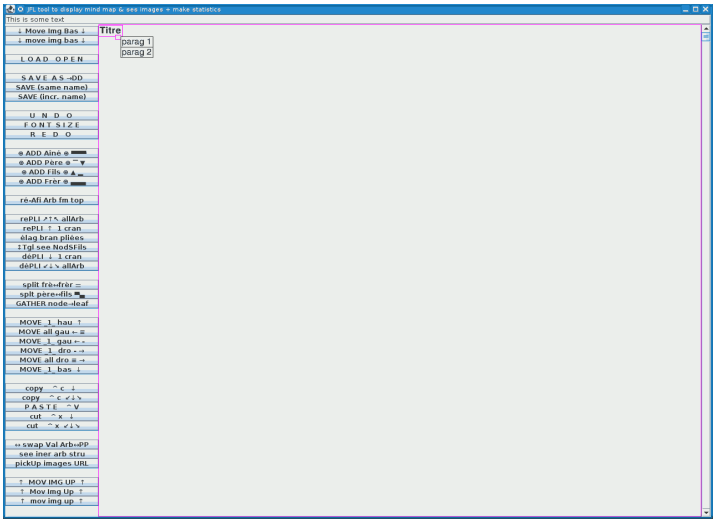


Illustration 3: page principale de jfMM

'java'

Pour lancer la JVM (Java Virtual Machine) : la machine virtuelle Java qui exécute le code compilé afin d'effectuer le travail.

"dossierDesClass.JFMindMap"

C'est le paramètre qu'on passe à Java, à la JVM. Il se décompose en deux parties :

'JFMindMap'

Car c'est le nom du programme principal du logiciel 'JFMindMap', qu'on demande à la JVM d'exécuter. J'ai obtenu le code compilé 'JFMindMap.class' en compilant le programme principal nommé "JFMindMap.java".

'dossierDesClass'

Car c'est le nom du dossier contenant JFMainPr.class et les autres *.class, c'est à dire les 40 autres programmes exécutables qui constituent le programme JFMindMap tournant sur la JVM (machine virtuelle Java).

Le point '.' qui sépare ces deux paramètres

En syntaxe Java il signifie que le fichier 'JFMindMap' est dans le dossier 'dossierDesClass'. C'est un peu l'équivalent du slash en syntaxe système.

c) Pour Window, en mode console, c'est-à-dire en ligne de commande, comment le lancer :

À la commande classique utilisée pour Linux : java dossierDesClass.JFMindMap il faut ajouter "-Dfile.encoding=UTF-8" l'instruction dédiée à Java pour forcer la console en mode 'UTF-8'. Au total la commande devient :

java "-Dfile.encoding=UTF-8" dossierDesClass.JFMindMap

à laquelle il faut ajouter 'entrée'.

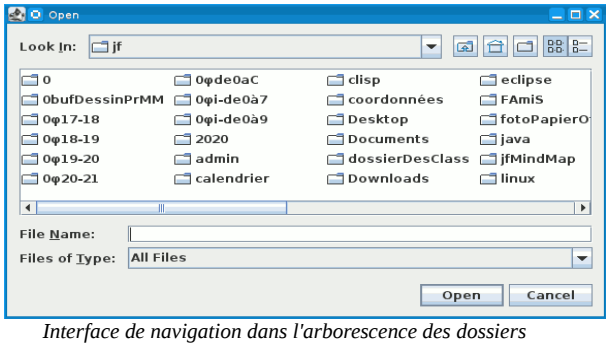
C'est un peu compliqué à mémoriser par cœur mais on peut utiliser le copier-coller !

d) Étudions le rôle de quelques boutons parmi ceux qui apparaissent à gauche de l'écran

Utiliser jfMM pour afficher à l'écran le texte arborescent correspondant à un fichier *.lsp

Quand on a lancé Java et qu'on obtient la page principale de jfMM (illustration 3) il nous faut maintenant aller chercher un fichier *.lsp, c'est-à-dire un des cours de 'φde0aC' / 'phiDe0aC' donc c'est à vous de vous souvenir où vous l'avez copié sur le disque dur pour aller le chercher.

Dans la colonne de gauche de la page principale, cliquer le bouton 'LOAD OPEN' affiche une interface graphique qui permet de naviguer sur le disque dur jusqu'à un fichier 'lsp' afin de l'ouvrir à l'écran.



Interface de navigation dans l'arborescence des dossiers

Les conventions définissant les gestes de navigation sont classiques et je vous laisse les découvrir vous-même.

Après une navigation au travers des dossiers, vous arrivez en bout de branche sur 1 ou plusieurs fichiers : sélectionnez-en un pour que son nom apparaisse dans 'File Name'. Un clic sur le bouton 'Open' l'ouvre et l'affiche à l'écran : on obtient la structure d'un texte indenté, Voir en page 1 l'illustration 1.

Dans cet exemple, presque chaque ligne contient un pointeur vers une illustration. Pour afficher à l'écran le dessin pointé par un lien, il suffit de cliquer sur lui et jfMM escamote le texte pour le remplacer par l'illustration sélectionnée. Ensuite pour quitter l'illustration et retourner au texte, il suffit de cliquer n'importe où sur l'écran.

Plus précisément, la règle d'affichage en fonction du lieu d'un clic sur une ligne est ainsi :

- Un clic sur sa moitié gauche la sélectionne : elle est entourée en rouge et son texte apparaît dans le cartouche d'édition en haut de la page.
- Un clic sur sa moitié droite, si jfMM y trouve un lien, produit l'affichage de l'image correspondante.

Note : cette remarque n'est pas superflue car si une ligne est très courte et contient un nœud très long cliquer sur sa tête, par exemple sur le '§' peut ne pas être suffisant, il faut bien cliquer sur la partie droite de la ligne (au-delà de son milieu) !

Les boutons pour monter ou descendre le texte arborescent à l'écran

```
"↓ Move Img Bas ↓" bouge l'image vers le bas à grands pas
"↓ move img bas ↓" bouge l'image vers le bas à moyens pas
"↑  MOV IMG UP  ↑" bouge l'image vers le haut à grands pas
"↑  Mov Img Up  ↑" bouge l'image vers le haut à moyens pas
"↑  mov img up  ↑" bouge l'image vers le haut à petits pas
```

Note :

Le tamponnage de l'image de l'arbre indenté

Sur jfMM dessiner la structure d'un texte indenté est lent. Donc pour accélérer cet affichage l'ordinateur dessine la structure dans une mémoire tampon qu'il utilise tant que l'arbre demeure inchangé.

Le tamponnage progressif de l'image

Alors, quand un texte est long, pour accélérer cette mémorisation, on génère seulement ses premières lignes et les lignes suivantes sont tronquées. Dans ce cas, une ligne horizontale rouge apparaît au niveau de la dernière ligne afin d'indiquer cette troncature.

Ré-affichage de l'arbre à partir du nœud initial

Si par la suite l'utilisateur fait monter l'image vers le haut, vient un moment où jfMM génère la partie suivante du texte. De cette façon, à mesure qu'on affiche l'arbre à l'écran, l'image est progressivement générée.

Mais pour générer l'arbre dans sa totalité, on peut cliquer sur le bouton : "ré-Afi Arb fm top", mais, puisque ça génère la totalité de l'arbre d'un seul coup, c'est plus long : il faut attendre plus longtemps.

Je sais bien que l'affichage de l'arbre est bogué...

Quand on fait défiler l'arbre, ou à d'autres occasions encore, les morceaux se raccordent mal ou se chevauchent. Alors vous pouvez demander à jfMM de ré-afficher l'arbre à partir du nœud initial ; ça corrige momentanément l'affichage, mais à terme le problème réapparaît !

Mais je dois d'abord corriger un autre problème plus importants : la lenteur de l'affichage. Et pour cela je vais tout reprendre l'algorithme ce qui va me prendre des mois. Donc, il faut faire avec pendant quelques semestres.

Au lieu d'afficher l'arbre entier, montrer à l'écran seulement les têtes de chapitres

Note : il faut d'abord cliquer sur un nœud pour le sélectionner. Quand c'est le cas :

Pour obtenir une approche plus synthétique d'un exposé, on peut souhaiter seulement voir les têtes de chapitres.

```
"rePLI ↗↑↖ allArb"   : replie toute la branche, c'est-à-dire toutes les branches du nœud sélectionné.
"rePLI  ↑  1 cran"   : replie d'un cran les branches du nœud sélectionné (càd replie la dernière branche, la + fine)
"dépLI  ↓  1 cran"   : déplie d'un cran les branches du nœud sélectionné (càd déplie la dernière branche, la + fine)
"dépLI ↙↓↘ allArb"   : expanse toute la branche, c'est-à-dire toutes les branches du nœud sélectionné.
```

Note : Si on clique sur le petit rectangle juste au-dessous d'un nœud, ça inverse (toggle) son affichage : s'il était déplié ça le replie et réciproquement.

```
"↕Tgl see NodSFils"   : inverse (toggle) l'affichage du nœud sélectionné : s'il était plié ça le déplie et réciproquement.
```

```
"élag bran pliées"    : Attention Danger : réellement élague les branches pliées, c'est-à-dire les coupe définitivement. jfMM demande confirmation, mais ça reste dangereux.
```

Extrait du code source : le rôle de chaque bouton de jfMM

```
button20 = new JButton ("↓ Move Img Bas ↓");  button20.addActionListener(this);this.add(button20); // scroll down the image à grands pas
button17 = new JButton ("↓ move img bas ↓");  button17.addActionListener(this);this.add(button17); // scroll down the image à moyens pas

button28 = new JButton ("L O A D   O P E N");  button28.addActionListener(this);this.add(button28); // Ouvrir un file *.lsp et charger en mémoire

button13 = new JButton ("S A V E  A S →DD"); button13.addActionListener(this);this.add(button13); // Surf in le DD pr y sauv l'arb édité à 1 adresse existante
button32 = new JButton ("SAVE (same name)"); button32.addActionListener(this);this.add(button32); // sauve l'arbre que l'on a édité, sous le même nom
button33 = new JButton ("SAVE (incr. name)");  button33.addActionListener(this);this.add(button33); // sau l'arbr & incr son nom. si syntax= *_02.lsp → *_03.lsp

button34 = new JButton (" U   N   D   O ");       button34.addActionListener(this);this.add(button34); // ⮐ pour défaire la précédente instruct°, Mind : marche MAL
button39 = new JButton ("F O N T  S I Z E");  button39.addActionListener(this);this.add(button39); // set initFontSize vu val du Neu in PP
button35 = new JButton (" R   E   D   O ");       button35.addActionListener(this);this.add(button35); // ↳ pour refaire la précédente undone ins, Mind : marche MAL

button03 = new JButton ("⊕ ADD Aîné ⊕ ▀▀▀");button03.addActionListener(this);this.add(button03); // ajouter un node frère au-dessus
button04 = new JButton ("⊕ ADD Père ⊕ ▼");button04.addActionListener(this);this.add(button04); // ajouter un node fils au dessus
button02 = new JButton ("⊕ ADD Fils ⊕ ▲ ");  button02.addActionListener(this);this.add(button02); // ajouter un node fils au-dessous
button01 = new JButton ("⊕ ADD Frèr ⊕ ▄▄▄");button01.addActionListener(this);this.add(button01); // ajouter un node frère au-dessous

button24 = new JButton ("ré-Afi Arb fm top");  button24.addActionListener(this);this.add(button24); // ré-affiche l'arbre depuis la lig 0

button16 = new JButton ("rePLI ↗↑ allArb");  button16.addActionListener(this);this.add(button16); // fold tout et profond dans le sens de lourd
button23 = new JButton ("rePLI  ↑  1 cran");  button23.addActionListener(this);this.add(button23); // toggle pliage/affichage du sélected node
button27 = new JButton ("élag bran pliées");  button27.addActionListener(this);this.add(button27); // élague les branches qui sont pliées
button14 = new JButton ("↕Tgl see NodSFils");  button14.addActionListener(this);this.add(button14); // toggle pliage/affichage du sélected node
button22 = new JButton ("dépLI  ↓  1 cran");  button22.addActionListener(this);this.add(button22); // toggle pliage/affichage du sélected node
button15 = new JButton ("dépLI ↙↓↘ allArb");  button15.addActionListener(this);this.add(button15); // toggle pliage/affichage du sélected node

button25 = new JButton ("split frè↔frèr ");  button25.addActionListener(this);this.add(button25); // split ce Neu entre lui&1frèr direct à créer
button31 = new JButton ("splt père↔fils ▀▄");  button31.addActionListener(this);this.add(button31); // split ce Neu entre lui&1frèr direct à créer
button38 = new JButton ("GATHER node→leaf");button38.addActionListener(this);this.add(button38); // rassemble les branches de ce Neu→feuille
```

```
button07 = new JButton ("MOVE _1_ hau  ↑ ");  button07.addActionListener(this);this.add(button07); // monter un node
button19 = new JButton ("MOVE all gau  ← ≡"); button19.addActionListener(this);this.add(button19); // promo la fratry, càd lui et ts ls pti Frèr
button08 = new JButton ("MOVE _1_ gau  ← -");button08.addActionListener(this);this.add(button08); // promouvoir only ce node
button06 = new JButton ("MOVE _1_ dro -  →");button06.addActionListener(this);this.add(button06); // dégrader un node
button26 = new JButton ("MOVE all dro ≡  →"); button26.addActionListener(this);this.add(button26); // dégrader un node
button05 = new JButton ("MOVE _1_ bas  ↓ ");  button05.addActionListener(this);this.add(button05); // descendre un node

button10 = new JButton ("copy    ^ c   ↓ ");        button10.addActionListener(this);this.add(button10); // copy → an inner PresPapier l'arb+fils but pas ses frères
button36 = new JButton ("copy    ^ c  ▯↓↘");       button36.addActionListener(this);this.add(button36); // copy → an inner PresPapier l'arb+fils ET ses frères
button12 = new JButton ("P A S T E    ^ V");       button12.addActionListener(this);this.add(button12); // colle from inner PresPapier l'arbr qu'on vient d'y mettre
button11 = new JButton (" cut    ^ x   ↓ ");       button11.addActionListener(this);this.add(button11); // coupe / rub l'arb cliqué + fils but pas ses frères
button37 = new JButton (" cut    ^ x  ▯↓↘");       button37.addActionListener(this);this.add(button37); // coupe / rub l'arb cliqué + ses fils ET ses frères

button29 = new JButton ("↔ swap Val Arb↔PP");button29.addActionListener(this);this.add(button29); // en intern. invert les val entre l'arb et le presse-papier
button30 = new JButton ("see iner arb stru");       button30.addActionListener(this);this.add(button30); // see inner mainNode struct sur la trace sur la console
button40 = new JButton ("pickUp images URL");button40.addActionListener(this);this.add(button40); // pickUp ls imag's URL used in la présenta°→cpyUrlUséBuf.sh

button09 = new JButton ("↑  MOV IMG UP  ↑");button09.addActionListener(this);this.add(button09); // scroll up the image à grands pas
button21 = new JButton ("↑  Mov Img Up  ↑");    button21.addActionListener(this);this.add(button21); // scroll up the image à moyens pas
button18 = new JButton ("↑  mov img up  ↑");     button18.addActionListener(this);this.add(button18); // scroll up the image à petits pas
```

5) Comment me joindre ?

Le samedi 21 mars 2020, j'ai mis en ligne la version 1 de jfMM.

Ainsi a commencé une période de test et de mise au point. Si vous éprouvez un problème pour installer jfMM ou l'utiliser, sentez-vous donc à l'aise pour m'écrire et me poser des questions qui m'aideront et serviront à tous.

Mais je rappelle deux restrictions :

- Pour installer Java voir mon pdf 'généSurJava-installYUsing', si c'est insuffisant reportez-vous à la doc de Sun.
- Je fournis jfMM pour que vous puissiez étudier mes cours (textes et illustrations), mais pas pour développer vos propres fichiers *.lsp.

Vous pouvez me joindre à l'adresse suivante :

jflucas at free.fr (remplacer le 'at' par l'arobas @).

Portez-vous bien.
Bien à vous !
Jean-François Lucas.