

ASSOCIATION APLDI
www.apldi.fr.st

SUPPORT DE COURS
D'INFORMATIQUES

CHAPITRE OPENOFFICE

Manuel d'installation

Contenu

À propos de ce manuel 5

Icônes utilisées dans ce manuel 5

Commentaires 6

Informations générales sur l'installation 6

Types d'installation 6

Configuration système requise 7

Généralités 7

Particularités concernant l'environnement d'exploitation Solaris™ (SPARC® Platform Edition) - environnement d'exploitation Solaris (SPARC) 7

Particularités concernant Linux 7

Particularités concernant Windows 8

Contenu des paquetages d'installation téléchargés 9

Opérations préalables à l'installation à partir d'un jeu téléchargé 10

Mise à jour d'une installation existante 10

Mise à jour d'une installation multiutilisateur 10

Installation monoutilisateur 10

Conditions requises pour l'installation 11

Démarrage de l'installation 11

Installation à partir d'un jeu téléchargé sur plate-forme UNIX 11

Installation à partir d'un jeu téléchargé sous Windows 11

Déroulement de l'installation 13

Écran de bienvenue 13

Informations importantes 14

Contrat de licence 14

Données d'identité 14

Choix du type d'installation 16

Sélection des composants 16

Répertoire d'installation	17
Vérification des options d'installation	18
Assignation des types de fichier	18
Installation de l'environnement d'exécution Java™	20
Démarrage du processus de copie	21
Installation : fin	21
Démarrage d'OpenOffice.org	21
Installation multiutilisateur ou en réseau	22
Installation serveur	23
Conditions requises pour l'installation	23
Démarrage de l'installation	23
Déroulement de l'installation	24
Installation de station de travail	31
Conditions requises pour l'installation	31
Démarrage de l'installation	31
Déroulement de l'installation	33
Appendice	38
Installation de l'environnement d'exécution Java™ sous Windows	39
Paramétrage des imprimantes, fax et polices pour les plates-formes UNIX®	39
Paramétrage des imprimantes	39
Intégration d'un périphérique fax	42
Connexion d'un convertisseur PostScript - PDF	43
Installation de polices	44
Installation d'un patch dans l'environnement d'exploitation Solaris™	45
Modification d'une installation OpenOffice.org existante	46
Modification	47
Réparation	47
Suppression	47
Paramètres d'installation	47
Démarrage d'OpenOffice.org avec des paramètres	48

Démarrage d'OpenOffice.org à partir de lignes de commande	48
Paramètres de ligne de commande	48
Enregistrement d'OpenOffice.org	50
Extension d'OpenOffice.org	51
Installation, mise à jour et suppression d'extensions dans une installation monoutilisateur	
52	
Installation, mise à jour et suppression d'extensions dans une installation multiutilisateur	
52	

À propos de ce manuel

Icônes utilisées dans ce manuel

Certaines icônes ont pour but d'attirer votre attention sur des informations supplémentaires.



L'icône "Important", représentée par un point d'exclamation, signale des informations importantes relatives à la sécurité des données et du système.



L'icône "Remarque", représentée par une punaise, signale des informations supplémentaires, par exemple d'autres méthodes permettant d'atteindre le but recherché.



L'icône "Astuce", représentée par une ampoule électrique, signale des astuces permettant de travailler plus efficacement avec le programme.



L'icône "Solaris" signale des informations qui ne s'appliquent qu'à l'environnement d'exploitation Solaris™.



L'icône "Linux" signale des informations qui ne s'appliquent qu'aux plates-formes Linux.



L'icône "Unix" signale des informations qui s'appliquent à toutes les plates-formes basées sur UNIX® (environnement d'exploitation Solaris et Linux).



L'icône "Win" signale des informations concernant le système d'exploitation Microsoft Windows.

Commentaires

Notre souci constant est d'améliorer notre documentation. Aussi, n'hésitez pas à nous envoiez vos suggestions, plaintes, critiques, éloges à l'adresse :

<http://fr.openoffice.org>
mailing liste : doc@fr.openoffice.org

Nous vous enverrons un message confirmant la réception de votre e-mail. Nous sommes très concernés par vos commentaires et nous les utiliserons dans le seul but de parfaire la documentation d'OpenOffice.org. Notez qu'il ne s'agit pas de l'adresse e-mail de transmission des demandes de support technique.

Informations générales sur l'installation

Ce chapitre présente des informations générales sur l'installation d'OpenOffice.org ainsi que sur les configurations système requises pour exécuter ce logiciel. Les informations relatives à la modification d'une installation OpenOffice.org 1.x existante sont regroupées dans l'annexe, section "Modification d'une installation OpenOffice.org existante " à la page 47.

Types d'installation

Type d'installation	But d'utilisation
Installation monoutilisateur (page 11)	Installation de OpenOffice.org sur un seul ordinateur pour un utilisateur.
Installation multiutilisateur ou en réseau (page 23)	Installation du logiciel OpenOffice.org soit sur un seul ordinateur destiné à plusieurs utilisateurs ou à des utilisateurs possédant des droits d'accès limités, soit sur un ordinateur relié au réseau pour une utilisation partagée. L'installation en réseau comprend deux étapes : <ol style="list-style-type: none">1. Installation serveur - installation des composants communs sur un seul ordinateur (installation généralement effectuée par un administrateur système).2. Installation de station de travail - installation des fichiers utilisateur (paramètres d'utilisateur compris) sur une seule station de travail.



Si vous installez OpenOffice.org sur un système Windows® en tant qu'utilisateur avec des droits d'accès limités (utilisateur ne possédant pas les droits d'administrateur), le logiciel n'est pas installé correctement.

Dans ce cas, procédez à une **Installation serveur** en tant qu'administrateur avant d'effectuer une **Installation de station de travail** en tant qu'utilisateur (voir page 23).

Configuration système requise

Généralités

- unité CD-ROM (si installation à partir d'un CD-ROM) ;
- environ 300 Mo d'espace disponible sur le disque dur pour une installation standard ;
- écran avec une résolution de 800 x 600 et 256 couleurs.



Particularités concernant l'environnement d'exploitation Solaris™ (SPARC® Platform Edition) - environnement d'exploitation Solaris (SPARC)

- environnement d'exploitation Solaris™ 8 ou version ultérieure (Solaris 9 recommandé) ;
- patchs correctifs 108434-10, 108773-17, 109147-21 et 108435-10 (64 bits uniquement) requis pour l'environnement d'exploitation Solaris 8 ;
- XServer (résolution d'écran 800 x 600 et 256 couleurs) avec un gestionnaire de fenêtres (par exemple, OpenWindows™, CDE ou GNOME) - GNOME 2.0 ou plus requis pour le support d'accessibilité ;
- 128 Mo de RAM au minimum.



Vous pouvez obtenir les patchs pour l'environnement d'exploitation Solaris sur Internet à l'adresse <http://sunsolve.sun.com>.



Particularités concernant Linux

- PC équipé d'un processeur Pentium ou compatible ;
- Linux Kernel 2.2.13 ou plus ;
- glibc2 version 2.2.0 ou ultérieure ;
- XServer (résolution d'écran 800 x 600 et 256 couleurs) avec un gestionnaire de fenêtres (par exemple, GNOME ou KDE) - GNOME 2.0 ou plus requis pour le support d'accessibilité ;
- 128 Mo de RAM au minimum.



L'installation de OpenOffice.org n'est pas possible sur une partition (V)FAT d'un système Linux, car la création de liens symboliques n'est pas supportée par les systèmes de fichiers FAT.



Particularités concernant Windows

- Microsoft Windows 98, ME, NT (Service Pack 6 ou version ultérieure), 2000 ou XP - système Windows 98/ME natif requis pour le support des langues asiatiques sous Windows 98/ME ;
- PC équipé d'un processeur Pentium ou compatible ;
- 64 Mo de RAM.
- Notez qu'il est impossible de procéder à l'installation sans posséder les droits d'administrateur requis.

Contenu des paquetages d'installation téléchargés

Les paquetages d'installation téléchargés comprennent l'application Setup de OpenOffice.org et le manuel d'installation (ce document) au format PDF (Portable Document Format). Le contenu des paquetages d'installation pour les différentes plates-formes est répertorié dans le tableau suivant :

Plate-forme	Fichiers d'installation
Windows	Programme d'installation de OpenOffice.org : Ooo_1.1_Win32Intel_Install_{lang}.zip
Environnement d'exploitation Solaris (SPARC)	Programme d'installation de OpenOffice.org : Ooo_1.1_SolarisSparc_Install_{lang}.tar.gz
Linux	Programme d'installation de OpenOffice.org : Ooo_1.1_LinuxIntel_Install_{lang}.tar.gz



{lang} fait référence au code langue à deux caractères spécifié par l'ISO 639 (par exemple, "fr" pour le français). Dans certains cas, l'ISO 639 a été élargi aux deux caractères du code pays comme le spécifie l'ISO 3166 (par exemple, "US" pour les États-Unis).

Opérations préalables à l'installation à partir d'un jeu téléchargé

Placez tous les fichiers d'installation téléchargés dans le même dossier.



Vérifiez que les fichiers d'installation ont les droits exécutables. Si ce n'est pas le cas, utilisez la commande **chmod** pour modifier les droits.

Mise à jour d'une installation existante

Vous pouvez mettre à jour une installation existante d'OpenOffice.org (1.0.3 et plus) vers OpenOffice.org 1.1 et continuer à utiliser les paramètres de configuration existants. Si l'application Setup détecte une version antérieure d'OpenOffice.org sur votre système, vous devez spécifier si vous voulez mettre à jour cette installation ou installer la dernière version dans un répertoire différent. Vous ne pouvez pas mettre à jour une version antérieure si elle n'est pas dans la même langue.



Nous déconseillons toute mise à jour avec une version bêta d'OpenOffice.org.

Mise à jour d'une installation multiutilisateur

L'application Setup ne met à jour que la partie serveur d'une installation multiutilisateur (voir aussi page 24). Au démarrage suivant d'OpenOffice.org, l'utilisateur est invité à mettre à jour les données utilisateur locales.

Installation monoutilisateur

L'installation monoutilisateur installe OpenOffice.org 1.1 sur un seul ordinateur pour un utilisateur.

Pour ce type d'installation, connectez-vous au système en tant qu'utilisateur normal, puis installez OpenOffice.org dans votre répertoire principal ou dans un autre répertoire local dans lequel vous possédez des droits d'accès illimités.



Vous pouvez interrompre le processus d'installation à tout moment en cliquant sur le bouton **Annuler** dans la boîte de dialogue d'installation. À ce stade, vous pouvez également supprimer le répertoire d'installation. Dans ce cas, **tous les fichiers et sous-dossiers** sont supprimés.

Conditions requises pour l'installation

Selon les options choisies, l'installation monoutilisateur requiert entre 190 et 250 Mo d'espace sur le disque dur. Au cours de l'installation, environ 40 Mo supplémentaires sont nécessaires pour les fichiers temporaires sur les systèmes Windows et 80 Mo sur les plates-formes UNIX®.

Sur les systèmes sur lesquels une version antérieure d'OpenOffice.org est déjà installée, ouvrez l'un des fichiers suivants pour vérifier le numéro de la version installée :

- **.sversionrc** (systèmes **UNIX**) ;
- **sversion.ini** (systèmes **Windows**).

Sur les systèmes **Windows NT/2000** et **Win9x** configurés pour plusieurs utilisateurs, le fichier sversion.ini se trouve dans "C:\Documents and Settings\{nom de l'utilisateur}\Application Data".

Ces fichiers contiennent le chemin et le numéro de la version d'OpenOffice.org installée. Si le numéro de version est **identique** à celui de la version à installer, vous devez préalablement désinstaller OpenOffice.org pour pouvoir le réinstaller. S'il est antérieur à la version à installer et postérieur à la version 1.0.2, vous pouvez opter pour la mise à jour de l'installation (voir Mise à jour d'une installation existante, page 11).

Démarrage de l'installation

Installation à partir d'un jeu téléchargé sur plate-forme **UNIX**

1. Connectez-vous sous votre nom d'utilisateur (l'installation monoutilisateur ne requiert aucun droit d'administrateur système).
2. Activez l'interface graphique X Window.
3. Accédez au répertoire contenant les fichiers d'installation téléchargés.
4. Lancez l'application Setup à l'aide de la commande :

```
./setup
```

Reportez-vous également à la section Contenu des paquetages d'installation téléchargés, page 10.

Installation à partir d'un jeu téléchargé sous Windows

1. Si nécessaire, connectez-vous sous votre nom d'utilisateur (l'installation monoutilisateur ne requiert aucun droit d'administrateur système).
2. Accédez au répertoire contenant les fichiers d'installation téléchargés.

3. Effectuez l'une des opérations suivantes :

- Double-cliquez sur setup.exe
- Ouvrez le menu **Démarrer** de Windows, choisissez **Exécuter**, puis saisissez X:\inst-dir\setup dans le champ **Ouvrir**.
- X:\instdir est le répertoire contenant les fichiers du jeu téléchargé.

4. Cliquez sur **OK** pour lancer l'installation.

Reportez-vous également à la section Contenu des paquetages d'installation téléchargés, page 10.

Déroulement de l'installation

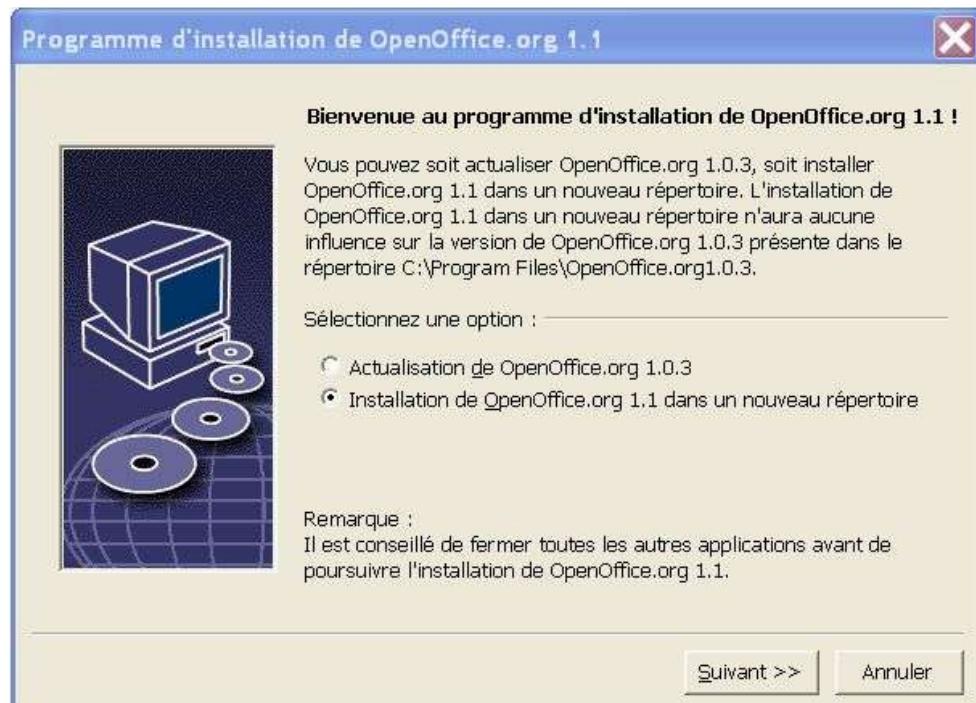
Au cours de l'installation, cliquez sur le bouton **Aide** pour afficher des informations sur la boîte de dialogue active, puis cliquez sur **Précédent** pour revenir au programme d'installation. NE FERMEZ PAS la fenêtre d'aide à l'aide du bouton "X" (en haut à droite) de la fenêtre : ceci aurait pour conséquence d'interrompre le programme d'installation.

Écran de bienvenue

L'application Setup commence par afficher la boîte de dialogue de bienvenue.



- cliquez sur **Suivant**.



Si l'application Setup détecte une version antérieure d'OpenOffice.org (1.0.3 ou plus), vous devez spécifier si vous voulez mettre à jour l'installation ou installer la dernière version dans un nouveau répertoire.

- Choisissez une option d'installation, puis cliquez sur **Suivant**.

Informations importantes

Le fichier "readme" s'affiche dans une fenêtre. Comme ce fichier est placé dans le répertoire d'installation sur votre ordinateur, vous pouvez facilement y accéder ultérieurement.

- Lisez les informations affichées, puis cliquez sur **Suivant**.

Contrat de licence

La boîte de dialogue suivante affiche le contrat de licence. Si vous n'acceptez pas le contrat de licence, l'installation d'OpenOffice.org est interrompue.

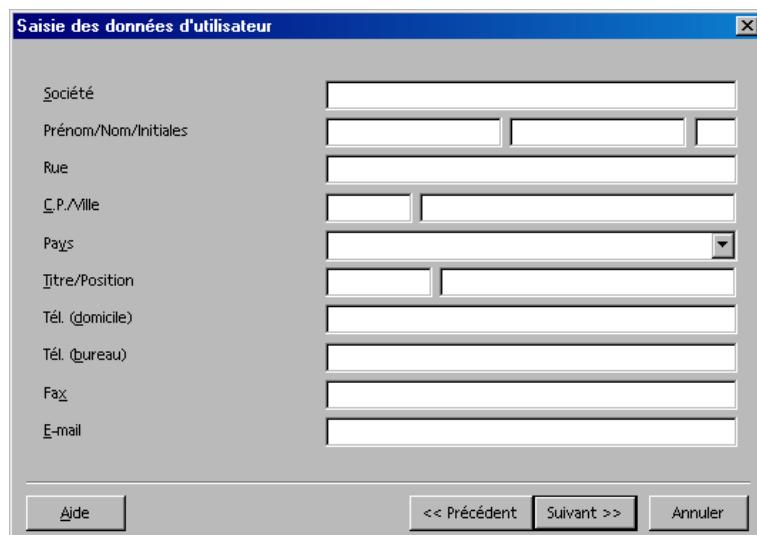
- Lisez attentivement ce contrat. Si vous en acceptez les termes, faites défiler le texte jusqu'à la fin, sélectionnez **J'accepte les conditions du contrat**, puis cliquez sur **OK**.
- Dans le cas contraire, cliquez sur **Refuser** pour quitter le programme d'installation.

Données d'identité

Si vous n'importez pas vos données personnelles, la boîte de dialogue **Saisie des données d'identité** apparaît.

Les données saisies ici sont utilisées par OpenOffice.org pour personnaliser les modèles et les documents.

Vous pouvez ensuite accéder à cette boîte de dialogue à partir de n'importe quelle application OpenOffice.org, en sélectionnant **Outils - Options - OpenOffice.org - Données d'identité**.



- Saisissez vos données personnelles.
- Cliquez sur **Suivant** pour poursuivre l'installation.

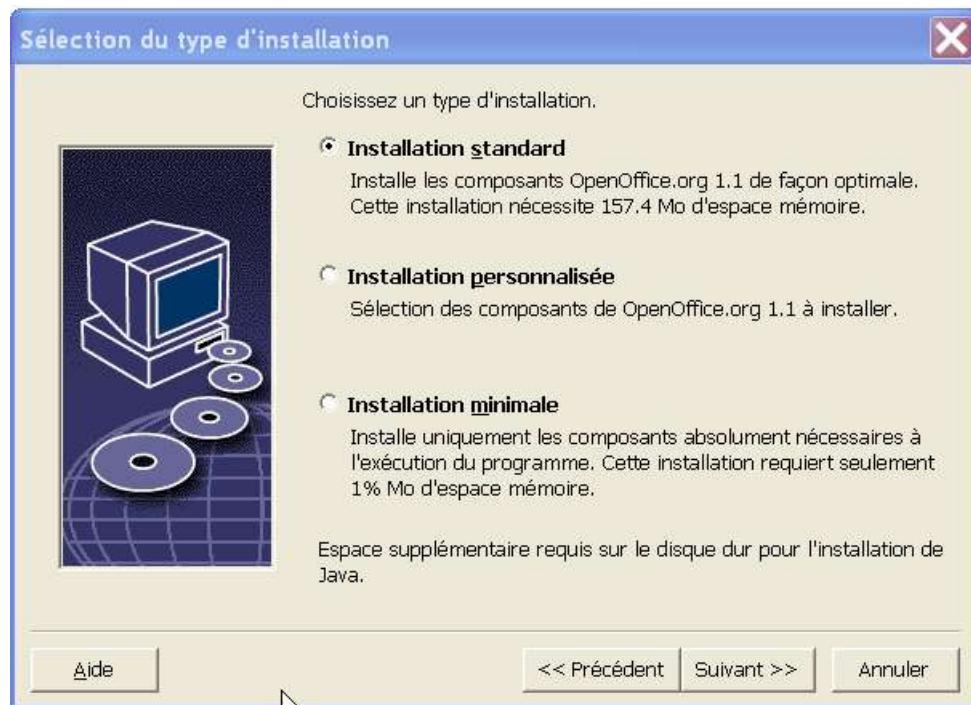
Choix du type d'installation

Cette boîte de dialogue permet de sélectionner le type d'installation souhaité. Elle n'apparaît pas si vous effectuez une mise à jour d'une version antérieure d'OpenOffice.org.

L'espace disque requis indiqué ici n'est qu'une estimation et dépend des propriétés du média de stockage.

Vous avez le choix entre trois types d'installation :

- **Installation standard** - recommandée à la plupart des utilisateurs. Elle installe tous les composants d'OpenOffice.org, ainsi qu'une sélection de filtres.
- **Installation personnalisée** - recommandée aux utilisateurs expérimentés. Elle permet de sélectionner les composants à installer.
- **Installation minimale** - recommandée aux utilisateurs ayant des contraintes d'espace. Cette option installe uniquement les composants nécessaires à l'exécution d'OpenOffice.org.



- Sélectionnez un type d'installation.
- Cliquez sur **Suivant** pour poursuivre l'installation.

Sélection des composants

Si vous sélectionnez l'option **Installation personnalisée**, vous pouvez ensuite choisir les modules et les composants à installer.

Cliquez sur le signe (+) placé devant le nom d'une catégorie de modules ou de composants pour en afficher la liste. Cliquez sur la catégorie ou le composant à installer. Si une catégorie

contient des composants non sélectionnés, la case située devant le nom de la catégorie est semi-transparente.

- Les catégories en **bleu foncé** sont installées avec tous leurs composants.
- Les catégories en **bleu clair** sont installées avec seulement certains de leurs composants.
- Les catégories en **blanc** ne sont pas installées.

Pour restaurer les options de l'installation standard, cliquez sur le bouton **Par défaut**

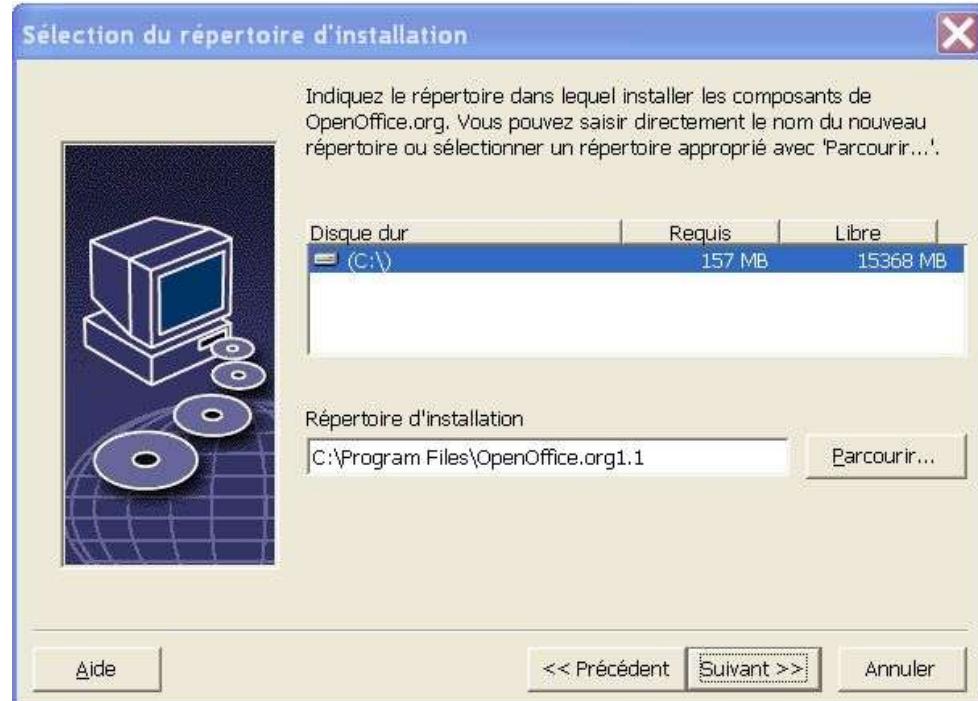


- Sélectionnez les modules et les composants à inclure dans l'installation.
- Cliquez sur **Suivant** pour poursuivre l'installation.

Répertoire d'installation

La boîte de dialogue suivante vous permet de sélectionner le répertoire d'installation. Elle n'apparaît pas si vous procédez à une mise à jour de l'installation.

Les unités de disque disponibles sur votre système sont affichées dans cette boîte de dialogue. L'espace disque requis indiqué ici n'est qu'une estimation et dépend des propriétés du média de stockage.



- Saisissez le chemin du **répertoire d'installation** dans la zone de texte ou cliquez sur **Parcourir** pour sélectionner le répertoire.
Si ce répertoire n'existe pas, vous devez le créer.
- Cliquez sur **Suivant** pour poursuivre l'installation.

Vérification des options d'installation

Après la sélection des options d'installation, une boîte de dialogue s'affiche pour vous permettre de vérifier les paramètres.

- Cliquez sur **Installer** pour poursuivre l'installation ou sur **Précédent** pour modifier les paramètres sélectionnés.

Assignation des types de fichier

Utilisez cette boîte de dialogue pour assigner à OpenOffice.org des types de fichier Microsoft et des documents HTML. Ces informations sont enregistrées dans le système d'exploitation.



- Sélectionnez les **types de fichier** qu'OpenOffice.org doit ouvrir.
- Si vous souhaitez utiliser OpenOffice.org Writer comme éditeur par défaut pour les fichiers HTML, cochez la case au-dessous de **Éditeur HTML par défaut**.
- Cliquez sur **OK**.

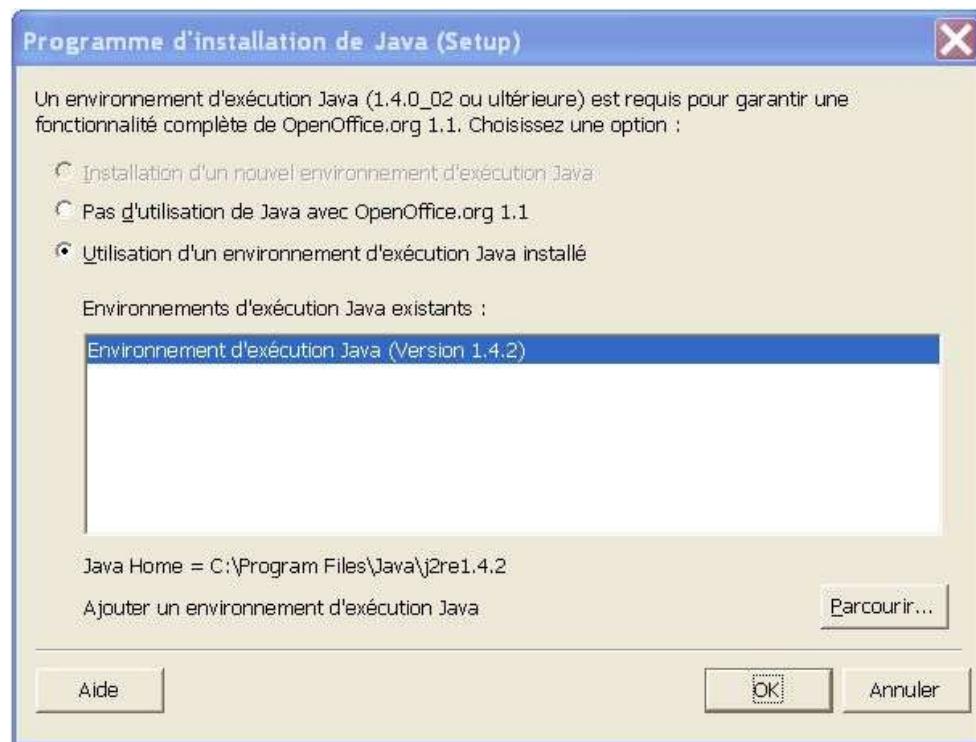


Pour modifier les assignations de types de fichier d'une installation existante, lancez l'application Setup et sélectionnez l'option **Modifier** (voir Modification, page 48).

Installation de l'environnement d'exécution Java™

La boîte de dialogue suivante répertorie les différentes versions de l'environnement d'exécution Java™ (JRE) présentes sur votre système, ou vous propose d'installer cet environnement.

Sélectionnez la version JRE à utiliser. La version recommandée est la 1.4.1_01 ou plus. Si l'environnement d'exécution Java est déjà installé, vous n'avez pas besoin d'installer la version fournie.



- Pour plus d'informations sur l'installation de l'environnement d'exécution Java, reportez-vous à l'appendice de cette documentation.
- Si l'environnement d'exécution Java est déjà installé sur votre système, sélectionnez la version à utiliser et cliquez sur **OK**.

Démarrage du processus de copie

La progression de l'installation est indiquée sous forme de pourcentage avec une estimation du temps restant.

Installation : fin

Lorsque l'installation est terminée, une dernière boîte de dialogue s'affiche.

- Cliquez sur **Terminer** pour quitter l'installation.



Selon le gestionnaire de fenêtres utilisé, vous devez vous déconnecter et redémarrer le gestionnaire de fenêtres.

Démarrage d'OpenOffice.org



Accédez au répertoire **{installpath}/program** ou au répertoire local d'OpenOffice.org dans votre répertoire principal et saisissez :

`./soffice`

Vous pouvez aussi ajouter le répertoire **{installpath}/program** dans le chemin des programmes ; il vous suffit ensuite de saisir "soffice" à partir de n'importe quel répertoire pour lancer OpenOffice.org.

Pour paramétriser l'imprimante par défaut pour OpenOffice.org, activez le programme de gestion des imprimantes **spadmin** (reportez-vous à l'appendice de cette documentation).

Dans l'environnement d'exploitation **Solaris**, déconnectez-vous une fois l'installation terminée et reconnectez-vous afin d'actualiser l'intégration CDE.



Les icônes permettant de lancer les applications OpenOffice.org sont ajoutées dans le menu **Démarrer** de Windows sous **Programmes/OpenOffice.org 1.1**.

Au cours de l'installation, un lien renvoyant au démarrage rapide de OpenOffice.org est créé dans le dossier Démarrage du menu Démarrer de **Windows**. L'icône de démarrage rapide est ajoutée dans la barre des tâches au démarrage de Windows.

Un clic avec le bouton droit de la souris sur l'icône de démarrage rapide permet de sélectionner l'application OpenOffice.org à activer.

Pour plus d'informations sur le démarrage rapide, reportez-vous à l'aide en ligne.

Vous trouverez des informations sur le démarrage d'OpenOffice.org avec des paramètres de ligne de commande dans l'appendice de cette documentation.

Installation multiutilisateur ou en réseau

L'installation d'OpenOffice.org pour plusieurs utilisateurs en réseau comporte deux étapes :

1. **Installation serveur** - installation par l'administrateur système à l'aide de la commande **setup -net**. Les fichiers OpenOffice.org sont copiés dans un répertoire dans lequel tous les utilisateurs ont des droits de lecture et d'exécution (voir page 24).
 2. **Installation d'utilisateur** - chaque utilisateur installe une copie complète d'OpenOffice.org dans son répertoire principal ou seulement les fichiers utilisateur. Cette dernière option est appelée **installation de station de travail** ; elle nécessite l'accès à l'**installation serveur** pour l'exécution des applications d'OpenOffice.org (voir page 32).
-



Si vous installez OpenOffice.org sur un système Windows en tant qu'utilisateur avec des droits d'accès limités (utilisateur ne possédant pas les droits d'administrateur), le logiciel n'est pas installé correctement.

Dans ce cas, procédez à une **Installation serveur** en tant qu'administrateur avant d'effectuer une **Installation de station de travail** en tant qu'utilisateur (voir page 23).

Installation serveur

L'installation serveur **n'est pas** une version autonome d'OpenOffice.org™. Pour pouvoir utiliser OpenOffice.org, vous **devez** effectuer une installation de station de travail à partir de l'installation serveur (voir "Installation de station de travail", page 32).

Conditions requises pour l'installation

L'installation serveur d'OpenOffice.org nécessite environ 250 Mo d'espace disque. Les fichiers d'installation temporaires générés au cours de l'installation demandent environ 20 Mo supplémentaires. Ces fichiers temporaires sont automatiquement supprimés après l'installation. Sous UNIX®, un volume d'échange temporaire de 80 Mo est créé au cours de l'installation et supprimé ensuite.

Démarrage de l'installation

Installation à partir d'un jeu téléchargé sur plate-forme UNIX

1. Connectez-vous en tant qu'**administrateur système**.
2. Activez l'interface graphique X Window.
3. Accédez au répertoire contenant les fichiers d'installation téléchargés.
4. Lancez l'application Setup à l'aide de la commande :

```
./setup -net
```

Reportez-vous également à la section "Contenu des paquetages d'installation téléchargés", page 10.

Installation à partir d'un jeu téléchargé sur plate-forme Windows

1. Connectez-vous en tant qu'**administrateur système**.
2. Ouvrez le menu **Démarrer** de Windows, choisissez **Exécuter**, puis saisissez X:\instdir\setup -net dans le champ **Ouvrir**.

X:\instdir est le répertoire contenant les fichiers du jeu téléchargé.

3. Cliquez sur **OK** pour lancer l'installation.

Reportez-vous également à la section Contenu des paquetages d'installation téléchargés, page 10.



Lorsque vous mettez à jour une installation existante d'OpenOffice.org, vous devez éventuellement spécifier l'emplacement de cette version.

Si le programme d'installation ne détecte pas automatiquement l'ancienne version à mettre à jour, exécutez l'application Setup en spécifiant le paramètre **-update** :

```
./<setup> -net -update:<chemin_vers_installationserveur_précédente>
```

<setup> est le nom du programme d'installation à exécuter (voir ci-dessus) et **<chemin_vers_installationserveur_précédente>** est le chemin d'accès complet à l'installation précédente de OpenOffice.org.

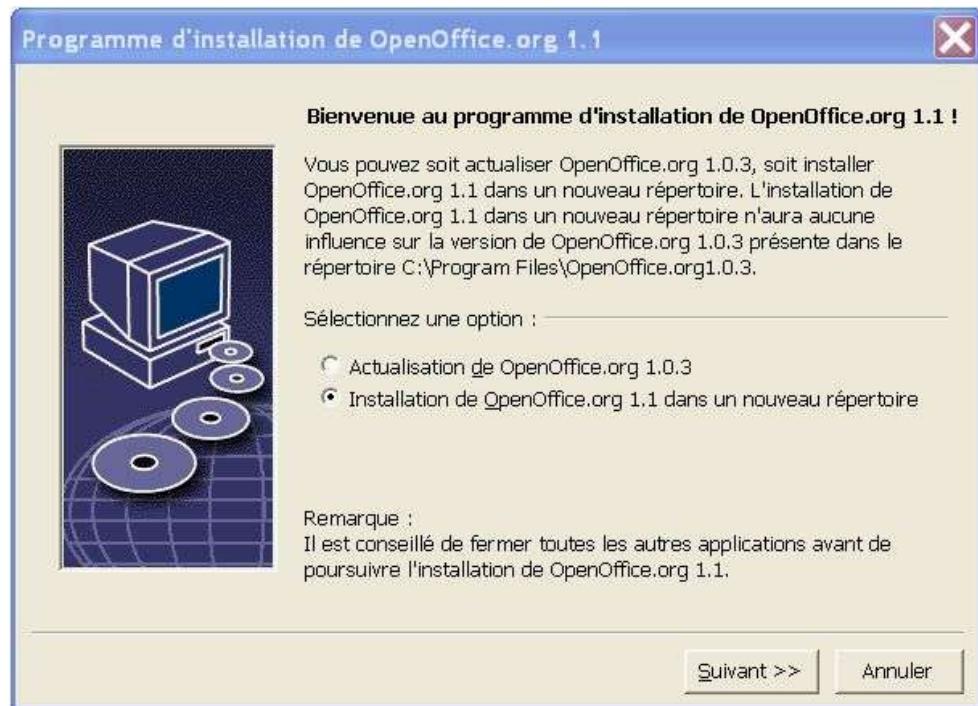
Déroulement de l'installation

Écran de bienvenue

L'application Setup commence par afficher la boîte de dialogue de bienvenue.



- Cliquez sur **Suivant**.



Si l'application Setup détecte une version antérieure d'OpenOffice.org (1.0.3 ou plus), vous devez spécifier si vous voulez mettre à jour l'installation ou installer la dernière version dans un nouveau répertoire.

- Choisissez une option d'installation, puis cliquez sur **Suivant**.

Informations importantes

Le fichier "readme" s'affiche dans une fenêtre. Comme ce fichier est placé dans le répertoire d'installation sur votre ordinateur, vous pouvez facilement y accéder ultérieurement.

- Lisez les informations affichées, puis cliquez sur **Suivant**.

Contrat de licence

La boîte de dialogue suivante affiche le contrat de licence. Si vous n'acceptez pas le contrat de licence, l'installation d'OpenOffice.org est interrompue.

- Lisez attentivement ce contrat. Si vous en acceptez les termes, faites défiler le texte jusqu'à la fin, sélectionnez **J'accepte les conditions du contrat**, puis cliquez sur **OK**.
- Dans le cas contraire, cliquez sur **Refuser** pour quitter le programme d'installation.

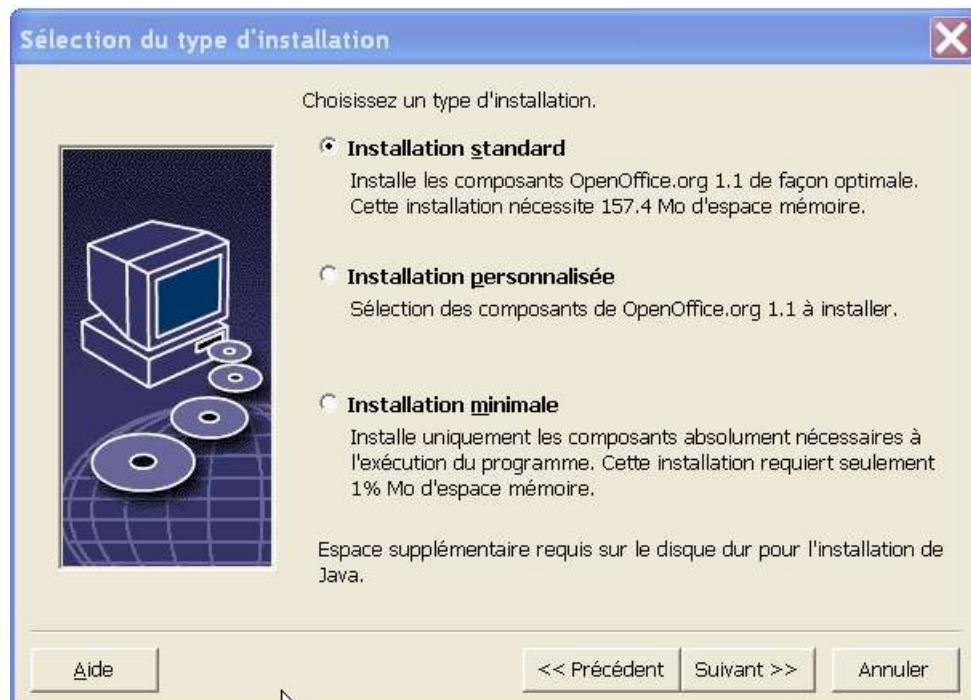
Choix du type d'installation

Cette boîte de dialogue permet de sélectionner le type d'installation souhaité. Elle n'apparaît pas si vous effectuez une mise à jour d'une version antérieure d'OpenOffice.org.

L'espace disque requis indiqué ici n'est qu'une estimation et dépend des propriétés du média de stockage.

Vous avez le choix entre trois types d'installation :

- **Installation standard** - recommandée à la plupart des utilisateurs. Elle installe tous les composants d'OpenOffice.org, ainsi qu'une sélection de filtres.
- **Installation personnalisée** - recommandée aux utilisateurs expérimentés. Elle permet de sélectionner les composants à installer.
- **Installation minimale** - recommandée aux utilisateurs ayant des contraintes d'espace. Cette option installe uniquement les composants nécessaires à l'exécution d'OpenOffice.org.



- Choisissez l'**installation personnalisée**. Au cours de l'installation serveur, installez tous les composants d'OpenOffice.org afin que chaque utilisateur puisse accéder à ces options lors de l'installation de station de travail.
- Cliquez sur **Suivant** pour poursuivre l'installation.

Sélection des composants

Si vous sélectionnez l'option **Installation personnalisée**, vous pouvez ensuite choisir les modules et les composants à installer.

Cliquez sur le signe (+) placé devant le nom d'une catégorie de modules ou de composants pour en afficher la liste. Cliquez sur la catégorie ou le composant à installer. Si une catégorie contient des composants non sélectionnés, la case située devant le nom de la catégorie est semi-transparente.

- ⬇ Les catégories en **bleu foncé** sont installées avec tous leurs composants.
- ⬇ Les catégories en **bleu clair** sont installées avec seulement certains de leurs composants.

-  Les catégories en **bleu foncé** sont installées avec tous leurs composants.
-  Les catégories en **blanc** ne sont pas installées.

Pour restaurer les options de l'installation standard, cliquez sur le bouton **Par défaut**.

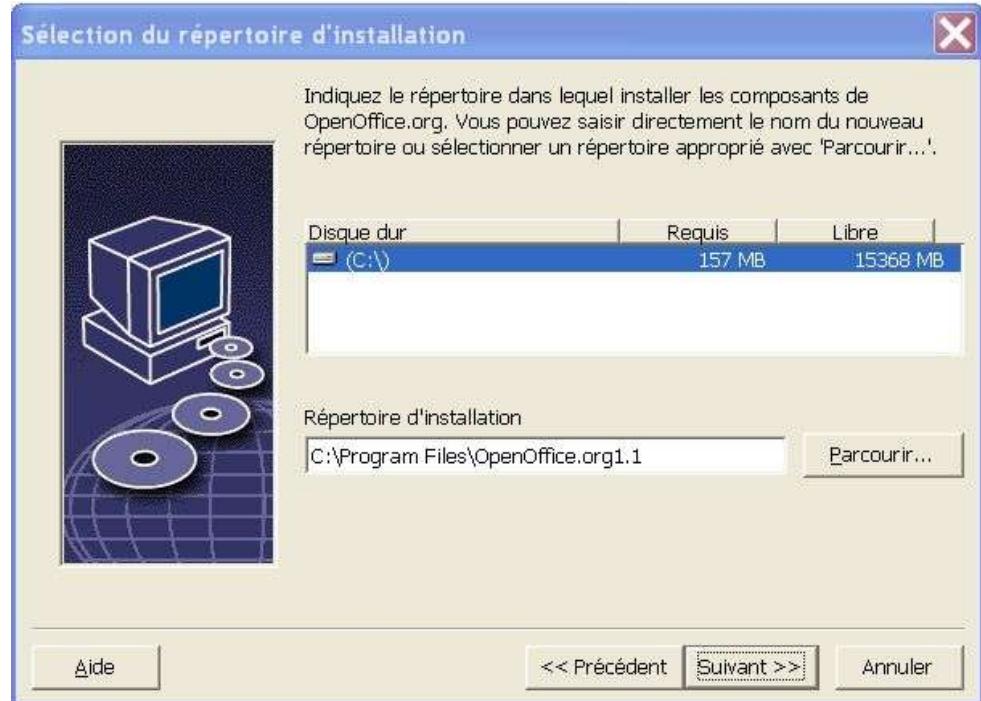


- Sélectionnez les modules et les composants à inclure dans l'installation.
- Cliquez sur **Suivant** pour poursuivre l'installation.

Répertoire d'installation

La boîte de dialogue suivante vous permet de sélectionner le répertoire d'installation. Elle n'apparaît pas si vous procédez à une mise à jour de l'installation.

Les unités de disque disponibles sur votre système sont affichées dans cette boîte de dialogue. L'espace disque requis indiqué ici n'est qu'une estimation et dépend des propriétés du média de stockage.



- Saisissez le chemin du **répertoire d'installation** dans la zone de texte ou cliquez sur **Parcourir** pour sélectionner le répertoire.

Si ce répertoire n'existe pas, vous devez le créer.

- Cliquez sur **Suivant** pour poursuivre l'installation.

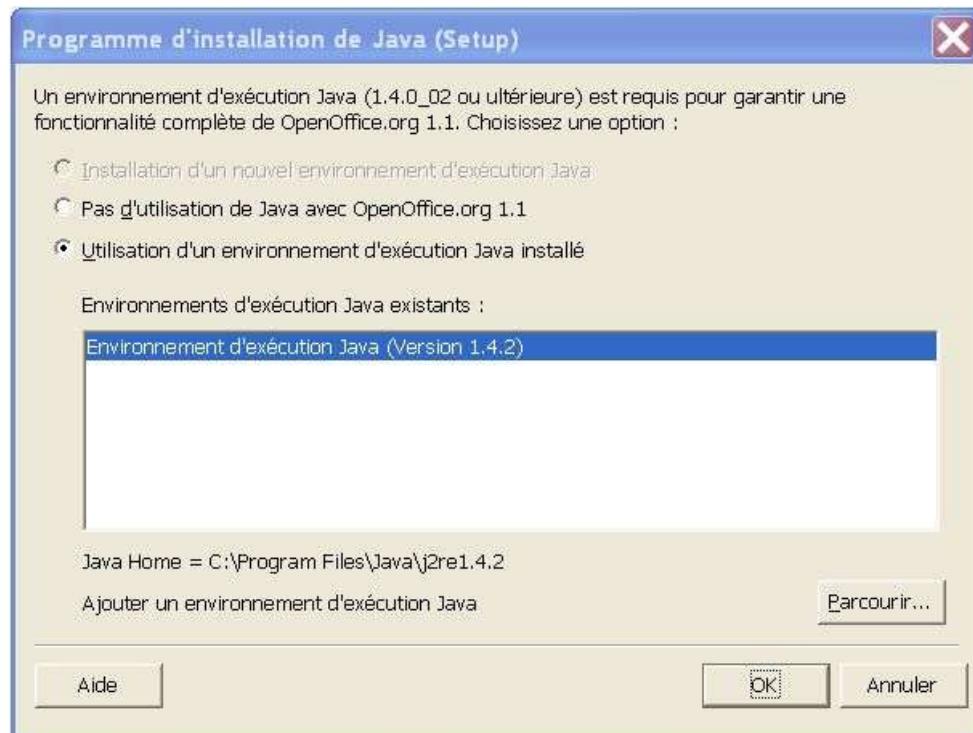


Vérifiez que le répertoire sélectionné pour l'installation serveur possède des droits de lecture et d'exécution pour tous les utilisateurs : ceci est indispensable pour l'exécution des installations de station de travail.

Installation de l'environnement d'exécution Java™

La boîte de dialogue suivante répertorie les différentes versions de l'environnement d'exécution Java™ (JRE) présentes sur votre système, ou vous propose d'installer cet environnement.

Sélectionnez la version JRE à utiliser. La version recommandée est la 1.4.1_01 ou plus. Si l'environnement d'exécution Java est déjà installé, vous n'avez pas besoin d'installer la version fournie.



- Pour plus d'informations sur l'installation de l'environnement d'exécution Java, reportez-vous à l'appendice de cette documentation.
- Si l'environnement d'exécution Java est déjà installé sur votre système, sélectionnez la version à utiliser et cliquez sur **OK**.



L'installation de l'environnement d'exécution Java (JRE) est une installation partagée pour tous les utilisateurs. Si vous n'effectuez pas l'installation à ce stade, les utilisateurs peuvent s'en charger lors de l'installation de station de travail.

Vérification des options d'installation

Après la sélection des options d'installation, une boîte de dialogue s'affiche pour vous permettre de vérifier les paramètres.

- Cliquez sur **Installer** pour poursuivre l'installation ou sur **Précédent** pour modifier les paramètres sélectionnés.

Démarrage du processus de copie

La progression de l'installation est indiquée sous forme de pourcentage avec une estimation du temps restant.

Installation : fin

Lorsque l'installation est terminée, une dernière boîte de dialogue s'affiche.

- Cliquez sur **Terminer** pour quitter l'installation.



Selon le gestionnaire de fenêtres utilisé, vous devez vous déconnecter et redémarrer le gestionnaire de fenêtres.

Une fois l'installation serveur effectuée, chaque utilisateur peut effectuer l'**installation de station de travail**, c'est-à-dire copier les fichiers requis dans son répertoire principal ou disque dur local. L'installation de station de travail est décrite dans le chapitre suivant.

Si vous envisagez d'effectuer un grand nombre d'installations à partir d'une installation serveur, vous pouvez désactiver la boîte de dialogue d'enregistrement qui apparaît au deuxième démarrage d'OpenOffice.org. Pour plus de détails, reportez-vous à la page 51 en appendice.

Installation de station de travail

Avant de pouvoir utiliser OpenOffice.org en réseau, vous devez effectuer une **installation de station de travail**. L'accès aux principaux composants d'OpenOffice.org, présents dans l'installation serveur centrale, est indispensable pour les installations de station de travail.



Après la mise à jour d'une installation serveur existante, les utilisateurs doivent également mettre à jour les installations de station de travail.

Conditions requises pour l'installation

L'installation de station de travail nécessite au minimum 20 Mo d'espace disponible.

Démarrage de l'installation

Avant d'installer OpenOffice.org sur une station de travail, vous devez effectuer une **installation serveur**. Pour plus de détails, reportez-vous à la section Installation serveur, page 24.

Sur les systèmes sur lesquels une version antérieure de OpenOffice.org est déjà installée, ouvrez l'un des fichiers suivants pour vérifier le numéro de la version installée :

- **.sversionrc** (systèmes UNIX) ;
- **sversion.ini** (systèmes Windows).

Sur les systèmes **Windows NT/2000** et **Win9x** configurés pour plusieurs utilisateurs, le fichier sversion.ini se trouve dans "C:\Documents and Settings\{nom de l'utilisateur}\Application Data".

Ces fichiers contiennent le chemin et le numéro de la version d'OpenOffice.org installée. Si le numéro de version est **identique** à celui de la version à installer, vous devez préalablement désinstaller OpenOffice.org pour pouvoir le réinstaller. S'il est antérieur à la version à installer et postérieur à la version 1.0.2, vous pouvez opter pour la mise à jour de l'installation (voir Mise à jour d'une installation existante, page 11).

Installation sur plate-forme UNIX

1. Connectez-vous sous votre **nom d'utilisateur**.
2. Activez l'interface graphique X Window.
3. Accédez au sous-répertoire **program** dans le répertoire d'installation de l'**installation serveur**. **N'utilisez pas** l'original du fichier d'installation (setup) d'OpenOffice.org.
4. Lancez le script d'installation à l'aide de la commande :
`./setup`

Installation sur plate-forme Windows

1. Connectez-vous sous votre **nom d'utilisateur**.
2. Accédez au sous-répertoire **program** dans le répertoire d'installation de l'**installation serveur**. **N'utilisez pas** l'original du fichier d'installation (setup) d'OpenOffice.org.
3. Effectuez l'une des opérations suivantes
 - Double-cliquez sur setup
 - Ouvrez le menu **Démarrer** de Windows, choisissez **Exécuter**, puis saisissez {répertoire installation serveur}\program\setup dans le champ **Ouvrir**.
{répertoire installation serveur} est le répertoire dans lequel est placé le programme d'installation dans une installation serveur.
4. Cliquez sur **OK** pour lancer l'installation de station de travail.

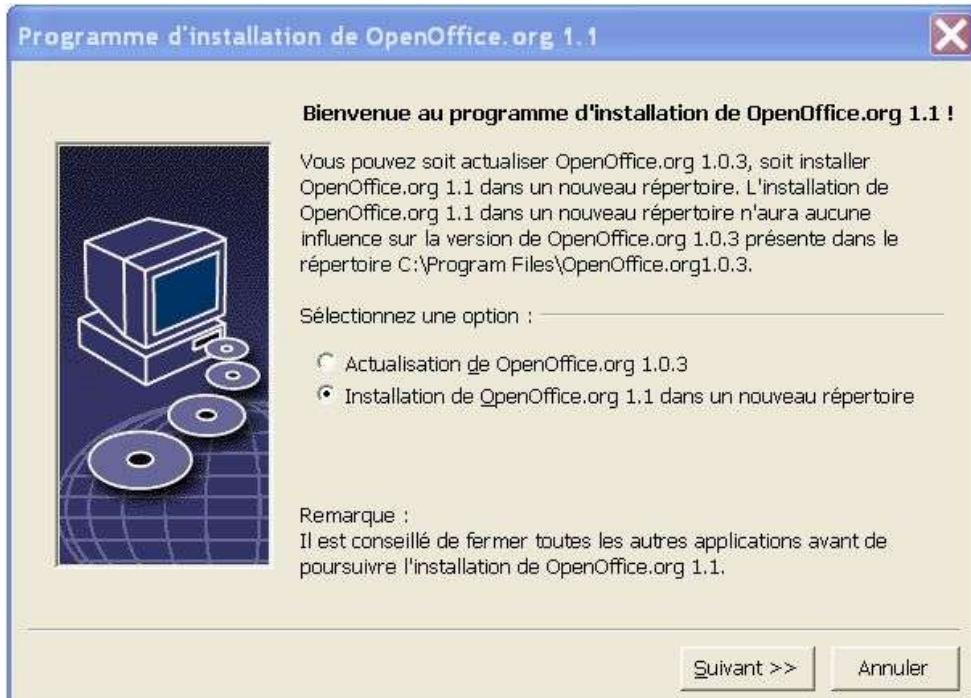
Déroulement de l'installation

Écran de bienvenue

L'application Setup commence par afficher la boîte de dialogue de bienvenue.



- Cliquez sur **Suivant**.



Si l'application Setup détecte une version antérieure d'OpenOffice.org (1.0.3 ou plus), vous devez spécifier si vous voulez mettre à jour l'installation ou installer la dernière version dans un nouveau répertoire.

- Choisissez une option d'installation, puis cliquez sur **Suivant**.

Informations importantes

Le fichier "readme" s'affiche dans une fenêtre. Comme ce fichier est placé dans le répertoire d'installation sur votre ordinateur, vous pouvez facilement y accéder ultérieurement.

- Lisez les informations affichées, puis cliquez sur **Suivant**.

Contrat de licence

La boîte de dialogue suivante affiche le contrat de licence. Si vous n'acceptez pas le contrat de licence, l'installation d'OpenOffice.org est interrompue.

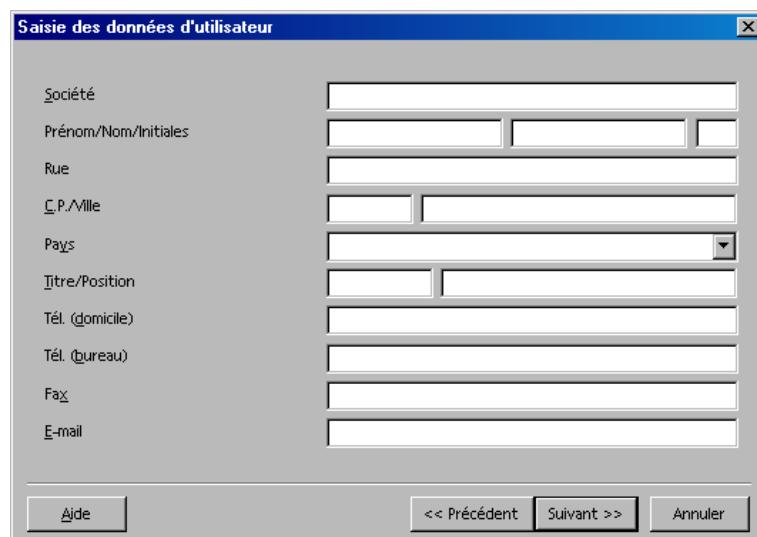
- Lisez attentivement ce contrat. Si vous en acceptez les termes, faites défiler le texte jusqu'à la fin, sélectionnez **J'accepte les conditions du contrat**, puis cliquez sur **OK**.
- Dans le cas contraire, cliquez sur **Refuser** pour quitter le programme d'installation.

Données d'identité

Si vous n'importez pas vos données personnelles, la boîte de dialogue **Saisie des données d'identité** apparaît.

Les données saisies ici sont utilisées par OpenOffice.org pour personnaliser les modèles et les documents.

Vous pouvez ensuite accéder à cette boîte de dialogue à partir de n'importe quelle application OpenOffice.org, en sélectionnant **Outils - Options - OpenOffice.org - Données d'identité**.



- Saisissez vos données personnelles.
- Cliquez sur **Suivant** pour poursuivre l'installation.

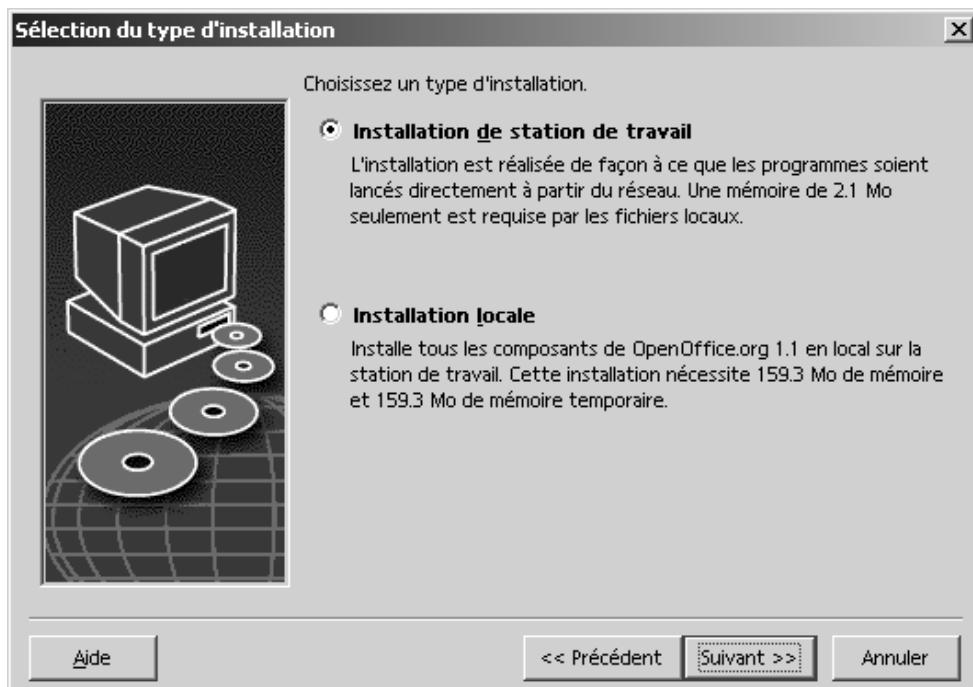
Type d'installation



Si, au lieu de la boîte de dialogue décrite ci-dessous, vous obtenez la boîte de dialogue proposant les types d'installation standard, personnalisée et minimale, fermez cette boîte de dialogue et exécutez l'application Setup à partir du sous-répertoire **program** de l'**installation serveur**.

L'option **Installation de station de travail** installe sur l'ordinateur uniquement les fichiers utilisateur ; l'accès à l'installation serveur est indispensable pour l'exécution d'OpenOffice.org.

L'option **Installation locale** installe une copie complète d'OpenOffice.org sur l'ordinateur. Il n'est pas nécessaire d'accéder à l'installation serveur pour exécuter OpenOffice.org.

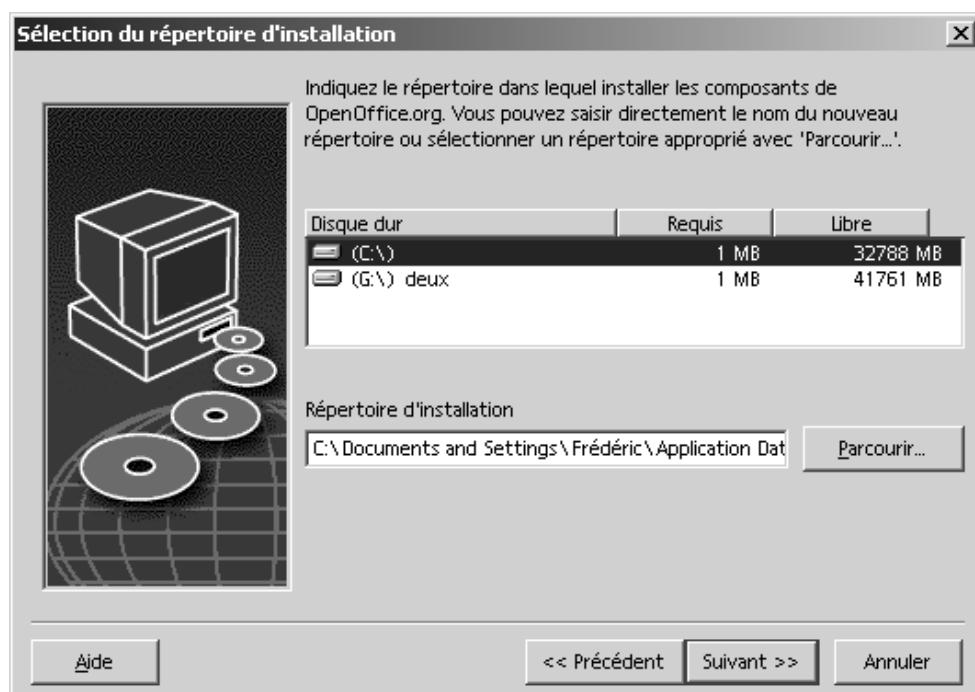


- Sélectionnez **Installation de station de travail** comme type d'installation. Vous devez effectuer une installation de station de travail distincte pour chaque utilisateur de l'ordinateur.
- Cliquez sur **Suivant** pour poursuivre l'installation.

Répertoire d'installation

La boîte de dialogue suivante vous permet de sélectionner le répertoire d'installation. Elle n'apparaît pas si vous procédez à une mise à jour de l'installation.

Les unités de disque disponibles sur votre système sont affichées dans cette boîte de dialogue. L'espace disque requis indiqué ici n'est qu'une estimation et dépend des propriétés du média de stockage.



- Saisissez le chemin du **répertoire d'installation** dans la zone de texte ou cliquez sur **Parcourir** pour sélectionner le répertoire.

Si ce répertoire n'existe pas, vous devez le créer.

- Cliquez sur **Suivant** pour poursuivre l'installation.

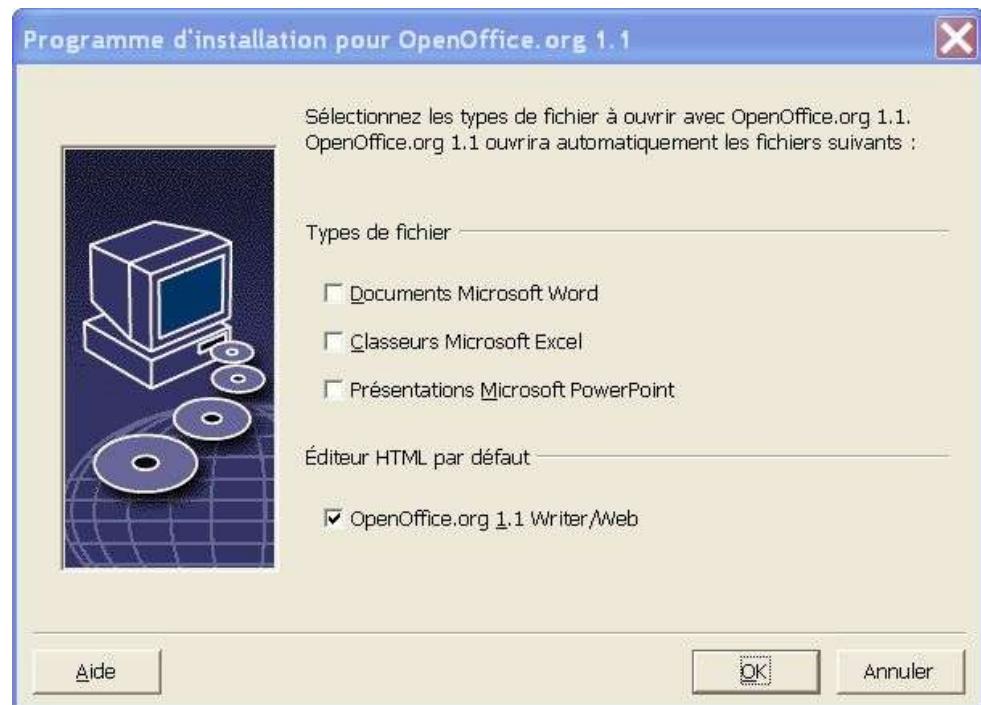
Vérification des options d'installation

Après la sélection des options d'installation, une boîte de dialogue s'affiche pour vous permettre de vérifier les paramètres.

- Cliquez sur **Installer** pour poursuivre l'installation ou sur **Précédent** pour modifier les paramètres sélectionnés.

Assignation des types de fichier

Utilisez cette boîte de dialogue pour assigner à OpenOffice.org des types de fichier Microsoft et des documents HTML. Ces informations sont enregistrées dans le système d'exploitation.



- Sélectionnez les **types de fichier** qu'OpenOffice.org doit ouvrir.
- Si vous souhaitez utiliser OpenOffice.org Writer comme éditeur par défaut pour les fichiers HTML, cochez la case au-dessous de **Éditeur HTML par défaut**.
- Cliquez sur **OK**.



Pour modifier les assignations de types de fichier d'une installation existante, lancez l'application Setup et sélectionnez l'option **Modifier** (voir Modification, page 48).

Environnement d'exécution Java™

Si l'environnement d'exécution Java™ (JRE) n'a pas été installé au cours de l'installation serveur, vous devez le faire à ce stade. Pour installer le JRE, suivez les instructions de la page 21.

Démarrage du processus de copie

La progression de l'installation est indiquée sous forme de pourcentage avec une estimation du temps restant.

Installation : fin

Lorsque l'installation est terminée, une dernière boîte de dialogue s'affiche.

- Cliquez sur **Terminer** pour quitter l'installation.



Selon le gestionnaire de fenêtres utilisé, vous devez vous déconnecter et redémarrer le gestionnaire de fenêtres.

Démarrage d'OpenOffice.org



Accédez au répertoire **{installpath}/program** ou au répertoire local d'OpenOffice.org dans votre répertoire principal et saisissez :

`./soffice`

Vous pouvez aussi ajouter le répertoire **{installpath}/program** dans le chemin des programmes ; il vous suffit ensuite de saisir "soffice" à partir de n'importe quel répertoire pour lancer OpenOffice.org.

Pour paramétrier l'imprimante par défaut pour OpenOffice.org, activez le programme de gestion des imprimantes **spadmin** (reportez-vous à l'appendice de cette documentation).

Dans l'environnement d'exploitation **Solaris**, déconnectez-vous une fois l'installation terminée et reconnectez-vous afin d'actualiser l'intégration CDE.



Les icônes permettant de lancer les applications OpenOffice.org sont ajoutées dans le menu **Démarrer de Windows** sous **Programmes/OpenOffice.org 1.1**.

Au cours de l'installation, un lien renvoyant au démarrage rapide d'OpenOffice.org est créé dans le dossier Démarrage du menu Démarrer de **Windows**. L'icône de démarrage rapide est ajoutée dans la barre des tâches au démarrage de Windows.

Un clic avec le bouton droit de la souris sur l'icône de démarrage rapide permet de sélectionner l'application OpenOffice.org à activer.

Pour plus d'informations sur le démarrage rapide, reportez-vous à l'aide en ligne.

Vous trouverez des informations sur le démarrage d'OpenOffice.org avec des paramètres de ligne de commande dans l'appendice de cette documentation.

Appendice

Cet appendice contient des instructions relatives à l'installation de l'environnement d'exécution Java (JRE), ainsi que des informations sur le programme de gestion des imprimantes dans l'environnement d'exploitation Solaris™ et sous Linux. Vous y trouverez en outre des indications concernant la modification, la réparation et la désinstallation d'OpenOffice.org™.

Installation de l'environnement d'exécution Java™ sous Windows

Au cours de l'installation d'OpenOffice.org, vous devez installer l'environnement d'exécution Java (JRE). Ces fichiers sont nécessaires pour exécuter les applets JAVA et certains composants d'OpenOffice.org, comme l'AutoPilote d'état.

Pour installer l'environnement d'exécution Java, choisissez l'option correspondante, puis suivez les instructions.

Si vous optez pour une installation ultérieure de l'environnement d'exécution Java, vous devrez exécuter le programme **jvmsetup** placé dans le répertoire **program** de l'installation d'OpenOffice.org.



Pour utiliser l'environnement d'exécution Java (JRE) sous Linux, reportez-vous au fichier **readme**, car il contient les informations les plus récentes à ce propos.

Paramétrage des imprimantes, fax et polices pour les plates-formes UNIX®

Sur les plates-formes UNIX®, le programme de gestion des imprimantes **spadmin** vous aide à paramétrier les imprimantes, fax et polices pour OpenOffice.org.

Activation du programme de gestion des imprimantes **spadmin** :

- Accédez au répertoire **{installpath}/program**.
- Saisissez : **./spadmin**

Après le démarrage, vous obtenez la fenêtre du programme de gestion des imprimantes **spadmin**.

Dans le cas d'une installation serveur, l'**administrateur système** doit d'abord se connecter au système en tant qu'utilisateur **root**, puis lancer le programme de gestion des imprimantes **spadmin**. Il devra ensuite créer un fichier de configuration général des imprimantes nommé **{installpath}/share/psprint/psprint.conf** pour tous les utilisateurs. Les modifications effectuées sont aussitôt actives pour tous les utilisateurs.

L'administrateur système peut également ajouter des polices pour tous les utilisateurs de l'installation en réseau. Toutefois, celles-ci ne sont disponibles qu'après le redémarrage d'OpenOffice.org.

Paramétrage des imprimantes

Sous UNIX, OpenOffice.org offre un support direct uniquement pour les imprimantes PostScript™. Les autres imprimantes doivent être paramétrées comme indiqué dans la section **Pilotes d'imprimantes dans OpenOffice.org**. Pour chaque file d'attente du système, Ope-

nOffice.org propose automatiquement une imprimante avec un pilote par défaut. Si nécessaire, vous pouvez ajouter des imprimantes.

Ajout d'une imprimante

1. Cliquez sur le bouton **Nouvelle imprimante**.
2. Sélectionnez l'option **Ajouter une imprimante** et cliquez sur **Suivant**.
3. Sélectionnez le pilote approprié pour votre imprimante. Si vous n'utilisez pas une imprimante PostScript ou si votre modèle n'est pas mentionné, utilisez le pilote **Generic Printer** (imprimante générique) ou suivez les étapes ci-dessous. Vous pouvez aussi ajouter de nouveaux pilotes à l'aide du bouton **Importer** ou supprimer les pilotes que vous n'utilisez plus à l'aide du bouton **Supprimer**. Cliquez sur **Suivant**.
4. Choisissez une ligne de commande permettant d'imprimer sur votre imprimante (par exemple, `lp -d my_queue`). Cliquez sur **Suivant**.
5. Attribuez un nom à l'imprimante et spécifiez si elle doit être utilisée comme imprimante par défaut. Cliquez sur **Créer**.
6. Pour imprimer une page de test, cliquez sur **Page de test**. Si la page de test ne s'imprime pas ou est mal imprimée, vérifiez tous les paramètres comme indiqué dans la section **Modification des paramètres de l'imprimante**.

Pilotes d'imprimante dans OpenOffice.org

Lors de l'installation d'une imprimante ne fonctionnant pas avec le langage PostScript, vous devez paramétrier votre système de sorte que le langage PostScript puisse être converti dans celui de l'imprimante. Nous recommandons l'utilisation d'un logiciel de conversion PostScript courant, tel que Ghostscript (<http://www.cs.wisc.edu/~ghost/>).

À défaut d'une imprimante PostScript, paramétrez l'imprimante **Generic Printer**. Vérifiez que les paramètres de marges sont corrects.

Si vous utilisez une imprimante PostScript™, vous devez installer un fichier de description adapté à l'imprimante (PostScript Printer Definition - PPD) pour pouvoir utiliser les fonctions spécifiques de l'imprimante, comme la sélection du bac d'alimentation, l'impression duplex et toutes les polices intégrées. Vous pouvez aussi utiliser le pilote d'imprimante générique car il contient les données les plus importantes et est adapté à la plupart des imprimantes. Dans ce cas, la sélection du bac d'alimentation est impossible et les marges doivent être paramétrées correctement.

Certains fichiers PPD sont installés par défaut. Si aucun fichier PPD adapté à votre imprimante n'est installé, vous en trouverez d'autres sur le site <http://www.adobe.com/products/printerdrivers/>. Vous pouvez aussi vous adresser au fabricant de votre imprimante.

Import de pilote lors de l'ajout d'une nouvelle imprimante

1. Cliquez sur **Importer** dans la boîte de dialogue de sélection des pilotes.

2. Cliquez sur **Parcourir** pour sélectionner le répertoire dans lequel vous avez décompressé les fichiers PPD.
3. Dans la zone de liste **Pilotes disponibles**, sélectionnez le pilote d'imprimante à installer.
4. Cliquez sur **OK**.

Suppression de pilote lors de l'ajout d'une nouvelle imprimante

- Sélectionnez le pilote d'imprimante.
- Cliquez sur **Supprimer**.

Veillez à ne pas supprimer le pilote d'imprimante générique et n'oubliez pas que les pilotes supprimés d'une installation serveur ne sont plus accessibles aux utilisateurs de cette même installation.

Si l'imprimante comporte davantage de polices intégrées que les polices PostScript habituelles, vous devez charger les fichiers AFM correspondant à ces polices supplémentaires. Copiez les fichiers AFM dans le répertoire **{installpath}/share/psprint/fontmetric** ou dans le répertoire **{installpath}/user/psprint/fontmetric**. Vous trouverez les fichiers AFM sur le site <ftp://ftp.adobe.com/pub/adobe/type/win/all/afmfiles/>, par exemple.

Modification des paramètres de l'imprimante

Dans le programme de gestion des imprimantes **spadmin**, sélectionnez l'imprimante dans la zone de liste **Imprimantes installées** et cliquez sur **Propriétés**. La boîte de dialogue **Propriétés** contient plusieurs onglets. Vous pouvez définir les paramètres appropriés en fonction du fichier PPD de l'imprimante sélectionnée.

- Sélectionnez la commande sur l'onglet **Commande**. Vous pouvez supprimer les commandes superflues à l'aide du bouton **Supprimer**.
- Sur l'onglet **Papier**, spécifiez le format du papier et le bac d'alimentation à utiliser par défaut pour l'imprimante concernée.
- Sur l'onglet **Périphérique**, activez les options spécifiques de l'imprimante. Si votre imprimante n'imprime qu'en noir et blanc, sélectionnez "Niveaux de gris" sous **Couleur**, sinon sélectionnez "Couleur". Si les résultats obtenus avec la conversion en niveaux de gris sont mauvais, sélectionnez "Couleur" sous **Couleur** et laissez l'imprimante ou l'émulateur PostScript effectuer la conversion. Sur cet onglet, vous pouvez également définir la précision de description des couleurs ainsi que le niveau PostScript.
- L'onglet **Substitution de police** vous permet de sélectionner une police d'imprimante pour chaque police installée sur votre ordinateur. Vous pouvez ainsi réduire la quantité de données à transmettre à l'imprimante. La substitution de police peut être activée ou désactivée séparément pour chaque imprimante.
- En cas d'utilisation du pilote d'imprimante générique, il est recommandé de définir correctement les marges des pages sur l'onglet **Paramètres supplémentaires** afin d'éviter toute coupure du document à l'impression. Vous pouvez également saisir une description

dans le champ **Commentaires**, qui sera ensuite affichée dans la boîte de dialogue **Impression**.

Certains de ces paramètres peuvent aussi être définis au moment de l'impression dans la boîte de dialogue **Impression** ou **Paramétrage de l'imprimante** dans OpenOffice.org via le bouton **Propriétés**.

Attribution d'un nouveau nom à l'imprimante ou suppression

- Sélectionnez une imprimante dans la zone de liste **Imprimantes installées**.
- Pour attribuer un nouveau nom à l'imprimante sélectionnée, cliquez sur **Renommer**. Saisissez le nom approprié dans la boîte de dialogue et cliquez sur **OK**. Choisissez un nom unique permettant d'identifier facilement l'imprimante et l'application. Attribuez les mêmes noms d'imprimante pour tous les utilisateurs ; lorsque les documents sont échangés, l'imprimante sélectionnée est conservée si elle existe sous le même nom pour le destinataire.
- Pour supprimer l'imprimante sélectionnée, cliquez sur **Supprimer**. Cette boîte de dialogue ne permet pas de supprimer l'imprimante par défaut ni une imprimante créée par l'administrateur système dans une installation serveur.
- Pour définir l'imprimante sélectionnée dans la zone de liste **Imprimantes installées** comme imprimante par défaut, double-cliquez sur son nom ou cliquez sur le bouton **Par défaut**.

Intégration d'un périphérique fax

Pour envoyer des fax avec OpenOffice.org, un logiciel de fax (par exemple Efax ou HylaFax) doit être installé sur l'ordinateur.

1. Cliquez sur **Nouvelle imprimante**. Vous obtenez la boîte de dialogue **Ajouter une imprimante**.
2. Sélectionnez **Connecter un périphérique fax**. Cliquez sur **Suivant**.
3. Choisissez d'utiliser le pilote par défaut ou un autre pilote d'imprimante. Cliquez sur **Suivant**.
4. Dans la boîte de dialogue suivante, saisissez une ligne de commande permettant d'accéder au fax. Dans la ligne de commande de chaque fax envoyé, "(TMP)" est remplacé par un fichier temporaire et "(PHONE)" par le numéro de fax du destinataire. Si "(TMP)" apparaît dans la ligne de commande, le code PostScript est transmis dans un fichier, sinon il est transmis comme entrée par défaut via un tube. Cliquez sur **Suivant**.
5. Attribuez un nom à la nouvelle imprimante fax et spécifiez si les numéros de téléphone (voir ci-dessous) sélectionnés dans le texte doivent être supprimés à l'impression. Cliquez sur **Créer**.

Vous pouvez maintenant envoyer des fax via l'imprimante qui vient d'être créée.

Dans le document, saisissez le numéro de fax sous forme de texte. Vous pouvez aussi insérer un champ reprenant le numéro de fax de la base de données active. Dans tous les cas, le numéro de fax doit commencer par les caractères @@# et se terminer par @@ (@@#1234567@@).

Si vous préférez ignorer ces caractères ainsi que le numéro de téléphone pour l'impression, activez l'option **Le numéro de fax ne sera pas indiqué** sous **Propriétés** sur l'onglet **Commande**. Si aucun numéro de téléphone n'est indiqué dans le document, vous obtenez, après l'impression, une boîte de dialogue permettant de le spécifier.

Dans OpenOffice.org, vous pouvez aussi activer un bouton pour l'envoi de fax vers un fax par défaut. Pour ce faire, cliquez avec le bouton droit de la souris sur la barre de fonctions, ouvrez le sous-menu **Boutons visibles** et cliquez sur le bouton **Envoyer par fax standard**. Définissez le fax qui sera utilisé ici sous **Outils - Options - Texte - Impression**.

Pensez à créer un travail d'impression spécifique pour chaque fax, sinon c'est le premier destinataire qui reçoit tous les fax. Dans la boîte de dialogue **Fichier - Lettre type**, sélectionnez l'option **Imprimante**, puis cochez la case **Travaux d'impression individuels**.

Connexion d'un convertisseur PostScript - PDF

Si un convertisseur PostScript - PDF (tel que Ghostscript ou Adobe Acrobat Distiller) est installé sur votre ordinateur, vous pouvez très facilement créer des documents PDF dans OpenOffice.org.

1. Cliquez sur **Nouvelle imprimante**. Vous obtenez la boîte de dialogue **Ajouter une imprimante**.
2. Sélectionnez **Connecter un convertisseur PDF**. Cliquez sur **Suivant**.

Indiquez si vous désirez utiliser le pilote par défaut, le pilote Acrobat Distiller ou un autre pilote. Le pilote "Generic Printer (T42 enabled)" fonctionne mieux avec les polices de type 42 plutôt que de type 3 et est donc approprié si vous utilisez un interpréteur PostScript. Cliquez sur **Suivant**.

3. Dans la boîte de dialogue suivante, saisissez une ligne de commande pour le convertisseur PostScript->PDF. Indiquez le répertoire d'enregistrement des fichiers PDF. Si aucun répertoire n'est indiqué, les fichiers sont enregistrés dans le répertoire principal de l'utilisateur. Dans la ligne de commande, "(TMP)" est remplacé par un fichier temporaire et "(OUTFILE)" par le fichier cible dont le nom découle du nom du document. Si "(TMP)" apparaît dans la ligne de commande, le code PostScript est transmis dans un fichier, sinon il est transmis comme entrée par défaut via un tube. Si Ghostscript ou Adobe Acrobat Distiller figurent dans le chemin de recherche, vous pouvez utiliser une des lignes de commande prédéfinies. Cliquez sur **Suivant**.
4. Attribuez un nom au nouveau convertisseur PDF. Cliquez sur **Créer**.

Vous pouvez maintenant créer des documents PDF via le convertisseur qui vient d'être créé.

Installation de polices

Au cours de l'utilisation d'OpenOffice.org, vous constaterez certainement que les polices proposées varient en fonction du type de document. En effet, toutes les polices ne peuvent pas être utilisées dans chaque cas.

- Dans un document texte, la liste de sélection des polices ne contient que les polices pouvant être imprimées car le programme considère que les documents créés sont destinés à être imprimés sur papier.
- Dans un document HTML ou dans une mise en page En ligne, seules les polices disponibles à l'écran sont proposées.
- En revanche, dans les classeurs ou les dessins, vous pouvez utiliser toutes les polices pouvant être imprimées ou représentées à l'écran.

OpenOffice.org tente d'afficher la représentation à l'écran de manière identique à l'impression (WYSIWYG). Les problèmes éventuels sont indiqués dans la partie inférieure de la boîte de dialogue **Format - Caractères**.

Ajout de polices

Vous pouvez ajouter des polices supplémentaires dans OpenOffice.org. Les polices que vous ajoutez sont disponibles uniquement pour OpenOffice.org ; elles peuvent être utilisées avec différents serveurs X sans avoir été installées sur ces serveurs. Pour que les polices soient accessibles à d'autres programmes, procédez comme d'habitude en les ajoutant à votre serveur X. OpenOffice.org peut afficher et imprimer les polices PostScript Type1, ainsi que les polices TrueType (y compris TrueType Collections).

Intégration de polices supplémentaires dans OpenOffice.org :

1. Lancez spadmin.
2. Cliquez sur **Polices**.
3. La boîte de dialogue liste toutes les polices ajoutées dans OpenOffice.org. Vous pouvez sélectionner et supprimer des polices en cliquant sur le bouton **Supprimer** ou en ajouter de nouvelles en cliquant sur **Ajouter**.
4. Cliquez sur **Ajouter**. La boîte de dialogue **Ajout de polices** s'affiche.
5. Spécifiez le répertoire à partir duquel ajouter les polices. Cliquez sur le bouton ... et sélectionnez le répertoire approprié dans la boîte de dialogue de sélection du chemin, ou saisissez directement le nom du répertoire.
6. La liste des polices contenues dans ce répertoire s'affiche. Sélectionnez les polices à ajouter. Pour ajouter toutes les polices, cliquez sur **Sélectionner tout**.
7. La case à cocher **Créer uniquement des soft links** permet de spécifier si les polices doivent être copiées dans le répertoire OpenOffice.org ou si seuls des liens symboliques doivent y être créés. Si les polices à ajouter se trouvent sur un média qui n'est pas disponible en permanence (par exemple, un CD-ROM), copiez-les.
8. Cliquez sur **OK** pour ajouter les polices.

Dans le cas d'une installation serveur, les polices doivent, si possible, être installées à ce niveau. Si l'utilisateur n'a pas les droits d'écriture requis, elles sont installées dans l'installation de station de travail correspondante ; dans ce cas, seul l'utilisateur qui les a installées y a accès.

Suppression de polices

Pour supprimer des polices, procédez de la façon suivante :

1. Lancez spadmin.
2. Cliquez sur **Polices**.
3. Toutes les polices ajoutées dans OpenOffice.org sont répertoriées dans la boîte de dialogue qui s'affiche. Sélectionnez les polices à supprimer et cliquez sur **Supprimer**.

Vous ne pouvez supprimer que les polices qui ont été ajoutées dans OpenOffice.org.

Attribution de nouveaux noms aux polices

Il est possible de renommer les polices ajoutées dans OpenOffice.org. Cette fonction est particulièrement utile pour les polices comportant plusieurs noms localisés.

1. Lancez spadmin.
2. Cliquez sur **Polices**.
3. Sélectionnez les polices à renommer et cliquez sur **Renommer**.
4. Saisissez un nouveau nom dans la boîte de dialogue qui apparaît. Si la police comporte plusieurs noms, ceux-ci sont indiqués à titre de suggestion dans la zone combinée de saisie du nouveau nom.
5. Cliquez sur **OK**.

Si vous sélectionnez plusieurs polices à renommer, une boîte de dialogue s'affiche pour chaque police sélectionnée.

Si vous avez sélectionné TrueType Collection (TTC), une boîte de dialogue apparaît pour chacune des polices correspondantes.

Installation d'un patch dans l'environnement d'exploitation SolarisTM

Avant de procéder à l'installation d'OpenOffice.org dans l'environnement d'exploitation Solaris, vous devez installer les patches système indiqués dans le chapitre "Configuration système requise", page 8.

Pour installer un patch Solaris, suivez les étapes de l'exemple ci-après pour le patch #106327-08. Ce patch est au format zip et a été téléchargé de l'adresse <http://sunsolve.sun.com>.

1. Connectez-vous en tant qu'utilisateur root :

`su -`

- Créez un répertoire temporaire destiné à la décompression du fichier patch comprimé (par ex. /tmp/patches).

```
mkdir /tmp/patches
```

- Commencez par copier le fichier patch comprimé dans ce répertoire puis décompressez-le à ce même emplacement :

```
unzip 106327-08.zip
```

- Installez le patch en utilisant la commande **patchadd** :

```
patchadd 106327-08
```

- Une fois le patch correctement installé, vous pouvez supprimer le répertoire temporaire :

```
rm -rf /tmp/patches
```



Pour afficher la liste des patches déjà installés sur le système, utilisez la commande **sho-wrev -p** ou **patchadd -p**. Pour désinstaller un patch, utilisez la commande **patchrm**.

Modification d'une installation OpenOffice.org existante

Pour modifier, réparer ou supprimer une installation existante de OpenOffice.org, exécutez l'application Setup qui se trouve dans votre répertoire local.



Modification

L'option **Modification** permet d'ajouter ou de supprimer des composants d'une installation existante, ou encore de modifier les affectations de types de fichier. Pour plus d'informations sur la sélection des composants, reportez-vous au chapitre "Sélection des composants", page 17.

Réparation

L'option **Réparation** permet de réparer une installation endommagée en restaurant les entrées de la base de registres du système et en réinstallant les fichiers du programme qui avaient été supprimés.

Suppression

L'option **Suppression** efface les entrées OpenOffice.org de la base de registres du système et supprime tous les fichiers du programme OpenOffice.org. Les fichiers que vous avez créés ou modifiés ne sont **pas** supprimés.



Vous pouvez aussi supprimer du répertoire d'installation d'OpenOffice.org tous les fichiers de configuration et les fichiers utilisateur.

Sous Windows, certains fichiers ne peuvent être supprimés qu'après le redémarrage du système.

Pour supprimer l'installation serveur sur le réseau, supprimez le dossier OpenOffice.org sur le serveur. Après cette opération, aucune installation de station de travail dépendant de cette installation serveur ne peut fonctionner.

Paramètres d'installation

Vous pouvez utiliser les paramètres suivants pour appeler le programme d'installation et exécuter des actions spécifiques :

- Utilisez **setup -net** ou **-n** pour lancer l'installation serveur.
- Utilisez **setup -D:chemin_destination** pour installer OpenOffice.org dans le répertoire spécifié dans le chemin.
- Utilisez **setup -F:nom_application** pour exécuter l'application spécifiée immédiatement après l'installation.

Démarrage d'OpenOffice.org avec des paramètres

Au démarrage d'OpenOffice.org via la ligne de commande, vous pouvez spécifier divers paramètres pour modifier son mode de fonctionnement. Il est préférable de réserver l'utilisation des paramètres de ligne de commande aux utilisateurs expérimentés.



L'utilisation des paramètres de ligne de commande n'est pas nécessaire pour le travail habituel. Certains de ces paramètres exigent une connaissance approfondie de la technologie à la base d'OpenOffice.org.

Vous trouverez plus d'informations à ce sujet à l'adresse <http://fr.openoffice.org>

Démarrage d'OpenOffice.org à partir de lignes de commande

1. Sous Windows, sélectionnez **Exécuter** dans le menu Démarrer ; sur les plates-formes UNIX®, ouvrez un Shell.
2. Sous Windows, saisissez le texte ci-après dans le champ **Ouvrir** et cliquez sur **OK**.
3. Sur les systèmes UNIX, saisissez la ligne de texte ci-après et appuyez sur **Entrée**.

{installation}\program\soffice.exe {paramètre}

Remplacez {installation} par le chemin d'installation d'OpenOffice.org (par exemple, C:\Program Files\Office, ou ~/office)

Selon le cas, remplacez {paramètre} par un ou plusieurs des paramètres de ligne de commande mentionnés ci-après.

Paramètres de ligne de commande

Paramètre	Signification
-help / -h / -?	Liste des paramètres en ligne de commande disponibles dans une boîte de dialogue. -help affiche un texte d'aide détaillé, -h un texte d'aide court.
-writer	Démarrage avec un document Writer vide.
-calc	Démarrage avec un document Calc vide.
-draw	Démarrage avec un document Draw vide.
-impress	Démarrage avec un document Impress vide.
-math	Démarrage avec un document Math vide.
-global	Démarrage avec un document maître vide.

Paramètre	Signification
-web	Démarrage avec un document HTML vide.
-minimized	Démarrage en mode simplifié. L'écran de démarrage ne s'affiche pas.
-invisible	Démarrage en mode invisible. Ni l'écran de démarrage ni la fenêtre initiale du programme ne sont visibles. Malgré tout, il est possible de contrôler OpenOffice.org et d'ouvrir les documents et les boîtes de dialogue via l'API.
	Lorsque OpenOffice.org a été démarré avec ce paramètre, il ne peut être fermé qu'avec le gestionnaire des tâches (Windows) ou via la commande kill (systèmes UNIX). Ce paramètre ne peut pas être utilisé en conjonction avec le paramètre -quickstart . Vous trouverez des informations supplémentaires dans le guide du développeur d'OpenOffice.org (Developer's Guide).
-norestore	Désactivation du redémarrage et de la récupération des fichiers après une panne du système.
-quickstart	Activation du démarrage rapide. L'écran de démarrage ne s'affiche pas.
-terminate_after_init	Enregistrement des services UNO, puis fermeture. L'écran de démarrage ne s'affiche pas. Vous trouverez des informations supplémentaires dans le guide du développeur d'OpenOffice.org (Developer's Guide).
-accept={chaîne UNO}	Notification de OpenOffice.org sur l'utilisation d'une chaîne d'acceptation UNO pour créer des threads d'acceptation UNO. Vous trouverez des informations supplémentaires dans le guide du développeur d'OpenOffice.org (Developer's Guide).
-userid={id utilisateur}	Spécification d'un répertoire utilisateur pour remplacer les données issues des fichiers soffice.ini , bootstrap.ini et sversion.ini . Vous trouverez des informations supplémentaires dans le guide du développeur d'OpenOffice.org (Developer's Guide).
-pt {nomdefichier1} {nomdefichier2} ...	Impression des fichiers {nomdefichier1} {nomdefichier2} ... sur l'imprimante par défaut, puis fermeture. L'écran de démarrage ne s'affiche pas. Si le nom de fichier contient des espaces, il doit être mis entre guillemets droits (par exemple, "C:\Mon fichier.sxw").

Paramètre	Signification
-pt {nomimprimante1} {nomdefichier1} {nomdefichier2} ...	Impression des fichiers {nomdefichier1} {nomdefichier2} ... sur l'imprimante {nomimprimante}, puis fermeture. L'écran de démarrage ne s'affiche pas.
	Si le nom de fichier contient des espaces, il doit être mis entre guillemets droits (par exemple, "C:\Mon fichier.sxw").
-o {nomdefichier}	Ouverture de {nomdefichier} pour édition, même s'il s'agit d'un modèle.
-view {nomdefichier}	Création d'une copie temporaire de {nomdefichier} et ouverture en lecture seule.
-n {nomdefichier}	Création d'un nouveau document sur la base de {nomdefichier} comme modèle.
-nologo	Désactivation de l'écran de démarrage.
-display {display}	Affectation de la valeur {display} à la variable d'environnement DISPLAY sur les plates-formes UNIX. Ce paramètre n'est supporté que par le script de démarrage de OpenOffice.org sur plate-forme UNIX.
-headless	Démarrage en mode "headless" (sans tête), qui permet d'utiliser l'application sans interface utilisateur. Ce mode spécial peut être utilisé lorsque l'application est contrôlée par des clients externes via l'API.

Enregistrement d'OpenOffice.org

La boîte de dialogue d'enregistrement s'affiche au deuxième démarrage d'OpenOffice.org. Vous pouvez également accéder à cette boîte de dialogue via **Aide - Enregistrement**

Pour modifier la boîte de dialogue d'enregistrement, éditez le fichier **common.xml** dans le répertoire **{installpath}/share/config/registry/instance/org/OpenOffice.org/Office/**. Pour désactiver en permanence la boîte de dialogue d'enregistrement, spécifiez la valeur **0** pour **RequestDialog** sous **Registration**. Pour désactiver en permanence la commande de menu Enregistrement, spécifiez la valeur **false** pour **ShowMenuItem** sous **Registration**.

```
<node oor:name="Help">
  <node oor:name="Registration">
    <prop oor:name="RequestDialog" oor:type="xs:int">
      <value>0</value>
    </prop>
    <prop oor:name="ShowMenuItem" oor:type="xs:boolean">
      <value>false</value>
    </prop>
  </node>
</node>
```

Extension d'OpenOffice.org

Des outils contenus dans le kit de développement de logiciel d'OpenOffice.org permettent d'ajouter des extensions aux fonctionnalités d'OpenOffice.org. Il s'agit de bibliothèques partagées, de fichiers de classes Java, de scripts OpenOffice.org Basic ou de nouvelles descriptions de type d'interface.



Pour plus d'informations sur la création d'extensions pour OpenOffice.org, reportez-vous au **guide du développeur** d'OpenOffice.org (Developer's Guide).

Le gestionnaire de paquetage **pkgchk** est installé et enregistré par l'application Setup d'OpenOffice.org. Il est placé dans le répertoire **{office_install}/program**. La syntaxe d'utilisation du gestionnaire de paquetage est la suivante :

```
pkgchk <commutateurs> [paquetage1 paquetage2 ...]
```

Commutateurs :

-s ou --shared	Vérification/installation des composants partagés
-r ou --renewal	Réinstallation de tous les paquetages (en cas d'erreurs de cache)
-v ou --verbose	Impression d'une trace détaillée des appels système au cours de l'installation
-f ou --force	Force la réécriture d'un paquetage existant lors d'une copie par ligne de commande
-l <fichier> ou --log <fichier>	Écriture d'un journal personnalisé <fichier>
--strict_error	Arrêt de l'installation du paquetage dès qu'une erreur se produit
--supersede_basic_libs	Remplacement des entrées de bibliothèques de base du même nom
-h ou --help	Impression d'un texte d'aide court

Si vous exécutez la commande **pkgchk** sans autre paramètre, seul le répertoire **{office_install}/user/uno_packages** est scanné :

- Si le programme détecte un fichier de paquetage d'extension, l'extension est installée pour l'utilisateur concerné.
- Si, dans une extension installée, un fichier du paquetage est manquant, l'enregistrement de l'extension correspondante est annulé.
- Si, dans une extension installée, un fichier du paquetage est différent du fichier du paquetage d'origine, l'extension correspondante est mise à jour.

Si vous lancez la commande **pkgchk** suivie du nom de fichier du paquetage et du chemin, le fichier correspondant est copié et installé dans le répertoire **{office_install}/user/uno_packages**.

Si vous lancez la commande **pkgchk** suivie du commutateur **-shared** ou **-s**, puis du nom de fichier du paquetage et du chemin, le fichier correspondant est copié et installé dans le répertoire **{office_net_install}/share/uno_packages** pour tous les utilisateurs de l'installation en réseau (partagée).

Installation, mise à jour et suppression d'extensions dans une installation monoutilisateur

1. Fermez toutes les fenêtres OpenOffice.org.

Vérifiez que vous avez quitté toutes les applications OpenOffice.org, **y compris le démarrage rapide**.

2. Effectuez l'une des opérations suivantes :

- Pour **installer** un paquetage d'extension, copiez le paquetage dans **{office_install}/user/uno_packages**.
- Pour **mettre à jour** un paquetage d'extension, copiez le paquetage mis à jour dans **{office_install}/user/uno_packages**, puis vérifiez que le paquetage existant a été écrasé.
- Pour **supprimer** un paquetage d'extension, supprimez le paquetage du répertoire **{office_install}/user/uno_packages**.

3. Exécutez la commande **pkgchk** dans le répertoire **{office_install}/program**.

Le gestionnaire de paquetage scanne le répertoire **{office_install}/user/uno_packages** pour rechercher les paquetages d'extensions et exécute toutes les étapes d'installation ou de désinstallation nécessaires. Après l'installation du paquetage, **ne supprimez pas** le fichier du paquetage du répertoire. La liste de toutes les actions exécutées par le gestionnaire de paquetage se trouve dans **{office_install}/user/uno_packages/cache/log.txt**.

Installation, mise à jour et suppression d'extensions dans une installation multiutilisateur

1. Fermez toutes les fenêtres OpenOffice.org.

Vérifiez que vous avez quitté toutes les applications OpenOffice.org, sur votre ordinateur et sur le serveur. **N'oubliez pas de fermer aussi le démarrage rapide**.

2. Effectuez l'une des opérations suivantes (où **{office_net_install}/share** est le répertoire partagé de l'installation serveur) :

- Pour **installer** un paquetage d'extension, copiez le paquetage dans le répertoire **{office_net_install}/share/uno_packages**.
- Pour **mettre à jour** un paquetage d'extension, copiez le paquetage mis à jour dans le répertoire **{office_net_install}/share/uno_packages**, puis vérifiez que le paquetage existant a été écrasé.

- Pour **supprimer** un paquetage d'extension, supprimez le paquetage du répertoire **{office_net_install}/share/uno_packages**.
3. Exécutez la commande **pkgchk -shared** dans le répertoire **{office_net_install}/program**.

Le gestionnaire de paquetage scanne le répertoire pour rechercher les paquetages d'extensions et exécute toutes les étapes d'installation ou de désinstallation nécessaires. Après l'installation du paquetage, le fichier du paquetage du répertoire. La liste de toutes les actions exécutées par le gestionnaire de paquetage se trouve dans



Sommaire

À propos de ce manuel.....	5
Icônes utilisées dans ce manuel.....	5
Commentaires.....	6
Informations générales sur l'installation.....	6
Types d'installation.....	6
Configuration système requise	7
Généralités.....	7
Particularités concernant l'environnement d'exploitation Solaris™ (SPARC® Platform Edition) - environnement d'exploitation Solaris (SPARC).....	7
Particularités concernant Linux.....	7
Particularités concernant Windows.....	8
Contenu des paquetages d'installation téléchargés	9
Opérations préalables à l'installation à partir d'un jeu téléchargé.....	10
Mise à jour d'une installation existante	10
Mise à jour d'une installation multiutilisateur.....	10
Installation monoutilisateur	10
Conditions requises pour l'installation.....	11
Démarrage de l'installation.....	11
Installation à partir d'un jeu téléchargé sur plate-forme UNIX.....	11
Installation à partir d'un jeu téléchargé sous Windows.....	11

Déroulement de l'installation.....	13
Écran de bienvenue.....	13
Informations importantes.....	14
Contrat de licence.....	14
Données d'identité.....	14
Choix du type d'installation.....	16
Sélection des composants	16
Répertoire d'installation.....	17
Vérification des options d'installation.....	18
Assignation des types de fichier.....	18
Installation de l'environnement d'exécution Java™	20
Démarrage du processus de copie.....	21
Installation : fin.....	21
Démarrage d'OpenOffice.org	21
Installation multiutilisateur ou en réseau	22
Installation serveur	23
Conditions requises pour l'installation.....	23
Démarrage de l'installation.....	23
Déroulement de l'installation.....	24
Installation de station de travail	31
Conditions requises pour l'installation.....	31
Démarrage de l'installation.....	31
Déroulement de l'installation.....	33
Appendice.....	38
Installation de l'environnement d'exécution Java™ sous Windows.....	39
Paramétrage des imprimantes, fax et polices pour les plates-formes UNIX®.....	39
Paramétrage des imprimantes.....	39
Intégration d'un périphérique fax.....	42
Connexion d'un convertisseur PostScript - PDF.....	43
Installation de polices.....	44

Installation d'un patch dans l'environnement d'exploitation Solaris™.....	45
Modification d'une installation OpenOffice.org existante	46
Modification	47
Réparation.....	47
Suppression	47
Paramètres d'installation.....	47
Démarrage d'OpenOffice.org avec des paramètres.....	48
Démarrage d'OpenOffice.org à partir de lignes de commande.....	48
Paramètres de ligne de commande.....	48
Enregistrement d'OpenOffice.org	50
Extension d'OpenOffice.org	51
Installation, mise à jour et suppression d'extensions dans une installation monoutilisateur.	
52	
Installation, mise à jour et suppression d'extensions dans une installation multiutilisateur .	
52	

Présentation d'OpenOffice.org

La suite

(OOo) est une suite bureautique comprenant :

- * un traitement de texte,
- * un tableur (comme Excel de Microsoft),
- * un logiciel de présentation assistée par ordinateur (PréAO) (comme Powerpoint de Microsoft).
- * un logiciel de dessin vectoriel.

OOo est un produit libre téléchargeable sur le site officiel « ». Vous pouvez donc l'utiliser et le copier librement pour vos élèves ou vos collègues... La version 1.1 d'OpenOffice.org n'a plus grand chose à envier à Microsoft Office, elle propose même des fonctions inédites, comme par exemple l'exportation au format PDF bien pratique pour publier sur un intranet.

Objectif de ce document

L'objectif de ce document est double :

* revoir les fonctions de base du traitement de texte, en insistant sur des points souvent mal maîtrisés par les autodidactes,

* vous faire découvrir et utiliser ses fonctions avancées, en particulier :

- l'utilisation des styles,
- le navigateur et son mode plan,
- la numérotation automatique des chapitres,
- l'insertion de sommaire automatique, note de bas de page, numérotation de pages,
- l'insertion et la mise en page d'images et tableaux,
- la création de schémas simples.

Bref, tout ce que doit savoir un utilisateur régulier du traitement de texte pour être efficace dans la rédaction d'un mémoire professionnel ou l'élaboration de documents pédagogiques.

Feuilleter ce document pour en avoir un aperçu !

Mode d'emploi du document

Ce document vous propose un parcours guidé :

❖ Les paragraphes encadrés d'un trait simple ombré avec l'icône
❖ présentent la démarche et les objectifs des activités proposées juste après.

❖ Les paragraphes encadrés doubles avec l'icône ❖ contiennent des informations générales ou présentent des concepts.

Les paragraphes avec un trait vertical double à gauche décrivent les tâches à réaliser.

L'encadré de droite détaille la procédure pas à pas pour les utilisateurs les moins expérimentés ou les manipulations délicates. L'icône en début de ligne précise la nature des activités demandées:

- utiliser la souris,
- utiliser le clavier,
- l'ordinateur réalise l'action,
- il faut observer,
- il faut répondre par écrit sur le document.

Condition de réutilisation de ce document

Le contenu de ce document est soumis à la licence "

" dont le contenu peut être consulté à l'adresse : « _____ ». Cela signifie que vous êtes libre de le reproduire, le recopier, le réutiliser, le modifier et le distribuer à condition de lui etc...

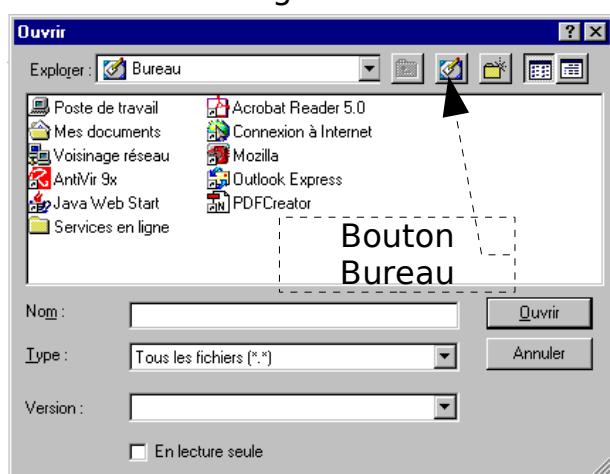
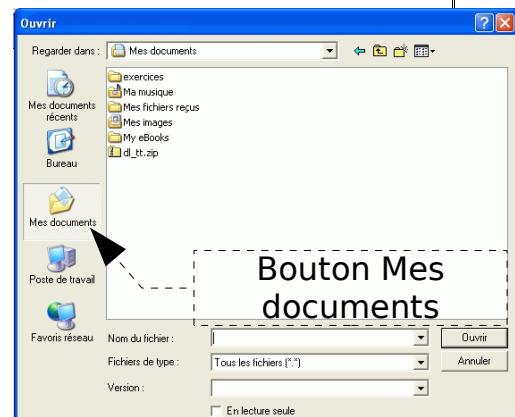
❖ Dans la suite de ce T.P. nous allons utiliser différents fichiers préparés. Il faut donc récupérer un fichier archive « » sur Internet et le décompresser dans votre dossier « ».



Le dossier « » :

Le dossier « » est un dossier dont l'emplacement réel peut varier suivant la configuration de votre ordinateur, particulièrement dans un environnement réseau.

Windows XP propose dans ses boîtes de dialogues « » ou « », un accès rapide au dossier « » comme dans la figure ci-contre à droite.



contre à gauche.

Sous Windows 98, si vous ne retrouvez pas le dossier « », dans une boîte de dialogue « » ou « », cliquer sur le bouton Bureau de la boîte de dialogue, le dossier « » apparaîtra alors dans la liste comme dans la figure ci-

Récupérer le fichier archive

Télécharger le fichier

« _____ » qui se trouve sur Internet à l'adresse suivante :

« _____ »

Enregistrer ce fichier dans votre dossier :

« _____ ».

Refermer toutes les fenêtres après le téléchargement.

Ouvrir votre navigateur web, Internet Explorer par exemple ou Mozilla.

Saisir à l'adresse suivante :

« _____ »

Repérer le lien hypertexte «

».

Cliquer droit sur le lien et sélectionner la commande

« _____ » (Mozilla)

ou « _____ »

(Internet Explorer).

ouvre une boîte de dialogue :

« _____ ».

Sélectionner l'emplacement « _____ ». Valider.

L'ordinateur télécharge le fichier

« _____ » depuis Internet vers votre ordinateur.

Referme la fenêtre de téléchargement et la fenêtre du navigateur.

Extraire les fichiers de l'archive



Les fichiers archives Zip

Un fichier archive est un fichier contenant, sous une forme compressée un ensemble de fichiers et/ou dossiers. Le format que l'on rencontre le plus souvent sur Internet est le format Zip. Le système d'exploitation Windows XP prend en charge ce type de fichier archive. Sous Windows 95 ou 98, il est nécessaire d'installer un logiciel capable de gérer ces fichiers comme par exemple « - » logiciel libre que vous pouvez télécharger sur Internet à l'adresse suivante :

«

».

Décompresser le fichier
« » dans votre
dossier « ».
Vérifier la présence d'un nouveau
dossier « » dans votre
dossier « »

Ouvrir une fenêtre du poste de travail.
Ouvrir le dossier « ».
Vérifier la présence du fichier archive « » téléchargé précédemment.

- 1. Avec « » :
 - Cliquer droit sur le fichier « ».
 - Sélectionner la commande « ».
 - Windows ouvre une fenêtre « ».
 - Suivre les instructions : lorsque Windows vous propose un emplacement : cliquer sur le bouton « » et sélectionner le dossier « ».
 - « » l'assistant sans afficher les fichiers extraits.
- Avec « » :
 - Cliquer droit sur le fichier « ».
 - Sélectionner la commande « ».

Afficher les extensions de fichiers

☞ Sous Windows XP ou 98, le nom des fichiers comporte deux parties :

- * la partie principale
- * une extension, séparée de la partie principale par un point.

Par exemple « » est un fichier dont la partie principale du nom est « » et son extension « ».L'extension permet au système d'exploitation Windows XP ou 98 de reconnaître le type de fichier, par exemple :

- * correspond à un fichier exécutable, une application,
- * correspond à un fichier d'aide,
- * correspond à un fichier créé par un traitement de texte comme Microsoft Word.

Les fichiers créés par l'application « » d'OpenOffice ont une extension « ».Grâce à cette extension, Windows peut associer une application et donc un icône au fichier document. Par défaut, Windows masque

les extensions de fichiers. Cela peut être gênant pour les utilisateurs expérimentés ... que vous allez devenir.

❖ Nous allons donc vérifier la configuration de Windows et si nécessaire corriger cette configuration afin d'afficher les extensions de fichiers.

Ouvrir le dossier « » dans « ». Vérifier que Windows ne masque pas les extensions de fichiers.

Si nécessaire, à l'aide de la commande :

sous Windows XP «

»

sous Windows 98 «

»,

à l'onglet « », décocher la ligne «

»

Ouvrir le dossier « » dans « ».

Voyez-vous les extensions des fichiers ? Sinon :

Sélectionner la commande :

sous Windows XP, «

»,

sous Windows 98, «

»

Sélectionner l'onglet « ».

Décocher la ligne «

».

Valider en cliquant sur le bouton

« ».

N-B : Attention, ne pas modifier l'extension d'un fichier par erreur, le format du fichier ne serait plus reconnu correctement par le système d'exploitation.

❖ Dans cette partie « », nous allons balayer rapidement les fonctions de base du traitement de texte en insistant sur des points méconnus.

Démarrer l'application « Texte »

Démarrer l'application « » d'OpenOffice.

»

Cliquer sur le bouton « de la barre de tâches.

Sélectionner la commande
« »

N.B. :

Si l'application « » ouvre une boîte de dialogue « », sélectionner l'option « » et cliquer sur le bouton « ».

L'application « » ouvre un document vierge « ».

Noter le titre de la fenêtre d'application et l'apparition d'un nouveau bouton dans la barre de tâches en bas de l'écran.

Saisir quelques caractères dans le document.

Ouvrir et enregistrer des documents

❖ Maîtriser le chargement et la sauvegarde de documents à l'intérieur d'une application est absolument nécessaire. De plus, maîtriser le format d'enregistrement des documents est aussi très utile notamment si vous souhaitez échanger des documents avec d'autres utilisateurs. Tels sont les objectifs de ce chapitre ...

Ouvrir le fichier « » qui se trouve dans le dossier « » de « »

Sélectionner la commande « » de la barre de menus.

OOo ouvre une boîte de dialogue « ».

Par défaut, l'application affiche le dernier dossier ouvert.

Retrouver le dossier « ».

Noter la présence du dossier « ».

Double-cliquer sur l'icône du dossier « »

OOo affiche le contenu du dossier « ».

Repérer le fichier « ».

Double-cliquer sur l'icône de ce fichier.

OOo ouvre le document.

Observer l'apparition d'un nouveau bouton dans la barre de tâches.

N.B. : Chaque document de OpenOffice ouvre une nouvelle fenêtre d'application : on peut donc basculer d'un document à un autre en cliquant sur le bouton correspondant de la barre de tâches ou en utilisant le raccourci clavier **Alt** . (Il faut appuyer sur la touche **Alt**, la maintenir enfoncée et appuyer plusieurs fois sur la touche [à gauche de la touche A], pour sélectionner l'application.)



Savoir distinguer les commandes « » et « » :

La commande « » permet d'enregistrer le document avec un nouveau nom et/ou dans un nouvel emplacement.

La commande « » fait un enregistrement avec le nom par défaut dans l'emplacement par défaut. Retenir son raccourci clavier **Ctrl** .

N-B : Il est conseillé d'enregistrer votre document tous les $\frac{1}{4}$ d'heure environ. Vous pouvez aussi activer l'enregistrement automatique (Menu

« » : rubrique « C »).

Sélectionner le document « ».

Placer le curseur au début du document.

Appuyer sur la touche pour créer un nouveau paragraphe.

Remonter le curseur sur la première ligne. Saisir votre nom et votre prénom.

Enregistrer le fichier ainsi modifié :

dans le dossier « » (pas le dossier

« »)
avec « » comme nom de fichier « » représentant vos initiales.

Sélectionner la commande « ».

OOo affiche une boîte de dialogue « ».

L'application a ouvert le dernier emplacement utilisé, en l'occurrence le dossier « ».

Repérer le bouton « » ou « ».

Cliquer sur ce bouton.

L'application affiche le contenu du dossier « ».

Saisir le nom « », « » représentant vos initiales. Valider.



Gestion des extensions de fichier par OOo :

OOo a ajouté l'extension « » au nom « ». OOo, comme la plupart des applications, gère lui-même les extensions de fichiers : dans la boîte de dialogue « », ne pas décocher la case « ».



La boîte de dialogue « » permet aussi de sélectionner un autre format d'enregistrement, notamment :

* le format RTF (Rich Text Format) qui est un format d'échange reconnu par tous les traitements de texte, format certifié sans virus car sans macros-commandes (à privilégier dans les échanges par courrier électronique),

* le format Word 95 /2000 pour ceux qui utilisent Microsoft Word à un endroit et OpenOffice ailleurs ...

Il est donc important de maîtriser le fonctionnement de cette boîte de dialogue.

Enregistrer le fichier « format RTF « dans le dossier « ».

» au
»

Sélectionner la commande

« ».

Repérer la liste déroulante sous le nom du fichier :



Cliquer sur la flèche bas de la liste déroulante.

Sélectionner le format « ».

Cliquer sur le bouton « ».

Quitter OOo en ignorant les modifications du document « ».

Sélectionner la commande « ».

OpenOffice affiche un message d'avertissement. Lire le message.

Cliquer sur le bouton « », nous avons déjà sauvegardé le fichier au format OpenOffice.

Fermer la fenêtre du document « » en ignorant les modifications .

Quitter l'application OpenOffice.

Ouvrir le dossier « ».

Vérifier la présence des deux fichiers « » et « » dans ce dossier.

Exercice :

*Ouvrir le document le document « **tp2.sxw** » qui se trouve dans le dossier « **exercices** ». Enregistrer ce document sous le nom « **xxtp2** » dans le dossier « **Mes Documents** » : au format OpenOffice **puis** au format Word 97/2000.*

Configurer l'interface

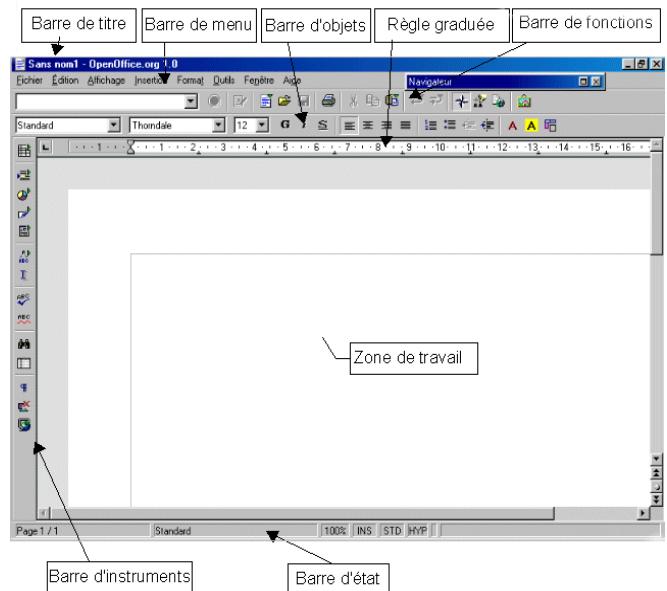
Comme la plupart des logiciels, OpenOffice propose de personnaliser son interface utilisateur et de modifier certains paramètres par défaut. Il est utile et parfois nécessaire, de savoir rétablir ou adapter la configuration du logiciel.

L'interface d'OpenOffice possède 4 barres d'outils
la barre de fonctions, sous la barre de menus
la barre d'objets, sous la barre de fonctions
la barre d'instruments à gauche de l'écran
la barre d'hyperliens non affichée par défaut.

Ouvrir votre fichier « ».

A l'aide de la commande

« »
s'exercer à masquer et à rétablir les différentes barres.



Sélectionner la commande

« »
Observer les coches devant le nom des barres d'outils.
Décocher l'item « ».
Rétablir l'affichage de la barre d'instruments.

Le clic droit

Sous Windows XP ou 98, les principales commandes peuvent être lancées à partir du menu contextuel : il suffit de cliquer droit sur l'objet, un menu spécifique, contextuel, surgit : il vous reste alors à saisir la commande désirée avec un clic gauche.

Masquer puis rétablir les différentes barres d'outils à l'aide des menus contextuels (clic droit dans les différentes barres)

Cliquer droit dans une zone inoccupée de la barre de fonctions.

OOo ouvre un menu contextuel

Décocher l'affichage de la commande « ».

Rétablissement la barre de fonctions à l'aide d'un clic droit sur la barre d'objets, par exemple.



Un fichier créé par un traitement de texte contient, en plus des caractères qui seront imprimés, des caractères de mise en forme, invisibles à l'impression mais nécessaires à la structuration du document, en particulier : les marques de paragraphes, les tabulations, les retours à la ligne, les sauts de pages, ...

A l'aide du bouton «

»  de la barre d'instruments, masquer puis rétablir l'affichage des caractères non imprimables.

Repérer le bouton «

»  dans la barre d'instruments verticale.

Masquer les caractères non imprimables.

Rétablissement l'affichage des caractères non imprimables.



Ce bouton  est un bouton à deux états : caractères non imprimables visibles ou bien cachés.

N.B. : En phase de saisie ou de modification du document, il est vraiment utile de voir ces caractères non imprimables afin de mieux appréhender la structure du document. En phase de finition, on les masque pour mieux apprécier la présentation du document.

A l'aide de la commande « **Mise en forme** », tester les options « **Format de page** », puis « **Retrait** ».

Sélectionner la commande « **Mise en forme** ». OOo ouvre une boîte de dialogue « **Format de page** ».

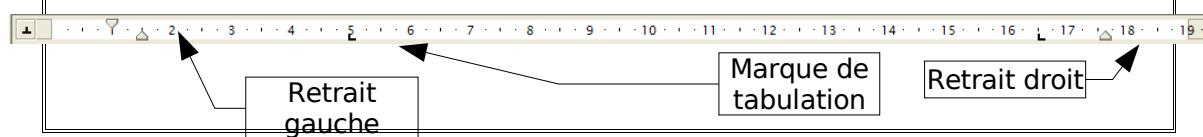
Sélectionner l'option « **Retrait** » Valider.

Ce mode d'affichage permet de vérifier et de modifier la mise en page du document.

Sélectionner à nouveau la commande « **Mise en forme** ».

Sélectionner l'option « **Retrait** » Valider.

La règle permet de régler les retraits, marques de tabulation, largeurs de colonnes des tableaux... Comme pour les marques de mise en forme, il est utile de la faire apparaître en **Retrait de la ligne**.



Masquer puis rétablir la règle à partir de la commande « **Réglage de la page** ».

Sélectionner la commande « **Réglage de la page** » de la barre de menus.

Observer la présence ou l'absence de coche devant les différents éléments du sous-menu « **Retrait** ».

Décocher la coche devant la commande « **Retrait** » en cliquant dessus.

Observer l'écran.

Rétablissement de l'affichage de la règle.

Observer la position des triangles de la règle d'un paragraphe à un autre.

Cliquer sur la première ligne du document.

Repérer la position des triangles de gauche de la règle.

Cliquer au milieu du paragraphe «

».

Que constatez-vous au niveau de la règle ?

Par glisser-déplacer, modifier le retrait droit du paragraphe « . ».

Par glisser-déplacer (drag and drop), déplacer le triangle de droite de la règle.

Constater la modification du retrait droit.



Lorsque la vérification automatique d'orthographe est activée, OOo souligne en rouge tous les mots mal orthographiés. Comme tout correcteur orthographique, sa fiabilité est toute relative.

Activer si nécessaire la vérification automatique d'orthographe à partir de la commande « . ».

Sélectionner la commande « . ».

Vérifier la présence de la coche devant l'item « . »

Cocher si nécessaire cet item.

Corriger l'orthographe du mot « . » à l'aide du correcteur orthographique (clic droit)

Le mot « . » doit être souligné en rouge, car mal orthographié

N-B : Si le mot n'est pas souligné en rouge, soit le correcteur orthographique a été mal installé, soit l'option « . » n'a pas été sélectionnée.

Cliquer droit sur ce mot.

OpenOffice propose une correction.
Sélectionner cette correction.

N-B : Dans la barre d'instruments, vous trouverez deux boutons concernant le correcteur orthographique :



qui correspond à la commande «
»



qui correspond à la commande «
»

Utiliser les mises en forme de paragraphes et de caractères standards

❖ Mettre en caractères gras, centrer un paragraphe, ... tout utilisateur de traitement de texte apprend rapidement à le faire... Mais savez-vous rétablir une mise en forme standard en un seul clic...

Centrer le paragraphe «
» à l'aide de la barre d'objets.

Placer le curseur sur la ligne «
»
Cliquer sur le bouton «
» de la barre d'objets

Justifier le paragraphe «
.»

Placer le curseur sur la première
ligne du paragraphe «
.».
Cliquer sur le bouton «
»

N.B. : Il n'est pas utile de sélectionner tout le paragraphe d'avant d'appliquer une mise en forme de paragraphe : il suffit de placer le curseur dans le paragraphe pour lui appliquer la mise en forme désirée. Par contre, pour appliquer la mise en forme à plusieurs paragraphes consécutifs, il faut les sélectionner ensemble puis leur appliquer la mise en forme.

Rétablissement la mise en forme du paragraphe « » à l'aide de la commande « » du menu contextuel.

Cliquer droit sur le paragraphe.« »
Sélectionner la commande « » (1ère ligne du menu contextuel).
rétablit la mise en forme du paragraphe.

Mettre « » en caractères gras italiques

Sélectionner l'expression « ».
Cliquer sur le bouton puis sur le bouton pour mettre l'expression en caractères gras et italiques.

N.B. : Pour sélectionner une zone de texte :

avec la souris : cliquer gauche au début de la zone à sélectionner puis maintenir appuyé le bouton de gauche et déplacer la souris

avec le clavier : placer le curseur au début de la zone, maintenir la touche appuyée et déplacer le curseur avec les touches Flèches gauche, droite.

Rétablissement la mise en forme « » au mot « »

Sélectionner le mot « ».
Cliquer droit sur le mot.
Sélectionner la commande « » (1ère ligne du menu contextuel).
OOo rétablit la mise en forme de caractères.

La barre d'objets contient les commandes de mise en forme les plus fréquemment utilisées. Pour obtenir la palette complète des mises en forme, sélectionner les commandes « » ou « » de la barre de menus ou du menu contextuel (clic droit).

 Un texte n'est pas une simple suite de caractères, c'est un document structuré contenant des paragraphes standards, des titres de différents niveaux, des paragraphes remarques, etc. La touche  insère un nouveau paragraphe, ce qui entraîne à fortiori un passage à la ligne. La combinaison de touches   permet de passer à la ligne sans changer de paragraphes. Si la mise en forme du paragraphe est la même avant et après le retour à la ligne, il faut saisir, non pas un caractère « Nouveau Paragraphe »  (touche ) mais un caractère « Retour à la ligne »  ( ).

Insérer plusieurs sauts de ligne devant l'expression «
»

Placer le curseur devant l'expression
«
»

Appuyer sur la combinaison de touches   pour insérer un passage à la ligne.

Observer le caractère non imprimable utilisé par le logiciel pour marquer le passage à la ligne.

Répéter le processus plusieurs fois.

Annuler les dernières modifications

   Annuler les dernières modifications  

La combinaison de touches   permet d'annuler les dernières frappes. C'est le raccourci de la commande «
».

S'il n'y avait qu'une seule combinaison de touches à retenir c'est   qu'il faudrait conserver !!!

Annuler les dernières modifications à l'aide du raccourci clavier  .

Appuyer successivement sur la combinaison de touches   pour retrouver le texte dans son état initial.

N.B. : La commande «
»   permet de restaurer ce qui a été précédemment annulé.

Appuyer sur   pour sauvegarder votre travail.

Fermer le fichier «
».

Ouvrir le fichier «
».

Déplacer des zones de texte

❖ L'ordre des paragraphes de ce document a été modifié. Nous allons rétablir l'ordre du document en nous fiant aux numéros de paragraphe. Nous allons utiliser différentes méthodes.

N.B. :

Pour déplacer correctement des zones de texte, il est indispensable d'afficher les marques de paragraphe (caractères non imprimables visibles, bouton  de la barre d'instruments



  : supprime la zone de texte sélectionnée et la copie dans une mémoire tampon qu'on appelle communément le presse-papier.

  : copie le contenu du presse-papier vers l'emplacement sélectionné.

  : recopie la zone de texte sélectionnée dans le presse-papier sans le supprimer de son emplacement de départ.

A l'aide des raccourcis claviers

  et

 , placer le paragraphe « »

avant le paragraphe « ».

Placer le curseur au début de la ligne

«

»

Maintenir appuyé la touche  et déplacer le curseur en fin de ligne à l'aide de la touche « ».

Etendre la sélection jusqu'au caractère paragraphe en fin de ligne  à l'aide de la touche flèche vers la gauche .

Appuyer sur la combinaison de touches   pour couper la sélection.

Placer le curseur devant la ligne « ».

Appuyer sur la combinaison de touche   pour coller la zone de texte coupée précédemment.



	Début de ligne		Fin de ligne
 	Début document	 	Fin Document

Mot à droite

Mot à gauche

A l'aide de la souris et des menus contextuels « » – « », placer l'expression « » en début de document.

Selectionner la zone de texte « » en prenant soin d'inclure dans la sélection la marque de paragraphe 1.

Cliquer droit dans la zone sélectionnée et sélectionner la commande « » du menu contextuel.

Cliquer en début de document, clic droit et sélectionner la commande « » du menu contextuel.

*Exercice : Déplacer en début de document la zone de texte allant de « **2. 1 Documents ...** » jusqu'à « **... d'une imprimante et d'un accès à internet depuis 1998.** ». Si nécessaire travailler avec le clavier ...*

Déplacer la zone de texte allant de « » ...à « ». devant le paragraphe « » par glisser-déplacer.

Selectionner la zone de texte allant de « » ...à « ».

Cliquer à l'intérieur de la zone sélectionnée, maintenir appuyé le bouton gauche de la souris.

Déplacer le pointeur de la souris avant le paragraphe « » en maintenant le bouton gauche appuyé.

Relâcher le bouton gauche de la souris ... Magique !

pour enregistrer vos modifications.

Copier – coller entre deux documents



Les commandes « » ou « » fonctionnent aussi d'un document à un autre, d'une application à une autre...

Ouvrir le document « ».
Copier le contenu du document « » au début du document « ».

Ouvrir le document « ».
Appuyer sur la combinaison de touches (raccourci clavier de la commande « » de la barre de menus). OOo sélectionne la totalité du document « ».
Appuyer sur la combinaison de touches pour copier dans le presse-papier la zone sélectionnée.
Activer la fenêtre « » à l'aide de la barre de tâches de Windows ou avec la combinaison de touches .

Appuyer sur la combinaison de touches pour placer le curseur en début de document.
Appuyer sur la combinaison de touches pour coller le contenu du presse-papier dans le document.

pour enregistrer les modifications apportées au document « ». Refermer la fenêtre du document « » sans enregistrer les modifications.

Mise en page

Tester une mise en page « » à l'aide de la commande « ».
Rétablir une mise en page « » avec des marges respectant le tableau ci-dessous :
Marge gauche : 3 cm
Marge droite : 1 cm
Marge haute : 2 cm
Marge basse : 2 cm

Sélectionner la commande « ».
Sélectionner l'onglet « ».
Sélectionner l'option Orientation « ». Valider.
Noter l'effet sur la mise en page.
Sélectionner la commande « ».
Sélectionner l'onglet « ».
Sélectionner l'option Orientation « ». Valider.
Avec la même commande « », fixer les marges suivant les valeurs ci-contre.

Propriétés du document

 Pour chaque document, OOo enregistre des propriétés le concernant : titre, auteur, ... qui permettent de le caractériser (méta-données) et vous aident à gérer et à identifier vos documents. Ces données peuvent être ensuite réutilisées à l'intérieur du document (cf notion de champs au chapitre 4.9).

A l'aide de la commande « **Propriétés** », saisir le titre « **Titre** » dans les propriétés du document.

Sélectionner la commande « **Format > Propriétés** ».
OOo affiche une boîte de dialogue « **Propriétés de document** ».
Sélection l'onglet « **Titre** ».
Saisir le titre « **Titre de la page** ».
Valider.

Noter la modification dans la barre de titre et le bouton de la barre de tâches.

Enregistrer vos modifications.

Imprimer

 En réseau, vous avez souvent accès à plusieurs imprimantes, l'imprimante par défaut n'est pas toujours celle que l'on croit ... Il faut donc se méfier du bouton « **Imprimer** »  de la barre d'outils qui utilise l'imprimante par défaut ou la dernière imprimante utilisée ...

Identifier les imprimantes disponibles sur votre ordinateur.

Noter-les

.....
.....
.....
.....
.....

Sélectionner la commande « **Fichier > Imprimer** ».
OOo affiche une boîte de dialogue « **Impression** ».
Repérer la liste déroulante des imprimantes disponibles.
Identifier les imprimantes disponibles sur votre ordinateur
Cliquer sur le bouton « **Annuler** » pour refermer sans imprimer.

 La commande « **Impression avant impression** » permet de constater des problèmes de mise en page avant impression : il faut toujours utiliser « l'aperçu avant impression » avant l'impression,

d'où son nom ...

Basculer en mode « » à l'aide de la commande « »
Tester les différents mises en page de l'aperçu.
Refermer l'aperçu.

Sélectionner la commande « » de la barre de menus.

Oo affiche une nouvelle fenêtre avec une nouvelle barre d'outils.

Identifier la fonction des différents boutons, en plaçant le pointeur de la souris au dessus des boutons pour faire apparaître la bulle d'aide.

Tester les boutons « », puis « ».

Cliquer sur le bouton « » pour refermer la fenêtre d'aperçu avant impression.

Enregistrer vos modifications.

Exercice :

*Ouvrir le fichier « **exo1.sxw** » dans le sous-dossier « **exercices** ». Exécuter le travail demandé. Refermer le fichier sans enregistrer les modifications en fin de travail.*

*Répéter les mêmes opérations pour les fichiers : « **exo2.sxw** », « **exo3.sxw** », « **exo4.sxw** », « **exo5.sxw** », « **exo6.sxw** », « **exo7.sxw** ».*

Un petit break et passons aux choses sérieuses ...

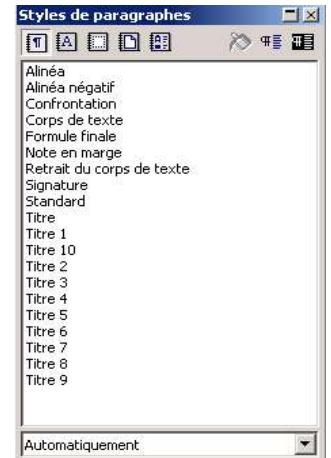
❖ Appliquer une mise en forme uniforme à un document long, insérer un sommaire automatique, pouvoir récupérer la mise en forme d'un document vers un autre, restructurer rapidement un document, obtenir une numérotation automatique des chapitres, ... tels sont les enjeux de l'utilisation des styles dans un traitement de texte : c'est simple, surtout dans OpenOffice et puissant ...

Utiliser les styles

Si nécessaire, charger votre document « ».

❖ Le styliste : permet d'appliquer, créer, éditer, ajouter et supprimer des styles de mise en forme au document. Vous pouvez l'afficher :

- à l'aide de la commande « »,
- à l'aide du bouton  de la barre de fonction,
- ou en appuyant sur la touche .



Activer le styliste
Afficher tous les styles disponibles.

Appuyer sur la touche .
OOo affiche une fenêtre de style appelée « S ».
Repérer en bas du « S » une liste déroulante.



Sélectionner l'option « ».

❖ OOo gère différents types de style : paragraphes, caractères, cadres, pages, numérotation.

Visualiser les styles des différents types.

Identifier les différents boutons en haut du styliste en pointant la souris au-dessus des boutons pour faire apparaître les bulles d'aide :

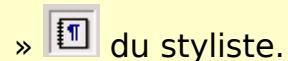


Cliquer sur ces différents boutons.
Noter les différents styles proposés par OpenOffice. Il y en a beaucoup ...

Appliquer le style de paragraphe « » au paragraphe :
« ».

Observer le changement de mise en forme.

Cliquer sur le bouton «



du styliste.
Le styliste affiche la liste des styles de paragraphes.

Placer le curseur sur la ligne « ».

Double-cliquer sur le style « » du styliste.

OOo attribue le style « » au paragraphe « ».

Observer le changement de mise en forme.

*Exercice : Appliquer le style « **Titre 1** » aux paragraphes suivants :*

2 Le fonds documentaire

3 Accès au fonds documentaire

4 La politique d'acquisition

*Appliquer le style « **Titre 2** » aux paragraphes suivants :*

2.2 Equipement technologique et informatique

3.1 Spatialisation

3.2 Signalisation

3.3 Techniques documentaires

Enregistrer vos modifications en fin d'exercice.

Modifier le style « » en respectant les consignes ci-dessous :

- police :« »
- style :« »
- taille :« »,
- couleur de police : « »
- soulignement : « »
- couleur de soulignement : « »
- alignement :
- retrait avant le texte (gauche) : « »
- écart avant de
- écart après de

Balayer tout le texte.

Conclusions :

Cliquer droit sur la ligne « » du styliste.

Sélectionner la commande « ».

Ouvre une boîte de dialogue « ».

Sélectionner l'onglet « ».

Modifier le style « » en respectant les consignes suivantes : Police : « » style : « » taille : « »,

N-B : Par défaut, OpenOffice affiche la taille des caractères en %. Vous pouvez basculer en pt, il suffit de saisir au clavier directement la valeur « »

Sélectionner l'onglet « ».

Sélectionner un soulignement

« » de couleur « » avec une police de couleur « » aussi.

Sélectionner l'onglet « ».

Sélectionner un alignement « ».

Sélectionner l'onglet « ».

Sélectionner un « » (retrait gauche) de « », un écart avant de « », un écart après « ».

Valider vos modifications.

Balayer tout le texte et observer les changements de mise en forme sur les titres de niveau 1.

Exercice :

*Modifier le style « **Titre 2** » en respectant les consignes suivantes :*

- police :« **Arial** »
- taille :« **12 pt** »

- style :« **gras** »
- soulignement :

« **simple** »
 - couleur de soulignement : « **à gauche** »
 alignement :« **à gauche** »
 - retrait avant le texte : « **0,21 cm** »
 avant de :« **0,21 cm** »
 - écart après de« **0,21 cm** »

« **bleue** ».....
 « **1,5 cm** »....- écart

*Modifier le style « **standard** » en respectant les consignes suivantes :*

- police :« **Times New roman** » - taille :
 « **10 pt** »,
 - alignement :« **justifié** » - interligne :
 « **double** »
 - retrait avant le texte : « **0 cm** »- écart
 avant de :« **0 cm** »
 - écart après de« **0 cm** »

Enregistrer vos modifications en fin d'exercice.

Utiliser le navigateur

☞ Le navigateur : permet d'accéder rapidement aux différents éléments du document : titre, images, tableaux, ... Vous pouvez l'afficher : à l'aide de la commande « », à l'aide du bouton de la barre de fonction, ou en appuyant sur la touche **F5**.



Afficher le navigateur.

Appuyer sur la touche **F5**.
 OO ouvre une fenêtre comme ci-dessus :

☞ Avant d'étudier le fonctionnement du navigateur, nous allons apprendre à maîtriser l'affichage du styliste et du navigateur.

☞ Vous pouvez ancrer, détacher et redimensionner le Navigateur ou le Styliste. Pour ancrer ou détacher le Navigateur ou le Styliste, maintenez la touche **Ctrl** enfoncée et double-cliquez sur une zone grise de la fenêtre à ancrer. Pour redimensionner le Navigateur ou

le Styliste, faites glisser un bord ou un angle de la fenêtre.

Ancrer le navigateur

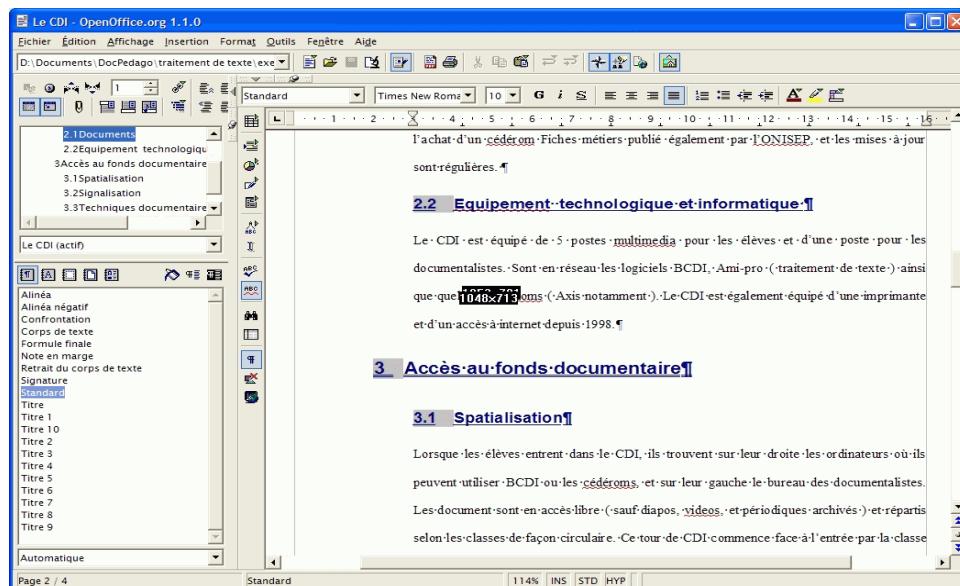
Maintenir la touche **Ctrl** enfoncée et Double-cliquer sur une zone grise du navigateur.
Relâcher la touche **Ctrl**
OOo ancre à gauche le navigateur.

Ancrer le styliste

Maintenir la touche **Ctrl** enfoncée et Double-cliquer sur une zone grise du styliste.
Relâcher la touche **Ctrl**
OOo ancre le styliste

Si nécessaire, réorganiser le navigateur au-dessus du styliste comme sur la figure ci-dessous.

Maintenir la touche **Ctrl** enfoncée et Cliquer dans la zone grise du navigateur, maintenir le bouton gauche de la souris appuyé.
Glisser le navigateur au dessus du styliste.
Relâcher le bouton de la souris puis la touche **Ctrl**.



 Chaque fenêtre ancrée contient deux icônes permettant de contrôler les propriétés d'affichage de cette fenêtre.

 permet d'afficher une fenêtre ancrée,  permet de masquer une fenêtre ancrée,

 pour fixer une fenêtre ancrée flottante,  pour rendre flottante, une fenêtre ancrée.

Masquer puis afficher à nouveau le styliste et le navigateur.

Cliquer sur le bouton  en haut à droite des fenêtres ancrées pour masquer le styliste et le navigateur.

Cliquer sur le bouton  pour masquer le styliste et le navigateur.

Rendre flottante puis fixer à nouveau le styliste et le navigateur

Cliquer sur le bouton  pour rendre flottant le styliste et le navigateur.

Cliquer sur le bouton  pour ancrer à nouveau le styliste et le navigateur.

 En attribuant des styles « », « » à nos chapitres, nous avons, non seulement mis en forme le document, mais nous avons aussi construit son plan : nous l'avons structuré.

 Dans ce T.P., nous n'utiliserons qu'une partie des fonctionnalités du navigateur : la partie afférente au plan du document, (l'équivalent du mode plan de Microsoft Word).

<p>Identifier le rôle du bouton</p> <p>« » </p> <p>Régler l'affichage du navigateur pour ne voir que les titres du document.</p>	<p>Cliquer plusieurs fois sur le bouton « » </p> <p>Observer l'affichage du contenu du navigateur.</p> <p>Faire apparaître tous les objets du navigateur.</p> <p>Cliquer sur « » puis sur le bouton </p> <p>Cliquer à nouveau sur le bouton  pour afficher tous les objets.</p> <p>Cliquer sur « » puis sur le bouton </p>
<p>Utiliser le navigateur pour se déplacer rapidement dans le document.</p>	<p>Double-cliquer sur un titre du navigateur</p> <p>Oo place le curseur devant le paragraphe.</p>

<p>Afficher uniquement les titres de niveau 1 à l'aide du bouton « N » </p> <p>Afficher les titres de niveau 1 à 3.</p>	<p>Cliquer sur le bouton « » </p> <p>Maintenir appuyé et sélectionner le niveau « ».</p> <p>Le navigateur affiche les titres de niveau 1</p> <p>Cliquer à nouveau sur le bouton </p> <p>Sélectionner le niveau « ».</p>
--	---

Numérotation automatique des chapitres

 En s'appuyant sur les styles « », « », ..., Oo est capable de numérotter automatiquement les chapitres : laissons le faire ...

 Nous allons insérer une numérotation automatique des chapitres du document. Avant d'insérer cette numérotation automatique, nous allons supprimer la numérotation manuelle.

Effacer la numérotation manuelle des chapitres.

Insérer une numérotation des chapitres à l'aide de la commande « » en respectant la présentation suivante :
1. Chapitre 1
 1.1 Chapitre 1.1
 1.2 Chapitre 1.2
etc ...
comme le document que vous êtes en train de lire.
Vérifier la numérotation des chapitres du document
 dans le document,
 dans le navigateur.
Enregistrer vos modifications

Balayer le document et Effacer les chiffres placés en début de ligne des chapitres.

Sélectionner la commande « ».
OOo affiche une boîte de dialogue « ».
Sélectionner le niveau « ».
Vérifier que le style paragraphe est à « ».
Sélectionner « » pour les numéros.
Saisir le « » (point) comme séparateur après
Sélectionner le niveau 2
Vérifier que le style paragraphe est à « ».
Sélectionner « » pour les numéros.
Sélectionner la valeur « » pour le champ complet.
Saisir le « » (point) comme séparateur après.
Valider
OOo met à jour le document en insérant une numérotation automatique des chapitres.
 pour enregistrer vos modifications.

N.B. : Ne pas confondre la numérotation automatique des chapitres avec les listes numérotées. Si la numérotation automatique des chapitres ne fonctionne pas, vérifier que le style du titre ne possède pas de numérotation dans sa définition. (onglet Numérotation du style)

Utiliser le navigateur pour modifier la structure du document

 Le navigateur permet non seulement d'accéder rapidement aux différentes parties d'un document : il permet aussi de modifier son plan, bien entendu la numérotation des chapitres suit... Dans le navigateur :

le bouton place le titre sélectionné, avec le texte associé, à un niveau de plan plus haut,
le bouton place le titre sélectionné, avec le texte associé, à un niveau de plan plus bas,
le bouton hausse le titre sélectionné, ainsi que les titres sous-jacents, d'un niveau de plan
le bouton abaisse le titre sélectionné, ainsi que les titres sous-jacents, d'un niveau de plan.

Remonter le chapitre

« » avant le chapitre
« ».

Vérifier le corps du texte du chapitre a bien suivi le déplacement du titre

Cliquer sur la ligne

« » du navigateur.

Cliquer sur le bouton

Noter le déplacement du titre, la mise à jour de la numérotation.

Hausser d'un niveau le chapitre

« ».

Cliquer sur la ligne

« » du navigateur.

Cliquer sur le bouton

Noter le déplacement du titre, la mise à jour de la numérotation.

N-B : Attention la commande « »

ne gère pas les modifications de structure dans le navigateur.

Exercice :

A l'aide du navigateur, annuler les dernières modifications concernant la structure du document.

Insérer un titre au document



Nous allons insérer un titre à notre document, en utilisant le style « Titre ».

N-B : OOo gère une forme d'héritage (liens) entre les différents styles. Par exemple, les styles « », « », ... héritent du style « ». Avant de modifier le style « », nous allons rompre le liens entre les styles « », « » et le style « ».

Supprimer le lien entre le style « et le style ».
Supprimer le lien entre le style et le style « ».

Cliquer droit sur le style « » du styliste et sélectionner la commande « ».
Sélection l'onglet « ».
Répéter le champ « »
Sélection l'option « »
Valider.
Répéter le même processus pour le styler « ».

En début de document, saisir le titre suivant :
« ».
Appliquer le style « » à ce paragraphe.
Modifier ce style « » en respectant les consignes suivantes :
- police :« »
- style :« »
- taille :« »,
- alignement :
- retrait avant le texte : « »
- retrait après le texte : « »
- écart avant de :.....
bordure :
«
style :
ombre :« »
Arrière plan :«

Placer le curseur en début de document.
saisir le texte «
Appuyer sur la touche pour insérer un paragraphe.
Remonter sur la ligne «
A l'aide du styliste, appliquer le style « » à ce paragraphe.

N-B : Sélectionner « » en bas du styliste si vous ne trouvez pas le style « ».

Cliquer droit sur « » du styliste et sélectionner la commande « ».
Modifier le style comme indiqué ci-dessous. Valider.

Gérer un sommaire automatique

Encore une fois, l'utilisation des styles « », « », ... va nous permettre de générer le sommaire du document automatiquement.

Sous le titre «
», à l'aide de la
commande «
», insérer une table des
matières automatique
Enregistrer vos modifications

Placer le curseur sous le titre «
».
Sélectionner la commande
« » de la
barre de menus.
affiche une boîte de dialogue
« »
Valider.


Modifier la structure du
document à l'aide du navigateur.

Sélectionner la ligne «
» dans le navigateur.
Cliquer sur le bouton  pour
abaisser d'un niveau le chapitre «
».
Vérifier le changement de
numérotation du chapitre.

Vérifier que le sommaire n'a pas
pris en compte la modification .

Visualiser le sommaire.

Mettre à jour le sommaire à l'aide
de la commande «
» du menu contextuel du
sommaire.

Cliquer gauche sur le sommaire.
Cliquer droit et sélectionner la
commande « ».

Vérifier la modification du
sommaire.

vérifier la modification du sommaire.

Rechercher du texte



Comme tout traitement de texte, Oo propose une fonction
« ».

A l'aide de la commande
« » , rechercher toutes les occurrences du mot « ».

Se placer en début de document.
Sélectionner la commande
« ».
OOo affiche une boîte de dialogue
« ».
Saisir le texte « ».
Cliquer sur le bouton
« R ».
positionne le curseur devant la première occurrence du mot « BCDI ».
Cliquer sur le bouton
« » plusieurs fois pour trouver toutes les occurrences du mot.
Cliquer sur le bouton « » pour refermer la boîte de dialogue
« ».

Insérer des notes de bas de page

❖ Nous allons insérer les notes de bas de page suivantes :

BCDI	Logiciel de gestion de fonds documentaire publié par le C.R.D.P. de Poitiers
ONISEP	Office National D'Informations Sur Les Enseignements et les Professions

A l'aide de la commande
« », insérer la note de page ci-dessus après la 1ère occurrence du « »

Placer le curseur juste après la 1ère occurrence du mot « ».
Sélectionner la commande
« ».
OOo ouvre une boîte de dialogue
« ».
Sélectionner l'option
« ».
Sélectionner l'option « ». Valider
positionne le curseur en bas de page.
Saisir le texte de la note suivant le tableau ci-dessus.

N.B. : Attention ne pas confondre les commandes : « **I** » qui permet d'un insérer un « post-it » dans votre document et la commande « **»**.

Exercice :

Répéter le même processus pour le mot « **ONISEP** ».
Enregistrer vos modifications.

en-tête et pied de page



Les en-têtes et pieds de page sont des zones de texte qui se répètent sur chaque page., comme sur ce document.

Insérer un pied de page contenant votre nom et votre prénom.

Sélectionner la commande « **»**
OOo insère une zone de texte en pied de la page.
Saisir

Vérifier l'insertion du pied de page grâce à l'aperçu avant impression

Sélectionner la commande « **»**.
Vérifier la présence du pied de page.
Fermer l'aperçu avant impression.



Un champ est une information dont le contenu est « calculé » directement par le logiciel. Quelques champs sont très utiles comme par exemple : le numéro de page, le nombre de pages, la date d'impression, ...

A l'aide des commandes

« », modifier le pied de page pour obtenir résultat ci-dessous :

Nom:Prénom → page 1 sur 2

Placer le curseur dans le pied de page après votre prénom saisi précédemment.

Appuyer sur la touche « » (à gauche de la touche A) pour insérer une tabulation.

Saisir le texte « ».

Selectionner la commande

« »

OOo insère le « ».

Saisir le texte « ».

Selectionner la commande

« » .

OOo insère le « ».

Vérifier votre pied de page à l'aide de l'aperçu avant impression

Selectionner la commande « ».

Vérifier votre pied de page ainsi modifié.

Fermer l'aperçu avant impression.

Par déplacement de la marque de tabulation dans la règle, ajuster la position de la pagination.

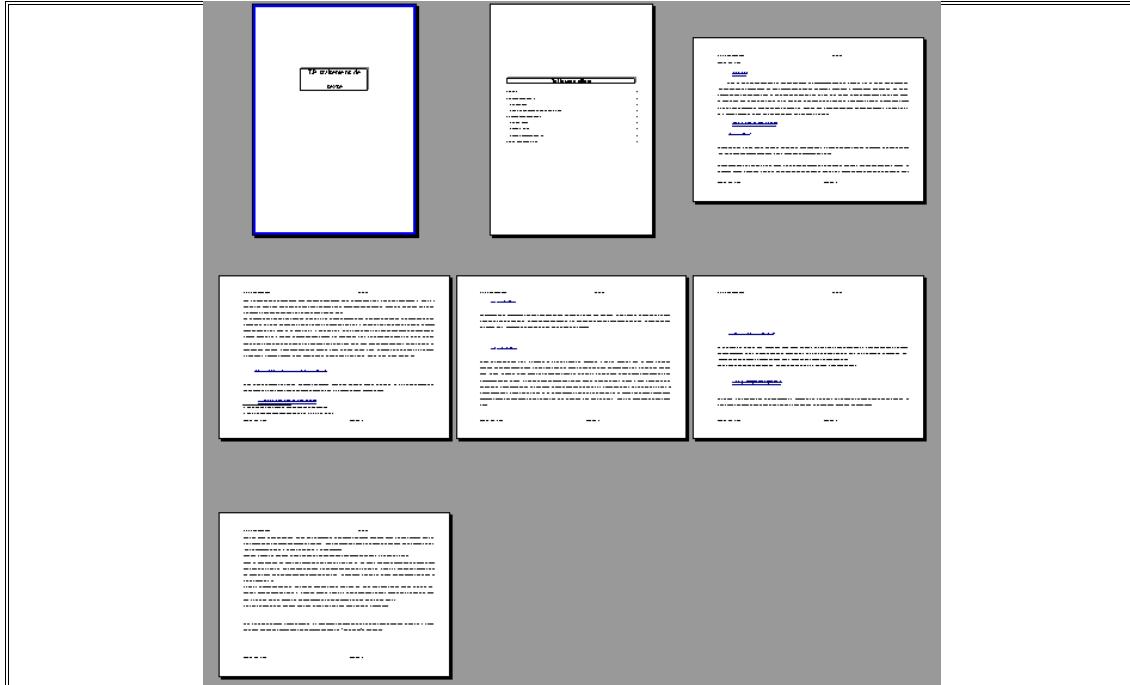
Cliquer dans le pied de page.
Afficher la règle
Repérer la marque de tabulation dans la règle.
Par glisser-déplacer, modifier la position de la marque de tabulation

Exercice : Insérer un en-tête de page contenant le champ « Titre » et le champ « Date ». Enregistrer vos modifications.

Utiliser les styles de page



Nous allons insérer des sauts de page et utiliser des styles pour obtenir une présentation similaire à celle de la figure ci-dessous :



Les styles de page permettent dans un même document de modifier l'orientation des pages, leurs marges, leurs en-têtes ou pieds de page.

Appliquer le style de page
« » à la première
page du document.

Placer le curseur sur le titre « ».
Afficher « » des
« » du styliste.
Double-cliquer sur le style
« ».

Insérer un saut de page manuel
après le titre « ».
à l'aide de la commande
« » en
appliquant le style « » à la
page suivante.

Placer le curseur à la fin de la ligne
« ».
Sélectionner la commande
« ».
OOo ouvre une boîte de dialogue
« ».
Sélectionner le type « ».
Sélectionner le style
« ». Valider.

Insérer un saut de page manuel après le titre « » à l'aide de la commande « » en appliquant le style « » à la page suivante.

Insérer un second saut de page en appliquant le style « » aux pages suivantes après le sommaire.

Modifier le style de page « » en respectant les consignes suivantes :
orientation..paysage
marge à gauche 3 cm
marge à droite 2 cm
marge en haut 2 cm
marge en bas 2 cm
en-tête de page Activer
pied de page Activer

Placer le curseur à la fin de la ligne « ».
Sélectionner la commande « » OOo ouvre une boîte de dialogue « ».
Sélectionner le type « ».
Sélectionner le style « ». Valider.

Placer le curseur après le sommaire.
Sélectionner la commande « ».
Sélectionner le type « ».
Sélectionner le style « ». Valider.

Afficher les styles de pages du stylsite
cliquer droit sur le style « ».
Sélectionner la commande « ».
Sélectionner l'onglet « P ».
Modifier si nécessaire les paramètres Orientations et les marges
Sélectionner l'onglet « ».
Si nécessaire, cocher la case « ».
Sélectionner l'onglet « ».
Si nécessaire, cocher la case « ». Valider

Modifier le style de page « » en respectant les consignes suivantes :
orientation.....portrait
marge à gauche 3 cm
marge à droite. .3 cm
marge en haut .2 cm
marge en bas....2 cm
en-tête de page désactiver
pied de pagedésactiver

Cliquer droit sur le style « » et sélectionner la commande « ».
Sélectionner l'onglet « ».
Modifier l'orientation et les marges si nécessaire.
Désactiver l'en-tête et le pied de page dans les onglets correspondants.

Modifier le style de page « » en respectant les consignes suivantes :
orientation.....portrait
marge à gauche 2 cm
marge à droite..2 cm
marge en haut .2 cm
marge en bas....2 cm
en-tête de page désactiver
pied de pagedésactiver

Cliquer droit sur le style « » et sélectionner la commande « ».
Sélectionner l'onglet « ».
Modifier l'orientation et les marges si nécessaire.
Désactiver l'en-tête et le pied de page dans les onglets correspondants.

Vérifier votre mise en page à l'aide de l'aperçu avant impression.

Sélectionner la commande « Fichier Aperçu avant Impression »
Vérifier votre mise en page.
Fermer l'aperçu.

Imprimer votre document.

Sélectionner la commande « Fichier Imprimer ».
Sélectionner l'imprimante.
Valider.

Enregistrer vos modifications.

 pour enregistrer votre travail.

Exporter au format PDF

 Le format PDF est un format de fichiers créé par la société Adobe permettant de visualiser et d'imprimer un fichier sur n'importe quelle plateforme via l'outil Acrobat Reader. Ooo propose une fonction d'exportation au format PDF.

Exporter au format PDF votre document avec comme nom de fichier « » dans votre dossier « », avec une qualité « ».

Sélectionner la commande « ». OOo ouvre une boîte de dialogue « ». Sélectionner l'emplacement « ». Saisir le titre ». Valider. OOo ouvre une seconde boîte de dialogue : « ». Sélectionner l'option « ». Valider. OOo génère un fichier au format PDF.

Enregistrer vos modifications.

Refermer OpenOffice

Ouvrir le dossier « » et vérifier la présence du document « »

Ouvrir le document à l'aide du logiciel « ». Conclusions.

Exercice : l'exercice qui vous est proposé est un exercice de consolidation. Charger le document exo8.sxw.

Appliquer le style titre 1 aux titres :

- 1) les différents champs disciplinaires et niveaux d'enseignement*
- 2) la gestion de la classe*
- 3) la coopération entre élèves, classes et enseignants*
- 4) Les systèmes d'information et de communication*
- 5) la responsabilité éducative de l'enseignant*

Retirer la numérotation manuelle.

Insérer une numérotation automatique des titres.

Insérer un saut de page après le titre « La formation initiale des enseignants et les Technologies de l'Information et de la Communication ».

Insérer un sommaire automatique et une numérotation de page en pied de page.

Enregistrer et refermer votre document en fin de travail.

Exporter le document au format PDF, qualité Impression



Dans un premier temps, vous allez créer un tableau et le mettre en forme.

Créer un tableau

Créer un nouveau document.

Saisir le texte «

Insérer un nouveau paragraphe.

Sélectionner la commande «

».

Saisir le texte

Appuyer sur la touche  pour créer un nouveau paragraphe

A l'aide de la commande «  », créer un tableau à 4 lignes et 4 colonnes.

Sélectionner la commande «  ». OOo affiche une boîte de dialogue «  ». Saisir 4 lignes et 4 colonnes pour la taille du tableau. Valider.

Noter l'apparition de la barre d'outils Tableau.

Noter l'apparition d'un bouton  à l'extrême droite de la barre d'objets : ce bouton permet de permuter entre la barre d'outils «  » et la barre d'outil «  ».

Compléter le tableau comme ci-dessous :

	Élèves estimant ne pas avoir de difficulté	Élèves estimant avoir des difficultés	Ensemble
OUI	11	10	21
NON	4	1	5
TOTAL	15	11	26

Mettre en forme un tableau

Centrer la première colonne du tableau.

Cliquer gauche sur la première cellule du tableau.

Maintenir appuyé le bouton gauche de la souris et déplacer la souris pour sélectionner la première colonne.

Relâcher le bouton de la souris.

Cliquer droit et sélectionner la commande «  » du menu contextuel.

Centrer la première ligne du tableau

Cliquer gauche sur la première cellule du tableau.

Maintenir appuyé le bouton gauche de la souris et déplacer la souris pour sélectionner la première colonne.

Relâcher le bouton de la souris.

Cliquer droit et sélectionner la commande « » du menu contextuel.

Mettre en caractères gras italiques la première ligne.

Sélectionner la première ligne.

Cliquer droit et sélectionner la commande « » du menu contextuel.

Cliquer droit et sélectionner la commande « » du menu contextuel.

Enregistrer votre document sous le nom « » dans votre dossier « ».

Sélectionner la commande « ».

Sélectionner le dossier « ».

Saisir le nom du fichier « ».

Valider.

Arrière-plan et bordure dans un tableau

Ajouter un arrière-plan à la première ligne du tableau.

Sélectionner la première ligne du tableau.

Repérer le bouton « »

dans la barre d'outils « ».

Cliquer sur ce bouton.

Oo affiche une boîte de dialogue « ».

Sélectionner la couleur de votre choix.

  pour enregistrer votre travail.

Insérer un fichier dans un fichier

Vous allez insérer un tableau provenant d'un autre de fichier.
Deux objectifs : gagner du temps de saisie et apprendre à insérer un fichier dans un document.

Après avoir insérer un paragraphe vide à la fin de votre document, insérer le fichier « » qui se trouve dans le dossier « » à l'aide de la commande « ».

Placer le curseur en fin de document.

Appuyer sur la touche pour créer un nouveau paragraphe.

Sélectionner la commande « ».

OOo affiche une boîte de dialogue « ».

Ouvrir le dossier « ».

Sélectionner le fichier « » Valider.

OOo insère le tableau contenu dans « ».

Insérer une ligne et une colonne dans un tableau

Insérer une ligne au tableau avant la ligne « ».

Cliquer sur la ligne « ».

Cliquer droit et sélectionner la commande « » du menu contextuel.

OOo ouvre une boîte de dialogue « ».

Sélectionner l'option « » et le nombre de ligne « ».

Insérer une colonne au tableau avant la dernière colonne.

Cliquer gauche dans la dernière colonne du tableau.

Cliquer droit et sélectionner la commande « C » du menu contextuel.

Sélectionner l'option « » et le nombre de colonne « ».

Compléter le tableau comme ci-dessous.

Compléter le contenu de la ligne et de la colonne qui viennent d'être insérées.

	ELEVES ESTIMANT NE PAS AVOIR DE DIFFICULTE (heures/sema- ne)	ELEVES ESTIMANT AVOIR DES DIFFICULTES (heures/sema- ne)	ENSEMBLE
TEMPS DE FORMATION	38,17	42,41	40,29
dont présence au lycée	32,64	35,80	34,22
dont trajet scolaire	5,53	6,61	6,07
AIDE DOMESTIQUE	3,41	6,89	5,15
LOISIRS D'INTERIEUR	22,54	27,17	24,86
dont télévision	10,82	14,87	12,85
dont lecture	2,27	1,37	1,82
dont écoute de la musique	8,27	10,43	9,35
dont autres	1,18	0,50	0,84
LOISIRS DE PLEIN AIR	2,64	2,11	2,38
dont sport	2,64	2,11	2,38
SORTIES	4,36	5,86	5,11
dont visites familiales	1,00	1,33	1,17
dont sorties avec camarades	3,36	4,53	3,95

*Exercice : Modifier le tableau pour obtenir la présentation ci-dessus.
Un conseil : ne pas chercher à fusionner des cellules mais
jouer simplement sur la visibilité des bordures. Enregistrer
votre travail en fin d'exercice.*



Dans cette partie, vous allez apprendre à insérer des images et à les disposer dans la page.

Insérer une image à partir d'un fichier

Ouvrir le fichier « » qui se trouve dans le dossier « ». Enregistrer ce document dans votre dossier « » sous le nom « ».

Sélectionner la commande « ». Ouvrir le dossier « ». Double-cliquer sur le fichier « ». OOo charge le document. Sélectionner la commande « ». Sélectionner le dossier « ». Saisir le nom « ». Valider.

Insérer le fichier image « » qui se trouve dans le dossier « e » dans un paragraphe vide devant le texte « ».

Placer le curseur devant le texte « ». Insérer un nouveau paragraphe en appuyant sur la touche . Placer le curseur au début du paragraphe que vous venez de créer. Sélectionner la commande « ». affiche une boîte de dialogue « ». Sélectionner le fichier image « » qui se trouve dans le dossier « ». Valider.

Enregistrer votre travail.

N-B : On peut bien entendu insérer des images par copier-coller, depuis le navigateur Internet ou un autre document par exemple.

Modifier la taille de l'image

Selectionner l'image et modifier sa taille à l'aide des poignées de sélection.

Cliquer gauche sur le l'image.
OOo affiche 8 poignées vertes de sélection.

Cliquer en dehors de l'image.
Les poignées vertes disparaissent.
Cliquer sur l'image pour la sélectionner à nouveau.

Glisser-déplacer la poignée verte en bas au milieu pour réduire la hauteur de l'image.

[Ctrl] [Z] pour annuler la dernière modification.

Modifier la taille de l'image en conservant les proportions
(Maintenir la touche enfoncee lors du deplacement des poignees de selection).

Maintenir la touche enfoncee.
Glisser-déplacer la poignée verte en bas au milieu pour réduire la taille de l'image.

[Ctrl] [S] pour enregistrer vos modifications.

Modifier l'habillage d'une image



Pour l'adaptation du texte autour de l'image, OOo propose plusieurs options :

	Place l'image sur une ligne distincte dans le document. Le texte du document est affiché au-dessus et en dessous de l'image, mais pas sur les côtés de l'image.
	Adapte le texte à gauche de l'image s'il y a suffisamment de place.
	Adapte le texte à droite de l'image s'il y a suffisamment de place.
	Adapte le texte tout autour du cadre de l'image.
	Place l'image devant le texte.
	Adapte automatiquement le texte à gauche, à droite ou tout autour du cadre de l'image. Si la distance entre l'image et la marge de page est inférieure à 2 cm, le texte n'est pas adapté.

Sélectionner une adaptation dynamique à l'aide du menu contextuel ou de la commande « ».

Cliquer droit sur l'image que vous venez d'insérer et sélectionner la commande « » du menu contextuel.
Observer la mise en page.
Glisser-déplacer l'image pour la placer sur le côté gauche de la page.

Répéter le même processus pour les images suivantes :

Clientserveur.gif	Avant le texte « »	Aucun
Login.gif	Avant le texte « »	A gauche
Internet.gif	Avant le texte « »	parallèle

Enregistrer votre travail. Ne pas refermer le document.

Ancre de l'image dans le texte



Une fois l'image insérée dans le document, nous sommes souvent amené à modifier le reste du document : ajout, suppression, modification de texte... ce qui peut désorganiser la mise en page des images. OOo propose différents mode d'ancre pour les images :

	Ancre l'élément sélectionné à la page active. L'image restera fixe sur la page.
	Ancre l'élément sélectionné au paragraphe actif. L'image suit le paragraphe.
	Ancre l'élément sélectionné à un caractère. L'image suit le caractère.
	L'image se comporte comme un caractère.
	Ancre l'élément sélectionné au cadre qui l'entoure.

« » l'image
« » à l'aide du
menu contextuel.

Sélectionner l'image « ». (clic gauche)
Cliquer droit et sélectionner la
commande « ».
Régler la position de l'image.

Enregistrer vos modifications.

Insérer une image à partir de la galerie



OpenOffice gère une galerie d'images, plus généralement d'objets que vous pouvez facilement insérer dans votre document.

Ouvrir la galerie.

Sélectionner la commande
« ».
Cocher si nécessaire l'item
« ».
OOo ouvre la galerie.

N-B : Comme le styliste ou le navigateur, la fenêtre « Gallery » peut être ouverte mais masquée.

Masquer puis afficher la galerie.

Cliquer sur le bouton « »
de la galerie.
OOo masque la galerie.
Cliquer sur le bouton « »
OOo affiche la galerie.

Sélectionner le thème « »
de la galerie
Insérer une puce de votre choix
par glisser-déplacer.

Cliquer sur le thème « » de la
galerie.
Affiche le contenu du thème.
Cliquer sur une puce de votre choix.
Maintenir appuyé le bouton gauche de
la souris.
Glisser-déposer la puce dans votre
document.
Relâcher le bouton
OOo insère la puce.

N-B : La galerie d'OpenOffice est très pauvre. Vous pourrez l'enrichir à partir

d'images récupérées sur Internet.
Bien entendu, si vous avez une licence StarOffice (gratuit pour l'éducation), vous pourrez récupérer sa « gallery ».

*Exercice : Régler l'ancre de la puce « **comme caractère** ». Par copier-coller, modifier la présentation du paragraphe « **Ressources partagées** » comme sur la figure ci-dessous.*

Ressources partagées (services) de l'Internet

Sur l'Internet, des serveurs vont donc proposer leurs ressources partagées, leurs services, à ses utilisateurs. Potentiellement tous les services accessibles sur un réseau interne d'entreprise peuvent être rendus accessibles sur l'Internet. Plusieurs types de service se sont développés. On distingue principalement :

- le world wide web,
- le courrier électronique,
- le forum de discussion (newsgroup),
- la conversation à deux ou plus en temps réel (chat),
- le téléchargement de fichiers informatiques (FTP)

Masquer la galerie.

Enregistrer vos modifications. Ne pas refermer le document.

Rappel : OpenOffice vous propose, par ailleurs, un module de dessin vectoriel très complet.

Créer un nouveau texte.

 Nous allons réaliser le dessin de la figure ci-dessous puis nous l'insérerons dans le document « _____ ».

 Les boutons des barres d'outils ayant un petit triangle vert permettent à d'accéder à une barre d'outils complémentaire. Un clic gauche prolongé sur le bouton ouvre cette barre d'outils complémentaire : cette barre peut alors se détacher et rester apparente de manière permanente.

Afficher la barre d'outils Dessin

Afficher et détacher la barre d'outils Dessin



Repérer le bouton « » de la barre d'instruments (4ème bouton en partant du haut)

Cliquer gauche sur ce bouton. Maintenir appuyé.

Oo ouvre une nouvelle barre d'outils.

Cliquer gauche dans la barre de titre.

Détacher la barre d'outils par glisser-déplacer.

Identifier la fonction des différents boutons de la barre d'outils.

	Permet de sélectionner un objet ou (avec la touche Maj. enfoncée) plusieurs objets pour les éditer ou les déplacer ensemble.		Permet de dessiner un arc d'ellipse. Maintenez la touche Maj. enfoncée pour dessiner un arc de cercle.
	Ceci vous permet de tracer une ligne droite. Lorsque vous maintenez la touche (Maj) enfoncée pendant le traçage, la ligne sera alignée sur des angles incrémentés de 45° par rapport au point de départ.		Permet de dessiner un secteur d'ellipse. Maintenez la touche Maj. enfoncée pour dessiner un secteur de cercle.
	Ceci vous permet de dessiner un rectangle ; il pourra également s'agir d'un carré si vous maintenez la touche (Maj) enfoncée tout en procédant au dessin.		Permet de dessiner un segment d'ellipse. Maintenez la touche Maj. enfoncée pour dessiner un segment de cercle. Placez le pointeur sur la zone de marge de l'ellipse prévue.
	Ceci vous permet de dessiner une ellipse ; il pourra également s'agir d'un cercle si vous maintenez la touche (Maj) enfoncée tout en procédant au dessin.		permet de définir un cadre texte pour y saisir un texte.
	Permet de dessiner un polygone. Pour fermer le polygone, double-cliquez sur le point de départ.		Permet d'insérer un défilement de texte dans le document.
	Permet de définir une courbe de Bézier libre.		Permet de définir une légende avec trait de renvoi.
	Permet de dessiner une ligne à main levée.		

Insérer un cadre de texte

Insérer un cadre de texte contenant le texte « ».

Cliquer sur le bouton « ».
Par glisser-déplacer, dessiner un premier rectangle.
Saisir le texte « ».

T

Identifier le mode édition de texte

Cliquer en dehors du cadre de texte pour quitter le mode édition.

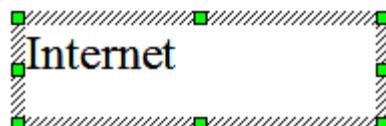
L'objet n'est pas sélectionné.

Cliquer gauche sur le cadre de texte.



L'objet est sélectionné : 8 poignées vertes apparaissent, la barre d'objet a changé.

Double-cliquer à l'intérieur du cadre de texte.



Vous êtes en mode édition de texte : une bordure est apparue, la barre d'objet a encore changé.



En mode édition de texte, vous pouvez modifier le contenu du cadre de texte et sa mise en forme de caractères et de paragraphes : police, alignement, ... (voir le menu contextuel et la barre d'objets)

Modifier le contenu du cadre de texte

Modifier la mise en forme du cadre de texte en respectant les consignes ci-dessous :

alignement : « »

police : « »

taille : « »

style : « ».

Double-cliquer à l'intérieur du cadre de texte.

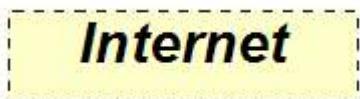
Centrer le paragraphe à l'intérieur du cadre de texte à l'aide de la barre d'objets.

Sélectionner le mot « ».

Modifier sa police, son style, sa taille à l'aide de la barre d'outils.

Modifier les propriétés du cadre de texte

Modifier le contour et le remplissage du cadre de texte à l'aide des commandes « » et « » pour obtenir l'effet ci-dessous :



Cliquer à l'extérieur du cadre de texte pour sortir du mode édition de texte.

Cliquer sur le bouton Sélection de la barre d'outils Dessin.

Cliquer sur le cadre de texte.

Oo sélectionne le cadre de texte et affiche des poignées de sélection (petits carrés verts autour du cadre de texte).

Cliquer droit sur le cadre de texte et sélectionner la commande « » du menu contextuel.

affiche une boîte de dialogue « ».

Sélectionner le style « ». Valider.

Cliquer droit et sélectionner la commande « » du menu contextuel.

affiche une boîte de dialogue « ».

Sélectionner l'onglet « ». Choisir une couleur. Valider.

Modifier la position du texte dans son cadre

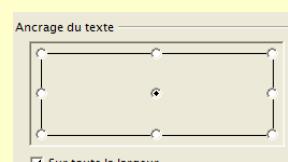


Pour le moment le mot Internet est centré horizontalement, pas verticalement dans son cadre. Nous allons y remédier.

A l'aide de la commande « », ancrer le texte au milieu du cadre de texte.

Sélectionner le cadre de texte.

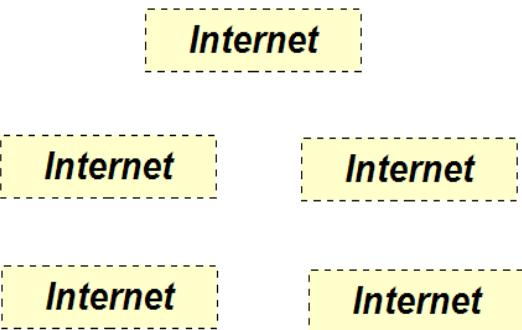
Cliquer droit et sélectionner la commande « ».



Sélectionner un ancrage centré comme ci-contre. Valider.

Copier - coller un cadre de texte

Par copier-coller du cadre de texte puis déplacement construire la figure suivante :



Modifier le contenu des cadres de texte en remplaçant le mot « Internet » par les mots : « », « », « », « » suivant la 1^{ère} figure du chapitre 7.

Insérer des lignes

A l'aide de l'outil ligne , tracer une ligne entre le mot « » et le « ».

Sélectionner le cadre de texte.

Cliquer droit sur le cadre de texte et sélectionner la commande « » du menu contextuel.

Cliquer droit à l'extérieur du cadre de texte et sélectionner la commande « ».

Déplacer le cadre de texte.

Répéter 3 fois le processus ci-dessus pour obtenir la figure ci-contre.

Déplacer approximativement les différentes zones afin de s'approcher de la disposition finale du schéma à obtenir.

Modifier le contenu des cadres de texte en remplaçant le mot « Internet » par les mots : « », « », « », « » suivant la 1^{ère} figure du chapitre 7.

Sélectionner l'outil ligne .

Par glisser-déplacer , dessiner une ligne entre le cadre de texte « » et le cadre de texte « ».

Appuyer sur la touche « » pour quitter l'outil « ».

Transformer la ligne en flèche à l'aide de la commande « ».

Cliquer droit sur la flèche et sélectionner la commande « » du menu contextuel.

Oo affiche une boîte de dialogue « ».

Sélectionner un style de flèche à l'extrémité de la ligne.

Valider.

Par copier-coller, créer 3 autres flèches.

Par glisser déplacer sur les extrémités des flèches, ajuster la position des flèches.

Sélectionner plusieurs objets



Il est possible de sélectionner plusieurs objets en même temps pour les aligner, les déplacer, ... Deux techniques : soit les encadrer à l'aide de la flèche de sélection, soit cliquer sur chaque objet en maintenant appuyée la touche .

Sélectionner par encadrement les cadres « » et « », puis les aligner en haut à l'aide de la commande « ».

Cliquer sur le bouton « »

de la barre d'outils Dessin.

Cliquer juste en haut à gauche du cadre de texte « », maintenir appuyé le bouton de la souris.

Déplacer la souris pour encadrer les cadres de texte « » et « ».

Relâcher le bouton de la souris.

Oo a sélectionné les 2 cadres de texte.

Cliquer droit et sélectionner la commande « »

Oo a aligné les deux cadres de texte.

Sélectionner par ajout (touche appuyée) les cadres de texte « » et « ». puis les aligner en haut

Cliquer sur le bouton « »

de la barre d'outils Dessin.

Maintenir appuyée la touche .

Cliquer sur le cadre de texte « » puis « ».

Relâcher la touche .

Oo a sélectionné les deux cadres.

Cliquer droit et sélectionner la commande « »

*Exercice : Aligner par un centrage vertical les cadres de texte « **Inter** », « **Inter connexion** » et la flèche qui les relient.
Aligner par un centrage vertical les cadres de texte « **Net** », « **de Réseau** » et la flèche qui les relient.*

Grouper des objets

 Le schéma se complète peu à peu... Il devient nécessaire de figer les objets les uns par rapport aux autres... La commande Grouper le permet.

Sélectionner tous les objets et les grouper à l'aide de la commande « < > »

Cliquer sur le bouton « > »

 de la barre d'outils Dessin.

Sélectionner tous les objets.

Cliquer droit et sélectionner la commande « > »

Oo a groupé l'ensemble des objets sélectionnés.

N-B : Vous pouvez dissocier un groupe (Commande « > »).

 Pour modifier un objet d'un groupe sans dissocier, vous pouvez entrer dans un groupe (Commande « > »)

Exercice : Modifier la couleur de remplissage du cadre de texte « Internet » sans dissocier le groupe.

Copier un dessin dans un autre document

 Notre schéma est terminé, nous allons le copier dans le document « > ».

Sélectionner le mode « » pour l'adaptation de texte du groupe.
Par copier coller, insérer le schéma dans le document
« »
après le titre du chapitre
« ».

Sélectionner le groupe.
Cliquer droit sur le groupe et sélectionner la commande « ».
Cliquer droit et sélectionner la commande « ».
Activer le document « ».
Placer le curseur après le titre « ».
Appuyer sur la touche pour créer un paragraphe vide.
Cliquer droit et sélectionner la commande « ».
OOo insère notre schéma dans le document.

Enregistrer vos modifications dans le document « ».

Quitter OpenOffice.org sans enregistrer le document qui nous a servi à construire le schéma, nous l'avons déjà dans le document « ».

	Nouveau paragraphe		Passage à ligne
	Saut de page		Tabulation
	Espace insécable		
	Aide		Aide contextuelle
	Activer / désactiver le navigateur		Activer / désactiver le styliste
	Annuler la dernière action		Caractères soulignés
	Caractères gras		Caractères italique
	Enregistrer		Sélectionner tout
	Copier		Couper
	Coller		
	Début de ligne		Fin de ligne
	Début document		Fin Document
Pour obtenir les caractères suivants : ~# { [` \ ^ @] } € utiliser la touche			

Bibliographie

Manuel de typographie française élémentaire, Yves Perrousseaux, Atelier Perrousseaux Editeur

Le style du Monde édité par la SA Le Monde

Webographie

<http://www.perrousseaux.com>

<http://bokane.citi2.fr/typo/>

http://www.synapse-fr.com/typographie/TTM_0.htm

<http://www.interpc.fr/mapage/billaud/typopao.htm>

Prise en main de draw

Table des Matières

À propos de ce manuel.....	5
Icônes utilisées dans ce manuel.....	5
Commentaires.....	6
Informations générales sur l'installation.....	6
Types d'installation.....	6
Configuration système requise	7
Généralités.....	7
Particularités concernant l'environnement d'exploitation Solaris™ (SPARC® Platform Edition) - environnement d'exploitation Solaris (SPARC).....	7
Particularités concernant Linux.....	7
Particularités concernant Windows.....	8
Contenu des paquetages d'installation téléchargés	9
Opérations préalables à l'installation à partir d'un jeu téléchargé.....	10
Mise à jour d'une installation existante	10
Mise à jour d'une installation multiutilisateur.....	10

Installation monoutilisateur	10
Conditions requises pour l'installation.....	11
Démarrage de l'installation.....	11
Installation à partir d'un jeu téléchargé sur plate-forme UNIX.....	11
Installation à partir d'un jeu téléchargé sous Windows.....	11
Déroulement de l'installation.....	13
Écran de bienvenue.....	13
Informations importantes.....	14
Contrat de licence.....	14
Données d'identité.....	14
Choix du type d'installation.....	16
Sélection des composants	16
Répertoire d'installation.....	17
Vérification des options d'installation.....	18
Assignation des types de fichier.....	18
Installation de l'environnement d'exécution Java™	20
Démarrage du processus de copie.....	21
Installation : fin.....	21
Démarrage d'OpenOffice.org	21
Installation multiutilisateur ou en réseau	22
Installation serveur	23
Conditions requises pour l'installation.....	23
Démarrage de l'installation.....	23
Installation à partir d'un jeu téléchargé sur plate-forme UNIX.....	23
Installation à partir d'un jeu téléchargé sur plate-forme Windows.....	23
Déroulement de l'installation.....	24
Écran de bienvenue.....	24
Informations importantes.....	25
Contrat de licence.....	25
Choix du type d'installation.....	25
Sélection des composants	26
Répertoire d'installation.....	27
Installation de l'environnement d'exécution Java™	29
Vérification des options d'installation.....	29

Démarrage du processus de copie.....	30
Installation : fin.....	30
Installation de station de travail	31
Conditions requises pour l'installation.....	31
Démarrage de l'installation.....	31
Installation sur plate-forme UNIX.....	31
Installation sur plate-forme Windows.....	32
Déroulement de l'installation.....	33
Écran de bienvenue.....	33
Informations importantes.....	34
Contrat de licence.....	34
Données d'identité.....	34
Type d'installation.....	35
Répertoire d'installation.....	36
Vérification des options d'installation.....	36
Attribution des types de fichier.....	36
Environnement d'exécution Java™	37
Démarrage du processus de copie.....	37
Installation : fin.....	37
Démarrage d'OpenOffice.org	38
Appendice.....	38
Installation de l'environnement d'exécution Java™ sous Windows.....	39
Paramétrage des imprimantes, fax et polices pour les plates-formes UNIX®.....	39
Paramétrage des imprimantes.....	39
Ajout d'une imprimante.....	40
Pilotes d'imprimante dans OpenOffice.org	40
Import de pilote lors de l'ajout d'une nouvelle imprimante.....	40
Suppression de pilote lors de l'ajout d'une nouvelle imprimante.....	41
Modification des paramètres de l'imprimante.....	41
Attribution d'un nouveau nom à l'imprimante ou suppression.....	42
Intégration d'un périphérique fax.....	42
Connexion d'un convertisseur PostScript - PDF.....	43
Installation de polices.....	44
Ajout de polices.....	44
Suppression de polices.....	45
Attribution de nouveaux noms aux polices.....	45
Installation d'un patch dans l'environnement d'exploitation Solaris™.....	45
Modification d'une installation OpenOffice.org existante	46
Modification	47
Réparation.....	47

Suppression	47
Paramètres d'installation.....	47
Démarrage d'OpenOffice.org avec des paramètres.....	48
Démarrage d'OpenOffice.org à partir de lignes de commande.....	48
Paramètres de ligne de commande.....	48
Enregistrement d'OpenOffice.org	50
Extension d'OpenOffice.org	51
Installation, mise à jour et suppression d'extensions dans une installation monoutilisateur.	
52	
Installation, mise à jour et suppression d'extensions dans une installation multiutilisateur .	
52	

Draw est un outil de dessin vectoriel. Il propose des outils puissants permettant de réaliser rapidement des graphiques de toutes sortes.

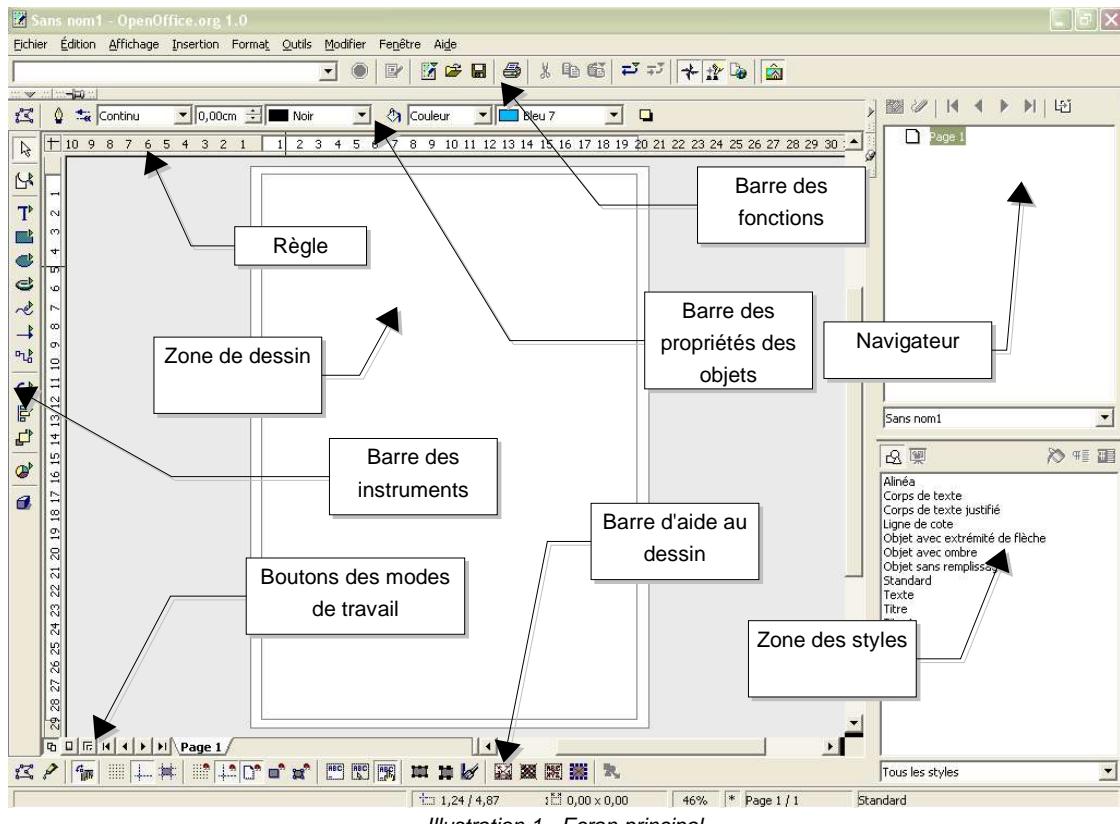
Son intégration parfaite dans la suite OpenOffice.org va faciliter les échanges des graphiques entre tous les modules. Il est ainsi particulièrement aisé de tracer un graphique et de le réutiliser dans Writer. Il est aussi possible de travailler directement dans Writer en utilisant un sous-ensemble des fonctions et des outils de Draw.

Les fonctionnalités de Draw sont très complètes. Sans pouvoir rivaliser avec les ténors du domaine, il possède néanmoins plus de fonctions que les outils de dessin habituellement intégrés dans les suites bureautique.

On peut citer (liste non exhaustive) la gestion de couches, un système complet de points magnétiques, le tracé de cotations associatives, des connecteurs facilitant la réalisation d'organigrammes, des fonctions 3D permettant de réaliser de petits dessins tridimensionnels (avec entre autres la gestion des textures et des éclairages), l'intégration de styles de dessin et de pages, le tracé de courbes de Bézier...

Ce document ne décrira que les fonctions propres aux tracés de dessins. Certaines notions comme la gestion des fichiers ou le fonctionnement du bureau d'OpenOffice.org ne seront abordées que de façon extrêmement succincte.

Lors du démarrage de Draw, l'écran de travail se présente sous la forme suivante:



La zone principale située au centre de l'écran est celle dans laquelle vos dessins vont être réalisés. Elle est entourée de barres d'outils et de zones d'information dont le nombre et la disposition peuvent varier en fonction de l'agencement que vous avez choisi pour votre espace de travail. Le dessin ci-dessus correspond à un exemple d'organisation parmi d'autres.

Les différentes barres d'outils de Draw peuvent être affichées ou masquées selon vos besoins.



Pour afficher ou masquer les barres d'outils, il vous suffit de cliquer sur n'importe quelle barre avec le bouton droit de la souris. Un menu vous permettant de sélectionner les barres d'outils à afficher apparaît alors.

Ce menu vous permet aussi de choisir les boutons que vous souhaitez voir apparaître sur la barre d'outils correspondante. Il vous suffit de choisir l'option 'Boutons visibles'. Toutes les barres affichables ne sont pas accessibles via ce menu.

Si vous cliquez sur une barre d'outil avec le bouton gauche en pressant en même temps sur la touche [Control], vous aurez la possibilité de la déplacer sur une autre zone de l'écran ou même de la détacher complètement pour en faire une fenêtre flottante. Il s'agit là d'un fonctionnement commun à toutes les barres d'outils de la suite OpenOffice.org.

Les fonctions de ces différentes barres d'outils vont être explicitées dans les paragraphes suivants.

La barre de fonctions

Cette barre se présente sous la forme suivante:



Illustration 3 - La barre de fonctions

Elle est commune à tous les modules d'OpenOffice et ne sera donc pas décrite précisément dans ce document.

La barre d'objets

Cette barre permet de visualiser rapidement et de modifier les propriétés principales des objets manipulés. Les icônes affichées vont varier en fonction de l'objet sélectionné.



Illustration 4 - La barre d'objets

Dans le cas ci-dessus, les fonctions de la barre permettent de modifier la couleur de tracé, la couleur de remplissage etc... de l'objet actuellement sélectionné. Si l'objet sélectionné est un texte, la barre des objets va prendre l'aspect suivant:



Illustration 5 - Barre d'objets en mode texte

La barre d'instruments

Cette barre est la plus importante de Draw. Elle contient les fonctions permettant de tracer les différentes figures et d'organiser les objets entre eux.

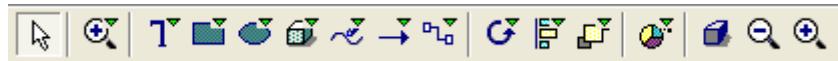


Illustration 6 - La barre d'instruments

La barre d'outils est ici présentée horizontalement mais elle est par défaut placée verticalement sur le bord gauche de la zone de travail. Comme pour les autres barres d'outils d'OpenOffice, les icônes possédant une petite flèche verte permettent d'accéder à des sélections supplémentaires. Par exemple, si l'on clique sur l'icône représentant un rectangle, la fenêtre suivante va s'afficher:

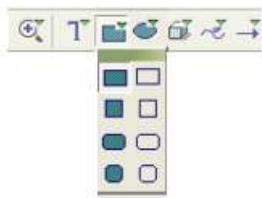


Illustration 7 - Sous-fonctions

Il est ainsi possible de choisir le type de rectangle que l'on va tracer (droit ou arrondi, rempli ou pas...). D'autre part, si l'on choisit une option différente de l'icône précédemment affichée, elle deviendra la nouvelle icône par défaut.

La barre de couleurs

Pour accéder à cette barre, il faut passer par le menu 'Affichage / barre d'outils'. Elle n'est en effet pas accessible via le menu décris page 123. La barre apparaît alors en bas de l'écran de la façon suivante:

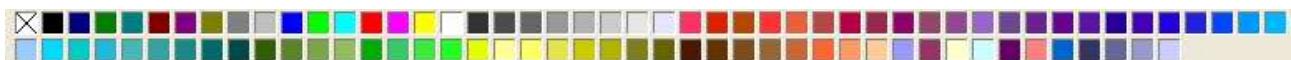


Illustration 8 - Barre de couleurs

Cette barre permet de sélectionner rapidement la couleur des objets. La première case de la barre marquée d'une croix correspond à la couleur transparente. La palette des couleurs affichée par défaut peut être modifiée par le menu 'Format / remplissage'. Choisissez l'onglet 'Couleurs':

Si vous cliquez sur le bouton entouré, la boîte de sélection de fichier vous invite à choisir un fichier de palette (extension SOC). Plusieurs palettes sont fournies en standard avec OpenOffice.org. Web.soc par exemple est une palette plus particulièrement adaptée à la réalisation de dessins destinés à apparaître dans des pages WEB et pouvant s'afficher correctement sur des postes de travail avec des écrans en 256 couleurs.

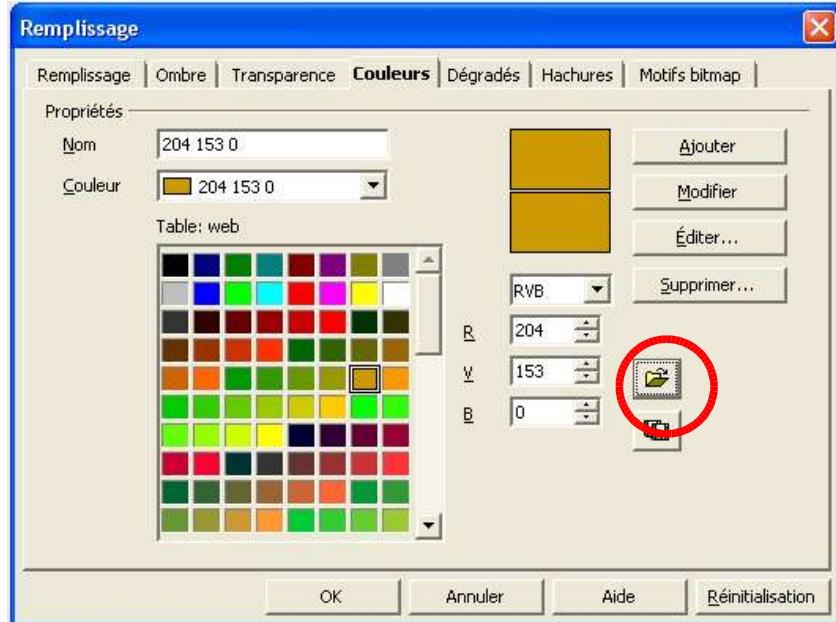


Illustration 9 - Boîte de choix des couleurs

Cette boîte de sélection vous permet aussi de changer individuellement n'importe quelle couleur en utilisant les zones de saisie numérique situées à droite de la palette des couleurs. Vous pouvez aussi cliquer sur 'Éditer' et une boîte de dialogue facilitant le choix des couleurs s'affiche:

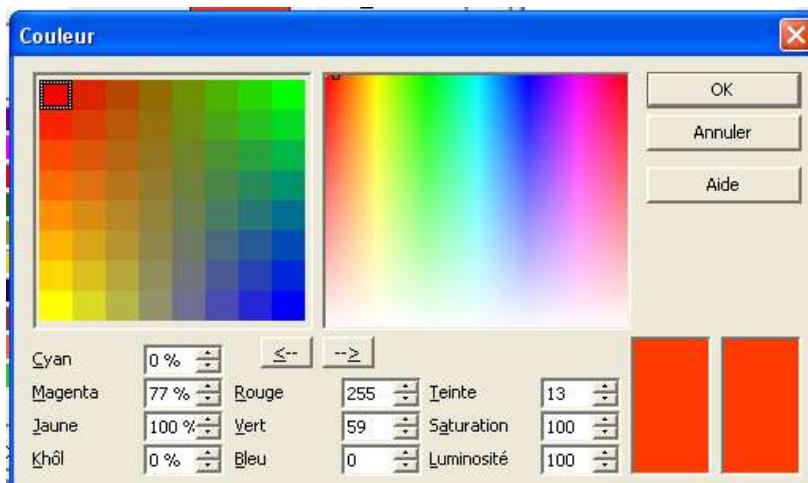


Illustration 10 - Edition des couleurs

Vous pouvez utiliser les modèles de couleurs CMJN (Cyan, Magenta, Jaune, Noir¹), RVB (Rouge, Vert, Bleu) ou TSL (Teinte, Saturation, Luminosité).

¹ Un petit défaut de traduction dans la version 1.0 d'OpenOffice.org fait que le texte 'noir' du modèle CMJN est remplacé par 'Khôl' qui correspond au noir dans le modèle anglo-saxon (CMYK). Le khôl correspondant à un noir très profond.

La barre d'options

Cette barre permet d'activer ou de désactiver différents mécanismes permettant de faciliter les tracés. Contrairement aux autres barres d'outils, la barre d'options n'est pas visible par défaut. Pour l'afficher, reportez-vous à la méthode décrite page 123. Lorsqu'elle est visible, elle se trouve en général en bas de l'écran et se présente sous la forme suivante:



Illustration 11 - Barre des options

Sur les parties supérieure et gauche de la zone de travail figurent les règles permettant de connaître à tout instant les dimensions des objets tracés. Le déplacement de la souris dans la zone de travail est matérialisé dans les règles pour permettre de positionner précisément les objets.

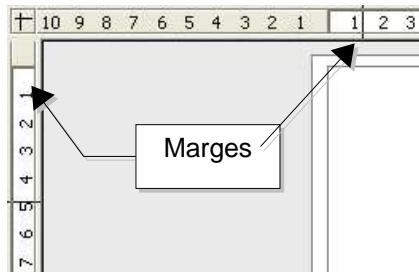


Illustration 12 - Les règles

Les marges de travail sur la zone de dessin sont aussi indiquées sur les règles. Il est possible de modifier directement les marges sur les règles en les faisant glisser avec la souris.

Vous pouvez aussi directement modifier l'unité des règles en cliquant sur une des deux règles avec le bouton droit de la souris:

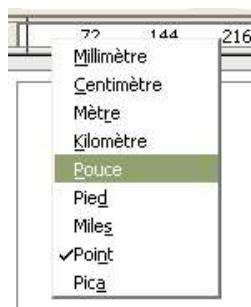


Illustration 13 - Unité des règles

Il est à noter que les unités des deux règles peuvent être différentes.

Les règles sont aussi utilisées pour gérer des points et des lignes de capture facilitant le positionnement des objets. Ce sujet sera abordé à la page 207.

Cette barre est située tout en bas de l'écran. Nous nous intéresserons ici à la partie centrale de cette zone car elle concerne plus particulièrement Draw. Les trois sections qui nous intéressent sont décrites dans le schéma suivant:



Illustration 14 - Barre d'état

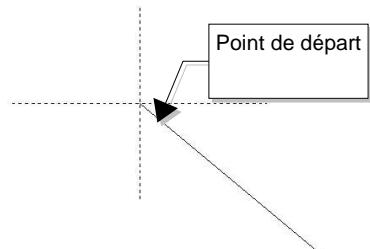
Les dimensions sont affichées dans l'unité courante (à ne pas confondre avec l'unité des règles). Cette unité peut-être définie via la boîte des options, section 'Dessin', sous-section 'Général'. Cette boîte vous permet aussi de modifier l'échelle de la page. Vous pouvez aussi modifier l'échelle en double-cliquant sur le chiffre affiché dans la barre d'état.

Ce chapitre décrira les mécanismes de base permettant de tracer des figures simples. Dans la suite du document, j'utiliserai le terme 'objets' pour désigner les différentes figures tracées (qu'il s'agisse de simples traits, de rectangles ou de figures plus complexes). Cette dénomination est courante en matière de logiciels de dessins vectoriels.

Les trois chapitres suivants illustreront le tracé de trois figures élémentaires: la droite, le rectangle et le cercle. Vous trouverez un descriptif de tous les types de tracés disponibles à la page 203.

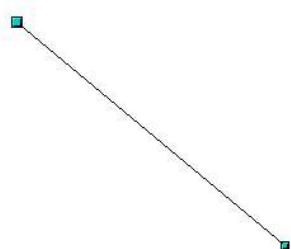
Tracé d'un segment de droite

Nous allons maintenant tracer la plus simple des figures, un simple segment de droite. La première étape consiste à sélectionner l'icône . Si cette icône n'apparaît pas dans la barre des instruments, reportez-vous à la méthode décrite page 124 pour la voir.



Pour tracer le segment, il vous suffit de placer le curseur de la souris au point de départ souhaité (matérialisé sur le dessin à gauche par une croix en pointillés) de cliquer sur le bouton droit puis de faire glisser la souris tout en maintenant le bouton appuyé. La droite sera tracée avec les attributs par défaut (couleur et type de ligne)

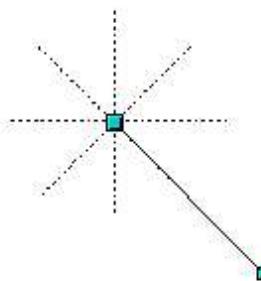
Illustration 15 - Tracé d'un segment



Relâchez le bouton de la souris pour terminer le segment. Deux poignées vertes ou bleues situées à chaque extrémité du segment indiquent qu'il s'agit de l'objet actuellement sélectionné.

La couleur des poignées dépend du mode de sélection par défaut (elles seront vertes pour la sélection simple et bleue pour le mode d'édition de points). Reportez-vous à la page 133 pour plus d'informations.

Illustration 16 - Segment sélectionné



Pendant le tracé du segment si vous maintenez en plus la touche [Maj], vous allez contraindre la droite à être dessinée avec un angle qui sera un multiple de 45° par rapport à l'horizontale.

Dans le schéma ci-contre, l'étoile en pointillés matérialise les angles autorisés pour la droite lorsque la touche [Maj] est enfoncee.

Illustration 17 - Tracé de segment avec [Maj]

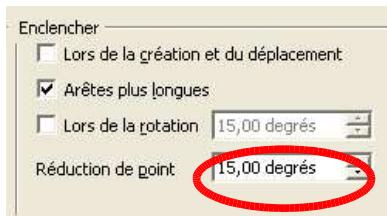
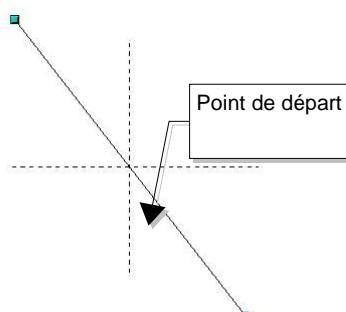


Illustration 18 - réglage de l'angle de rotation

Si dans l'exemple ci-dessus vous remplacez la touche [Maj] par la touche [Control], l'angle de contrainte sera de 15° . Vous pouvez régler cette valeur via la boîte des options, section 'Dessin', sous-section 'Grille'.



Lors du tracé, si vous utilisez la touche [Alt], le segment sera tracé symétriquement par rapport au point de départ. Cette technique permet de tracer des droites en commençant par leur milieu.

Vous pouvez cumuler les effets de la touche [Alt] avec ceux des touches [Maj] ou [Control].

Illustration 19 - Tracé de segment avec [Alt]

Il existe d'autres techniques d'aide au tracé. Elles seront décrites à partir de la page 203.

Tracé d'un rectangle

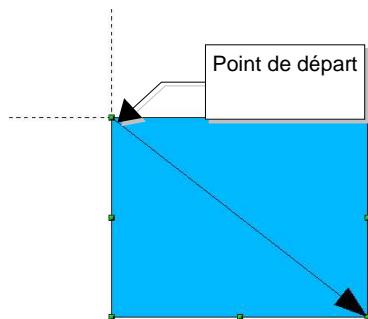


Illustration 20 - Tracé d'un rectangle

La technique de tracé de rectangle est similaire au tracé de segments de droite, l'icône  de la barre des instruments devra être utilisée. Cette icône est située au même niveau que celle qui sert à dessiner des droites. La droite tracée avec la souris matérialise la diagonale du rectangle.

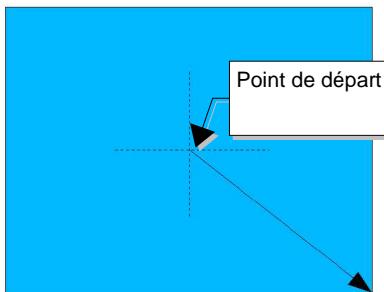
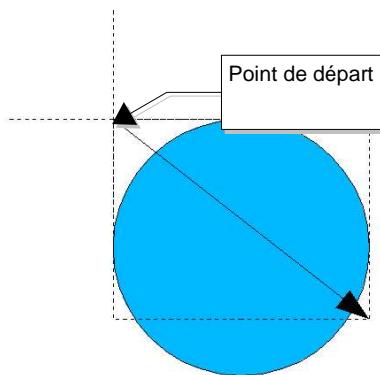


Illustration 21 - Tracé d'un rectangle avec [Alt]

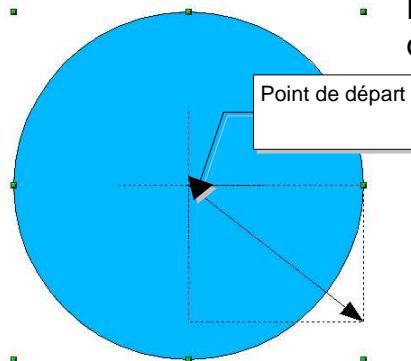
Lors du tracé du rectangle, si vous maintenez la touche [Maj] appuyée, vous obtiendrez un carré. De la même façon, la touche [Alt] permet de tracer le rectangle en commençant par son centre comme le montre le dessin à gauche.

Tracé d'un cercle



Pour tracer un cercle, il faut utiliser l'icône . Le cercle va être délimité par le rectangle associé à la diagonale que l'on trace avec la souris. Le diamètre du cercle correspondra au plus grand côté du rectangle.

Illustration 22 - Tracé d'un cercle



- L'autre méthode importante de tracé de cercle consiste à utiliser la touche [Alt]. Dans ce cas, le point de départ est le *centre* du cercle comme le montre le schéma à gauche.

Illustration 23 - Tracé de cercle à partir du centre

Dans ce chapitre, nous aborderons les outils et les fonctions permettant de modifier des dessins existants.

Sélectionner des objets

Toutes les fonctions de modification que nous allons aborder s'appliquent sur l'objet ou le groupe d'objets sélectionnés. L'objet sélectionné se distingue par la présence de petits carrés ou cercles de couleurs situés autour du cadre de cet objet (ces propos sont aussi valables lorsque plusieurs objets sont sélectionnés simultanément). Dans le reste du document, j'appellerai ces points des .

Le cadre d'un objet correspond au plus petit rectangle pouvant contenir l'objet dans son intégralité. Dans le cas de plusieurs objets le cadre correspond au plus petit rectangle pouvant contenir tous les objets. Ce cadre s'appelle en général le rectangle de sélection.

Si la barre des options est affichée (Cf page 126), il est possible de changer la taille des poignées en utilisant les deux icônes: 'Poignées simples' qui affiche des poignées sans effet de relief et 'Grandes poignées' qui affiche des poignées de plus grande taille. Vous pouvez combiner les effets obtenus par appui sur ces deux icônes. Il est ainsi possible d'avoir des poignées de grande taille avec relief ou des poignées de petite taille sans relief.

Il existe trois grands types de sélection:

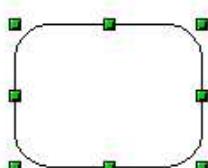


Illustration 24 -
Sélection simple

Les sélections pour déplacement ou changement de taille matérialisées par des petits carrés verts. C'est le mode de sélection des objets par défaut lorsque l'icône du mode édition par points n'est pas enfoncée.

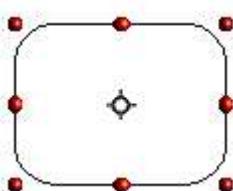


Illustration 25 - Sélection
pour rotation

Les sélections pour rotation matérialisées par de petits ronds rouges et d'un symbole représentant le point de rotation.

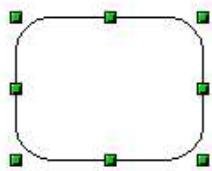


Illustration 24 -
Sélection simple

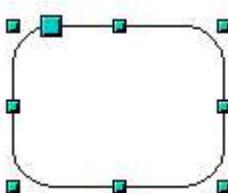


Illustration 26 - Sélection
pour édition de points

Les sélections pour déplacement ou changement de taille matérialisées par des petits carrés verts. C'est le mode de sélection des objets par défaut lorsque l'icône du mode édition par points n'est pas enfoncée.

Les sélections pour le mode édition de points qui sont caractérisées par des carrés bleus. Une poignée supplémentaire, de plus grande taille, apparaît sur la plupart des figures. Lorsque l'icône du mode édition par points est enfoncée, c'est le mode par défaut.

Le mode par défaut de sélection est défini par l'enfoncement ou non de l'icône .

Pour passer d'un mode de sélection à un autre, il faut utiliser les techniques suivantes:

L'icône fait passer du mode sélection simple au mode sélection pour édition des points. Vous pouvez aussi utiliser le raccourci clavier [F8]¹ (Éditer des points).

L'icône fait passer en mode rotation.

Si l'icône de la barre des options (Cf page 126) est enfoncée, on passe du mode sélection normale au mode rotation en double-cliquant sur la figure sélectionnée.

Sélectionner des objets

Pour sélectionner un objet, la méthode la plus simple consiste à cliquer directement dessus. Les poignées de sélection du mode par défaut apparaissent alors.

Il est possible de sélectionner un ou plusieurs objets en traçant avec la souris un grand rectangle autour des objets à sélectionner:

¹ Les raccourcis clavier pouvant être redéfinis par l'utilisateur, je donnerai bien entendu la fonction associée par défaut.

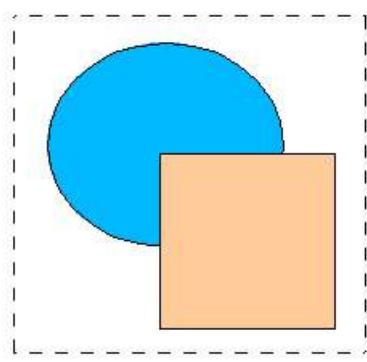


Illustration 27 - Rectangle de sélection

Pour qu'un objet soit sélectionné, il doit être entièrement contenu dans le rectangle.

Lorsque des objets sont situés derrière d'autres objets, ils peuvent néanmoins être sélectionnés. Il faut tout d'abord sélectionner l'objet du dessus de façon classique puis cliquer sur l'objet du dessous (ou sur sa position si on ne le voit pas) en pressant la touche [Alt]. Dans la figure ci-dessous, le carré situé sous le cercle a été sélectionné de cette façon (le cercle a été rendu transparent afin de visualiser le carré):

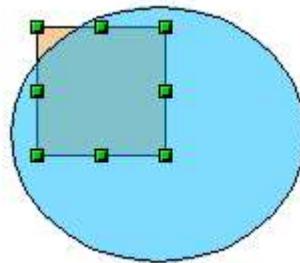


Illustration 28 - Sélection d'objets cachés

PS: Dans ce cas de figure particulier, il aurait bien entendu été possible de cliquer sur l'angle supérieur gauche du cercle.

Lorsqu'un objet est sélectionné, la touche [Tab] permet de sélectionner / désélectionner successivement tous les autres objets de la page par ordre de création. Cette autre méthode peut donc être utilisée pour sélectionner des objets cachés. [Maj] + [Tab] permet de parcourir les objets par ordre inverse.

Pour sélectionner / désélectionner des objets un à un, il suffit de presser la touche [Maj] et de cliquer sur les différents objets à sélectionner ou à désélectionner.

Déplacements et changement dynamiques de la taille d'objet

Il existe plusieurs méthodes pour déplacer ou changer la taille des objets. La méthode décrite ici sera qualifiée de *dynamique* dans la mesure où elle est

réalisée via la souris.

Lors de la modification dynamique de l'objet, n'hésitez pas à regarder la zone gauche de la barre d'état qui affiche des informations précises sur l'opération en cours. Par exemple, pendant une opération de redimensionnement, vous verrez apparaître l'information suivante:

Redimensionner 2 Objets de dessin (x=133% y=155%)

Illustration 29 - Informations dans la barre d'état

Ces informations changent bien entendu lors du déplacement de la souris.

Pour déplacer un objet, il suffit de le sélectionner puis de cliquer à l'intérieur et maintenir le bouton de la souris appuyé tout en la déplaçant. Pour déposer la figure à sa nouvelle position, il suffit de relâcher le bouton de la souris. Pendant le déplacement, la forme de la figure apparaît sous forme de pointillés:

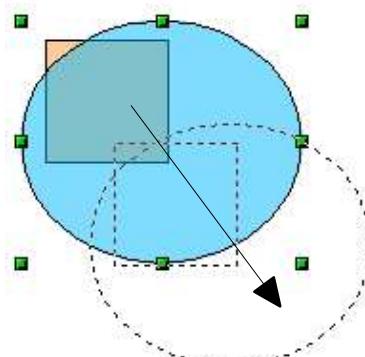


Illustration 30 - Déplacement d'un objet

Pour modifier à la souris la taille d'un objet (ou d'un groupe d'objets sélectionnés), il faut déplacer une des poignées situées sur le pourtour de la sélection. Comme le montre le dessin suivant, la trace du nouvel objet résultant de la modification de taille apparaît sous forme de pointillés.

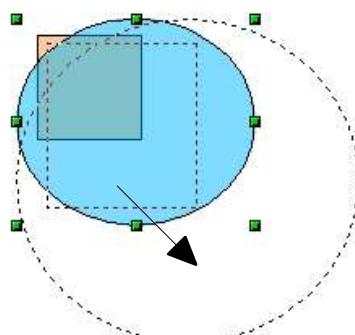


Illustration 31 - Changement de la taille d'un objet

En fonction de la poignée que vous utiliserez, les résultats seront différents. Si vous choisissez les poignées situées dans les coins, vous redimensionnerez les objets sur deux axes à la fois. Si vous utilisez les poignées situées sur les arêtes de la boîte de sélection, les objets ne seront dimensionnés que sur un

seul axe.

Note: Si vous utilisez la touche [Maj] en même temps que vous effectuez l'option de redimensionnement, la modification de taille sera effectuée symétriquement sur les deux axes ce qui permettra de conserver le rapport largeur / hauteur de l'objet. Avec la touche [Alt] l'objet est redimensionné symétriquement à partir de son centre.

Rotation d'un objet

La méthode permettant de passer en mode rotation d'objet est décrite à la page 134.

L'opération de rotation permet aussi d'incliner un objet sur un axe. Pour la mettre en œuvre dynamiquement, il faut, tout comme pour l'opération de changement de taille, utiliser les poignées rouges.

L'opération de rotation fonctionne d'une façon légèrement différente pour les objets 3D (elle agit dans un espace tridimensionnel et pas dans un plan). Je décrirai son fonctionnement dans ce cas particulier dans la chapitre consacré à la gestion des objets 3D (page 211).

Pour faire tourner un objet (ou un groupe d'objets), il vous suffit de faire glisser avec la souris les points rouges de la sélection situés sur les coins du rectangle de sélection. Le curseur de la souris prend la forme d'un arc de cercle avec deux flèches aux extrémités. Comme dans tous les cas de figure, une ombre en pointillés correspondant à la figure en cours de rotation apparaît et l'angle de rotation courant est dynamiquement affiché dans la barre d'état.

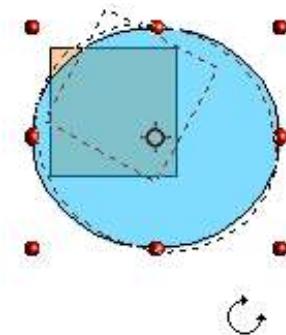
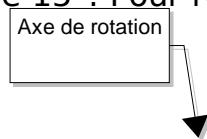


Illustration 32 - Rotation d'un objet

Les opérations de rotations sont effectuées autour d'un axe matérialisé par un petit symbole. Vous pouvez déplacer ce symbole avec la souris pour modifier cet axe de rotation comme le montre le schéma suivant:

Lors de l'opération de rotation, si vous maintenez la touche [Maj] enfoncée, la rotation s'effectuera par pas de 15°. Pour régler cette valeur, reportez-vous à la page 130.



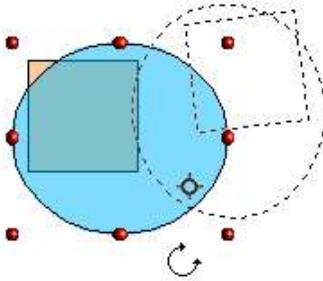


Illustration 33 - Déplacement de l'axe de rotation

Pour incliner un objet, il suffit d'utiliser les poignées rouges situées sur les arêtes du rectangle de sélection. L'axe de l'inclinaison est alors la poignée située sur l'arête opposée:

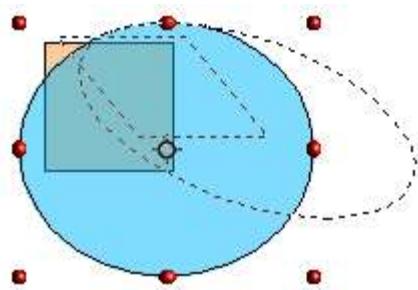


Illustration 34 - Inclinaison d'un objet

L'opération d'inclinaison fonctionnant de façon similaire à l'opération de rotation, vous pourrez contraindre l'inclinaison à s'effectuer par pas de 15° en pressant la touche [Maj] pendant le déplacement de la poignée.

Édition des points d'un objet

Draw propose un ensemble complet d'outils permettant d'éditer précisément le contour d'un objet. Comme nous le verrons, les fonctions d'édition de points fonctionnent essentiellement sur des courbes. Pour pouvoir bénéficier pleinement de ces outils, il vous faudra donc convertir vos objets en courbes comme nous le verrons plus loin.

Certains objets peuvent être manipulés en mode édition de points sans avoir à convertir l'objet en courbe. On agit alors sur une des caractéristiques de cet objet. Les objets concernés se caractérisent par la présence d'une ou plusieurs poignées supplémentaires. En manipulant cette poignée (de plus grande dimension que les poignées situées sur le pourtour du rectangle de sélection), vous obtiendrez différents effets. Le curseur de la souris prend la forme d'une main repliée lorsqu'il passe sur ces points. Voici la liste exhaustive des figures concernées (le tracé des arcs et segments de cercle ou d'ellipses sera abordé au chapitre 8):

Dans ce cas, on rend les bords plus ou moins arrondis

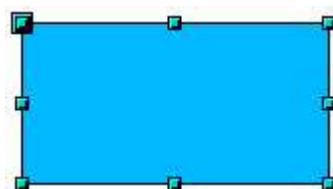


Illustration 35 - Rectangle simple

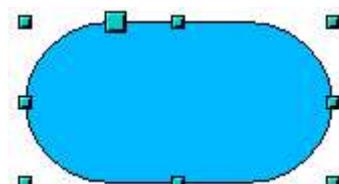


Illustration 36 - Rectangle arrondi

On agit dans ce cas sur l'angle associé. Les arcs de cercles sont munis de deux poignées de contrôle.

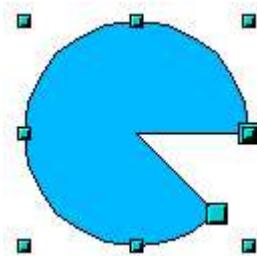


Illustration 37 - Camembert

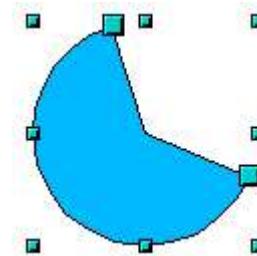


Illustration 38 -
Déplacement des points de
contrôle du camembert

On change ici la position de l'arête du segment.

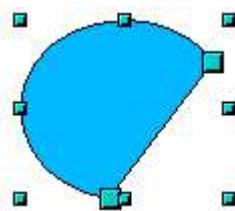


Illustration 39 - Segment
de cercle

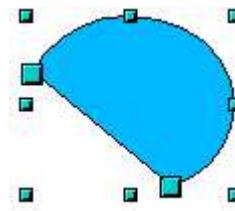


Illustration 40 -
Déplacement des points
de contrôle du segment

L'édition des points des courbes est basée sur une méthode appelée courbes de Bézier¹. L'étude complète des courbes de Bézier sort du cadre de cet ouvrage. Néanmoins, je rappellerai les bases de cette puissante méthode d'édition du pourtour d'un objet.



Illustration 41 - Définition
d'une tangente

L'édition du point d'une courbe de Bézier met en jeu plusieurs mécanismes représentés dans le dessin de gauche: Le point principal sera le point de jonction. Autour de ce point vont partir deux tangentes permettant de manipuler la courbe passant par le point. L'idée de base est qu'autour de ce point de jonction, les deux côtés de la courbe 's'aplatissent' plus ou moins sur la tangente en fonction de la taille et de la position de cette tangente.

¹ Les courbes de Bézier ont été inventées par Pierre Bézier, ingénieur chez Renault qui a développé dans les années 60 la technique qui porte son nom. Cette technologie était destinée à faciliter la modélisation des surfaces des véhicules.

En choisissant précisément le type de tangente et en déplaçant avec la souris le point de jonction et les deux tangentes (via les points ronds situés à leur extrémité), on peut arriver à créer toutes sortes de figures.

Lorsque vous travaillez en mode édition de points, une nouvelle barre d'outils apparaît. Elle se présente sous la forme suivante:



Illustration 42 - Barre d'outils édition de points

Sur cette barre d'outils, certains boutons peuvent être enfouis ou non. Leur comportement est alors différent. Le rôle des trois boutons permettant de choisir le type de tangente sera évoqué au chapitre suivant. Il s'agit des sixième, septième et huitième boutons. La fonction des autres boutons sera explicitée au travers d'exemples d'utilisation.

Trois boutons de la barre d'outils permettent de choisir le type de tangente et de convertir une tangente d'un type en un autre. Un seul de ces boutons peut être enfoui à un instant donné.



Illustration 43 - Définition symétrique

Le bouton permet de travailler avec une tangente symétrique. Tout déplacement de l'une ou l'autre des poignées sera répercuté de façon symétrique sur la deuxième.



Illustration 44 - Tangente dissymétrique

L'icône permet de désolidariser la longueur des deux parties d'une tangente. Dans le dessin ci-contre, on voit nettement que la courbe est plus 'aplatie' du côté de la tangente la plus longue. Ce type de tangente est qualifié de jonction lisse. Cette icône ne devra pas être confondue avec la précédente car leurs dessins sont assez similaires.



Illustration 45 - Point d'inflexion

Il est possible de désolidariser complètement les deux côtés d'une tangente. Dans ce cas, le point central sera qualifié de 'point d'inflexion'. Grâce à cette technique, il sera possible de réaliser des pointes et des creux dans les objets. Utilisez l'icône pour créer un point d'inflexion autour du point sélectionné.

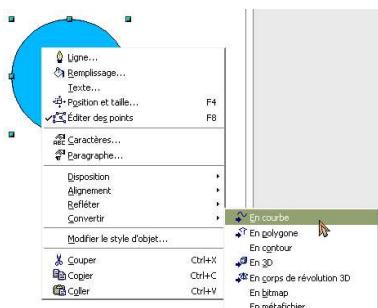


Illustration 46 - Convertir un objet en courbe

Les exemples suivants vont être réalisés à partir d'un cercle rempli. Comme nous l'avons déjà indiqué, pour pouvoir utiliser le mode édition de points, il faut tout d'abord convertir l'objet en une courbe. Cette opération peut être réalisée sur l'objet sélectionné en cliquant avec le bouton droit de la souris et en choisissant l'option 'Convertir / En courbe'.

Vous pourrez remarquer qu'après la conversion les poignées situées sur les coins du rectangle de sélection disparaissent. Ce comportement est logique dans la mesure où les points sur lesquels on agit en mode édition de point sont situés sur le tracé de l'objet.

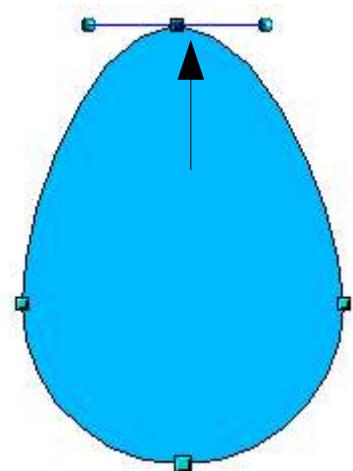
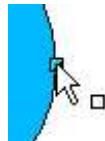


Illustration 47 - Déplacement point de jonction

Le mode déplacement de points est le mode par défaut de l'édition de points. S'il n'est pas actif, on peut le mettre en œuvre en cliquant sur l'icône représentée ici. Lorsqu'il est actif, le curseur de la souris prend la forme suivante lorsqu'il est situé sur un point:



Le déplacement du point de jonction est l'opération la plus simple que l'on puisse réaliser. Le dessin ci-contre représentant un œuf peut être réalisé très simplement à partir d'un cercle en 'tirant' le côté pointu vers le haut.

Pour modifier la position des tangentes, il suffit d'agir sur les poignées circulaires situés aux extrémités. Le curseur de la souris prend alors la forme suivante:

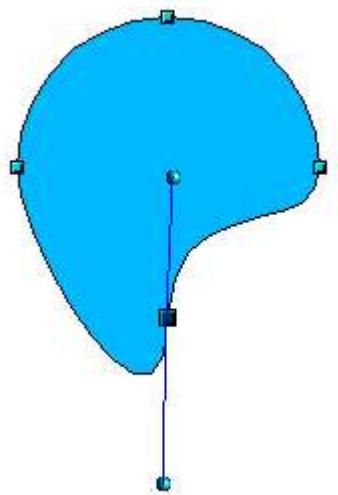
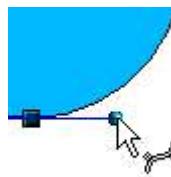
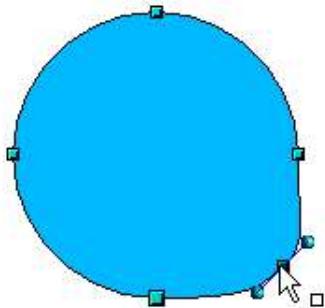


Illustration 48 - Rotation d'une tangente

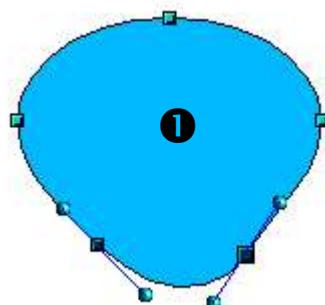




➊ Cette icône permet de rajouter un point sur une courbe existante. Il vous suffit de cliquer avec le bouton gauche de la souris sur la courbe à l'endroit où vous souhaitez que le point soit inséré puis de déplacer la souris en maintenant le bouton appuyé. Si vous vous contentez de cliquer, le point ne sera pas créé.

Les attributs de la tangente associée au nouveau point dépendent des boutons enfouis dans la barre d'outils.

Illustration 49 - Ajout d'un point sur une courbe



➋ Cette icône a le comportement inverse car elle vous permet de supprimer un (ou plusieurs) point(s) d'une courbe. La courbe résultante se 'tendra' automatiquement autour des points restants à l'issue de la coupure.

Tout d'abord, sélectionnez le point à supprimer.

Vous pouvez sélectionner plusieurs points avec la touche [Maj] (➊).

Cliquez ensuite sur la touche ⏺. Les points sélectionnés disparaissent de la courbe qui se reforme autour des points restants (➋).

Note: Vous pouvez aussi supprimer les points sélectionnés en utilisant la touche [Suppr].

Illustration 50 - Suppression de points d'une courbe

➊

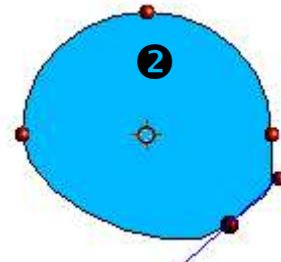
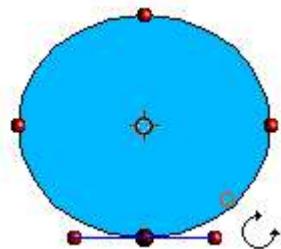


Illustration 51 - Déplacement d'un point sur une courbe

L'icône de rotation des objets que nous avons décrite page 137 peut être utilisée en mode édition de points. Elle sert dans ce cas à déplacer un point le long du contour de l'objet.

Passez en mode rotation en cliquant sur l'icône . Notez qu'en mode rotation, tous les points des tangentes deviennent des points rouges.

Sélectionnez ensuite le point à déplacer puis glissez-le sur le contour tout en gardant le bouton gauche de la souris enfoncé (❶). Lorsque vous relâcherez la souris, le point sera déplacé à la nouvelle position (❷).

Si, pendant l'opération de rotation, vous déplacez une des poignées situées à l'extrémité des tangentes, vous ferez tourner la figure exactement comme lors de l'opération de rotation classique.

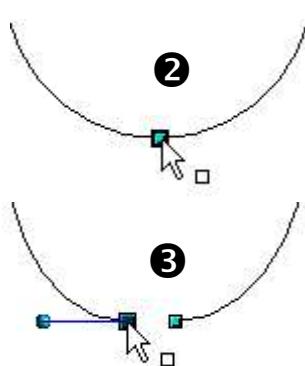
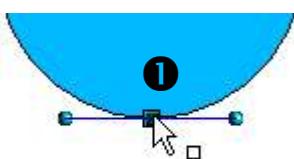
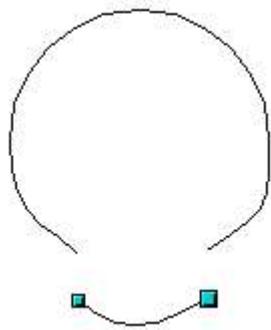


Illustration 52 - Technique de coupe d'une courbe

Cette icône vous permet de couper une courbe à la hauteur de la poignée sélectionnée. Si la figure était remplie, elle va se vider dans la mesure où la courbe qui la délimitait n'est plus fermée.

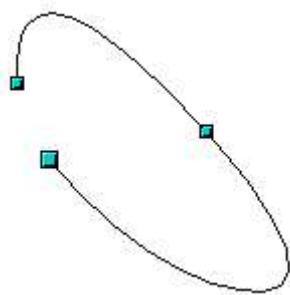
Vérifiez que la poignée est bien sélectionnée (❶), Appuyez sur l'icône (❷) et notez que la figure n'est plus remplie. Vous pouvez vérifier ensuite en déplaçant le point que la courbe est bien scindée (❸).

Note: dans le cas d'une courbe ouverte, le point de départ de la courbe est plus gros que les autres.



Il est possible de couper une courbe en plusieurs endroits simultanément. Il suffit de garder la touche [Maj] enfoncee et de selectionner tous les points sur lesquels la coupure doit être effectuée.

Illustration 53 - Déplacement d'un segment



L'icône  permet de fermer une courbe existante. Sélectionnez une courbe ouverte et cliquez sur cette icône pour la fermer.

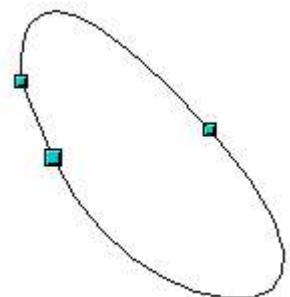
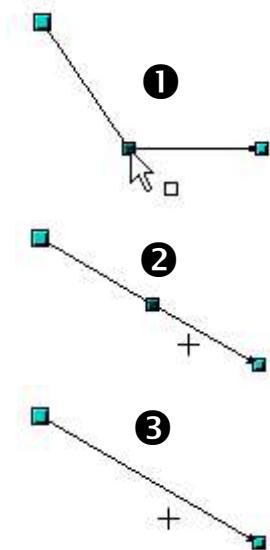


Illustration 54 - Fermeture d'une courbe ouverte



L'icône 'réduire les points' fonctionne comme une bascule. Son principe de fonctionnement est le suivant:

Si vous avez deux points reliés entre eux par une droite (il ne faut pas avoir une courbe entre les deux points) et que vous insérez un point entre les deux, vous vous trouvez dans la situation ①.

Si vous déplacez le point inséré, pour le remettre à peu près sur la position de la droite initiale, si la fonction 'réduire le points' n'est pas activée, vous vous retrouvez dans la situation ②.

Si la fonction est activée, le point est aussitôt effacé (situation ③).

Illustration 55 - Utilisation de la fonction 'réduire les points'

Changement des attributs des objets

A chaque objet est associé un ensemble d'attributs caractérisant l'apparence de cet objet. Ces attributs, tels que la couleur, le type de pourtour, l'épaisseur des traits... sont modifiables à tout instant.

Une barre d'outils et une boîte de dialogue sont dédiées à la modification de ces attributs. Les attributs principaux sont directement modifiables à partir de la barre des objets (située par défaut au-dessus de la zone de travail). En fonction de l'objet actuellement sélectionné, la barre des objets peut prendre plusieurs apparences.

La barre des objets classiques concerne tous les types d'objets:



Illustration 56 - Barre des objets

Dans le cas de l'édition des textes, une nouvelle barre d'outils apparaît:



Illustration 57 - Barre d'édition des objets texte



Dans ce dernier cas, la barre d'outil se décompose en deux parties. Pour passer d'une partie à une autre, utilisez le bouton muni d'une flèche à l'extrémité droite de la barre. La deuxième partie de la barre d'outils étant dans ce cas la barre d'édition des objets texte décrite au début du chapitre.

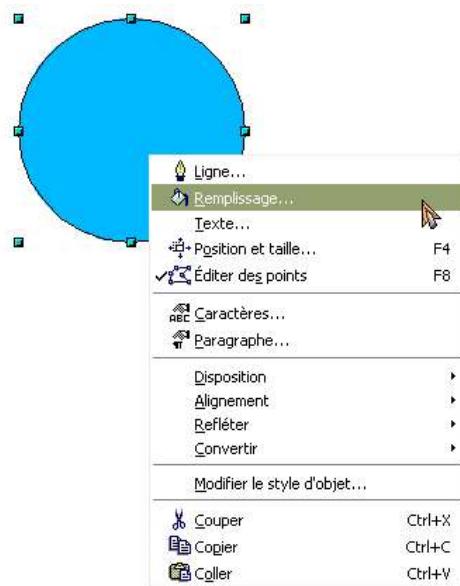


Illustration 58 - Menu local d'un objet

Lorsqu'un objet est sélectionné, un clic dans l'objet avec le bouton droit de la souris affiche un menu se présentant sous la forme suivante:

Ce menu constitue une autre méthode pour accéder aux fonctions de modifications des attributs.

Les cinq premières options vont afficher les différentes boîtes de dialogue permettant de modifier l'aspect des objets qui seront décrites dans les pages suivantes.

Les icônes de cette barre vont permettre d'agir sur les propriétés suivantes:



Type de ligne: L'appui sur cet icône va afficher une boîte de dialogue permettant de spécifier les propriétés de la ligne courante. Si vous souhaitez modifier le style des flèches, de la ligne, son épaisseur ou sa couleur, vous pouvez utiliser directement les quatre zones situées en regard de cette icône (Cf plus loin).

Cette boîte est divisée en trois onglets.

Cette première page permet d'éditer les propriétés les plus courantes d'une ligne. Notez que le réglage de la transparence et le paramétrage pointu des extrémités de la ligne ne sont pas présents dans la barre d'outils et ne peuvent être éditées que via cette boîte.

La case à cocher 'synchroniser les extrémités' permet de forcer le même type de flèche aux deux extrémités de la ligne.

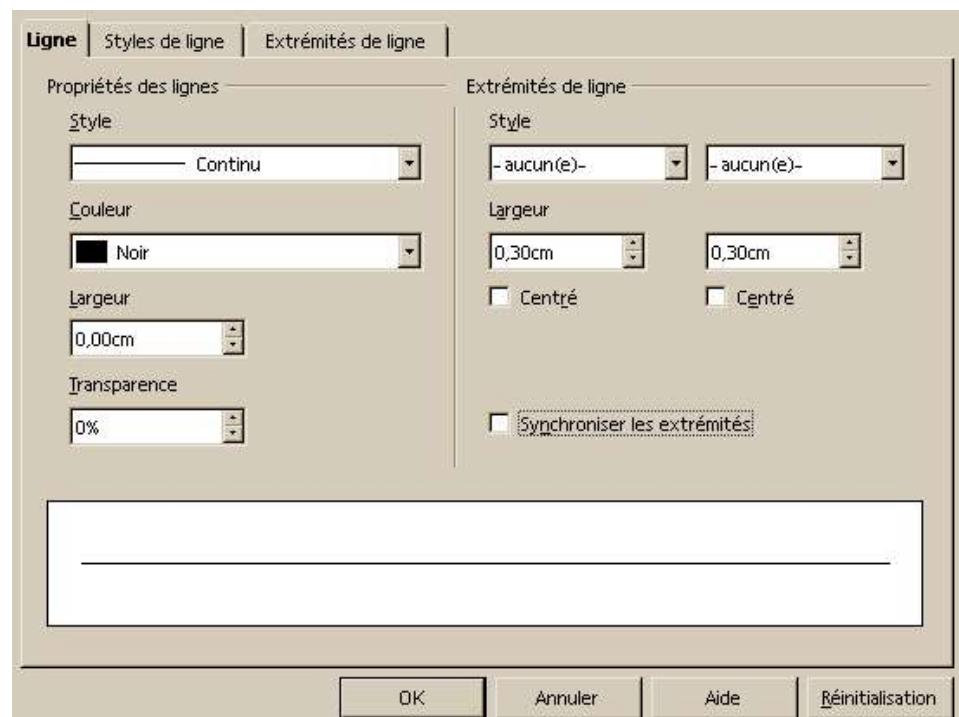


Illustration 59 - Edition des propriétés de lignes

La deuxième partie de la boîte permet de spécifier précisément l'aspect du trait.

Il est possible sur cette boîte de définir l'apparence, le nombre, la taille, la largeur et l'espacement des points ou des traits composant le trait.

Via le bouton 'Ajouter', vous pourrez créer un nouveau type de traits et l'insérer dans la liste de choix. Deux icônes vous permettent d'enregistrer ou de relire un fichier contenant la définition d'un trait (fichier .SOD).

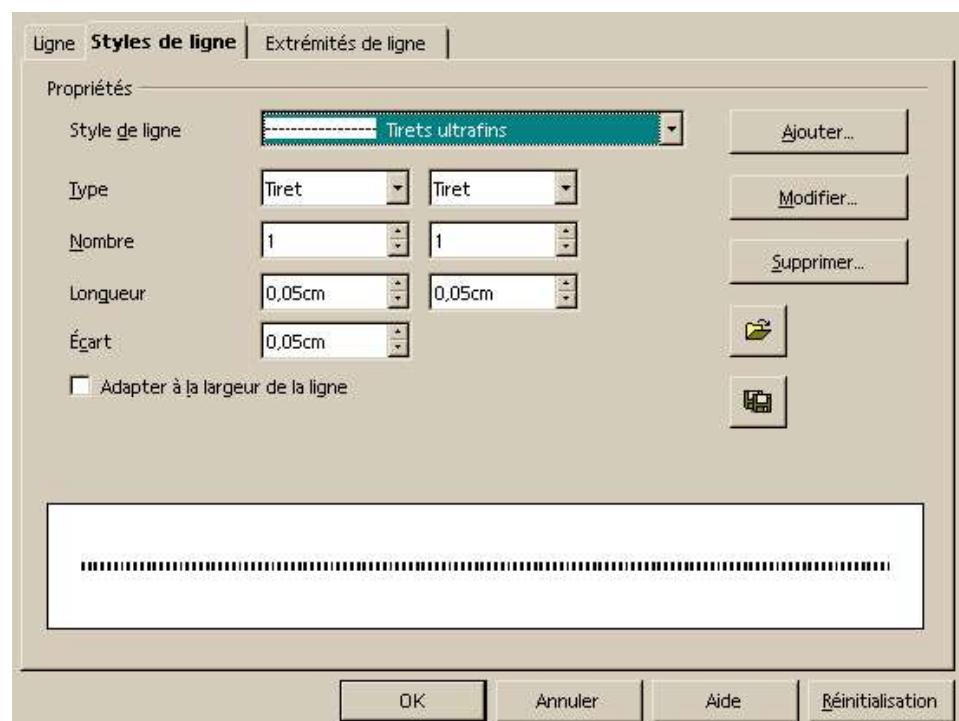


Illustration 60 - Édition du style des lignes

Cette boîte de dialogue est destinée à ajouter un nouveau type d'extrémités de ligne. Comme on peut le constater, il n'existe pas de possibilité de dessiner le type d'extrémités au travers de la boîte de dialogue. L'astuce consiste en fait à utiliser la sélection courante.

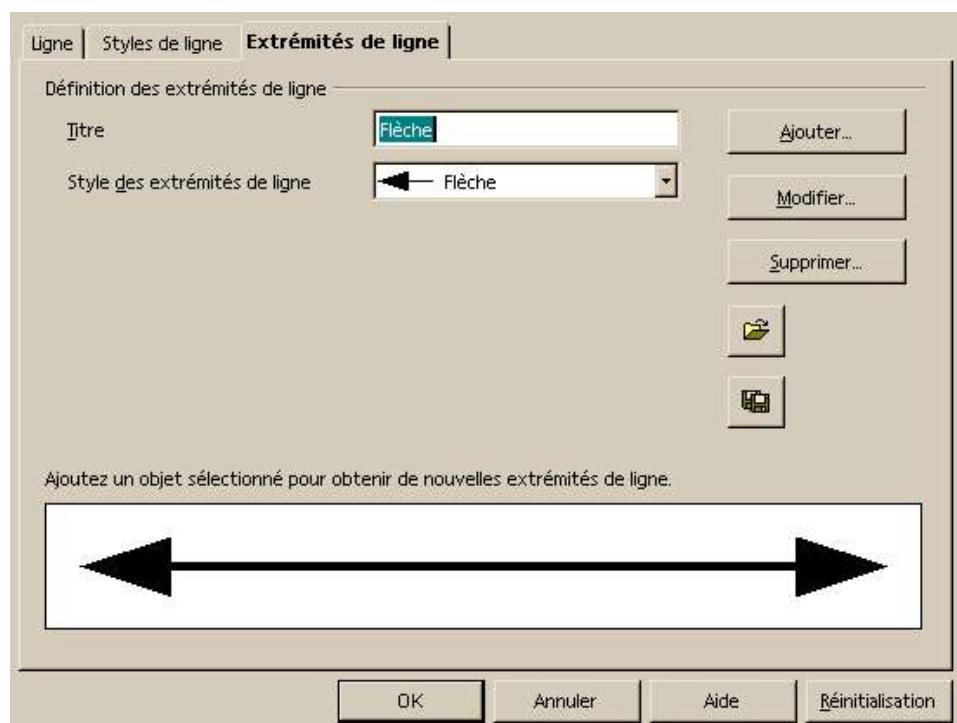


Illustration 61 - Édition de l'extrémité des lignes

L'exemple suivant va vous permettre de comprendre le principe:



Là aussi, vous avez la possibilité d'enregistrer ou de relier un fichier contenant une définition de pointe de ligne (fichier .SOE).

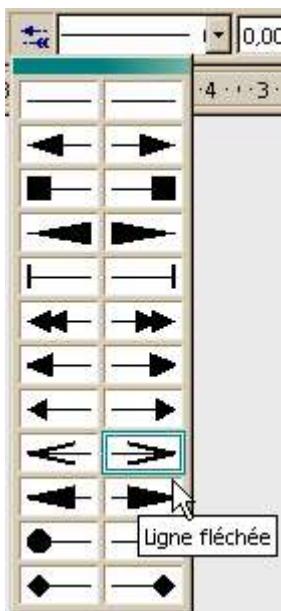


Illustration 62 - Choix de l'extrémité des lignes

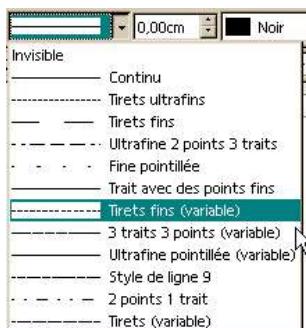


Illustration 63 - Choix du type de ligne

L'icône située à droite de l'icône d'affichage de la boîte de choix du type de ligne permet de sélectionner directement les extrémités des lignes en les sélectionnant dans la fenêtre qui apparaît. Vous pouvez sélectionner indépendamment les extrémités gauche et droite de la ligne.

Note 1: Si vous avez besoin de recourir fréquemment à cette fonction, n'oubliez pas que vous pouvez détacher la fenêtre en cliquant et en la déplaçant (via sa barre de titre). Elle restera ainsi à demeure sur l'espace de travail.

Note 2: Si vous avez ajouté une nouvelle extrémité de ligne, elle apparaîtra dans la liste.

La liste déroulante placée à droite de l'icône décrite au paragraphe précédent permet de sélectionner directement le type de ligne sans qu'il soit nécessaire d'ouvrir la boîte de choix du type de ligne.

Note: Si vous avez ajouté un type de ligne, il apparaîtra dans la liste déroulante.



Illustration 64 - Épaisseur de la ligne

La zone suivante permet de choisir l'épaisseur de la ligne exprimée dans l'unité courante. Vous pouvez la taper dans la zone de saisie ou utiliser les flèches pour la modifier. Une épaisseur de 0,00 cm correspond à un trait très fin.



Illustration 65 - Choix de la couleur de la ligne

La liste déroulante située immédiatement à droite permet de sélectionner la couleur du tracé. Les différentes couleurs de la palette courante sont affichées. Pour changer la palette courante, reportez-vous à la page 124.



Cette icône va afficher la boîte de sélection du mode de remplissage de la figure. Cette boîte, très complète, est divisée en onglets. Nous allons les décrire dans les pages suivantes.

Le premier onglet permet d'effectuer une sélection rapide du type de remplissage alors que les autres onglets permettent de modifier, d'effacer ou d'ajouter un type de remplissage donné. Il existe quatre types de remplissage:

- Une couleur simple
- Un dégradé de couleurs
- Des hachures
- Un motif généré à partir d'une image bitmap

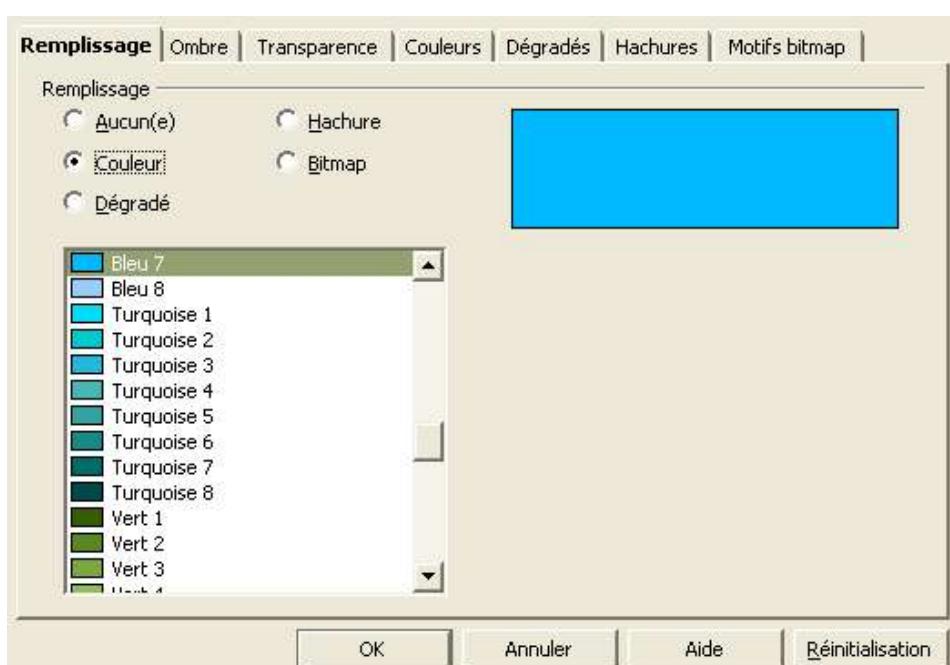


Illustration 66 - Choix rapide du remplissage

Il est extrêmement simple de créer une ombre au-dessous des objets. L'ombre permettant de simuler le 'flottement' de l'objet au-dessus de la zone de dessin.

Le dessin de l'ombre épousera précisément le contour de la figure sélectionnée. Cette boîte vous permet d'indiquer la couleur, la distance par rapport à la figure principale, la position et la transparence de l'ombre.

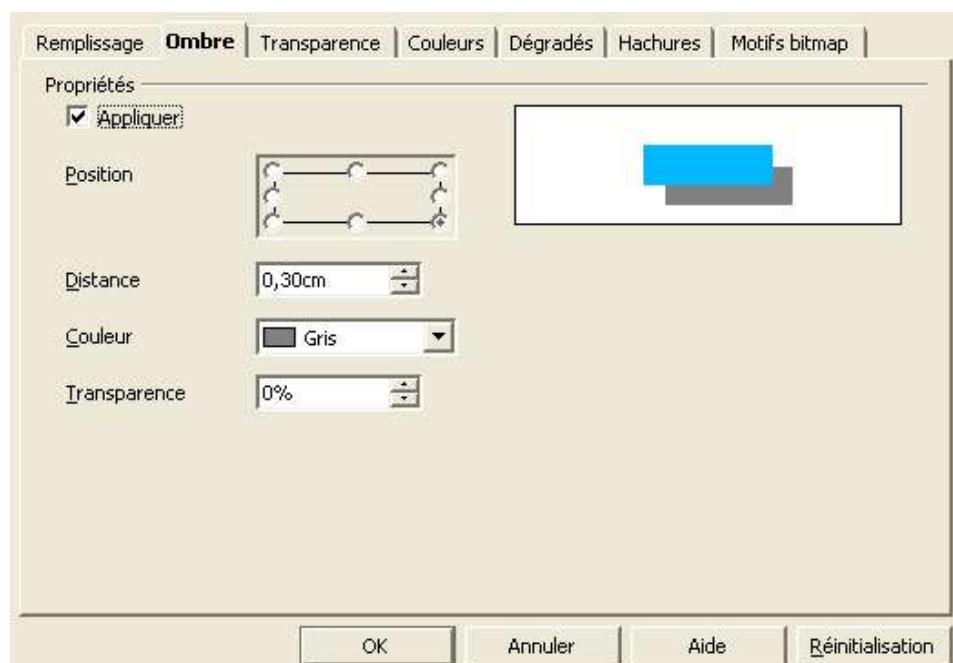


Illustration 67 - Édition de l'ombre

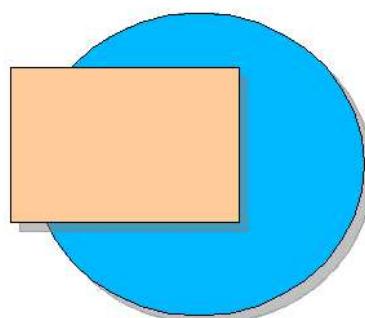


Illustration 68 - Exemple d'ombre

Les figures élémentaires du dessin ci-dessus comportent toutes les deux une ombre grise avec 50% de transparence (le cercle apparaissant par transparence sous l'ombre du rectangle).

OpenOffice.org permet de régler avec une grande précision la transparence des objets. Il est par exemple possible d'indiquer que la transparence n'est pas uniforme.

Dans tous les cas, le dégradé courant peut être prévisualisé dans la zone située en haut et à droite de la boîte de dialogue.

La case à cocher 'Transparence' permet de définir une transparence uniforme en indiquant son pourcentage. 0% correspondant à une figure opaque et 100% à une couleur totalement transparente.

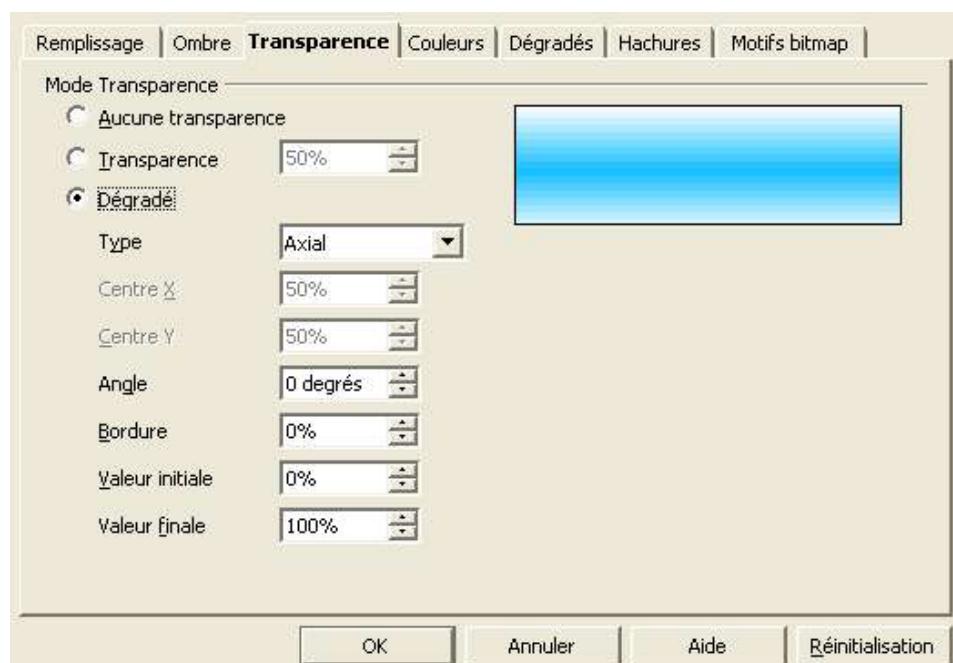


Illustration 69 - Édition de la transparence

La case 'dégradé' va permettre de choisir les paramètres de la transparence.

Il faut tout d'abord choisir le type de transparence: Linéaire, Axial, Radial, Ellipsoïde, Carré ou Rectangulaire.

Les deux zones de saisie 'Centre' servent à indiquer où se trouvera le point central de la transparence par rapport à la figure (0% = gauche ou haut, 100% = droite ou bas).

L'angle permet de choisir l'inclinaison de l'effet de transparence. Sur le dessin ci-dessus, l'angle de transparence est de 45°.

Le pourcentage de bordure indique l'épaisseur de la bordure autour de la figure. Sur le dessin 70, la bordure a été définie à environ 33%. La couleur de la bordure est définie dans la zone 'Valeur finale'.

Les deux zones suivantes servent à indiquer le pourcentage de transparence sur le point de départ et d'arrivée de l'effet. Dans le cas du dessin, la transparence de départ est de 0% (couleur opaque près de la bordure) et la transparence finale est de 100% (totalement transparent – au centre du cercle).

Les illustrations suivantes montrent des exemples de ce que l'on peut obtenir.

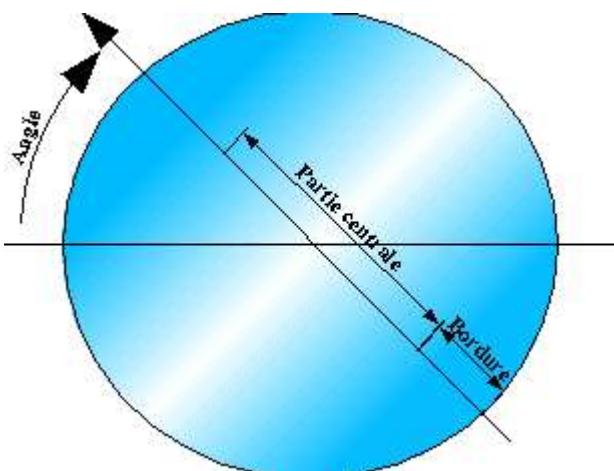


Illustration 70 - Paramètres de la transparence

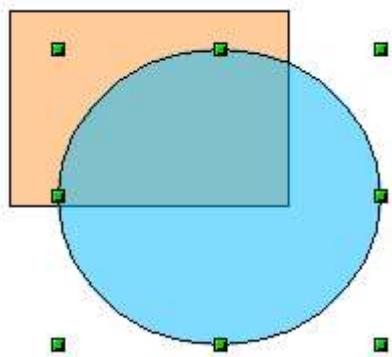


Illustration 71 - Transparence simple à 50%

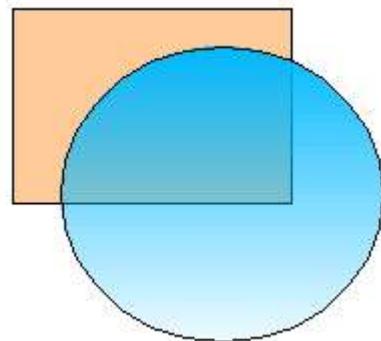


Illustration 72 - Transparence linéaire verticale

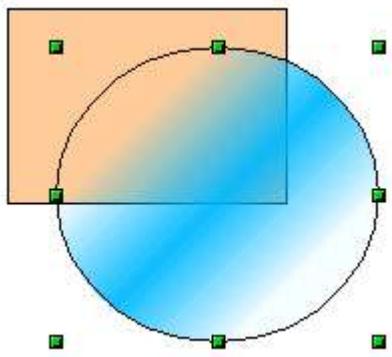


Illustration 73 - Transparence axiale

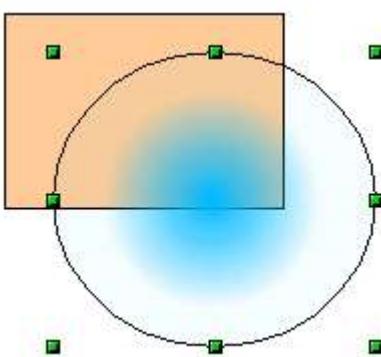


Illustration 74 - Transparence radiale

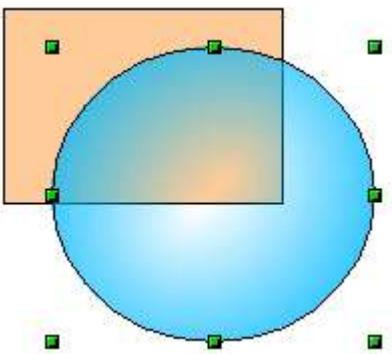


Illustration 75 - Transparence ellipsoïdale

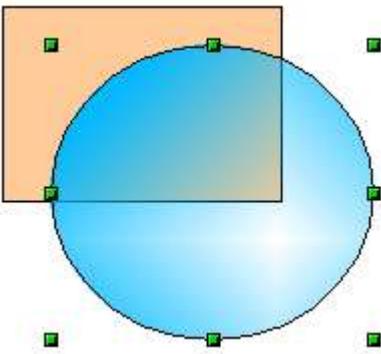


Illustration 76 - Transparence de type carré avec point central déplacé vers le bord inférieur gauche

Il existe une autre méthode pour définir la transparence d'un objet. Elle est décrite à la page 171.

Cette boîte, déjà décrite à la page 124 permet d'éditer le jeu de couleurs disponibles. On peut choisir la couleur du remplissage en double-cliquant sur une des cases colorées.

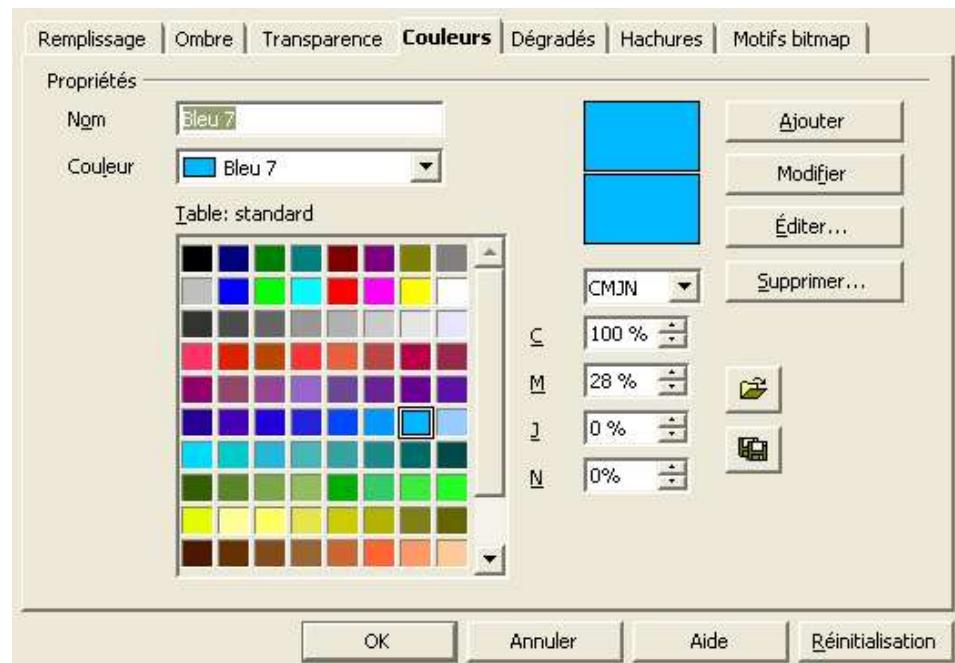


Illustration 77 - Édition des couleurs

Cet onglet contient les outils permettant de définir et de sélectionner un type de dégradé. Vous pouvez sélectionner le dégradé que vous souhaitez appliquer à l'objet sélectionné en le sélectionnant dans la liste et en cliquant sur le bouton 'Ok'.

Plusieurs dégradés sont fournis en standard. Les paramètres des dégradés sont similaires à ceux qui définissent les transparences.

Il est possible de sauver les différents dégradés et de les recharger depuis un fichier disque (fichier SOG). Des exemples de dégradés sont fournis en standard avec OpenOffice.org (moderne.sog et classique.sog par exemple).



Illustration 78 - Édition des dégradés

Cette boîte permet de définir un motif sous forme de hachures. Il vous suffit d'indiquer l'écart entre les lignes (dans l'unité courante), l'angle des hachures, le type de ligne (simple, croisé, triple) et la couleur des hachures.

Comme pour les autres onglets, pour appliquer un motif en hachures aux objets sélectionnés, choisissez-le dans la liste puis cliquez sur 'Ok'.

Vous pouvez sauver et relire les hachures depuis un fichier (extension SOH).

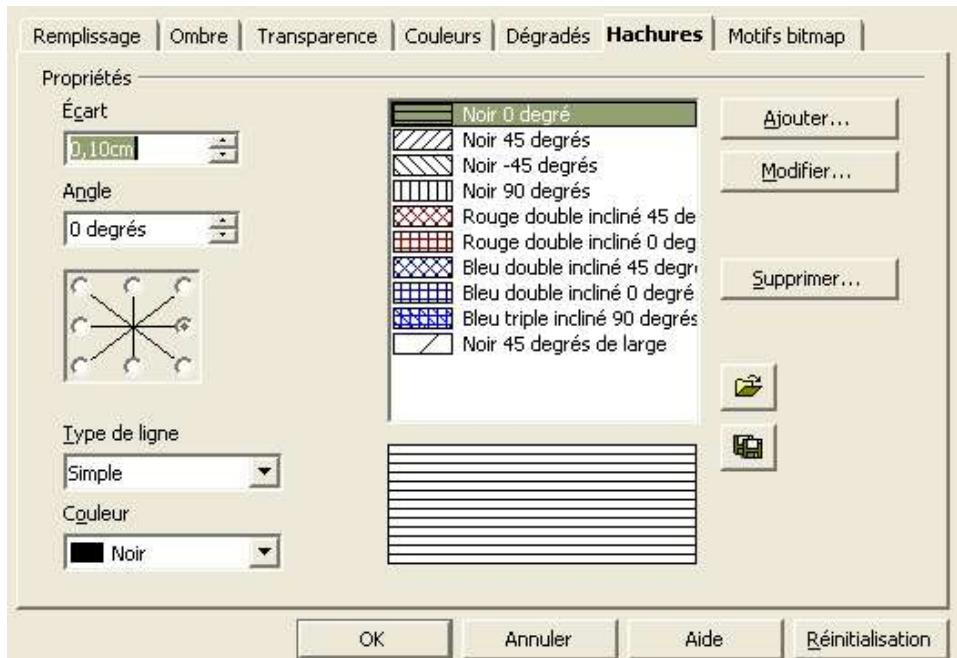


Illustration 79 - Édition des hachures

Cette dernière partie de la boîte permet d'habiller les objets sélectionnés d'un motif obtenu à partir d'une image bitmap. N'importe quel fichier bitmap pourra faire l'affaire pour autant que le format du fichier soit reconnu par OpenOffice.org.

Pour que l'application du motif sur l'image soit de bonne qualité, il importe de disposer d'images bitmaps constituées de motifs répétitifs.

Le bouton 'Importer' vous permet de créer un nouveau motif à partir d'un fichier.

L'éditeur de motif sur la gauche vous permettra de dessiner un motif simple à partir de deux couleurs. Pour dessiner le motif, cliquez dans les différentes cases de la grille en cliquant sur le bouton droit de la souris pour obtenir la couleur de premier plan et avec le bouton gauche pour obtenir la couleur d'arrière-plan.

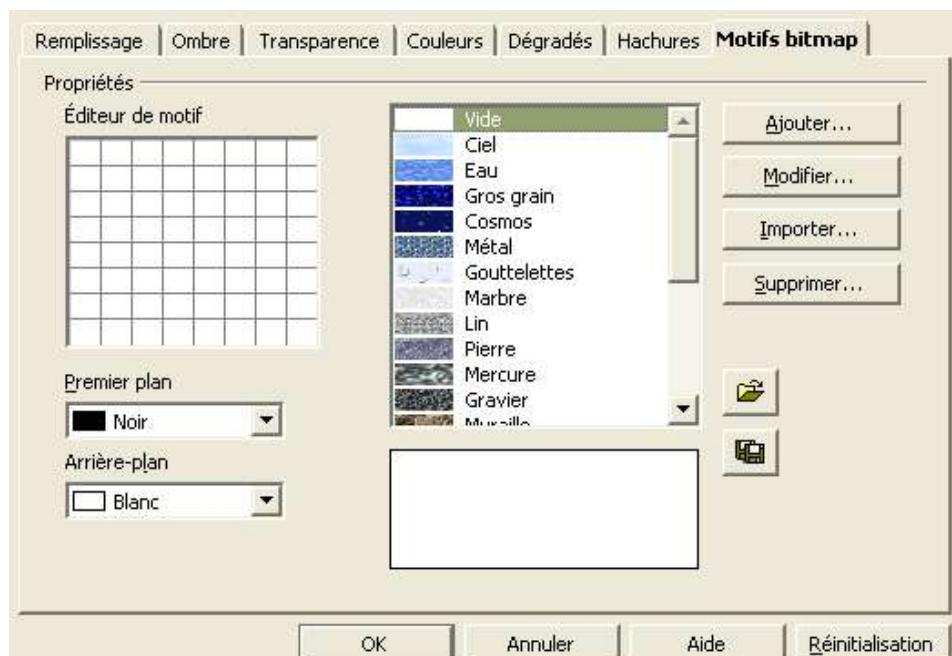


Illustration 80 - Édition de la texture

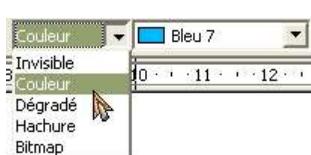


Illustration 81 - Édition rapide de motif

Les deux listes déroulantes situées sur la partie droite de la barre des objets permettent de choisir rapidement un motif sans avoir à passer par la boîte de dialogue que nous venons de décrire. La première liste permet de définir le type de motif (Couleur, Dégradé, Hachure et Bitmap) tandis que la deuxième liste sert à sélectionner le motif lui-même. Le contenu de cette liste dépend du choix qui a été effectué sur la première liste.

Si dans la première liste, vous choisissez 'Invisible', les figures sélectionnées deviendront transparentes.



La dernière icône de la barre d'outils permet de placer ou de supprimer une ombre sur les objets sélectionnés. Elle fonctionne comme une bascule.

La barre d'édition du mode texte

Lorsque l'objet sélectionné est un cadre de texte, la barre d'édition se dédouble pour rajouter un ensemble d'icônes dédiées à la mise en forme de texte. Le triangle bleu sur la gauche permet d'afficher la barre d'outils d'édition des objets. Dans ce cas, l'objet mis en forme est le cadre du texte qui peut subir toutes les transformations et les modifications décrites précédemment.



Illustration 82 - Barre d'édition des objets texte

Cette barre d'outils comporte les fonctions traditionnelles (similaires à celles du module writer) permettant de choisir la police des caractères, l'alignement du paragraphe etc... Elles sont classiques et ne seront pas évoquées plus précisément ici.

Les chapitres précédents ont été consacrés à la modification dynamique des attributs d'un (ou de plusieurs) objet(s). Le qualificatif dynamique signifiant que toutes les modifications sont effectuées à la souris. Cette technique a l'avantage de permettre la visualisation en continu des modifications mais a le désavantage de ne pas être très précise.

Lorsqu'un objet est sélectionné, un appui sur la touche [F4] ('Position et taille') ou l'utilisation du menu accessible via le bouton droit de la souris, entrée 'Position et taille' (ce menu est décrit à la page 146), affiche une boîte divisée en quatre onglets autorisant la saisie précise des informations de taille et de position.

1. Saisie de la position

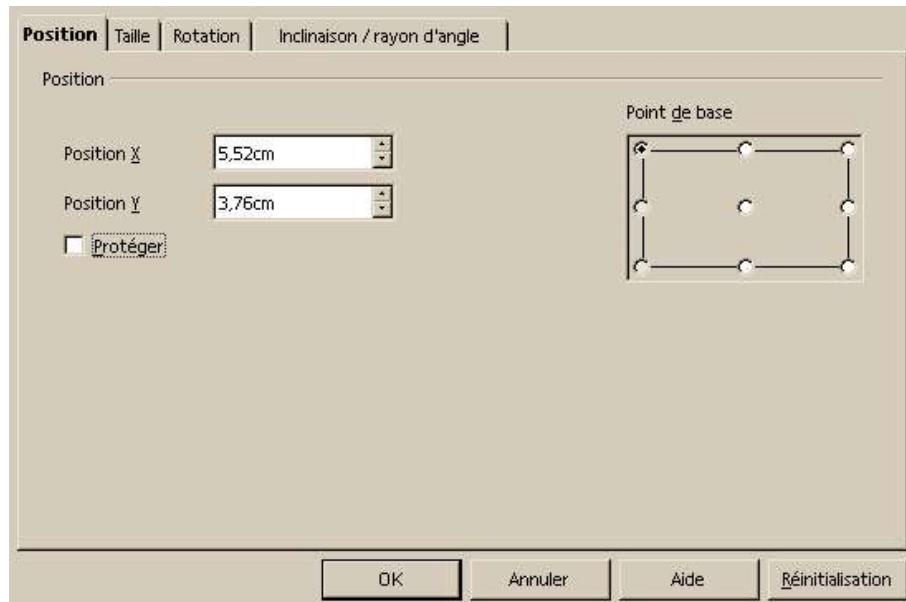


Illustration 83 - Boîte de saisie de la position

Cette première partie de la boîte permet de saisir la position de l'objet sur les axes X et Y dans l'unité courante. On peut aussi indiquer quel est le point de base de l'objet. Par défaut, ce point est situé en haut et à gauche (exemple de la partie gauche de la figure 84). Si vous le déplacez en bas à gauche, la même position (X, Y) correspondra à un objet placé comme indiqué dans la partie droite du dessin.

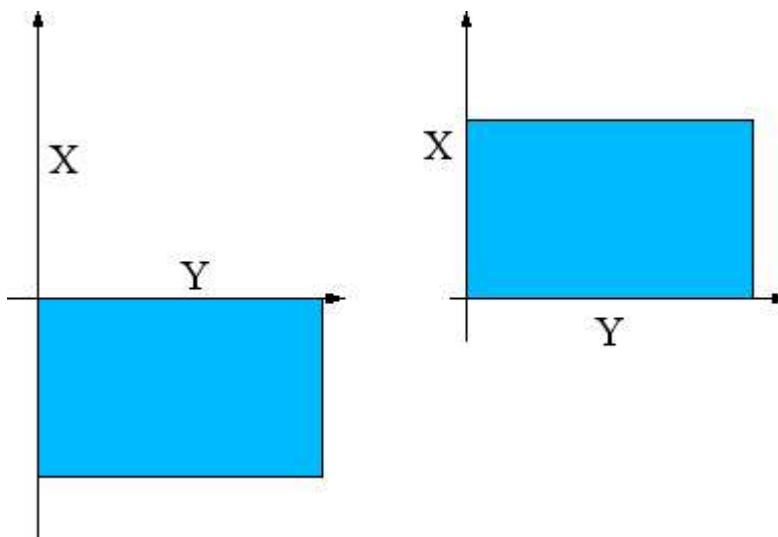


Illustration 84 - Exemple de points de base

Si vous cocher la case 'protéger', l'objet ne sera plus déplaçable. Il faudra décocher la case pour pouvoir le déplacer de nouveau.

2. Saisie de la taille

Cet onglet est destiné à la saisie des dimensions des objets. Si la case 'Proportionnel' est cochée, toute modification sur une des deux dimensions se répercutera symétriquement sur la deuxième, ce qui permettra de conserver le

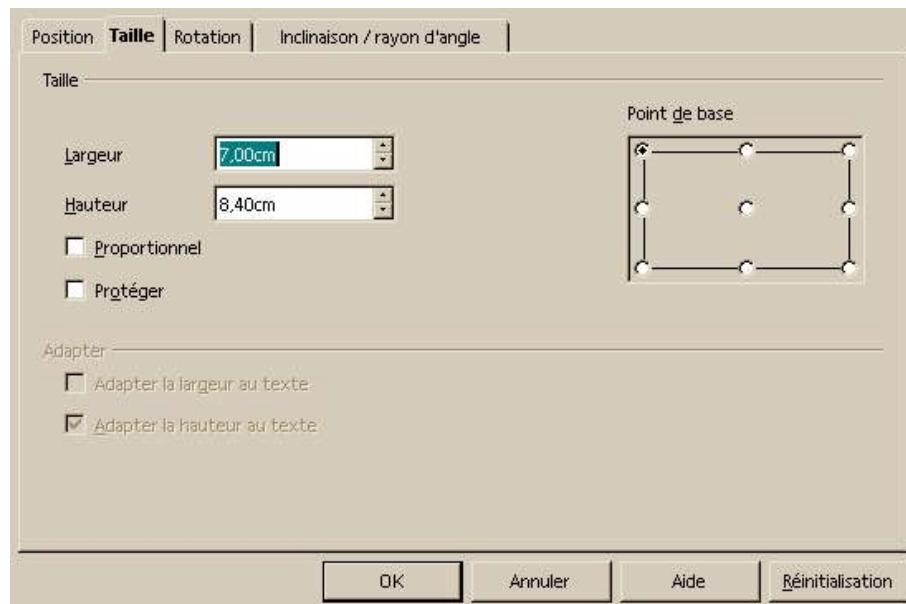


Illustration 85 - Boîte de saisie de la taille

ratio longueur / largeur de l'objet. Si le cadre contient du texte, deux cases supplémentaires sont sélectionnables. Elles permettent d'adapter automatiquement la largeur et la hauteur du cadre au texte.

Comme pour la boîte précédente, vous devrez indiquer par rapport à quel point s'effectuera la modification de taille. Par exemple, si vous cliquez sur le point central, toute modification de taille sera faite de façon symétrique par rapport au centre de l'objet.

3. Saisie de l'angle de rotation

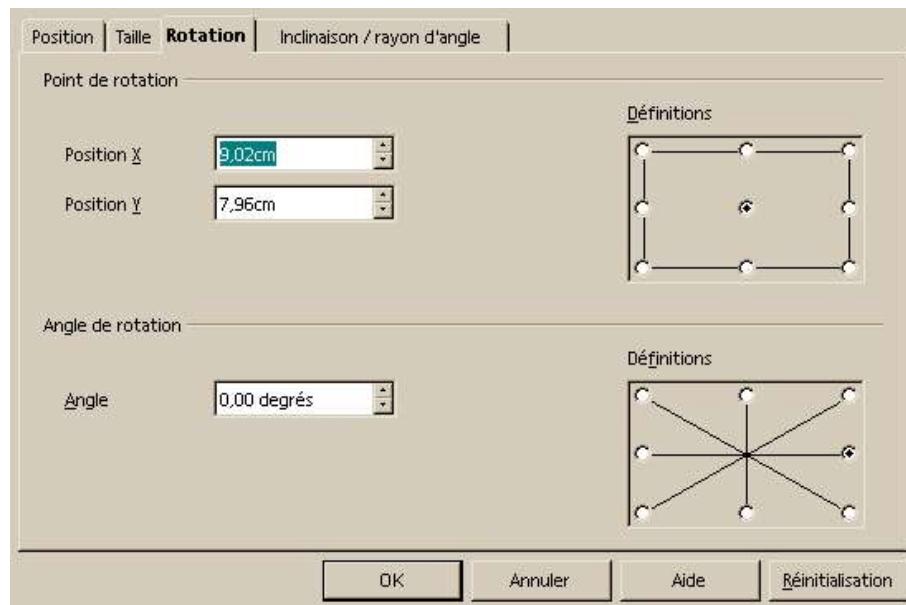


Illustration 86 - Boîte de saisie de l'angle de rotation

Cet onglet permet de définir la position de l'axe de rotation et l'angle de rotation de l'objet. Il faudra définir le point de base à partir duquel est mesuré l'axe de rotation (cf chapitre précédent)

La zone définition permet la saisie rapide de l'angle de rotation par pas de 45°.

4. Saisie de l'inclinaison

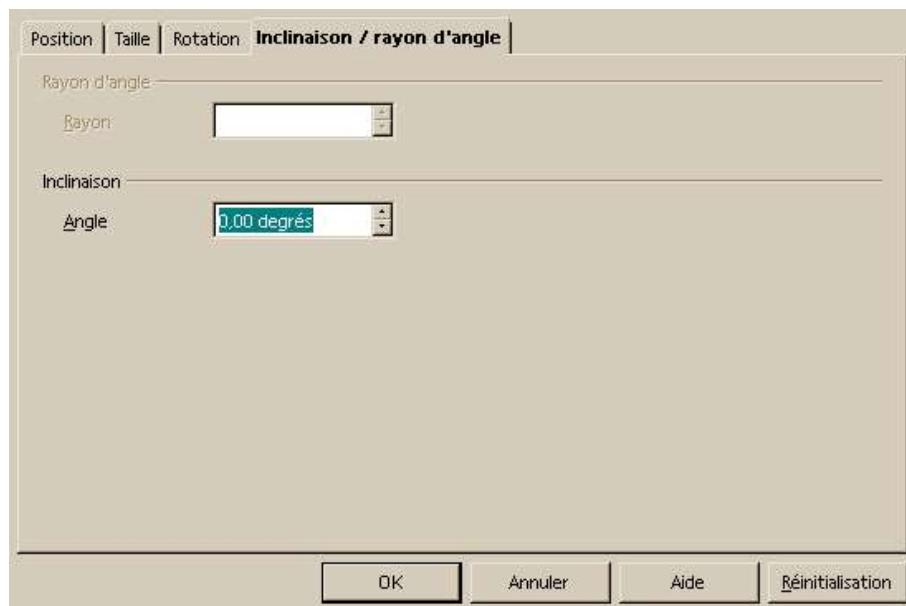


Illustration 87 - Boîte de saisie de l'inclinaison

La dernière partie de la boîte sert à saisir l'angle d'inclinaison de l'objet.

Le styliste

L'application d'un même style à plusieurs objets différents peut se révéler une opération fastidieuse avec les outils de dessin. Heureusement, OpenOffice.org vient à notre secours avec le styliste. Cette fenêtre agit comme avec Writer en permettant de définir des styles particuliers et de les appliquer en une seule opération sur plusieurs objets à la fois. Certaines fonctions du styliste concernant plus particulièrement le logiciel de présentation Impress ne seront pas décrites ici.

Vous pouvez activer ou désactiver le styliste avec l'icône  de la barre d'outils principale. La fenêtre du styliste apparaît. Cette fenêtre pouvant être ancrée comme les autres fenêtres d'OpenOffice (appui maintenu sur la touche [Control] en cliquant sur la barre de titre pour déplacer la fenêtre) se présente sous la forme suivante:

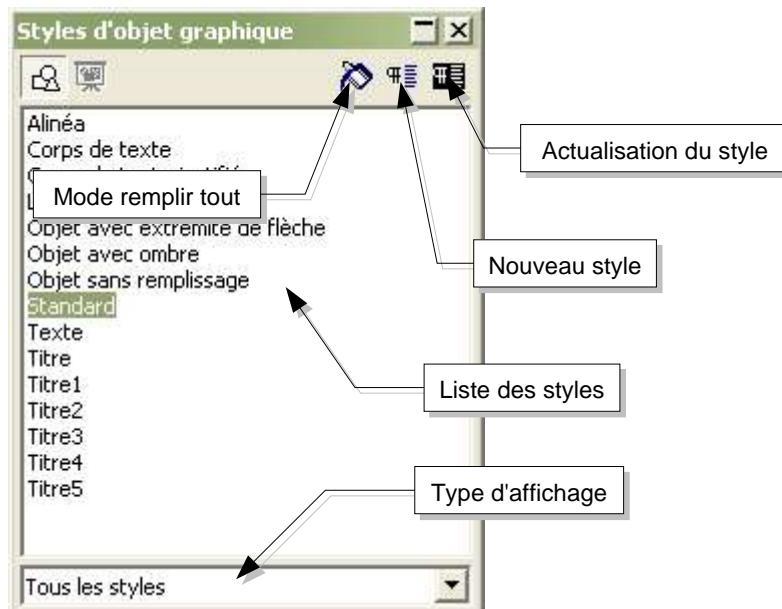


Illustration 88 - Fenêtre du styliste

Pour appliquer un style existant sur un ou plusieurs objets, il suffit de les sélectionner et de double-cliquer sur le style choisi. Vous pouvez aussi utiliser l'icône 'Mode remplir tout'. Lorsque cette icône est enfoncée, le curseur de la souris se transforme en pot de peinture. Il suffit alors de cliquer sur tous les objets sur lesquels vous souhaitez appliquer le style courant.

Dans l'exemple suivant, nous disposons d'un style particulier composé d'une ombre, d'une bordure bleue épaisse et d'un remplissage avec un motif bitmap

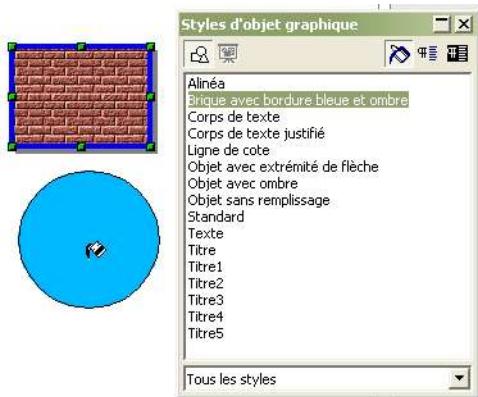


Illustration 89 - Utilisation du styliste

affichant des briques.

Si on enfonce l'icône avec un pot de peinture, il suffit ensuite de cliquer sur le cercle pour voir tous les enrichissements du style s'appliquer aussitôt:



Illustration 90 - Application d'un style

Tous les objets gardent la trace du style que l'on a appliqué. Si on modifie le style de base, tous les objets associés vont être transformés. Pour modifier un style, il existe plusieurs méthodes:

- On peut modifier le style de l'objet directement en utilisant les différentes techniques que nous avons abordées dans les sections précédentes. Il suffit ensuite de cliquer sur l'icône 'Actualisation du style' du styliste.
- On peut aussi modifier directement le style dans le styliste. Il suffit de cliquer sur le style à modifier avec le bouton droit de la souris puis de choisir l'option 'Modifier'.

A titre d'exemple, voici une illustration de la première méthode:

Nous sommes dans la situation de la figure 90 et le cercle est sélectionné.

Nous modifions le style de la texture du cercle dans la liste déroulante de la barre d'outils d'édition des objets.



Illustration 91 - Modification de la texture

②



On voit aussitôt la nouvelle texture s'appliquer sur le cercle. Le rectangle n'est pas touché.

Illustration 92 - Texture du cercle modifiée

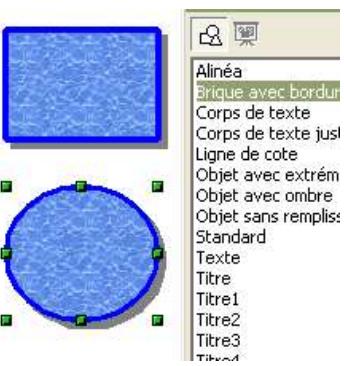
③



On clique ensuite sur l'icône 'Actualisation du style'

Illustration 93 - Actualisation du style

④



On voit alors le nouveau style s'appliquer aussi sur le rectangle.

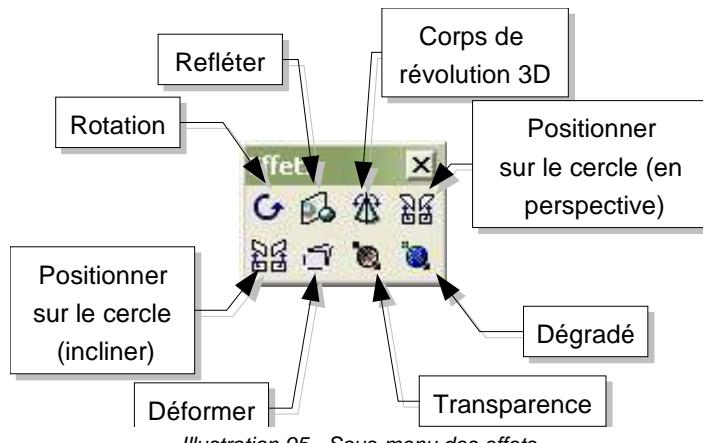
Illustration 94 - Application automatique du style sur le rectangle

Pour créer un nouveau style, deux méthodes sont à votre disposition:

- Cliquez avec le bouton droit dans le styliste et choisissez l'option 'Nouveau' dans le menu. Saisissez toutes les options du style dans la boîte, nommez le style et cliquez sur 'Ok'. Le nouveau style apparaîtra dans le styliste.
- Vous pouvez aussi utiliser une méthode 'par l'exemple' en créant un nouveau style à partir des propriétés d'un objet. Sélectionnez l'objet à partir duquel vous souhaitez créer un style puis cliquez sur l'icône 'Nouveau style à partir de la sélection' du styliste. Une boîte vous invite à saisir le nom du style qui apparaîtra ensuite dans le styliste.

Transformation particulières

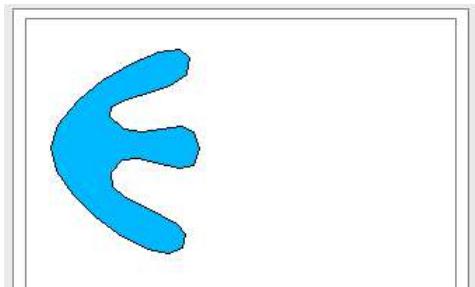
Dans la barre des instruments, l'icône de rotation vous permet d'accéder à un sous-menu de fonctions regroupées sous le libellé 'effets'. À chacune de ces icônes sont associées des fonctions de transformations particulières. La fenêtre du sous-menu se présente sous la forme suivante:



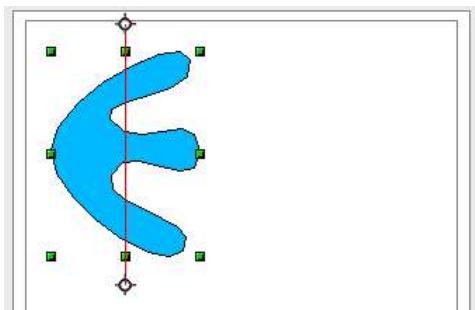
Nous avons abordé cette fonction à la page 137.

Cette fonction va permettre de refléter un objet par rapport à un axe donné dont vous allez pouvoir finement définir les caractéristiques. Voici une illustration en cinq étapes du fonctionnement de cet outil:

- ① Voici l'objet que nous allons refléter



- ② Après avoir cliqué sur l'icône 'refréter' , un axe rouge muni de deux poignées apparaît au centre de l'objet. C'est par rapport à cet axe que le reflet sera obtenu.



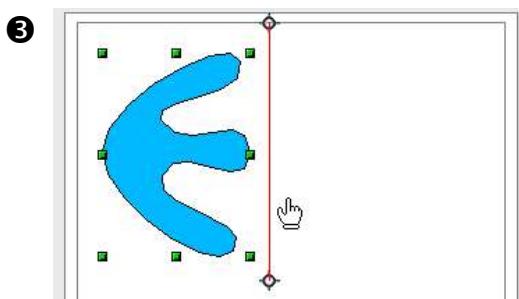


Illustration 98 - Déplacement de l'axe

L'axe peut être déplacé en le faisant glisser. Il est aussi possible de changer son inclinaison en faisant glisser ses deux poignées.

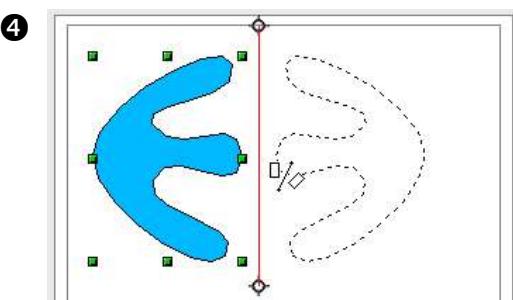


Illustration 99 - Exécution de la réflexion

L'axe de symétrie étant à la position que nous avons choisie, il ne reste plus qu'à effectuer l'opération. Il faut pour cela cliquer avec la souris sur une des poignées de l'objet et, tout en maintenant le bouton de la souris appuyé, la faire glisser sur l'axe rouge. Une image du futur objet se dessine en pointillés de l'autre côté de l'axe.

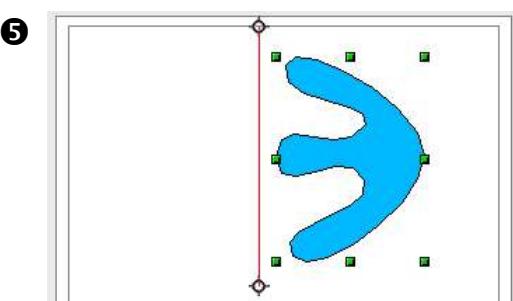


Illustration 100 - Résultat de la réflexion

Il ne reste plus qu'à relâcher le bouton de la souris et l'opération de réflexion est terminée.

Cette fonction va nous faire pénétrer dans le monde des objets tridimensionnels que nous aborderons d'une façon plus complète à partir de la page 211. Cet outil va nous permettre d'obtenir un objet 3D en par rotation d'un profil autour d'un axe. Il est par exemple utile pour obtenir des objets de la vie courante tels que des vases, des verres ou des bouteilles. Voici des exemples de figures 3D obtenues à partir de profils simples:

Notez que le profil peut être ouvert ou fermé. Le fonctionnement de cet outil est similaire à celui de réflexion. On positionne un axe autour duquel l'opération de révolution sera exécutée. Voici un exemple illustré du fonctionnement de cette fonction:

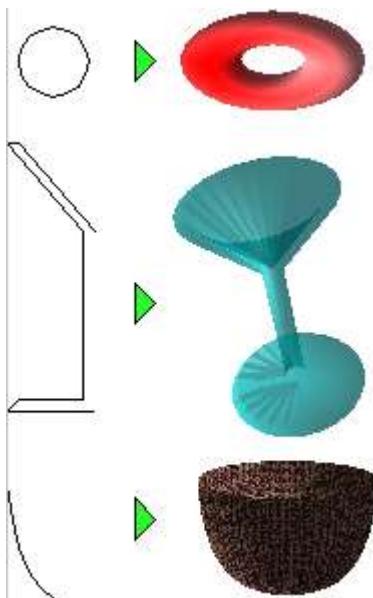


Illustration 101 - Exemples de corps de révolution 3D

❶

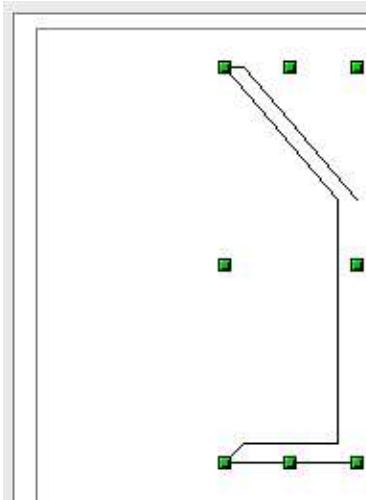


Illustration 102 - Tracé de base pour la révolution 3D

Voici le profil de départ. Il a été réalisé avec l'outil de tracé de lignes polygonales.

❷

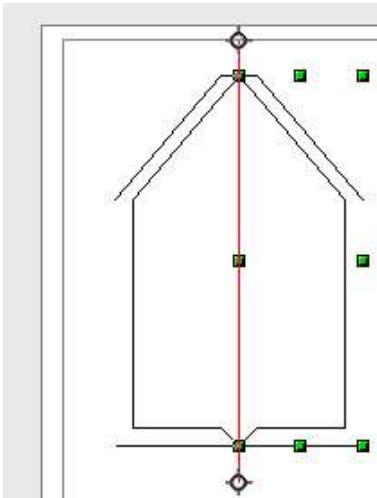


Illustration 103 - Axe de évolution initial

Lorsqu'on clique sur l'icône du mode de révolution 3D , un axe rouge muni de deux poignées aux extrémités s'affiche sur le bord gauche du profil.

Le profil résultant est affiché de l'autre côté de l'axe pour permettre d'avoir une meilleure idée du résultat.

③

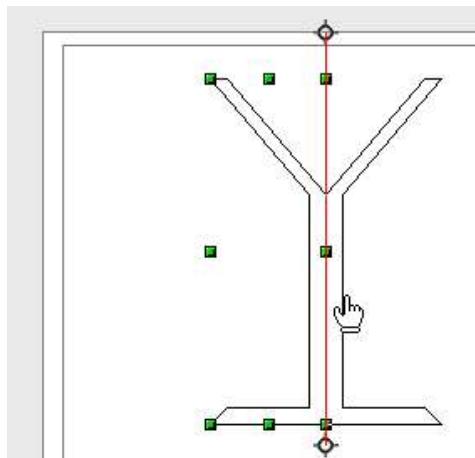


Illustration 104 - Déplacement de l'axe de révolution

Pour que la figure obtenue ait l'apparence d'un verre, il faut déplacer l'axe rouge en le faisant glisser de l'autre côté du profil. La forme résultante se dessine autour de l'axe de symétrie.

Il est aussi possible d'incliner l'axe en agissant sur les deux poignées situées aux extrémités.

④

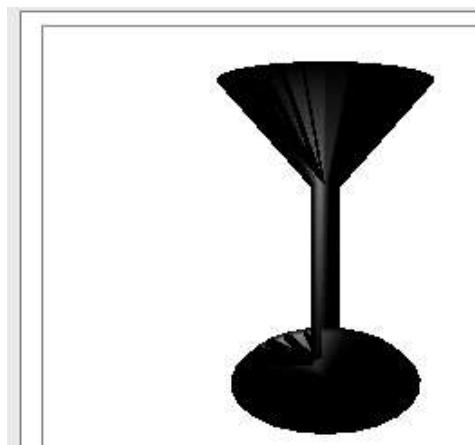


Illustration 105 - Exécution de la fonction de révolution 3D

Pour voir le résultat, il suffit de cliquer sur le tracé 2D.

⑤



Illustration 106 - Application d'une couleur transparente sur l'objet 3D

Dans ce cas, pour rendre l'objet plus lisible, une couleur bleue transparente a été appliquée.

Cet outil va permettre de déformer un objet en déplaçant librement les poignées situées aux extrémités de son rectangle de sélection. Un rectangle par exemple pourra très facilement être transformé en la figure suivante:

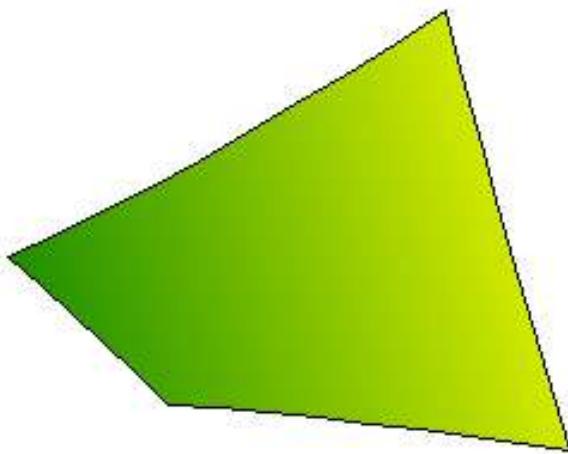


Illustration 107 - Résultat de la déformation d'un rectangle

Le principe de la déformation est très simple: En agissant sur les poignées de sélection situées sur les coins du rectangle, on étire plus ou moins l'objet. En agissant sur les poignées situées sur les arêtes, on incline plus ou moins l'objet.

Note: Lorsque vous sélectionnez la fonction de déformation sur un objet, OpenOffice va tout d'abord vous demander de le transformer en courbe (comme pour l'opération d'édition de points). La transformation sera donc irréversible.

Nous avons vu à la page 153 que draw fournissait des mécanismes puissants pour définir la transparence d'un objet. La boîte de dialogue de transparence permet d'ajuster finement tous les paramètres de cette dernière mais il n'est pas toujours simple de voir rapidement le résultat des modifications avant d'avoir enregistré.

La fonction de définition dynamique de la transparence permet de pallier à ce petit inconvénient. Sa mise en œuvre est extrêmement simple. Lorsque vous cliquez sur l'icône , une flèche apparaît sur l'objet sélectionné.

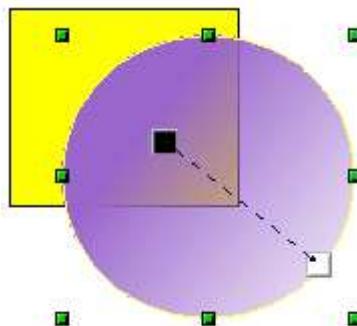


Illustration 108 - Modification dynamique de la transparence

En déplaçant les deux carrés au bout de cette flèche, vous modifierez dynamiquement la transparence de l'objet. Le carré noir permet de positionner le côté opaque et le carré blanc le côté transparent. Il est possible de cliquer sur une case de la barre des couleurs pour modifier la couleur de l'objet (et donc la couleur du côté opaque). Cliquez en dehors de l'objet pour appliquer les modifications sur la transparence.

Note: Les modifications que vous pouvez effectuer sur la flèche vont dépendre du type de transparence que vous avez choisi. Par exemple, dans le cas d'une transparence axiale, le carré situé au centre de l'objet ne pourra pas être déplacé, il sera simplement possible d'agir sur l'autre carré. Dans une transparence de forme carrée, la flèche pourra librement être déplacée.

Cet outil se manipule exactement comme l'outil de gestion de la transparence. Il n'est actif que si l'objet sélectionné est colorié avec un dégradé. Un appui sur l'icône de dégradé  affiche sur l'objet sélectionné une flèche se présentant ainsi:

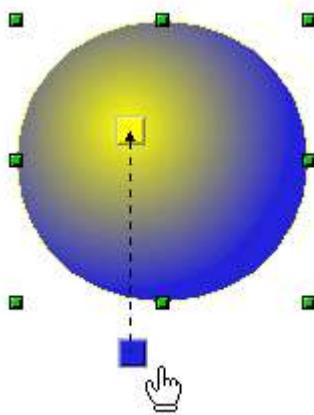


Illustration 109 - Définition dynamique d'un dégradé

Les deux extrémités de la flèche correspondent aux couleurs de départ et de fin du dégradé. Vous pouvez changer les couleurs de ces deux carrés en cliquant sur une case de la barre des couleurs et en la faisant glisser sur le carré souhaité. Les carrés et la flèche peuvent être déplacés librement. Les déplacements de ces carrés se reflètent immédiatement sur le dégradé de l'objet sélectionné. Cliquez en dehors de l'objet pour appliquer les modifications sur le dégradé.

Note: Les modifications que vous pouvez effectuer sur la flèche vont dépendre du type de dégradé que vous avez choisi. Par exemple, dans le cas d'un dégradé linéaire, les carrés de départ et de fin du dégradé seront toujours situés de part et d'autre du point central de l'objet.

Cette section donnera un aperçu complet des différents types d'objets élémentaires que l'on peut tracer avec draw. Ces objets peuvent ensuite être édités, combinés etc... pour former des figures plus complexes.

Dans les pages précédentes, nous avons déjà évoqué l'utilisation de certaines de ces primitives.

Toutes les palettes d'outils décrites ici sont affichables depuis la barre des instruments.

Les textes

La palette du mode texte se présente sous la forme suivante:

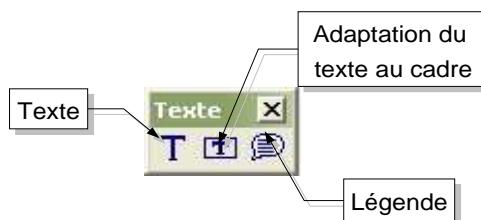


Illustration 110 - Les outils de la palette du mode texte

- Cet outil permet d'écrire des textes dans la police, la couleur et la taille par défaut définies dans la barre des objets du mode texte.

Si vous cliquez sur une zone vide de l'espace de travail, le texte sera écrit à cet endroit. Si vous cliquez sur un objet, le texte sera écrit au centre de l'objet et restera localisé à l'intérieur de l'objet dont la bordure deviendra le cadre du texte.

Lorsque vous avez fini de taper le texte, cliquez à l'extérieur avec la souris. Vous pouvez si vous le souhaitez passer à la ligne en utilisant la touche [Enter]. Notez que le texte reste éditable à tout instant (il suffit de cliquer dessus avec la souris).

Lorsque vous tapez le texte, notez que la règle supérieure comporte les attributs classiques des paragraphes: décalages, première ligne, tabulation.

Vous pouvez tout comme avec Writer modifier le style de tout ou partie des caractères. Il est intéressant de noter que le styliste fonctionne aussi dans ce cas, ce qui vous permet de créer des styles réutilisables sur d'autres cadres de textes exactement comme vous pourriez le faire avec Writer.

Les cadres de texte sont aussi manipulables comme tous les autres objets de draw. Vous pouvez y affecter des couleurs de remplissage, des ombres etc... Vous pouvez bien entendu faire pivoter le cadre et avoir ainsi du texte écrit sur un angle quelconque.

-  Cet outil permet aussi de taper des textes mais vous pouvez agir sur la taille du texte résultat en manipulant les poignées de sélection de l'objet.

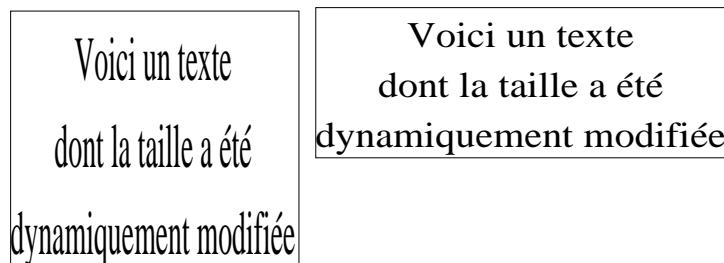


Illustration 111 - Exemple de textes modifiés

Dans l'exemple ci-dessus, il s'agit du même texte qui a été dupliqué et dont j'ai modifié le rectangle de sélection. Pour rendre l'effet plus clair, j'ai affiché la bordure du rectangle.

Là aussi, le texte reste toujours modifiable.

En combinant les fonctions disponibles, on peut arriver à créer des effets très nombreux:



Illustration 112 - Exemple de déformation

Tous les textes créés avec cet outils sont manipulables avec Fontwork qui est un puissant outil de déformation de textes. Reportez-vous à la page 234 pour plus d'informations.

-  La dernière icône de la palette sert à créer des légendes. Il s'agit en fait de texte entouré d'un cadre et disposant d'un connecteur (une flèche articulée). J'ai utilisé à de nombreuses reprises les légendes dans ce manuel. Par exemple, le descriptif des palettes des différentes primitives est réalisé avec une légende.

Les rectangles et les carrés

Cette palette comporte huit outils:

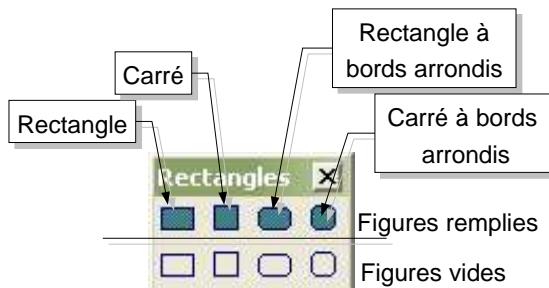


Illustration 113 - Palette des rectangles et des carrés

■ Ces icônes permettent de dessiner des rectangles remplis ou vides.

■ Ces icônes permettent de tracer des carrés remplis ou vides.

Note: Si vous maintenez la touche [Maj] enfoncee lors du tracé d'un rectangle, vous obtiendrez aussi un carré.

■ Ces icônes permettent de tracer des rectangles arrondis.

Note: Nous avons déjà vu comment transformer un rectangle droit en rectangle arrondi et vice-versa.

■ Ces icônes permettent de tracer des carrés arrondis. Ces carrés seront vides ou remplis.

Les cercles, les ellipses et les arcs

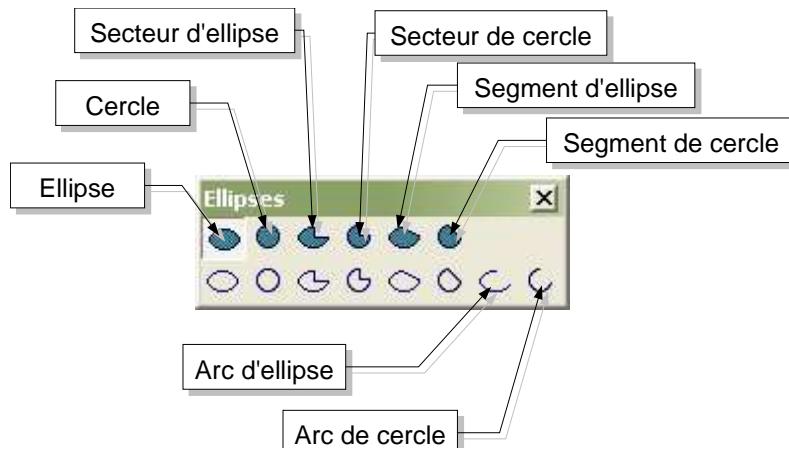
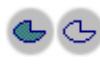


Illustration 114 - Palette des ellipses, des cercles et des arcs

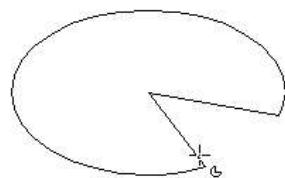
■ Ces deux outils vont permettre de tracer des ellipses vides ou remplies.

 Ces deux icônes permettent de tracer des cercles remplis ou pas.

Note: Tracer une ellipse en maintenant la touche [Maj] enfoncee permet d'obtenir un cercle.

 Grâce à ces deux icônes, vous pourrez tracer un secteur d'ellipse. La méthode de tracé est la suivante:

On trace d'abord l'ellipse. On place ensuite sur le tracé de l'ellipse la position du premier segment et on fait de même pour le deuxième segment. OpenOffice affiche en permanence une trace du secteur l'ellipse en cours de dessin:



*Illustration 115 - Secteur d'ellipse
en cours de dessin*

Notez la forme particulière du curseur de la souris lors du dessin du secteur d'ellipse.

Note: Nous avons déjà vu qu'en mode édition de points, vous pouvez repositionner les segments (Cf page 139).

 Ces deux outils permettent de tracer des secteurs de cercle. La méthode de tracé est similaire à celle du secteur d'ellipse.



Il est possible de tracer des segments d'ellipse. Un segment étant une ellipse 'coupée' en deux le long d'un segment de droite. Le principe de tracé est simple: on trace d'abord l'ellipse, on place ensuite le premier point du segment puis le deuxième. Dans ce cas aussi, OpenOffice affiche une trace du segment en cours de réalisation:

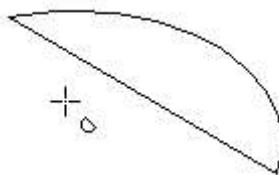


Illustration 116 - Réalisation d'un segment d'ellipse

Notez la forme du curseur de la souris.

Comme nous l'avons déjà mentionné (page 139), le segment reste modifiable en déplaçant les points de contrôle en mode édition de points.



Ces deux icônes permettent de réaliser des segments de cercle.



Cette icône permet de réaliser un arc d'ellipse. Le fonctionnement est similaire au secteur d'ellipse mais la figure résultante n'est pas fermée.

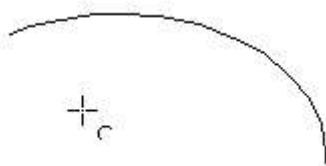


Illustration 117 - Tracé d'un arc d'ellipse

Dans ce cas aussi, vous pouvez noter la forme du curseur de la souris. Comme les secteurs et les segments, les arcs sont modifiables à tout moment.



Le dernier outil de la palette des ellipses permet la réalisation d'arcs de cercles.

Les objets 3D

Draw permet de tracer des objets tridimensionnels. Ces objets seront plus particulièrement abordés au chapitre (page 211). La palette des objets 3D va autoriser la création de huit primitives de base qui, par fusion ou combinaison, permettront d'obtenir des objets plus complexes.

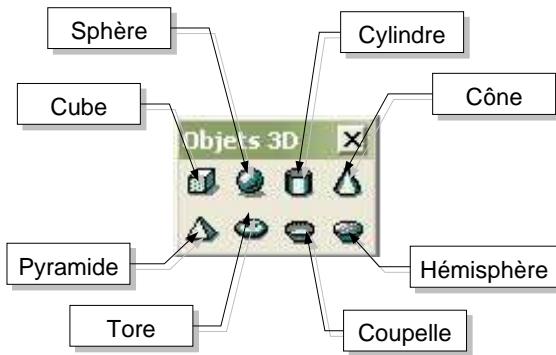


Illustration 118 - Palette des objets 3D

La création d'un objet 3D fonctionne toujours de la même façon: Cliquez sur l'icône associée puis tracez sur la zone de travail une rectangle correspondant à la zone avant de la figure à tracer. Une trace du cube contenant le futur objet est affichée.

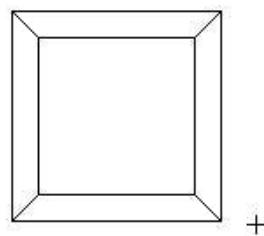


Illustration 119 - Tracé d'un objet 3D



Cet outil va créer un cube.

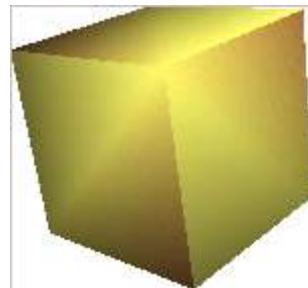


Illustration 120 - Cube

Plus généralement, cette fonction va créer une boîte parallélépipédique 3D. Si vous souhaitez créer un cube, pressez sur la touche [Control] pendant le tracé de la figure.



Cette icône va permettre de créer une sphère.

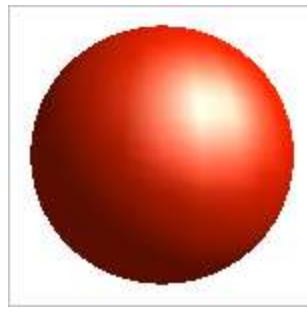


Illustration 121 - Sphère

Dans ce cas aussi, la figure tracée sera en fait un ellipsoïde. Pour tracer exactement une sphère, pressez sur la touche [Control].



Cet outil va permettre de créer un cylindre.

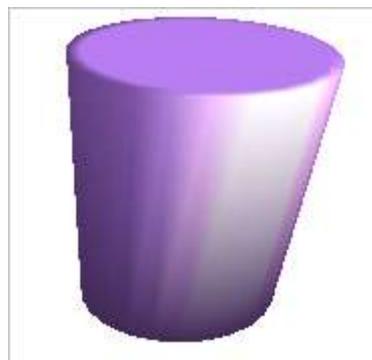


Illustration 122 - Cylindre 3D



Cet outil va permettre de créer un cône.

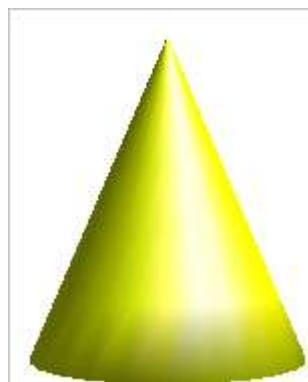


Illustration 123 - Cône 3D



Cet outil va créer une pyramide.



Illustration 124 - Pyramide



Cet outil va permettre de créer un tore (un anneau).



Illustration 125 - Tore



Cet outil va permettre de créer une coupelle. Il s'agira donc d'une demi-sphère évidée.



Illustration 126 - Coupelle



Cet outil va créer un hémisphère (une demi-sphère). Par rapport à une coupelle, l'hémisphère est une figure fermée.



Illustration 127 - Hémisphère

La palette de tracé de courbes propose huit outils destinés à réaliser des profils

non linéaires.

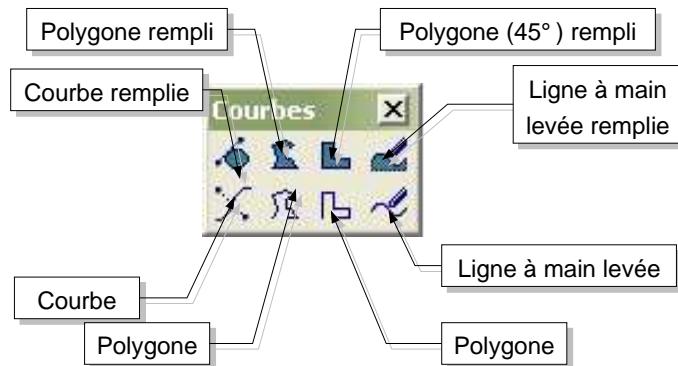


Illustration 128 - Palette des courbes



Cette icône est l'outil de base pour le tracé de courbes. La courbe résultante pourra être manipulée par l'intermédiaire des courbes de Bézier que nous avons déjà étudiées. Le principe de tracé est le suivant:

Cliquez pour placer le premier point. Tout en gardant le bouton de la souris appuyé, déplacez là pour tracer la tangente à la courbe passant par le point de départ. Relâchez ensuite le bouton et déplacez la souris pour tracer la courbe.

Pour terminer la figure, double-cliquez avec le bouton gauche de la souris.

①

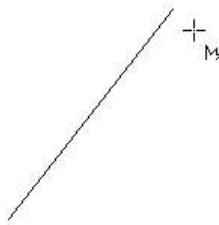


Illustration 129 - Tracé du segment initial

②



Illustration 130 - Tracé de la courbe

③

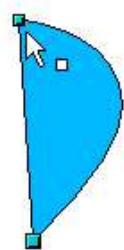


Illustration 131 - Courbe terminée

Cette opération peut être recommencée immédiatement à partir du nouveau point d'arrivée.

La figure que vous avez créée va être remplie ou vide en fonction de l'icône sélectionnée. Dans le cas d'une figure remplie, un segment va être tracé entre le premier point de départ et le dernier point d'arrivée.

L'objet créé va être manipulable en particulier en mode édition de points pour jouer sur les tangentes.



Via cette fonction, vous pourrez créer des polygones. Un polygone étant constitué d'une succession de segments de droite.

Le fonctionnement est très simple: Cliquez dans l'espace de travail pour placer le premier point, déplacez la souris puis cliquez de nouveau pour tracer le premier segment, puis continuez à déplacer la souris et cliquer pour les segments suivants. Double-cliquez pour terminer la figure. Si vous avez choisi une figure fermée, un segment reliera le point de départ à l'extrémité du dernier segment que vous avez tracé.



Cet outil est similaire au précédent mais les différents segments auront tous des angles multiples de 45° par rapport à l'horizontale. La figure pourra être fermée ou ouverte en fonction de l'icône choisie.



Lorsque vous cliquez sur cet outil, vous allez créer une ligne à main levée. Le principe est le suivant: Il vous suffit de déplacer la souris tout en gardant le bouton gauche appuyé. La trace du déplacement de la souris devient la courbe que vous tracez. OpenOffice va en fait tracer une courbe constituée de nombreux points de contrôle. Vous pourrez vous en apercevoir en passant en mode édition de points. Là aussi, le choix de l'icône permettra d'obtenir une courbe ouverte ou fermée.

Note: Pour que la courbe résultante soit de bonne qualité, il vaut mieux bouger la souris assez lentement.

Les lignes et les flèches

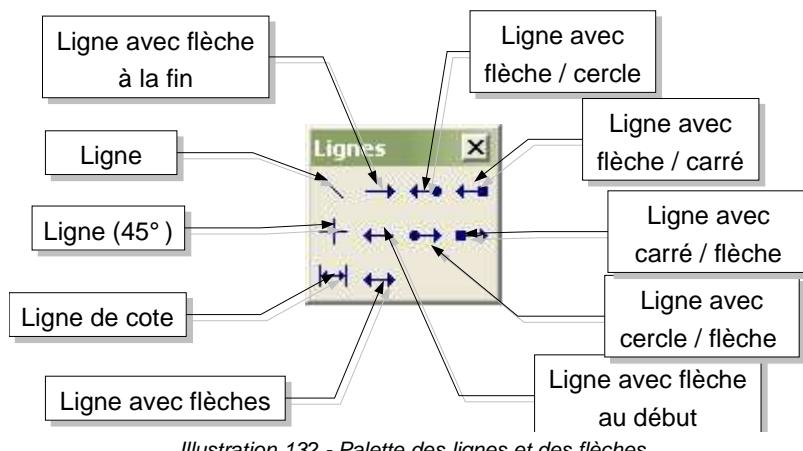


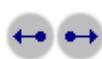
Illustration 132 - Palette des lignes et des flèches



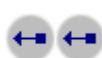
Cette fonction correspond à la plus simple des primitives: un segment de droite.



Le segment tracé débutera ou finira (en fonction de l'icône choisie) par une flèche.



Les deux extrémités du segment seront constituées d'un cercle d'un côté et d'une flèche de l'autre.



Les deux extrémités du segment seront constituées d'un carré d'un côté et d'une flèche de l'autre.



Les segment tracé fera un angle multiple de 45° avec l'horizontale. Vous pouvez obtenir le même effet en pressant la touche [Control] lors du tracé d'un segment.



Cette icône va permettre de créer des cotations. Une cotation associe une ligne fléchée, deux segments permettant d'indiquer les extrémités de la zone mesurée et une mesure dans l'unité courante. Les cotations permettent par exemple de tracer des plans tout en indiquant automatiquement les différents métrés:

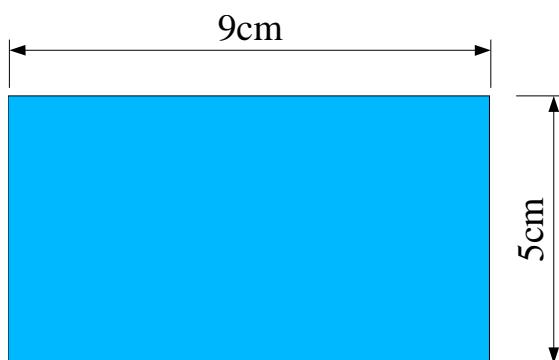


Illustration 133 - Exemple simple de cotation

En cas de redimensionnement d'une cotation, la mesure correspondante est automatiquement recalculée. Si vous groupez un objet avec ses cotations, tout changement de taille de l'objet se reflétera immédiatement sur les mesures des cotations. On parle alors de cotations associatives. Les cotations disposent d'une boîte de paramétrage propre accessible via le menu local de la cotation.

Les cotations s'affichent par défaut dans l'unité de la feuille que vous pouvez régler via le menu Outils / Options / Dessin / Général.

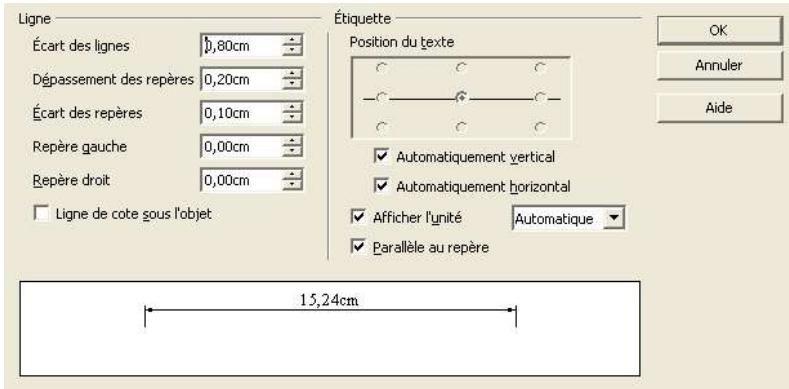


Illustration 134 - Boîte de paramétrage des cotations

Cette boîte vous permettra d'indiquer la position des différents éléments constitutifs de la cotation. Vous pourrez y entrer les dimensions des lignes, l'écart par rapport aux repères, la position du texte et éventuellement changer l'unité d'affichage de la mesure (voire choisir de ne pas afficher l'unité).

Vous pouvez aussi paramétrier l'aspect de la ligne et du texte en utilisant les fonctions traditionnelles.

 Vous tracerez grâce à cet outil, un segment muni de flèches aux deux extrémités.

Les connecteurs

Les connecteurs sont des courbes particulières dont les extrémités sont 'fixées' sur les points d'autres objets et qui suivent ces points lors du déplacement des objets associés. Ils sont particulièrement utiles lors de la réalisation d'organigrammes. En effet, vous pourrez réorganiser les blocs de votre organigramme tout en maintenant les liaisons entre eux.

OpenOffice draw vous apporte des fonctionnalités réellement remarquables en matière de gestion des connecteurs. L'accès à certaines fonctionnalités est caché mais nous étudierons la méthode permettant de le rendre visible.

Le principe de tracé d'un connecteur est le suivant: A tous les objets sont associés des points de collage (invisibles par défaut). Les connecteurs vont se positionner sur ces points de collage. Par défaut tous les objets possèdent 4 points de collage que draw a placés par défaut. Nous verrons plus loin qu'il est possible de modifier la position des points par défaut et de rajouter de nouveaux points de collage.

Ces points de collage sont situés sur les arêtes du rectangle de sélection de l'objet. Comme vous pouvez le voir sur la figure suivante, cela signifie que les points de collage ne sont pas forcément situés sur le pourtour de l'objet:

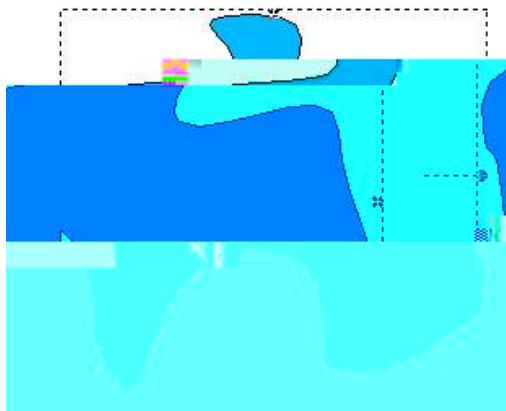


Illustration 135 - Points de collage d'un objet

Lors du déplacement d'une des extrémités d'un connecteur sur un objet, ses points de collage sont visualisés. Vous pouvez 'lâcher' l'extrémité du connecteur un des points de collage. Par la suite, lors de tout déplacement de l'objet ou du connecteur, l'extrémité du connecteur restera fixée sur le point de collage .

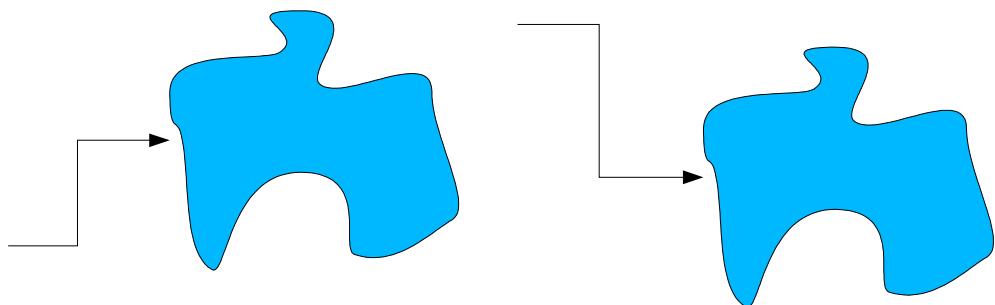


Illustration 136 - Impact du déplacement d'un objet sur un connecteur

Vous pouvez aussi lâcher l'extrémité d'un connecteur au contre de l'objet. Dans ce cas, lors du déplacement de l'objet ou du connecteur, OpenOffice choisira automatiquement le point de collage le plus judicieux pour éviter de trop rallonger le connecteur:

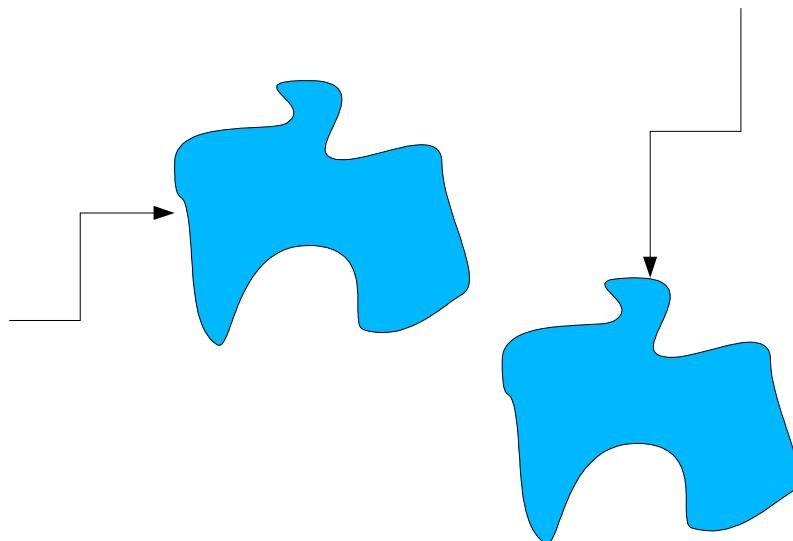


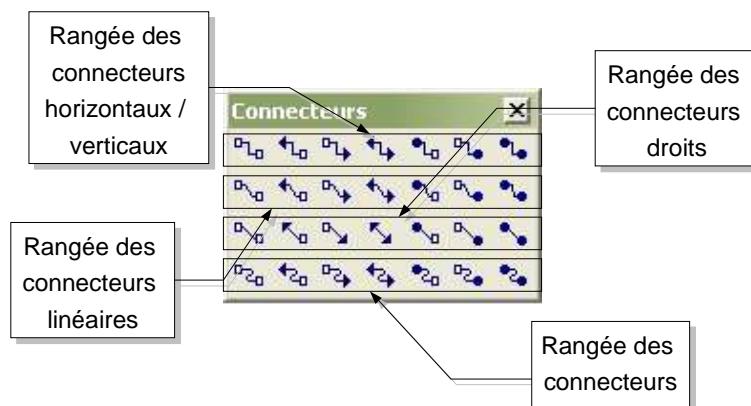
Illustration 137 - Modification automatique du point de collage

Dans la mesure du possible, OpenOffice va éviter de superposer le tracé du connecteur avec celui de l'objet.

Vous pouvez casser le lien entre un connecteur et un objet en éloignant du point de collage l'extrémité du connecteur qui y était fixée.

Comme pour tous les objets, des points de contrôle sont associés aux connecteurs pour vous permettre de mieux maîtriser leur tracé. Le principal point de contrôle est situé au centre des connecteurs et il permet de maîtriser la taille des deux segments de chaque côté.

La palette d'outil des connecteurs comporte de nombreuses icônes.

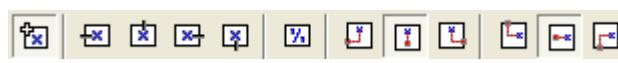


Comme le montre la copie d'écran ci-dessus, elles sont facilement classables en quatre catégories pour le type de ligne du connecteur; chaque catégorie étant subdivisée en 7 types de connecteurs en fonction de la flèche choisie.

- ➊ Les icônes de cette rangée vont permettre de tracer des connecteurs classiques comme ceux que nous avons vus dans les pages précédentes.
- ➋ Ces connecteurs sont composés d'un segment de ligne et de deux petits segments aux extrémités horizontaux ou verticaux.
- ➌ Ces connecteurs sont constitués d'une simple droite
- ➍ Ces connecteurs sont basés sur une courbe de Bézier

La gestion des points de collage est dévolue à une barre d'icône particulière. Cette barre n'est pas affichée par défaut. Pour la visualiser, il faut utiliser l'icône présente sur la barre des options (Cf page 126).

Cette barre se présente ainsi:



Voici un descriptif de la fonction des différentes icônes:

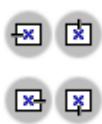


Cette icône permet d'insérer un nouveau point de collage. Si la figure est remplie, ce point peut être situé n'importe où dans l'objet et pas seulement sur son pourtour. Après avoir sélectionné cet outil, cliquez dans la figure pour y ajouter des points de collage.



Illustration 140 - Exemple
de points de collage

Les points de collage restent visible tant que l'icône reste enfoncée. Il apparaissent sous la forme d'une petite croix bleue et le point de collage actuellement sélectionné apparaît en surbrillance. Vous pouvez déplacer les points à la souris et les supprimer avec la touche [Suppr].



Ces quatre icônes permettent d'indiquer quelles sont les directions autorisées pour la jonction d'un connecteur autour d'un point de collage. Il est possible de sélectionner plusieurs de ces icônes pour un point donné. Voici un exemple concret d'utilisation de ces icônes:

Voici une figure simple sur laquelle un point de collage a été rajouté.

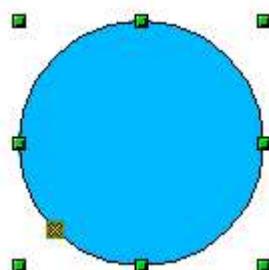


Illustration 141 - Rajout d'un
point de collage

Par défaut, les lignes des connecteurs peuvent atteindre ce point en arrivant sur n'importe quelle direction.

Cliquez sur l'icône . Cela va avoir pour effet de forcer tout connecteur placé sur ce point à 'arriver' par la gauche comme le montre l'exemple suivant:

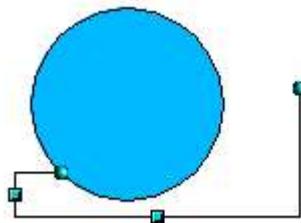


Illustration 142 - Effet sur un connecteur

Si en mode édition des points de collage, vous cliquez en plus sur l'icône , vous ajouterez une nouvelle direction possible sur un point de collage. Dans l'exemple précédent, cela donnerait:

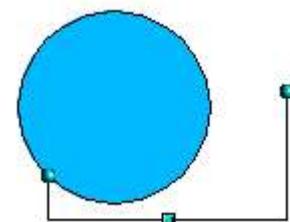


Illustration 143 - Changement de la direction d'un connecteur

L'ajout de cette direction supplémentaire a permis à OpenOffice de tracer un connecteur plus court.

Lorsque cette icône est activée (c'est le cas par défaut), tout redimensionnement d'un objet va se traduire par un déplacement des points de collage associés. Si cette icône est désactivée, ce ne sera pas le cas comme le montrent les deux illustrations suivantes:

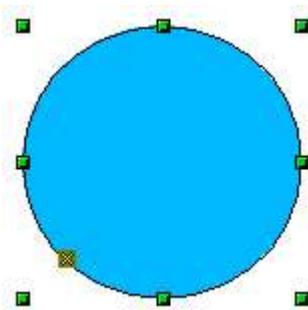


Illustration 144 - Redimensionnement d'un objet avec suivi des points de collage

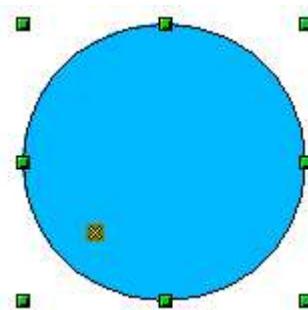


Illustration 145 - Redimensionnement d'un objet sans suivi des points de collage

Lorsque l'icône est désactivée, les six dernières icônes de la barre d'outil qui étaient grises par défaut deviennent activables. Ces icônes permettent d'indiquer comment se repositionneront les points de collage lors du redimensionnement de l'objet.



Ces trois icônes permettent de spécifier le positionnement horizontal du point de collage. Celui-ci peut garder la même position par rapport au bord gauche (1^{ère} icône), garder sa position par rapport au centre (2^{ème} icône) ou par rapport au bord droit (3^{ème} icône).



Ces trois icônes permettent de spécifier le positionnement vertical du point de collage. Celui-ci peut garder la même position par rapport au bord supérieur (1^{ère} icône), garder sa position par rapport au centre (2^{ème} icône) ou par rapport au bord inférieur (3^{ème} icône).



Draw intègre de nombreux mécanismes destinés à combiner des objets entre eux afin de permettre des modifications de tout le groupe ou même de créer de nouveaux objets.

Les regroupements permettent de combiner des objets afin d'appliquer des modifications globales. Un regroupement est toujours réversible et les objets composant le groupe peuvent toujours être manipulés indépendamment.

Une combinaison est une opération de fusion d'objets donnant naissance à un nouvel objet. Les objets initiaux ne sont plus utilisables individuellement et cette opération est irréversible (sauf à utiliser la fonction Édition / Annuler bien entendu).

Dans tous les cas, ces fonctions nécessitent évidemment que plusieurs objets soient sélectionnés. Je rappelle les deux principales méthodes de sélection 'multi-objets': cliquer sur chaque objet en maintenant la touche [Maj] enfoncée ou tracer un rectangle de sélection autour des objets à sélectionner. Reportez-vous à la page 135 pour plus d'informations.

Les fonctions de regroupement

Lorsque plusieurs objets sont sélectionnés, les différentes opérations s'appliquent sur les différents objets. Vous pouvez par exemple faire tourner le groupe dans son intégralité. Les groupements obtenus par sélection commune de plusieurs objets sont annulés dès que vous cliquez à l'extérieur du regroupement. Il existe une méthode de regroupement permettant de maintenir groupés les objets sélectionnés.

Pour grouper les objets sélectionnés, vous pouvez utiliser le menu local des objets et cliquer sur l'option 'Grouper':



Illustration 146 - Menu local pour la gestion des groupements

Vous pouvez aussi utiliser le raccourci [Control] + [Maj] + [G]. Il est aussi possible d'utiliser le menu principal 'Modifier / Grouper'.

Dès que des objets sont groupés, toutes les opérations d'édition s'effectuent sur tous les objets du regroupement. Si vous cliquez sur un des objets du groupe, tout le groupe se sélectionnera. Cette méthode est donc plus sûre que la méthode par sélection directe qui doit être réservée à des cas de figure simples.

Comme indiqué au début du chapitre, les objets d'un groupement gardent leur individualité et il est possible d'arrêter un groupement en utilisant le menu local que vous avez utilisé pour grouper les objets (fonction 'dissocier') ou en utilisant le raccourci [Alt] + [Control] + [Maj] + [G] ou il est possible d'utiliser le menu principal 'Modifier / dissocier'.

Les objets d'un groupement gardant toutes leurs propriétés, il reste toujours possible de les éditer individuellement sans casser le groupe. Il faut pour cela utiliser la fonction 'Entrer dans le groupement' accessible via le menu local du groupe ou par appui sur la touche [F3]. Il est aussi possible de double-cliquer dans le groupe.

Dès que vous êtes dans ce mode, il devient possible de cliquer sur chaque objet et de l'édition individuellement.

Pour sortir de ce mode, il faut utiliser la fonction 'sortir du groupement' ou utiliser le raccourci [Control] + [F3]. Vous pouvez aussi, plus simplement, cliquer à l'extérieur du groupe.

Lorsque vous travaillez à l'intérieur d'un groupement, les objets qui ne sont pas contenus dans le groupement ne sont plus sélectionnables et apparaissent estompés.

Voici un exemple d'utilisation de cette fonction:

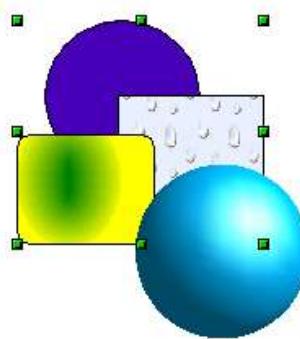


Illustration 147 - Disposition des objets pour l'exemple de groupement

Les trois figures 2D ci-dessus font partie d'un groupement. La sphère 3D n'en fait pas partie. Lorsque le groupement est sélectionné (comme ci-dessus) et que vous appuyez sur la touche [F3], vous trouverez dans le cas de figure suivant:

La sphère devient estompée pour montrer qu'elle ne fait pas partie du groupement et qu'elle n'est pas sélectionnable. Dans le mode, il devient possible de sélectionner un des trois objets du groupement pour l'édition individuelle:

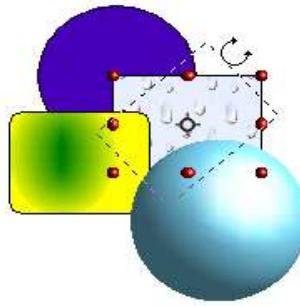


Illustration 149 - Édition d'un objet
du groupement

Dans ce cas, on fait pivoter le rectangle droit. Lorsque vous appuyez sur la combinaison de touches [Control] + [F3], vous quitterez le mode groupement

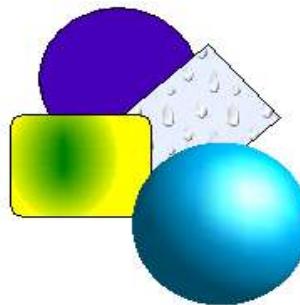


Illustration 150 - Sortie du
groupement

et la sphère redeviendra totalement visible pour montrer qu'elle est de nouveau sélectionnable:

Notez qu'il est possible de faire des groupes à partir d'autres groupes. OpenOffice garde dans ce cas la hiérarchie originale. Si vous dissociez un groupe formé d'autres groupes, vous retrouverez les groupes individuels que vous pourrez à leur tour dissocier.

Les fonctions de combinaison

Au contraire des opérations de groupage, les fonctions de combinaisons vont donner naissance à un nouvel objet. Le principe est le suivant: Sélectionnez tous les objets que vous souhaitez combiner:

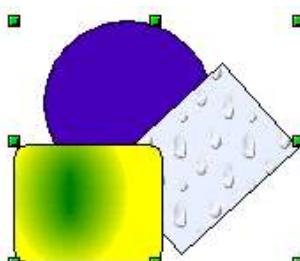


Illustration 151 - Objets pour la
fusion

Utilisez ensuite l'option 'Combiner' du menu local du groupe. Le raccourci de cette fonction est [Control] + [Maj] + [K].

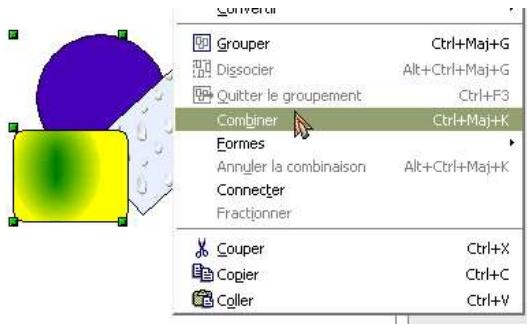


Illustration 152 - Inspecteur local pour la combinaison

Le résultat de la combinaison est représenté sur la figure suivante:

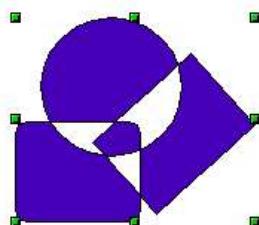


Illustration 153 - Résultat de la combinaison

A priori, il peut sembler surprenant. Les règles régissant la combinaison permettent heureusement de l'expliquer:

- L'attribut de l'objet résultant est l'attribut de l'objet qui était situé le plus à l'arrière plan. Dans le cas décrit ici, il s'agit du cercle. La figure résultante a la même couleur, épaisseur de ligne etc... que le cercle.
- Le résultat de la combinaison est une courbe de Bézier.
- Lorsque des objets se chevauchent, on va obtenir pour les zones de chevauchement une zone remplie ou une zone vide en fonction du nombre de chevauchements. Lorsque les chevauchements sont en nombre pair, on obtient un vide (représenté par un 'trou') dans la figure. Lorsque les chevauchements sont en nombre impair, on obtient une zone pleine. Dans l'exemple, on peut compter le nombre de superpositions :

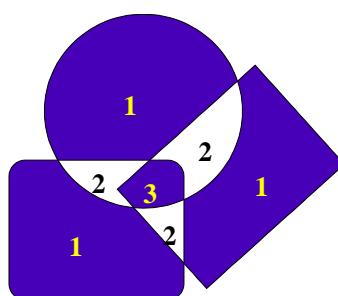


Illustration 154 - Règle des chevauchements

Il existe d'autres techniques de combinaison permettant d'arriver à des résultats différents. Elles seront décrites dans le chapitre suivant.

Les fonctions de fusion de formes

Toutes les fonctions sont accessibles via le menu local du groupe, sous-entrée 'Formes'. Il y a trois combinaisons possibles:

La figure résultant de la fusion correspond à l'union des figures.

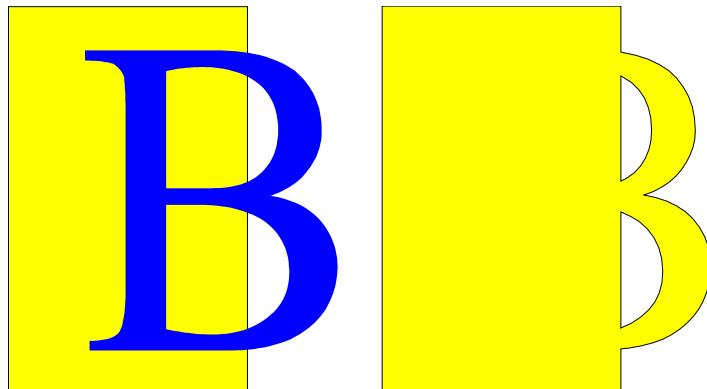


Illustration 155 - Résultat d'une fusion

La soustraction enlève au premier objet la partie du deuxième objet qui la chevauche:

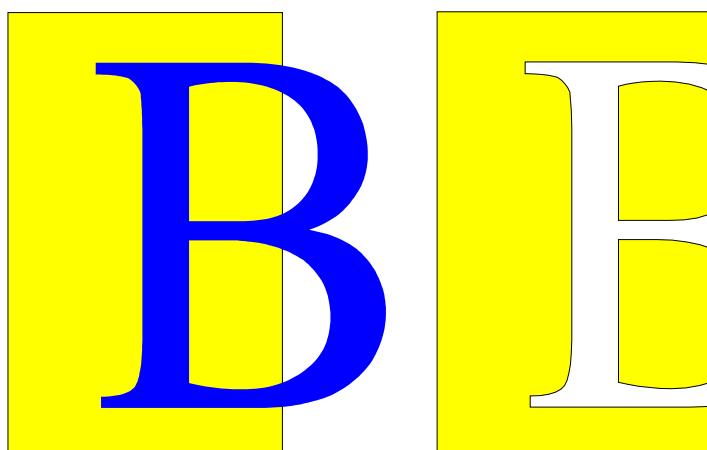


Illustration 156 - Résultat d'une soustraction

L'intersection permet d'obtenir une figure correspondant aux parties des objets se chevauchant.

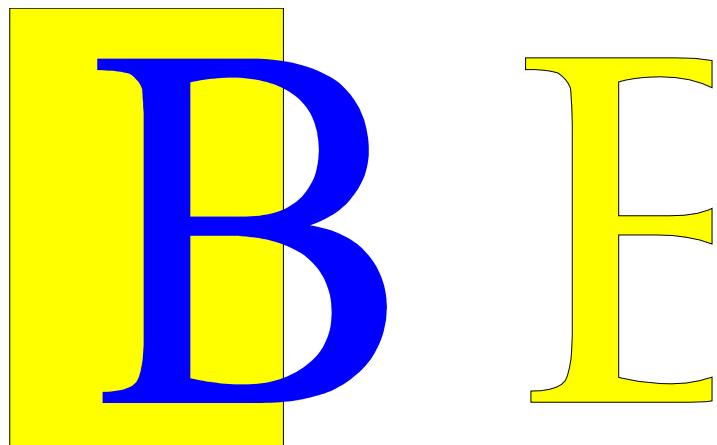


Illustration 157 - Résultat d'une intersection

Les fonctions d'aide au placement permettent de modifier l'alignement et l'ordre relatif des objets. Elles sont accessibles via deux icônes situées sur la barre d'instrument.

- ▢ Cette icône donne accès à la fenêtre de gestion des alignements des objets.
- ▢ Cette icône donne accès à la fenêtre de gestion de la disposition des objets

Alignment d'objets

La palette d'alignement comporte six icônes et se présente sous la forme suivante:

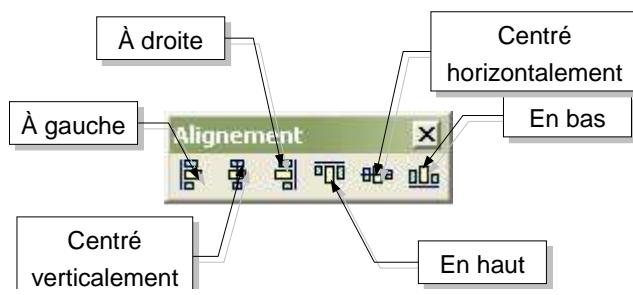


Illustration 158 - Palette d'alignement

Cette fenêtre peut être 'extraite' de la barre des instruments pour rester visible sur le plan de travail. Les icônes sont actives lorsque des objets sont sélectionnés. Elle va permettre d'aligner tous les objets sélectionnés d'une façon déterminée.

Pour montrer l'effet de l'alignement, je partirai des trois objets suivants:

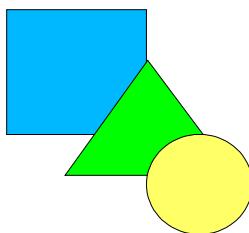
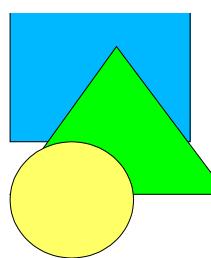
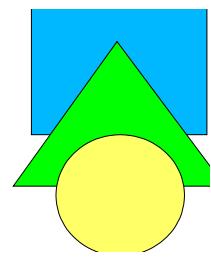


Illustration 159 - Exemple pour l'alignement



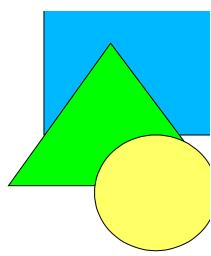
*Illustration 160 -
Alignement à gauche*

Tous les objets s'alignent sur leur côté gauche.



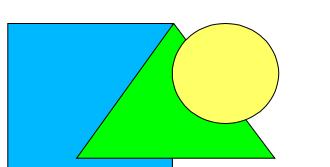
*Illustration 161 -
Centrage vertical*

Tous les objets se centrent verticalement.



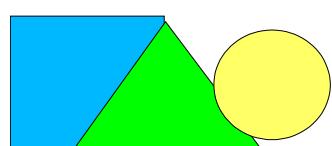
*Illustration 162 -
Alignement à droite*

Tous les objets s'alignent sur leur côté droit.



*Illustration 163 - Alignement en
haut*

Tous les objets s'alignent sur leur bord supérieur.



*Illustration 164 - Centrage
horizontal*

Tous les objets se centrent horizontalement.

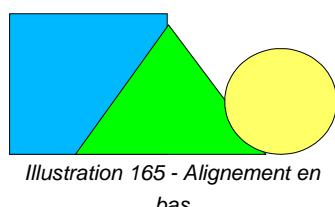


Illustration 165 - Alignement en bas

Tous les objets s'alignent sur leur bord inférieur.

Les fonctions de répartition d'objets

Lorsque plusieurs objets sont sélectionnés, draw vous fournit des mécanismes permettant de répartir harmonieusement ces objets horizontalement et verticalement. Vous pouvez afficher la boîte de dialogue de répartition d'objets en sélectionnant l'option 'Répartition' du menu local du groupe d'objets sélectionnés (il faut que trois objets au moins soient sélectionnés). Cette boîte se présente ainsi:

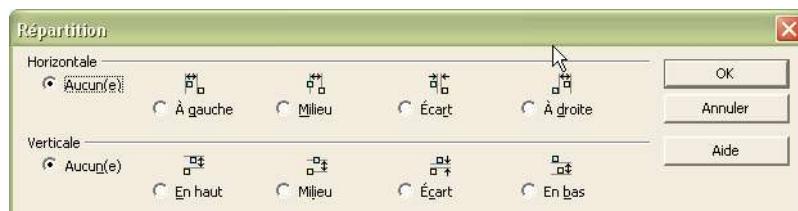
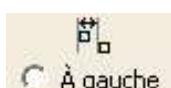


Illustration 166 - Boîte de répartition d'objets

Les différentes cases à cocher permettent de choisir la répartition des objets indépendamment sur l'axe horizontal et vertical. Les deux cases 'Aucun(e)' permettant de choisir de ne pas aligner sur un des deux axes.



Les objets sont répartis horizontalement en se basant sur leur côté gauche.



Les objets sont répartis horizontalement en se basant sur leur milieu (en fait le milieu du rectangle de sélection de chacun des objets).



Les objets sont répartis en mettant le même écart horizontal entre deux objets.



Les objets sont répartis horizontalement en se basant sur leur côté droit.



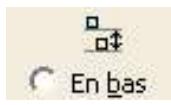
Les objets sont répartis verticalement en se basant sur leur côté supérieur.



Les objets sont répartis verticalement en se basant sur leur milieu.



Les objets sont répartis en mettant le même écart vertical entre deux objets.



Les objets sont répartis verticalement en se basant sur leur côté inférieur.

Gestion de la disposition des objets

La barre d'outil de gestion de la disposition de présente de la façon suivante:

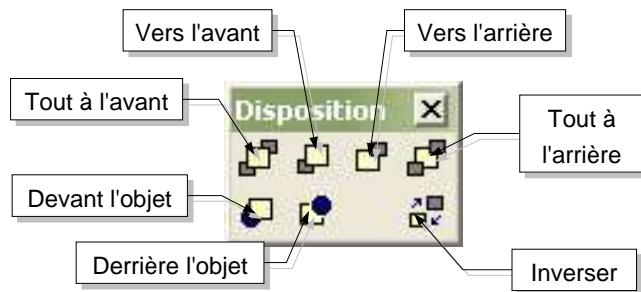


Illustration 167 - Palette de gestion de la disposition des objets

Il s'agit d'une fenêtre qui peut être détachée. Elle permet de gérer la superposition des objets en changeant leur ordre relatif. Les exemples que je prendrai seront basés sur cette figure:

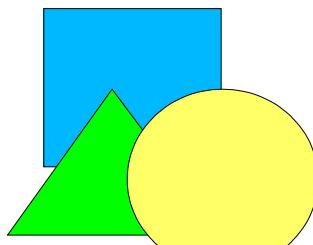


Illustration 168 - Figure
d'exemple de la gestion de la
disposition

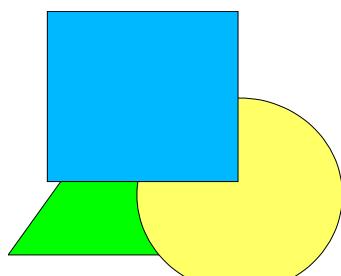


Illustration 169 - Fonction 'tout à l'avant'

Le carré était sélectionné.

Cette fonction permet de faire passer l'objet sélectionné au-dessus de la pile des objets. Dans le cas de la figure ci-contre, le carré est maintenant au-dessus de la pile.

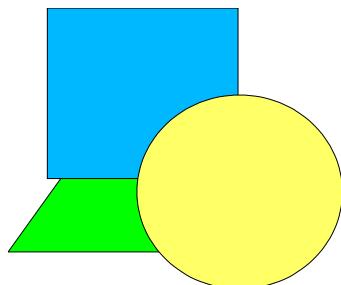


Illustration 170 - Fonction 'vers l'avant'

Le carré était sélectionné.

Cette icône permet de remonter d'un cran l'objet sélectionné. Le carré est passé devant le triangle mais est resté derrière le cercle.

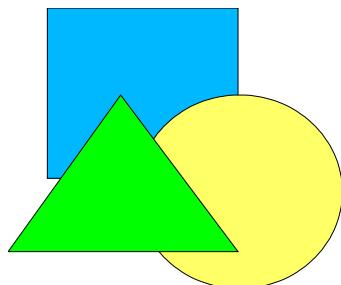


Illustration 171 - Fonction 'vers l'arrière'

Le cercle était sélectionné.

Cette outil permet de reculer d'un cran l'objet sélectionné. Dans la figure ci-contre, le cercle est passé derrière le triangle mais est resté devant le carré.

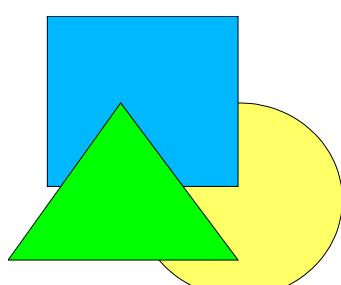


Illustration 172 - Fonction 'tout vers l'arrière'

Le cercle était sélectionné.

Cette fonction va faire passer l'objet sélectionné tout au-dessous de la pile. C'est le cas du cercle dans la figure ci-contre.

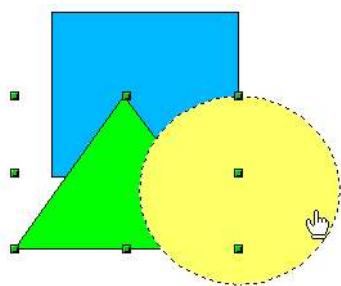


Illustration 173 - Fonction 'devant l'objet'

Cette fonction permet de faire passer un objet devant un autre objet désigné.

On sélectionne d'abord l'objet à déplacer (le triangle dans le cas ci-contre).

Après avoir cliqué sur l'icône, le curseur de la souris se transforme en main pour désigner l'objet devant lequel l'objet sélectionné doit se placer (le cercle ici).

Le résultat du déplacement est affiché à gauche.

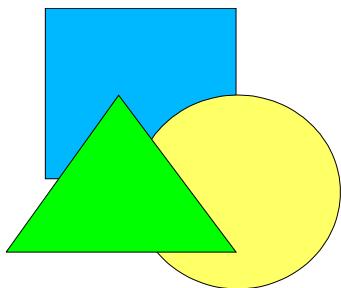


Illustration 174 - Résultat de la fonction 'devant l'objet'

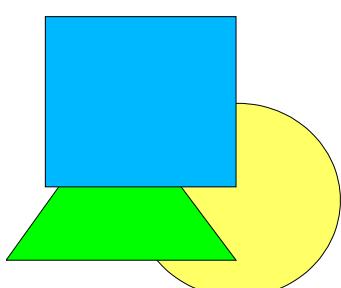


Illustration 175 - Fonction 'inverser'

Cette outil fonctionne comme l'outil précédent mais fait passer l'objet sélectionné derrière l'objet désigné avec la souris.

Cette fonction nécessite d'avoir au moins deux objets sélectionnés. Elle va inverser l'ordre des objets dans la pile. Dans l'exemple ci-contre, le carré et le cercle étaient sélectionnés et ont été inversés.

Cette section mettra en évidence les nombreux systèmes que draw met à la disposition des utilisateurs afin de leur faciliter la réalisation des dessins.

Choix de l'échelle de l'image / gestion des zooms

Pour pouvoir travailler plus précisément sur des zones définies des figures, draw fournit de nombreux outils destinés à zoomer plus ou moins sur des zones des images.

La valeur en pourcentage du zoom courant est affichée dans la barre d'état:



Illustration 176 - Zoom dans la barre d'état

En double-cliquant sur la valeur du zoom, on affiche une fenêtre permettant de modifier la valeur de ce pourcentage:

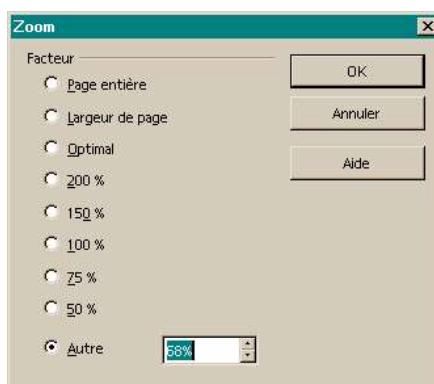
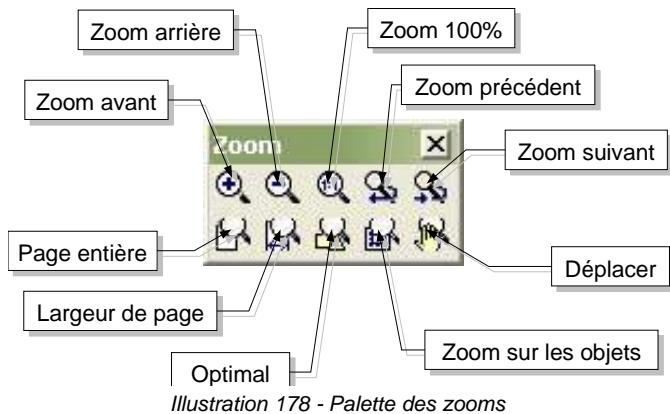


Illustration 177 - Fenêtre de choix du zoom

Vous pouvez choisir un des pourcentages prédéfinis, sélectionner 'Optimal' qui règle le facteur de zoom pour voir les objets présents le mieux possible, 'largeur de page' qui cale les bords droit et gauche de la page sur la zone de travail, 'Page entière' qui permet d'afficher toute la page de travail ou taper directement la valeur du zoom dans la zone de saisie 'Autre'.

Une barre d'outil est entièrement dévolue aux fonctions de zooms. Il s'agit d'une fenêtre détachable qui est accessible via l'icône de la barre des instruments.



-  Cette fonction permet à chaque clic de zoomer en avant d'un facteur 2. Vous pouvez aussi utiliser la touche [+] du clavier numérique. Il est aussi possible de tracer un rectangle de sélection autour de la zone sur laquelle on souhaite zoomer.
-  Cet outil permet à chaque clic de réduire la facteur de zoom d'un facteur 2. Vous pouvez aussi utiliser la touche [-] du clavier numérique.
-  Cette fonction permet d'afficher l'image présente à l'écran en taille réelle, 1 cm sur le moniteur correspondant à 1 cm de dessin. Pour que cet outil fonctionne, il faut en général que le driver de votre moniteur soit correctement installé.
-  Cette icône permet de revenir au zoom précédent.
-  Cette icône permet de revenir au zoom suivant (lorsqu'on a utilisé le zoom précédent au moins une fois). Ces deux icônes fonctionnent sur le principe des flèches gauche et droite de la barre d'outils des navigateurs internet.
-  Cette icône permet d'afficher la page dans sa totalité.
-  Cette icône permet de caler la largeur de la page sur celle de la zone de travail.
-  Cette icône permet d'obtenir un facteur de zoom permettant d'afficher de façon optimale les objets de la zone de travail.
-  Cette icône permet de zoomer sur les objets sélectionnés.



Cette fonction permet de déplacer la fenêtre de travail à un autre endroit de la feuille. Le curseur de la souris se transforme en main qu'il vous suffit de faire glisser pour afficher la zone voulue.

La gestion des couches de travail

L'espace de travail de draw peut être optionnellement constitué d'un empilement de calques sur lesquels vous pouvez tracer les différentes parties de votre dessin. Le gros avantage des calques provient du fait qu'ils peuvent à loisir être rendus visibles ou invisibles. Cette technique est très souvent utilisée dans les plans en architecture. Sur la feuille de base est dessiné le plan de masse. Sur les différents calques sont ensuite tracés les canalisations, les circuits électriques, les meubles, les cotations etc...

La gestion des calques est dévolue à un mode particulier de draw. Pour l'activer, vous devez utiliser le menu 'Affichage / couche' ou cliquer sur l'icône appropriée  en bas à gauche de l'espace de travail:

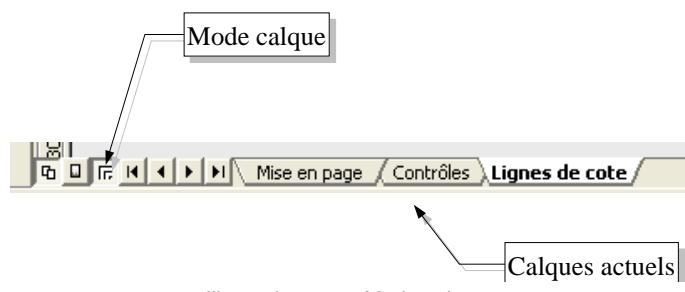


Illustration 179 - Mode calques

Lorsque vous passez en mode calques (ou couches), des onglets listant tous les calques apparaissent dans la zone inférieure de la zone de travail. Par défaut, les calques suivants sont présents: 'Mise en page', 'Contrôles', 'Lignes de côte'.

Vous pouvez activer le calque courant en cliquant sur son onglet. Les dessins sont effectués par défaut sur le calque courant. Dans l'exemple ci-dessus, les dessins seront effectués sur la couche intitulée 'Lignes de cotes'.

Si vous cliquez avec le bouton droit de la souris sur une couche, vous afficherez un menu permettant d'insérer une nouvelle couche, de supprimer une couche existante, de renommer une couche ou de la modifier. Dans ce dernier cas, une boîte de dialogue apparaîtra:

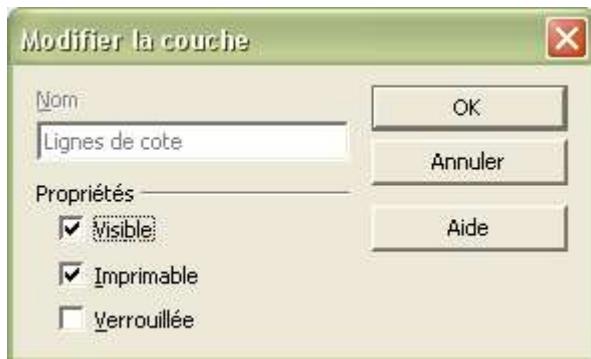


Illustration 180 - Propriété des couches

Cette boîte vous permet d'indiquer si une couche sera visible ou pas, si elle sera imprimée lors de l'impression du dessin (utile par exemple pour tracer des

annotations de dessin que vous ne souhaitez pas voir figurer à l'impression) et si elle sera verrouillée.

Tous les dessins effectués sur une couche verrouillée sont protégés contre les modifications éventuelles (déplacement, redimensionnement etc...).

Captures d'objets

OpenOffice offre quelques mécanismes destinés à aider faciliter le placement des objets sur la feuille de travail ou les uns par rapport aux autres. Les techniques que nous allons décrire ici sont appelées techniques de capture ou d'accrochage.

Toutes les fonctions mentionnées ici sont activables via la barre d'outil des options.

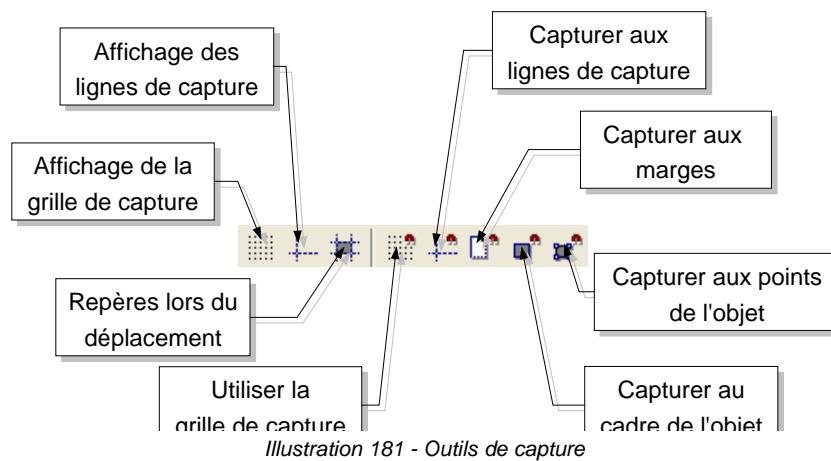


Illustration 181 - Outils de capture

Les outils de capture se décomposent en trois catégories: la grille magnétique, les lignes de capture et les points d'accrochage.

La première méthode d'accrochage s'appelle la grille magnétique. Cet outil consiste en une grille de points sur lesquels les objets peuvent s'accrocher. Pour activer la grille, il faut appuyer sur l'icône de la barre des options qui permet de la visualiser et cliquer sur l'icône pour la mettre en fonction. La surface de travail se remplit alors de petits points:

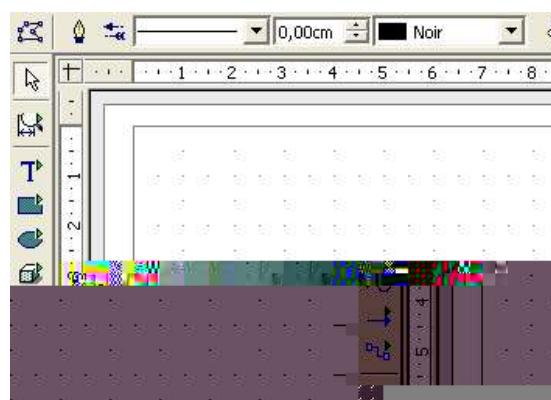


Illustration 182 - Affichage de la grille de capture

Lorsque la grille est active, le tracé de figure peut se faire facilement en s'aidant du système des points:

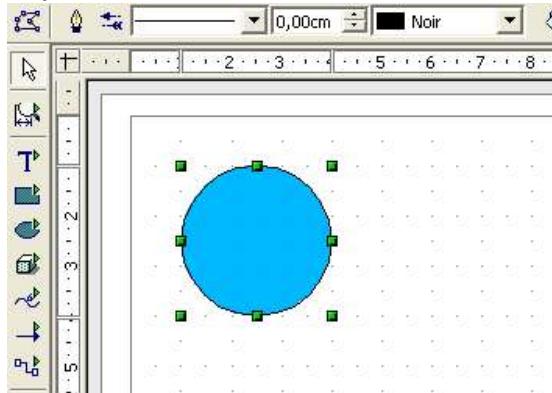


Illustration 183 - Tracé d'un objet basé sur la grille

Dans cet exemple, les poignées de l'objet sont exactement positionnées sur les points de la grille.

L'espacement entre les points est réglable via la boîte des options.



Illustration 184 - Boîte des options de la grille

Vous pouvez régler sur cette boîte:

- L'espacement horizontal et vertical des points. Les valeurs sont par défaut indiquées dans l'unité courante mais vous pouvez les modifier.
- La finesse qui correspond au nombre de subdivisions¹
- La taille en pixels de la zone de capture. Lorsque le curseur de la souris sera plus proche d'un point (ou d'une ligne) de capture que le nombre de pixels indiqué ici, la capture aura lieu.

Les lignes de capture sont des lignes horizontales ou verticales sur lesquelles vous pourrez accrocher les objets.

¹ Si vous consultez le système d'aide, la finesse est référencée sous le nom 'subdivision' qui semble une traduction plus appropriée que le terme 'finesse' qui apparaît dans la boîte des options.

Pour visualiser les lignes de capture, vous devez sélectionner l'icône  de la barre des options tandis que l'icône  permet de les activer.

Insertion dynamique d'une ligne de capture

L'ajout d'une ligne de capture se fait en faisant glisser le curseur de la souris de la règle horizontale (pour les lignes de capture verticales) ou de la règle horizontale (pour les lignes de capture horizontales) vers l'espace de travail.

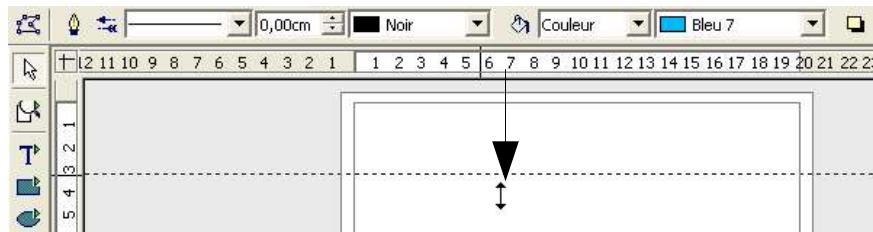


Illustration 185 - Insertion d'une ligne de capture

Dans le cas ci-dessus, une ligne de capture horizontale a été créée en faisant glisser le curseur de la souris (bouton gauche maintenu appuyé) de la règle supérieure vers la zone de travail. Il sera par la suite très simple d'aligner des objets sur cette ligne:

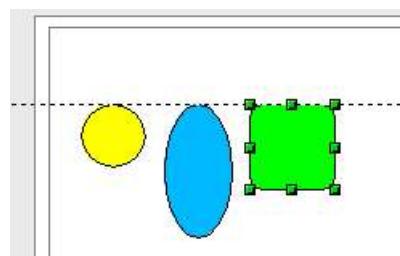


Illustration 186 - Capture d'objets sur une ligne de capture

Lorsqu'elle a été positionnée, vous pouvez toujours déplacer une ligne de capture avec l'aide de la souris. Notez que le déplacement d'une ligne de capture n'entraînera pas le déplacement des objets capturés sur cette ligne.

Insertion manuelle d'une ligne ou d'un point de capture

Vous pouvez insérer manuellement une ligne ou un point de capture via l'option du menu 'Insertion / Insérer un point/une ligne de capture...'. La boîte suivante s'affiche:

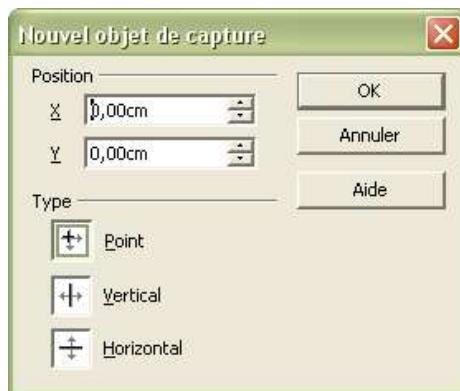
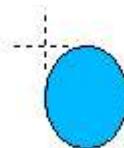


Illustration 187 - Boîte d'insertion d'un objet de capture

Elle vous permettra de préciser les coordonnées X, Y du point de capture, la coordonnée X d'une ligne de capture verticale ou la coordonnée Y d'une ligne de capture horizontale.

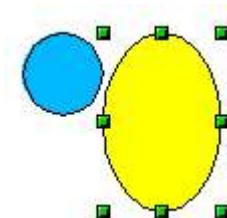
Visuellement, un point de capture se présente sous la forme suivante:



*Illustration 188 -
Point de capture*

Les points de capture peuvent eux aussi être déplacés à un autre endroit de la zone de travail avec la souris. Vous pouvez aussi éditer un point de capture en amenant la souris sur ce point ou cette ligne et en cliquant sur le bouton droit. Dans le menu qui apparaît, il ne reste plus qu'à sélectionner la fonction d'édition.

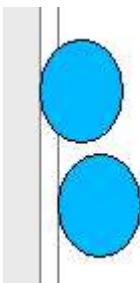
L'icône permet d'activer la capture au cadre de l'objet le plus proche du pointeur de la souris.



*Illustration 189 - Capture
au cadre d'un objet*

Dans l'exemple ci-dessus, le cercle bleu a été capturé au cadre de l'ellipse jaune (plus précisément sur l'arête gauche de ce cadre).

Si vous utilisez l'icône , vous pourrez accrocher vos objets sur les marges de la zone de travail. La capture peut se faire sur la marge ou sur la limite de la zone de travail comme le montre l'illustration suivante:



*Illustration 190 -
Capture aux
marges*

L'icône  permet d'afficher des repères verticaux et horizontaux pendant le déplacement des objets.

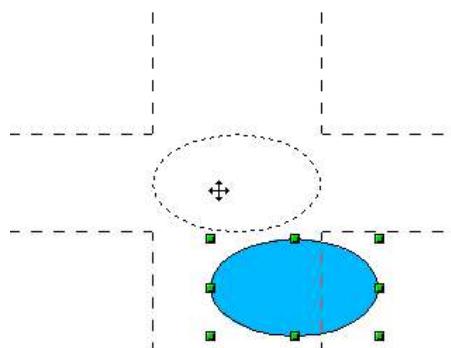


Illustration 191 - Repères lors du déplacement

Sans prétendre pouvoir rivaliser avec les ateliers de création d'images de synthèse, OpenOffice intègre des outils autorisant la réalisation de dessins tridimensionnels du plus bel effet. Nous avons étudié les différentes primitives de dessin 3D (page 177) et nous avons aussi décrit la création d'un objet 3D par profil de révolution (page 167). Nous verrons dans ce chapitre une autre méthode d'obtention d'objets 3D.

Rotation des objets 3D

La fonction de rotation agit aussi sur les objets 3D mais d'une façon différente des objets traditionnels. En effet, la rotation agira dans un espace tridimensionnel comme le suggère la figure suivante:

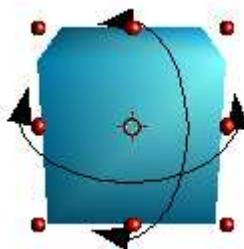


Illustration 192 - Rotation d'un objet 3D

Si vous agissez sur les poignées situées sur les bords des arêtes du rectangle de sélection, l'objet pivotera autour de l'axe horizontal ou vertical. Si vous agissez sur un des quatre coins du rectangle de sélection, l'effet sera la même et l'objet pivotera autour d'un axe perpendiculaire au plan de travail.

Comme pour les objets 3D, l'axe de rotation est matérialisé par un petit symbole que vous pouvez déplacer.

Les effets 3D

Les objets 3D disposent d'une boîte de paramétrage propre. Cette boîte s'appelle le contrôleur 3D et porte aussi le nom de boîte des effets 3D¹. Elle est accessible dans la barre des objets par appui sur l'icône . La fenêtre qui s'affiche (et qui peut être détachée pour rester visible sur l'espace de travail) est composée de plusieurs pages sélectionnables par une rangée d'icônes situées dans la partie supérieure². Ce chapitre décrira précisément ces différentes pages

Toutes les pages contiennent au moins les icônes suivantes:

- Permet de convertir un objet 2D en 3D. Cette fonction est similaire à celle qui apparaît dans l'inspecteur local des objets 2D 'Convertir en 3D'

¹ Il y a une petite incohérence entre la bulle d'aide de l'icône et le nom de la boîte affichée

² Là aussi, une petite incohérence par rapport aux autres boîtes de dialogue. Elle est probablement liée au fait que la boîte est détachable contrairement aux boîtes à onglets.



Permet d'obtenir un corps de révolution 3D (Cf page 167).



Ce bouton permet d'activer ou de désactiver l'affichage en perspective de l'objet sélectionné.



Permet d'appliquer les modifications effectuées sur la boîte à l'objet courant.

Cette page permet d'appliquer des attributs 3D à un objet quelconque. Elle se présente sous la forme suivante:

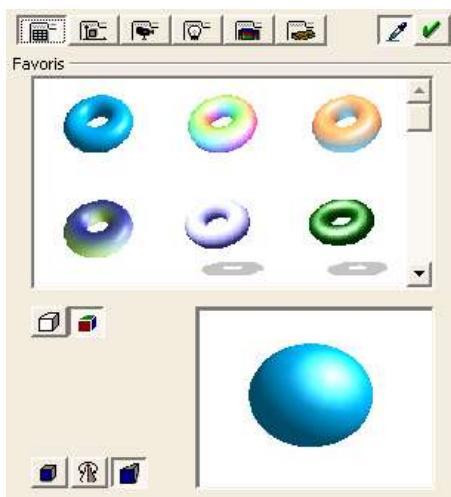


Illustration 194 - Effets 3D - Page des favoris

Pour appliquer les attributs 3D sur l'objet sélectionné, il vous suffit de choisir les attributs dans la zone 'Favoris' et de cliquer sur l'icône . Les objets exemples sont les mêmes que ceux qui sont contenus dans la galerie (dont le fonctionnement est décrit à partir de la page 244). Notez que seuls les attributs sont appliqués, les objets ne vont pas être transformés (par exemple, un cercle ne se transformera pas en tore si vous cliquez sur la première figure des favoris – il récupérera tous les attributs du tore – dont la couleur bleue).

Si l'objet sélectionné n'est pas un objet 3D, il va être automatiquement converti en objet tridimensionnel. L'opération de conversion d'un objet 2D en un objet 3D s'appelle une extrusion.

Vous pouvez l'exécuter en appliquant les attributs d'un des exemples et en cliquant sur , en cliquant directement sur le bouton ou en choisissant l'option 'Convertir en 3D' du menu local de l'objet.

Ce mécanisme permet d'obtenir de nombreuses figures:

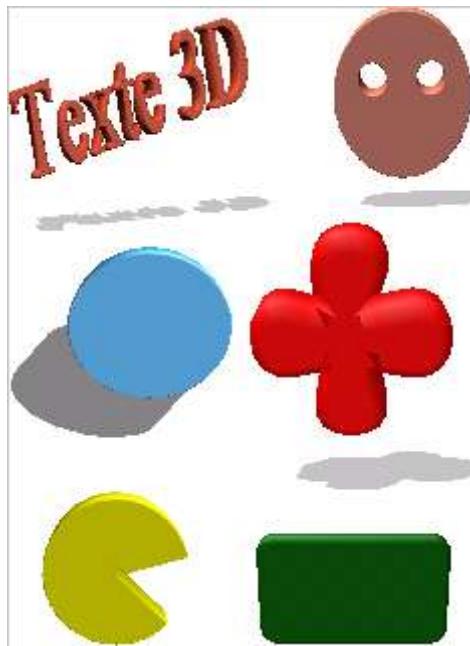


Illustration 195 - Exemples de figures 3D

Notez que la deuxième figure sur la première ligne a été obtenue par extrusion d'une combinaison de trois cercles (un grand cercle et deux petits cercles qui par combinaison sont devenus des 'trous').

-  Cette icône permet d'indiquer que l'on n'appliquera que les paramètres 3D de l'objet choisi dans les attributs sur l'objet courant. Les paramètres liés à la lumière et aux textures ne seront pas appliqués.

-  Cette icône signifie que l'on appliquera la totalité des attributs de l'objet choisi dans les favoris.

Cette page va permettre de définir les paramètres géométriques liés à un objet. Elle se présente sous la forme suivante:

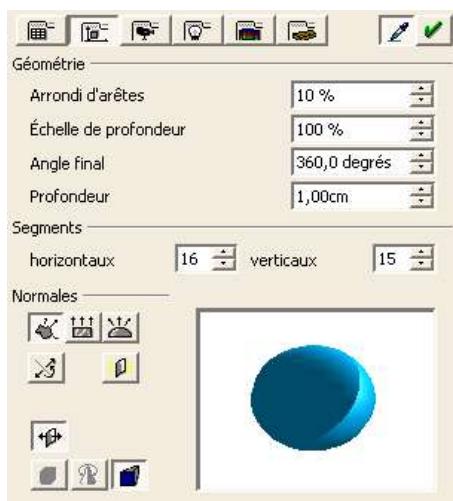


Illustration 196 - Effets 3D - page de la géométrie

Le paramètre 'Arrondi d'arêtes'¹ permet de déterminer pour une figure 2D extrudée en 3D le degré d'arrondi des arêtes de la figure. Voici deux exemples obtenus à partir d'un rectangle extrudé:

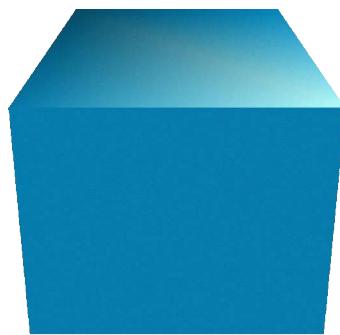


Illustration 197 - Arrondi d'arêtes de 0%

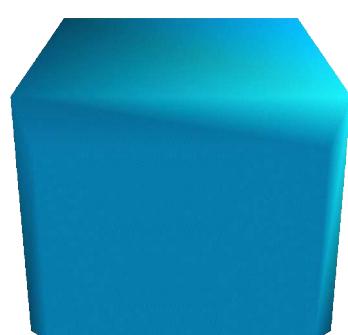


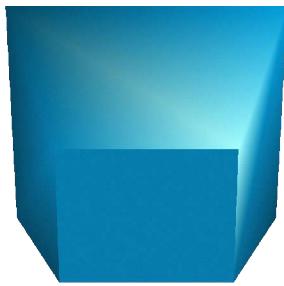
Illustration 198 - Arrondi d'arêtes de 25%

Ce paramètre est aussi particulièrement utile pour les textes extrudés en 3D.

L'échelle de profondeur va déterminer le rapport de dimension entre la face avant et arrière de l'objet. Par défaut, l'échelle est de 100% signifiant que les deux faces ont les mêmes dimensions. Si on met 50%, dans le cas du cube ci-dessus, nous obtiendrons la figure suivante:

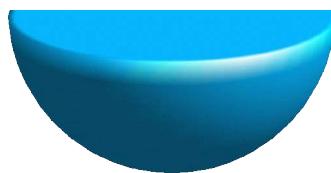
On voit très nettement que la face avant a une longueur et une largeur 50% plus petite que la face arrière. Il est possible d'avoir une échelle de profondeur supérieure à 100% et donc d'avoir une face avant plus grande que la face arrière.

¹ Dans l'aide en ligne, cette section est appelée 'Coins arrondis'



*Illustration 199 - Échelle de profondeur
de 50%*

Le paramètre 'angle final' est utile pour certaines figures obtenues par révolution d'un profil autour d'un axe (cf page 167). Il permet d'indiquer si la rotation doit être complète (360°) ou pas. Voici par exemple ce que l'on obtient sur un hémisphère avec une valeur de 180° :



*Illustration 200 - Angle de rotation de
180°*

Le paramètre 'profondeur' permet de donner la profondeur d'un objet 2D qui a été converti en 3D. Cette valeur peut être modifiée à tout instant. Ce paramètre n'est pas effectif pour les primitives 3D.

Les deux paramètres suivants (segments horizontaux / verticaux) permettent de définir le nombre de segments pour les figures arrondies. Plus le nombre de segments est important, plus la figure aura un effet 'lissé' mais plus elle sera longue à afficher. Dans l'exemple ci-dessous, la sphère de gauche est composée de 10 segments horizontaux et verticaux alors que la sphère de droite est composée de 25 segments:



*Illustration 201 - Exemple d'utilisation du nombre de
segments*

Les cinq icônes suivantes (normales) permettent d'agir sur les normales des objets. Une normale est une droite orientée traversant perpendiculairement la surface d'un objet. Voici quelques normales dessinées sur une sphère avec dix segments:

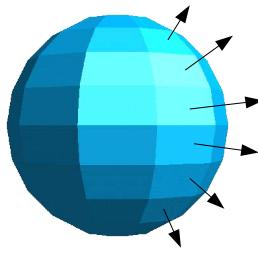


Illustration 202 - Normales d'un objet

Les normales permettent de définir l'aspect extérieur d'un objet et ses interactions avec les textures et les lumières. En agissant sur le mode de gestion des normales (on parle de mode de projection), on peut agir sur la géométrie de l'objet. Voici le descriptif des fonctions agissant sur les normales:

-  Correspond au mode de projection de la sphère ci-dessus pour laquelle chacune des facettes est visible.

-  Permet d'obtenir des facettes lissées. Dans l'exemple de la sphère, nous obtiendrons une sphère parfaite:

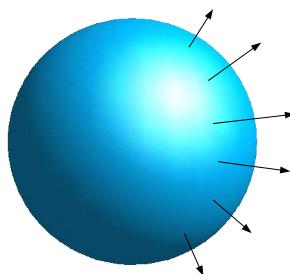


Illustration 203 - Sphère avec facettes lissées

J'ai gardé sur le dessin le tracé des normales afin de ne pas oublier que les facettes sont toujours présentes.

-  Cette icône permet de choisir le mode de projection le plus adapté à l'objet.

-  Cette icône permet d'inverser le sens des normales. Les normales permettent, comme nous l'avons vu, d'indiquer comment l'objet sera éclairé. En inversant les normales, on simulera un éclairage de l'objet de l'intérieur. Voici un exemple sur deux coupelles. La première a les normales par défaut tandis que j'ai inversé les normales de la deuxième:



Illustration 204 - Exemple d'inversion des normales

La deuxième coupelle semble éclairée de l'intérieur.



Permet d'éclairer un objet à la fois de l'intérieur et de l'extérieur.



Cette icône permet de générer des objets double face ou simple face¹. Lorsqu'on utilise la fonction d'extrusion, les objets résultants sont 'fermés' (par exemple, un carré donne un cube). Si vous utilisez cette fonction, draw tracera des objets ouverts.

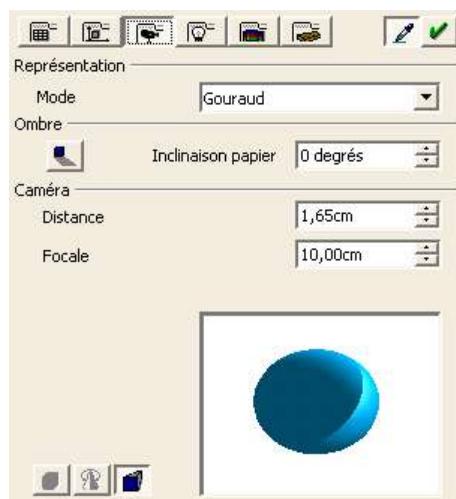
Voici un exemple obtenu à partir d'un rectangle vide:



Illustration 205 - Exemple d'objet simple face

¹ Dans la documentation online, cette icône est dénommée 'pages vis à vis'.

Cette page regroupe les paramètres liés à la représentation des objets (mode d'affichage, présence d'une ombre...). Elle se présente ainsi:



*Illustration 206 - Effets 3D - Page
'représentation'*

Le mode de présentation indique la technique que va utiliser draw pour dessiner les objets et gérer leurs interactions avec les lumières. Draw propose trois réglages: Plat, Phong et Gouraud. Ces trois techniques donnent des objets de plus ou moins bonne qualité au prix d'un temps de recalculation plus ou moins long. Plat est la technique la plus rapide mais donnant les moins bons résultats (toutes les facettes sont visibles), Phong est la technique intermédiaire et Gouraud est la technique affichant les objets avec la meilleure qualité. Par expérience, draw ne gérant pas de technique sophistiquée d'ombrage, vous pouvez garder Gouraud en permanence.



Illustration 207 - Exemples de lissages

Dans le dessin ci-dessus, le lissage plat (à gauche) est nettement le moins bon alors qu'il est plus difficile de voir la différence entre les lissages de Phong (au milieu) et de Gouraud (à droite). Tous au plus pourrait-on regarder, en examinant attentivement le dessin, que la zone ombrée est plus réaliste sur le lissage de Gouraud.

La zone suivante va vous permettre d'afficher des ombres sous les objets. L'inclinaison du papier vous permettant d'obtenir des ombres plus ou moins allongées:

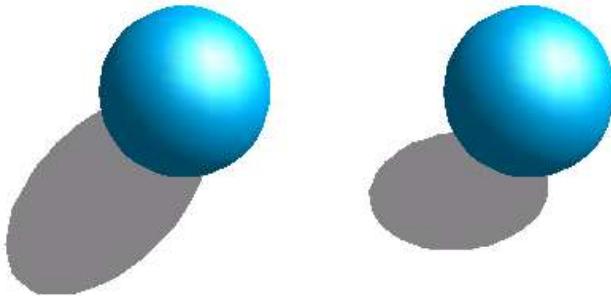


Illustration 208 - Affichage des ombres 3D

La première sphère a une ombre définie à 0° (le papier est vertical) et la deuxième sphère est projetée sur un papier incliné à 45°.

Notez sur cette représentation que l'affichage de l'ombre est cohérent avec l'éclairage de la sphère.

Les deux derniers paramètres de cette page permettent de définir la position et la focale de la caméra virtuelle permettant de visualiser l'objet. Plus la caméra est proche et plus la focale sera courte, plus l'effet de perspective sera accentué. Dans le cas du dessin ci-dessous, on a représenté la même boîte mais avec une focale plus courte pour la boîte de gauche:

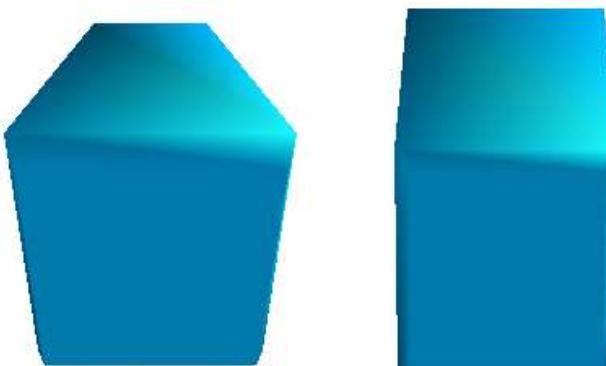


Illustration 209 - Exemples avec des focales différentes

Ces deux paramètres sont propres à l'objet (ou aux objets) sélectionné(s). Pour des raisons évidentes de cohérence, il vaut mieux que ces paramètres soient réglés de la même façon pour tous les objets appartenant à une même feuille.

Sur cette page, vous allez pouvoir définir la façon dont les objets 3D seront éclairés. Contrairement à ce qui se passe avec un logiciel 3D traditionnel, les paramètres sont modifiables pour un objet indépendamment des autres objets. Pour des raisons de cohérence, il pourra être intéressant de modifier en une fois les paramètres de tous les objets d'une page en les sélectionnant tous.

Cette boîte va vous permettre de placer huit sources de lumières destinées à éclairer l'objet courant. Pour chaque source, vous pouvez préciser sa couleur et sa position par rapport à l'objet.

Pour sélectionner la source courante, utilisez les boutons représentant une petite ampoule. Les ampoules allumées correspondent à une source de lumière active et les ampoules éteintes à une source de lumière inactive. Vous pouvez activer ou désactiver les sources lumineuses en double-cliquant sur les boutons. Le bouton enfoncé correspond à la source de lumière courante.

La couleur se sélectionne dans la liste déroulante ou via le petit bouton situé à droite de cette liste. La position de la source de lumière est définie en déplaçant avec la souris le point blanc sur le dessin situé dans la zone inférieure de la boîte.

Cette boîte vous permet aussi de choisir l'intensité de la lumière ambiante. Ce réglage est commun à toutes les sources de lumière.

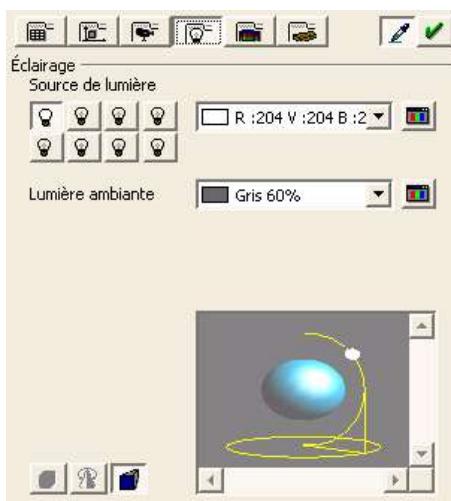


Illustration 210 - Effets 3D - Page 'Éclairage'

Cette page va vous permettre de définir les caractéristiques d'une texture bitmap appliquée à un objet.

Avant de commencer à l'utiliser, nous allons étudier les méthodes permettant d'appliquer une image bitmap sur un objet. Il existe deux façons de le faire:

1. Utiliser la boîte de remplissage avec un motif bitmap (étudiée à la page 158).
2. Passer par l'intermédiaire de la galerie. Reportez-vous à la page 244 pour voir comment l'activer. Lorsque la galerie est affichée, vous pouvez appliquer une texture sur un objet en la faisant glisser sur l'objet tout en maintenant le bouton de la souris appuyé et en pressant simultanément sur les touches [Maj] + [Control].

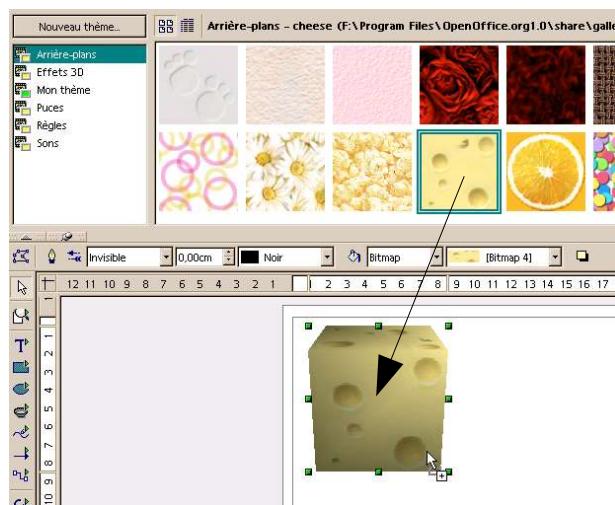


Illustration 211 - Application d'une texture

La première méthode est plus puissante dans la mesure où vous pourrez facilement régler le nombre de répétitions du motif de l'image bitmap sur la figure.

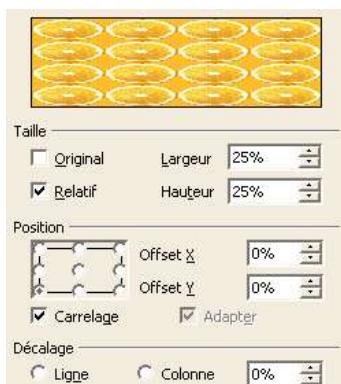


Illustration 212 - Paramètres d'application de bitmap

Voici quelques exemples obtenus en modifiant les paramètres d'application de la même image bitmap sur le même cube:

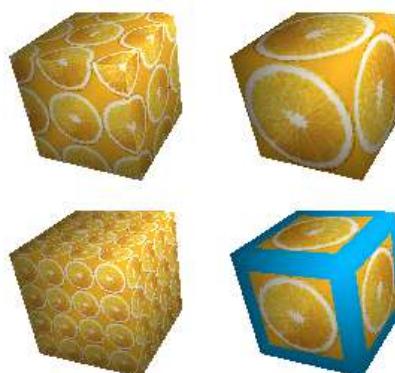


Illustration 213 - Exemples d'application de bitmaps

Note: La case à cocher 'Carrelage' sur la boîte de dialogue permet d'indiquer si l'on souhaite que le motif soit répété (cas du cube en haut et à gauche) ou pas (cas du cube en bas à droite) sur la figure.

Nous venons de voir comment modifier le nombre de répétitions du motif. Il nous reste à voir comment gérer la méthode que draw va utiliser pour projeter l'image bitmap sur l'objet. Ce rôle est dévolu à la page de gestion des textures qui se présente sous la forme suivante:

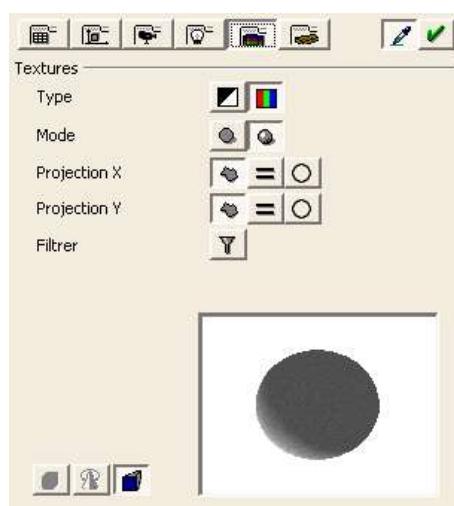


Illustration 214 - Effets 3D - Page 'textures'

Le type de projection permet d'indiquer si on souhaite que la texture bitmap soit appliquée en noir et blanc () ou en couleur ().

Le mode de projection va indiquer à draw si l'on souhaite gérer ou pas les lumières et les ombres sur l'objet courant. Dans la figure ci-dessous, la première sphère correspond au réglage 'texture seulement' et la deuxième (plus réaliste) au réglage 'texture et ombre':

Notez que l'ombre dont il s'agit ici n'a pas de rapport avec l'ombre tracée sur la feuille de travail que nous avons évoquée à la page 219. Il s'agit de la représentation sous forme ombrée du côté opposé à la lumière.

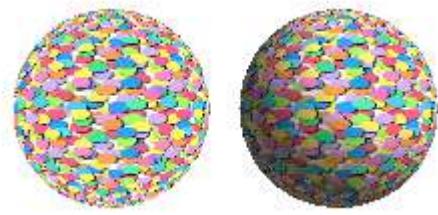


Illustration 215 - Exemple de paramétrage de texture

Les six icônes suivantes (Projection X / Y) permettent de paramétrer le type de projection utilisé pour dessiner le motif sur l'objet. Les réglages sur les axes X et Y sont les suivants: Spécifique à l'objet (■), parallèle (□) et circulaire (○). En général, le réglage par défaut 'spécifique à l'objet' donne les meilleurs résultats.

La dernière page de paramétrage des effets 3D concerne l'aspect de la surface des objets. Cette boîte permettra de donner à la surface des objets l'apparence de la surface de matériaux connus tels que le plastique ou le métal.

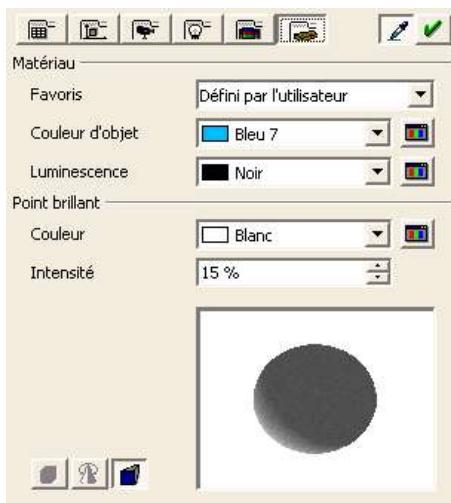


Illustration 216 - Effets 3D - Page 'Matériaux'

La liste déroulante 'Favoris' vous permettra de sélectionner directement un matériau prédéfini dans une liste. Les possibilités fournies sont les suivantes: Métal, Or, Chrome, Plastique et Bois. Voici l'effet obtenu sur une sphère (les images sont de gauche à droite définies dans l'ordre des matériaux ci-dessus):



Illustration 217 - Exemple de textures

Les paramètres que vous devrez définir sont la couleur par défaut de l'objet, sa luminescence et la couleur et l'intensité de son point brillant. Le point brillant indique la façon dont les lumières vont réagir sur l'objet.

Il est intéressant de noter que les matériaux sont compatibles avec les textures. La notion de matériel permettant en général d'accentuer l'effet des textures.

Jusqu'à présent, nous avons manipulé des dessins vectoriels. Draw intègre aussi des fonctionnalités minimales de gestion de dessins bitmaps.

Dans ce chapitre, ces fonctions seront abordées en insistant plus particulièrement sur leur interaction avec les outils de dessin vectoriel. J'insisterai assez peu sur les outils de retouche car d'autres outils (mêmes gratuits tels que The Gimp) fournissent des fonctions plus évoluées.

Voici un exemple simple de ce que l'on peut facilement obtenir à partir d'une photo rectangulaire. Avec plus de talent et d'imagination que moi, vous pourrez arriver à des résultats sympathiques.

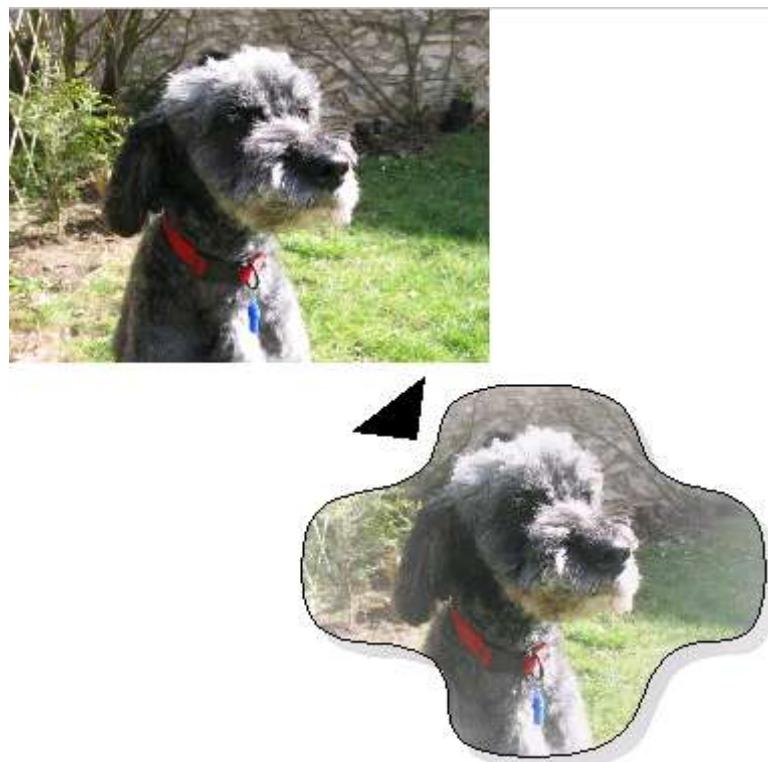


Illustration 218 - Exemple simple de manipulation d'image

La barre d'outils des images bitmaps

Lorsqu'une bitmap est sélectionnée sur l'écran de travail, une barre d'outil dédiée apparaît. Cette barre se présente sous la forme suivante:



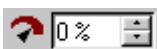
Illustration 219 - Barre d'outil de gestion des bitmaps

Il est important de noter que toutes les modifications que vous effectuerez sur une image bitmap ne concerneront que la copie de l'image bitmap dans votre document et que l'image originale ne sera pas modifiée.

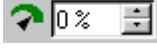


Permet de choisir le mode d'affichage par défaut de l'image. Vous avez quatre choix possibles:

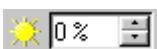
- Par défaut
- Niveau de gris: Affiche l'image bitmap avec 256 niveaux de gris
- Noir et blanc: Transforme l'image en noir et blanc
- Filigrane: Estompe les couleurs de l'image bitmap pour la rendre extrêmement claire.



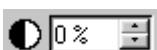
Ces trois contrôles permettent de modifier le niveau de couleur du bitmap sélectionnée. Vous pouvez agir indépendamment sur le niveau de rouge, de vert et de bleu.



Vous pouvez régler un des niveaux de -100% à + 100%. -100% correspondant à l'absence de la couleur concernée.



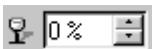
Permet de modifier la luminosité de l'image. Ce paramètre peut varier de -100% à +100%. -100% correspondant à une image totalement noire et +100% à une image totalement blanche.



Permet de régler le niveau de contraste de l'image de -100% pour une image grise sans contraste à +100% pour un contraste maximum.



Permet de régler le niveau de gamma de 0,10 à 10. Ce paramètre permet en général d'équilibrer pour un moniteur donné une image qui a été créée sur un autre moniteur.



Permet de régler la transparence de l'image de 0% (image opaque) à 100% (image totalement transparente).



Permet de rogner une image. Si vous cliquez sur cette icône, la boîte suivante s'affiche:

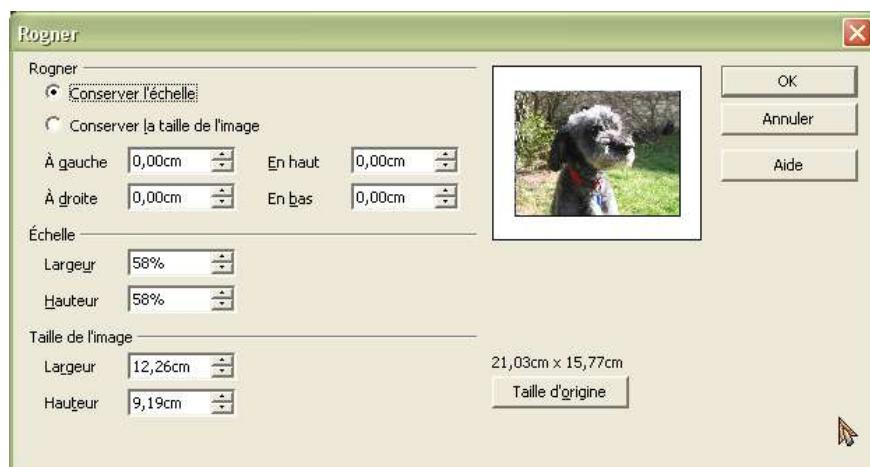


Illustration 220 - Rogner une image

Cette boîte va vous permettre d'effectuer les actions suivantes:

- Couper une partie du pourtour de l'image. Il vous suffit de modifier les valeurs 'à gauche...' de la section 'Rogner'. La zone de prévisualisation affiche nouveau cadre de l'image.
- Changer l'échelle de l'image en 100%
- Changer les dimensions du bitmap dans l'unité courante.

La palette de gestion des images bitmaps

L'icône la plus à gauche de la barre d'outil bitmap permet d'afficher la palette des filtres. OpenOffice fournit dix filtres (ce qui reste très loin du catalogue de filtres à votre disposition lorsque vous utilisez un logiciel tel que The Gimp).



Permet d'invertir les couleurs d'une image. Le résultat correspondant à ce que l'on obtient avec un négatif couleur. Cette fonction est particulièrement intéressante pour rétablir les couleurs d'un négatif scanné.



Illustration 221 - Inversion des couleurs



Permet d'adoucir une image. Dans l'exemple, j'ai cliqué plusieurs fois sur cette icône.



Illustration 222 - Adoucissement d'une image



Permet d'augmenter la netteté d'une image.



Illustration 223 - Augmentation de la netteté



Permet de supprimer les interférences d'une image en supprimant les pixels isolés.



Permet d'appliquer un effet de solarisation sur l'image. Une boîte de dialogue vous permet de choisir la valeur du seuil de solarisation.



Illustration 224 - Solarisation d'une image



Permet de simuler le vieillissement d'une image par dégradation de ses pigments. Vous pouvez choisir le degré de vieillissement (20% ici).



Illustration 225 - Vieillissement d'une image



Permet de diminuer le nombre de couleurs d'une image. Vous devez entrer dans la boîte le nombre de couleurs choisies (16 ici). Ce filtre permet de donner à une image l'apparence d'un dessin.



Illustration 226 - Diminution du nombre de couleurs



Pop'Art: Ce filtre permet d'obtenir un effet similaire à celui de l'illustration ci-dessous:



Illustration 227 - Filtre Pop'Art



Permet de simuler un dessin au fusain



Illustration 228 - Simulation de fusain



Permet de simuler une sculpture à plat.



Illustration 229 - Mise en relief d'une image



Permet d'afficher un effet de mosaïque

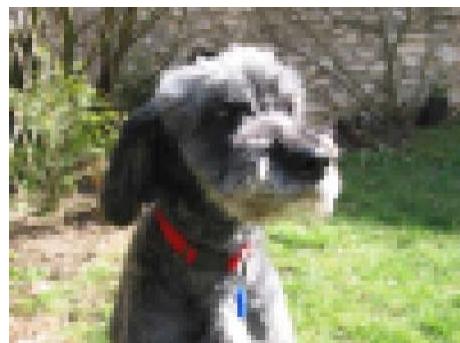


Illustration 230 - Effet de mosaïque

Draw intègre des fonctions avancées utiles dans certains cas particuliers (réalisation de dessins pour le web, échange de données etc...).

La duplication

Cette fonction va permettre la duplication en grand nombre d'une figure donnée en permettant de changer directement les paramètres des figures dupliquées.

Pour lancer cette opération, cliquez sur un objet (ou sur un groupe d'objets sélectionnés) puis choisissez l'option 'Dupliquer' dans le menu 'Édition'.

La boîte qui apparaît permet de régler les paramètres de la duplication:

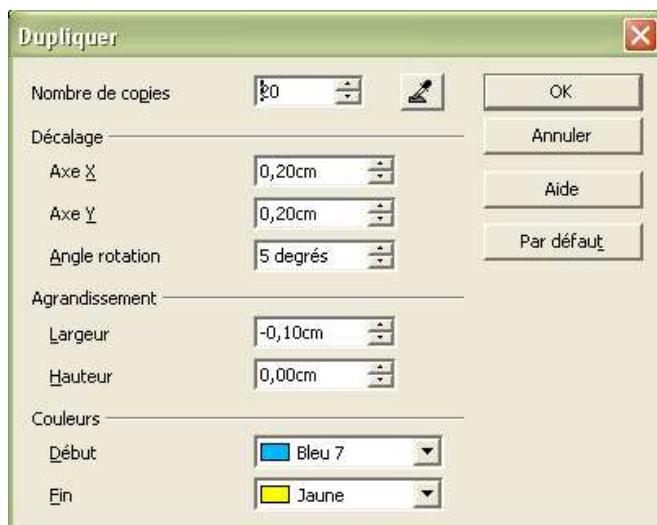


Illustration 231 - Boîte de duplication

Vous pouvez indiquer:

- Le nombre de copies
- Le décalage sur les axes X et Y entre deux copies
- L'angle de rotation entre deux copies
- Un changement de taille entre chaque copie
- La couleur de départ et de fin des copies.

Les paramètres ci-dessus appliqués à un rectangle bleu permettent d'obtenir la figure suivante:

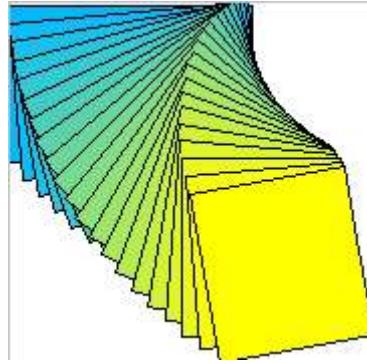


Illustration 232 - Résultat de la duplication

Le résultat de la duplication est un nouveau groupe.

Le fondu enchaîné

Cette opération permet de 'transformer' une figure en une autre, OpenOffice se changeant de calculer les étapes intermédiaires. Le résultat de cette opération est un nouveau groupe.

Voici un exemple d'utilisation:

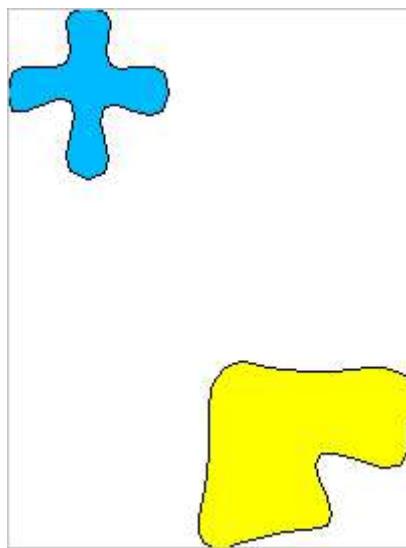


Illustration 233 - Figures de départ pour le fondu enchaîné

Nous allons partir de ces deux figures et lancer l'opération de fondu avec quatre étapes qui nous permettra d'obtenir le dessin suivant:

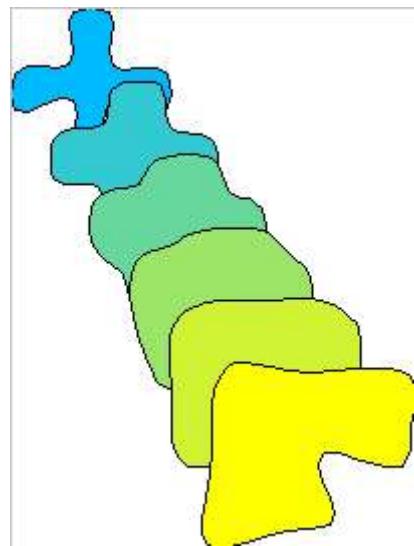


Illustration 234 - Résultat du fondu enchaîné

Pour effectuer cette opération, vous devez sélectionner deux figures et choisir la fonction 'fondu enchaîné' dans le menu 'Édition'. La boîte suivante s'affiche:

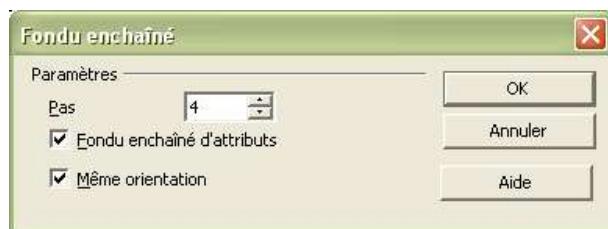


Illustration 235 - Paramètres du fondu enchaîné

Le nombre de pas correspond au nombre d'étapes intermédiaires que le programme va générer.

Si vous cochez 'Fondu enchaîné d'attributs', OpenOffice appliquera à chaque étape une transformation des attributs afin de passer en douceur des attributs de la première figure vers ceux de la deuxième. Dans l'exemple précédent, la couleur passe du bleu au jaune. Si cette case n'est pas cochée, toutes les étapes obtenues ont les attributs du premier objet.

Si vous ne cochez pas la case 'Même orientation', le fondu ne se fera pas dans un espace à deux dimensions et vous obtiendrez le résultat suivant:

On voit nettement sur cet exemple qu'OpenOffice semble faire pivoter les étapes intermédiaires dans un espace 3D pour passer de la première figure à la deuxième.

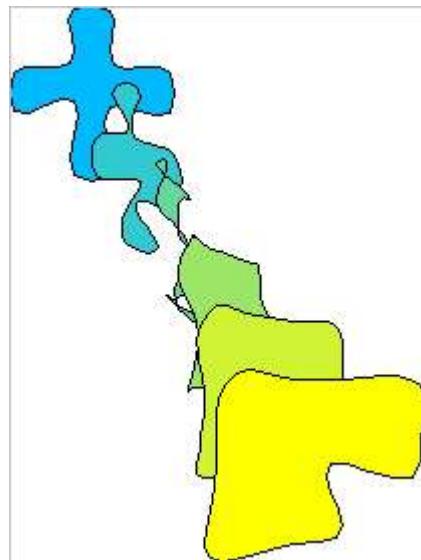


Illustration 236 - Fondu enchaîné tridimensionnel

Note: Pour que l'opération de fondu enchaîné puisse fonctionner, il faut que les objets aient des attributs proches. Il est par exemple impossible de faire un fondu enchaîné entre une figure remplie avec une couleur simple et une figure remplie avec un motif bitmap.

Fontwork

OpenOffice intègre un outil très puissant permettant de modifier la représentation des textes. Grâce à cet outil, il devient possible d'écrire des textes sur des arcs, sur des courbes et plus généralement sur n'importe quelle figure. Cet outil se présente sous la forme d'une fenêtre affichable via 'Format / FontWork'. Elle se présente sous la forme suivante:

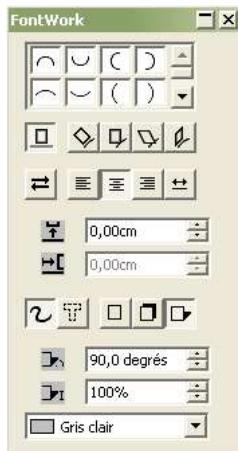


Illustration 237 - La fenêtre FontWork

Fontwork va fonctionner différemment sur un texte seul ou sur un texte tapé dans un objet.

Via copier / coller, il est possible de récupérer les textes réalisés sous Fontwork directement dans writer.

Les douze icônes situées dans la partie supérieure de la fenêtre vont vous permettre de définir l'aspect de la courbe qui suivra le texte.



Le texte suivra un demi-cercle dont l'orientation est définie par l'icône choisie. On obtient l'effet suivant:

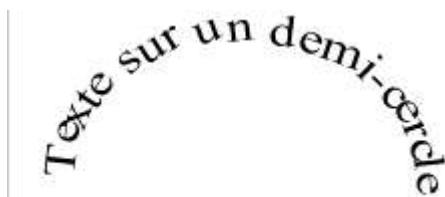


Illustration 238 - Exemple de texte sur un demi-cercle



Le principe est le même mais le texte est cette fois écrit sur un arc de cercle comme le montre l'exemple suivant:

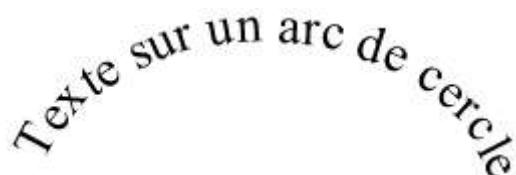
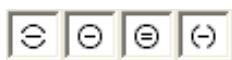


Illustration 239 - Exemple de texte sur un arc de cercle



Ce groupe d'icônes vous permet d'obtenir des effets de placement de textes sur un cercle comme le montre la figure suivante:

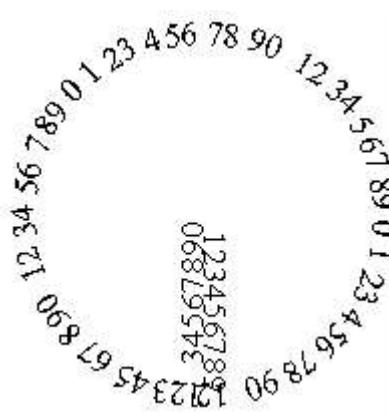


Illustration 240 - Exemple de texte sur un cercle

Ce chapitre va décrire les fonctions des différentes icônes de la boîte.



Cette icône permet de désactiver les paramètres de placement de texte sur une courbe.



Permet au texte d'épouser l'orientation de la courbe de l'objet.

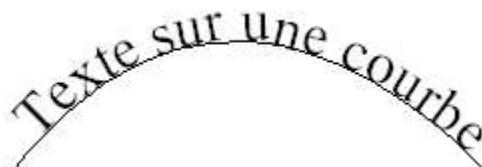


Illustration 241 - Texte adapté sur une courbe



Cette icône permet de garder les lettres individuelles verticales.

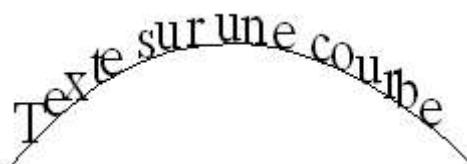


Illustration 242 - Texte vertical



Si on sélectionne cette option, le texte bascule horizontalement en fonction de sa position sur la courbe.



Illustration 243 - Texte basculé à l'horizontale



Si on sélectionne cette option, le texte bascule verticalement en fonction de sa position sur la courbe.



Illustration 244 - Texte basculé à la verticale



Cette icône permet d'inverser le sens du texte par rapport à la courbe:



Illustration 245 - Texte inversé



Le texte est cadré à gauche sur la courbe:

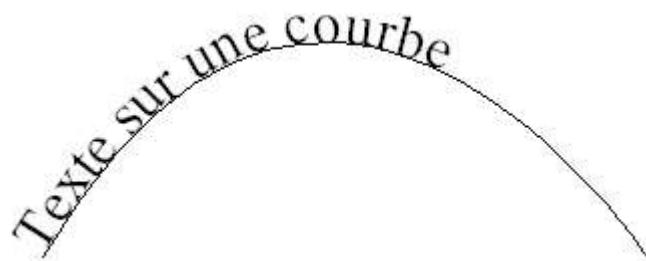


Illustration 246 - Texte aligné à gauche



Le texte est aligné par rapport au bord droit de la courbe:



Illustration 247 - Texte aligné à droite



Le texte est centré par rapport aux deux extrémités de la courbe:

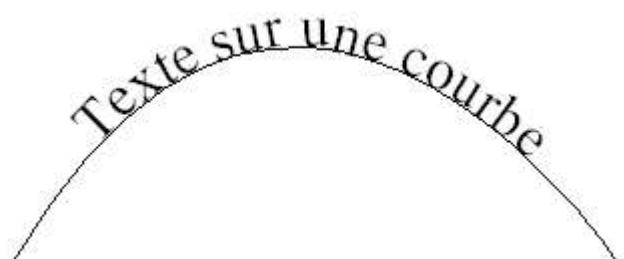


Illustration 248 - Texte centré sur la courbe



Le texte est agrandi ou diminué de façon à s'adapter aux deux bords de la courbe:



Illustration 249 - Texte adapté à la courbe



Cette icône permet d'augmenter la distance entre la courbe et le texte.



Illustration 250 - Modification de la distance entre le texte et la courbe



Permet d'indiquer le retrait entre le début du texte et le début de la courbe



Permet de rendre visible ou invisible la courbe associée au texte.

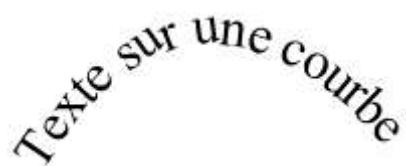


Illustration 251 - Courbe invisible



Permet d'afficher le contour des lettres du texte. Cet effet n'est apparent que si les lettres du texte sont dans une couleur différente du trait.



Illustration 252 - Contour de texte



Permet de désactiver l'affichage de l'ombre.



Permet d'afficher une ombre parallèle au texte.

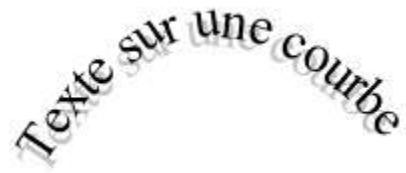
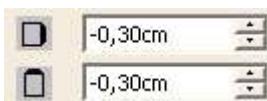


Illustration 253 - Ombre verticale sous le texte



Ces deux paramètres vous permettront de choisir la distance horizontale et verticale entre le texte et l'ombre verticale.



Permet d'afficher une ombre 'couchée' sous le texte.

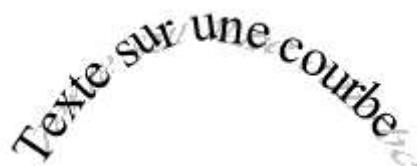
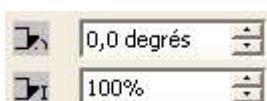


Illustration 254 - Ombre sous le texte



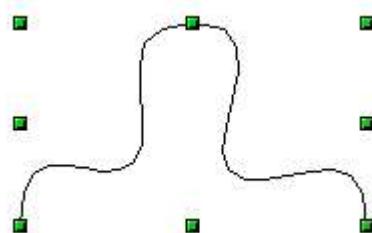
Ces deux paramètres vous permettront de sélectionner l'angle de l'ombre et sa longueur (en pourcentage du texte originel).



La liste déroulante vous permet de choisir la couleur de l'ombre dans la palette courante.

A titre de conclusion de ce chapitre, je vais maintenant décrire en quatre étapes une technique qui vous permettra d'écrire un texte et de l'appliquer sur une courbe quelconque.

1



Voici la courbe le long de laquelle nous souhaitons écrire le texte.

Illustration 255 - Courbe initiale

2



Illustration 256 - Association d'un texte à la courbe

, nous avons cliqué sur l'outil 'Texte' de la barre des instruments (T) pour écrire un texte dans la courbe. La courbe et le texte forment dans le cas un seul et même objet.

③



Illustration 257 - Écriture du texte le long de la courbe

Nous cliquons maintenant sur l'icône de la boîte fontwork et nous voyons le texte s'aligner automatiquement le long de la courbe. Comme vous le constatez sur la recopie d'écran, le texte peu ne pas tenir sur la courbe. Il faut dans ce cas utiliser l'icône .

Il est possible d'annuler l'alignement à tout moment en cliquant sur l'icône .

④

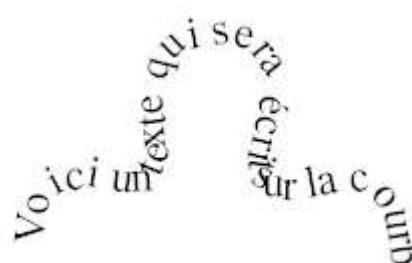


Illustration 258 - Résultat final

Il ne reste plus qu'à rendre la courbe invisible en cliquant sur le bouton pour obtenir le résultat souhaité.

Échange d'objets avec d'autres programmes

Ce paragraphe listera quelques méthodes d'échange de données entre les différents modules de la suite OpenOffice et entre draw et d'autres programmes Windows.

Comme je l'ai indiqué à plusieurs reprises dans ce document, draw est un outil de dessin vectoriel et même s'il possède quelques fonctions lui permettant de manipuler des bitmaps, il vaut mieux conserver les travaux que vous réalisez dans des formats de fichiers appropriés pour stocker des informations vectorielles.

Le gros intérêt des fichiers vectoriels apparaît lorsqu'ils sont affichés avec des rapports de zooms importants ou lorsque on les redimensionne. L'image suivante montre le même détail grossi 700 fois d'un objet en vectoriel à gauche et en bitmap à droite.

Notez bien qu'il s'agissait du même objet à la base.

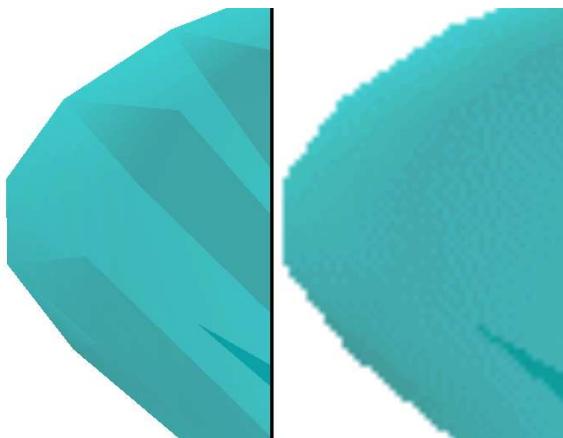


Illustration 259 - Comparaison vectoriel / bitmap

Pour convertir un objet, il vous suffit d'utiliser le menu local de cet objet qui a deux options: Convertir en métafichier et convertir en bitmap¹. La conversion d'un objet vectoriel en bitmap ne doit donc être réalisée que pour des cas où on ne peut pas faire autrement (utilisation d'une image pour le Web ou dans un logiciel n'acceptant que ce type d'image)².

L'enregistrement d'une image draw dans un format étranger se fait via l'option 'Exporter' du menu 'Fichier'. L'option par défaut vous propose d'enregistrer votre page au format html en passant par un assistant de conversion.

Cette fonction va créer autant de pages WEB que de pages incluses dans votre document. Vous pourrez choisir d'afficher ou pas les pages sous forme de frames avec un navigateur et vous pourrez choisir une page de résumé³.

En partant du fichier suivant (notez les différents onglets):

L'aspect du résultat se présentera sous la forme suivante:

1 En fonction de la complexité de l'objet, la fonction de conversion en métafichier pourra donner une image bitmap.

2 Il est à noter que l'outil de définition des imagemap pour le WEB n'accepte pas les fichiers en mode natif d'OpenOffice mais uniquement les bitmaps et les métafichiers.

3 Cet assistant est exactement le même que celui qui est intégré au logiciel de présentation d'OpenOffice

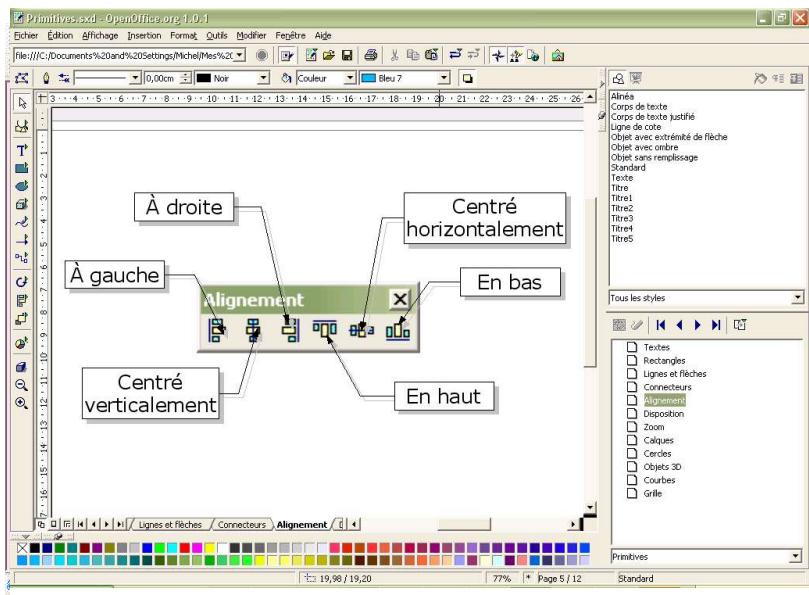


Illustration 260 - Document à transformer en HTML

Voici le descriptif des différentes pages de l'assistant. Notez que vous pouvez choisir de créer la page WEB résultante à tout instant via le bouton 'Créer' et que vous n'êtes donc pas obligé de parcourir l'assistant jusqu'à son terme.



Illustration 262 - Export HTML - Choix de la charte graphique

.La première page vous permettra de choisir une charte graphique pour toutes les pages parmi un design existant ou d'en créer un nouveau.

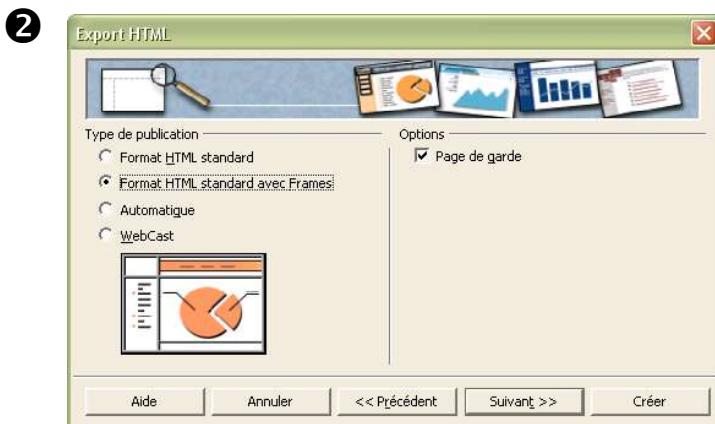


Illustration 263 - Export HTML - Choix du type de publication

Vous choisirez sur cette page le type de navigations que vous souhaiterez: via des pages simples, des frames (exemple de la copie d'écran sur l'illustration page 242) ou des pages scriptées (via ASP ou Perl, malheureusement aucun support de PHP n'étant fourni en standard).

Vous pouvez choisir ici d'insérer ou pas une page de résumé.



Illustration 264 - Export HTML - Choix du type d'image

Sur la troisième page, vous choisirez la façon dont les images seront enregistrées (en GIF ou JPEG) et les paramètres associés.

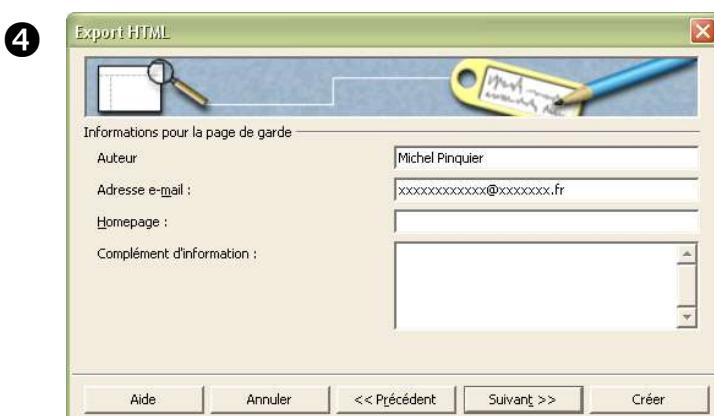


Illustration 265 - Export HTML - Informations sur l'auteur

Cette page vous permettra de saisir les coordonnées de l'auteur.

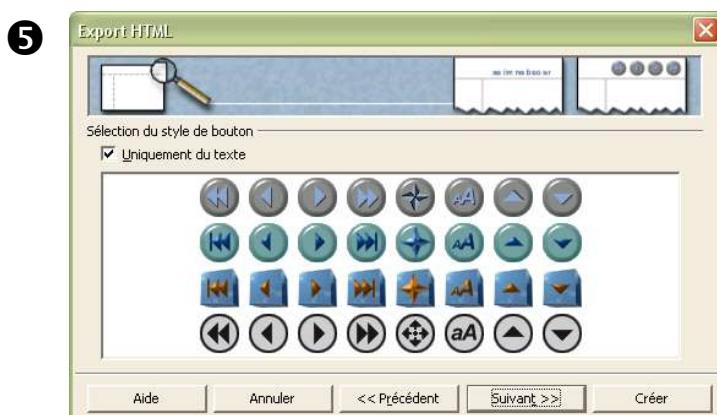


Illustration 266 - Export HTML - Choix des boutons du navigateur

Vous choisirez ici le style du bouton du navigateur à utiliser pour changer de page. Si vous n'en choisissez aucun, OpenOffice générera un navigateur texte.

6

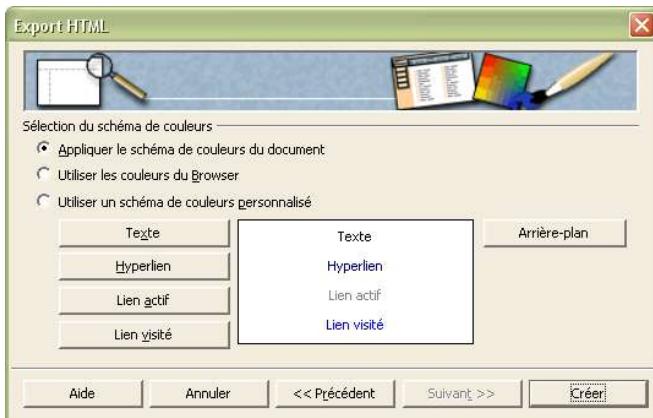


Illustration 267 - Export HTML - Définition du schéma de couleurs

Cette dernière page vous permettra de définir une charte graphique pour les pages générées. Elle vous proposera d'enregistrer votre charte afin de la retrouver sur la première page lors de la prochaine utilisation de l'export HTML.

Sur la page d'export, si vous n'utilisez pas le réglage par défaut, OpenOffice vous proposera plusieurs formats de fichiers vectoriels ou bitmap.

L'utilisation du presse-papier se fait d'une façon traditionnelle via la fonction de copier / coller. Il est à noter que les objets copiés dans le presse-papier sont convertis en métafichier. OpenOffice ne gérant pas ce format d'une façon optimale, vous verrez en général apparaître des déperditions d'informations lors de la copie dans d'autres logiciels. C'est la raison pour laquelle

(Collage spécial). *De mon point de vue, cette option est aussi souhaitable pour échanger des dessins entre les modules OpenOffice via le presse-papier.*

La galerie va vous permettre de vous constituer des bibliothèques de dessins, de réutiliser des dessins fournis avec l'installation d'OpenOffice ou de récupérer des dessins tous prêts réalisés par exemple par la communauté OpenOffice.org. Elle constitue aussi une deuxième méthode d'échange de données entre les programmes de la suite OpenOffice.

Pour afficher ou cacher la galerie, utilisez l'icône  située sur la barre d'outils principale des logiciels de la suite OpenOffice. La fenêtre de la galerie se comporte comme les autres fenêtres principales d'OpenOffice et peut être détachée, fixée etc:...

La galerie se présente sous la forme de petites vignettes:



Illustration 268 - La galerie

Réutilisation d'un objet de la galerie

Pour réutiliser un objet de la galerie dans un document OpenOffice, il suffit de le faire glisser de la galerie vers la zone de travail en maintenant le bouton de la souris appuyé.

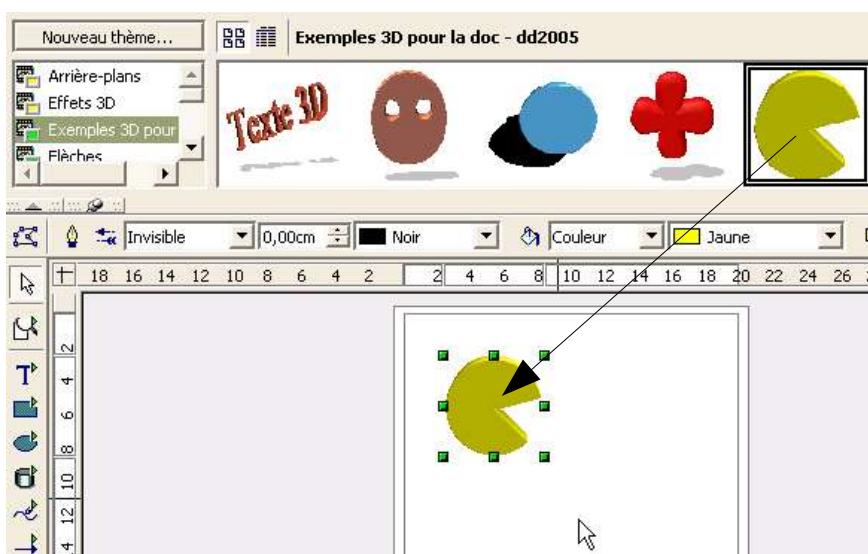


Illustration 269 - Insertion d'un objet de la galerie dans le document

Cette technique fonctionne bien entendu qu'il s'agisse d'un document dessin, d'un texte, d'un tableur ou d'une présentation.

Insertion d'un objet dans la galerie

Il existe plusieurs méthodes permettant d'insérer des objets dans la galerie. S'agissant d'objets créés avec le logiciel de dessin, je préconise une seule de ces méthodes, les autres¹ ayant le désavantage de perdre les attributs en les transformant en images bitmaps ou en métachiers².

¹ La principale autre méthode est plus adaptée pour les images bitmaps et passe par la boîte de dialogue de création d'un thème décrite dans ce chapitre. Cette méthode passe par des fichiers disque et elle ne sait bizarrement pas gérer le format de dessins de draw. Elle est donc à proscrire dans le cas qui nous intéresse.

² Le format métachier (fichiers WMF – Windows MetaFile) est un format de stockage et d'échange de dessins vectoriels propre à Windows. L'utilisation de ce format par OpenOffice n'est pas optimale car les objets copiés perdent la plupart du temps une partie de leur attribut. Je déconseille donc fortement l'utilisation de ce format dans le cadre d'OpenOffice.

①



Illustration 270 -
Dessin à copier dans
la galerie

Voici le dessin que nous avons créé et que nous souhaitons copier dans la galerie.

②

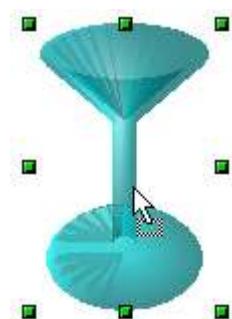


Illustration 271 - Copie
de l'objet dans la
tampon

Il faut sélectionner l'objet à copier en cliquant dessus

. Lorsque l'objet est copié dans le tampon mémoire interne de draw, le curseur de la souris change comme vous pouvez le voir sur le dessin ci-contre.

③

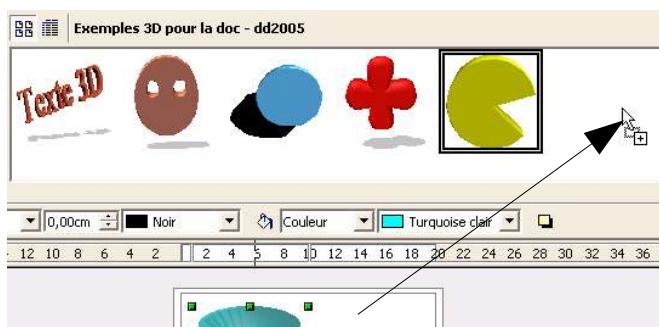


Illustration 272 - Glissement de l'image dans la galerie

, il suffit maintenant de faire glisser l'objet dans la galerie et l'opération est terminée.



Illustration 273 - Image insérée dans la galerie

Création d'un nouveau thème

Vous pouvez organiser les images que vous copiez dans la galerie en créant de nouveaux thèmes. Il vous suffit de cliquer sur le bouton 'Nouveau thème' situé en haut et à gauche de la fenêtre de la galerie et de taper le nom de la galerie dans la boîte qui s'affiche.

La boîte de dialogue qui vous invite à taper un nom de thème comporte aussi une deuxième page qui vous permet d'intégrer des fichiers directement dans la

galerie. Cette option est tout à fait inappropriée s'agissant de dessins vectoriels car elle ne fonctionne pas avec le format natif de draw.

Crédits

: Michel Pinquier

: A Sophie Gautier et Blaise Drayer pour leurs encouragements et leur aide précieuse.

Je remercie Bernard Siaud, Frédéric Hoyez et Raphaël Bolle qui ont relu les épreuves.

Je voudrais aussi remercier tous les membres de la communauté OpenOffice.org qui, de part leur travail et leur implication, contribuent à la réalisation de cette formidable suite bureautique.

Enfin, je remercie tout particulièrement ma compagne qui a su faire preuve d'une grande patience pendant la réalisation de cette documentation.

:

: 19/11/2004

Contacts: OpenOffice.org Documentation <http://fr.openoffice.org>

Historique:

Date	Version	Commentaire
27/08/02	0.97	Intégration des corrections de Barnard Siaud.
01/09/02	0.98	Intégration des corrections de Raphaël Bolle
02/09/02	0.99	Intégration des commentaires de Frédéric Hoyez. Changement du genre du mot bitmap de féminin à masculin.

Licence :

Appendix

Public Documentation License Notice

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the "License"); you may only use this Documentation if you comply with the terms of this License. A copy of the License is available at <http://www.openoffice.org/licenses/PDL.html>.

The Original Documentation is Prise en Main de Draw. The Initial Writer of the Original Documentation is Michel Pinquier (C) 2002. All Rights Reserved. (Initial Writer contact(s): michel.pinquier@club-internet.fr).

Contributor(s): _____.

Portions created by _____ are Copyright (C) _____ [*Insert year(s)*]. All Rights Reserved.
(Contributor contact(s): _____ [*Insert hyperlink/alias*]).

NOTE: The text of this **Appendix** may differ slightly from the text of the notices in the files of the Original Documentation. You should use the text of this **Appendix** rather than the text found in the Original Documentation for Your Modifications.

Table des matières

<u>À propos de ce manuel</u>	5
<u>Icônes utilisées dans ce manuel</u>	5
<u>Commentaires</u>	6
<u>Informations générales sur l'installation</u>	6
<u>Types d'installation</u>	6
<u>Configuration système requise</u>	7
<u>Généralités</u>	7
<u>Particularités concernant l'environnement d'exploitation Solaris™ (SPARC® Platform Edition) - environnement d'exploitation Solaris (SPARC)</u>	7
<u>Particularités concernant Linux</u>	7
<u>Particularités concernant Windows</u>	8
<u>Contenu des paquetages d'installation téléchargés</u>	9
<u>Opérations préalables à l'installation à partir d'un jeu téléchargé</u>	10
<u>Mise à jour d'une installation existante</u>	10
<u>Mise à jour d'une installation multiutilisateur</u>	10

<u>Installation monoutilisateur</u>	10
<u>Conditions requises pour l'installation</u>	11
<u>Démarrage de l'installation</u>	11
<u>Installation à partir d'un jeu téléchargé sur plate-forme UNIX</u>	11
<u>Installation à partir d'un jeu téléchargé sous Windows</u>	11
<u>Déroulement de l'installation</u>	13
<u>Écran de bienvenue</u>	13
<u>Informations importantes</u>	14
<u>Contrat de licence</u>	14
<u>Données d'identité</u>	14
<u>Choix du type d'installation</u>	16
<u>Sélection des composants</u>	16
<u>Répertoire d'installation</u>	17
<u>Vérification des options d'installation</u>	18
<u>Assignation des types de fichier</u>	18
<u>Installation de l'environnement d'exécution Java™</u>	20
<u>Démarrage du processus de copie</u>	21
<u>Installation : fin</u>	21
<u>Démarrage d'OpenOffice.org</u>	21
 <u>Installation multiutilisateur ou en réseau</u>	22
<u>Installation serveur</u>	23
<u>Conditions requises pour l'installation</u>	23
<u>Démarrage de l'installation</u>	23
<u>Installation à partir d'un jeu téléchargé sur plate-forme UNIX</u>	23
<u>Installation à partir d'un jeu téléchargé sur plate-forme Windows</u>	23
<u>Déroulement de l'installation</u>	24
<u>Écran de bienvenue</u>	24
<u>Informations importantes</u>	25
<u>Contrat de licence</u>	25
<u>Choix du type d'installation</u>	25
<u>Sélection des composants</u>	26
<u>Répertoire d'installation</u>	27
<u>Installation de l'environnement d'exécution Java™</u>	29
<u>Vérification des options d'installation</u>	29

<u>Démarrage du processus de copie</u>	30
<u>Installation : fin</u>	30
<u>Installation de station de travail</u>	31
<u>Conditions requises pour l'installation</u>	31
<u>Démarrage de l'installation</u>	31
<u>Installation sur plate-forme UNIX</u>	31
<u>Installation sur plate-forme Windows</u>	32
<u>Déroulement de l'installation</u>	33
<u>Écran de bienvenue</u>	33
<u>Informations importantes</u>	34
<u>Contrat de licence</u>	34
<u>Données d'identité</u>	34
<u>Type d'installation</u>	35
<u>Répertoire d'installation</u>	36
<u>Vérification des options d'installation</u>	36
<u>Attribution des types de fichier</u>	36
<u>Environnement d'exécution Java™</u>	37
<u>Démarrage du processus de copie</u>	37
<u>Installation : fin</u>	37
<u>Démarrage d'OpenOffice.org</u>	38
 <u>Appendice</u>	38
<u>Installation de l'environnement d'exécution Java™ sous Windows</u>	39
<u>Paramétrage des imprimantes, fax et polices pour les plates-formes UNIX®</u>	39
<u>Paramétrage des imprimantes</u>	39
<u>Ajout d'une imprimante</u>	40
<u>Pilotes d'imprimante dans OpenOffice.org</u>	40
<u>Import de pilote lors de l'ajout d'une nouvelle imprimante</u>	40
<u>Suppression de pilote lors de l'ajout d'une nouvelle imprimante</u>	41
<u>Modification des paramètres de l'imprimante</u>	41
<u>Attribution d'un nouveau nom à l'imprimante ou suppression</u>	42
<u>Intégration d'un périphérique fax</u>	42
<u>Connexion d'un convertisseur PostScript - PDF</u>	43
<u>Installation de polices</u>	44
<u>Ajout de polices</u>	44
<u>Suppression de polices</u>	45
<u>Attribution de nouveaux noms aux polices</u>	45
<u>Installation d'un patch dans l'environnement d'exploitation Solaris™</u>	45
<u>Modification d'une installation OpenOffice.org existante</u>	46
<u>Modification</u>	47
<u>Réparation</u>	47

<u>Suppression</u>	47
<u>Paramètres d'installation</u>	47
<u>Démarrage d'OpenOffice.org avec des paramètres</u>	48
<u>Démarrage d'OpenOffice.org à partir de lignes de commande</u>	48
<u>Paramètres de ligne de commande</u>	48
<u>Enregistrement d'OpenOffice.org</u>	50
<u>Extension d'OpenOffice.org</u>	51
<u>Installation, mise à jour et suppression d'extensions dans une installation monoutilisateur</u>	52
<u>Installation, mise à jour et suppression d'extensions dans une installation multiutilisateur</u>	52

Quand j'ai voulu écrire ma première macro pour OpenOffice, je me suis noyé dans la complexité de l'API. Pour rendre la programmation de macros plus accessible, j'ai commencé une compilation de macros accomplissant des tâches élémentaires. Quand je voyais une requête pour une macro et que je ne savais pas comment faire, je prenais cela comme un défi, je l'écrivais et la documentais. Cette quête pour comprendre comment les macros fonctionnent dans OpenOffice est concrétisée par ce document.

La version la plus récente de ce document, mis à jour très fréquemment, peut être trouvée sur mon site :

<http://www.pitonyak.org/AndrewMacro.sxw>

La date de dernière modification se trouve sur la première page. La page principale de mon site Web indique également la date et l'heure de la dernière mise à jour. Ce document est basé sur un modèle également disponible sur mon site. Le modèle n'est pas requis pour lire le document, je ne le fournis que pour les plus curieux d'entre vous.

Dans ce document, OpenOffice.org est souvent abrégé en OOo. OOBasic est le nom du langage de macro disponible dans OpenOffice. OOBasic est très proche de Visual Basic, donc connaître ce langage sera d'un grand secours.

Incluses dans OpenOffice

Ne sous-estimez pas la puissance des pages d'aide. On y trouve beaucoup d'informations sur la syntaxe des macros. Les pages d'aide sont classées en plusieurs sections. Après avoir affiché le sommaire de l'aide, déroulez la liste déroulante en haut à gauche, pour sélectionner "Aide sur OpenOffice.org Basic".

Il est également instructif d'étudier et d'utiliser les macros fournies avec OpenOffice. On y trouve par exemple des macros donnant les propriétés et les noms des méthodes supportées par un objet. Quand j'ignorais quelles étaient les propriétés et méthodes d'un objet, j'ai utilisé ces macros pour trouver ce que je pouvais sur l'objet donné. Pour cela, ouvrez un document et choisissez le menu "Outils/Macros/Macro". Dans la liste, cherchez un module intitulé "Tools". Développez ce module, vous trouverez une entrée intitulée "Debug". Ces macros implémentent la possibilité d'afficher des informations de débogage, sur les services, les attributs, etc... Regardez précisément les procédures *WritedbglInfo(document)* ou *printdbglInfo(sheet)* par exemple.

Pour utiliser la librairie de macros "Tools", il faut tout d'abord la charger. Depuis l'EDI Basic (Interface de développement) ou depuis un document, choisissez le menu "OutilsMacros/Macro", sélectionnez la librairie "Tools" et appuyez sur F5 ou cliquez sur "Exécuter".

Ressources en ligne

Il y a une grande richesse d'informations disponibles en ligne qui aide à décrypter la difficulté initiale de ce modèle de programmation. Voici quelques liens et références :

- <http://fr.openoffice.org> (lien principal) ;
- <http://www.pitonyak.org/AndrewMacro.sxw> (dernière copie à jour de ce document, en anglais)
- <http://docs.sun.com/db/doc/817-1826-10> Sun a écrit un livre sur la programmation Basic. Il contient quelques erreurs (reprises dans le fichier d'aide), mais il reste excellent pour démarrer. Très bien écrit et présenté, en anglais ;
- <http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html> (beaucoup d'informations, en anglais) ;
- <http://api.openoffice.org> Ce site, en anglais, est quelque peu difficile à utiliser mais il est vraiment complet ;
- <http://api.openoffice.org/basic/man/tutorial/tutorial.pdf> (en anglais, excellent) ;
- http://udk.openoffice.org/common/man/tutorial/office_automation.html (en anglais) ;
- <http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html> API plus anciennes ;
- <http://documentation.openoffice.org> Télécharger le "how to" suivant, en anglais :

http://documentation.openoffice.org/HOW_TO/various_topics/How_to_use_basic_macros.sxw
<http://fr.openoffice.org/Documentation/Index.html> (excellent site en français).

Télécharger le "how to" suivant, en français :

http://fr.openoffice.org/Documentation/How-to/Basic/ht01_basic.sxw

- <http://docs.sun.com/db/coll/999.2?q=star+office> (documentation Sun originale).

Suivent d'autres sites contenant des exemples de code et généralement des informations intéressantes.

- <http://kienlein.com/pages/oo.html> (exemples, anglais, allemand) ;
- http://www.darwinwars.com/lunatic/bugs/oo_macros.html (Exemples) ;
- <http://disemia.com/software/openoffice/> (exemples, en anglais) ;
- http://sourceforge.net/project/showfiles.php?group_id=43716 (Exemples et documentations, en anglais et en italien).

Quand je cherche une information spécifique, j'utilise habituellement le "Guide du développeur", le tutoriel et, ultimement, je lance une requête Google, comme par exemple "cursor OpenOffice". Quand je veux affiner la recherche, j'utilise "site: api.openoffice.org cursor" et je peux ainsi voir à quoi ressemble l'interface de cette fonctionnalité.

Si j'ai une idée du nom du package, j'essaye de deviner sa localisation sur le web. L'idée est que vous trouverez module-ix.html plutôt que index.html. Ainsi

vous pourrez trouver

<http://api.openoffice.org/docs/common/ref/com/sun/module-ix.html>

mais pas

<http://api.openoffice.org/docs/common/ref/com/sun/index.html>

A partir de la page web d'un module, vous pouvez suivre les liens vers les sous-modules.

Traductions

http://www.pitonyak.org/AndrewMacro.sxw	Anglais
http://fr.openoffice.org/Documentation/Guides/Indexguide.html	Français

OOBasic est similaire à Visual Basic, aussi la connaissance de ce langage sera d'un grand secours. Le langage macro de OOo ressemble beaucoup à celui de Microsoft Office car ils sont tous deux basés sur le Basic. Ces deux langages permettent d'accéder au modèle objet respectif de la suite à laquelle ils appartiennent. Donc, en dehors de la syntaxe du langage elle-même, la comparaison devra s'arrêter là et on peut raisonnablement dire que les paradigmes des deux suites sont différents.

Ma première Macro : Hello World

Ouvrir un nouveau document OOo. Aller dans le menu Outils/Macros/Macro. Ce menu ouvre la boite de dialogue des macros. Du côté gauche de cette boite de dialogue, rechercher le document ouvert préalablement. Il portera probablement le nom « Sans nom1 ». Cliquer sous ce nom où il est écrit "standard ". Cliquer sur le bouton "Nouveau" pour créer un nouveau module. Toujours utiliser le nom "Module1" proposé par défaut n'est probablement pas le bon choix. Quand vous avez plusieurs documents ouverts contenant tous un "Module1" il risque d'être difficile de les différencier. Pour l'instant, intituler ce premier module "MonPremierModule". L'interface OOBASIC (IDE) s'ouvre alors. Modifier le code pré-inscrit pour qu'il ressemble à ceci :

```
REM ***** BASIC *****
Sub Main
    Print "Hello World"
End Sub
```

appuyer sur le bouton "Exécuter" de la barre d'outils et... hop, vous venez d'exécuter votre première macro.

Regrouper le code

OOBasic est basé sur des sous-routines et des fonctions. Elles sont implémentées avec les mots clés Sub et Function. Elles seront dénommées comme "Procédures" tout au long de cet ouvrage. Chaque procédure donne accès à un ensemble de fonctionnalités et peut appeler d'autres procédures (la récursivité n'est permise qu'à partir de la version 1.1). La différence entre une Sub et une Function est que cette dernière peut retourner une valeur et donc à ce titre est autorisée à figurer à droite de l'affectation d'une variable.

```
Sub HelloWorld
    MsgBox HelloWorldString()
End Sub
Function HelloWorldString() As String
    HelloWorldString = "Hello World"
End Function
```

Une collection de procédures est contenue dans un module. Un document peut contenir plusieurs modules qui peuvent également exister indépendamment du document (module global). Une collection de modules est contenue dans une librairie (library).

Créer et accéder aux objets dans OOBasic

Dans OpenOffice.org Basic, on accède au document ouvert existant ou à l'application par l'intermédiaire des deux variables ThisComponent et StarDesktop . Une fois que vous avez l'objet document, vous pouvez accéder à son interface. Voici un exemple simple :

```
Sub Example
Dim oText As Variant, oDoc As Variant
oDoc = ThisComponent      ' Récupère le document actif
oText = oDoc.Text          ' Accède au service TextDocument
```

Pour charger un document existant ou un nouveau document vierge, l'objet desktop possède la méthode loadComponentFromURL(). Les structures OOBasic peuvent être créées pendant leur déclaration comme ceci (*NdT : attention, la casse de PropertyValue est importante*) :

```
Dim args1(2) as new com.sun.star.beans.PropertyValue 'Array 0 to 2
```

Pour créer une instance d'un service, utiliser la méthode globale createUnoService().

```
Sub PerformDispatch(vObj, uno$)
Dim vParser, vDisp
Dim oUrl As New com.sun.star.util.URL
oUrl.Complete = uno$

vParser = createUnoService("com.sun.star.util.URLTransformer")
vParser.parseStrict(oUrl)

vDisp = vObj.queryDispatch(oUrl,"",0)
If (Not IsNull(vDisp)) Then vDisp.dispatch(oUrl,noargs())
End Sub
```

Le guide du développeur préconise d'utiliser le type Variant plutôt que le type Object. Voir la section [Dois-je utiliser Object ou Variant](#) pour plus de détail.

Vous pourrez rencontrer ce genre de code :

```
createUnoService("com.sun.star.frame.Desktop")
```

Cette ligne crée une instance de OOo. La variable StarDesktop étant en principe déjà disponible, ne vous en souciez pas et utilisez cette variable.

Services

Dans OOo, la plupart des fonctionnalités sont implémentées en tant que services. Conceptuellement, un service est défini par une interface. Vous n'avez pas besoin de savoir comment un service est implémenté, seulement comment l'appeler. Un très bon exemple de service simple est le "Searchable" service. décrit ici :

<http://api.openoffice.org/docs/common/ref/com/sun/star/util/XSearchable.html>
Ce lien indique que les méthodes suivantes sont supportées :

createSearchDescriptor	Crée un descripteur définissant comment chercher.
findAll	Cherche toutes les occurrences
findFirst	Cherche la première occurrence
findNext	Cherche l'occurrence suivante

Ce lien indique également que tout objet TextDocument supporte cette interface. Il ne donne aucune indication sur comment la recherche est effectuée mais il dit comment utiliser une recherche. Il donne également des informations sur les valeurs retournées. [Voir la section Rechercher et remplacer](#)

Si vous manipulez un objet et que vous voulez savoir s'il supporte un certain service, vous pouvez appeler la méthode SupportsService(NomDuService). Par exemple, si vous voulez savoir si l'objet vDoc est un document texte, vous pouvez utiliser le code suivant :

```
If vDoc.supportsService("com.sun.star.sheet.TextDocument") Then
```

Malgré le risque d'évoquer des sujets un peu trop tôt dans ce document, faites attention à la gestion d'erreur comme mentionné plus loin. Que se passe-t-il si vDoc est de type Null ? Si vDoc est une structure ou s'il n'implémente pas la méthode SupportsService() ? Pour vous assurer contre de telles erreurs, pensez à initialiser un gestionnaire d'erreurs en utilisant la syntaxe "On Error" .

Vous trouverez plus d'informations sur comment obtenir des renseignements complémentaires à l'adresse :

<http://api.openoffice.org/docs/common/ref/com/sun/star/lang/XServiceInfo.html>

Si vous avez besoin d'un service et que vous n'avez pas une instance disponible, vous pouvez utiliser la méthode globale createUnoService(). Vous en trouverez des exemples tout au long de ce document.

Examiner un objet

Il est utile connaître le type d'un objet afin de pouvoir l'utiliser correctement. Voici une brève liste des méthodes d'inspection :

IsArray	Le paramètre est-il un tableau ?
IsEmpty	Le paramètre est-il un variant non initialisé ?
IsNull	Le paramètre ne contient-il aucune donnée ?
isObject	Le paramètre est-il un objet OLE ?
isUnoStruct	Le paramètre est-il une structure UNO ?

TypeName	Quel est le nom du type de l'argument ?

Le nom du type du paramètre va donner ces informations :

Variant	“Empty” ou le nom de l'objet contenu
Object	“Object” même si il est null. Idem pour les structures
regular type	Un nom de type régulier comme “String” ou “Long”
array	Nom suivi de parenthèses “()”

La macro suivante illustre ces méthodes :

```

Sub TypeTest
    Dim oSimpleFileAccess
    Dim aProperty As New com.sun.star.beans.Property
    oSimpleFileAccess = CreateUnoService( "com.sun.star.ucb.SimpleFileAccess" )
    Dim v, o As Object, s As String, ss$, a(4) As String
    ss = "Empty Variant: " & GetSomeObjInfo(v) & chr(10) &
        "Empty Object: " & GetSomeObjInfo(o) & chr(10) &
        "Empty String: " & GetSomeObjInfo(s) & chr(10)
    v = 4
    ss = ss & "int Variant: " & GetSomeObjInfo(v) & chr(10)
    v = o
    ss = ss & "null obj Variant: " & GetSomeObjInfo(v) & chr(10) &
        "struct: " & GetSomeObjInfo(aProperty) & chr(10) &
        "service: " & GetSomeObjInfo(oSimpleFileAccess) & chr(10) &
        "array: " & GetSomeObjInfo(a())
    MsgBox ss, 64, "Type Info"
End Sub

REM Retourne des informations de base pour le paramètre.
REM Cela retourne également la dimension d'un tableau.
Function GetSomeObjInfo(obj) As String
    Dim s As String
    s = "TypeName = " & TypeName(vObj) & CHR$(10) &
        "VarType = " & VarType(vObj) & CHR$(10)
    If IsNull(vObj) Then
        s = s & "IsNull = True"
    ElseIf IsEmpty(vObj) Then
        s = s & "IsEmpty = True"
    Else
        If IsObject(vObj) Then
            On Local Error GoTo DebugNoSet
            s = s & "Implementation = " & NotSafeGetImplementationName(vObj) & CHR$(10)
            DebugNoSet:
            On Local Error Goto 0
            s = s & "IsObject = True" & CHR$(10)
        End If
        If IsUnoStruct(vObj) Then s = s & "IsUnoStruct = True" & CHR$(10)
        If IsDate(vObj) Then s = s & "IsDate = True" & CHR$(10)
        If IsNumeric(vObj) Then s = s & "IsNumeric = True" & CHR$(10)
        If IsArray(vObj) Then
            On Local Error Goto DebugBoundsError:
            Dim i%, sTemp$
            s = s & "IsArray = True" & CHR$(10) & "range = ("
            Do While (i% >= 0)
                i% = i% + 1
        End If
    End If
End Function

```

```

sTemp$ = LBound(vObj, i%) & " To " & UBound(vObj, i%)
If i% > 1 Then s = s & ","
s = s & sTemp$
Loop
DebugBoundsError:
On Local Error Goto 0
s = s & ")" & CHR$(10)
End If
End If
adp_GetObjTypeInfo = s
End Function

REM Cela crée un gestionnaire d'erreur pour gérer le problème
REM et renvoie quelque-chose quoiqu'il arrive !
Function SafeGetImplementationName(vObj) As String
On Local Error GoTo ThisErrorHere:
SafeGetImplementationName = NotSafeGetImplementationName(vObj)
Exit Function
ThisErrorHere:
On Local Error GoTo 0
SafeGetImplementationName = "*** Unknown ***"
End Function

REM Le problème est que si cette fonction est appelée et que le type vObj
REM ne supporte PAS l'appel getImplementationName(), alors je reçois
REM une erreur "Object variable not set" lors de la définition de la fonction.
Function NotSafeGetImplementationName(vObj) As String
NotSafeGetImplementationName = vObj.getImplementationName()
End Function

```

Vous avez recherché son type et vous savez que vous avez un objet. Si vous avez une structure, il vous faut découvrir de quel type elle est pour connaître ses propriétés. Je fais ça généralement en ligne en cherchant sur le site web dédié à l'API. Les objets UNO supportent en général le service ServiceInfo comme mentionné plus haut. La méthode getImplementationName() de l'objet retourne le nom complet de l'objet. A partir de là, je recherche dans le guide du développeur ou sur Google pour trouver plus d'informations. La méthode getSupportedServiceNames() retourne la liste de toutes les interfaces supportées par l'objet. Pour découvrir ce que peut faire un objet, vous pouvez appeler ces trois méthodes :

```

MsgBox vObj.dbg_methods
MsgBox vObj.dbg_supportedInterfaces
MsgBox vObj.dbg_properties

```

Comment procéder avec les retours de type UNO

Utiliser les informations de Xserviceinfo comme mentionné auparavant ?

Débogage et vérification de macros

Il est peut être difficile de déterminer quelles méthodes et propriétés sont disponibles pour un objet. Les méthodes de cette section devraient vous y aider.

Dans OOo, la plupart des fonctionnalités sont définies par des services. Pour déterminer le type de document, regardez si le service les supportent. La macro suivante utilise cette manière de faire. Je suppose que c'est plus sûr qu'appeler ThisComponent.getImplementationName().

```
'*****  
'Auteur : Inclu dans OpenOffice.org  
'  
Function GetDocumentType(oDoc)  
    On Local Error GoTo NODOCUMENTTYPE  
    If oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then  
        GetDocumentType() = "scalc"  
    Elseif oDoc.SupportsService("com.sun.star.text.TextDocument") Then  
        GetDocumentType() = "swriter"  
    Elseif oDoc.SupportsService("com.sun.star.drawing.DrawingDocument") Then  
        GetDocumentType() = "sdraw"  
    Elseif oDoc.SupportsService("com.sun.star.formula.FormulaProperties") Then  
        GetDocumentType() = "smath"  
    End If  
    NODOCUMENTTYPE:  
    If Err <> 0 Then  
        GetDocumentType = ""  
        Resume GOON  
        GOON:  
    End If  
End Function
```

Cette macro, écrite à partir d'une décrite plus haut par Alain Viret [Alain.Viret@bger.admin.ch], retourne le filtre d'export pdf. Notez qu'elle peut aussi contrôler à partir des documents Impress.

```
Function GetPDFFilter(oDoc)  
    On Local Error GoTo NODOCUMENTTYPE  
    If  
    oDoc.SupportsService("com.sun.star.presentation.PresentationDocument")  
    Then  
        GetPDFFilter() = "impress_pdf_Export"  
    Elseif oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument")  
    Then  
        GetPDFFilter() = "calc_pdf_Export"  
    Elseif oDoc.SupportsService("com.sun.star.text.TextDocument") Then  
        GetPDFFilter() = "writer_pdf_Export"  
    Elseif oDoc.SupportsService("com.sun.star.drawing.DrawingDocument")  
    Then  
        GetPDFFilter() = "draw_pdf_Export"  
    Elseif oDoc.SupportsService("com.sun.star.formula.FormulaProperties")  
    Then  
        GetPDFFilter() = "math_pdf_Export"  
    End If  
    NODOCUMENTTYPE:  
    If Err <> 0 Then
```

```

GetPDFFilter = ""
Resume GOON
GOON:
End If
End Function

```

Cette excellente procédure affiche les noms des méthodes et propriétés supportées par un objet. Si le second paramètre est une chaîne vide, les noms des méthodes sont affichés. D'autres valeurs provoquent l'affichage des noms des propriétés. Comme cet affichage est fréquemment très long, la liste est découpée en morceaux.

```

'*****
'Une procédure pour afficher toutes les méthodes et propriétés d'un objet
'*****
'Author : Tony Bloomfield
'email : tonyb.lx@btinternet.com
'Modification : hal@thresholddigital.com pour supporter les services et vérifier qu'oObj existe
Sub DisplayMethods(oObj As Object, SWhat As String)
    Dim sMethodList As String, sMsgBox As String
    Dim fs, ep As Integer
    Dim i As Integer
    Dim EOL As Boolean
    If IsNull(oObj) Then
        print "L'objet n'existe pas."
    Else
        If SWhat = "m" Then
            sMethodList = oObj.DBG_Methods
        Elseif SWhat = "s" Then
            sMethodList = oObj.DBG_SupportedInterfaces
        Elseif SWhat = "p" Then
            sMethodList = oObj.DBG_Properties
        End If
        fs = 1
        EOL = FALSE
        While fs <= Len(sMethodList)
            sMsgBox = ""
            For i = 0 To 15
                ep = InStr(fs, sMethodList, ";")
                If ep = 0 Then
                    ep = Len(sMethodList)
                End If
                sMsgBox = sMsgBox & Mid$(sMethodList, fs, ep - fs) & Chr$(13)
                fs = ep + 1
            Next i
            MsgBox sMsgBox
        Wend
    End If
End Sub

```

Notez que, bien que ces méthodes fonctionnent pour moi, vous n'avez pas les boîtes de dialogue, donc ça ne marchera pas pour vous. Même si je les

fournissais, c'est loin d'être terminé. J'espère pouvoir terminer cet explorateur d'objet si j'arrive à m'en sortir avec les boîtes de dialogue.

```
Option Explicit
Sub ExampleCreateDialog
    Dim oObj As Object
    oObj = ThisComponent

    Dim oDlgDesc As Object, oDlg As Object
    DialogLibraries.LoadLibrary("Standard")
    ' Récupère la description de la boîte de dialogue à partir de la bibliothèque
    oDlgDesc = DialogLibraries.Standard
    Dim oNames(), i%
    oNames = DialogLibraries.Standard.getElementNames()
    For i = LBound(oNames()) To UBound(oNames())
        MsgBox "How about " + oNames(i)
    Next
    MsgBox(DialogLibraries.dbg_methods)
    MsgBox(DialogLibraries.dbg_properties)
    MsgBox(DialogLibraries.dbg_supportedInterfaces)
    oDlgDesc = DialogLibraries.Standard.ObjectViewer
    ' Instancie la boîte de dialogue
    oDlg = CreateUnoDialog( oDlgDesc )

    Dim oModel As Object, oListBox As Object
    dim iCount As Integer, iPos As Integer, s$, j%, sNew$
    oModel = oDlg.Model

    AddToListBox(oObj.DBG_Properties, oDlg.GetControl("PropertiesBox"), ";")
    AddToListBox(oObj.DBG_Methods, oDlg.GetControl("MethodsBox"), ";")
    AddToListBox(oObj.DBG_SupportedInterfaces, oDlg.GetControl("InterfacesBox"), chr(10))

    ' Affiche la boîte de dialogue
    oDlg.execute
End Sub

Sub AddToListBox(s$, oList As Object, sep$)
    Dim iCount%, j%, iPos%, sNew$
    iCount = oList.ItemCount
    iPos = InStr(1, s, ":") + 1
    iPos = FirstNonWhiteSpace(iPos, s)
    Do While iPos <= Len(s)
        j = InStr(iPos, s, sep)
        If j = 0 Then j = Len(s)
        iPos = FirstNonWhiteSpace(iPos, s)
        sNew = Mid$(s, iPos, j - iPos)
        oList.addItem(sNew, iCount)
        iPos = j + 1
        iCount = iCount + 1
    Loop
End Sub
Function FirstNonWhiteSpace(ByVal i%, s$) As Integer
    If i <= Len(s) Then
        Do While IsWhiteSpace(Asc(Mid$(s, i, 1)))
            i = i + 1
            If i > Len(s) Then
                Exit Do
            End If
        Loop
    End If
    FirstNonWhiteSpace = i
End Function
```

Il suffit de faire tourner la macro pour le croire. Elle crée un nouveau classeur et remplit les feuilles avec les informations sur l'objet. J'ai beaucoup modifié cette macro car elle utilisait des variables globales qui interféraient avec mes propres variables locales. Cela m'a permis de l'utiliser dans mon code existant.

```
'Author : Hermann Kienlein
'email : info@kienlein.com
'online :http://www.kienlein.com/pages/oo.html
Option Explicit ' Oblige toute variable à être déclarée avant utilisation
Sub Main
    MainObjectDisplay(ThisComponent)
End Sub

' -----
'Créé un nouveau classeur et de nouvelles feuilles de calcul dedans. Nomme les feuilles
'et insère les informations dedans
Sub MainObjectDisplay(oObject As Object)
    Dim oInfo As Object, oDeskNeu As Object, oNewDoc As Object
    Dim sNewUrl As String, nSheetsUsed As Long
    Dim sInterfaces As String
    Dim NoArgs()

    nSheetsUsed = 0
    oDeskNeu = createUnoService("com.sun.star.frame.Desktop")
    'sNewUrl = "staroffice.factory:scalc"      ' Cette ligne ne marchait pas
    sNewUrl = "private:factory/scalc"
    oNewDoc = oDeskNeu.loadComponentFromURL( sNewUrl, "_blank", 0, NoArgs())

    ObjInfo(oNewDoc, nSheetsUsed, "ThisComponent", oObject)

    On Local Error GoTo AllDone
    sInterfaces = oObject.dbg_supportedinterfaces

    On Local Error GoTo NoController
    If InStr(sInterfaces, "com.sun.star.frame.XModel") <> 0 Then
        oInfo = oObject.getCurrentController()
        ObjInfo(oNewDoc, nSheetsUsed, "getCurrentController", oInfo)
    End If
    NoController:

    On Local Error GoTo NoDocInfo
    '?
    If InStr(sInterfaces, "com.sun.star.document.XDocumentInfoSupplier") <> 0 Then
        oInfo = oObject.getDocumentInfo()
        ObjInfo(oNewDoc, nSheetsUsed, "getDocumentInfo", oInfo)
    End If
    NoDocInfo:

    On Local Error GoTo NoSelection
    '?
    If InStr(sInterfaces, "com.sun.star.frame.XSelectionSupplier") <> 0 Then
        oInfo = oObject.getCurrentSelection()
        ObjInfo(oNewDoc, nSheetsUsed, "getCurrentSelection", oInfo)
    End If
    NoSelection:

    On Local Error GoTo NoLibraryContainer
    If InStr(sInterfaces, "com.sun.star.script.XStarBasicAccess") <> 0 Then
        oInfo = oObject.getLibraryContainer()
        ObjInfo(oNewDoc, nSheetsUsed, "getLibraryContainer", oInfo)
    End If
    NoLibraryContainer:
```

```

On Local Error GoTo NoViewData
If InStr(sInterfaces, "com.sun.star.document.X ViewDataSupplier") <> 0 Then
    oInfo = oObject.getViewData()
    ObjInfo(oNewDoc, nSheetsUsed, "getViewData", oInfo)
End If
NoViewData:

On Local Error GoTo NoEvents
If InStr(sInterfaces, "com.sun.star.document.X EventsSupplier") <> 0 Then
    oInfo = oObject.getEvents()
    ObjInfo(oNewDoc, nSheetsUsed, "getEvents", oInfo)
End If
NoEvents:
AllDone:
On Local Error GoTo 0
End Sub

' -----
Sub ObjInfo(oDoc As Object, nSheetsUsed&, sSheetName$, obj As Object)
    Dim i2 As Integer, bProp As Boolean
    Dim nRow&, nCol&, oSheet As Object
    nRow = 0
    ncol = 0
    'on error goto err_ObjInfo

    If Not IsNull(obj) Then
        GoToNextSheet(oDoc, nSheetsUsed, sSheetName)
        oSheet = oDoc.Sheets.getByIndex(nSheetsUsed - 1)
        'oCell.String=CStr(obj.WindowServiceName)
        SetCell(nCol, nRow, oSheet, TypeName(obj), 1, 0)
        SetCell(nCol, nRow, oSheet, VarType(obj), -1, 2)
        i2 = InStr(obj.dbg_methods, "getPropertySetInfo")
        If i2 > 0 Then
            bProp = true
        Else
            bProp = false
        End If
        ListItems (nCol, nRow, obj.dbg_methods, ";", obj, true, oSheet)
        MoveRow(nCol, nRow, 2)
        If Not IsNull (obj.dbg_properties) Then
            'NextSheet()
            ListItems (nCol, nRow, obj.dbg_properties, ";", obj, false, oSheet)
        End If
        If bProp Then
            GetProps(nCol, nRow, obj, oSheet)
            bProp = false
        End If
        MoveRow(nCol, nRow, 1)
        ListItems (nCol, nRow, obj.dbg_supportedinterfaces, Chr$(10), obj, false, oSheet)
        'CleanDbg      'Actuellement aucune idée de ce à quoi cela sert
        'NextSheet()
    End If
    exit_ObjInfo:
    Exit Sub
    err_ObjInfo:
    'print err
    If err=423 Then
        obName = inputbox("L'objet n'a pas de nom." + " Quel nom doit-on donner à la nouvelle feuille ?")
        Resume Next
    Else
        msgbox error$, 16
        Resume exit_ObjInfo
    End If
End Sub

```

```

' -----
' ListItems – recherche de caractères de séparation dans la chaîne transmise
' et affichage ligne par ligne
' -----

Sub ListItems(nCol&, nRow&, itemstring$, sep$, oBj As Object, gt As Boolean, oSheet As Object)
    'dim oCurs As Object
    Dim frag As String, sleft As String, sPrf As String
    Dim act As Integer, ex As Integer, nextpos As Integer
    Dim nextchr As Integer, lstr As Integer, lfrag As Integer
    Dim ch As Integer
    lstr = Len(itemstring)
    act = 1
    'Tout ce qu'il y a à gauche du premier deux points
    SetCell(nCol, nRow, oSheet, Left(itemstring, InStr(1, itemstring, ":")), 0, 0)
    'Si il n'y a pas deuxpoints alors c'est fini
    If InStr(1, itemstring, ":") < 1 Then
        SetCell(nCol, nRow, oSheet, itemstring, 0, 0)
        Exit Sub
    End If
    act = act + InStr(1, itemstring, ":")      ' on commence par le premier séparateur
    'act = act + 1                      ' puis on se positionne sur un caractère plus loin
    MoveRow(nCol, nRow, 1)
    While act < lstr

        nextpos = InStr(act, itemstring, sep)      'déterminer la position du premier ; après le deux
        points
        frag = Mid(itemstring, act, nextpos - act)
        lfrag = Len(frag)
        act = act + lfrag + 1
        frag = LTrim(frag)      'Aligner à gauche
        If frag > "" Then
            Do
                nextchr = Asc(Mid(frag, 1, 1))
                If nextchr = "10" Then
                    Mid(frag, 1, 1, " ")
                    frag = LTrim(frag)      'Aligner à gauche
                Else
                    exit Do
                End If
            Loop
        End If
        MoveRow(nCol, nRow, 1)
        SetCell(nCol, nRow, oSheet, frag, 0, 0)
        If gt Then
            GtVal(nCol, nRow, frag, oBj, oSheet)
        End If
    Wend
    itemstring = ""
End Sub

' -----
' Subroutine pour relire les propriétés, l'objet doit supporter la méthode .PropertySetInfo
Sub GetProps(nCol&, nRow&, obj, oSheet As Object)
    Dim vVariant As Variant
    dim nVar As Integer
    dim mProperties as variant
    dim mProps1 as variant
    dim sItemDescription
    dim nCount As Integer
    dim iP As Integer, iP1 As Integer
    dim n$
    dim p
    dim tmp$
    dim j%
    dim vItem
    dim sString

```

```

MoveRow(nCol, nRow, 2)
mProperties = obj.PropertySetInfo.Properties
nCount = UBound(mProperties)-LBound(mProperties) + 2
SetCell(nCol, nRow, oSheet, "Properties With Values", 0, 1)
SetCell(nCol, nRow, oSheet, "Name", 1, 0)
SetCell(nCol, nRow, oSheet, "Value", -1, 1)
For iP = LBound(mProperties) To UBound(mProperties)
    p = mProperties(iP)
    n$ = p.name
    vVariant = obj.getPropertyValue(n$)
    SetCell(nCol, nRow, oSheet, n$, 1, 0)
    nVar = VarType(vVariant)
    Select Case nVar
        Case 1 'isNull
            SetCell(nCol, nRow, oSheet, "NULL-VALUE", 0, 1)
        Case 9 'object
            If Not IsNull (vVariant.dbg_properties) Then
                ListItems (nCol, nRow, vVariant.dbg_properties, ";", vVariant, false, oSheet)
                MoveRow(nCol, nRow, 2)
            End If
            If not IsNull (vVariant.dbg_supportedinterfaces) then
                ListItems (nCol, nRow, vVariant.dbg_supportedinterfaces, _
                           chr$(10), vVariant, false, oSheet)
                MoveRow(nCol, nRow, 2)
            End If
            If Not IsNull (vVariant.dbg_methods) Then
                ListItems (nCol, nRow, vVariant.dbg_methods, ";", _
                           vVariant, false, oSheet)
                MoveRow(nCol, nRow, 2)
            End If
        Case Else
            If IsArray(vVariant) Then
                tmp$ = ""
                For j% = LBound(vVariant) To UBound(vVariant)
                    vItem = vVariant(j%)
                    If IsNull(vItem) Then
                        sItemDescription = "NULL-Value"
                    ElseIf IsObject(vItem) Then
                        If Not IsNull(vItem.dbg_properties) Then
                            sItemDescription = CStr(vItem.dbg_properties)
                        End If
                    Else
                        sItemDescription = cstr(vItem)
                    End If
                    tmp$ = tmp$ & sItemDescription
                Next j%
                ListItems(nCol, nRow, tmp$,";",vVariant,false, oSheet)
            Else
                SetCell(nCol, nRow, oSheet, cstr(vVariant), 0, 1)
            End If
        End Select
        MoveRow(nCol, nRow, 1)
        MoveCol(nCol, nRow, -1)
    Next iP
End Sub

' -----
' GetValue – relire le contenu
' -----

Sub GtVal (nCol&, nRow&, sGVal As String, oBje As Object, oSheet As Object)
    dim is1 As Integer, iAr As Integer
    dim s1 As String, s2 As String, s3 As String
    dim aR1(10) as variant
    dim o1 As Object
    is1 =InStr(sGVal," ")  'Rechercher le premier espace
    s1 = Mid(sGVal,1,is1)

```

```

s2 = Mid(sGVal,1,is1," ")
sGVal = LTrim(sGVal)

is1 = InStr(sGVal," ")
s2 = Mid(sGVal,1,is1)
s1 = LTrim(s1)
s1= RTrim(s1)
s2 = LTrim(s2)
s2 = RTrim(s2)
Select Case s1
Case "SbxSTRING"
    Select Case s2
        Case "getURL"
            s3 = oBje.getURL()
        Case "getLocation"
            s3 = oBje.getLocation()
        Case "getImplementationName"
            s3 = oBje.getImplementationName()
        Case "getUserFieldName"
            s3 = oBje.getUserFieldName(0)
        Case "getUserFieldValue"
            s3 = oBje.getUserFieldValue(0)
    Case Else
        s3 = s2
    End Select
    's3 = oBje.&s2
    'msgbox(CStr(oBje)&s2)
    MoveCol(nCol, nRow, 4)
    SetCell(nCol, nRow, oSheet, s3, -4, 0)
Case "SbxBOOL"
    Select Case s2
        Case "hasControllersLocked"
            s3= CStr(oBje.hasControllersLocked())
        Case "isModified"
            s3= CStr(oBje.isModified())
        Case "AutoloadEnabled"
            s3= CStr(oBje.AutoloadEnabled())
        Case "hasElements"
            s3= CStr(oBje.hasElements())
        Case "IsEncrypted"
            s3= CStr(oBje.IsEncrypted())
        Case "isReadonly"
            s3= CStr(oBje.isReadonly())
    Case Else
        s3 = " "
    End Select
    MoveCol(nCol, nRow, 4)
    SetCell(nCol, nRow, oSheet, s3, -4, 0)
Case "SbxINTEGER"
    Select Case s2
        Case "getUserFieldCount"
            s3 = CStr(oBje.getUserFieldCount())
        Case "EditingCycles"
            s3 = CStr(oBje.EditingCycles())
    Case Else
        s3 = ""
    End Select
    MoveCol(nCol, nRow, 4)
    SetCell(nCol, nRow, oSheet, s3, -4, 0)
Case "SbxLONG"
    Select Case s2
        Case "getCount"
            s3 = CStr(oBje.getCount())
    Case Else
        s3 = ""
    End Select

```

```

MoveCol(nCol, nRow, 4)
SetCell(nCol, nRow, oSheet, s3, -4, 0)
Case "SbxOBJECT"
Select Case s2
Case "getElementType"
    s3 = CStr(VarType(oBje.getElementType()))
    MoveCol(nCol, nRow, 4)
    SetCell(nCol, nRow, oSheet, s3, -4, 0)
Case "getText"
    o1 = oBje.getText()
    MoveCol(nCol, nRow, 4)
    SetCell(nCol, nRow, oSheet, o1.dbg_properties, 3, 0)
    SetCell(nCol, nRow, oSheet, o1.dbg_methods, -7, 0)
Case Else
End Select
Case "SbxARRAY"
Select Case s2
Case "getImplementationId"
    aR1() = oBje.getImplementationId()
    MoveCol(nCol, nRow, 4)
    For iAr = LBound(oBje.getImplementationID()) To _
        UBound(oBje.getImplementationID())
        s3 = CStr(aR1(iAr))
        SetCell(nCol, nRow, oSheet, s3, 1, 0)
    Next iAr
    MoveCol(nCol, nRow, -(4+1+UBound(oBje.getImplementationID())))
Case "getArgs"
    '?? Pourquoi ceci est il décommenté pour afficher ?
    aR1() = oBje.getArgs()
    MoveCol(nCol, nRow, 4)
    For iAr = LBound(oBje.getArgs()) To UBound(oBje.getArgs())
        o1 = aR1(iAr)
        s3 = o1.dbg_properties
        'GetProps(aR1(iAr))
        'oCell.String = s3
        MoveCol(nCol, nRow, 1)
    Next iAr
    MoveCol(nCol, nRow, -(4+1+UBound(oBje.getArgs())))
Case "getTypes"
    aR1() = oBje.getTypes()
    MoveCol(nCol, nRow, 4)
    ' For iAr = LBound(oBje.getTypes()) To Ubound(oBje.getTypes())
    For iAr = LBound(aR1()) To UBound(aR1())
        o1 = aR1(iAr)
        s3 = VarType(o1)
        SetCell(nCol, nRow, oSheet, s3, 1, 0)
    Next iAr
    MoveCol(nCol, nRow, -(4+1+UBound(oBje.getTypes())))
Case "getElementNames"
    aR1() = oBje.getElementNames()
    MoveCol(nCol, nRow, 4)
    For iAr = LBound(oBje.getElementNames()) To _
        UBound(oBje.getElementNames())
        'o1 = aR1(iAr)
        's3 = VarType(o1)
        SetCell(nCol, nRow, oSheet, aR1(iAr), 1, 0)
    Next iAr
    MoveCol(nCol, nRow, -(4+1+UBound(oBje.getElementNames())))
Case "getSupportedServiceNames"
    aR1() = oBje.getSupportedServiceNames()
    MoveCol(nCol, nRow, 4)
    For iAr = LBound(oBje.getSupportedServiceNames()) To _
        UBound(oBje.getSupportedServiceNames())
        'o1 = aR1(iAr)
        's3 = VarType(o1)
        SetCell(nCol, nRow, oSheet, aR1(iAr), 1, 0)

```

```

Next iAr
MoveCol(nCol, nRow,
-(4+1+UBound(oBje.getSupportedServiceNames())))
Case "getPrinter"
aR1() = oBje.getPrinter()
MoveCol(nCol, nRow, 4)
For iAr = LBound(oBje.getPrinter()) To UBound(oBje.getPrinter())
o1 = aR1(iAr)
s3 = CStr(VarType(aR1(iAr)))
'?? On n'affiche jamais ceci
MoveCol(nCol, nRow, 1)
Next iAr
MoveCol(nCol, nRow, -(4+1+UBound(oBje.getPrinter())))
Case Else
s3 = " "
MoveCol(nCol, nRow, 4)
SetCell(nCol, nRow, oSheet, s3, -4, 0)
End Select
Case Else
s3 = " "
MoveCol(nCol, nRow, 4)
SetCell(nCol, nRow, oSheet, s3, -4, 0)
End Select
End Sub

' -----
Sub SetCell(nCol&, nRow&, oSheet As Object, s$, colInc%, rowInc%)
oSheet.getCellByPosition(nCol, nRow).String = s$
If colInc <> 0 Then MoveCol(nCol, nRow, colInc%)
If rowInc <> 0 Then MoveRow(nCol, nRow, rowInc)
End Sub

' -----
Sub MoveCol(nCol&, nRow&, i%)
nCol = nCol + i
If nCol < 0 Then
nRow = nRow + 1
nCol = 0
End If
End Sub

' -----
Sub MoveRow(nCol&, nRow&, i%)
nRow = nRow + i
If nRow < 0 Then
nRow = 0
End If
End Sub

' -----
'Créé une nouvelle feuille si nécessaire avec son nom
Sub GoToNextSheet(oDoc As Object, nSheetsUsed&, sSheetName$, Optional nWhichSheet%)
Dim oSheets As Object, oSheet As Object
oSheets = oDoc.Sheets
If isNumeric(nWhichSheet) Then
oSheets.insertNewByName("Sheet"&CStr(oSheets.Count()+1), nWhichSheet)
oSheet = oSheet.getByIndex(nWhichSheet)
Else
If nSheetsUsed > oSheets.Count() - 1 Then
nSheetsUsed = oSheets.Count() - 1
oSheets.insertNewByName("Sheet"&CStr(oSheets.Count()+1), _
nSheetsUsed)
End If
oSheet = oSheets.getByIndex(nSheetsUsed)
nSheetsUsed = nSheetsUsed + 1
End If

```

```

oSheet.Name = sSheetName
End Sub

```

Dispatch: Utiliser Universal Network Objects (UNO)

Le guide du développeur et l'URL <http://udk.openoffice.org> sont de bonnes références dans votre quête de la compréhension de UNO. UNO est un modèle de composant offrant l'interopérabilité entre des langages, des modèles d'objet, des architectures matérielles différentes et des processus. Cet exemple utilise une commande UNO pour effectuer la commande "Annuler" (*NdT : le Undo*)

```

Sub UnoUndo
    PerformDispatch(ThisComponent.CurrentController.Frame, ".uno:Undo")
End Sub

Sub PerformDispatch(oObj As Object, uno$)
    Dim oParser As Object
    Dim oUrl As New com.sun.star.util.URL
    Dim oDisp As Object
    Rem Le service UNO est représenté comme une URL
    oUrl.Complete = uno$

    Rem Analyse l'URL comme requis
    oParser = createUnoService("com.sun.star.util.URLTransformer")
    oParser.parseStrict(oUrl)

    Rem Regarde si la Fenêtre active supporte la commande UNO
    oDisp = oObj.queryDispatch(oUrl, "", 0)
    If (Not IsNull(oDisp)) Then
        oDisp.dispatch(oUrl, noargs())
    Else
        MsgBox uno$ & " was not found"
    End If
End Sub

```

A partir de la version 1.1, ceci peut s'écrire

```

Sub Undo
    Dim oDisp as object
    oDisp = createUnoService("com.sun.star.frame.DispatchHelper")
    oDisp.executeDispatch(ThisComponent.CurrentController.Frame, ".uno:Undo", "", 0, Array())
End Sub

```

La partie difficile est de connaître l'interface UNO et les paramètres à utiliser. Par exemple, les deux exemples suivants qui devraient marcher dans la version 1.1 :

```

Dim args1(2) as new com.sun.star.beans.PropertyValue
Dim args2(0) as new com.sun.star.beans.PropertyValue
args1(0).Name = "URL"
args1(0).Value = "my_file_name_.pdf"
args1(1).Name = "FilterName"
args1(1).Value = "writer_pdf_Export"
oDisp.executeDispatch(document, ".uno:ExportDirectToPDF", "", 0, args1())

```

```

' position to B3
args2(0).Name = "ToPoint"
args2(0).Value = "$B$3"
dispatcher.executeDispatch(document, ".uno:GoToCell", "", 0, args2())
dispatcher.executeDispatch(document, ".uno:Copy", "", 0, Array())
dispatcher.executeDispatch(document, ".uno:Paste", "", 0, Array())

```

Hal Vaughan a demandé : "Est-ce juste moi, ou bien y a t-il une raison pour que le dispatcher ne fonctionne pas sur la plupart des fonctions de la version 1.0.3 ?" Et Mathias Bauer de répondre :

"Le Dispatcher utilise certaines fonctionnalités non présentes dans la branche OOo 1.0.x (Dispatcher Helper) comme par exemple le code pour exécuter un dispatch avec des paramètres."

Une autre question de Hal Vaughan et réponse de Mathias Bauer.

Les macros utilisant les noms des Dispatch et pas les nombres ne changeront pas entre les différentes versions de OOo.

Encore une autre question de Hal Vaughan et réponse de Mathias Bauer.

Y a t-il une raison quelconque d'utiliser les appels d'API classique plutôt que le dispatcher avec le nom de la fonction ?

Les appels Dispatch ne marchent pas sans une interface utilisateur. Si OOo tourne en mode serveur réel (ce qui pourrait arriver avec la version 2.0) où les documents sont chargés et travaillés en script sans interface graphique, seules les macros utilisant les appels directs à l'API fonctionneront. Les appels directs à l'API sont plus puissants et donnent une meilleure vue des objets manipulés. A mon humble avis, le dispatcher ne devrait être utilisé que pour ces deux seules raisons :

1. l'enregistreur de macros
2. comme une échappatoire quand une API n'est pas disponible ou si elle est défectueuse.

Afficher du texte dans la barre de statut

```
'Auteur : Sasa Kelecevic
'email : scat@teol.net
'Modifié par : Andrew Pitonyak
'Voici deux méthodes qui peuvent être utilisées
'pour obtenir la barre de statut
Function ProgressBar
    ProgressBar = ThisComponent.CurrentController.StatusIndicator
    'ou bien
    'ProgressBar = StarDesktop.getCurrentComponent.StatusIndicator
    REM Le code suivant a été ajouté, mais n'a pas été testé !
    ProgressBar.reset
    ProgressBar.start("a label", MaxValue)
    aValue = 1
    ProgressBar.setValue(aValue)
End Function

REM affichage du texte dans la barre de statut
Sub StatusText(sInformation)
    Dim sSpaces As String
    Dim iLen,iRest As Integer
    'sSpaces=SPACE(270)
    iLen=Len(sInformation)
    iRest=270-iLen
    ProgressBar.start(sInformation+SPACE(iRest),0)
End Sub
```

D'après Christian Erpelding [erpelding@ce-data.de], la macro ci-dessus ne permet de changer la barre de statut qu'UNE SEULE FOIS, après cela, les changements sont oubliés. Utilisez setText plutôt que start, comme montré ci-dessous.

```
Sub StatusText(sInformation)
    Dim iLen,iRest As Integer
    iLen=Len(sInformation)
    iRest=350-iLen
    StatusBar.setText(sInformation+SPACE(iRest))
End Sub
```

Afficher tous les modèles dans le document courant

Ce n'est pas aussi passionnant qu'il y paraît. Les modèles suivants existent pour un document texte : CharacterStyles, FrameStyles, NumberingStyles, PageStyles, et ParagraphStyles.

```
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub DisplayAllStyles
    Dim mFamilyNames As Variant
    Dim mStyleNames As Variant
    Dim sMsg As String
    Dim oFamilies As Object
    Dim oStyle As Object
    Dim oStyles As Object

    oFamilies = ThisComponent.StyleFamilies
    mFamilyNames = oFamilies.getElementNames()
    For n = LBound(mFamilyNames) To UBound(mFamilyNames)
```

```

sMsg = ""
oStyles = oFamilies.getByName(mFamilyNames(n))
mStyleNames = oStyles.getElementNames()
For i = LBound(mStyleNames) To UBound (mStyleNames)
    sMsg=sMsg + i + " : " + mStyleNames(i) + Chr(13)
    If ((i + 1) Mod 20 = 0) Then
        MsgBox sMsg,0,mFamilyNames(n)
        sMsg = ""
    End If
Next i
MsgBox sMsg,0,mFamilyNames(n)
Next n
End Sub

```

Itération au travers des documents ouverts

```

Sub Main
    Dim oDesktop As Object, oDocs As Object
    Dim oDoc As Object, oComponents As Object
    'Le hasMoreElements() échouera avec l'oDesktop,
    'Je ne sais pas pourquoi !
    'oDesktop = createUnoService("com.sun.star.frame.Desktop")
    oComponents = StarDesktop.getComponents()
    oDocs = oComponents.createEnumeration()
    Do While oDocs.hasMoreElements()
        oDoc = oDocs.nextElement()

        Loop
End Sub

```

NdT : Juste avant le loop, rajouter un “Print “test”” permet de mieux visualiser l’effet ;-). Il faut avoir plusieurs documents ouverts.

Liste des Fontes et d'autres propriétés d'affichage

En fabriquant une fonte, il est courant de générer des versions différentes pour les différents attributs de style comme le gras ou l’italique. Quand vous listez les fontes supportées par votre système, vous trouverez toutes ces variantes. Windows contient par exemple “Courier New”, “Courier New Italic”, “Courier New Bold”, et “Courier New Bold Italic”.

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XToolkit.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XDevice.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/awt/FontDescriptor.html>

```

'Auteur : Paul Sobolik
'email :psobolik@lycos.com
Sub ListFonts
    Dim oToolkit as Object

```

```

oToolkit = CreateUnoService("com.sun.star.awt.Toolkit")

Dim oDevice As Variant
oDevice = oToolkit.createScreenCompatibleDevice(0, 0)

Dim oFontDescriptors As Variant
oFontDescriptors = oDevice.FontDescriptors

Dim oFontDescriptor As Object

Dim sFontList As String
Dim iIndex As Integer, iStart As Integer, iTotal As Integer, iAdjust As Integer
iTotal = UBound(oFontDescriptors) - LBound(oFontDescriptors) + 1
iStart = 1
iAdjust = iStart - LBound(oFontDescriptors)
For iIndex = LBound(oFontDescriptors) To UBound(oFontDescriptors)
    oFontDescriptor = oFontDescriptors(iIndex)
    sFontList = sFontList & iIndex + iAdjust & ":" & oFontDescriptor.Name & " " &
        oFontDescriptor.StyleName & Chr(10)
    If ((iIndex + iAdjust) Mod 20 = 0) Then
        MsgBox sFontList, 0, "Fonts " & iStart & " to " & iIndex + iAdjust & " of " & iTotal
        iStart = iIndex + iAdjust + 1
        sFontList = ""
    End If
Next iIndex
If sFontList <> "" Then MsgBox sFontList, 0, "Fonts " & iStart & " to " & iIndex & " of " & iTotal
End Sub

```

Imprimer le document courant

Je me suis amusé avec ceci et j'ai pu imprimer. J'ai cessé de chercher comment imprimer du A4 sur mon imprimante au format lettre ! Je voulais paramétrier ceci par défaut mais j'ai décidé qu'il n'y avait pas lieu d'y consacrer trop de temps.

```

'*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub PrintCurrentDocument
    Dim mPrintopts1(), x as Variant
    'Dimensionné à 0, si vous paramétrez autre chose, soyez certains de positionner ceci à une

    'valeur plus élevée
    Dim mPrintopts2(0) As New com.sun.star.beans.PropertyValue
    Dim oDocument As Object, oPrinter As Object

    oDocument = ThisComponent

    '*****
    'Voulez vous choisir une imprimante particulière ?
    'Dim mPrinter(0) As New com.sun.star.beans.PropertyValue
    'mPrinter(0).Name="Name"
    'mPrinter(0).value="Other printer"
    'oDocument.Printer = mPrinter()

    '*****
    'Pour imprimer simplement les documents, faire ceci :
    'oDocument.Print(mPrintopts1())

    '*****
    'pour imprimer les pages 1-3, 7, et 9
    'mPrintopts2(0).Name="Pages"
    'mPrintopts2(0).Value="1-3; 7; 9"

```

```

'oDocument.Printer.PaperFormat=com.sun.star.view.PaperFormat.LETTER
'DisplayMethods(oDocument, "propr")
'DisplayMethods(oDocument, "")
oPrinter = oDocument.getPrinter()
MsgBox "printer is from " + LBound(oPrinter) + " to " + UBound(oPrinter)
sMsg = ""
For n = LBound(oPrinter) To UBound(oPrinter)
    sMsg = sMsg + oPrinter(n).Name + Chr(13)
Next n
MsgBox sMsg,0,"Print Settings"

'DisplayMethods(oPrinter, "propr")
'DisplayMethods(oPrinter, "")

'mPrintopts2(0).Name="PaperFormat"
'mPrintopts2(0).Value=com.sun.star.view.PaperFormat.LETTER
'oDocument.Print(mPrintopts2())
End Sub

```

```

dim aPrintOps(0) as new com.sun.star.beans.PropertyValue
oDoc = ThisComponent
oViewCursor = oDoc.CurrentController.getViewCursor()
aPrintOps(0).Name = "Pages"
aPrintOps(0).Value = trim(str(oViewCursor.getPage()))
oDoc.print(aPrintOps())

```

Un autre paramètre à prendre en compte est le paramètre Wait avec une valeur de VRAI. Ceci permet l'impression synchrone et l'appel n'est pas renvoyé à la routine avant que l'opération d'impression ne soit terminée. Ceci permet d'éviter d'inclure un « listener » sur la fin de l'impression, si tant est que vous en ayez besoin.

Je ne sais pas comment déterminer la zone d'impression manuellement, mais je sais le faire par macro ! La solution réside dans l'utilisation d'une CellRangeAddress, qui comporte les propriétés suivantes.

Sheet	index de la feuille
StartColumn	index de la colonne du bord gauche
StartRow	index de la ligne du bord supérieur
EndColumn	Index de la colonne du bord droite
EndRow	index de la ligne du bord inférieur

```

Dim oPrintArea(0) as New com.sun.star.table.CellRangeAddress
With oPrintArea(0)
    .Sheet = 0
    .StartColumn = 0
    .StartRow = 0
    .EndColumn = 14
    .EndRow = 91
End With
ThisComponent.Sheets.getByName("Sheet1").setPrintAreas(oPrintArea())

```

Information de Configuration

NdT : Cette macro permet de changer le nombre d'entrées dans la liste de fichiers récemment ouverts de OpenOffice.

Pour citer le site Web :

Vous pouvez utiliser cette macro mais, tant que le problème ne sera pas résolu, vous devrez "tuer" votre instance d'OpenOffice.org. En effet, lors d'un arrêt normal, la modification effectuée par la macro n'est pas conservée. Il existe d'autres moyens, impliquant la manipulation directe des fichiers de configuration, mais ce n'est pas l'objet de notre discours dans ce document.

```
'Auteur : Unknown
'email :http://ui.openoffice.org/howto/index.html
Option Explicit
Sub SetPickListNine
    ChangePickListSize( 9 )
End Sub

Sub ChangePickListSize( nSize As Integer )
    ' accède à l'objet de configuration
    Dim aConfigProvider As Object
    aConfigProvider = createUnoService( "com.sun.star.configuration.ConfigurationProvider" )

    ' Crée un objet pour le noeud d'historique
    Dim aHistorySettings As Object
    Dim aParams(0) As new com.sun.star.beans.PropertyValue
    aParams(0).Name = "nodepath"
    aParams(0).Value = "/org.openoffice.Office.Common/History"
    aHistorySettings = aConfigProvider.createInstanceWithArguments
    ( "com.sun.star.configuration.ConfigurationUpdateAccess", aParams() )

    ' Définit la taille de la liste
    aHistorySettings.replaceByName( "PickListSize", nSize )

    ' Valide les changements effectués
    aHistorySettings.commitChanges
End Sub
```

Malheureusement la fonction GetSolarVersion a tendance à ne pas changer, même lorsque la version de OOo change. La version 1.0.3.1 renvoie "641" et la 1.1RC3 renvoie 645, mais ceci ne suffit pas toujours pour donner assez de précisions. La macro suivante renvoie la version actuelle de OOo.

```
Function OOoVersion() As String
    'récupère la version de OOo en cours d'exécution
    'Auteur : Laurent Godard
    'e-mail : listes.godard@laposte.net
    '

    Dim aSettings, aConfigProvider
    Dim aParams2(0) As new com.sun.star.beans.PropertyValue
    Dim sProvider$, sAccess$
    sProvider = "com.sun.star.configuration.ConfigurationProvider"
```

```

sAccess = "com.sun.star.configuration.ConfigurationAccess"
aConfigProvider = createUnoService(sProvider)
aParams2(0).Name = "nodepath"
aParams2(0).Value = "/org.openoffice.Setup/Product"
aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())

OOVersion=aSettings.getbyname("ooSetupVersion")
End Function

```

Cette macro retourne la langue dans laquelle OOo a été compilé. Cela permet de savoir donc dans quelle langue est l'interface utilisateur.

```

Function OOOLang() as string
'retire la version de OOo en cours d'exécution
'Auteur : Laurent Godard
'e-mail : listes.godard@laposte.net
'

Dim aSettings, aConfigProvider
Dim aParams2(0) As new com.sun.star.beans.PropertyValue
Dim sProvider$, sAccess$
sProvider = "com.sun.star.configuration.ConfigurationProvider"
sAccess = "com.sun.star.configuration.ConfigurationAccess"
aConfigProvider = createUnoService(sProvider)
aParams2(0).Name = "nodepath"
aParams2(0).Value = "/org.openoffice.Setup/L10N"
aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())

Dim OOLangue as string
OOLangue= aSettings.getbyname("ooLocale")  'en-US
OOLang=lcase(Left(trim(OOLangue),2))      'en
End Function

```

Ouvrir et fermer des documents (et l'application)

Tous les documents OpenOffice et ses fenêtres (services) supportent l'interface XCloseable. Pour fermer ces objets, vous devrez appeler close(bForce As Boolean). Si bForce est faux, alors l'objet pourra refuser de se fermer. Si bForce est vrai, alors l'objet ne sera pas capable de refuser.

L'objet Bureau ne supporte pas l'interface XCloseable pour des raisons historiques. Cette méthode cause un événement queryTermination émis à tous les objets à l'écoute. Si aucun TerminationVetoException n'est positionné, un événement notifyTermination est émis et « vrai » est retourné. Sinon, un événement abortTermination est émis et « faux » est retourné. Pour citer Mathias Bauer, “la méthode terminate() a été utilisée pendant longtemps, bien avant que nous ne découvrions que ce n'est pas la bonne manière de manipuler les documents ou les fenêtres se fermant. Si cette méthode n'avait pas été là, nous aurions employé XCloseable pour le bureau également.”[Bauer001]

```

If HasUnoInterfaces(oDoc, "com.sun.star.util.XCloseable") Then
    oDoc.close(true)
Else
    oDoc.dispose
End If

```

J'ai eu de Sasa Kelecevic [scat@teol.net] cette méthode, que je n'ai pas testée

```

'----- save_and_close -----
'Utilisez l'une de ces deux méthodes
'oDocClose=StarDesktop.CurrentFrame.Close
'oDocClose=StarDesktop.ActiveFrame.Close
'----- close_no_save -----
'Utilisez l'une de ces deux méthodes
'oDocClose=ThisComponent.Dispose
'oDocClose=StarDesktop.ActiveFrame.Dispose

```

Pour fermer un document modifié sans sauvegarder, appelez la méthode setModified(False) avant de fermer le document. Dans OOo1.1, vous avez accès à une autre option : appeler la méthode Close(TRUE) du document ce qui fermera le document sans l'enregistrer, même si celui-ci a été modifié.

Pour charger un document depuis une URL, utilisez la méthode LoadComponentFromURL() depuis le bureau. Ceci charge un composant dans un cadre nouveau ou existant.

```

loadComponentFromURL(
    string aURL,
    string aTargetFrameName,
    long nSearchFlags,
    sequence< com::sun::star::beans::PropertyValue > aArgs)

```

com::sun::star::lang::XComponent

aURL : URL du document à charger. Pour créer un nouveau document, utilisez "private:factory/scalc", "private:factory/swriter", etc.

aTargetFrameName : Nom du cadre qui contiendra le nouveau document. Si un cadre portant ce nom existe, il est utilisé, autrement il est créé. "_blank" crée un nouveau cadre, "_self" utilise le cadre courant, "_parent" utilise le cadre parent, et "_top" utilise le plus élevé des cadres du chemin courant dans l'arbre.

nSearchFlags : Utilisation des valeurs de FrameSearchFlag pour spécifier comment chercher le *aTargetFrameName* spécifié. Normalement, utilisez simplement 0.

<http://api.openoffice.org/docs/common/ref/com/sun/star/frame/FrameSearchFlag.html>

0	Auto	SELF+CHILDREN
1	PARENT	Inclut le cadre parent
2	SELF	Inclut le cadre de départ
4	CHILDREN	Inclut les cadres enfants du cadre de départ

8	CREATE	Le cadre sera créé si non trouvé
16	SIBLINGS	Inclut les autres cadres enfants du parent de cadre de départ
32	TASKS	Inclut tous les cadres de toutes les tâches dans la hiérarchie actuelle des cadres
23	ALL	Inclut tous les cadres non engagés dans d'autres tâches. $23 = 1+2+4+16 = \text{PARENT} + \text{SELF} + \text{CHILDREN} + \text{SIBLINGS}$.
55	GLOBAL	Recherche dans toutes la hiérarchie de frames. $55 = 1+2+4+16+32 = \text{PARENT} + \text{SELF} + \text{CHILDREN} + \text{SIBLINGS} + \text{TASKS}$.
63		GLOBAL + CREATE

Aargs : Indique le comportement spécifique de composant ou de filtre.

"ReadOnly" avec une valeur booléenne indique si le document est ouvert en lecture seulement. "FilterName" indique le composant à créer ou le filtre à utiliser, par exemple : "scalc: Text - csv". Voir : <http://api.openoffice.org/docs/common/ref/com/sun/star/document/MediaDescriptor.html>

```
Rem Charge deux documents dans le même cadre
oDesk = createUnoService("com.sun.star.frame.Desktop")
Dim NoArgs()
Rem Le cadre "MyName" sera créé s'il n'existe pas car il inclut "CREATE"
oDoc1 = oDesk.LoadComponentFromUrl(sUrl_1, "MyName", 63, Noargs())
Rem Utilise un cadre "MyName" existant
oDoc2 = oDesk.LoadComponentFromUrl(sUrl_2, "MyName", 55, Noargs())
```

Dans OOo1.1 le cadre implémente loadComponentFromURL, aussi vous pouvez utiliser :

```
oDoc = oDesk.LoadComponentFromUrl(sUrl_1, "_blank", 0, Noargs())
oFrame = oDoc.CurrentController.Frame
oDoc = oFrame.LoadComponentFromUrl(sUrl_2, "", 2, Noargs())
```

Notez l'argument « drapeau » de la recherche et l'argument nom.

Dans OOo1.1 vous pouvez réutiliser un cadre seulement si vous connaissez son nom.

```
Sub insertDocumentAtCursor(sFileUrl As String, oText As Object, oDoc As Object)
    Dim oCur As Object
    Dim oProperties As Object
    oCur=oText.createTextCursorByRange(oDoc.getCurrentController().getViewCursor().getStart())
    oCur.insertDocumentFromURL(sFileURL,oProperties)
End Sub
```

```
'----- Ouvrir un nouveau document -----
'Dim NoArgs()
'oDocNew=StarDesktop.loadComponentFromURL("private:factory/swriter","_blank",0,NoArgs())
----- Ouvrir un document existant-----
'Dim NoArg()
```

```
'oDocOldFile=StarDesktop.loadComponentFromURL(sUrl,"_blank",0,NoArg())
```

Pour créer un nouveau document basé sur un modèle, utiliser le code suivant :

```
basic = createUnoService("com.sun.star.frame.Desktop")
args(0).Name = "AsTemplate"
args(0).Value = true
oDoc = basic.LoadComponentFromUrl("file:///C|/Templates%20Files/Special.stw","_blank",0,args())
```

Si vous désirez éditer le modèle, mettez « AsTemplate » à « False ».

Lorsqu'un document est chargé par une macro, les macros qui y sont contenues sont désactivées. C'est le réglage par défaut pour des questions de sécurité.. A partir de la version 1.1, vous pouvez activer des macros dès le chargement du document. Il faut mettre la propriété "MacroExecutionMode" à la valeur 2 ou 4 et cela devrait fonctionner. Ceci est basé sur un e-mail de la liste dev . Merci à Mikhail Voitenko <Mikhail.Voitenko@Sun.COM>

<http://www.openoffice.org/servlets/ReadMsg?msgId=782516&listName=dev>

Voici sa réponse sous forme condensée :

Il existe une propriété 'MediaDescriptor' qui s'appelle 'MacroExecutionMode', et qui utilise des valeurs provenant des constantes

'com.sun.star.document.MacroExecMode' . Si cette propriété n'est pas spécifiée, le comportement par défaut empêche l'exécution de la macro. Les valeurs constantes supportées sont données au lien suivant :

<http://api.openoffice.org/source/browse/api/offapi/com/sun/star/document/MacroExecMode.idl?rev=1.5&content-type=text/x-cvsweb-markup>

0	NEVER_EXECUTE	Ne jamais exécuter
1	FROM_LIST	Exécuter les macros à partir d'une liste définie, la possibilité d'émettre un avertissement est donnée par la configuration générale.
2	ALWAYS_EXECUTE	Une macro sera toujours exécutée, la possibilité d'émettre un avertissement est donnée par la configuration générale
3	USE_CONFIG	Utiliser la configuration générale pour récupérer la configuration d'exécution de macro. Dans le cas où une confirmation de la part de l'utilisateur est nécessaire, une boîte de dialogue s'affiche.
4	ALWAYS_EXECUTE_NO_WARN	Une macro sera toujours exécutée sans avertissement.
5	USE_CONFIG_REJECT_CONFIRMATION	Utiliser la configuration générale pour récupérer la configuration d'exécution des macros. Cas où l'utilisateur a rejeté la demande de confirmation
6	USE_CONFIG_APPROVE_CONFIRMATION	Utiliser la configuration générale pour récupérer la configuration d'exécution des macros. Cas où l'utilisateur autorise la macro

Il existe quelques points sensibles qui méritent attention. Si vous chargez un document avec le paramètre "AsTemplate" (c-à-d en tant que modèle), celui-ci

n'est pas ouvert, il est créé. Vous devez lier vos évènements à la commande "create document" (créer un document) plutôt que "open document" (ouvrir un document). Afin de couvrir les deux cas, vous pouvez lier la macro aux deux évènements.

```
Dim mFileProperties(1) As New com.sun.star.beans.PropertyValue  
mFileProperties(0).Name="AsTemplate"  
mFileProperties(0).Value=True  
mFileProperties(1).Name="MacroExecutionMode"  
mFileProperties(1).Value=4
```

Ceci devrait fonctionner pour des macros liées à l'évènement "OnNew" (Create Document), si vous chargez un modèle ou un document sww (mais je ne l'ai pas essayé). Si vous utilisez "OnLoad" (Open Document), vous devez mettre "AsTemplate" à *False* (Faux) (ou bien utiliser un document sww , parce que par défaut la valeur est mise à *False* (Faux), alors que les modèles (stw) ont une valeur par défaut de *True* (Vrai)).

Quand un document échoue au chargement, un message est affiché donnant des indications sur l'échec. Quand le document est chargé depuis C++, il est possible qu'aucune exception ne soit générée et vous ne serez pas informé de l'erreur.

Mathias Bauer a expliqué que l'interface XComponentLoader est incapable de générer une exception arbitraire et donc que le concept de "Interaction Handler" est utilisé. Quand un document est chargé par la méthode loadComponentFromURL, un "InteractionHandler" est passé dans le tableau d'argument. L'interface utilisateur donne un "InteractionHandler" qui converti les erreurs en interactions avec l'utilisateur comme afficher un message d'erreur ou demander un mot de passe (Voir le guide du développeur pour quelques exemples). Si aucun "InteractionHandler" n'est donné en argument, un « handler » par défaut est utilisé. Ce « Handler » par défaut intercepte toutes les erreurs et fait suivre les quelques unes qui pourraient être générées par loadComponentFromURL. Il est cependant impossible d'implémenter son propre Handler en OOBasic. Le guide du développeur donne des exemples dans d'autres langages.

Un publipostage crée un nouveau document pour chaque enregistrement de la base de données. La macro suivante récupère tous les documents Writer dans un seul répertoire et en fait un seul fichier contenant tous les documents du publipostage. J'ai modifié la macro d'origine de manière à ce que toutes les variables soient déclarées et ceci fonctionne même si le premier fichier trouvé n'est pas un document Writer.

```
'Auteur : Laurent Godard  
'Modifié par : Andrew Pitonyak  
Sub MergeDocumentsInDirectory()  
' On Error Resume Next  
Dim DestDirectory As String  
Dim FileName As String  
Dim SrcFile As String, DstFile As String  
Dim oDesktop, oDoc, oCursor, oText
```

```

Dim argsInsert()
Dim args()
'Enlever les commentaires suivants afin de faire l'opération sous forme cachée
'dim args(0) as new com.sun.star.beans.PropertyValue
'args(0).name="Hidden"
'args(0).value=true

'Quel répertoire cible ?
DestDirectory=Trim(GetFolderName())

If DestDirectory = "" Then
    MsgBox "Aucun répertoire sélectionné, quittant l'opération",16,"Fusion des Documents"
    Exit Sub
End If

REM obliger l'insertion d'un anti-slash à la fin. Ceci fonctionne parce qu'on utilise la notation URL
If Right(DestDirectory,1) <> "/" Then
    DestDirectory=DestDirectory & "/"
End If

oDeskTop=CreateUnoService("com.sun.star.frame.Desktop")

REM Le code suivant lit le premier fichier !
FileName=Dir(DestDirectory)
DstFile = ConvertToURL(DestDirectory & "ResultatFusion.sxw")
Do While FileName <> ""
    If lcase(right(FileName,3))="sxw" Then
        SrcFile = ConvertToURL(DestDirectory & FileName)
        If IsNull(oDoc) OR IsEmpty(oDoc) Then
            FileCopy( SrcFile, DstFile )
            oDoc=oDeskTop.Loadcomponentfromurl(DstFile, "_blank", 0, Args())
            oText = oDoc.getText
            oCursor = oText.createTextCursor()
        Else
            oCursor.gotoEnd(false)
            oCursor.BreakType = com.sun.star.style.BreakType.PAGE_BEFORE
            oCursor.insertDocumentFromUrl(SrcFile, argsInsert())
        End If
    End If
    FileName=dir()
Loop

If IsNull(oDoc) OR IsEmpty(oDoc) Then
    MsgBox "Aucun document fusionné!",16,"Fusion des Documents"
    Exit Sub
End If

'Enregistrement du document
Dim args2()
oDoc.StoreAsURL(DestDirectory & "ResultatFusion.sxw",args2())
If HasUnoInterfaces(oDoc, "com.sun.star.util.XCloseable") Then
    oDoc.close(true)
Else
    oDoc.dispose
End If

'Rechargez le document !
oDoc=oDeskTop.Loadcomponentfromurl(DstFile,"_blank",0,Args2())
End Sub

```

Créer une table

Je n'ai rien fait avec ces macros de Kienlein ? ?

```

Sub InsertNextItem(what, oCursor, oTable)
    Dim oCelle As Object

```

```

'nom de la plage de cellules sélectionnées par ce curseur
sName = oCursor.getRangeName()
' Le nom de cellule, qui sera quelque chose comme D3
oCelle = oTable.getCellByName(sName)
oCelle.String = what
oCursor.goRight(1, FALSE)
End Sub

Function CreateTable() As Object
    oDocument = StarDesktop.ActiveComponent
    oTextTable = oDocument.createInstance("com.sun.star.text.TextTable")
    CreateTable = oTextTable
End Function

```

Appeler un programme externe

Utilisez la commande shell.

Nom de fichier externe avec espaces

Voir la section sur la notation URL ! En résumé, utilisez un %20 là où devrait se trouver un espace.

```

Sub ExampleShell
    Shell("file:///C|/Andy/My%20Documents/oo/tmp/h.bat",2)
    Shell("C:\Andy\My%20Documents\oo\tmp\h.bat",2)
End Sub

```

Lire et écrire un nombre dans un fichier

Cet exemple lit un texte d'un fichier texte. Ce nombre est converti en nombre et incrémenté. Le nombre est alors réécrit dans le fichier sous forme de chaîne de caractères.

```

*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub Read_Write_Number_In_File
    DIM CountFileName AS String, NumberString AS String
    DIM LongNumber AS Long, iNum AS Integer
    Dim oDocument As Object
    CountFileName = "C:\Andy\My Documents\oo\NUMBER.TXT"
    NumberString = "00000000"
    LongNumber = 0
    'Si erreur locale, on va à NoFile
    If FileExists(CountFileName) Then
        ON ERROR GOTO NoFile
        iNum = FreeFile
        OPEN CountFileName for input as #iNum
        LINE INPUT #iNum ,NumberString
        CLOSE #iNum
        MsgBox("Lu " & NumberString, 64, "Lu")
    NoFile:
        If Err <> 0 Then
            MsgBox("Impossible de lire " & CountFileName, 64, "Erreur")
            NumberString = "00000001"
        End If
        On Local Error Goto 0
    Else
        MsgBox(CountFileName & " n'existe pas", 64, "Attention!")
    End If
End Sub

```

```

        NumberString = "00000001"
End If

ON ERROR GOTO BadNumber
LongNumber = Int(NumberString)
LongNumber = LongNumber + 1
BadNumber:
If Err <> 0 Then
    MsgBox(NumberString & " n'est pas un nombre", 64, "Erreur")
    LongNumber = 1
End If
On Local Error Goto 0
NumberString=Trim(Str(LongNumber))
While LEN(NumberString) < 8
    NumberString="0"&NumberString
Wend
MsgBox("Le nombre est (" & NumberString & ")", 64, "Information")
iNum = FreeFile
OPEN CountFileName for output as #iNum
PRINT #iNum,NumberString
CLOSE #iNum
End Sub

```

Créer un style de format de nombre

Si vous voulez un format de nombre particulier, alors vous pouvez soit déjà l'avoir, soit le créer si vous ne l'avez pas. Pour de plus amples informations sur les formats valides voir le contenu de l'aide avec le mot clé « formats de nombre ; formats ». Ils peuvent être très complexes.

```

*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Function FindCreateNumberFormatStyle (
    sFormat As String, Optional doc, Optional locale)
    Dim oDocument As Object
    Dim aLocale as new com.sun.star.lang.Locale
    Dim oFormats As Object
    oDocument = IIf(IsMissing(doc), ThisComponent, doc)
    oFormats = oDocument.getNumberFormats()
    'Si vous choisissez de chercher des types, vous aurez à utiliser
    'com.sun.star.util.NumberFormat.DATE
    'Je pourrais utiliser les valeurs de locales stockées à
    'http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt
    'http://www.chemie.fu-berlin.de/diverse/doc/ISO\_3166.html
    'J'utilise une locale NULL locale et je lui laisse employer ce qui convient
    'D'abord, vérifier si le format de nombre existe
    If ( Not IsMissing(locale)) Then
        aLocale = locale
    End If
    formatNum = oFormats.queryKey (sFormat, aLocale, TRUE)
    MsgBox "Le format numérique courant est" & formatNum
    ' Si le format n'existe pas, alors on l'ajoute
    If (formatNum = -1) Then
        formatNum = oFormats.addNew(sFormat, aLocale)
        If (formatNum = -1) Then formatNum = 0
        MsgBox "Le nouveau format numérique est " & formatNum
    End If
    FindCreateNumberFormatStyle = formatNum
End Function

```

Retourner un tableau de Fibonacci

Cette fonction renvoie un tableau de nombres de Fibonacci.

```
'*****
' http://disemia.com/software/openoffice/macro_arrays.html
' Renvoie une suite de nombres de Fibonacci
' On présume que count est supérieur ou égal à deux, afin de simplifier le code
Function Fibonacci( Count As Integer )
    Dim result( 1 To Count, 1 ) As Double
    result( 1, 1 ) = 0
    result( 2, 1 ) = 1

    For i = 3 To Count
        result( i, 1 ) = result( i - 2, 1 ) + result( i - 1, 1 )
    Next i

    Fibonacci = result()
End Function
```

Insérer un texte à la position d'un signet

```
oDoc.getBookmarks().getByName("<yourBookmarkName>").getAnchor.setString(
"ce que vous souhaitez insérer")
```

Sauvegarder et exporter un document

Sauvegarder un document est vraiment simple. La macro suivante sauvegardera un document, mais seulement s'il a été modifié, s'il n'est pas en lecture seule et qu'un emplacement de sauvegarde est paramétré.

```
If (oDoc.isModified) Then
    If (oDoc.hasLocation And (Not oDoc.isReadOnly)) Then
        oDoc.store()
    End If
End If
```

Si le document doit être sauvegardé ailleurs, alors vous devez paramétrier quelques propriétés pour indiquer où et comment le document doit être sauvegardé.

```
Dim mFileProperties(0) As New com.sun.star.beans.PropertyValue
Dim sUrl As String
sUrl = "file:///complete/path/To/New/document"
Rem Mettre la valeur à vrai (c'est-à-dire remplacer False par True) pour écraser le document.
mFileProperties(0).Name = "Overwrite"
mFileProperties(0).Value = False
oDoc.storeAsURL(sUrl, mFileProperties())
```

Le code montré jusque là n'exportera pas le document avec un format différent. Pour faire cela, un filtre d'export particulier doit être défini et toutes les propriétés requises doivent être paramétrées. Vous devrez connaître le nom du filtre d'exportation et l'extension du fichier. Il y a une liste de filtres d'import et d'export à :

http://www.openoffice.org/files/documents/25/111/filter_description.html

et il y a pas mal d'infos intéressantes à :

http://oooconv.free.fr/engine/OOOconv.php_

Une méthode séparée est requise pour les filtres graphiques et le reste. Pour

exporter en utilisant un format non graphique, utilisez un formulaire semblable à ce qui suit :

```
Dim args2(1) as new com.sun.star.beans.PropertyValue
args2(0).Name = "InteractionHandler"
args2(0).Value = ""
args2(1).Name = "FilterName"
args2(1).Value = "MS Excel 97"
Rem Change le filtre d'export
oDoc.storeToURL("file:///c|/new_file.xls",args2())
Rem Utilise l'extension de fichier correcte
```

Notez que j'ai utilisé une extension de fichier correcte et que j'ai spécifié le bon filtre d'export. C'est un peu différent pour les documents graphiques.

Premièrement, vous instancez un GraphicExportFilter et vous lui dites d'exporter une page à la fois.

```
oFilter=CreateUnoService("com.sun.star.drawing.GraphicExportFilter")
Dim args3(1) as new com.sun.star.beans.PropertyValue
For i=0 to oDoc.drawpages.getcount()-1
    oPage=oDoc.drawPages(i)
    oFilter.setSourceDocument(opage)
    args3(0).Name = "URL"
    nom=opage.name
    args3(0).Value = "file:///c|/"&opage.name&"."&extension
    args3(1).Name = "MediaType"
    args3(1).Value = "image/jpeg"
    oFilter.filter(args3())
Next
```

Champs utilisateurs

J'ai passé un peu de temps sur les champs utilisateurs et même si je ne comprends pas réellement tout ce qu'il y a savoir, je peux au moins les utiliser ! La plupart des gens choisira d'utiliser des champs Maîtres (Master Fields) qui permettent de définir leurs propres noms et valeurs.

Dans les propriétés du document, il existe 4 champs portant les noms "Info 1", "Info 2", "Info 3" et "Info 4". Je ne les utilise pas mais comme ils existent et que vous pouvez y accéder, je les mentionne.

```
' Accède aux champs utilisateurs des propriétés du document
vInfo = vDoc.getDocumentInfo()
vVal = oData.ElementNames
s = "====User Fields===="
For i = 0 to vInfo.GetUserFieldCount() - 1
    sKey = vInfo.GetUserFieldName(i)
    sVal = vInfo.GetUserFieldValue(i)
    s = s & Chr$(13) & "(" & sKey & "," & sVal & ")"
Next i
'(Info 1.)
'(Info 2.)
'(Info 3.)
'(Info 4.)
MsgBox s, 0, "User Fields"
```

Je ne connais pas la raison d'être de ce type de champs. J'ai deux champs

comme ça dans mon document de test mais ils n'ont aucune valeur associée.

```
s ==>Text Fields=="
Dim vEnum
vEnum = vDoc.getTextFields().createEnumeration()
If Not IsNull(vEnum) Then
    Do While vEnum.hasMoreElements()
        vVal = vEnum.nextElement()
        s = s & Chr(13) & "(" & vVal.TextFieldMaster.Name & ")"
        ' Je ne sais pas quoi faire avec ça ???
    Loop
End If
MsgBox s, 0, "Text Fields"
```

Les champs Maîtres sont sympas car vous pouvez y mettre vos propres valeurs, formules ou nombres. Cette section n'est qu'une brève investigation mais cela devrait suffire pour se lancer. Il y a 5 types de champs maîtres : « Illustration », « Table », « Text », « Drawing » et « User ». Les noms de ces champs commencent tous par “com.sun.star.text.FieldMaster.SetExpression.” suivi du type puis d'un autre point et enfin du nom du champ.

Voici un exemple simple :

```
vDoc = ThisComponent
sName = "Author Name"
If vDoc.getTextFieldMasters().hasByName("com.sun.star.text.FieldMaster.User." & sName) Then
    vField = vDoc.getTextFieldMasters().getByName("com.sun.star.text.FieldMaster.User." &
sName)
    vField.Content = "Andrew Pitonyak"
    'vField.Value = 2.3 Rem Si vous préférez un nombre
Else
    vField = vDoc.createInstance("com.sun.star.text.FieldMaster.User")
    vField.Name = sName
    vField.Content = "Andrew Pitonyak"
    'vField.Value = 2.3 Rem Si vous préférez un nombre
End If
```

Cette macro affiche tous les champs maîtres d'un document :

```
Sub FieldExamples
    Dim vDoc, vInfo, vVal, vNames
    Dim i%, sKey$, sVal$, s$
    vDoc = ThisComponent
    Dim vTextFieldMaster
    Dim sUserType$
    sUserType = "com.sun.star.text.FieldMaster.User"

    vVal = vDoc.getTextFieldMasters()
    vNames = vVal.getElementNames()
    'vous pouvez avoir des noms tels que:
    'com.sun.star.text.FieldMaster.SetExpression.Illustration
    'com.sun.star.text.FieldMaster.SetExpression.Table
    'com.sun.star.text.FieldMaster.SetExpression.Text
    'com.sun.star.text.FieldMaster.SetExpression.Drawing
    'com.sun.star.text.FieldMaster.User
    s ==>Text Field Masters==="
    For i = LBound(vNames) to UBound(vNames)
        sKey = vNames(i)
        s = s & Chr$(13) & "(" & sKey
        vTextFieldMaster = vVal.getByName(sKey)
        If Not IsNull(vTextFieldMaster) Then
            s = s & "," & vTextFieldMaster.Name
```

```

'Je n'ai pas vérifié si c'est le cas!
If (Left$(sKey,Len(sUserType)) = sUserType) Then
    'Les types User ont également un type (double) et vous pouvez interroger
    ' pour savoir s'il s'agit d'expressions
    'http://api.openoffice.org/docs/common/ref/com/sun/star/text/FieldMaster/User.html
    s = s & "," & vTextFieldMaster.Content
End If
End If
s = s & ")"
Next i
MsgBox s, 0, "Text Field Masters"
End Sub

```

Les routines suivantes ont été postées par Rodrigo V Nunes
[\[rodrigo.nunes@net-linx.com\]](mailto:[rodrigo.nunes@net-linx.com]) mais je ne les ai pas testées.

```

'=====
=====
' CountDocVars – Routine comptant le nombre de variables document disponibles pour le document
courant
' In - DocVars: tableau des variables document courantes (nom) présentes dans l'annonce
' DocVarValue: tableau des variables document courantes (valeurs) présentes dans l'annonce
' Out – entier contenant le nombre total de variables doc trouvées
'=====
=====
Function CountDocVars(DocVars , DocVarValue) As Integer
Dim VarCount As Integer
Dim Names as Variant

VarCount = 0
Names = thisComponent.getTextFieldMasters.getElementNames()
for i%=LBound(Names) To UBound(Names)
    if (Left$(Names(i%),34) = "com.sun.star.text.FieldMaster.User") Then
        xMaster = oActiveDocument.getTextFieldMasters.getByName(Names(i%))
        DocVars(VarCount) = xMaster.Name
        DocVarValue(VarCount) = xMaster.Value
        VarCount = VarCount + 1 ' variable document créée par l'utilisateur
    End if
Next i%

CountDocVars = VarCount
End Function
'

'=====
'
' SetDocumentVariable -la routine utilisée pour créer/paramétrer la valeur d'une variable
'de document dans la liste de textField utilisateur dans le document, sans
'insérer physiquement son contenu dans le texte de l'annonce
' In - strVarName: chaîne contenant le nom de la variables à créer/paramétrer
' aValue: chaîne avec la valeur de la variable document
' Out – drapeau booléen contenant le statut de l'opération: TRUE=OK,
' FALSE= la variable ne peut être créée ou modifiée
'=====
'
Function SetDocumentVariable(ByVal strVarName As String, ByVal aValue As String ) As Boolean
Dim bFound As Boolean

On Error GoTo ErrorHandler
oActiveDocument = thisComponent
oTextmaster = oActiveDocument.getTextFieldMasters()
sName = "com.sun.star.text.FieldMaster.User." + strVarName
bFound = oActiveDocument.getTextFieldMasters.hasbyname(sName) ' teste si la variable existe.
if bFound Then

```

```

xMaster = oActiveDocument.getTextFieldMasters.getByName( sName )
REM la valeur MEMBER est utilisée pour les valeurs décimales
REM et la valeur CONTENT pour les chaînes de caractères
'xMaster.value = aValue
xMaster.Content = aValue
Else      ' La variable document n'existe pas encore.
    sService = "com.sun.star.text.FieldMaster.User"
    xMaster = oActiveDocument.createInstance( sService )
    xMaster.Name = strVarName
    xMaster.Content = aValue
End If
SetDocumentVariable = True      'Succes
Exit Function

ErrorHandler:
    SetDocumentVariable = False
End Function
'

=====
' InsertDocumentVariable - routine insérant une variable de document dans la liste des textfields
' utilisateur du document 'et dans le texte d'annonce, à la position courante du curseur
' In - strVarName: chaîne avec le nom r de la variable document à insérer
'           oTxtCursor: objet curseur courant avec la position où placer la variable doc
' Out - rien
=====
Sub InsertDocumentVariable(strVarName As String, oTxtCursor As Object)

    oActiveDocument = thisComponent
    objField = thisComponent.createInstance("com.sun.star.text.TextField.User")
    sName = "com.sun.star.text.FieldMaster.User." + strVarName
    bFound = oActiveDocument.getTextFieldMasters.hasbyname(sName)
    ' contrôle l'existence de la variable
    if bFound Then
        objFieldMaster = oActiveDocument.getTextFieldMasters.getByName(sName)
        objField.attachTextFieldMaster(objFieldMaster)
        ' insère le champ texte
        oText = thisComponent.Text
        'oCursor = oText.createTextCursor()
        'oCursor.gotoEnd(false)
        oText.insertTextContent(oTxtCursor, objField, false)
    End If
End Sub

=====
' DeleteDocumentVariable - routine éliminant une variable document de la liste des testfields
' utilisateur du document'
' In - strVarName: chaîne avec le nom de la variable document à éliminer
' Out - rien
=====
Sub DeleteDocumentVariable(strVarName As String)

    oActiveDocument = thisComponent
    objField = oActiveDocument.createInstance("com.sun.star.text.TextField.User")
    sName = "com.sun.star.text.FieldMaster.User." + strVarName
    bFound = oActiveDocument.getTextFieldMasters.hasbyname(sName)
    ' contrôle l'existence de la variable
    if bFound Then
        objFieldMaster = oActiveDocument.getTextFieldMasters.getByName(sName)
        objFieldMaster.Content = ""

```

```

        objFieldMaster.dispose()
    End If

End Sub
'=====
' SetUserVariable - fonction utilisée pour créer/paramétrer les variables utilisateurs dans le corps du
document.
' Ces variables servent seulement à des usages/contrôles du système interne et ne sont PAS
disponibles ou employées en Java (voir 'SetDocumentVariables' pour la création/le paramétrage
de variables document partagées)
'
' In - strVarName: chaîne avec le nom de la variable document à créer/paramétrer. Si elle n'existe
pas, elle sera créée
' avalue: valeur variant avec le nouveau contenu de la variable définie dans strVarName
' Out - drapeau booléen contenant le statut de l'opération :
' TRUE=OK, FALSE=La variable, ne peut pas être créée ou modifiée
'=====

Function SetUserVariable(ByVal strVarName As String, ByVal avalue As Variant) As Boolean

Dim aVar As Variable
Dim index As Integer           'Index des noms de variables existants
Dim vCount As Integer

On Error GoTo ErrorHandler
'Vérifie que la variable document existe déjà
oDocumentInfo = thisComponent.Document.Info
vCount = oDocumentInfo.getUserFieldCount()
bFound = false
For i% = 0 to (vCount - 1)
    If strVarName = oDocumentInfo.getUserFieldName(i%) Then
        bFound = true
        oDocumentInfo.setUserFieldValue(i%, avalue)
    End If
Next i%
If not bFound Then           'Variable document n'existant plus
    oDocumentInfo.setUserFieldName(i, strVarName)
    oDocumentInfo.setUserValue(i, avalue)
End If
'teste si la valeur est supérieure au nombre de variables utilisateurs !
SetUserVariable = True      'Success
Exit Function

ErrorHandler:
    SetUserVariable = False
End Function

```

Types définis par l'utilisateur

Bien qu'OOBasic permette syntaxiquement de déclarer vos propres types, il est incapable de les utiliser. Une issue a été créée pour demander l'ajout de cette fonctionnalité (Bug ?)

http://www.openoffice.org/project/www/issues/show_bug.cgi?id=14465

```

Type PersonType
    Dim FirstName As String
    Dim LastName As String
End Type

Sub TypeExample

```

```

Dim person As PersonType
'On ne peut rien faire avec cette variable
End Sub

```

Correcteur orthographique, césure et thésaurus

Faire une correction orthographique, appliquer les césures ou utiliser le thésaurus est très simple. Cependant, ces éléments retourneront une valeur Null s'ils ne sont pas configurés. Lors de mes tests initiaux, la césure a toujours retourné Null tant que je ne l'ai pas configurée dans les options.

```

Sub SpellCheckExample
    Dim s() As Variant
    Dim vReturn As Variant, i As Integer
    Dim emptyArgs(0) as new com.sun.star.beans.PropertyValue
    Dim aLocale As New com.sun.star.lang.Locale
    aLocale.Language = "en"
    aLocale.Country = "US"

    s = Array("hello", "anesthesiologist",
    "PNEUMONOULTRAMICROSCOPICSILICOVOLCANOCONIOSIS", "Pitonyak", "misspell")

    '*****Exemple de correcteur orthographique !
    'http://api.openoffice.org/docs/common/ref/com/sun/star/linguistic2/XSpellChecker.html
    Dim vSpeller As Variant
    vSpeller = createUnoService("com.sun.star.linguistic2.SpellChecker")
    'Utilisez vReturn = vSpeller.spell(s, aLocale, emptyArgs()) si vous voulez des options!
    For i = LBound(s()) To UBound(s())
        vReturn = vSpeller.isValid(s(i), aLocale, emptyArgs())
        MsgBox "Spell check on " & s(i) & " returns " & vReturn
    Next

    '*****Exemple de Césure !
    'http://api.openoffice.org/docs/common/ref/com/sun/star/linguistic2/XHyphenator.html
    Dim vHyphen As Variant
    vHyphen = createUnoService("com.sun.star.linguistic2.Hyphenator")
    For i = LBound(s()) To UBound(s())
        'vReturn = vHyphen.hyphenate(s(i), aLocale, 0, emptyArgs())
        vReturn = vHyphen.createPossibleHyphens(s(i), aLocale, emptyArgs())
        If IsNull(vReturn) Then
            'La césure est probablement désactivée
            MsgBox "Hyphenating " & s(i) & " returns null"
        Else
            MsgBox "Hyphenating " & s(i) & " returns " & vReturn.getPossibleHyphens()
        End If
    Next

    '***** Exemple Thesaurus !
    'http://api.openoffice.org/docs/common/ref/com/sun/star/linguistic2/XThesaurus.html
    Dim vThesaurus As Variant, j As Integer, k As Integer
    vThesaurus = createUnoService("com.sun.star.linguistic2.Thesaurus")
    s = Array("hello", "stamp", "cool")
    For i = LBound(s()) To UBound(s())
        vReturn = vThesaurus.queryMeanings(s(i), aLocale, emptyArgs())
        If UBound(vReturn) < 0 Then
            Print "Le Thesaurus n'a rien trouvé pour " & s(i)
        Else
            Dim sTemp As String
            sTemp = "Hyphenated " & s(i)
            For j = LBound(vReturn) To UBound(vReturn)
                sTemp = sTemp & Chr(13) & "Meaning = " & vReturn(j).getMeaning() & Chr(13)
            Dim vSyns As Variant
            vSyns = vReturn(j).querySynonyms()
            For k = LBound(vSyns) To UBound(vSyns)
                sTemp = sTemp & vSyns(k) & " "
            End If
        End If
    Next

```

```

        Next
        sTemp = sTemp & Chr(13)
    Next
    MsgBox sTemp
End If
Next
End Sub

```

Changer le curseur de la souris

La réponse rapide : ce n'est pas implémenté.

Une question sur le changement du curseur de la souris a induit une discussion intéressante que j'ai pris le temps de suivre mais pas de tester. J'ai édité les messages ci-dessous :

anindya@agere.com a demandé : Je voudrais que le curseur soit un sablier quand ma macro tourne. Qu'est ce qui ne va pas avec ce code ?

```

oDocument = oDeskTop.loadComponentFromURL(fileName,"_blank",0,mArg())
oCurrentController = oDocument.getCurrentController()
oFrame = oCurrentController.getFrame()
oWindow = oFrame.getContainerWindow()
oPointer = createUnoService("com.sun.star.awt.Pointer")
oPointer.SetType(com.sun.star.awt.SystemPointer.WAIT)
oWindow.setPointer(oPointer)

```

Mathias Bauer, que nous aimons tous, a répondu : On ne peut pas changer le curseur de la souris d'une fenêtre d'un document par l'API-UNO. VCL gère le pointeur de souris sur la fenêtre, pas la fenêtre mère. Toute fenêtre VCL peut avoir son curseur. Si vous voulez changer le pointeur de la souris de la fenêtre du document, vous devez avoir accès à son XwindowPeer et ce n'est pas disponible dans l'API. Un autre problème pourrait être qu'OOo change le pointeur en interne, écrasant vos changements.

Berend Cornelius donne la réponse finale : Votre routine marche très bien pour toute fenêtre fille de votre document. Le code suivant se réfère à un contrôle dans le document :

```

Sub Main
    GlobalScope.BasicLibraries.LoadLibrary("Tools")
    oController = Thiscomponent.getCurrentController()
    oControl = oController.getControl(ThisComponent.Drawpage().getbyIndex(0).getControl())
    SwitchMousePointer(oControl.getPeer(), False)
End Sub

```

Cette routine change le pointeur de la souris quand il est au-dessus du contrôle et revient à son état normal quand il le quitte. Vous voulez une fonction de « Wait » qui place le pointeur dans un état d'attente mais ce n'est pas possible avec l'API.

A mon avis, vous pouvez le changer mais pas pour tout :

```
oDoc.getCurrentController().getFrame().getContainerWindow().setPointer(...)
```

Changer le fond de page

```

Sub Main
    ' Les familles de style
    oStyleFamilies= ThisComponent.getStyleFamilies()
    ' Les styles de page

```

```

oPageStyles= oStyleFamilies.getByName("PageStyles")
' VOTRE style de page
oMyPageStyle= oPageStyles.getByName("Standard")
' Votre fond
with oMyPageStyle
    .BackGraphicUrl=
        convertToUrl( <CheminVersVotreGraphique> )
    .BackGraphicLocation=
        com.sun.star.style.GraphicLocation.AREA
end with
End Sub

```

Manipuler le presse-papier

?? Voir le guide du développeur page 331 !

Je ne connais pas la meilleure méthode pour accéder au presse-papier mais ce qui est présenté ici a marché, un jour, pour quelqu'un, quelque part...

Pour copier des données dans le presse-papier, il faut tout d'abord les sélectionner. L'interface optionnelle donne la possibilité de sélectionner des objets et d'accéder aux objets en cours de sélection. Cette section contient des exemples pour obtenir le texte sélectionné pour à la fois des documents Calc et Writer.

Le premier exemple qui m'a été envoyé sélectionne des cellules dans une feuille de calcul et les colle dans une autre :

```

'Auteur : Ryan Nelson
'email : ryan@aurelius-mfg.com
'Cette macro copie une plage de cellules et la colle dans une feuille existante ou une nouvelle
Sub CopyPasteRange()
    'Inclure cette bibliotheque pour utiliser la commande DispatchSlot
    GlobalScope.BasicLibraries.LoadLibrary("Tools")

    Dim oSourceDoc As Object, oSourceSheet As Object, oSourceRange As Object
    Dim oTargetDoc As Object, oTargetSheet As Object, oTargetCell As Object
    Dim sUrl As String

    'Définit le document source/la feuille/la plage
    oSourceDoc=ThisComponent
    oSourceSheet= oSourceDoc.Sheets(0)
    oSourceRange = oSheet.getCellRangeByPosition(0,5,100,10000)

    'Sélectionne la plage source
    ThisComponent.CurrentController.Select(oSourceRange)

    'Copie la selection courante dans le presse-papier
    DispatchSlot(5711)

    oDesk = createUnoService("com.sun.star.frame.Desktop")
    'Définit l'URL du fichier cible ou ouvre une nouvelle feuille
    sUrl = "File:///C:/temp/testing2.sxc"

    'ouvre le fichier.
    Dim NoArg()
    oTargetDoc=oDesk.loadComponentFromURL(sUrl,"_blank",0,NoArg())
    oTargetSheet = oTargetDoc.Sheets(0)
    'Vous pouvez nettoyer la plage cible avant de coller si elle contient des données ou du
    formatage
    'Met le focus sur la cellule 0,0 avant de coller

```

```

'On peut mettre le focus sur n'importe quelle cellule. Si on ne définit pas la position,
' le collage s'effectuera à partir de la dernière cellule active lors de la dernière fermeture du
document
oTargetCell = oTargetSheet.getCellByPosition(0,0)
oTargetDoc.CurrentController.Select(oTargetCell)

'Colle le presse-papier à la position courante
DispatchSlot(5712)
End Sub

```

Cet exemple utilise la méthode DispatchSlot pour copier-coller du texte en utilisant le presse-papier. Une liste des « slots » supportés dans la version 1.0.3.1 peut être trouvée ici :

http://www.openoffice.org/files/documents/25/60/commands_103.html

Une autre possibilité est d'utiliser un dispatcher avec les arguments “.uno:Copy” et “.uno:Paste”.

Il est inutile d'utiliser le presse-papier quand on veut copier une plage de cellules dans la même feuille. Il est possible de copier, insérer, supprimer et déplacer une plage de cellule d'un endroit à l'autre dans la même feuille. Voir pour plus de détails :

<http://api.openoffice.org/docs/common/ref/com/sun/star/sheet/XCellRangeMovement.html>

Le code suivant a été posté sur la liste dev@api.openoffice.org

```

'Auteur : Oliver Brinzing
'email : OliverBrinzing@t-online.de
Sub CopySpreadsheetRange
    oSheet1 = ThisComponent.Sheets.getByIndex(0)    ' feuille no 1, originale
    oSheet2 = ThisComponent.Sheets.getByIndex(1)    ' feuille no 2

    oRangeOrg = oSheet1.getCellRangeByName("A1:C10").RangeAddress    ' copie la plage
    oRangeCpy = oSheet2.getCellRangeByName("A1:C10").RangeAddress    ' insère la plage

    oCellCpy = oSheet2.getCellByPosition(oRangeCpy.StartColumn,_
        oRangeCpy.StartRow).CellAddress ' Position d'insertion

    oSheet1.CopyRange(oCellCpy, oRangeOrg)                      ' copie ...
End Sub

```

Paramétrier la localisation

Dans OOo, les caractères sont localisés (*NdT* : associés à une langue). Dans le style « Macro Code » de ce document, j'ai mis la localisation sur inconnue et ainsi le texte n'est pas analysé par le correcteur orthographique. Pour dire à OOo qu'un mot est en français, on paramètre sa localisation à Français. On m'a demandé comment faire pour un document entier. Cela paraissait évident à première vue. Un curseur supporte les propriétés des caractères qui permettent de définir la localisation. J'ai donc créé un curseur, et paramétré la localisation. J'ai obtenu une erreur d'exécution. J'ai découvert que la propriété de localisation devait être de type Void (impliquant que l'on ne peut pas la paramétriser). Bien que mon essai suivant fonctionne pour mon document, vous devriez vérifier plus précisément, avec les tableaux entre autres.

```

Sub SetDocumentLocale
    Dim vCursor
    Dim aLocale As New com.sun.star.lang.Locale
    aLocale.Language = "fr"
    aLocale.Country = "FR"

    Rem Utilisation sous entendue d'un document writer
    Rem Récupere le composant Text du document
    Rem Crée le curseur sur le texte
    vCursor = ThisComponent.Text.createTextCursor()
    Rem Navigue au début du document
    Rem Navigue alors à la fin du document en sélectionnant tout le texte
    vCursor.GoToStart(False)
    Do While vCursor.gotoNextParagraph(True)
        vCursor.CharLocale = aLocale
        vCursor.goRight(0, False)
    Loop
    MsgBox "successfully francophonized"
End Sub

```

Il serait prudent d'utiliser “On Local Error Resume Next” mais je ne l'ai pas fait car cela aurait caché toute erreur durant mes tests préliminaires.

Vous devriez pouvoir définir la localisation pour le texte sélectionné ou un texte trouvé lors d'une recherche.

AutoTexte

Je n'ai pas testé ce code, mais l'on m'a assuré qu'il fonctionnait. Vous ne pourrez pas utiliser ce code directement car il requiert une boîte de dialogue non fournie mais la technique utilisée devrait être utile.

```

'Auteur : Marc Messeant
'email : marc.liste@free.fr
'Pour copier un autoTexte d'une groupe à l'autre
'ListBox1 : Le groupe initial
'ListBox2 : Le groupe destination
'ListBox3 : Elément du groupe initial à copier
'ListBox4 : Elément du groupe destination (pour information)

Dim oDialog as object
Dim oAutoText as object

' Cette procédure ouvre la boîte de dialogue et initialise la liste des groupes

Sub OuvrirAutoText
    Dim aTableau() as variant
    Dim i as integer
    Dim oListGroupDepart as object, oListGroupArrivee as object

    oDialog = LoadDialog("CG95","DialogAutoText")
    oListGroupDepart = oDialog.getControl("ListBox1")
    oListGroupArrivee = oDialog.getControl("ListBox2")
    oAutoText = createUnoService("com.sun.star.text.AutoTextContainer")
    aTableau = oAutoText.getElementNames()
    oListGroupDepart.removeItems(0,oListGroupDepart.getItemCount())
    oListGroupArrivee.removeItems(0,oListGroupArrivee.getItemCount())
    For i = LBound(aTableau()) To UBound(aTableau())
        oListGroupDepart.addItem(aTableau(i),i)
        oListGroupArrivee.addItem(aTableau(i),i)
    Next
    oDialog.Execute()
End Sub

```

```

'Ces 3 procédures sont appelées quand l'utilisateur sélectionne un groupe
' pour initialiser la liste des AutoTextes de chaque groupe
Sub ChargerList1()
    ChargerListeGroupe("ListBox1","ListBox3")
End Sub
Sub ChargerList2()
    ChargerListeGroupe("ListBox2","ListBox4")
End Sub

Sub ChargerListeGroupe(ListGroupe as string,ListElement as string)
    Dim oGroupe as object
    Dim oListGroupe as object
    Dim oListElement as object
    Dim aTableau() as variant
    Dim i as integer

    oListGroupe = oDialog.getControl(ListGroupe)
    oListElement = oDialog.getControl(ListElement)
    oGroupe = oAutoText.getByIndex(oListGroupe.getSelectedItemPos())
    aTableau = oGroupe.getTitles()
    oListElement.removeItems(0,oListElement.getItemCount())
    For i = LBound(aTableau()) To UBound(aTableau())
        oListElement.addItem(aTableau(i),i)
    Next
End Sub

'Transfère un élément d'un groupe vers un autre
Sub TransfererAutoText()
    Dim oGroupDepart as object,oGroupArrivee as object
    Dim oListGroupDepart as object, oListGroupArrivee as object
    Dim oListElement as object
    Dim oElement as object
    Dim aTableau() as string
    Dim i as integer

    oListGroupDepart = oDialog.getControl("ListBox1")
    oListGroupArrivee = oDialog.getControl("ListBox2")
    oListElement = oDialog.getControl("ListBox3")
    i = oListGroupArrivee.getSelectedItemPos()
    If oListGroupDepart.getSelectedItemPos() = -1 Then
        MsgBox ("Vous devez sélectionner un groupe de départ")
        Exit Sub
    End If
    If oListGroupArrivee.getSelectedItemPos() = -1 Then
        MsgBox ("Vous devez sélectionner un groupe d'arrivée")
        Exit Sub
    End If
    If oListElement.getSelectedItemPos() = -1 Then
        MsgBox ("Vous devez sélectionner un élément à copier")
        Exit Sub
    End If
    oGroupDepart = oAutoText.getByIndex(oListGroupDepart.getSelectedItemPos())
    oGroupArrivee = oAutoText.getByIndex(oListGroupArrivee.getSelectedItemPos())
    aTableau = oGroupDepart.getElementNames()
    oElement = oGroupDepart.getByIndex(oListElement.getSelectedItemPos())
    If oGroupArrivee.HasByName(aTableau(oListElement.getSelectedItemPos())) Then
        MsgBox ("Cet élément existe déjà")
        Exit Sub
    End If
    oGroupArrivee.insertNewByName(aTableau(oListElement.getSelectedItemPos()),_
        oListElement.getSelectedItem(),oElement.Text)
    ChargerListeGroupe("ListBox2","ListBox4")
End Sub

```

« Pieds » décimaux en fraction

On m'a demandé de convertir des macros Microsoft Office en macros OOo. J'ai décidé de les améliorer. Le premier jeu de macros prenait un nombre décimal de pieds et le convertissait en pieds et pouces. J'ai décidé d'écrire une routine plus générale, ignorant le code existant. Cela m'a également permis d'éviter quelques bugs dans le code existant. La manière la plus rapide que je connaisse pour réduire une fraction est de trouver le PGCD (*NdT : GCD en anglais*), le Plus Grand Commun Diviseur. La macro de fraction appelle la fonction GCD pour simplifier la fraction.

```
'e-mail : olivier.bietzer@free.fr
'Ceci utilise les algorithmes d'Euclide et c'est très rapide !
Function GCD(ByVal x As Long, ByVal y As Long) As Long
    Dim pgcd As Long, test As Long

    ' Nous devons avoir x >=y et des valeurs positives
    x=abs(x)
    y=abs(y)
    If (x < y) Then
        test = x : x = y : y = test
    End If
    If y = 0 Then Exit Function

    ' Euclide dit ....
    pgcd = y      ' par définition PGCD est le plus petit
    test = x MOD y ' reste de la division
    Do While (test) ' Tant que le reste n'est pas 0
        pgcd = test  ' pgcd est le reste
        x = y      ' x,y et permutation courante de pgcd
        y = pgcd
        test = x MOD y ' nouveau test
    Loop
    GCD = pgcd ' pgcd est le dernier reste différent de 0 ! Magique ...
End Function
```

La macro suivante détermine la fraction. Si x est négatif, alors le numérateur et la valeur retournée de x sont négatifs. Veuillez noter que le paramètre x est modifié.

```
'n: en sortie, contient le numérateur
'd: en sortie, contient le dénominateur
'x: Nombre à mettre en fraction en entrée, en sortie la partie entière
'max_d: Dénominateur maximum
Sub ToFraction(n&, d&, x#, ByVal max_d As Long)
    Dim neg_multiply&, y#
    n = 0 : d = 1 : neg_multiply = 1 : y = Fix(x)
    If (x < 0) Then
        x = -x : neg_multiply = -1
    End If
    n = CLng((x - Fix(x)) * max_d)
    d = GCD(n, max_d)
    n = neg_multiply * n / d
    d = max_d / d
    x = y
End Sub
```

Pour tester cette routine, j'ai créé le code suivant :Sub FractionTest

```
Dim x#, inc#, first#, last#, y#, z#, epsilon#
Dim d&, n&, max_d&
first = -10 : last = 10 : inc = 0.001
max_d = 128
```

```

epsilon = 1.0 / CDbl(max_d)
For x = first To last Step inc
    y = x
    ToFraction(n, d, y, max_d)
    z = y + CDbl(n) / CDbl(d)
    If abs(x-z) > epsilon Then Print "Conversion incorrecte " & x & " to " & z
Next
End Sub

```

Bien que j'aie beaucoup ignoré le code initial, j'ai voulu conserver les formats d'entrée-sortie initiaux même si ils ne sont pas adaptés.

```

Rem [-]feet'-inches n/d"
Rem Rien n'est retourné si c'est 0.
Function DecimalFeetToString64(ByVal x#) As String
    'J'utilise 64 car c'est ce qui était à l'origine
    DecimalFeetToString64 = DecimalFeetToString(x, 64)
End Function

Function DecimalFeetToString(ByVal x#, ByVal max_denominator&) As String
    Dim numerator&, denominator&
    Dim feet#, declInch#, s As String

    s = ""
    If (x < 0) Then
        s = "-"
        x = -x
    End If

    feet = Fix(x)      'Nombre entier de pieds
    x = (x - feet) * 12 'pouces
    ToFraction(numerator, denominator, x, max_denominator)
    Rem gère quelques traitements d'arrondis
    If (numerator = denominator AND numerator <> 0) Then
        numerator = 0
        x = x + 1
    End If

    If feet = 0 AND x = 0 AND numerator = 0 Then
        s = s & "0"
    Else
        If feet <> 0 Then
            s = s & feet & "'"
            If x <> 0 OR numerator <> 0 Then s = s & "-"
        End If
        If x <> 0 Then
            s = s & x
            If numerator <> 0 Then s = s & " "
        End If
        If numerator <> 0 Then s = s & "/" & denominator
        If x <> 0 OR numerator <> 0 Then s = s & "''"
    End If
    DecimalFeetToString = s
End Function

Function StringToDecimalFeet(s$) As Double
    Rem Le maximum de sortie devrait contenir
    Rem <pieds><'><-><pouces><espace><numérateur></><dénominateur><">
    Rem La première sortie doit être un nombre !
    Dim tokens(8) As String '0 to 8
    Dim i%, j%, num_tokens%, c%
    Dim feet#, inches#, n#, d#, leadingNeg#
    feet = 0 : inches = 0 : n = 0 : d = 1 : i = 1 : leadingNeg = 1.0
    s = Trim(s) 'Enlève les espaces superflus
    If (Len(s) > 0) Then
        If Left(s,1) = "-" Then

```

```

        leadingNeg = -1.0
        s = Mid(s, 2)
    End If
End If

num_tokens = 0 : i = 1
Do While i <= Len(s)
    Select Case Mid(s, i, 1)
        Case "-", "0" To "9"
            j = i
            If Left(s, i, 1) = "-" Then j = j + 1
            c = Asc(Mid(s, j, 1))
            Do While (48 <= c And c <= 57)
                j = j + 1
                If j > Len(s) Then Exit Do
                c = Asc(Mid(s, j, 1))
            Loop
            tokens(num_tokens) = Mid(s, i, j-i)
            num_tokens = num_tokens + 1
            i = j
        Case """
            feet = CDbl(tokens(num_tokens-1))
            tokens(num_tokens) = """
            num_tokens = num_tokens + 1
            i = i + 1
            If (i <= Len(s)) Then
                If Mid(s,i,1) = ":" Then i = i + 1
            End If
        Case "", "/", " "
            tokens(num_tokens) = Mid(s, i, 1)
            i = i + 1
            Do While i < Len(s)
                If Mid(s, i, 1) <> tokens(num_tokens) Then Exit Do
                i = i + 1
            Loop
            If tokens(num_tokens) = "/" Then
                n = CDbl(tokens(num_tokens-1))
                num_tokens = num_tokens + 1
            ElseIf tokens(num_tokens) = " " Then
                Inches = CDbl(tokens(num_tokens-1))
            ElseIf tokens(num_tokens) = """ Then
                If num_tokens = 1 Then
                    Inches = CDbl(tokens(num_tokens-1))
                Elseif num_tokens > 1 Then
                    If tokens(num_tokens-2) = "/" Then
                        d = CDbl(tokens(num_tokens-1))
                    Else
                        Inches = CDbl(tokens(num_tokens-1))
                    End If
                End If
            End If
        End If
    Case Else
        'Hmm, ceci est une erreur
        i = i + 1
        Print "In the else"
    End Select
Loop

If d = 0 Then d = 1
StringToDecimalFeet = leadingNeg * (feet + (inches + n/d)/12)
End Function

```

Envoyer un Email

OOo donne le moyen d'envoyer un document en pièce jointe par e-mail. OOo utilise un client existant plutôt que d'implémenter son propre protocole de mail. Sous Linux il devrait pouvoir utiliser les clients les plus courants comme Mozilla/Netscape, Evolution ou K-Mail. Sous Windows, OOo utilise MAPI donc tout client compatible devrait fonctionner. On va utiliser "com.sun.star.system.SimpleSystemMail".

Cet exemple a été fourni par Laurent Godard. Comme lui, je n'ai pas réussi à mettre du texte dans le corps du mail généré, simplement envoyer une pièce jointe.

```
Sub SendSimpleMail()
    Dim vMailSystem, vMail, vMessage

    vMailSystem=createUnoService("com.sun.star.system.SimpleSystemMail")
    vMail=vMailSystem.querySimpleMailClient()
    'Pour en savoir plus sur les possibilités offertes
    'http://api.openoffice.org/docs/common/ref/com/sun/star/system/XSimpleMailMessage.html
    vMessage=vMail.createSimpleMailMessage()
    vMessage.setRecipient("Andrew.Pitonyak@qwest.com")
    vMessage.setSubject("This is my test subject")

    'Les pieces jointes sont définies dans une séquence donc un tableau sous OOBASIC
    'On aurait pu utiliser ConvertToURL() pour construire l'URL à partir du chemin système !
    Dim vAttach(0)
    vAttach(0) = "file:///c:/macro.txt"
    vMessage.setAttachment(vAttach())

    'DEFAULTS Lance le client mail par défaut du système
    'NO_USER_INTERFACE Pas d'interface utilisateur !
    'NO_LOGON_DIALOG Pas de boîte d'authentification – Génère une exception si une est requise
    vMail.sendSimpleMailMessage(vMessage,
        com.sun.star.system.SimpleMailClientFlags.NO_USER_INTERFACE)
End Sub
```

Ni le service SimpleSystemMail ni SimpleCommandMail ne sont capables de générer un contenu texte au mail. D'après Mathias Bauer, l'objectif de ces services est de pouvoir envoyer un document en tant que pièce jointe. Il est cependant possible d'utiliser une URL « mailto » avec un message dans le corps du mail mais qui ne contient pas de pièce jointe.

```
MyURL = createUnoStruct( "com.sun.star.util.URL" )
MyURL.Complete = "mailto:demo@someplace.com?subject=Test&Body=Text"
trans = createUnoService( "com.sun.star.util.URLTransformer" )
trans.parseStrict( MyURL )
disp = StarDesktop.queryDispatch( MyURL, "", 0 )
disp.dispatch( MyURL, noargs() )
```

Dans Ooo1.1, c'est encore plus facile :

```
dim noargs()
email_dispatch_url = "mailto:demo@someplace.com?subject=Test&Body=Text"
dispatcher = createUnoService( "com.sun.star.frame.DispatchHelper" )
dispatcher.executeDispatch( StarDesktop,email_dispatch_url, "", 0, noargs() )
```

Bibliothèques

Si vous désirez distribuer des bibliothèques et inclure des macros dans votre document pour les installer, un traitement spécial est requis. Sunil Menon a

automatisé ce processus avec l'aide de Oliver Brinzing :

```
'Auteur : Sunil Menon
'email :sunil.menon@itb-india.com
Set service_name = "com.sun.star.script.ApplicationScriptLibraryContainer"
Set oLibLoad = objServiceManager.createInstance(service_name)
If Not oLibLoad Is Nothing Then
    On Error Resume Next
    If oLibLoad.isLibraryLoaded("mymacros") Then
        oLibLoad.removeLibrary ("mymacros")
    End If
    spath = "file:///D|/StarOfficeManual/mymacros"
    slib = "mymacros"
    Call oLibLoad.CreateLibraryLink(slib, spath, False)
    oLibLoad.loadLibrary ("mymacros")
    oLibLoad = Nothing
End If
```

Ce qui suit est un résumé de ce que Sun dit sur le sujet.

J'ai une application VB qui utilise l'interface de StarOffice. Plutôt que de coder des fonctionnalités comme « Chercher-Replacer », « Imprimer » et « Extraction complexe de texte » en VB, j'utilise des macros StarBasic. Pour distribuer ces macros aux utilisateurs, je crée des bibliothèques de macros (script.xlb, appmacro.xba). La bibliothèque est placée à l'endroit où mon application VB est installée. La bibliothèque doit être enregistrée et chargée avant d'être utilisable. Je peux alors appeler cette macro depuis Visual Basic en utilisant la commande Shell.

```
Shell "D:\StarOffice6.0\program\soffice.exe macro:///Standard.Module1.MAIN("""Hello Andrew""")",
vbNormalFocus
```

Si je modifie les macros, elles doivent être de nouveau enregistrées avant que les changements ne soient pris en compte. Elles sont enregistrées à partir du répertoire de l'application et copiées ensuite par OOo.

Cet appel crée un lien à une bibliothèque externe accessible en utilisant le gestionnaire de bibliothèques. Le format de l'URL dépend de l'implémentation. Le paramètre booléen est indicateur de lecture seule.

Modifier la taille d'une Bitmap

Si vous chargez une image dans un document OOo, sa taille risque de ne pas convenir. Vance Lankhaar a attiré mon attention le premier sur ce problème. Sa première solution donnait une image de très petite taille :

```
'Auteur : Vance Lankhaar
'email :vlankhaar@linux.ca
Dim oDesktop As Object, oDocument As Object
Dim mNoArgs()
Dim sGraphicURL As String
Dim sGraphicService As String, sUrl As String
```

```

Dim oDrawPages As Object, oDrawPage As Object
Dim oGraphic As Object
sGraphicURL = "http://api.openoffice.org/branding/images/logonew.gif"
sGraphicService = "com.sun.star.drawing.GraphicObjectShape"
sUrl = "private:factory/simpress"
oDesktop = createUnoService("com.sun.star.frame.Desktop")
oDocument = oDesktop.loadComponentFromURL(sUrl, "_default", 0, mNoArgs())
oDrawPages = oDocument.DrawPages
oDrawPage = oDrawPages.insertNewByIndex(1)
oGraphic = oDocument.createInstance(sGraphicService)
oGraphic.GraphicURL = sGraphicURL
oDrawPage.add(oGraphic)

```

La première solution donnée par Laurent Godard change la taille à la taille maximum possible :

```

'Taille maximum, perte du ratio de proportionnalité.
dim TheSize as new com.sun.star.awt.Size
dim TheBitmapSize as new com.sun.star.awt.Size
dim TheBitmap as object
dim xmult as double, ymult as double

TheBitmap=oGraphic.GraphicObjectFillBitmap
TheBitmapSize=TheBitmap.GetSize

xmult=TwipsPerPixelX/567*10*100 '567 twips = 1 cm *1*100 for 1/100th mm
ymult=TwipsPerPixelY/567*10*100

TheSize.width=TheBitmapSize.width*xmult
TheSize.height=TheBitmapSize.height*ymult

oGraphic.setSize(TheSize)

```

Vance Lankhaar en a déduit la solution finale maximisant la taille mais conservant le rapport de proportionnalité :

```

oBitmap = oGraphic.GraphicObjectFillBitmap
aBitmapSize = oBitmap.GetSize
iWidth = aBitmapSize.Width
iHeight = aBitmapSize.Height

iPageWidth = oDrawPage.Width
iPageHeight = oDrawPage.Height
dRatio = CDbl(iHeight) / CDbl(iWidth)
dPageRatio = CDbl(iPageHeight) / CDbl(iPageWidth)

REM C'est la dimension la plus grande de redimensionnement
If (dRatio < dPageRatio) Then
    aSize.Width = iPageWidth
    aSize.Height = CInt(CDbl(iPageWidth) * dRatio)
Else
    aSize.Width = CInt(CDbl(iPageHeight) / dRatio)
    aSize.Height = iPageHeight
End If

aPosition.X = (iPageWidth - aSize.Width)/2
aPosition.Y = (iPageHeight - aSize.Height)/2

oGraphic.SetSize(aSize)
oGraphic.SetPosition(aPosition)

```

David Woody [dwoody1@airmail.net] avait besoin d'insérer une image à une

Association APLDI 305 www.apldi.fr.st

position et à une taille précises. Avec un peu d'aide et beaucoup de travail, il a réussi à élaborer la solution suivante :

Cette réponse m'a pris du temps, parce que je devais résoudre un autre problème lié à la détermination correcte des coordonnées X et Y. Le code suivant insère une image, la dimensionne, et la positionne à l'endroit voulu. J'ai dû ajouter la ligne suivante dans le code d'Andrew dans la section portant sur la spécification de la taille de l'image.

```
Dim aPosition as new com.sun.star.awt.Point
```

L'autre problème que j'ai eu, c'était de déterminer le rapport nécessaire entre aPosition.X et aPosition.Y afin de positionner correctement l'image. Sur mon ordinateur, la valeur de 2540 pour la coordonnée X ou Y était égale à un pouce à l'écran. Les valeurs ci-dessous positionneront l'image à un pouce du haut de la page et à un pouce depuis le bord gauche.

```
Sub InsertAndPositionGraphic
REM Récupérer la feuille
Dim vSheet
vSheet = ThisComponent.Sheets(0)

REM Insérer l'image
Dim oDesktop As Object, oDocument As Object
Dim mNoArgs()
Dim sGraphicURL As String
Dim sGraphicService As String, sUrl As String
Dim oDrawPages As Object, oDrawPage As Object
Dim oGraphic As Object
sGraphicURL = "file:///usr/local/openoffice1.1RC/share/gallery/bullets/blkpearl.gif"
sGraphicService = "com.sun.star.drawing.GraphicObjectShape"
oDrawPage = vSheet.getDrawPage()
oGraphic = ThisComponent.createInstance(sGraphicService)
oGraphic.GraphicURL = sGraphicURL
oDrawPage.add(oGraphic)

REM Dimensionner l'image
Dim TheSize as new com.sun.star.awt.Size
TheSize.width=400
TheSize.height=400
oGraphic.setSize(TheSize)

REM Positionner l'image
Dim aPosition as new com.sun.star.awt.Point
aPosition.X = 2540
aPosition.Y = 2540
oGraphic.setPosition(aPosition)
End Sub
```

Cette macro est de Sven Jacobi [Sven.Jacobi@sun.com]

Il est possible de spécifier la résolution, mais elle n'est jamais parvenue au guide du développeur et j'en suis désolé. Cette possibilité existe à partir de la version 1.1 de OOo. Chaque filtre de conversion d'image supporte une séquence de propriétés appelée "FilterData" dans laquelle on peut préciser la taille en pixels avec les propriétés "PixelWidth" et "PixelHeight", la taille logique pouvant être définie(en 1/100mm) avec les propriétés "LogicalWidth" et "LogicalHeight". La macro suivante illustre cette possibilité.

```
Sub ExportCurrentPageOrSelection
'création des propriétés dépendantes des filtres
Dim aFilterData (4) as new com.sun.star.beans.PropertyValue
aFilterData(0).Name = "PixelWidth" '
```

```

aFilterData(0).Value = 1000
aFilterData(1).Name = "PixelHeight"
aFilterData(1).Value = 1000
aFilterData(2).Name ="LogicalWidth"
aFilterData(2).Value = 1000
aFilterData(3).Name ="LogicalHeight"
aFilterData(3).Value = 1000
aFilterData(4).Name ="Quality"
aFilterData(4).Value = 60
Dim sFileUrl As String
sFileUrl = "file:///d:/test2.jpg"

xDoc = thiscomponent
xView = xDoc.currentController
xSelection = xView.selection
If isEmpty( xSelection ) then
    xObj = xView.currentPage
else
    xObj = xSelection
End If
Export( xObj, sFileUrl, aFilterData() )
End Sub

Sub Export( xObject, sFileUrl As String, aFilterData )
    xExporter = createUnoService( "com.sun.star.drawing.GraphicExportFilter" )
    xExporter.SetSourceDocument( xObject )

    Dim aArgs (2) as new com.sun.star.beans.PropertyValue
    Dim aURL as new com.sun.star.util.URL

    aURL.complete = sFileUrl
    aArgs(0).Name = "MediaType"
    aArgs(0).Value = "image/jpeg"
    aArgs(1).Name = "URL"
    aArgs(1).Value = aURL
    aArgs(2).Name = "FilterData"
    aArgs(2).Value = aFilterData
    xExporter.filter( aArgs() )
End Sub

```

David Woody [dwoody1@airmail.net] a fourni la macro suivante :
Le code ci-dessous fonctionne pour moi. Notez toutefois que les variables 'TheSize' sont relatives à la variable 'aPosition', de telle sorte que si vous voulez x1 = 500 et x2 = 2000, alors TheSize.width = x2 - x1. La même chose s'applique pour la coordonnée Y.

```

Sub DrawLineInCalcDocument
    Dim xPage as object, xDoc as object, xShape as object
    Dim aPosition as new com.sun.star.awt.Point
    Dim TheSize as new com.sun.star.awt.Size

    xDoc = thiscomponent
    xPage = xDoc.DrawPages(0)
    xShape = xDoc.createInstance( "com.sun.star.drawing.LineShape" )
    xShape.LineColor = rgb( 255, 0, 0 )
    xShape.LineWidth = 100
    aPosition.X = 2500
    aPosition.Y = 2500
    xShape.setPosition(aPosition)
    TheSize.width = 2500
    TheSize.height=5000
    xShape.setSize(TheSize)
    xPage.add( xShape )

```

End Sub

Extraction d'un Fichier Zippé

Laurent Godard [listes.godard@laposte.net] frappe encore et à nouveau avec cette solution. J'ai modifié un peu son mail.

Bonjour à tous

Merci beaucoup de ton aide ! En combinant les conseils divers que vous m'avez tous donnés, j'ai enfin réussi à le faire fonctionner ! L'objectif est de gérer le contenu du flux de données rentrantes de la même manière que l'API de OOo, indépendamment de savoir ce qui est dedans !

Afin de résoudre mon problème, j'ai créé un flux sortant OutputStream et y ai écrit mon flux de données rentrantes, et c'est tout. Cela semble fonctionner (testé avec un fichier texte, mais devrait fonctionner pour les autres...).

Comme promis, voici ma première tentative de la macro pour dézipper un fichier, dont le nom est connu, se trouvant dans un fichier ZIP. Il reste encore beaucoup de boulot à faire, mais cela peut permettre de faire avancer les choses.... Andrew, tu peux inclure ce code dans ta documentation sur les macros.

Merci à tous encore pour votre aide précieuse

Laurent Godard.

```
*****
Sub UnzipAFile(ZipURL as string, SrcFileName as string, DestFile as string)
'Auteur : Laurent Godard
'E-mail : listes.godard@laposte.net

Dim bExists as boolean

ozip=createUnoService("com.sun.star.packages.Package")

Dim args1(0)
args1(0)=ConvertToURL(ZipURL)
ozip.initialize(args1())

'le fichier source existe-t-il ?
bExists=ozip.HasByHierarchicalName(SrcFileName)
if bnot bExists then exit sub

'récupérer un flux de données
ThePackageStream=ozip.GetByHierarchicalName(SrcFileName)

'récupérer le flux de données rentrantes depuis SrcFileName
MyInputStream=ThePackageStream.GetInputStream()

'définir la sortie
oFile = createUnoService("com.sun.star.ucb.SimpleFileAccess")
oFile.WriteFile(ConvertToURL(DestFile),MyInputStream)

'et voilà !!!
```

End Sub

Dan Juliano <daniel.juliano@rainhail.com> <djuliano@dmacc.edu> capitalise à partir de l'exemple de Laurent Godard. L'exemple suivant extrait tous les

documents d'un fichier zippé.

```
' Instructions d'appel des subroutines suivantes
call unzipFileFromArchive("c:\test.zip", "test.txt", "c:\test.txt")
call unzipArchive("c:\test.zip", "c:\")

Sub unzipFileFromArchive( _
    strZipArchivePath As String, _
    strSourceFileName As String, _
    strDestinationFilePath As String)

    Dim bInExists      As Boolean
    Dim args(0)        As Variant
    Dim objZipService  As Variant
    Dim objPackageStream As Variant
    Dim objOutputStream As Variant
    Dim objInputStream  As Variant
    Dim i               As Integer

    '=====
    ' Dézippe un fichier unique à partir d'une archive zippée. Vous devez connaître le nom exact du
    ' fichier
    ' présent dans l'archive pour que cette sub puisse le trouver
    ' strZipArchivePath = chemin entier (répertoire et nom du fichier) identifiant l'archive .zip.
    ' strSourceFileName = nom du fichier à extraire de l'archive .zip.
    ' strDestinationFilePath = chemin entier (répertoire et nom du fichier) indiquant l'endroit
    ' où le fichier doit être extrait.
    '=====

    ' Créer une instance pour le dézippeur
    objZipService = createUnoService("com.sun.star.packages.Package")
    args(0) = ConvertToURL(strZipArchivePath)
    objZipService.initialize(args())

    ' Le fichier source existe-t-il ?
    If Not objZipService.HasByHierarchicalName(strSourceFileName) Then Exit Sub

    ' Prendre l'information (stream) sur le fichier dans les données de l'archive
    objPackageStream = objZipService.GetByHierarchicalName(strSourceFileName)
    objInputStream = objPackageStream.GetInputStream()

    ' Définir l'extraction
    objOutputStream = createUnoService("com.sun.star.ucb.SimpleFileAccess")
    objOutputStream.writeFile(ConvertToURL(strDestinationFilePath), objInputStream)
End Sub

Sub unzipArchive( _
    strZipArchivePath As String, _
    strDestinationFolder As String)

    Dim args(0)        As Variant
    Dim objZipService  As Variant
    Dim objPackageStream As Variant
    Dim objOutputStream As Variant
    Dim objInputStream  As Variant
    Dim arrayNames()   As Variant
    Dim strNames       As String
    Dim i               As Integer

    '=====
    ' Dézippe une archive .zip entière dans un répertoire destination
    '
```

```

' strZipArchivePath = chemin entier (répertoire et nom du fichier) identifiant l'archive .zip.
' strDestinationFolderPath = dossier destination (seulement le répertoire) dans lequel les fichiers de
l'archive seront dézippés.
'=====
' Créer une instance pour le dézippeur
objZipService = createUnoService("com.sun.star.packages.Package")
args(0) = ConvertToURL(strZipArchivePath)
objZipService.initialize(args())

' Récupérer les informations sur le contenu de l'archive
objPackageStream = objZipService.GetByHierarchicalName("")

' Récupérer la liste de tous les fichiers zippés dans l'archive
arrayNames = objPackageStream.getElementNames()

' Pour chaque fichier listé dans arrayNames, extraire/dézipper le fichier
' depuis l'archive vers le répertoire de destination
For i = LBound(arrayNames) To UBound(arrayNames)
    strNames = strNames & arrayNames(i) & Chr(13)

    ' Lire et extraire un seul fichier à la fois pour le système de fichiers
    objInputStream = objZipService.GetByHierarchicalName(arrayNames(i)).GetInputStream()
    objOutputStream = createUnoService("com.sun.star.ucb.SimpleFileAccess")
    objOutputStream.WriteFile(ConvertToURL(strDestinationFolder & arrayNames(i)),_
        objInputStream)
Next
MsgBox strNames
End Sub

```

Laurent Godard fournit aussi cet exemple. Cette macro zippe le contenu d'un répertoire en incluant les sous-répertoires.

```

'-----
sub ExempleAppel
    call ZipUnRepertoire("C:\MesFichiers\Ooo\Rep","C:\resultat.zip")
end sub
'-----
sub ZipUnRepertoire(source as string, cible as string)
'Auteurr : Laurent Godard
'e-mail : listes.godard@laposte.net
    dim retour() as string

    'création de l'instance du fichier Zip
    LeFichierZip=createUnoService("com.sun.star.packages.Package")
    dim args(0)
    args(0)=ConvertToURL(cible)
    LeFichierZip.initialize(args())

    'création de la structure des répertoires dans le zip
    call Recursedirectory(source, retour)

    dim argsDir(0)
    argsDir(0)=true

    'Le premier élément le répertoire contenant --> on ne le traite pas dans la boucle
    'Pourra être une option à terme
    Repbase=retour(1)

    For i=2 To UBound(retour)
        chaine=mid(retour(i),len(repbase)+2)
        decoupe=split(mid(retour(i),len(repbase)+1),getPathSeparator)

```

```

repZip=decoupe(ubound(decoupe))
azipper=LeFichierZip.createInstanceWithArguments(argsDir())

If len(chaine)<>len(repZip) then
    RepPere=left(chaine,len(chaine)-len(repZip)-1)
    RepPere=ReplaceChaine(reppere, getpathseparator, "/", false)
Else
    RepPere=""
Endif

RepPereZip=LeFichierZip.getByHierarchicalName(RepPere)
RepPereZip.insertbyname(repzip, azipper)
Next i

'insertion des fichiers dans les bons répertoires
dim args2(0)
args2(0)=false
oUcb = createUnoService("com.sun.star.ucb.SimpleFileAccess")

for i=1 to ubound(retour)
    chaine=mid(retour(i),len(repbase)+2)
    repzip=remplacechaine(chaine, getpathseparator, "/", false)
    fichier=dir(retour(i)+getPathSeparator,0)
    While fichier<> ""
        azipper=LeFichierZip.createInstanceWithArguments(args2())
        oFile = oUcb.OpenFileRead(ConvertToURL(retour(i)+"."+fichier))
        azipper.SetInputStream(ofile)
        RepPere=LeFichierZip.getByHierarchicalName(repZip)
        RepPere.insertbyname(fichier, azipper)
        fichier=dir()
    Wend
next i

'Valide les changements
LeFichierZip.commitChanges()
msgbox "C'est fini"
End Sub
'-----
sub RecurseDirectory(chemin, reponse as variant)
'Auteur: Laurent Godard
'e-mail : listes.godard@laposte.net
'reponse est un tableau contenant la liste de tous les sous répertoires de chemin
redim preserve reponse(1 to 1)
compte=1
reponse(1)=chemin
repbase=1
rep=dir(convertTouri(chemin+"/"),16)

while rep<> ""
    if rep<>"." and rep<>.." then
        compte=compte+1
        redim preserve reponse(1 to compte)
        reponse(compte)=convertfromurl(reponse(RepBase)+"."+rep)
    endif
    rep=dir()

    while rep="" and repbase<compte
        repbase=repbase+1
        rep=dir(convertToURL(reponse(repbase)+"/"),16)
    wend
wend
End Sub
'-----
Function ReplaceChaine(ByVal chaine As String, src As String, dest As String,_
casse As Boolean)
'Auteurs: Laurent Godard & Bernard Marcelly

```

```

' fournit une chaîne dont toutes les occurrences de src ont été remplacées par dest
'casse = true pour distinguer majuscules/minuscules, = false sinon
Dim lsrc As Integer, i As Integer, kas As Integer
dim limite as string

limite=""
kas = iif(casse, 0, 1)
lsrc = len(src)
i = instr(1, chaine, src, kas)
while i<>0
    while i<0
        limite=limite+left(chaine,32000)
        chaine=mid(chaine,32001)
        i=instr(1, chaine, src, kas)
    wend
    ' ici i est toujours positif non nul
    if i>1 then
        limite = limite + Left(chaine, i-1) +dest
    else ' ici i vaut toujours 1
        limite = limite +dest
    endif
    ' raccourcir en deux temps car risque : i+src > 32767
    chaine = Mid(chaine, i)
    chaine = Mid(chaine, 1+lsrc)
    i = instr(1, chaine, src, kas)
wend
RemplaceChaine = limite + chaine
End Function

```

S'agit-il d'un document tableur ?

Un document tableur est composé d'un ensemble de feuilles (sheets en anglais). Avant de pouvoir utiliser les méthodes spécifiques du tableur, vous devez disposer d'un document tableur. Vous pouvez vérifier cela comme suit :

```

Function IsSpreadsheetDoc(oDoc) As Boolean
    On Local Error GoTo NODOCUMENTTYPE
    IsSpreadsheetDoc =oDoc.SupportsService(_
        "com.sun.star.sheet.SpreadsheetDocument")
NODOCUMENTTYPE:
    If Err <> 0 Then
        IsSpreadsheetDoc = False
        Resume GOON
        GOON:
    End If
End Function

```

Afficher la Valeur, le Texte ou la Formule d'une cellule

```

*****
'Auteur : Sasa Kelecevic
'email : scat@teol.net
Sub ExampleGetValue
    Dim oDocument As Object, oSheet As Object, oCell As Object
    oDocument=ThisComponent
    oSheet=oDocument.Sheets.getByName("Feuille1")
    oCell=oSheet.getCellByPosition(0,0) 'A1
    print oCell.getValue
    'print oCell.getString
    'print oCell.getFormula
End sub

```

Définir la Valeur, le Texte ou la Formule d'une cellule

```
*****  
'Auteur : Sasa Kelecevic  
'email : scat@teol.net  
Sub ExampleSetValue  
    Dim oDocument As Object, oSheet As Object, oCell As Object  
    oDocument=ThisComponent  
    oSheet=oDocument.Sheets.getByName("Feuille1")  
    oCell=oSheet.getCellByPosition(0,0) 'A1  
    oCell.setValue(23658)  
    'oCell..NumberFormat=2 '23658.00  
    'oCell.SetString("Oups")  
    'oCell.setFormula("=FUNCTION()")  
    'oCell.IsCellBackgroundTransparent = TRUE  
    'oCell.CellBackColor = RGB(255,141,56)  
End Sub
```

Dans votre document tableur, vous pouvez accéder à une cellule dans un autre document, avec une expression du type :

file:///CHEMIN/NomFichier'#\$Feuil1.P40

On peut aussi le faire dans une macro.

```
oCell = thiscomponent.sheets(0).getcellbyposition(0,0) ' A1  
oCell.setFormula("= " &"file:///home/USER/CalcFile2.sxc'#$Sheet2.K89")
```

Effacer une cellule

Une liste des éléments pouvant être effacés peut être trouvée dans le SDK : com/sun/star/sheet/CellFlags.html

```
*****  
'Auteur : Andrew Pitonyak  
'email : andrew@pitonyak.org  
Sub ClearDefinedRange  
    Dim oDocument As Object, oSheet As Object, oSheets As Object  
    Dim oCellRange As Object  
    Dim nSheets As Long  
        oDocument = ThisComponent  
        oSheets = oDocument.Sheets  
        nSheets = oDocument.Sheets.Count  
        oSheet = oSheets.getByIndex(2) Rem la plage va de 0 à n-1  
        oCellRange = oSheet.getCellRangeByName("range_you_set")  
        oCellRange.clearContents(  
            com.sun.star.sheet.CellFlags.VALUE |_  
            com.sun.star.sheet.CellFlags.DATETIME |_  
            com.sun.star.sheet.CellFlags.STRING |_  
            com.sun.star.sheet.CellFlags.ANNOTATION |_  
            com.sun.star.sheet.CellFlags.FORMULA |_  
            com.sun.star.sheet.CellFlags.HARDATTR |_  
            com.sun.star.sheet.CellFlags.STYLES |_  
            com.sun.star.sheet.CellFlags.OBJECTS |_  
            com.sun.star.sheet.CellFlags.EDITATTR)  
End Sub
```

Qu'est-ce qui est sélectionné ?

Le texte sélectionné dans un tableur peut revêtir différentes formes ; j'en comprends certaines, d'autres non.

1. Une cellule sélectionnée. Cliquez sur une cellule une fois, puis, tout en appuyant sur la touche MAJ, cliquez à nouveau sur la cellule.

2. Une partie de texte sélectionnée dans une cellule. Double cliquez dans une seule cellule, puis sélectionnez du texte.
3. Rien n'est sélectionné. Cliquez une seule fois sur une cellule ou naviguez entre plusieurs cellules.
4. Plusieurs cellules sélectionnées. Un clic unique dans une cellule, puis étirez la sélection.
5. Plusieurs cellules disjointes sélectionnées. Sélectionnez quelques cellules. Maintenez enfoncée la touche « Contrôle » et sélectionnez en quelques autres.

Jusqu'ici, je n'ai pas été en mesure de distinguer les trois premiers cas. Si j'arrive à extraire le texte sélectionné dans le cas 2, alors je peux résoudre ce problème.

```
Function CalcIsAnythingSelected(oDoc As Object) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object, oCursor As Object
    IsAnythingSelected = False
    If IsNull(oDoc) Then Exit Function
    ' La sélection courante dans le contrôleur courant.
    'S'il n'y a pas de contrôleur courant, cela renvoie NULL.
    oSelections = oDoc.getCurrentSelection()
    If IsNull(oSelections) Then Exit Function
    If oSelections.supportsService("com.sun.star.sheet.SheetCell") Then
        Print "Une Cellule sélectionnée = " & oSelections.getImplementationName()
        MsgBox "getString() = " & oSelections.getString()
    ElseIf oSelections.supportsService("com.sun.star.sheet.SheetCellRange") Then
        Print "Une plage de cellules sélectionnée = " & oSelections.getImplementationName()
    ElseIf oSelections.supportsService("com.sun.star.sheet.SheetCellRanges") Then
        Print "Plusieurs plages de cellules sélectionnées = " & oSelections.getImplementationName()
    End If
    Print "Count = " & oSelections.getCount()
    Else
        Print "Autre sélection = " & oSelections.getImplementationName()
    End If
End Function
```

Adresse “affichable” d'une cellule

'Étant donnée une cellule, extraire l'adresse de la cellule sous sa forme habituelle
'D'abord, on extrait le nom de la feuille contenant la cellule.
'Ensuite, on récupère le numéro de la colonne et on le convertit en lettre.
'Enfin, on récupère le numéro de la ligne. Les lignes débutent à zéro mais sont affichées en débutant à 1.

```
Function PrintableAddressOfCell(the_cell As Object) As String
    PrintableAddressOfCell = "Unknown"
    If Not IsNull(the_cell) Then
        PrintableAddressOfCell = the_cell.getSpreadSheet().getName + ":" +
            ColumnNumberToString(the_cell.CellAddress.Column) + (the_cell.CellAddress.Row+1)
    End If
End Function

' Les colonnes sont comptées en partant de 0 où 0 correspond à A
' Elles vont de A à Z, puis AA à AZ,BA à BZ,...,jusqu'à IV
' Il s'agit donc essentiellement de la façon de convertir un nombre en base 10 en un nombre
' en base 26.
' Notez que la colonne est passée en valeur (ByVal) !
Function ColumnNumberToString(ByVal the_column As Long) As String
    Dim s$
```

```

'Enregistrez le paramètre dans une variable pour NE PAS le changer.
'C'est un sale bug que j'ai mis du temps à trouver
Do
    s$ = Chr(65 + the_column MOD 26) + s$
    the_column = the_column / 26
Loop Until the_column = 0
ColumnNumberToString = s$
End Function

```

Insérer une date formatée dans une cellule

Cette macro insère la date dans la cellule sélectionnée. Si le document en cours n'est pas un document de tableur, un message d'erreur apparaîtra. Du code permet de formater la date selon votre choix, vous devez dé-commenter celui correspondant à votre choix :

```

*****Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
'utilise :FindCreateNumberFormatStyle
Sub InsertDateIntoCell
    Dim oDesktop As Object, oController As Object, oSelection As Object
    Dim doc As Object

    oDesktop = createUnoService("com.sun.star.frame.Desktop")
    oController = oDesktop.CurrentFrame.Controller
    doc = oController.Model

    If doc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
        oSelection = oController.Selection

        ' Définit la valeur de la date
        oFunction = CreateUnoService("com.sun.star.sheet.FunctionAccess")
        oFunction.NullDate = doc.NullDate
        Dim aEmpty()
        oSelection.Value = oFunction.callFunction("NOW", aEmpty())

        ' Met le format de date à sa valeur par défaut
        oFormats = doc.NumberFormats
        dim aLocale as new com.sun.star.lang.Locale
        oSelection.NumberFormat = oFormats.getStandardFormat(
            com.sun.star.util.NumberFormat.DATETIME, aLocale)

        ' Met le format à une valeur complètement différente
        'oSelection.NumberFormat = FindCreateNumberFormatStyle(
        '    "YYYYMMDD.hhmmss", doc)
    Else
        'Merci Russ Phillips pour avoir remarqué que je devais déplacer
        'la commande Else vers le bas !
        MsgBox "Cette macro doit être lancée dans un document tableur"
    End If
End Sub

```

Afficher la plage sélectionnée dans une boîte de dialogue

```

*****Auteur : Sasa Kelecevic
'email :scat@teol.net
Cette macro va prendre la plage sélectionnée et affichera une boîte de dialogue
'indiquant la plage sélectionnée et le nombre de cellules sélectionnées.
Sub SelectedCells
    oSelect=ThisComponent.CurrentSelection.getRangeAddress

```

```

oSelectColumn=ThisComponent.CurrentSelection.Columns
oSelectRow=ThisComponent.CurrentSelection.Rows

CountColumn=oSelectColumn.getCount
CountRow=oSelectRow.getCount

oSelectSC=oSelectColumn.getByIndex(0).getName
oSelectEC=oSelectColumn.getByIndex(CountColumn-1).getName

oSelectSR=oSelect.StartRow+1
oSelectER=oSelect.EndRow+1
NoCell=(CountColumn*CountRow)

If CountColumn=1 AND CountRow=1 Then
    MsgBox("Cellule " + oSelectSC + oSelectSR + chr(13) + "Nb Cellules = " + NoCell,,,
"Cellules Sélectionnées")
Else
    MsgBox("Plage(" + oSelectSC + oSelectSR + ":" + oSelectEC + oSelectER + ")" + chr(13) +
"Nb Cellules = " + NoCell,, "Cellules Sélectionnées")
End If
End Sub

```

Remplir la plage sélectionnée avec un texte

Cette macro simple balaye les lignes et colonnes sélectionnées en y mettant le texte “OOOPS” :

```

'*****
'Auteur : Sasa Kelecevic
'email :scat@teol.net
Sub FillCells
    oSelect=ThisComponent.CurrentSelection
    oColumn=oSelect.Columns
    oRow=oSelect.Rows
    For nc= 0 To oColumn.getCount-1
        For nr = 0 To oRow.getCount-1
            oCell=oSelect.getCellByPosition (nc,nr).setString ("OOOPS")
        Next nr
    Next nc
End Sub

```

Quelques stats sur une plage sélectionnée

```

'*****
'Auteur : Sasa Kelecevic
'email :scat@teol.net
'Cette macro va placer des formules contenant diverses statistiques
'sur la sélection en cours, dans les cellules H7 à H15

Sub Analize
    sSum=="=SUM("+GetAddress+")"
    sAverage=="=AVERAGE("+GetAddress+")"
    sMin=="=MIN("+GetAddress+")"
    sMax=="=MAX("+GetAddress+")"
    CellPos(7,6).setString(GetAddress)
    CellPos(7,8).setFormula(sSum)
    CellPos(7,8).NumberFormat=2
    CellPos(7,10).setFormula(sAverage)
    CellPos(7,10).NumberFormat=2
    CellPos(7,12).setFormula(sMin)
    CellPos(7,12).NumberFormat=2
    CellPos(7,14).setFormula(sMax)
    CellPos(7,14).NumberFormat=2
End sub

```

```

Function GetAddress  'selected cell(s)
    oSelect=ThisComponent.CurrentSelection.getRangeAddress
    oSelectColumn=ThisComponent.CurrentSelection.Columns
    oSelectRow=ThisComponent.CurrentSelection.Rows

    CountColumn=oSelectColumn.getCount
    CountRow=oSelectRow.getCount

    oSelectSC=oSelectColumn.getByIndex(0).getName
    oSelectEC=oSelectColumn.getByIndex(CountColumn-1).getName

    oSelectSR=oSelect.StartRow+1
    oSelectER=oSelect.EndRow+1
    NoCell=(CountColumn*CountRow)

    If CountColumn=1 AND CountRow=1 then
        GetAddress=oSelectSC+oSelectSR
    Else
        GetAddress=oSelectSC+oSelectSR+":"+oSelectEC+oSelectER
    End If
End Function

Function CellPos(lColumn As Long,lRow As Long)
    CellPos= ActiveSheet.getCellByPosition (lColumn,lRow)
End Function

Function ActiveSheet
    ActiveSheet=StarDesktop.CurrentComponent.CurrentController.ActiveSheet
End Function

Sub DeleteDbRange(sRangeName As String)
    oRange=ThisComponent.DatabaseRanges
    oRange.removeByName (sRangeName)
End Sub

```

Définir une plage comme plage de données

```

*****
'Auteur : Sasa Kelecevic
'email :scat@teol.net
Sub DefineDbRange(sRangeName As String) 'plage sélectionnée
    On Error GoTo DUPLICATENAME
    oSelect=ThisComponent.CurrentSelection.RangeAddress
    oRange=ThisComponent.DatabaseRanges.addNewByName (sRangeName,oSelect )
DUPLICATENAME:
    If Err <> 0 Then
        MsgBox("Nom dupliqué","","INFORMATION")
    End If
End Sub

```

Supprimer une plage de données

```

*****
'Auteur : Sasa Kelecevic
'email :scat@teol.net
Sub DeleteDbRange(sRangeName As String)
    oRange=ThisComponent.DatabaseRanges
    oRange.removeByName (sRangeName)
End Sub

```

Tracer le contour d'une plage

Vous modifiez ici une structure temporaire. Utilisez quelque chose comme :

```

*****
'Auteur : Niklas Nebel

```

```

'email :niklas.nebel@sun.com
'Définir les bordures sous Calc
oRange = ThisComponent.Sheets(0).getCellRangeByPosition(0,1,0,63)
aBorder = oRange.TableBorder

aBorder.BottomLine = lHor
oRange.TableBorder = aBorder

```

Il dit toujours que ce qui suit ne fonctionnera pas car cela modifie une structure temporaire.

```

lHor.Color = 0: lHor.InnerLineWidth = 0: lHor.OuterLineWidth = 150:
dim lHor as New com.sun.star.table.BorderLineHor.LineDistance = 0
ThisComponent.Sheets(0).getCellRangeByPosition(0,1,0,63).TableBorder.BottomLine = lHor

```

Trier une plage

La macro suivante trie les données sur la première colonne, par ordre croissant. Par défaut, vous spécifiez les colonnes à trier, et ce sont les lignes qui sont réorganisées.

```

'*****
'Auteur : Sasa Kelecevic
'email :scat@teol.net
Sub SortRange
    Dim oSheetDSC,oDSCRange As Object
    Dim aSortFields(0) as new com.sun.star.util.SortField
    Dim aSortDesc(0) as new com.sun.star.beans.PropertyValue

    'définissez le nom de votre feuille
    oSheetDSC = ThisComponent.Sheets.getByName("Feuille1")

    'Définissez l'adresse de votre plage
    oDSCRange = oSheetDSC.getCellRangeByName("A1:L16")

    ThisComponent.getCurrentController.select(oDSCRange)

    aSortFields(0).Field = 0
    aSortFields(0).SortAscending = FALSE

    aSortDesc(0).Name = "SortFields"
    aSortDesc(0).Value = aSortFields()
    oDSCRange.Sort(aSortDesc())
End sub

```

Supposons que l'on veuille faire un tri sur les seconde et troisième colonnes dont la première est du texte et la seconde doit être triée numériquement. Il nous faudra deux champs de tri au lieu d'un seul.

```

Sub Main
    Dim oSheetDSC As Object, oDSCRange As Object
    Dim aSortFields(1) As New com.sun.star.util.SortField
    Dim aSortDesc(0) As New com.sun.star.beans.PropertyValue

    'Définissez le nom de votre feuille
    oSheetDSC = THISCOMPONENT.Sheets.getByName("Feuille1")

    'définissez l'adresse de votre plage
    oDSCRange = oSheetDSC.getCellRangeByName("B3:E6")
    THISCOMPONENT.getCurrentController.select(oDSCRange)

    'Un autre type de tri pourrait utiliser
    'com.sun.star.util.SortFieldType.AUTOMATIC
    'Souvenez vous que les champs partent de zéro et que donc cette procédure commence

```

```

' à la colonne B et non à la colonne A
aSortFields(0).Field = 1
aSortFields(0).SortAscending = TRUE
'On suppose qu'il n'y a pas de raison de définir le type de champ quand
'on trie dans un tableur car il est ignoré
'Un tableur connaît déjà le type
'aSortFields(0).FieldType = com.sun.star.util.SortFieldType.ALPHANUMERIC

aSortFields(1).Field = 2
aSortFields(1).SortAscending = TRUE
aSortFields(1).FieldType = com.sun.star.util.SortFieldType.NUMERIC

aSortDesc(0).Name = "SortFields"
aSortDesc(0).Value = aSortFields()      ' aSortFields(0)
oDSCRange.Sort(aSortDesc())           ' aSortDesc(0)
End Sub

```

Pour définir la première ligne comme ligne d'en tête, utilisez une autre propriété. Soyez certain d'avoir suffisamment dimensionné vos propriétés.

```

aSortDesc(1).Name = "ContainsHeader"
aSortDesc(1).Value = True

```

Bernard Marcellly a vérifié que la propriété « Orientation » fonctionne correctement. Pour définir l'orientation utilisez la propriété « Orientation » et définissez la valeur à l'une des suivantes :

```

com.sun.star.table.TableOrientation.ROWS
com.sun.star.table.TableOrientation.COLUMNS

```

Quand on trie en utilisant l'interface graphique, on ne peut trier plus de trois colonnes à la fois. Bernard Marcellly a noté que l'on ne peut pas contourner cette limitation en utilisant la macro. On reste limité à trois lignes ou colonnes.

Trouver l'élément dupliqué

Avec un tableau trié, il est facile de trouver un élément dupliqué ! Je recherche le premier et le renvoie :

```

Function FirstDuplicate(sArray() As String) As String
    Dim i&
    FirstDuplicate = ""
    For i = LBound(sArray()) To UBound(sArray()) - 1
        If sArray(i) = sArray(i+1) Then
            FirstDuplicate = sArray(i)
            Exit For
        End If
    Next
End Function

```

Afficher toutes les données d'une colonne

Tout en parcourant une cellule, et en affichant sa valeur, je souhaite afficher des informations à propos de cette cellule. Voici comment je procède :

```

Sub PrintDataInColumn (a_column As Integer)
    Dim oCells As Object, aCell As Object, oDocument As Object
    Dim oColumn As Object, oRanges As Object

    oDocument = ThisComponent
    oColumn = oDocument.Sheets(0).Columns(a_column)
    Print "Utilisation de la colonne column " + oColumn.getName
    oRanges = oDocument.createInstance("com.sun.star.sheet.SheetCellRanges")
    oRanges.insertByName("", oColumn)
    oCells = oRanges.Cells.createEnumeration
    If Not oCells.hasMoreElements Then Print "Désolé, pas de texte à afficher"

```

```

While oCells.hasMoreElements
    aCell = oCells.nextElement
    'La fonction suivante est décrite ailleurs dans ce document !
    MsgBox PrintableAddressOfCell(aCell) + " = " + aCell.String
Wend
End Sub

```

Les Méthodes de Groupement

Ryan Nelson [ryan@aurelius-mfg.com] m'a parlé des possibilités de groupement dans Calc et m'a demandé comment les utiliser dans une macro. Il y a deux choses à garder à l'esprit. La première est que c'est la feuille qui crée et supprime les groupements, et la seconde, c'est que les paramètres doivent être corrects.

<http://api.openoffice.org/docs/common/ref/com/sun/star/sheet/XSheetOutline.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/table/TableOrientation.html>

```

Option Explicit
Sub CalcGroupingExample
    Dim oDoc As Object, oRange As Object, oSheet As Object
    oDoc = ThisComponent
    If Not oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
        MsgBox "Cette macro doit être exécutée dans un document Calc", 64, "Erreur"
    End If
    oSheet=oDoc.Sheets.getByName("Feuille1")
    ' Les paramètres sont (gauche, haut, droite, bas)
    oRange = oSheet.getCellRangeByPosition(2,1,3,2)
    'On aurait aussi pu utiliser COLUMNS ci-dessous
    oSheet.group(oRange.getRangeAddress(), com.sun.star.table.TableOrientation.ROWS)
    Print "Je viens de grouper la plage."
    oSheet.unGroup(oRange.getRangeAddress(), com.sun.star.table.TableOrientation.ROWS)
    Print "Je viens de dégrouper la plage."
End Sub

```

Protéger vos données

Il est facile de protéger vos classeurs, il vous suffit de prendre vos feuilles et de les protéger. Mes essais montrent qu'aucune erreur ne se produit lorsque vous choisissez de protéger un document entier, mais cela ne protège pas le document entier.

```

Sub ProtectSpreadsheet
    dim oDoc As Object, oSheet As Object
    oDoc = ThisComponent
    oSheet=oDoc.Sheets.getByName("Feuille1")
    oSheet.protect("motdepasse")
    Print "Protect value = " & oSheet.isProtected()
    oSheet.unprotect("motdepasse")
    Print "Protect value = " & oSheet.isProtected()
End Sub

```

Définir un texte d'en-tête et de pied de page

Cette macro va définir l'en-tête pour toutes les feuilles à « Feuille : <sheet_name> ». L'en-tête et le pied de page sont définis de façon identique, remplacez simplement « Header » par « Footer » dans les appels.

Remerciements chaleureux à Oliver Brinzing [OliverBrinzing@t-online.de] qui a rempli les cases que j'ignorais, principalement que je devais réécrire l'en-tête

dans le document.

```
Sub SetHeaderTextInSpreadSheet
    Dim oDoc As Object, oSheet As Object, oPstyle As Object, oHeader As Object
    Dim oText As Object, oCursor As Object, oField As Object
    oDoc = ThisComponent
    ' Prenez le style de la feuille sélectionnée.
    oSheet = oDoc.CurrentController.getActiveSheet
    oPstyle = oDoc.StyleFamilies.getByName("PageStyles").getByName(oSheet.PageStyle)

    ' Activez les en-têtes et partagez les !
    oPstyle.HeaderOn = True
    oPstyle.HeaderShared = True

    ' Il y a aussi un RightText et un LeftText
    oHeader = oPstyle.RightPageHeaderContent
    oText = oHeader.CenterText

    ' Vous pouvez maintenant affecter à l'objet texte la valeur que vous souhaitez
    ' Un texte simple peut être défini avec la méthode setString() de l'objet texte.
    ' Cependant, il faut utiliser un curseur pour insérer un champ comme le nom
    ' de la feuille en cours
    ' D'abord, je vais effacer le texte éventuellement existant !
    oText.setString("")
    oCursor = oText.createTextCursor()
    oText.insertString(oCursor, "Sheet: ", False)
    ' Ceci donnera le nom de la feuille active !
    oField = oDoc.createInstance("com.sun.star.text.TextField.SheetName")
    oText.insertTextContent(oCursor, oField, False)

    ' Et maintenant, concernant la partie qui contient l'ensemble,
    ' vous devez réécrire l'objet en-tête car nous avons
    ' modifié un objet temporaire.
    oPstyle.RightPageHeaderContent = oHeader
End Sub
```

Accéder à un contrôle de formulaire dans Calc via OOBasic

Laurent Godard a envoyé ce qui suit que je dois avouer ne pas avoir eu le temps de regarder et de comprendre ! Je regarterai plus tard !

Comment accéder à un contrôle de formulaire dans Calc via OOBasic :

```
Form Name= "Form1"
Control Name = "TheList"
sheet Name = "Feuille 1"
```

Je souhaite avoir la valeur de l'élément sélectionné dans "TheList". Tout est fait par une macro pour lier le changement dans une cellule. Merci à Oliver Brinzing pour m'avoir fourni un exemple :

```
oDocument = ThisComponent
oSheets = oDocument.Sheets
oSheet = oSheets.getByName(SheetName)
oDpage = Osheet.DrawPage
oForm = oDpage.Forms(SheetName).getByName(ControlName)
Result= oForm.text
```

Compter les entrées non vides dans une colonne

```
'oSheet Feuille contenant la colonne
'ICol colonne à examiner
'I_min ignorer les lignes précédentes
'I_max ignorer les lignes suivantes
'Return le nombre de cellules non vides dans la colonne
J'ai utilisé le type "variant" pour les arguments optionnels pour contourner un bug de la version
' 1.0.2 qui ne parvient pas correctement les arguments optionnels pour les autres types.
```

```

Function NonBlankCellsInColumn (oSheet As Object, ICol&,
    Optional l_min As Variant, Optional l_max As Variant) As Long
Dim oCells As Object, oCell As Object, oColumn As Object
Dim oAddr As Object, oRanges As Object
Dim n&, IMin&, IMax&

n = 0
IMin = 0
IMax = 2147483647
If Not IsMissing(l_min) Then IMin = l_min
If Not IsMissing(l_max) Then IMax = l_max

oColumn = oSheet.Columns(ICol)
oRanges = ThisComponent.createInstance("com.sun.star.sheet.SheetCellRanges")
oRanges.insertByName("", oColumn)
oCells = oRanges.Cells.createEnumeration

Do While oCells.hasMoreElements
    oCell = oCells.nextElement
    oAddr = oCell.CellAddress
    If oAddr.Row > IMax Then Exit Do
    If oAddr.Row >= IMin And Len(oCell.String) > 0 Then n=n+1
Loop
NonBlankCellsInColumn = n
End Function

```

Texte sélectionné, Qu'est-ce que c'est ?

Un texte sélectionné est essentiellement une étendue de texte, rien de plus. Après qu'une sélection ait été obtenue, il est possible d'obtenir le texte [getString()] et de définir le texte [setString()]. Bien que les chaînes de caractères (variables) soient limitées à 64K en taille, les sélections ne le sont pas. Il existe quelques occasions, parfois, où les méthodes getString() et setString() ont des résultats que je ne comprends pas. Il vaut donc, probablement, mieux utiliser un curseur pour naviguer dans le texte sélectionné et alors, utiliser les méthodes insertString() et insertControlCharacter() de l'objet Text. La documentation mentionne spécifiquement que les caractères blancs suivants sont supportés par la méthode insertString() : 'espace', 'TAB', 'CR' (qui insérera un saut de paragraphe), et 'LF' (qui insérera un retour à la ligne).

Les textes peuvent être sélectionnés manuellement de telle sorte que le curseur puisse être soit sur la gauche, soit sur la droite de la sélection. Une sélection a à la fois un début et une fin et vous ne pouvez pas savoir à l'avance laquelle est au début et laquelle est à la fin du texte sélectionné. Une méthode concernant ce problème est présentée ci-dessous :

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextRange.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRange.html>

Les Curseurs de Texte, Que Sont-ils ?

Un TextCursor est un TextRange qui peut être déplacé dans un objet texte. Les déplacements standards incluent goLeft, goRight, goUp, et goDown. Le premier paramètre est un entier indiquant de combien de caractères ou de lignes bouger. Le second paramètre est un booléen indiquant si l'étendue du texte sélectionné doit être augmentée (True) ou pas. La valeur True est retournée tant que le mouvement intervient. Si un curseur a sélectionné du texte en se déplaçant à gauche et que vous voulez maintenant le déplacer à droite, vous utiliserez probablement oCursor.goRight(0, False) pour indiquer au curseur de se déplacer à droite et de ne pas sélectionner de texte. Cela ne laissera aucun texte sélectionné.

Un TextCursor a à la fois un début et une fin. Si la position de début est la même que la position de fin, alors aucun texte est sélectionné et la propriété IsCollapsed sera True.

Un TextCursor implémente des interfaces qui autorisent les déplacements et la (re)connaissance de positions spécifiques aux mots, phrases et paragraphes. Ceci peut faire gagner beaucoup de temps.

Curseur.gotoStart() et Curseur.gotoEnd() vont au début et à la fin du document même si le curseur est créé sur une sélection.

Vous devez être attentif au comportement de vos curseurs, car ils ne fonctionnent pas toujours comme dans l'interface utilisateur. Par exemple, un

curseur de ligne vous permet de vous déplacer dans des lignes. Dans l'interface utilisateur, quand vous sautez à la fin de la ligne, le curseur se positionne en fin de ligne. Alors qu'avec un Curseur ***de Texte ?***, le curseur se positionne généralement au début de la ligne suivante. D'après Giuseppe Castagno [castagno@tecsa-srl.it], voici ci-dessous le comportement de la méthode gotoEndOfLine dans le type XlineCursor :

Le curseur saute de temps en temps au début de la ligne suivante, au lieu d'aller à la fin de la ligne courante. Cela semble ne se produire que quand il y a un hyperlien, ou un champ de texte, qui passent à la ligne suivante. Dans le cas d'un hyperlien, le curseur saute au début de la ligne suivante, tandis que pour un champ de texte, le curseur se positionne à la fin du champ de texte, dans la ligne suivante. Pour le champ de texte, ce comportement est conforme avec ce que l'on voit dans l'interface utilisateur, pour l'hyperlien, ce n'est pas le cas.

D'après Christoph Neumann [christoph@sun.com], c'est pour la raison suivante :

L'interface utilisateur place le curseur derrière le dernier caractère visible d'une ligne de texte. Les espaces d'un saut de ligne automatique ne sont pas pris en compte (ni affichés !), et donc ce ne sera pas la vraie fin de la ligne. Dans l'API, le curseur se place derrière le dernier vrai caractère de la ligne – ce qui correspond au même endroit que le premier caractère de la ligne suivante, dans le cas d'un saut de ligne automatique.

<http://api.openoffice.org/docs/common/ref/com/sun/star/view/XViewCursor.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextCursor.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/XWordCursor.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/XSentenceCursor.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/XParagraphCursor.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextRange.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRange.html>

Cadre de travail pour les textes sélectionnés

La plupart des problèmes utilisant une sélection de texte se ressemblent d'un point de vue abstrait (conceptuel).

Si rien n'est sélectionné alors
travailler sur le document entier
sinon
pour chaque zone de sélection
travailler sur chaque zone

La partie difficile qui changera tout le temps est d'écrire une macro qui incrémentera dans une sélection ou entre deux curseurs.

Les documentations stipulent que s'il n'y a pas de contrôleur courant,

`getCurrentSelection()` retournera un null plutôt que les sélections. Je n'ai qu'une compréhension limitée de ceci mais je les prendrai au mot et vérifierai. Si la longueur de sélection est zéro, alors rien n'est sélectionné. Je n'ai jamais vu de longueur de sélection à zéro, mais je le teste systématiquement. Si aucun texte n'est sélectionné, j'ai une sélection de longueur nulle. J'ai vu des exemples où une sélection de longueur nulle est déterminée comme suit :

```
If Len(oSel.getString()) = 0 Then rien n'est sélectionné
```

Le problème avec ceci est qu'il est possible que le texte sélectionné contienne plus de 64K caractères. Je considère que ce n'est pas sûr. La meilleure solution est de créer un curseur texte à partir de la sélection et de tester que le début et la fin sont les mêmes.

```
oCursor = oDoc.Text.CreateTextCursorByRange(oSel)
If oCursor.IsCollapsed() Then rien n'est sélectionné
```

Voici la fonction qui fera le test complet.

```
Function IsAnythingSelected(oDoc As Object) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object, oCursor As Object
    IsAnythingSelected = False
    If IsNull(oDoc) Then Exit Function
    ' La sélection courante dans le contrôleur courant
    ' S'il n'y a pas de contrôleur courant, retourne null
    oSelections = oDoc.getCurrentSelection()
    If IsNull(oSelections) Then Exit Function
    If oSelections.getCount() = 0 Then Exit Function
    If oSelections.getCount() > 1 Then
        IsAnythingSelected = True
    Else
        oSel = oSelections.getByIndex(0)
        oCursor = oDoc.Text.CreateTextCursorByRange(oSel)
        If Not oCursor.IsCollapsed() Then IsAnythingSelected = True
    End If
End Function
```

Obtenir une sélection est compliqué car il est possible d'avoir de multiples sélections non-contiguës. Certaines sélections peuvent être vides, et d'autres non. Le code écrit pour gérer la sélection de texte devrait gérer tous ces cas. L'exemple suivant navigue à travers toutes les sections sélectionnées et les imprime.

```
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub MultipleTextSelectionExample
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    ' La sélection courante dans le contrôleur courant.
    'S'il n'y a pas de contrôleur courant, retourne null.
    oSelections = ThisComponent.getCurrentSelection()
    If Not IsNull(oSelections) Then
        oText = ThisComponent.Text
        lSelCount = oSelections.getCount()
        For lWhichSelection = 0 To lSelCount - 1
            oSel = oSelections.getByIndex(lWhichSelection)
            MsgBox oSel.getString()
        Next
    End If
End Sub
```

Les sélections sont essentiellement des intervalles de texte avec un début et une fin. Bien que les sélections aient à la fois un début et une fin, quel côté du texte est celui qui est déterminé par la méthode de sélection. L'objet Text fournit des méthodes pour comparer les positions de début et de fin d'un intervalle de texte. La méthode "short compareRegionStarts (XTextRange R1, XTextRange R2)" retourne 1 si R1 débute avant R2, 0 si R1 a la même position que R2 et -1 si R1 débute après R2. La méthode "short compareRegionEnds (XTextRange R1, XTextRange R2)" retourne 1, si R1 termine avant R2, 0 si R1 termine à la même position que R2 et -1, si R1 termine derrière R2. J'utilise les deux méthodes suivantes pour trouver les positions du curseur la plus à gauche et la plus à droite d'une sélection de texte :

```
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function GetLeftMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object

    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getEnd()
    Else
        oRange = oSel.getStart()
    End If
    oCursor = oText.CreateTextCursorByRange(oRange)
    oCursor.goRight(0, False)
    GetLeftMostCursor = oCursor
End Function
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function GetRightMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object

    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getStart()
    Else
        oRange = oSel.getEnd()
    End If
    oCursor = oText.CreateTextCursorByRange(oRange)
    oCursor.goLeft(0, False)
    GetRightMostCursor = oCursor
End Function
```

Cela m'a pris un long moment pour comprendre comment balayer les textes sélectionnés en utilisant des curseurs, donc, j'ai écrit plusieurs macros qui font les choses, dans ce que je considère le mauvais sens. Maintenant, j'utilise une structure de haut niveau pour faire ceci. L'idée est que si aucun texte n'est sélectionné, alors, il demande si la macro doit être lancée sur le document entier. Si la réponse est oui, alors un curseur est créé au début et à la fin du document et la macro de travail est appelée. Si du texte est sélectionné, alors chaque sélection est récupérée et un curseur est obtenu au début et à la fin de la sélection ; la macro de travail est alors appelée pour chacune des sélections.

La structure rejetée

J'ai définitivement rejeté la structure qui suit parce qu'elle est trop longue et pénible à répéter chaque fois que je voulais balayer à travers le texte. Elle est pourtant défendable. Vous pouvez préférer cette structure et choisir de l'utiliser :

```
Sub IterateOverSelectedTextFramework
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim ISelCount As Long, IWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    oText = ThisComponent.Text
    If Not IsAnythingSelected(ThisComponent) Then
        Dim i%
        i% = MsgBox("Aucun texte sélectionné !" + Chr(13) + _
                    "Appeler la routine sur le document ENTIER ?", _
                    1 OR 32 OR 256, "Attention")
        If i% <> 1 Then Exit Sub
        oLCursor = oText.createTextCursor()
        oLCursor.gotoStart(False)
        oRCursor = oText.createTextCursor()
        oRCursor.gotoEnd(False)
        CallYourWorkerMacroHere(oLCursor, oRCursor, oText)
    Else
        oSelections = ThisComponent.getCurrentSelection()
        ISelCount = oSelections.getCount()
        For IWhichSelection = 0 To ISelCount - 1
            oSel = oSelections.getByIndex(IWhichSelection)
            'Si je veux savoir si aucun texte n'est sélectionné, je peux
            'faire la chose suivante :
            'oLCursor = oText.CreateTextCursorByRange(oSel)
            'If oLCursor.isCollapsed() Then ...
            oLCursor = GetLeftMostCursor(oSel, oText)
            oRCursor = GetRightMostCursor(oSel, oText)
            CallYourWorkerMacroHere(oLCursor, oRCursor, oText)
        Next
    End If
End Sub
```

Le modèle retenu

J'ai opté pour la création du modèle qui suit. Il retourne un tableau bi-dimensionnel des curseurs de début et de fin à partir desquels balayer le document. Ceci autorise l'utilisation d'un code minimal pour balayer les textes sélectionnés dans le document entier.

```
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
```

```

'sPrompt : Comment demander si balayage sur tout le document
'oCursors() : Contient les curseurs retournés
'Retourne true si balayage ou faux sinon
Function CreateSelectedTextIterator(oDoc As Object, sPrompt As String, oCursors() As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim ISelCount As Long, IWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    CreateSelectedTextIterator = True
    oText = oDoc.Text
    If Not IsAnythingSelected(ThisComponent) Then
        Dim i%
        i% = MsgBox("Aucun texte sélectionné!" + Chr(13) + sPrompt, _
                    1 OR 32 OR 256, "Attention")
        If i% = 1 Then
            oLCursor = oText.createTextCursor()
            oLCursor.gotoStart(False)
            oRCursor = oText.createTextCursor()
            oRCursor.gotoEnd(False)
            oCursors = DimArray(0, 1)
            oCursors(0, 0) = oLCursor
            oCursors(0, 1) = oRCursor
        Else
            oCursors = DimArray()
            CreateSelectedTextIterator = False
        End If
    Else
        oSelections = ThisComponent.getCurrentSelection()
        ISelCount = oSelections.getCount()
        oCursors = DimArray(ISelCount - 1, 1)
        For IWhichSelection = 0 To ISelCount - 1
            oSel = oSelections.getByIndex(IWhichSelection)
            'Si je veux savoir si aucun texte est sélectionné, je peux
            'faire la chose suivante :
            'oLCursor = oText.CreateTextCursorByRange(oSel)
            'If oLCursor.isCollapsed() Then ...
            oLCursor = GetLeftMostCursor(oSel, oText)
            oRCursor = GetRightMostCursor(oSel, oText)
            oCursors(IWhichSelection, 0) = oLCursor
            oCursors(IWhichSelection, 1) = oRCursor
        Next
    End If
End Function

```

La routine principale

Voici un exemple qui appelle la routine principale :

```

Sub PrintExample
    Dim oCursors(), i%
    If Not CreateSelectedTextIterator(ThisComponent,
        "Imprimer les caractères pour le document entier ?", oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        PrintEachCharacterWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub

```

J'ai pondu tout ceci rapidement, et sans trop réfléchir. À utiliser à vos risque et périls !

```

REM Cela ne marchera probablement pas s'il y a des tableaux et équivalents, car le
REM Curseur de Phrase ne pourra pas entrer dans le(s) tableau(x), mais je ne l'ai pas
REM effectivement testé.
Sub CountSentences

```

```

Dim vCursor As Variant 'Variant est plus sûr que Object
Dim vSentanceCursor As Variant 'Variant est plus sûr que Object
Dim vText As Variant
Dim i
vText = ThisComponent.Text
vCursor = vText.CreateTextCursor()
vSentanceCursor = vText.CreateTextCursor()

'Place le curseur au début du document
vCursor.GoToStart(False)
Do While vCursor.gotoNextParagraph(True)
    ' A ce point, le paragraphe entier est sélectionné
    vSentanceCursor.gotoRange(vCursor.getStart(), False)
    Do While vSentanceCursor.gotoNextSentence(True) AND_
        vText.compareRegionEnds(vSentanceCursor, vCursor) >= 0
        vSentanceCursor.goRight(0, False)
        i = i + 1
    Loop
    vCursor.goRight(0, False)
Loop
MsgBox i, 0, "Nombre de phrases"
End Sub

```

Cet exemple simple peut être utilisé avec le modèle ci-dessus :

```

'*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub PrintEachCharacterWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Do While oLCursor.goRight(1, True) AND oText.compareRegionEnds(oLCursor, oRCursor) >= 0
        Print "Caractère = " & oLCursor.getString() & ""
        Rem Ceci fera en sorte que le texte sélectionné
        Rem ne le soit plus
        oLCursor.goRight(0, False)
    Loop
End Sub

```

Cet ensemble de macros enlève (remplace) toutes les répétitions de caractères blancs par un seul caractère blanc. Il est facilement modifiable pour supprimer différents types de caractères blancs. Les différents types d'espaces sont classés par ordre d'importance et donc, si vous avez un espace normal suivi par un nouveau paragraphe, ce nouveau paragraphe restera et l'unique espace sera enlevé. Cela causera la suppression des espaces blancs avant et après une ligne.

Qu'est-ce qu'un espace blanc ?

En résolvant ce problème, ma première tâche a été de déterminer quels caractères sont des espaces blancs. Vous pouvez de façon évidente changer la définition du caractère blanc pour ignorer certains caractères.

'Normalement, ceci est fait avec une recherche dans un tableau, ce qui serait probablement

```
'plus rapide, mais je ne sais pas comment utiliser les déclarations (?) statiques en basic
Function IsWhiteSpace(iChar As Integer) As Boolean
    Select Case iChar
        Case 9, 10, 13, 32, 160
            IsWhiteSpace = True
        Case Else
            IsWhiteSpace = False
    End Select
End Function
```

Priorités des caractères pour la suppression

Ensuite, j'ai eu besoin de définir ce qu'il fallait enlever et ce qu'il fallait garder. J'ai choisi de faire ça avec la routine suivante :

```
'-1 signifie supprimer le caractère précédent
'0 signifie ignorer ce caractère
'1 signifie supprimer ce caractère
'La priorité, de la plus haute à la plus faible, est : 0, 13, 10, 9, 160, 32
Function RankChar(iPrevChar, iCurChar) As Integer
    If Not IsWhiteSpace(iCurChar) Then           'le caractère courant n'est pas un espace blanc,
    l'ignorer
        RankChar = 0
    ElseIf iPrevChar = 0 Then                   'Début d'une ligne et le caractère courant est un espace
    blanc
        RankChar = 1                            ' donc supprimer l'espace blanc.
    ElseIf Not IsWhiteSpace(iPrevChar) Then      'Le caractère courant est un espace blanc mais le
    précédent ne l'est pas
        RankChar = 0                            ' donc, l'ignorer.
    ElseIf iPrevChar = 13 Then                  'Le caractère précédent est un espace blanc avec la
    plus haute priorité
        RankChar = 1                            ' donc supprimer le caractère courant.
    ElseIf iCurChar = 13 Then                  'Le caractère courant est un espace blanc avec la plus
    haute priorité
        RankChar = -1                           ' donc supprimer le caractère précédent.
    ElseIf iPrevChar = 10 Then                'Pas de nouveau paragraphe pour voir si le caractère
    précédent est LF
        RankChar = 1                            ' donc supprimer le caractère courant.
    ElseIf iCurChar = 10 Then                'Pas de nouveau paragraphe pour voir si le caractère
    courant est LF
        RankChar = -1                           ' donc supprimer le caractère précédent.
    ElseIf iPrevChar = 9 Then                'Pas de nouvelle ligne pour voir si le caractère
    précédent est TAB
        RankChar = 1                            ' donc supprimer le caractère courant.
    ElseIf iCurChar = 9 Then                'Pas de nouvelle ligne pour voir si le caractère
    courant est TAB
        RankChar = -1                           ' donc supprimer le caractère précédent.
    ElseIf iPrevChar = 160 Then              'Pas de TAB pour voir si le caractère précédent est un
    espace insécable
        RankChar = 1                            ' donc supprimer le caractère courant.
    ElseIf iCurChar = 160 Then              'Pas de TAB pour voir si le caractère courant est un
    espace insécable
        RankChar = -1                           ' donc supprimer le caractère précédent.
    ElseIf iPrevChar = 32 Then              'Pas d'espace insécable pour voir si le caractère
    précédent est un espace
        RankChar = 1                            ' donc supprimer le caractère courant.
    ElseIf iCurChar = 32 Then              'Pas d'espace insécable pour voir si le caractère courant
    est un espace
        RankChar = -1                           ' donc supprimer le caractère précédent.
    Else
        RankChar = 0
    End If
End Function
```

L'itérateur standard de texte sélectionné

C'est la manière standard pour décider si le travail doit être fait sur le document entier ou juste sur une portion.

```
'Enlevez toutes les occurrences d'espace vide !
'Si le texte est sélectionné, alors il sera enlevé seulement de la région sélectionnée.
Sub RemoveEmptySpace
    Dim oCursors(), i%
    If Not CreateSelectedTextIterator(ThisComponent,
        "TOUS les espaces vides seront enlevés du document ENTIER ?", oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        RemoveEmptySpaceWorker (oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
```

La routine de travail

C'est là où le vrai travail se passe :

```
Sub RemoveEmptySpaceWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    Dim sParText As String, i As Integer
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub

    Dim iLastChar As Integer, iThisChar As Integer, iRank As Integer
    iLastChar = 0
    iThisChar = 0
    oLCursor.goRight(0, False)
    Do While oLCursor.goRight(1, True)
        iThisChar = Asc(oLCursor.getString())
        i = oText.compareRegionEnds(oLCursor, oRCursor)

        'Si au dernier caractère !
        'Alors toujours enlever l'espace blanc
        If i = 0 Then
            If IsWhiteSpace(iThisChar) Then oLCursor.setString("")
            Exit Do
        End If

        'Si on dépasse la fin, alors on sort de la routine
        If i < 0 Then Exit Do
        iRank = RankChar(iLastChar, iThisChar)
        If iRank = 1 Then

            'Je vais effacer ce caractère.
            'Je ne change pas iLastChar parce qu'il n'a pas changé !
            'Print "Effacer le Courant avec " + iLastChar + " et " + iThisChar
            oLCursor.setString("")
        ElseIf iRank = -1 Then

            'Cela désélectionnera le caractère sélectionné et alors en sélectionne un
            'plus à gauche.
            oLCursor.goLeft(2, True)

            'Print "Effacer avec à gauche " + iLastChar + " et " + iThisChar
            oLCursor.setString("")
            oLCursor.goRight(1, False)
            iLastChar = iThisChar
        Else
            oLCursor.goRight(0, False)
            iLastChar = iThisChar
        End If
    Loop
End Sub
```

Il est préférable de paramétrier « AutoFormat » pour supprimer les paragraphes vides, et de l'appliquer au document en question. Cliquez sur « Outils=>AutoCorrection/AutoFormat... » et choisissez l'onglet « Options ». Une des options est « Supprimer les paragraphes vides ». Vérifiez que cette entrée est cochée. Maintenant, vous pouvez appliquer l'AutoFormat (*NdT : Format-AutoFormat-Appliquer*) et tous les paragraphes vides sont supprimés.

Si vous ne voulez supprimer que les paragraphes vides sélectionnés, alors vous aurez besoin d'une macro. Si du texte est sélectionné, alors les paragraphes vides sont supprimés à l'intérieur de celui-ci. Si aucun texte n'est sélectionné, alors les paragraphes vides sont supprimés du document entier. Cette première macro se répète à travers tous les textes sélectionnés. Si aucun texte n'est sélectionné, elle crée un curseur au début et à la fin du document et travaille sur le document entier. La chose fondamentale à voir dans cette macro est comment traverser le texte basé sur les paragraphes. La macro enlevant les espaces vides est la macro la plus sûre parce qu'elle n'extrait pas de chaîne pour travailler.

```
Sub RemoveEmptyParsWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    Dim sParText As String, i As Integer
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Do While oLCursor.gotoNextParagraph(TRUE) AND oText.compareRegionEnds(oLCursor,
oRCursor) > 0

        'Oui, je sais, limité à 64K ici
        'Si nous avons un paragraphe qui fait plus de 64K
        'Alors j'ai un problème !
        sParText = oLCursor.getString()
        i = Len(sParText)
        'On ne dispose pas d'évaluation logique optimisée. Zut !
        Do While i > 0
            If (Mid(sParText,i,1) = Chr(10)) OR (Mid(sParText,i,1) = Chr(13)) Then
                i = i - 1
            Else
                i = -1
            End If
        Loop
        If i = 0 Then
            oLCursor.setString("")
        Else
            oLCursor.goLeft(0,FALSE)
        End If
    loop
End Sub
```

Toute personne ayant étudié l'algorithmie vous dira qu'il vaut mieux un algorithme meilleur plutôt qu'une machine plus puissante. Quand j'ai commencé à écrire une macro manipulant les lignes blanches et les espaces, je l'ai écrite en utilisant des variables de chaîne de caractères. Cela a introduit la possibilité de perdre les informations de formatage et des erreurs quand cette variable excédait 64 Ko. J'ai donc réécrit cette macro en utilisant les curseurs

mais j'ai alors reçu des plaintes comme quoi elle était trop lente. Une question surgit alors : Existe-t-il un meilleur moyen ?

Recherche dans le texte sélectionné pour compter les mots

Andrew Brown, qui maintient le site <http://www.darwinwars.com> (contenant des informations utiles sur les macros) a posé la question pour faire une recherche dans une région sélectionnée. J'ai découvert que c'était pour compter les mots d'un document et que c'était très lent, trop lent.

Utilisation de String pour compter les mots

Le code existant comptait le nombre d'espaces dans la portion sélectionnée et se servait de ce comptage pour déterminer le nombre de mots. J'ai écrit ma propre version un peu plus généraliste, légèrement plus rapide et donnant la bonne réponse.

```
Function ADPWordCountStrings(vDoc) As String
    Rem mettre ici le caractère que l'on veut comme séparateur de mot
    Dim sSeps$
    sSeps = Chr$(9) & Chr$(13) & Chr$(10) & " ;."
    Dim bSeps(256) As Boolean, i As Long
    For i = LBound(bSeps()) To UBound(bSeps())
        bSeps(i) = False
    Next
    For i = 1 To Len(sSeps)
        bSeps(Asc(Mid(sSeps, i, 1))) = True
    Next

    Dim nSelChars As Long, nSelwords As Long, nSel%, nNonEmptySel%, j As Long, s$
    Dim vSelections, vSel, vText, vCursor
    ' La sélection en cours dans le contrôleur courant.
    ' Si il n'y a pas de contrôleur en cours, retourne Null.
    vSelections = vDoc.getCurrentSelection()
    If IsNull(vSelections) Then
        nSel = 0
    Else
        nSel = vSelections.getCount()
    End If
    nNonEmptySel = 0
    Dim ITemp As Long, bBetweenWords As Boolean, blsSep As Boolean
    On Local Error Goto BadOutOfRange
    Do While nSel > 0
        nSel = nSel - 1
        s = vSelections.GetByIndex(nSel).getString()
        Rem Regarde si c'est une sélection vide
        ITemp = Len(s)
        If ITemp > 0 Then
            nSelChars = nSelChars + ITemp
            nNonEmptySel = nNonEmptySel + 1
            Rem Est ce que ça commence sur un mot ?
            If bSeps(Asc(Mid(s, 1, 1))) Then
                bBetweenWords = True
            Else
                bBetweenWords = False
                nSelWords = nSelWords + 1
            End If
            For j = 2 To ITemp
                blsSep = bSeps(Asc(Mid(s, j, 1)))
                If bBetweenWords <> blsSep Then
                    If bBetweenWords Then
                        Rem Compte un mot nouveau seulement si j'étais entre deux 2 mots
                        Rem et que je ne le suis plus
                    End If
                End If
            Next j
        End If
    Loop
    ADPWordCountStrings = nSelWords
End Function
```

```

        bBetweenWords = False
        nSelWords = nSelWords + 1
    Else
        bBetweenWords = True
    End If
End If
Next
End If
Loop
On Local Error Goto 0

Dim nAllChars As Long, nAllWords As Long, nAllPars As Long
' Accède aux statistiques du document
nAllChars = vDoc.CharacterCount
nAllWords = vDoc.WordCount
nAllPars = vDoc.ParagraphCount

Dim sRes$
sRes = "Compteurs du document:" & chr(13) & nAllWords & " mots. " &
      chr(13) & "(" & nAllChars & " caractères." & chr(13) & nAllPars & _
      " Paragraphes.)" & chr(13) & chr(13)
If nNonEmptySel > 0 Then
    sRes = sRes & "Compteurs du texte sélectionné:" & chr(13) & nSelWords & _
          " mots" & chr(13) & "(" & nSelChars & " caractères)" &
          chr(13) & "dans " & str(nNonEmptySel) & " sélection"
    If nNonEmptySel > 1 Then sRes = sRes & "s"
    sRes = sRes & "." & chr(13) & chr(13) & "Document minus selected:" &
          chr(13) & str(nAllWords - nSelWords) & " mots."
End If
'MsgBox(sRes,64,"ADP Word Count")
ADPWordCountStrings = sRes
Exit Function

BadOutOfRange:
bIsSep = False
Resume Next
End Function

```

Chaque plage sélectionnée est extraite dans une variable string. Cette méthode échoue si la plage est supérieure à 64 K. La valeur ASCII de chaque caractère est contrôlée pour vérifier s'il doit être considéré comme un caractère de séparation de mot. C'est fait par une recherche dans un tableau. C'était efficace mais échouait s'il y avait un caractère spécial avec une valeur ASCII supérieure au tableau. Une gestion d'erreur a donc été utilisée. Un traitement spécial est effectué pour que des valeurs correctes avec différentes selections. Cela a pris 2.7 secondes pour contrôler 8000 mots.

Utilisation d'un curseur de caractère pour compter les mots

Comme tentative de contourner la limite de 64K, j'ai écrit une version utilisant des curseurs pour traverser le texte caractère par caractère. Cette version a pris 47 secondes pour compter les 8000 mots. On utilise la même méthode que précédemment mais la surcharge pour utiliser un curseur sur chaque caractère est prohibitive.

```

*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Function ADPWordCountCharCursor(vDoc) As String
    Dim oCursors(), i%, INumWords As Long
    INumWords = 0
    If Not CreateSelectedTextIterator(vDoc, _
        "Compter les mots de tout le document ?", oCursors()) Then Exit Function

```

```

For i% = LBound(oCursors()) To UBound(oCursors())
    INumWords = INumWords + WordCountCharCursor(oCursors(i%, 0), oCursors(i%, 1),
vDoc.Text)
Next
ADPWordCountCharCursor = "Total des mots = " & INumWords
End Function
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Function WordCountCharCursor(oLCursor, oRCursor, vText)
    Dim INumWords As Long
    INumWords = 0
    WordCountCharCursor = INumWords
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(vText) Then Exit Function
    If vText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Function

    Dim sSeps$
    sSeps = Chr$(9) & Chr$(13) & Chr$(10) & " ;."
    Dim bSeps(256) As Boolean, i As Long
    For i = LBound(bSeps()) To UBound(bSeps())
        bSeps(i) = False
    Next
    For i = 1 To Len(sSeps)
        bSeps(Asc(Mid(sSeps, i, 1))) = True
    Next

    On Local Error Goto BadOutOfRange
    Dim bBetweenWords As Boolean, blsSep As Boolean
    oLCursor.goRight(0, False)
    oLCursor.goRight(1, True)
    Rem Est ce que ça commence sur un mot ?
    If bSeps(Asc(oLCursor.getString())) Then
        bBetweenWords = True
    Else
        bBetweenWords = False
        INumWords = INumWords + 1
    End If
    oLCursor.goRight(0, False)

    Do While oLCursor.goRight(1, True) AND vText.compareRegionEnds(oLCursor, oRCursor) >= 0
        blsSep = bSeps(Asc(oLCursor.getString()))
        If bBetweenWords <> blsSep Then
            If bBetweenWords Then
                Rem Compte un mot nouveau seulement si j'étais entre deux 2 mots
                Rem et que je ne le suis plus
                bBetweenWords = False
                INumWords = INumWords + 1
            Else
                bBetweenWords = True
            End If
        End If
        oLCursor.goRight(0, False)
    Loop
    WordCountCharCursor = INumWords
    Exit Function

BadOutOfRange:
    blsSep = False
    Resume Next
End Function

```

Utilisation d'un curseur de mot pour le comptage

C'est actuellement la méthode la plus rapide. Cette macro utilise un curseur de mots et laisse OOo trouver où les mots commencent et se terminent. Elle va

parcourir les 8000 mots en 1.7 secondes. Cette macro avance de mot en mot, comptant combien de ruptures de mots elle trouve. Le résultat doit donc être incrémenté de 1.

```
'*****  
'Auteur : Andrew Pitonyak  
'email : andrew@pitonyak.org  
Function ADPWordCountWordCursor(vDoc) As String  
    Dim oCursors(), i%, INumWords As Long  
    INumWords = 0  
    If Not CreateSelectedTextIterator(vDoc, _  
        "Compter les mots de tout le document ?", oCursors()) Then Exit Function  
    For i% = LBound(oCursors()) To UBound(oCursors())  
        INumWords = INumWords + WordCountWordCursor(oCursors(i%, 0), oCursors(i%, 1),  
vDoc.Text)  
    Next  
    ADPWordCountWordCursor = "Total des mots = " & INumWords  
End Function  
*****  
'Auteur : Andrew Pitonyak  
'email : andrew@pitonyak.org  
Function WordCountWordCursor(oLCursor, oRCursor, vText)  
    Dim INumWords As Long  
    INumWords = 0  
    WordCountWordCursor = INumWords  
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(vText) Then Exit Function  
    If vText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Function  
    oLCursor.goRight(0, False)  
    Do While oLCursor.gotoNextWord(False) AND vText.compareRegionEnds(oLCursor, oRCursor)  
    >= 0  
        INumWords = INumWords + 1  
    Loop  
    WordCountWordCursor = INumWords  
End Function
```

Réflexions finales sur le comptage et le temps d'exécution

Si votre solution à un problème est trop lente, il existe peut être un autre moyen. Dans OOo, les curseurs peuvent se déplacer suivant les caractères, les mots et les paragraphes. Le curseur utilisé peut amener de grandes différences lors de l'exécution.

Si vous désirez compter le nombre de mots d'une sélection, je vous recommande de regarder le site de Andrew Brown, <http://www.darwinwars.com> car il y travaille activement. Aux dernières nouvelles, il abordait les choses en mettant les mots dans des tables.

La macro suivante m'a été envoyée par Andrew Brown, comme mentionné précédemment. N'hésitez pas à aller sur son site web, il y a beaucoup de choses intéressantes. Et pour ceux qui l'ignorent, il a écrit un livre. Cela ne parle pas de programmation des macros, mais j'ai beaucoup aimé certaines parties. À tester !

```
Sub acbwc  
'v2.0.1  
'5 sept 2003  
' inclut les notes de bas de page et les sélections de toutes tailles  
' encore lent avec les sélections de grande taille, mais c'est de la faute d'Hamburg :-)  
' v 2.0.1 un peu plus rapide, avec une meilleure routine cursorcount  
' n'est plus infiniment longue quand il y a beaucoup de notes de pages.
```

```

' acb, juin 2003
' version réécrite
' de la macro dvwc, par moi et Daniel Vogelheim

' septembre 2003 J'ai changé le comptable en utilisant un curseur de mot pour les sélections de
grande taille
' D'après les conseils d'Andrew Pitonyak.
' Ce n'est pas parfait, malgré tout, essentiellement parce que les déplacements mot par mot sont
erratiques.
' Cela exagère légèrement le nombre de mots dans une sélection, en comptant
' Les sauts de paragraphe et quelques éléments de ponctuation comme des mots.
' Mais c'est bien plus rapide que l'ancienne méthode.

Dim xDoc, xSel, nSelcount
Dim nAllChars
Dim nAllWords
Dim nAllPars
Dim thisrange, sRes
Dim nSelChars, nSelwords, nSel
Dim atext, bigtext
Dim fnotes, thisnote, nfnotes, fnotecount
Dim oddthing,startcursor,stopcursor
xDoc = thiscomponent
xSel = xDoc.getCurrentSelection()
nSelCount = xSel.getCount()
bigText=xDoc.getText()

' À la demande de nos auditeurs...
fnotes=xdoc.getFootNotes()
If fnotes.hasElements() Then
    fnotecount=0
    For nfnotes=0 To fnotes.getCount()-1
        thisnote=fnotes.getByIndex(nfnotes)
        startcursor=thisnote.getStart()
        stopcursor=thisnote.getEnd()
        Do While thisnote.getText().compareRegionStarts(startcursor,stopcursor) AND _
            startcursor.gotoNextWord(FALSE)
            fnotecount=fnotecount+1
        Loop
        msgbox(startcursor.getString())
        fnotecount=fnotecount+stringcount(thisnote.getString())
        fnotecount=fnotecount+CursorCount(thisnote,bigtext)
    Next nfnotes
End If

' Le prochain "If" résout le problème suivant : si vous aviez sélectionné du texte,
' puis l'avez désélectionné, et refait le comptage, alors la sélection vide était encore décomptée
' ce qui était trompeur et inesthétique
If nSelCount=1 and xSel.getByIndex(0).getString()="" Then
    nSelCount=0
End If

' accès aux statistiques du document
nAllChars = xDoc.CharacterCount
nAllWords = xDoc.WordCount
nAllPars = xDoc.ParagraphCount

' mise à zéro des compteurs
nSelChars = 0
nSelWords = 0
' la partie marrante commence ici
' itération sur plusieurs sélections
For nSel = 0 To nSelCount - 1
    thisrange=xSel.GetByIndex(nSel)
    atext=thisrange.getString()
    If len(atext)< 220 Then

```

```

        nselwords=nSelWords+stringcount(atext)
    Else
        nselwords=nSelWords+Cursorcount(thisrange)
    End If
    nSelChars=nSelChars+len(atext)
Next nSel

' réécriture des boîtes de dialogue, pour une meilleure lisibilité
If fnotes.hasElements() Then
    sRes="Total du document (y compris notes de bas de page) : " + nAllWords + " Mots. " + chr(13)
    sRes= sRes + "Nb mots sans les notes : " + str(nAllWords-fnotecount) +_
        " mots. " + chr(13)+"(Total: " +nAllChars +" caractères dans "
Else
    sRes= "Total document : " + nAllWords + " mots. " + chr(13)+ "(" +_
        nAllChars +" caractères dans "
End If
sRes=sRes + nAllPars +" Paragraphes.)"+ chr(13)+ chr(13)
If nselCount>0 Then
    sRes=sRes + "Texte sélectionné : " + nSelWords + " mots" + chr(13) +_
        "(" + nSelChars + " caractères"
If nSelcount=1 Then
    sRes=sRes + " dans " + str(nselCount) + " sélection(s).)"
Else
    REM Je ne sais pas pourquoi, mais il faut l'ajustement suivant
    sRes=sRes + " dans " + str(nselCount-1) +" sélection(s).)"
End If
sRes=sRes+chr(13)+chr(13)+"Document sans les sélections :" + chr(13)+_
    str(nAllWords-nSelWords) + " mots." +chr(13) +chr(13)
End If
If fnotes.hasElements() Then
    sRes=sRes+"Il y a "+ str(fnotecount) + " mots dans "+ fnotes.getCount() +_
        " note(s) de bas de page." +chr(13) +chr(13)
End If
msgbox(sRes,64,"Compteur de Mots acb")
End Sub

function Cursorcount(aRange)
' acb septembre 2003
' compteur rapide pour les grandes sélections
' fondé sur la fonction WordCountWordCursor() développée par Andrew Pitonyak
' Mais rendue plus grossière, selon ma tendance naturelle,
Dim Inumwords as long
Dim atext
Dim startcursor, stopcursor as object
atext=arange.getText()
Inumwords=0
If not atext.compareRegionStarts(aRange.getStart(),aRange.getEnd()) Then
    startcursor=atext.createTextCursorByRange(aRange.getStart())
    stopcursor=atext.createTextCursorByRange(aRange.getEnd())
Else
    startcursor=atext.createTextCursorByRange(aRange.getEnd())
    stopcursor=atext.createTextCursorByRange(aRange.getStart())
End If
Do while aText.compareRegionEnds(startCursor, stopcursor) >= 0 and _
    startCursor.gotoNextWord(False)
    Inumwords=Inumwords+1
Loop
CursorCount=Inumwords-1
end function

Function stringcount(astring)
' acb juin 2003
' compteur de mots plus lent, mais plus précis
' à utiliser avec des sélections plus petites

```

```

' affûté par David Hammerton (http://crazney.net/) en septembre 2003
' pour sauver d'un juste courroux ceux qui mettent deux espaces après les signes de ponctuation.
Dim nspaces,i,testchar,nextchar
nspaces=0
For i= 1 To Len(astring)-1
    testchar=Mid(astring,i,1)
    Select Case testchar
        Case " ",Chr(9),Chr(13)
            nextchar = Mid(astring,i+1,1)
            Select Case nextchar
                Case " ",Chr(9),Chr(13),Chr(10)
                    nspaces=nspaces
                Case Else
                    nspaces=nspaces+1
            End Select
        End Select
    Next i
    stringcount=nspaces+1
End Function

```

Remplacer l'espace sélectionné en utilisant des chaînes de caractères

En général, vous ne devriez pas enlever d'espace supplémentaire en lisant le texte sélectionné et en écrivant de nouvelles valeurs en retour. Une des raisons est que les chaînes de caractères sont limitées à 64K, et l'autre, qu'il est possible de perdre de l'information de formatage. J'ai laissé ces exemples en place parce qu'ils fonctionnent pour résoudre les problèmes pour lesquels ils ont été écrits avant que j'aie appris comment je pouvais faire la même chose avec les curseurs. Ils démontrent également des techniques d'insertion de caractères spéciaux. Cette première macro remplace tous les nouveaux paragraphes et nouvelles lignes avec un caractère d'espace. Ce sont aussi des exemples qui démontrent comment insérer des caractères de contrôle (CR, LF etc.) dans le texte.

```

Sub SelectedNewLinesToSpaces
    Dim lSelCount&, oSelections As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim s$, bSomethingChanged As Boolean

    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    For iWhichSelection = 0 To lSelCount - 1
        bSomethingChanged = False

        Rem et si la chaîne de caractères est plus grande que 64K ? Oups
        s = oSelections.getByIndex(iWhichSelection).getString()
        lIndex = 1
        Do While lIndex < Len(s)
            Select Case Asc(Mid(s, lIndex, 1))
                Case 13
                    'Nous avons trouvé un nouveau marqueur du paragraphe.
                    'Le prochain caractère sera un 10 !
                    If lIndex < Len(s) And Asc(Mid(s, lIndex+1, 1)) = 10 Then
                        Mid(s, lIndex, 2, " ")
                    Else
                        Mid(s, lIndex, 1, " ")
                    End If
                    lIndex = lIndex + 1
                    bSomethingChanged = True
                Case 10
                    'Nouvelle ligne à moins que le caractère précédent soit un 13
            End Select
        Loop
    End For
End Sub

```

```

'Enlever cette déclaration "Case 10" pour ignorer seulement nouvelles lignes !
If lIndex > 1 And Asc(Mid(s, lIndex-1, 1)) <> 13 Then
    'C'est vraiment une nouvelle ligne et PAS un nouveau paragraphe.
    Mid(s, lIndex, 1, " ")
    lIndex = lIndex + 1
    bSomethingChanged = True
Else
    'et non ! celui-ci était vraiment un nouveau paragraphe !
    lIndex = lIndex + 1
End If
Case Else
    'Ne rien faire si nous ne trouvons pas quelque chose d'autre
    lIndex = lIndex + 1
End Select
Loop
If bSomethingChanged Then
    oSelections.getByIndex(iWhichSelection).setString(s)
End If
Next
End Sub

```

Il m'a aussi été demandé de convertir de nouveaux paragraphes en nouvelles lignes. Utiliser des curseurs est clairement une meilleure idée, mais je ne savais pas comment le faire. Je pense que cet exemple est encore instructif, donc je l'ai laissé. J'efface en premier le texte sélectionné et ensuite, commence à rajouter le texte :

```

Sub SelectedNewParagraphsToNewLines
    Dim lSelCount&, oSelections As Object, oSelection As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim oText As Object, oCursor As Object
    Dim s$, lLastCR As Long, lLastNL As Long

    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    oText = ThisComponent.Text

    For iWhichSelection = 0 To lSelCount - 1
        oSelection = oSelections.getByIndex(iWhichSelection)
        oCursor = oText.createTextCursorByRange(oSelection)
        s = oSelection.getString()

        'Supprimer le texte sélectionné
        oCursor.setString("")
        lIndex = 1
        Do While lIndex <= Len(s)
            Select Case Asc(Mid(s, lIndex, 1))
                Case 13
                    oText.insertControlCharacter(oCursor,
                        com.sun.star.text.ControlCharacter.LINE_BREAK, False)

                    'J'aurais aimé avoir un booléen court
                    'Passer le prochain LF s'il y en a un. Je pense
                    'qu'il y en aura toujours mais je ne peux pas le vérifier.
                    If (lIndex < Len(s)) Then
                        If Asc(Mid(s, lIndex+1, 1)) = 10 Then lIndex = lIndex + 1
                    End If
                Case 10
                    oText.insertControlCharacter(oCursor,
                        com.sun.star.text.ControlCharacter.LINE_BREAK, False)
                Case Else
                    oCursor.setString(Mid(s, lIndex, 1))
                    oCursor.GoRight(1, False)
            End Select
        lIndex = lIndex + 1
    Next
End Sub

```

```

    lIndex = lIndex + 1
Loop
Next
End Sub

```

Voici quelques macros que j'ai écrites en utilisant les méthodes du curseur et ensuite, la même façon dont je les avais faites avant d'avoir ma structure :

```

*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org

'Le but de cette macro est de faciliter l'utilisation de la méthode Texte<-->Tableau
'qui nécessite la suppression des espaces blancs avant et après.
'Elle nécessite aussi de nouveaux paragraphes et PAS de nouvelles lignes !
Sub CRToNLMain
    Dim oCursors(), i%, sPrompt$

    sPrompt$ = "Convertir les nouveaux paragraphes en nouvelles lignes pour le document
entier ?"
    If Not CreateSelectedTextIterator(ThisComponent, sPrompt$, oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        CRToNLWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
Sub CRToNLWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Do While oLCursor.gotoNextParagraph(False) AND oText.compareRegionEnds(oLCursor,
oRCursor) >= 0
        oLCursor.goLeft(1, True)
        oLCursor.setString("")
        oLCursor.goRight(0, False)
        oText.insertControlCharacter(oLCursor,
            com.sun.star.text.ControlCharacter.LINE_BREAK, True)
    Loop
End Sub
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org

'Le but réel de cette macro est de faciliter la méthode Texte<-->Tableau
'qui nécessite la suppression des espaces blancs avant et après.
'Elle nécessite aussi de nouveaux paragraphes et PAS de nouvelles lignes !
Sub SpaceToTabsInWordsMain
    Dim oCursors(), i%, sPrompt$

    sPrompt$ = "Convertir les espaces en TABULATIONS pour le document ENTIER ?"
    If Not CreateSelectedTextIterator(ThisComponent, sPrompt$, oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        SpaceToTabsInWordsWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
Sub SpaceToTabsInWordsWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    Dim icurrentState As Integer, iChar As Integer, bChanged As Boolean
    Const StartLineState = 0
    Const InWordState = 1
    Const BetweenWordState = 2

    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    icurrentState = StartLineState

```

```

bChanged = False
Do While oLCursor.goRight(1, True) AND oText.compareRegionEnds(oLCursor, oRCursor) >= 0
    iChar = Asc(oLCursor.getString())
    If icurrentState = StartLineState Then
        If IsWhiteSpace(iChar) Then
            oLCursor.setString("")
        Else
            icurrentState = InWordState
        End If
    ElseIf icurrentState = InWordState Then
        bChanged = True
        Select Case iChar
        Case 9
            Rem Il s'agit déjà d'une tabulation, l'ignorer
            icurrentState = BetweenWordState
        Case 32, 160
            Rem Convertit l'espace en une tabulation
            oLCursor.setString(Chr(9))
            oLCursor.goRight(1, False)
            icurrentState = BetweenWordState
        Case 10
            Rem
            oText.insertControlCharacter(oLCursor,
                com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, True)
            oLCursor.goRight(1, False)
            icurrentState = StartLineState
        Case 13
            icurrentState = StartLineState
        Case Else
            oLCursor.gotoEndOfWord(True)
        End Select
    ElseIf icurrentState = BetweenWordState Then
        Select Case iChar
        Case 9, 32, 160
            Rem Nous avons déjà ajouté une tabulation, chose superflue
            oLCursor.setString("")
        Case 10
            Rem Enlève la nouvelle ligne et insère un nouveau paragraphe
            Rem être certain d'effacer la TABulation de tête que nous avions déjà
            Rem ajoutée et cela devrait aller !
            oText.insertControlCharacter(oLCursor,
                com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, True)
            oLCursor.goLeft(0, False)
            Rem and over the TAB for deletion
            oLCursor.goLeft(1, True)
            oLCursor.setString("")
            oLCursor.goRight(1, False)
            icurrentState = StartLineState
        Case 13
            Rem En premier, sauvegarder le CR, ensuite, sélectionner la TAB et la supprimer
            oLCursor.goLeft(0, False)
            oLCursor.goLeft(1, False)
            oLCursor.goLeft(1, True)
            oLCursor.setString("")

            Rem Enfin, revenir sur le CR que nous ignorons
            oLCursor.goRight(1, True)
            icurrentState = StartLineState
        Case Else
            icurrentState = InWordState
            oLCursor.gotoEndOfWord(False)
        End Select
    End If

```

```

oLCursor.goRight(0, False)
Loop
If bChanged Then
    Rem Pour arriver jusqu'ici, nous sommes allés un caractère trop loin
    oLCursor.goLeft(1, False)
    oLCursor.goLeft(1, True)
    If Asc(oLCursor.getString()) = 9 Then oLCursor.setString("")
End If
End Sub
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub TabsToSpacesMain
    Dim oCursors(), i%, sPrompt$

sPrompt$ = "Convertir les TABS en Espaces pour le document ENTIER ?"
If Not CreateSelectedTextIterator(ThisComponent, sPrompt$, oCursors()) Then Exit Sub
For i% = LBOUND(oCursors()) To UBOUND(oCursors())
    TabsToSpacesWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
Next i%
End Sub
Sub TabsToSpacesWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Do While oLCursor.goRight(1, True) AND oText.compareRegionEnds(oLCursor, oRCursor) >= 0
        If Asc(oLCursor.getString()) = 9 Then
            oLCursor.setString("    ") 'Change une tab en 4 espaces
        End If
        oLCursor.goRight(0, False)
    Loop
End Sub
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org

'sPrompt: comment demander si doit répéter sur le texte entier
'oCursors (): A les curseurs retournés
'Retourne vrai si doit répéter et faux si ne doit pas
Function CreateSelectedTextIterator(oDoc As Object, sPrompt As String, oCursors() As Boolean)
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim ISelCount As Long, IWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    CreateSelectedTextIterator = True
    oText = oDoc.Text
    If Not IsAnythingSelected(ThisComponent) Then
        Dim i%

        i% = MsgBox("Aucun texte sélectionné !"+ Chr(13) + sPrompt, _
                    1 OR 32 OR 256, "Attention")
        If i% = 1 Then
            oLCursor = oText.createTextCursor()
            oLCursor.gotoStart(False)
            oRCursor = oText.createTextCursor()
            oRCursor.gotoEnd(False)
            oCursors = DimArray(0, 1)
            oCursors(0, 0) = oLCursor
            oCursors(0, 1) = oRCursor
        Else
            oCursors = DimArray()
            CreateSelectedTextIterator = False
        End If
    Else
        oSelections = ThisComponent.getCurrentSelection()
        ISelCount = oSelections.getCount()
    End If
End Function

```

```

oCursors = DimArray(ISelectionCount - 1, 1)
For IWhichSelection = 0 To ISelectionCount - 1
    oSel = oSelections.getByIndex(IWhichSelection)

    'Si je veux savoir si AUCUN texte n'est sélectionné, je pourrais
    'faire la chose suivante :
    'oLCursor = oText.CreateTextCursorByRange(oSel)
    'If oLCursor.isCollapsed() Then ...
    'oLCursor = GetLeftMostCursor(oSel, oText)
    'oRCursor = GetRightMostCursor(oSel, oText)
    'oCursors(IWhichSelection, 0) = oLCursor
    'oCursors(IWhichSelection, 1) = oRCursor

    Next
End If
End Function
*****  

'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org

'oDoc est un objet Writer
Function IsAnythingSelected(oDoc As Object) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object, oCursor As Object
    IsAnythingSelected = False
    If IsNull(oDoc) Then Exit Function

    'La sélection courante dans le contrôleur courant.
    S'il n'y a pas de contrôleur courant, retourne NULL.
    oSelections = oDoc.GetCurrentSelection()
    If IsNull(oSelections) Then Exit Function
    If oSelections.getCount() = 0 Then Exit Function
    If oSelections.getCount() > 1 Then
        IsAnythingSelected = True
    Else
        oSel = oSelections.getByIndex(0)
        oCursor = oDoc.Text.CreateTextCursorByRange(oSel)
        If Not oCursor.IsCollapsed() Then IsAnythingSelected = True
    End If
End Function
*****  

'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function GetLeftMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object

    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getEnd()
    Else
        oRange = oSel.getStart()
    End If
    oCursor = oText.CreateTextCursorByRange(oRange)
    oCursor.goRight(0, False)
    GetLeftMostCursor = oCursor
End Function
*****  

'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function GetRightMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object
    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getStart()
    Else
        oRange = oSel.getEnd()
    End If
End Function

```

```

End If
oCursor = oText.CreateTextCursorByRange(oRange)
oCursor.goLeft(0, False)
GetRightMostCursor = oCursor
End Function
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function IsWhiteSpace(iChar As Integer) As Boolean
    Select Case iChar
        Case 9, 10, 13, 32, 160
            IsWhiteSpace = True
        Case Else
            IsWhiteSpace = False
    End Select
End Function

*****

'Ici commencent les ANCIENNES macros !
'Auteur : Andrew Pitonyak
Sub ConvertSelectedNewParagraphTo.NewLine
    Dim ISelCount&, oSelections As Object, oSelection As Object
    Dim iWhichSelection As Integer, IIndex As Long
    Dim oText As Object, oCursor As Object
    Dim s$, ILastCR As Long, ILastNL As Long

    'Il peut y avoir de nombreuses sélections présentes
    oSelections = ThisComponent.getCurrentSelection()
    ISelCount = oSelections.getCount()
    oText=ThisComponent.Text

    For iWhichSelection = 0 To ISelCount - 1
        oSelection = oSelections.getByIndex(iWhichSelection)
        oCursor=oText.createTextCursorByRange(oSelection)
        s = oSelection.getString()
        oCursor.setString("")
        ILastCR = -1
        ILastNL = -1
        IIndex = 1
        Do While IIndex <= Len(s)
            Select Case Asc(Mid(s, IIndex, 1))
                Case 13
                    oText.insertControlCharacter(oCursor,
                        com.sun.star.text.ControlCharacter.LINE_BREAK, False)
                    'j'aurais aimé avoir un booléen 'short'
                    'Sauter le prochain LF s'il y en a un. Je pense qu'il
                    'y en aura toujours mais je ne peux pas le vérifier.
                    If (IIndex < Len(s)) Then
                        If Asc(Mid(s, IIndex+1, 1)) = 10 Then IIndex = IIndex + 1
                    End If
                Case 10
                    oText.insertControlCharacter(oCursor,
                        com.sun.star.text.ControlCharacter.LINE_BREAK, False)
                Case Else
                    oCursor.setString(Mid(s, IIndex, 1))
                    oCursor.GoRight(1, False)
            End Select
            IIndex = IIndex + 1
        Loop
    Next
End Sub

```

```

'J'ai décidé d'écrire ceci comme une machine d'états finis.
'Les machines d'états finis sont de merveilleuses choses :)
Sub ConvertSelectedSpaceToTabsBetweenWords
    Dim lSelCount&, oSelections As Object, oSelection As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim oText As Object, oCursor As Object
    Dim s$, lLastCR As Long, lLastNL As Long

    Rem Quels états sont supportés
    Dim iCurrentState As Integer
    Const StartLineState = 0
    Const InWordState = 1
    Const BetweenWordState = 2

    Rem Points de transition
    Dim iWhatFound As Integer
    Const FoundWhiteSpace = 0
    Const FoundNewLine = 1
    Const FoundOther = 2
    Const ActionIgnoreChr = 0
    Const ActionDeleteChr = 1
    Const ActionInsertTab = 2

    Rem Définir les états de transition
    Dim iNextState(0 To 2, 0 To 2, 0 To 1) As Integer
    iNextState(StartLineState, FoundWhiteSpace, 0)      = StartLineState
    iNextState(StartLineState, FoundNewLine, 0)          = StartLineState
    iNextState(StartLineState, FoundOther, 0)           = InWordState

    iNextState(InWordState, FoundWhiteSpace, 0)         = BetweenWordState
    iNextState(InWordState, FoundNewLine, 0)            = StartLineState
    iNextState(InWordState, FoundOther, 0)              = InWordState

    iNextState(BetweenWordState, FoundWhiteSpace, 0)= BetweenWordState
    iNextState(BetweenWordState, FoundNewLine, 0)     = StartLineState
    iNextState(BetweenWordState, FoundOther, 0)       = InWordState

    Rem Définir les états d'action
    iNextState(StartLineState, FoundWhiteSpace, 1)      = ActionDeleteChr
    iNextState(StartLineState, FoundNewLine, 1)          = ActionIgnoreChr
    iNextState(StartLineState, FoundOther, 1)           = ActionIgnoreChr

    iNextState(InWordState, FoundWhiteSpace, 1)         = ActionDeleteChr
    iNextState(InWordState, FoundNewLine, 1)            = ActionIgnoreChr
    iNextState(InWordState, FoundOther, 1)              = ActionIgnoreChr

    iNextState(BetweenWordState, FoundWhiteSpace, 1)= ActionDeleteChr
    iNextState(BetweenWordState, FoundNewLine, 1)     = ActionIgnoreChr
    iNextState(BetweenWordState, FoundOther, 1)       = ActionInsertTab

    'Il peut y avoir des sélections multiples présentes !
    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    oText=ThisComponent.Text

    For iWhichSelection = 0 To lSelCount - 1
        oSelection = oSelections.getByIndex(iWhichSelection)
        oCursor=oText.createTextCursorByRange(oSelection)
        s = oSelection.getString()
        oCursor.setString("")
        lLastCR = -1
        lLastNL = -1
        lIndex = 1
        iCurrentState = StartLineState
        Do While lIndex <= Len(s)
            Select Case Asc(Mid(s, lIndex, 1))

```

```

Case 9, 32, 160
    iWhatFound = FoundWhiteSpace
Case 10
    iWhatFound = FoundNewLine
    lLastNL = lIndex
Case 13
    iWhatFound = FoundNewLine
    lLastCR = lIndex
Case Else
    iWhatFound = FoundOther
End Select
Select Case iNextState(icurrentState, iWhatFound, 1)
Case ActionDeleteChr
    'En choisissant de ne pas insérer, il est effacé !
Case ActionIgnoreChr
    'Cela veut dire vraiment que je dois ajouter le caractère en arrière !
    If lLastCR = lIndex Then
        'Insérer un caractère du contrôle paraît déplacer le
        'curseur autour
        oText.insertControlCharacter(oCursor,
            com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
        oText.insertControlCharacter(oCursor,
            com.sun.star.text.ControlCharacter.APPEND_PARAGRAPH, False)
        'oCursor.goRight(1, False)
        'Print "CR inséré"
    ElseIf lLastNL = lIndex Then
        If lLastCR + 1 <> lIndex Then
            oText.insertControlCharacter(oCursor,
                com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
            'com.sun.star.text.ControlCharacter.LINE_BREAK, False)
            'oCursor.goRight(1, False)
            'Print "NL insérée"
        End If
        'Ignorer celui-ci
    Else
        oCursor.setString(Mid(s, lIndex, 1))
        oCursor.GoRight(1, False)
        'Print "Quelque chose insérée"
    End If
Case ActionInsertTab
    oCursor.setString(Chr$(9) + Mid(s, lIndex, 1))
    oCursor.GoRight(2, False)
    'Print "Tabulation insérée"
End Select
lIndex = lIndex + 1
'MsgBox "index = " + lIndex + Chr(13) + s
icurrentState = iNextState(icurrentState, iWhatFound, 0)
Loop
Next
End Sub

```

```

Sub ConvertAllTabsToSpace
    DIM oCursor As Object, oText As Object
    Dim nSpace%, nTab%, nPar%, nRet%, nTot%
    Dim justStarting As Boolean

    oText=ThisComponent.Text          'Récupérer le composant texte
    oCursor=oText.createTextCursor()  'Créer un curseur dans le texte
    oCursor.gotoStart(FALSE)         'Aller au début mais NE PAS sélectionner le texte en allant
    Do While oCursor.GoRight(1, True) 'Bouger à droite d'un caractère et le sélectionner
        If Asc(oCursor.getString()) = 9 Then
            oCursor.setString("    ") 'Change une tabulation en 4 espaces
        End If
        oCursor.goRight(0,FALSE)      'Désélectionne le texte !
    Loop

```

```

End Sub

Sub ConvertSelectedTabsToSpaces
    Dim lSelCount&, oSelections As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim s$, bSomethingChanged As Boolean

    'Il peut y avoir des sélections multiples présentes !
    'Il y en aura probablement une de plus qu'attendu parce que
    'il comptera l'emplacement du curseur courant comme un morceau
    'de texte sélectionné, soyez avertis !
    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    'Print "total selectionné = " + lSelCount
    For iWhichSelection = 0 To lSelCount - 1
        bSomethingChanged = False
        s = oSelections.getByIndex(iWhichSelection).getString()
        'Print "Le groupe de texte " + iWhichSelection + " est de taille " + Len(s)
        lIndex = 1
        Do While lIndex < Len(s)
            'Print "ascii à " + lIndex + " = " + Asc(Mid(s, lIndex, 1))
            If Asc(Mid(s, lIndex, 1)) = 9 Then
                s = ReplaceInString(s, lIndex, 1, "    ")
                bSomethingChanged = True
                lIndex = lIndex + 3
            End If
            lIndex = lIndex + 1
            'Print ":" + lIndex + "(" + s + ")"
        Loop
        If bSomethingChanged Then
            oSelections.getByIndex(iWhichSelection).setString(s)
        End If
    Next
End Sub

Function ReplaceInString(s$, index&, num&, replaces$) As String
    If index <= 1 Then
        '
        If num < 1 Then
            ReplaceInString = replaces + s
        ElseIf num > Len(s) Then
            ReplaceInString = replaces
        Else
            ReplaceInString = replaces + Right(s, Len(s) - num)
        End If
    ElseIf index + num > Len(s) Then
        ReplaceInString = Left(s, index - 1) + replaces
    Else
        ReplaceInString = Left(s, index - 1) + replaces + Right(s, Len(s) - index - num + 1)
    End If
End Function

```

Définir les attributs de texte

Quand cette macro est lancée, elle affecte le paragraphe contenant le curseur. La police et la taille sont définies. L'attribut CharPosture contrôle l'italique, CharWeight contrôle le gras, et CharUnderline contrôle le type de soulignement. Les valeurs valides se trouvent à :

<http://api.openoffice.org/docs/common/ref/com/sun/star/style/CharacterProperties.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/FontWeight.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/FontSlant.html>

|

```
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub SetTextAttributes
    Dim document As Object
    Dim Cursor
    Dim oText As Object
    Dim mySelection As Object
    Dim Font As String

    document=ThisComponent
    oText = document.Text
    Cursor = document.currentcontroller.getViewCursor()
    mySelection = oText.createTextCursorByRange(Cursor.getStart())
    mySelection.gotoStartOfParagraph(false)
    mySelection.gotoEndOfParagraph(true)
    mySelection.CharFontName="Courier New"
    mySelection.Charheight="10"
    'Il est temps de définir l'italique ou pas, selon le cas avec
    'NONE, OBLIQUE, ITALIC, DONTKNOW, REVERSE_OBLIQUE, REVERSE_ITALIC
    mySelection.CharPosture = com.sun.star.awt.FontSlantITALIC
    'Alors, vous voulez un texte en gras ?
    'DONTKNOW, THIN, ULTRALIGHT, LIGHT, SEMILIGHT,
    'NORMAL, SEMIBOLD, BOLD, ULTRABOLD, BLACK
    'Ces dernières sont vraiment des constantes avec THIN à 50, NORMAL à 100
    ' BOLD à 150, et BLACK à 200.
    mySelection.CharWeight = com.sun.star.awt.FontWeight.BOLD
    'Si le soulignement est votre tasse de thé
    'NONE, SINGLE, DOUBLE, DOTTED, DONTKNOW, DASH, LONGDASH,
    'DASHDOT, DASHDOTDOT, SMALLWAVE, WAVE, DOUBLEWAVE, BOLD,
    'BOLDDOTTED, BOLDDASH, BOLDLONGDASH, BOLDDASHDOT,
    'BOLDDASHDOTDOT, BOLDWAVE
    mySelection.CharUnderline = com.sun.star.awt.FontUnderline.SINGLE
    'Je n'ai pas assez expérimenté ce qui suit pour en connaître les réelles
    'implications mais je sais que cela semble définir
    'la localisation des caractères à Allemand.
    Dim aLanguage As New com.sun.star.lang.Locale
    aLanguage.Country = "de"
    aLanguage.Language = "de"
    mySelection.CharLocale = aLanguage
End Sub
```

Insérer du texte

```
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub InsertSimpleText
    Dim oDocument As Object
    Dim oText As Object
    Dim oViewCursor As Object
    Dim oTextCursor As Object

    oDocument = ThisComponent
    oText = oDocument.Text
    oViewCursor = oDocument.CurrentController.getViewCursor()
    oTextCursor = oText.createTextCursorByRange(oViewCursor.getStart())
    ' Place le texte à insérer ici
    oText.insertString(oTextCursor, "—", FALSE)
End Sub
```

Les champs

Ceci va insérer le texte "Aujourd'hui est le <date>" où la date est formatée selon "DD. MMM YYYY". Cela créera le format de date s'il n'existe pas. Pour plus d'informations sur les formats valides, consultez l'aide aux rubriques « formats numériques ; formats ».

```
*****  
'Auteur : Andrew Pitonyak  
'email : andrew@pitonyak.org  
'utilise :FindCreateNumberFormatStyle  
Sub InsertDateField  
    Dim oDocument As Object  
    Dim oText As Object  
    Dim oViewCursor As Object  
    Dim oTextCursor As Object  
    Dim oDateTime As Object  
  
    oDocument = ThisComponent  
    If oDocument.SupportsService("com.sun.star.text.TextDocument") Then  
        oText = oDocument.Text  
        oViewCursor = oDocument.CurrentController.getViewCursor()  
        oTextCursor = oText.createTextCursorByRange(oViewCursor.getStart())  
        oText.insertString(oTextCursor, "Aujourd'hui est le ", FALSE)  
        ' Crée le type DateTime.  
        oDateTime = oDocument.createInstance("com.sun.star.text.TextField.DateTime")  
        oDateTime.IsFixed = TRUE  
        oDateTime.NumberFormat = FindCreateNumberFormatStyle(_  
            "DD. MMMM YYYY", oDocument)  
        oText.insertTextContent(oTextCursor,oDateTime,FALSE)  
        oText.insertString(oTextCursor," ",FALSE) Else  
        MsgBox "Désolé, cette macro nécessite un document texte"  
    End If  
End Sub
```

```
Sub AddNoteAtCursor  
    Dim vDoc  
    Dim vViewCursor  
    Dim vCursor  
    Dim vTextField  
  
    ' Allez, on va prétendre l'avoir ajoutée il y a 10 jours !  
    'http://api.openoffice.org/docs/common/ref/com/sun/star/util/Date.html  
    Dim aDate As New com.sun.star.util.Date  
    With aDate  
        .Day = Day(Now - 10)  
        .Month = Month(Now - 10)  
        .Year = Year(Now - 10)  
    End With  
  
    vDoc = ThisComponent  
    vViewCursor = vDoc.getCurrentController().getViewCursor()  
    vCursor = vDoc.getText().createTextCursorByRange(vViewCursor.getStart())  
    ' *** ?! Positionne le curseur invisible à la position courante.  
  
    'http://api.openoffice.org/docs/common/ref/com/sun/star/text/textfield/Annotation.html  
    vTextField = vDoc.createInstance("com.sun.star.text.TextField.Annotation")  
    With vTextField  
        .Author = "AP"  
        .Content = "Qu'est-ce que c'est marrant d'insérer des notes dans mon document"  
    End With
```

```

' Si vous ne la mentionnez pas, la date est celle d'aujourd'hui !
.Date = aDate
End With
vDoc.Text.insertTextContent(vCursor, vTextField, False)
End Sub

```

Insérer une nouvelle page

Dans ma recherche pour insérer une nouvelle page dans un document, je suis tombé sur le lien suivant :

<http://api.openoffice.org/docs/common/ref/com/sun/star/style/ParagraphProperties.html>

qui discute de deux propriétés. Le *PageNumberOffset* stipule : "Si une propriété de saut de page est définie sur un paragraphe, cette propriété contient la nouvelle valeur pour le numéro de page". La propriété *PageDescName* stipule : "Si cette propriété est définie, elle crée un saut de page avant le paragraphe auquel il appartient et assigne la valeur comme étant le nom du style de la nouvelle page à utiliser". J'ai réfléchi que si je définissais *PageDescName*, alors je pourrais créer une nouvelle page et définir le numéro de page. Ce qui n'était pas dit est que *PageDescName* est le nom du style pour la nouvelle page à utiliser après le saut de page. Si vous n'utilisez pas un style de page existant, alors, cela ne fonctionnera pas !

```

Sub ExampleNewPage
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    oText = ThisComponent.Text
    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    For lWhichSelection = 0 To lSelCount - 1
        oSel = oSelections.getByIndex(lWhichSelection)
        oLCursor = oText.CreateTextCursorByRange(oSel)
        oLCursor.gotoStartOfParagraph(false)
        oLCursor.gotoEndOfParagraph(true)

        Rem Conserve le style de page existant
        oLCursor.PageDescName = oLCursor.PageStyleName
        oLCursor.PageNumberOffset = 7
    Next
End Sub

```

Gérer le style de page du document

Le style de page est paramétré en modifiant le nom de description de page. C'est très similaire au fait de démarrer une nouvelle page.

```

Sub SetDocumentPageStyle
    Dim oCursor As Object
    oCursor = ThisComponent.Text.createTextCursor()
    oCursor.gotoStart(False)
    oCursor.gotoEnd(True)
    Print "Style courant = " & oCursor.PageStyleName
    oCursor.PageDescName = "Wow"
End Sub

```

Insérer un objet OLE

Le rumeur dit qu'avec OpenOffice 1.1, le code suivant insérera un objet OLE dans un document Texte. Le CLSID peut être un objet externe.

```
obj = ThisComponent.createInstance("com.sun.star.text.TextEmbeddedObject")
obj.CLSID = "47BBB4CB-CE4C-4E80-A591-42D9AE74950F"
obj.attach( ThisComponent.currentController().Selection.getByIndex(0) )
```

Si vous avez sélectionné un objet embarqué dans Writer, vous pouvez accéder à son API avec :

```
oModel = ThisComponent.currentController().Selection.Model
```

Ceci fournit la même interface à l'objet que si vous l'aviez créé en chargeant un document avec `loadComponentFromURL`

Paramétriser le style de paragraphe

Différents styles peuvent être directement réglés en sélectionnant le texte concerné.

```
Option Explicit
Sub SetParagraphStyle
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim ISelCount As Long, IWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    oText = ThisComponent.Text
    oSelections = ThisComponent.getCurrentSelection()
    ISelCount = oSelections.getCount()
    For IWhichSelection = 0 To ISelCount - 1
        oSel = oSelections.getByIndex(IWhichSelection)
        oSel.ParaStyleName = "Heading 2"
    Next
End Sub
```

L'exemple suivant mettra tous les paragraphes au même style :

```
'Auteur : Marc Messeant
'email : marc.liste@free.fr
Sub AppliquerStyle()
    Dim oDocument As Object, oText As Object, oViewCursor As Object, oTextCursor As Object
    oDocument = ThisComponent
    oText = oDocument.Text

    oViewCursor = oDocument.CurrentController.getViewCursor()
    oTextCursor = oText.createTextCursorByRange(oViewCursor.getStart())

    While oText.compareRegionStarts(oTextCursor.getStart(), oViewCursor.getEnd()) = 1
        oTextCursor.paraStyleName = "YourStyle"
        oTextCursor.gotoNextParagraph(false)
    Wend
End Sub
```

Créer votre propre style

Je n'ai pas testé ce code par manque de temps, mais je pense qu'il marche :

```
vFamilies = ThisComponent.StyleFamilies
vStyle = ThisComponent.createInstance("com.sun.star.style.ParagraphStyle")
vParaStyles = vFamilies.getByName("ParagraphStyles")
vParaStyles.insertByName("MyStyle", vStyle)
```

Rechercher et remplacer

Un document susceptible de recherches supporte la capacité à créer un descripteur de recherche. Il sera également possible de trouver le premier, le suivant, et toutes les occurrences du texte recherché. Voir :

<http://api.openoffice.org/docs/common/ref/com/sun/star/util/XSearchable.html>

Chercher est assez simple et un exemple devrait fournir suffisamment

d'explications :

```
Sub SimpleSearchExample
    Dim vDescriptor, vFound
    ' Création d'un descripteur depuis un document susceptible de recherches
    vDescriptor = ThisComponent.createSearchDescriptor()
    ' Indiquer le texte à chercher et autre
    ' http://api.openoffice.org/docs/common/ref/com/sun/star/util/SearchDescriptor.html
    With vDescriptor
        .SearchString = "hello"
        ' Tout ceci est "false" par défaut
        .SearchWords = true
        .SearchCaseSensitive = False
    End With
    ' Chercher le premier
    vFound = ThisComponent.findFirst(vDescriptor)
    Do While Not IsNull(vFound)
        Print vFound.getString()
        vFound.CharWeight = com.sun.star.awt.FontWeight.BOLD
        vFound = ThisComponent.findNext( vFound.End, vDescriptor)
    Loop
```

L'objet retourné par `findFirst` et `findNext` se comporte comme un curseur et la plupart des choses qu'il est possible d'effectuer avec un curseur, comme le paramétrage des attributs, sont également faisables avec cet objet.

Remplacer du texte est semblable à la recherche à ceci près que cela doit supporter :

<http://api.openoffice.org/docs/common/ref/com/sun/star/util/XReplaceable.html>
La seule méthode utile fournie par ceci : un document susceptible de recherche n'est pas capable de remplacer toutes les occurrences du texte recherché par autre chose. L'idée était que vous en cherchiez une à la fois, et vous pouviez modifier chaque occurrence manuellement. L'exemple suivant recherche le texte et remplace des choses telles que l'"a@" avec le caractère unicode 257.

```
'Auteur : Birgit Kellner
'email : birgit.kellner@univie.ac.at
Sub AtToUnicode
    'Andy dit que dans le futur, ils devront peut-être être de type Variant pour travailler avec Array
()
    Dim numbered(5) As String, accented(5) As String
    Dim n as long
    Dim oDocument as object, oReplace as object
    numbered() = Array("A@", "a@", "I@", "i@", "U@", "u@", "Z@", "z@", "O@", "o@", "H@",
                      "h@", "D@", "d@", "L@", "l@", "M@", "m@", "G@", "g@", "N@", "n@", "R@", "r@",
                      "Y@", "y@", "S@", "s@", "T@", "t@", "C@", "c@", "j@", "J@")
    accented() = Array(Chr$(256), Chr$(257), Chr(298), Chr$(299), Chr$(362), Chr$(363),
                      Chr$(377), Chr$(378), Chr$(332), Chr$(333), Chr$(7716), Chr$(7717), Chr$(7692),
                      Chr$(7693), Chr$(7734), Chr$(7735), Chr$(7746), Chr$(7747), Chr$(7748), Chr$(7749),
                      Chr$(7750), Chr$(7751), Chr$(7770), Chr$(7771), Chr$(7772), Chr$(7773), Chr$(7778),
                      Chr$(7779), Chr$(7788), Chr$(7789), Chr$(346), Chr$(347), Chr$(241), Chr$(209))
    oReplace = ThisComponent.createReplaceDescriptor()
    oReplace.SearchCaseSensitive = True
    For n = lbound(numbered()) To ubound(accented())
        oReplace.SearchString = numbered(n)
        oReplace.ReplaceString = accented(n)
        ThisComponent.ReplaceAll(oReplace)
    Next n
End Sub
```

Le « truc » pour chercher seulement une plage de texte sélectionnée est basée sur le fait que le curseur peut être utilisé dans la routine findNext. Vous pouvez alors chercher les points finaux de la sélection pour éviter que la recherche n'aille trop loin. Ceci vous autorise à démarrer la recherche à n'importe quelle position du curseur. La méthode findFirst n'est pas nécessaire si vous avez un objet de type curseur à qui spécifier la position du début de la recherche avec findNext. L'exemple ci dessous utilise mon système basé sur le texte sélectionné et contient quelques améliorations suggérées par Bernard Marcellly.

Voyez également :

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRangeCompare.html>

```
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub SearchSelectedText
    Dim oCursors(), i%
    If Not CreateSelectedTextIterator(ThisComponent,
        "Search text in the entire document?", oCursors) Then Exit Sub
    For i% = LBound(oCursors) To UBound(oCursors)
        SearchSelectedWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent)
    Next i%
End Sub

*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub SearchSelectedWorker(oLCursor, oRCursor, oDoc)
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oDoc) Then Exit Sub
    If oDoc.Text.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Dim vDescriptor, vFound
    vDescriptor = oDoc.createSearchDescriptor()
    With vDescriptor
        .SearchString = "Paragraphe"
        .SearchCaseSensitive = False
    End With
    ' Il n'y a pas lieu d'exécuter findFirst.
    vFound = oDoc .findNext(oLCursor, vDescriptor)
    Rem Voudriez vous arrêter l'évaluation ?
    Do While Not IsNull(vFound)
        Rem Si vFound.hasElements() est vide alors on sort
        'Voyons si nous recherchons après la fin
        If -1 = oDoc.Text.compareRegionEnds(vFound, oRCursor) Then Exit Do
        Print vFound.getString()
        vFound = ThisComponent.findNext( vFound.End, vDescriptor)
    Loop
End Sub
```

```
'effacer du texte entre délimiteurs est actuellement aisé
Sub deleteTextBetweenDlimitters
    Dim vOpenSearch, vCloseSearch      'Ouvrir et fermer les délimiteurs
    Dim vOpenFound, vCloseFound       'Ouvrir et fermer les objets trouvés

    ' Création d'un descripteur depuis le document susceptible de recherches.
    vOpenSearch = ThisComponent.createSearchDescriptor()
    vCloseSearch = ThisComponent.createSearchDescriptor()
```

```

' Indiquer le texte à chercher et autre
' http://api.openoffice.org/docs/common/ref/com/sun/star/util/SearchDescriptor.html
vOpenSearch.SearchString = "["
vCloseSearch.SearchString = "]"

' Trouver et ouvrir le premier délimiteur
vOpenFound = ThisComponent.findFirst(vOpenSearch)
Do While Not IsNull(vOpenFound)

    'Rechercher le délimiteur fermant le plus proche du début
    vCloseFound = ThisComponent.findNext( vOpenFound.End, vCloseSearch)
    If IsNull(vCloseFound) Then
        Print "Trové une parenthèse ouvrante mais aucune parenthèse fermante !"
        Exit Do
    Else
        ' Dégager la parenthèse ouvrante sous peine de finir en haut
        ' Seulement avec le texte entre parenthèses
        vOpenFound.setString("")
        ' Selection du texte entre parenthèses
        vOpenFound.gotoRange(vCloseFound, True)
        Print "Trové " & vOpenFound.getString()
        ' Dégager le texte concerné
        vOpenFound.setString("")
        ' Dégager la parenthèse fermante
        vCloseFound.setString("")
        ' Voulez vous vraiment tout effacer dans cet espace
        ' Alors, allons y !
        If vCloseFound.goRight(1, True) Then
            If vCloseFound.getString() = " " Then vCloseFound.setString("")
        End If
        vOpenFound = ThisComponent.findNext( vOpenFound.End, vOpenSearch)
    End If
Loop
End Sub

```

Cette macro encadre tous les éléments en avec des accolades “{{ }}” et change l'attribut du en Normal. Une expression régulière est utilisée pour spécifier le texte dans lequel chercher.

```

Sub ReplaceFormatting
    'code original : Alex Savitsky
    'modifié par : Laurent Godard
    ' Le but de cette macro est d'encadrer tous les éléments en GRAS par {{ }}
    ' et de changer l'attribut Gras en NORMAL
    ' Ceci se fait avec des expressions régulières
    ' Les styles doivent aussi être pris en compte

    Dim oDocument As Object
    Dim oReplace As Object
    Dim SrchAttributes(0) as new com.sun.star.beans.PropertyValue
    Dim ReplAttributes(0) as new com.sun.star.beans.PropertyValue

    oDocument = ThisComponent
    oReplace = oDocument.createReplaceDescriptor

    oReplace.SearchString = "*"      'Expression régulière. Prendre tous les caractères
    oReplace.ReplaceString = "{{ & }}"
    oReplace.SearchRegularExpression=True 'Utiliser les expressions régulières
    oReplace.searchStyles=True      ' Nous voulons rechercher dans les styles
    oReplace.searchAll=True        ' Pour tout le document

```

```

REM Voilà l'attribut à trouver
SrchAttributes(0).Name = "CharWeight"
SrchAttributes(0).Value =com.sun.star.awt.FontWeight.BOLD

REM Voilà l'attribut par lequel remplacer le premier
ReplAttributes(0).Name = "CharWeight"
ReplAttributes(0).Value =com.sun.star.awt.FontWeight.NORMAL

REM Place les attributs dans le descripteur de remplacement
oReplace.SetSearchAttributes(SrchAttributes())
oReplace.SetReplaceAttributes(ReplAttributes())

REM Allez au boulot !
oDocument.replaceAll(oReplace)
End Sub

```

Changer la casse des mots

OOo détermine la casse d'un mot en se basant sur les propriétés de caractère. En théorie, cela signifie que l'on peut sélectionner le document entier et paramétrier la casse. Dans la pratique toutefois, les portions sélectionnées peuvent ne pas supporter la propriété casse de caractère. Comme compromis entre vitesse et problèmes possibles, j'ai choisi d'utiliser un curseur de mot pour traverser le texte paramétrant la propriété casse de chaque mot individuellement. J'ai écrit cette macro pour travailler sur des mots entiers, c'est une décision arbitraire. Si elle ne vous convient pas, changez-la. J'ai utilisé mon système de texte sélectionné, vous aurez donc besoin de ces macros pour ces travaux.

S'il se trouve que vous avez du texte qui ne supporte pas le paramétrage de la casse de caractère, vous pouvez éviter l'apparition d'erreur en ajoutant "On Local Error Resume Next" à SetWordCase().

Paramétrier la casse ne change pas le caractère mais seulement son affichage. Si vous paramétrez un bas de casse, vous ne pourrez pas passer en haut de casse manuellement.

Dans OOo 1.0.3.1, la casse des titres est perturbée : "heLLo" devient "HeLLo".

<http://api.openoffice.org/docs/common/ref/com/sun/star/style/CaseMap.html>

```

'*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Sub ADPSetWordCase()
    Dim vCursors(), i%, sMapType$, iMapType%
    iMapType = -1
    Do While iMapType < 0
        sMapType = InputBox("Quelle casse vais-je utiliser ?" & chr(10) &
            "Rien, MAJUSCULE, minuscule, Titre ou petites majuscules ?", "Changer le Type de
Casse", "Titre")
        sMapType = UCase(Trim(sMapType))
        If sMapType = "" Then sMapType = "EXIT"
        Select Case sMapType
        Case "EXIT"
            Exit Sub
        Case "NONE"
            iMapType = com.sun.star.style.CaseMap.NONE
        Case "UPPER"

```

```

    iMapType = com.sun.star.style.CaseMap.UPPERCASE
Case "LOWER"
    iMapType = com.sun.star.style.CaseMap.LOWERCASE
Case "TITLE"
    iMapType = com.sun.star.style.CaseMap.TITLE
Case "SMALL CAPS"
    iMapType = com.sun.star.style.CaseMap.SMALLCAPS
Case Else
    Print "Désolé, " & sMapType & " n'est pas un type reconnu valide "
End Select
Loop
If Not CreateSelectedTextIterator(ThisComponent,
    "Changer le document entier ?", vCursors()) Then Exit Sub
For i% = LBound(vCursors()) To UBound(vCursors())
    SetWordCase(vCursors(i%, 0), vCursors(i%, 1), ThisComponent.Text, iMapType%)
Next
End Sub
*****
'Auteur : Andrew Pitonyak
'email : andrew@pitonyak.org
Function SetWordCase(vLCursor, vRCursor, vText, iMapType%)
If IsNull(vLCursor) OR IsNull(vRCursor) OR IsNull(vText) Then Exit Function
If vText.compareRegionEnds(vLCursor, vRCursor) <= 0 Then Exit Function
vLCursor.goRight(0, False)
Do While vLCursor.gotoNextWord(True)
    If vText.compareRegionStarts(vLCursor, vRCursor) > 0 Then
        vLCursor.charCasemap = iMapType%
        vLCursor.goRight(0, False)
    Else
        Exit Function
    End If
Loop
Rem Si le dernier mot du document n'est pas suivi de ponctuation et de nouvelles lignes
Rem alors il est impossible d'aller au prochain mot. Je dois maintenant m'attaquer à ce cas de
figure
If vText.compareRegionStarts(vLCursor, vRCursor) > 0 AND vLCursor.gotoEndOfWord(True) Then
    vLCursor.charCasemap = iMapType%
End If
End Function

```

Andrew apprend à parcourir les paragraphes

Je voulais parcourir les paragraphes et éventuellement changer le style de paragraphe. Pour cela, je devais sélectionner chaque paragraphe tour à tour. Mon premier essai, qui fut une erreur, consistait à démarrer par la sélection d'un paragraphe. Puis je déplaçais le curseur de zéro espace à droite, pour désélectionner le paragraphe courant.

vCurCursor.goRight(0, False)

Ça marche bien, mais le curseur reste dans le paragraphe courant. Dans OOo, un paragraphe est du genre "blah blah blah.<cr><lf>". Quand j'utilise la méthode ci-dessus, je laisse le curseur juste avant <cr><lf>, qui font partie du paragraphe. Et quand j'utilisais vCursor.gotoNextParagraph(True), cela sélectionnait <cr><lf> et plaçait le curseur au début du paragraphe suivant . Ce n'est CERTAINEMENT PAS ce que je voulais. La bonne manière de passer au paragraphe suivant et de le sélectionner est la suivante :

vCurCursor.gotoNextParagraph(False) 'aller au début du prochain paragraphe.
vCurCursor.gotoEndOfParagraph(True) 'sélectionner le paragraphe.

Pour parcourir les paragraphes et imprimer leur style, j'utilise la macro ci-dessous. Attention, si j'ai sélectionné plus d'un paragraphe, je ne peux pas obtenir le style de paragraphe, d'où l'importance de pouvoir sélectionner un

seul paragraphe à la fois.

```
Sub PrintAllStyles
    Dim s As String
    Dim vCurCursor As Variant
    Dim vText As Variant
    Dim sCurStyle As String

    vText = ThisComponent.Text
    vCurCursor = vText.CreateTextCursor()
    vCurCursor.GoToStart(False)
    Do
        If NOT vCurCursor.gotoEndOfParagraph(True) Then Exit Do
        sCurStyle = vCurCursor.ParaStyleName
        s = s & "****" & sCurStyle & "****" & Chr$(10)
    Loop Until NOT vCurCursor.gotoNextParagraph(False)
    MsgBox s, 0, "Styles dans le Document"
End Sub
```

La dernière application consiste à parcourir ce document que vous lisez, en affectant un style de paragraphe aux lignes de code. Dans le cas d'une macro d'une seule ligne de code, le style sera _code_une_ligne. S'il y a plus d'une ligne, alors la première ligne sera _code_prem_ligne, la dernière sera _code_dern_ligne, et ce qui est entre le deux sera juste _code. Je n'ai pas été très rigoureux sur les conditions de démarrage ou de fin de travail, aussi cela pourra ne pas marcher si j'ai la première ou la dernière ligne en _code (ce qui n'est pas le cas, donc je ne m'inquiète pas pour ça).

```
Sub user_CleanUpCodeSections
    worker_CleanUpCodeSections("_code_prem_ligne", "_code", "_code_dern_ligne",
    "_code_une_ligne")
End Sub
```

REM Je ne peux pas utiliser simplement la commande goRight(0, False) pour avancer le curseur, car je ne veux

```
REM pas capturer les caractères <NL>. Beurk !
Sub worker_CleanUpCodeSections(firstStyle$, midStyle$, lastStyle$, onlyStyle$)
    Dim vCurCursor As Variant      ' Curseur actuel
    Dim vPrevCursor As Variant     ' Curseur précédent (un paragraphe au-dessus)
    Dim sPrevStyle As String       ' Style précédent
    Dim sCurStyle As String        ' Style actuel
```

```
REM Place le curseur actuel au début du second paragraphe
vCurCursor = ThisComponent.Text.CreateTextCursor()
vCurCursor.GoToStart(False)
If NOT vCurCursor.gotoNextParagraph(False) Then Exit Sub
```

```
REM Place le curseur précédent pour sélectionner le premier paragraphe
vPrevCursor = ThisComponent.Text.CreateTextCursor()
vPrevCursor.GoToStart(False)
If NOT vPrevCursor.gotoEndOfParagraph(True) Then Exit Sub
sPrevStyle = vPrevCursor.ParaStyleName
```

```
Do
    REM Je ne peux pas utiliser simplement goRight(0,
    If NOT vCurCursor.gotoEndOfParagraph(True) Then Exit Do
    sCurStyle = vCurCursor.ParaStyleName
```

```
REM C'est là que commence le travail.
If sCurStyle = firstStyle$ Then
    REM Le style actuel est le premier style
    REM Je cherche si le style précédent était un de ces styles
    Select Case sPrevStyle
        Case onlyStyle$, lastStyle$
            sCurStyle = midStyle$
```

```

vCurCursor.ParaStyleName = sCurStyle
vPrevCursor.ParaStyleName = firstStyle$

Case firstStyle$, midStyle$
  sCurStyle = midStyle$
  vCurCursor.ParaStyleName = sCurStyle
End Select

ElseIf sCurStyle = midStyle$ Then
  REM Le style actuel est le style du milieu
  REM Idem : je cherche si le style précédent était un de ces styles
  Select Case sPrevStyle
    Case firstStyle$, midStyle$
      REM Ne rien faire !

    Case onlyStyle$
      REM Le dernier style était un style seul, mais il arrive avant un style de milieu !
      vPrevCursor.ParaStyleName = firstStyle$

    Case lastStyle$
      vPrevCursor.ParaStyleName = midStyle$

    Case Else
      sCurStyle = firstStyle$
      vCurCursor.ParaStyleName = sCurStyle
  End Select

ElseIf sCurStyle = lastStyle$ Then
  Select Case sPrevStyle
    Case firstStyle$, midStyle$
      REM Ne rien faire !

    Case onlyStyle$
      REM Le dernier style était un style seul, mais il arrive avant un style de milieu !
      vPrevCursor.ParaStyleName = firstStyle$

    Case lastStyle$
      vPrevCursor.ParaStyleName = midStyle$

    Case Else
      sCurStyle = firstStyle$
      vCurCursor.ParaStyleName = sCurStyle
  End Select

ElseIf sCurStyle = onlyStyle$ Then
  Select Case sPrevStyle
    Case firstStyle$, midStyle$
      sCurStyle = midStyle$
      vCurCursor.ParaStyleName = sCurStyle

    Case lastStyle$
      sCurStyle = lastStyle$
      vCurCursor.ParaStyleName = sCurStyle
      vPrevCursor.ParaStyleName = midStyle$

    Case onlyStyle$
      sCurStyle = lastStyle$
      vCurCursor.ParaStyleName = sCurStyle
      vPrevCursor.ParaStyleName = firstStyle$
  End Select

Else
  Select Case sPrevStyle
    Case firstStyle$
      vPrevCursor.ParaStyleName = onlyStyle$

```

```

Case midStyle$
    vPrevCursor.ParaStyleName = lastStyle$
End Select
End If

REM Le travail est fait, donc on avance le curseur
vPrevCursor.gotoNextParagraph(False)
vPrevCursor.gotoEndOfParagraph(True)
sPrevStyle = vPrevCursor.ParaStyleName
Loop Until NOT vCurCursor.gotoNextParagraph(False)

End Sub

```

Où est le Curseur affiché ?

Je n'ai pas le temps de détailler, mais voici le mail envoyé par Giuseppe Castagno [castagno@tecsa-srl.it], qui m'a proposé ces idées.

Ce que vous avez écrit est très intéressant, mais je ne suis pas sûr que ce soit correct. D'abord, la position du curseur (commande getposition) semble être relative au premier endroit en haut de la feuille qui puisse contenir du texte. S'il y a un en-tête, la position sera relative au début de cet en-tête, tandis que s'il n'y en a pas, la position sera relative au début de la zone de texte. Il semble aussi que la marge du haut soit comprise entre le haut de la page et le premier endroit qui peut contenir du texte.

Vos mesures de la position du pied de page sont bien imaginées, car cela vous donne l'espace entre le haut du pied de page et le curseur. J'ai trouvé ça génial, je n'y avais jamais pensé. Néanmoins, qu'arrive-t-il si on a augmenté la taille du pied de page ? Je crois que vous n'avez pas pris cela en compte.

Vous pouvez probablement mesurer plutôt comme ceci :

Hauteur de la page – marge du haut – position du curseur

Ainsi, pas besoin de déplacer le curseur.

```

Sub PrintCursorPosition
Dim xDoc
Dim xViewCursor
Dim s As String

xDoc = ThisComponent
xViewCursor = xDoc.CurrentController.getViewCursor()
s = xViewCursor.PageStyleName

Dim xFamilyNames As Variant, xStyleNames As Variant
Dim xFamilies
Dim xStyle, xStyles

xFamilies = xDoc.StyleFamilies
xStyles = xFamilies.getByName("PageStyles")
xStyle = xStyles.getByName(xViewCursor.PageStyleName)
' RunSimpleObjectBrowser(xViewCursor)

Dim IHeight As Long
Dim IWidth As Long
IHeight = xStyle.Height
IWidth = xStyle.Width

s = "La taille de la page est " & CHR$(10) &
    " " & CStr(IWidth / 100.0) & " mm par " &
    " " & CStr(IHeight / 100.0) & " mm" & CHR$(10) &
    " " & CStr(IWidth / 2540.0) & " pouces par " &
    " " & CStr(IHeight / 2540.0) & " pouces" & CHR$(10) &

```

```

"    " & CStr(lWidth *72.0 / 2540.0) & " picas par " &
"    " & CStr(lHeight *72.0 / 2540.0) & " picas" & CHR$(10)

Dim dCharHeight As Double
Dim iCurPage As Integer

Dim dXCursor As Double
Dim dYCursor As Double
Dim dXRight As Double
Dim dYBottom As Double
Dim dBottomMargin As Double
Dim dLeftMargin As Double

dCharHeight = xViewCursor.CharHeight / 72.0
iCurPage = xViewCursor.getPage()

Dim v
v = xViewCursor.getPosition()
dYCursor = (v.Y + xStyle.TopMargin)/2540.0 + dCharHeight / 2
dXCursor = (v.X + xStyle.LeftMargin)/2540.0
dXRight = (lWidth - v.X - xStyle.LeftMargin)/2540.0
dYBottom = (lHeight - v.Y - xStyle.TopMargin)/2540.0 - dCharHeight / 2
' Print "Marge de gauche = " & xStyle.LeftMargin/2540.0

dBottomMargin = xStyle.BottomMargin / 2540.0
dLeftMargin = xStyle.LeftMargin / 2540.0
s = s & "Le curseur est à " & Format(dXCursor, "0.##") & " pouces du bord gauche " & CHR$(10)
s = s & "Le curseur est à " & Format(dXRight, "0.##") & " pouces du bord droit " & CHR$(10)
s = s & "Le curseur est à " & Format(dYCursor, "0.##") & " pouces du haut " & CHR$(10)
s = s & "Le curseur est à " & Format(dYBottom, "0.##") & " pouces du bas " & CHR$(10)
s = s & "Marge gauche = " & dLeftMargin & " pouces" & CHR$(10)
s = s & "Marge du bas = " & dBottomMargin & " pouces" & CHR$(10)
s = s & "Hauteur des caractères = " & Format(dCharHeight, "0.#####") & " pouces" & CHR$(10)

' RunSimpleObjectBrowser(xStyle)

' Dim dFinalX As Double
' Dim dFinalY As Double
' dFinalX = dXCursor + dLeftMargin
' dFinalY = (v.Y + xStyle.TopMargin)/2540 + dCharHeight / 2
' s = s & "Le curseur dans la page est à (" & Format(dFinalX, "0.#####") & ", " &
'     Format(dFinalY, "0.#####") & ") en pouces" & CHR$(10)

REM Maintenant vérifions le pied de page !
If xStyle.FooterIsOn Then
    v = IIF(iCurPage MOD 2 = 0, xStyle.FooterTextLeft, xStyle.FooterTextRight)
    If IsNull(v) Then v = xStyle.FooterText
    If Not IsNull(v) Then
        REM Sauvegarde la position
        Dim xOldCursor
        xOldCursor = xViewCursor.getStart()
        xViewCursor.gotoRange(v.getStart(), false)
        Print "Position du pied de page = " & CStr(xViewCursor.getPosition().Y/2540.0)
        dFinalY = xViewCursor.getPosition().Y/2540.0 - dYCursor + dBottomMargin
        xViewCursor.gotoRange(xOldCursor, false)
    End If
Else
    Print "Pas de pied de page"
End If
s = s & "Le curseur dans la page est à (" & Format(dFinalX, "0.#####") & ", " &
    Format(dFinalY, "0.#####") & ") en pouces" & CHR$(10)
MsgBox s, 0, "Information sur la page"
End Sub

```

Je connais très peu de choses sur les formulaires. Aussi, les informations données ici peuvent ne pas être tout à fait exactes. Le guide du développeur a de bons exemples sur les formulaires.

Frank Schönheit [fs@openoffice.org] attire l'attention sur ceci :

Pour modifier un contrôle AWT, il faut faire les changements sur le et le contrôle qui appartient à ce modèle est automatiquement mis à jour et synchronisé. Changer le contrôle directement pourrait amener à des incohérences. Par exemple, pour une liste de sélection, ne pas utiliser l'interface XListBox, fournie par le contrôle, pour sélectionner une entrée (XListBox::selectItem) mais plutôt la propriété com.sun.star.awt.UnoControlListBoxModel::SelectedItems du modèle.

Voir:

<http://api.openoffice.org/docs/common/ref/com/sun/star/awt/UnoControlListBoxModel.html>

Introduction

Un formulaire est un ensemble de 'forms' et/ou de 'contrôles' comme les boutons ou les listes déroulantes. Les formulaires peuvent être utilisés pour accéder à des sources de données ou des choses plus compliquées.

<http://api.openoffice.org/docs/common/ref/com/sun/star/form/XFormsSupplier.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/form/module-ix.html>

Pour obtenir un formulaire en OOBasic, il vous faut d'abord accéder à la 'DrawPage'. La méthode pour l'obtenir dépend du type de document. Comme test, j'ai inséré un bouton dans une feuille de calcul, je l'ai nommé 'TestButton' et le formulaire 'TestForm'. J'ai alors créé la macro suivante qui change l'étiquette du bouton quand on clique dessus.

```
Sub TestButtonClick
Dim vButton, vForm
vForm=THISCOMPONENT.CurrentController.ActiveSheet.DrawPage.Forms.getByName("TestForm")
vButton = vForm.getByName("TestButton")
vButton.Label = "Wow"
Print vButton.getServiceName()
End Sub
```

Si j'avais travaillé avec un document Texte, on obtiendrait le formulaire comme suit :

```
oForm=THISCOMPONENT.DrawPage.Forms.getByName("FormName")
```

Boîtes de dialogue

La plupart des exemples de boîtes de dialogue commencent ainsi :

```
Dim oDlg As Object
Sub StartDialog
```

```
oDlg = CreateUnoDialog(DialogLibraries.Standard.Dialog1 )
oDlg.execute()
End Sub
```

Notez la variable appelée oDlg qui est définie en dehors de la fonction où elle est créée. Ceci est nécessaire car elle peut être manipulée par d'autres procédures que l'on appelle comme gestionnaires d'événement. Si ces procédures accèdent à la boîte de dialogue, alors elles doivent pouvoir accéder à la variable qui la référence.

Les boîtes de dialogue, comme les bibliothèques de macros, ont leur propre hiérarchie. Dans cet exemple, j'ai entré le code de la macro montrée ci dessus. Puis j'ai choisi Outils>Macros et j'ai cliqué sur le bouton « Gérer ». Vous noterez que ceci organise les choses sous le document, avec une section nommée Standard. Lorsque vous cliquez sur « nouv. boîte de dialogue », le nom par défaut est « Dialog1 ». Parce que ce code tourne depuis une macro embarquée dans un document, DialogLibraries se réfère à la hiérarchie des librairies du document. Si vous voulez accéder à la hiérarchie des bibliothèques de l'application depuis la macro d'un document, vous devez utiliser GlobalScope.DialogLibraries.

La méthode standard de fermeture d'une boîte de dialogue implique de mettre un gestionnaire d'événement qui la fermera. Généralement, j'ajoute un bouton « Fermer » qui appelle une méthode semblable à ce qui suit :

```
Sub CloseDialog
    oDlg.endExecute()
End Sub
```

Si vous voulez que la boîte de dialogue se ferme d'elle même après une durée spécifiée, vous créez votre routine de fermeture de la boîte de dialogue et vous la reliez à un gestionnaire d'événement approprié.

```
Sub CloseDialogAfterWaiting
    'Attendre deux secondes
    wait(2000)
    oDlg.endExecute()
End Sub
```

S'il vous plaît, excusez la grande quantité de détails présents, mais je sentais que c'était nécessaire pour les novices. Lors de l'édition de ma nouvelle boîte de dialogue, je clique sur le bouton « Champs de contrôle » de la barre d'outils et je choisis un bouton de commande. Je le positionne. Je clique avec le bouton droit sur lui et choisis « propriétés ». A partir de l'onglet « Général », je mets « BoutonQuitter » comme nom pour ce bouton et comme étiquette « Quitter ». Puis j'ouvre l'onglet « Événements » et à « lors du déclenchement », je clique sur « ... » pour ouvrir la fenêtre qui me permet d'assigner des actions aux événements. Dans l'onglet « Événements », je choisis « lors du déclenchement ». Dans la partie « Macro », je sélectionne la routine « CloseDialog » et je clique sur « Assigner ». Bien que ceci gère aussi bien l'activation par la souris que par le clavier, il est possible d'avoir une procédure différente pour « KeyPressed » et « MouseClicked » mais je n'ai pas de raisons de les différencier.

Vous devez avoir une variable référençant la boîte de dialogue en dehors de la routine qui la crée afin de pouvoir y accéder depuis d'autres routines.

Depuis une macro d'un document,
DialogLibraries.Standard.Dialog1 référence une boîte de dialogue dans le document courant.
GlobalScope.DialogLibraries.Standard... référence une boîte de dialogue globale.

Les contrôles partagent tous certaines fonctionnalités communes. Quelques unes sont ici :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/UnoControl.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XWindow.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/module-ix.html>

Ceci permet de contrôler des choses telles que la visibilité, l'activation, la taille et autres caractéristiques communes. Plusieurs contrôles partagent des méthodes communes telles que setLabel(string). Plusieurs événements différents sont supportés. Par expérience, les plus communément utilisés se rapportent aux notifications de changements d'état des contrôles.

Un contrôle peut être obtenu depuis une boîte de dialogue utilisant la méthode getControl(control_name). Il est également possible de faire des itérations au travers de l'ensemble des contrôles si vous le désirez.

Un champ d'étiquette agit comme un texte régulier dans la boîte de dialogue. Il est habituellement utilisé pour étiqueter un contrôle. Il est possible de spécifier ou de récupérer le texte du champ d'étiquette en utilisant getText() et setText(string). Il est également possible de préciser l'alignement du texte à gauche, centré, ou à droite. Il est courant d'aligner à droite le texte d'un champs d'étiquette afin qu'il soit au plus près du contrôle qu'il identifie. Voir :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XFixedText.html>

Généralement, un contrôle Bouton est seulement utilisé pour appeler une procédure lorsque le bouton est pressé. Il est également possible d'appeler setLabel(string) pour changer le texte du bouton. Voir aussi :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XButton.html>

Une zone de texte est utilisée pour saisir du texte standard. Il est possible de limiter la taille maximum du texte et de contrôler le nombre maximum de lignes. Il est possible d'écrire vos propres systèmes de formatage si vous le souhaitez. Les méthodes les plus utilisées sont getText() et setText(string). Si vous voulez un ascenseur, vous devez le paramétriser dans les propriétés de la zone de Texte pendant la conception de la boîte de dialogue.

Il y a des boîtes de saisie particulières pour les dates, heures, nombres, masques de saisie, champs formatés et les monnaies. Si vous en utilisez, soyez certain de prêter une attention particulière aux propriétés pour voir ce qu'elles peuvent faire. Vous pouvez désactiver un contrôle de format strict ou fournir des plages d'entrées limitées, par exemple. Un champ masqué dispose d'un masque de saisie et d'un masque de caractère. Le masque de saisie détermine les données que l'utilisateur peut entrer. Le masque de caractères détermine l'état du champ masqué lors du chargement du formulaire. Le champ formaté autorise un formatage arbitraire permis par OOo. Si je voulais un champ pour un numéro de sécurité sociale ou pour un pourcentage, j'utiliserais ce champ.

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XTextComponent.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XTimeField.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XDateField.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XCurrencyField.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XNumericField.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XPatternField.html>

Une zone de liste fournit une liste de valeurs parmi lesquelles vous pouvez en sélectionner une. Vous pouvez choisir d'activer la sélection multiple. Pour additionner des éléments à la liste, j'utilise couramment quelque chose du type addItems(Array("one", "two", "three"), 0). Il est également possible d'ôter des éléments d'une zone de liste.

Pour une sélection unique, vous pouvez utiliser getSelectedltemPos() pour déterminer quel élément est sélectionné. -1 est renvoyé si il n'y a rien de sélectionné. Si quelque chose est sélectionnée, 0 signifie que c'est le premier de la liste. Pour des sélections multiples, utilisez getSelectedItemsPos() qui retourne un tableau contenant les indexs sélectionnés de la liste. Voir :
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XListBox.html>

Une zone combinée est un champ d'entrée avec une zone de liste liée. Ceci est parfois appelé une liste déroulante. Dans mon exemple, j'ai placé une zone combinée en deux temps. D'abord j'entre les valeurs de la liste et je positionne la zone combinée sur propriété :

```
aControl.addItems(Array("properties", "methods", "services"), 0)
aControl.setText("properties")
```

Je vais ensuite dans les événements et j'indique pour "statut changé" que la routine NewDebugType peut être appelée. Ceci affichera l'intégralité des méthodes, propriétés et services supportés par la boîte de dialogue. J'ai fait ceci pour montrer qu'un appel en retour d'un événement est réellement utile pour obtenir des informations de débogage. Voir :
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XComboBox.html>

Bien qu'une case à cocher soit généralement utilisée pour indiquer un unique état (« oui » ou « non »), il existe une case à cocher commune nommée case à cocher à trois état. Dans OOo une case à cocher peut avoir l'état 0, non cochée, ou 1, cochée. Vous pouvez utiliser getState() pour obtenir l'état courant. Si vous appelez enableTriState(true), alors l'état 2 est autorisé. Cet état positionne une valeur grisée dans la case traduisant un état intermédiaire 'au tout ou rien habituel'. Vous pouvez paramétrer l'état en utilisant setState (int). Voir :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XCheckBox.html>

L'utilité d'un bouton radio est généralement de sélectionner un choix parmi plusieurs propositions. Pour cette raison, on insère habituellement, d'abord une zone de groupe puis on place les boutons radio dedans. Pour savoir lequel a été sélectionné, vous pouvez appeler la méthode getState() sur chacun d'eux jusqu'à ce que vous le trouviez. J'ai pu utiliser les boutons radio plutôt qu'une liste déroulante dans mon exemple pour choisir quoi afficher dans ma boîte de liste de débogage. Voir :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XRadioButton.html>

Dans mon exemple, j'utilise une barre de progression pour montrer la progression du remplissage d'une boîte de liste de débogage. Ce n'est probablement pas un bon exemple parce que cela se produit trop vite mais cela reste un exemple.

Vous pouvez définir la plage de progression avec setRange(min, max). Ceci rend plus facile la mise à jour de la progression. Comme je traite une chaîne de caractère, je mets le minimum à 0 et le maximum à la longueur de la chaîne. J'appelle alors setValue(int) avec la position courante dans la chaîne pour montrer où j'en suis. Voir :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XProgressBar.html>

Obtention des Contrôles

Si vous voulez énumérer les contrôles présents dans un formulaire, ce qui suit marchera :

```
Sub EnumerateControlsInForm
    Dim oForm, oControl, iNumControls%, i%
    'Par défaut, c'est là que se trouvent lescontrôles
    oForm = ThisComponent.Drawpage.Forms.getByName("Standard")
    oControl = oForm.getByName("MyPushButton")
    MsgBox "getByName est utilisé pour obtenir le contrôle appelé " & oControl.Name
    iNumControls = oForm.Count()
    For i = 0 To iNumControls -1
        MsgBox "Le contrôle " & i & " est nommé " & oControl.Name
    Next
End Sub
```

Pour les contrôles dans une boîte de dialogue, utiliser le code qui suit :

```
x = oDlg.getControls()
For ii=LBound(x) To UBound(x)
    Print x(ii).getImplementationName()
```

Next

Généralement, les contrôles seront recherchés par noms plutôt que par énumérations.

La solution a finalement été fournie par Paolo Mantovani
[mantovani.paolo@tin.it]

Un contrôle de formulaire est placé sur une forme de dessin
(com.sun.star.drawing.XControlShape), il faut donc obtenir la forme sous-jacente afin de gérer la taille et la position du contrôle.

La bibliothèque Outils (module "Module Controls") fournit quelques méthodes qui permettent de travailler avec des contrôles de formulaire, notamment les fonctions suivantes :

```
GetControlModel()
GetControlShape()
GetControlView()
```

En espérant que cela vous aide... Paolo Mantovani

En effet Paolo, cela m'a permis d'écrire la macro suivante :

```
Sub SeeThingProp
    Dim oForm, oControl, iNumControls%, i%
    Dim v()
    'Par défaut, les contrôles se trouvent ici
    oForm = ThisComponent.Drawpage.Forms.getByName("Standard")
    oControl = oForm.getByName("MyPushButton")
    oControl = oForm.getByName("CheckBox")
    Dim vShape
    Dim vPosition, vSize
    Dim s$, vv
    vShape = GetControlShape(ThisComponent, "CheckBox")
    vPosition = vShape.getPosition()
    vSize = vShape.getSize()
    RunSimpleObjectBrowser(vSize)
    s = s & "Position = (" & vPosition.X & ", " & vPosition.Y & ")" & CHR$(10)
    s = s & "Height = " & vSize.Height & " Width = " & vSize.Width & CHR$(10)
    MsgBox s
End Sub
```

Sélection d'un fichier depuis une boîte de dialogue Fichier

L'exemple suivant affiche la boîte standard de dialogue Fichier

```
Sub ExampleGetAFileName
    Dim filterNames(1) As String
    filterNames(0) = "*.*"
    filterNames(1) = "*.*"
    Print GetAFileName(filterNames())
End Sub

Function GetAFileName(FilterNames() As String)
    Dim oFileDialog As Object
    Dim iAccept As Integer
    Dim sPath As String
    Dim InitPath As String
    Dim RefControlName As String
    Dim oUcb As Object
    'Dim ListAny(0)
    'Nota : Les services suivants doivent être appelés dans l'ordre suivant
    'sinon le Basic n'enlève pas le service FileDialog
    oFileDialog = CreateUnoService("com.sun.star.ui.dialogs.FilePicker")
```

```

oUcb = createUnoService("com.sun.star.ucb.SimpleFileAccess")
'ListAny(0) = com.sun.star.ui.dialogs.TemplateDescription.FILEOPEN_SIMPLE
'oFileDialog.initialize(ListAny())
AddFiltersToDialog(FilterNames(), oFileDialog)

'Mets ton chemin initial ici !
'InitPath = ConvertToUrl(oRefModel.Text)

If InitPath = "" Then
    InitPath = GetPathSettings("Work")
End If
If oUcb.Exists(InitPath) Then
    oFileDialog.SetDisplayDirectory(InitPath)
End If
iAccept = oFileDialog.Execute()
If iAccept = 1 Then
    sPath = oFileDialog.Files(0)
    GetAFileName = sPath
    'If oUcb.Exists(sPath) Then
        '    oRefModel.Text = ConvertFromUrl(sPath)
    'End If
End If
oFileDialog.Dispose()
End Function

```

Centrage d'une boîte de dialogue à l'écran

Remerciements à Berend Cornelius [Berend.Cornelius@sun.com] d'avoir fourni cette macro. L'astuce réside dans l'utilisation du contrôleur courant afin de récupérer la fenêtre du composant et, à partir de là, récupérer la position et la taille de la fenêtre.

```

Sub Main
    Dim CurPosSize as new com.sun.star.awt.Rectangle
    FramePosSize = ThisComponent.getCurrentController().Frame.getComponentWindow.PosSize
    xWindowPeer = oDialog.getPeer()
    CurPosSize = oDialog.getPosSize()
    WindowHeight = FramePosSize.Height
    WindowWidth = FramePosSize.Width
    DialogWidth = CurPosSize.Width
    DialogHeight = CurPosSize.Height
    iXPos = ((WindowWidth/2) - (DialogWidth/2))
    iYPos = ((WindowHeight/2) - (DialogHeight/2))
    oDialog.setPosSize(iXPos, iYPos, DialogWidth, DialogHeight, com.sun.star.awt.PosSize.POS)
    oDialog.execute()
End Sub

```

Programmer les réactions aux évènements de contrôle

Le petit extrait de code suivant assigne les évènements d'une boîte de dialogue à la macro Test du module Module1 de la librairie standard. Merci à Oliver Brinzing [OliverBrinzing@t-online.de] pour ce code.

```

Sub SetEvent
    Dim oDocument as Object
    Dim oView as Object
    Dim oDrawPage as Object
    Dim oForm as Object
    Dim oEvents(0) as New com.sun.star.script.ScriptEventDescriptor

    oDocument = StarDesktop.getCurrentComponent
    oView = oDocument.CurrentController
    oDrawPage = oView.getActiveSheet.DrawPage

```

```

' atteindre le premier formulaire
oForm = oDrawPage.getForms.getByIndex(0)

oEvents(0).ListenerType = "XActionListener"
oEvents(0).EventMethod = "actionPerformed"
oEvents(0).AddListenerParam = ""
oEvents(0).ScriptType = "StarBasic"

oEvents(0).ScriptCode = "document:Standard.Module1.Test"
oForm.registerScriptEvent(0, oEvents(0))
End Sub

Sub Test(oEvt)
    Print oEvt.Source.Model.Name
End Sub

```

Je suppose qu'une brève description sur la macro registerScriptEvent est nécessaire. Regardez

<http://api.openoffice.org/docs/common/ref/com/sun/star/script/XEventAttacherManager.html> pour de bons éléments de départ.

```
registerScriptEvent(index, ScriptEventDescriptor)
```

La variable oEvents est un tableau de descripteurs d'évènements, voir la page <http://api.openoffice.org/docs/common/ref/com/sun/star/script/ScriptEventDescriptor.html>, qui décrit l'évènement qui sera attaché (EventMethod) et quelle macro sera appelée (ScriptCode). L'exemple ci-dessus aurait pu utiliser une simple variable au lieu d'un tableau parce que le code utilise une seule entrée du tableau. Le premier argument passé à la méthode registerScriptEvent est un index vers l'objet qui < utilisera > le descripteur d'évènements. Les pages d'aide considèrent que vous connaissez quels objets sont indexés et indiquent que « si un objet est attaché à cet index, alors l'événement sera attaché automatiquement ». Ce qui n'est pas précisé est que le formulaire agit comme un conteneur d'objets pour les composants de type formulaire. Ces composants de type formulaire sont accessibles par leur nom ou leur index.

Comment rendre une boîte de dialogue non-modale

CP Hennessy [cphennedy@openoffice.org] a posé cette question et l'a résolue avec Python.

Comment rendre une boîte de dialogue UnoControlDialog non-modale ?

```

toolkit = smgr.createInstanceWithContext( "com.sun.star.awt.Toolkit", ctx)
aRect = uno.createUnoStruct( "com.sun.star.awt.Rectangle" )
aRect.X = 100
aRect.Y = 100
aRect.Width = posX
aRect.Height = 25
aDescriptor = uno.createUnoStruct( "com.sun.star.awt.WindowDescriptor");
aDescriptor.Type = TOP;
aDescriptor.WindowServiceName = "workwindow";
aDescriptor.ParentIndex = 1;
aDescriptor.Bounds = aRect;
peer = toolkit.createWindow( aDescriptor )

self.dialog.createPeer( toolkit, peer )

```

Je (Andrew Pitonyak) n'ai pas testé cette méthode, ni même essayé de la convertir en OOo Basic, mais cela devrait fonctionner.

Interception des Entrées Clavier

Une question a été posée pour lier l'évènement "touche enfoncée" dans un champ de texte à une sous-routine. Comment savoir quelle était la touche enfoncée ? Laurent Godard a fourni la réponse suivante :

```
Sub MySub(oEvent as object)
    If oEvent.keyCode=com.sun.star.awt.key.RETURN Then
        MsgBox "RETURN Pressed"
    End If
End Sub
```

Création d'une boîte de dialogue par programmation

Ceci représente en réalité deux problèmes. Il faut pouvoir créer une boîte de dialogue et puis y ajouter des contrôles. Ces deux situations ont été traitées par Paolo Mantovani et Thomas Benisch.

Beaucoup de questions trouvent réponse dans le guide du développeur, notamment dans le chapitre 'Basic and Dialogs', sections 'Programming Dialogs and Dialog Controls' et 'Creating Dialogs at Runtime'. Le code suivant insère un contrôle dans une boîte de dialogue existante.

```
Sub Main
    Dim oDialog As Object
    Dim oDialogModel As Object
    Dim oButtonModel As Object

    Const sButtonName = "Button1"

    REM charger la bibliothèque d'outils
    BasicLibraries.LoadLibrary( "Tools" )

    REM charger la boîte de dialogue
    oDialog = LoadDialog( "Standard", "Dialog1" )

    REM obtenir le modèle de la boîte de dialogue
    oDialogModel = oDialog.getModel()

    REM créer un modèle de bouton
    oButtonModel = oDialogModel.createInstance( _
        "com.sun.star.awt.UnoControlButtonModel" )
    oButtonModel.PositionX = 140
    oButtonModel.PositionY = 120
    oButtonModel.Width = 50
    oButtonModel.Height = 20
    oButtonModel.Label = "OK"
    oButtonModel.Name = sButtonName
    oButtonModel.TabIndex = 1

    REM insérer le modèle de bouton dans le modèle de boîte de dialogue
    oDialogModel.insertByName( sButtonName, oButtonModel )
    REM afficher la boîte de dialogue
    oDialog.execute()
End Sub
```

Un exemple conséquent est fourni dans cette section. Le code pour la création

de la boîte de dialogue est compliqué parce que je ne maîtrise pas assez les concepts subtils de l'awt, mais il fonctionne. (Note d'Andrew : les variables ne sont pas déclarées, donc le code échouera si on utilise Option Explicit)

```
REM ***** BASIC *****
Global oDlg As Object
Global oButton As Object
Global oTextEdit As Object

' Création d'une boîte de dialogue StarBasic à partir de rien et ajouter quelques contrôles.
Sub ExampleDialogFromScratch

' Obtenir la boîte de dialogue
sTitle = "Créée à partir de rien!!"
aRect = CreateUnoStruct("com.sun.star.awt.Rectangle")
    aRect.Y = 150
    aRect.X = 150
    aRect.Width = 400
    aRect.Height = 250

oDlg = CreateUnoDialogFromScratch(sTitle, aRect)

' Ajouter un bouton de Commande
sControlType = "Button"
sControlName = "BoutonCommande1"
aRect = CreateUnoStruct("com.sun.star.awt.Rectangle")
    aRect.Y = 5
    aRect.X = 5
    aRect.Width = 50
    aRect.Height = 12

oButton = AddControl(oDlg, sControlType, sControlName, aRect)
oButton.Label = "Cliquer ici..."

' Ajouter un Champ de Texte Editable
sControlType = "Edit"
sControlName = "TextEdit1"
aRect = CreateUnoStruct("com.sun.star.awt.Rectangle")
    aRect.Y = 5
    aRect.X = 60
    aRect.Width = 100
    aRect.Height = 12

oTextEdit = AddControl(oDlg, sControlType, sControlName, aRect)
oTextEdit.Text = "Salutation de France !!! :-)"

' Gestion des évènements
oActionListener =
createUnoListener("oButton_", "com.sun.star.awt.XActionListener")

oButton.addActionListener (oActionListener)

' Affichage de la boîte de dialogue
oDlg.execute

' Nettoyage des objets
oButton.removeActionListener (oActionListener)
oDlg.dispose
End Sub

*****
' Gestion des Evènements
*****
Sub oButton_disposing(oEvt)
```

```

' rien à faire
End Sub

Sub oButtonActionPerformed(oEvt)
    nColor = cLng(Rnd * 255 * 255 * 255)
    oTextEdit.Model.BackGroundColor = nColor
End Sub

*****  

' Fonctions d'Assistance  

*****  

Function AddControl(oDialog, sControlType, sControlName, aPosSize) As Object
    oDlgModel = oDialog.getModel

    sUnoName = "com.sun.star.awt.UnoControl" & sControlType & "Model"
    oControlModel = oDlgModel.createInstance( sUnoName )

    With oControlModel
        .PositionX = aPosSize.X
        .PositionY = aPosSize.Y
        .Width = aPosSize.Width
        .Height = aPosSize.Height
        .Enabled = True
    End With

    ' Insertion du modèle de contrôle dans le modèle de boîte de dialogue
    oDlgModel.insertByName(sControlName, oControlModel)

    ' ce qui donne la vue du contrôle
    AddControl = oDialog.getControl(sControlName)
End Function

Function CreateUnoDialogFromScratch( sTitle, aPosSize) As Object
    oDlgView = createUnoService("com.sun.star.awt.UnoControlDialog")
    oDlgModel = createUnoService("com.sun.star.awt.UnoControlDialogModel")

    oDlgView.setModel(oDlgModel)

    ' Ceci semble parfois être nécessaire afin d'éviter des erreurs
    ' mais les valeurs ne semblent avoir aucun effet
    ' dim aRect as new com.sun.star.awt.Rectangle
    ' aRect.Y = 0
    ' aRect.X = 0
    ' aRect.width = 0
    ' aRect.height = 0

    dim aDescriptor as new com.sun.star.awt.WindowDescriptor

    ' autres valeurs : TOP, MODALTOP, CONTAINER,
    ' je ne connais rien de leurs effets
    aDescriptor.Type = com.sun.star.awt.WindowClass.SIMPLE

    aDescriptor.WindowServiceName = "testdialog"
    aDescriptor.ParentIndex = 0
    aDescriptor.Parent = Null
    aDescriptor.Bounds = aRect
    est-ce vraiment nécessaire?

    oToolkit = createUnoService("com.sun.star.awt.Toolkit")
    oWindow = oToolkit.createWindow(aDescriptor)

    oDlgView.createPeer(oToolkit, oWindow)

    oDlgView.setPosSize(
        aPosSize.X, _

```

```
aPosSize.Y,  
aPosSize.Width,  
aPosSize.Height,  
com.sun.star.awt.PosSize.POSSIZE)  
  
oDlgView.setTitle(sTitle)  
  
CreateUnoDialogFromScratch = oDlgView  
End Function
```

NdT : En accord avec l'auteur, les éléments contenus dans ce chapitre ont été fusionnés avec le chapitre 5

Un gestionnaire, au sens de ce chapitre, est tout code qui utilise des appels en retour ou qui est amené d'une manière ou d'une autre à traiter certains événements.

XKeyHandler

Cet exemple de gestionnaire est fourni par Leston Buell [bulbul@ucla.edu]. Sa description du code suit :

Appeler "Compose" et taper une combinaison de deux lettres choisie parmi ce qui suit :

Ch, ch, Gh, gh, Hh, hh, Jh, jh, Sh, sh, Uh, uh

Celles-ci insèrent respectivement les caractères d'espéranto suivants dans le document :

Ĉ, ĉ, Ĝ, ĝ, Ĥ, ĥ, Ĵ, ĵ, Ĳ, ķ, Ÿ, ū

Note : peu de fontes ont ces caractères.

Commentaires de Leston avec seulement une modification mineure : Je suis un développeur Java/Python/Javascript et je ne connais pas grand chose au Basic. Ceci pourrait être réécrit pour être modulaire et extensible. Par exemple, on devrait pouvoir charger différents jeux de caractères comme un jeu latin, un jeu hébreu, un jeu cyrillique et les symboles mathématiques. Ces jeux pourraient être dans un tableau plutôt que de faire appel au long "Select...case" que j'ai utilisé. Ceci vous permettrait de générer une table des jeux à la volée. On pourrait également modifier ce code pour travailler sur plus d'un document.

Ceci dit, cette macro fonctionne et satisfait à mes besoins actuels. Ce document contient une version épurée, qui dispose seulement des caractères nécessaires pour l'espéranto. La vraie version, qu'à un certain point je rendrai disponible sur le web, supporte toutes les lettres non-ASCII de Latin1, ExtendedLatinA et ExtendedLatinB, plus d'autres particularités comme les guillemets français et le symbole de l'euro.

Pour utiliser cette macro, appeler "Compose" et saisir une des combinaisons de caractères : "Gh", "gh", "Jh", "uh" – ou n'importe quelle autre montrée dans la fonction GetTranslation. Le caractère Unicode correspondant est inséré dans le document. KeyHandler se désenregistre après deux tours, qu'il y ait eu traduction ou non.

J'ai (Leston) assigné Compose à Ctrl+H, ce qui est facile à taper (Note d'Andrew : *Vous pourrez également l'affecter à d'autres touches pour supporter d'autres raccourcis tels que les touches de contrôle de Word Star*). Je ne comprends pas les portées des variables du Basic, aussi j'ai tout mis en

variables globales au début. Si cela n'était pas la bonne manière de procéder, sentez vous libre de changer cela (Note d'Andrew : *Ceci peut amener des problèmes inattendus*). Je pense qu'il y a un bug dans la macro. Ce bug est noté en commentaire dans la fonction InsertString.

Lorsque que la version finale sera disponible, je l'hébergerai moi-même ou je fournirai un lien.

```
REM Auteur:Leston Buell [bulbul@ucla.edu]
Global oComposerDocView
Global oComposerKeyHandler
Global oComposerInputString

Sub Compose
    oComposerDocView = ThisComponent.getCurrentController
    oComposerKeyHandler = createUnoListener( "Composer_",
        "com.sun.star.awt.XKeyHandler" )
    oComposerDocView.addKeyHandler( oComposerKeyHandler )
    oComposerInputString = ""
End Sub

Sub ExitCompose
    oComposerDocView.removeKeyHandler( oComposerKeyHandler )
    oComposerInputString = ""
End Sub

Function InsertString( oString, oDocView, oKeyHandler )
    Dim oViewCursor
    Dim oText
    Dim oCursor

    oViewCursor = ThisComponent.getCurrentController().getViewCursor()
    oText = ThisComponent.getText()
    oCursor = oText.createTextCursorByRange( oViewCursor.getStart() )
    'Pour une raison indéterminée, l'insertion de texte relance les événements des touches (deux fois !)
    'Aussi nous enlevons le gestionnaire avant l'insertion, puis le remettons après
    oDocView.removeKeyHandler( oKeyHandler )
    oText.insertString( oCursor.getStart(), oString, true )
    oDocView.addKeyHandler( oKeyHandler )
End Function

Function Composer_keyPressed( oEvt ) as Boolean
    If len( oComposerInputString ) = 1 Then
        oComposerInputString = oComposerInputString & oEvt.KeyChar
        Dim translation
        translation = GetTranslation( oComposerInputString )
        InsertString( translation, oComposerDocView, oComposerKeyHandler )
        oComposerInputString = ""
        ExitCompose
    Else
        oComposerInputString = oComposerInputString & oEvt.KeyChar
    End If
    Composer_KeyPressed = True
End Function

Function Composer_keyReleased( oEvt ) as Boolean
    Composer_keyReleased = False
End Function

Function GetTranslation( oString ) as String
```

```

Select Case oString
    Case "^C", "Ch"
        GetTranslation = "Ĉ"
    Case "^c", "ch"
        GetTranslation = "ĉ"
    Case "^G", "Gh"
        GetTranslation = "Ĝ"
    Case "^g", "gh"
        GetTranslation = "ĝ"
    Case "^H", "Hh"
        GetTranslation = "Ĥ"
    Case "^h", "hh"
        GetTranslation = "ĥ"
    Case "^J", "Jh"
        GetTranslation = "Ĵ"
    Case "^j", "jh"
        GetTranslation = "ĵ"
    Case "^S", "Sh"
        GetTranslation = "Ŝ"
    Case "^s", "sh"
        GetTranslation = "ŝ"
    Case "uU", "Uh"
        GetTranslation = "Ŭ"
    Case "uu", "uh"
        GetTranslation = "ŭ"
End Select
End Function

```

Description des Auditeurs d'événements par Paolo Mantovani

Le texte dans cette nouvelle section a été écrit par Paolo Mantovani. J'ai (Andrew Pitonyak) effectué quelques modifications mineures mais je n'ai pas tenté de vérifier ce code. Ceci dit, je peux vous assurer que Paolo est une source extrêmement fiable. Merci Paolo d'avoir pris le temps. C'est l'une des meilleures descriptions que j'ai vues sur le sujet. Le document contenait ces mentions légales quand je l'ai reçu, aussi je les inclus ici :

© 2003 Paolo Mantovani

This document is released under the Public Documentation License Version 1.0
A copy of the License is available at <http://www.openoffice.org/licenses/pdl.pdf>

L'environnement d'exécution de OOBasic fournit une fonction nommée CreateUnoListener. Cette fonction comprend deux arguments : un préfixe (String) et un nom pleinement qualifié d'interface d'auditeur d'événement (String). Voici la syntaxe :

```
oListener = CreateUnoListener( sPrefix , sInterfaceName )
```

Depuis les versions 1.1, cette fonction est très bien décrite dans l'aide du Basic OO, mais nous voudrions ajouter quelques considérations à son sujet.

Considérez le code suivant :

```
oListener = CreateUnoListener("prefix_","com.sun.star.lang.XEventListener")
MsgBox oListener.Dbg_supportedInterfaces
MsgBox oListener.Dbg_methods
```

L'interface com.sun.star.lang.XEventListener est à la base des autres auditeurs d'événements, toutefois c'est le plus simple que vous puissiez trouver. Notez que cette interface travaille comme interface de base pour les autres, donc

vous ne devriez pas l'utiliser de manière explicite, mais ce code marche très bien pour cet exemple.

Les macros Basic sont capables d'appeler les méthodes et les propriétés de l'API. Généralement, une macro fait beaucoup d'appels à l'API. A l'inverse, l'API ne peut généralement pas appeler des routines Basic. Considérez l'exemple suivant :

```
Sub Example_Listener
    oListener = CreateUnoListener("prefix_", "com.sun.star.lang.XEventListener")
    Dim oArg As New com.sun.star.lang.EventObject
    oListener.disposing( oArgument )
End Sub

Sub prefix_disposing( vArgument )
    MsgBox "Hi all!!!"
End Sub
```

Lorsque la première routine appelle "oListener.disposing()", la routine prefix_disposing est appelée. Toutefois, la fonction CreateUnoListener crée un service capable d'appeler les procédures OOBasic.

Vous devrez créer les routines et sous-routines avec des noms correspondants à ceux des auditeurs d'événement en ajoutant le préfixe spécifié lorsque que vous appelez CreateUnoListener. Par exemple, l'appel de CreateUnoListener passe un premier argument "prefix_" et la sous-routine "prefix_disposing" démarre avec "prefix_" .

La documentation pour l'interface com.sun.star.lang.XEventListener dit que l'argument doit être une structure com.sun.star.lang.EventObject.

Bien que ceci ne soit pas le meilleur moyen pour appeler une macro depuis une autre, UNO nécessite un auditeur d'événement pour appeler vos macros.

Lorsque vous voudrez utiliser un auditeur pour intercepter des événements, vous aurez besoin d'un objet capable de parler à votre auditeur. L'objet UNO qui appellera votre auditeur est nommé un "broadcaster". Les objets UNO broadcaster utilisent des méthodes pour ajouter ou ôter les auditeurs appropriés. Pour créer un objet auditeur, passez le nom pleinement qualifié de l'interface de l'auditeur à la fonction CreateUnoListener. Récupérez les méthodes supportées par l'auditeur en accédant aux propriétés de la méthode Dbg_methods property (ou vérifiez la documentation IDL pour les interfaces d'auditeurs). Finalement, implémentez une routine basic pour chaque méthode, même pour celle qui libère l'objet.

La plupart des services UNO fournissent des méthodes pour enregistrer et déréférencer les auditeurs. Par exemple, le service com.sun.star.OfficeDocumentView supporte l'interface com.sun.star.view.XSelectionSupplier. Ceci est le broadcaster. Cette interface fournit les méthodes suivantes :

```
addSelectionChangeListener
```

enregistre un auditeur d'événement, qui est appelé lorsque la sélection change.

```
removeSelectionChangeListener
```

déréférence l'auditeur qui a été enregistré par addSelectionChangeListener.

Les deux méthodes prennent un argument SelectionChangeListener (qui est un service qui supportant l'interface com.sun.star.view.XSelectionChangeListener).

L'objet broadcaster introduit un ou plusieurs arguments dans l'appel. Le premier argument est une structure UNO, les suivants dépendent de la définition de l'interface. Vérifiez la documentation IDL de l'interface de l'auditeur que vous utilisez et voyez le détail des méthodes. Souvent, la structure passée est un objet com.sun.star.lang.EventObject. Toutefois, toutes les structures d'événement doivent étendre com.sun.star.lang.EventObject donc elles doivent avoir au moins l'élément source.

Ce qui suit est une implémentation complète d'un auditeur de changement de sélection. Il peut être utilisé avec tous les documents OpenOffice.org.

Option Explicit

```
Global oListener As Object  
Global oDocView As Object
```

```
'exécuter cette macro démarre l'interception d'événement
```

```
Sub Example_SelectionChangeListener  
    oDocView = ThisComponent.getCurrentController
```

```
'Crée un auditeur pour intercepter l'événement changement de sélection  
oListener = CreateUnoListener( "MyApp_ ", "com.sun.star.view.XSelectionChangeListener" )
```

```
' enregistre l'auditeur auprès du contrôleur du document  
oDocView.addSelectionChangeListener(oListener)
```

```
End Sub
```

```
'exécuter cette macro stoppe l'interception d'événement
```

```
Sub Remove_Listener  
    'déréférence l'auditeur  
    oDocView.removeSelectionChangeListener(oListener)  
End Sub
```

```
'Tous les auditeurs doivent supporter cet événement
```

```
Sub MyApp_disposing(oEvent)  
    msgbox "disposing the listener"  
End Sub
```

```
Sub MyApp_selectionChanged(oEvent)  
    Dim oCurrentSelection As Object  
    'La propriété source de la structure d'événement  
    'prend une référence dans la sélection courante  
    oCurrentSelection = oEvent.source  
    MsgBox oCurrentSelection.dbg_properties  
End sub
```

Notez que toutes les méthodes d'auditeurs doivent être implémentées dans votre programme basic car si le service appelant ne trouve pas la routine appropriée, une erreur d'exécution se produit.

Références de l' API concernées :

<http://api.openoffice.org/docs/common/ref/com/sun/star/view/OfficeDocumentView.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/view/XSelectionSupplier.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/view/XSelectionChange.html>

[eListener.html](#)

<http://api.openoffice.org/docs/common/ref/com/sun/star/lang/EventObject.html>

Un objet imprimable (disons un objet document) peut supporter l'interface com.sun.star.view.XPrintJobBroadcaster. Cette interface vous autorise à enregistrer (et à déréférencer) un com.sun.star.view.XPrintJobListener pour intercepter les événements lors de l'impression. Lorsque vous en interceptez, vous obtenez une structure com.sun.star.view.PrintJobEvent structure. Cette structure a généralement une propriété "source" ; la source de cet événement est un Print Job, qui est le service décrivant le travail courant d'impression et qui doit supporter l'interface com.sun.star.view.XPrintJob.

Option Explicit

Global oPrintJobListener As Object

'exécuter cette macro démarre l'interception d'événement

Sub Register_PrintJobListener

```
oPrintJobListener =  
CreateUnoListener("MyApp_", "com.sun.star.view.XPrintJobListener")
```

'Cette fonction est définie dans la bibliothèque "Tools"

'writedbinfo oPrintJobListener

```
ThisComponent.addPrintJobListener(oPrintJobListener)
```

End Sub

'exécuter cette macro stoppe l'interception d'événement

Sub Unregister_PrintJobListener

```
ThisComponent.removePrintJobListener(oPrintJobListener)
```

End Sub

'Tous les auditeurs doivent supporter cet événement

Sub MyApp_disposing(oEvent)

'Rien à faire ici

End sub

'Cet événement est appelé plusieurs fois

'durant l'impression

Sub MyApp_printJobEvent(oEvent)

'la source de l'événement printJob est un PrintJob,

'qui est un service supportant l'interface com.sun.star.view.XPrintJob

'qui est le service décrivant le travail courant d'impression.

MsgBox oEvent.source.Dbg_methods

Select Case oEvent.State

Case com.sun.star.view.PrintableState.JOB_STARTED

Msgbox "L'impression (mise en forme du document) a démarré"

Case com.sun.star.view.PrintableState.JOB_COMPLETED

sMsg = "L'impression (mise en forme du document) "

sMsg = sMsg & "est terminée, le spoulage a démarré"

Msgbox sMsg

Case com.sun.star.view.PrintableState.JOB_SPOOLED

sMsg = "Spoulage terminé avec succès."

```

sMsg = sMsg & " C'est le seul état qui "
sMsg = sMsg & "puisse être considéré comme un 'succès'"
sMsg = sMsg & "pour un travail d'impression"
Msgbox sMsg

Case com.sun.star.view.PrintableState.JOB_ABORTED
sMsg = "L'impression a été annulée (par ex. par l'utilisateur)"
sMsg = sMsg & "lors de l'impression ou du spoulage."
Msgbox sMsg

Case com.sun.star.view.PrintableState.JOB_FAILED
sMsg = "L'impression a rencontré une erreur."
Msgbox sMsg

Case com.sun.star.view.PrintableState.JOB_SPOOLING_FAILED
sMsg = "Le document a pu être imprimé, mais pas spoulé."
Msgbox sMsg

End Select

End sub

Référence de l'API en rapport :
http://api.openoffice.org/docs/common/ref/com/sun/star/document/OfficeDocument.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XPrintJobBroadcaster.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XPrintJobListener.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/PrintJobEvent.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XPrintJob.html

```

Les gestionnaires sont des types particuliers d'auditeurs. Comme les auditeurs, ils peuvent intercepter les événements mais en plus un gestionnaire agit comme un consommateur d'événements, en d'autres termes, un gestionnaire peut "avaler" des événements. A la différence des auditeurs, les méthodes d'un gestionnaire doivent avoir un résultat (booléen) : un résultat Vrai dit au broadcaster que l'événement est consommé par le gestionnaire, ceci fait que le broadcaster n'adressera pas l'événement aux gestionnaires subséquents. Le gestionnaire com.sun.star.awt.XKeyHandler permet l'interception des événements clavier dans un document. L'exemple suivant montre un gestionnaire clavier qui agit en consommateur pour certains événements "touches appuyées" (touche "t", "a", "b", "u") :

```

Option Explicit
Global oDocView
Global oKeyHandler

Sub RegisterKeyHandler
oDocView = ThisComponent.getCurrentController
oKeyHandler =
createUnoListener("MyApp_", "com.sun.star.awt.XKeyHandler")

' writedbginfo oKeyHandler

oDocView.addKeyHandler(oKeyHandler)
End Sub

```

```

Sub UnregisterKeyHandler
    oDocView.removeKeyHandler(oKeyHandler)
End Sub

Sub MyApp_Disposing(oEvt)
    'on ne fait rien ici
End Sub

Function MyApp_KeyPressed(oEvt) As Boolean
    select case oEvt.KeyChar
        case "t", "a", "b", "u"
            MyApp_KeyPressed = True
            msgbox "La touche """ & oEvt.KeyChar & """ n'est pas autorisée !"
        case else
            MyApp_KeyPressed = False
    end select
End Function

Function MyApp_KeyReleased(oEvt) As Boolean
    MyApp_KeyReleased = False
End Function

```

Référence de l'API en rapport :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XUserInputInterface.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XExtendedToolkit.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XKeyHandler.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/KeyEvent.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/Key.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/KeyFunction.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/KeyModifier.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/InputEvent.html>

Ce gestionnaire autorise la capture des clics de souris dans un document. Cet exemple montre une implémentation complète de ce gestionnaire :

```

Option Explicit

Global oDocView As Object
Global oMouseClickHandler As Object

Sub RegisterMouseClickHandler
    oDocView = ThisComponent.currentController
    oMouseClickHandler =
        createUnoListener("MyApp_", "com.sun.star.awt.XMouseClickHandler")

    ' writedbginfo oMouseClickHandler

    oDocView.addMouseClickHandler(oMouseClickHandler)
End Sub

Sub UnregisterMouseClickHandler
    on error resume next
    oDocView.removeMouseClickHandler(oMouseClickHandler)
    on error goto 0
End Sub

Sub MyApp_Disposing(oEvt)

```

```

End Sub

Function MyApp_mousePressed(oEvt) As Boolean
    MyApp_mousePressed = False
End Function

Function MyApp_mouseReleased(oEvt) As Boolean
Dim sMsg As String

    With oEvt
        sMsg = sMsg & "Modifieurs = " & .Modifiers & Chr(10)
        sMsg = sMsg & "Boutons = " & .Buttons & Chr(10)
        sMsg = sMsg & "X = " & .X & Chr(10)
        sMsg = sMsg & "Y = " & .Y & Chr(10)
        sMsg = sMsg & "Nb de clics = " & .ClickCount & Chr(10)
        sMsg = sMsg & "Déclenchement de Popup = " & .PopupTrigger & Chr(10)
        'sMsg = sMsg & .Source.dbg_Methods
    End With

    ThisComponent.text.string = sMsg

    MyApp_mouseReleased = False
End Function

```

Référence de l'API en rapport :

<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XUserInputInterface.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/XMouseClickHandler.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/MouseEvent.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/MouseButton.html>
<http://api.openoffice.org/docs/common/ref/com/sun/star.awt/InputEvent.html>

Normalement, en programmation OOBASIC, vous n'avez pas besoin d'auditeurs parce que vous pouvez lier manuellement un événement à une macro. Par exemple, depuis la boîte de dialogue "Adaptation" (menu "Outils"=>"Adaptation..."), sélectionner l'onglet "événements" vous permettra de lier des événements d'application ou de document. En outre, de nombreux objets pouvant être insérés dans un document ont une boîte de dialogue de propriétés avec un onglet "Événements". Finalement, les boîtes de dialogue OOBASIC et les contrôles font cela aussi bien.

Il est utile de noter que dans la liaison manuelle, le mécanisme sous-jacent est le même qu'avec les auditeurs, toutefois, vous pouvez ajouter des paramètres d'événement à vos macros pour obtenir des informations additionnelles sur l'événement.

Pour exécuter l'événement suivant, ouvrir un nouveau document Writer, ajouter un champ de texte et assigner manuellement la macro à l'événement "touche appuyée" du contrôle. Note : le nom de la macro et celui de l'événement sont arbitraires.

Option Explicit

```

' La macro est assignée manuellement à l'événement "Touche enfoncée"
' du contrôle champ de texte dans le document
Sub MyTextEdit_KeyPressed(oEvt)
Dim sMsg As String

```

```
With oEvt
    sMsg = sMsg & "Modifieurs = " & .Modifiers & Chr(10)
    sMsg = sMsg & "Code touche = " & .KeyCode & Chr(10)
    sMsg = sMsg & "Caractère = " & .KeyChar & Chr(10)
    sMsg = sMsg & "Fonction = " & .KeyFunc & Chr(10)
    sMsg = sMsg & .Source.Dbg_supportedInterfaces
End With

msgbox sMsg
End Sub
```

Commentaires

C'est toujours une bonne méthode de commenter largement votre code. Ce qui est clair aujourd'hui ne le sera pas demain. L'apostrophe simple et le mot-clé REM indiquent tous deux le début d'un commentaire. Tout le texte suivant sera ignoré.

```
REM Ceci est un commentaire  
REM Et voici un autre commentaire  
' Et encore un autre commentaire  
' Je pourrais continuer comme ça toute la journée  
Dim i As Integer      REM i est utilisé comme variable index dans les boucles  
Print i                REM Ceci va imprimer la valeur de i
```

Variables

Les noms de variables sont limités à 255 caractères, ils doivent commencer par une lettre de l'alphabet non accentuée et peuvent contenir des chiffres. Les caractères souligné et espace sont aussi valides. Aucune distinction n'est faite entre les caractères majuscules et minuscules. Les noms de variables contenant des espaces doivent être placés entre crochets “[]”.

Il est recommandé de déclarer vos variables avant de les utiliser. L'instruction “Option Explicit” vous oblige à le faire. Cette ligne doit se trouver dans votre code avant toute autre instruction. Si vous n'utilisez pas “Option Explicit”, des variables mal écrites peuvent générer des bogues parfois difficiles à détecter (erreurs logicielles).

Utilisez Dim pour déclarer une variable. Voici la syntaxe de Dim :

```
[ReDim]Dim Nom1 [(début To fin)] [As Type][, Nom2 [(début To fin)] [As Type]  
[,...]]
```

Cette syntaxe vous permet de déclarer plusieurs variables à la fois. *Nom* est un nom quelconque de variable ou de tableau. Les valeurs *début* et *fin* vont de -32768 à 32767. Elles définissent le nombre d'éléments (inclusivement), de sorte que *Nom1(début)* et *Nom1(fin)* sont tous deux des valeurs valides. Avec ReDim , les valeurs de *début* et *fin* peuvent être des expressions numériques. Les valeurs possibles pour *Type* sont Boolean, Currency, Date, Double, Integer, Long, Object, Single, String, et Variant.

Variant est le type par défaut si aucun type n'est spécifié, à moins que les commandes DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj, ou DefVar ne soient utilisées. Ces commandes vous permettent de définir le type de donnée d'après le premier caractère du nom de la variable.

Les variables String sont des chaînes de caractères, d'une longueur maximale de 64000 caractères.

Les variables Variant peuvent contenir tous les types et sont précisées à la définition.

Les variables Object doivent être suivies d'une instruction d'affectation Set.

L'exemple suivant illustre les problèmes qui peuvent arriver si vous ne déclarez pas vos variables. La variable non-définie "truc" sera par défaut de type Variant. Exécutez cette macro et voyez quels types Basic utilisera pour les variables non déclarées.

```
Sub TestNonDeclare
    print "1 : ", TypeName(truc), truc
    truc= "ab217"
    print "2 : ", TypeName(truc), truc
    truc= true
    print "3 : ", TypeName(truc), truc
    truc= 5=5 ' devrait être un Booléen
    print "4 : ", TypeName(truc), truc
    truc= 123.456
    print "5 : ", TypeName(truc), truc
    truc=123
    print "6 : ", TypeName(truc), truc
    truc= 1217568942 ' Pourrait être un Long
    print "7 : ", TypeName(truc), truc
    truc= 123123123123123.1234 'Devrait être un Currency
    print "8 : ", TypeName(truc), truc
End Sub
```

C'est un argument fort pour explicitement déclarer toutes les variables.

On doit déclarer le type de variables individuellement pour chacune, sinon il sera par défaut de type Variant. "Dim a, b As Integer" déclare "a" comme type Variant et "b" comme type Integer.

```
Sub MultipleDeclaration
    Dim a, b As Integer
    Dim c As Long, d As String
    Dim e As Variant, f As Float
    Print TypeName(a) Rem Empty , Variant par défaut
    Print TypeName(b) Rem Integer, Déclaré Integer
    Print TypeName(c) Rem Long, Déclaré Long
    Print TypeName(d) Rem String, Déclaré String
    Print TypeName(e) Rem Empty, Variant comme déclaré
    Print TypeName(f) Rem Object, car Float est un type inconnu
    Print TypeName(g) Rem Empty (Variant) car NON déclaré
End Sub
```

Les variables globales sont habituellement déconseillées car elles peuvent être modifiées par n'importe quelle routine, n'importe où, et il est difficile de savoir quelle routine modifie quelle variable, si on les utilise. C'est pourquoi j'ai placé le terme "malfaisante" avant le terme "variable globale" lorsque j'enseignais à l'Université d'État de l'Ohio. C'était un moyen de rappeler à mes étudiants que, bien qu'il y ait des occasions pour utiliser des variables globales, ils devaient réfléchir avant de les utiliser.

Une variable globale doit être déclarée en dehors d'une procédure. On peut

utiliser les mots-clés Public et Private pour préciser si la variable est globale à tous les modules ou seulement à ce module. En l'absence de Public ou Private, on suppose Private. La syntaxe est identique aux instructions Dim et ReDim.

Bien que les variables soient passées par référence si non stipulées autrement, les variables globales semblent être passées par valeur. Ce qui a causé au moins un bogue dans mon code.

A chaque appel d'une procédure, les variables locales à une procédure sont recréées. En déclarant la variable Static, elle gardera sa valeur. Dans l'exemple ci-dessous, le sous-programme Worker compte combien de fois il a été appelé. Rappel : les variables numériques sont initialisées à zéro et les chaînes de caractères sont initialisées à une chaîne vide.

```
Option Explicit
Public Auteur As String      REM Global à tous les modules
Private PrivateOne$    REM Global à ce module seulement
Dim PrivateTwo$      REM Global à ce module seulement

Sub PublicPrivateTest
    Author = "Andrew Pitonyak"
    PrivateOne$ = "Hello"
    Worker()
    Worker()
End Sub

Sub Worker()
    Static Counter As Long      REM mémorise sa valeur entre deux appels
    Counter = Counter + 1      REM compte chaque appel de Travailleur
    Print "Compteur = " + Counter
    Print "Auteur = " + Author
End Sub
```

D'un point de vue abstrait, le Basic OpenOffice.org supporte les types de variables numérique, chaîne, booléen, et objet. Les objets sont principalement utilisés pour se référer à des éléments internes comme des documents, des tables, etc... Avec un objet, vous pouvez utiliser les méthodes et les propriétés qui lui sont associées. Les types numériques sont initialisés à zéro et les chaînes de caractères sont initialisées à une chaîne vide "".

Pour connaître à l'exécution le type d'une variable, la fonction TypeName renvoie une chaîne de caractères désignant le type de la variable. La fonction VarType renvoie un entier Integer correspondant au type.

Boolean	Booléen	11		DefBool
Currency	Monétaire avec 4 chiffres après la virgule	6	@	
Date	Date	7		DefDate
Double	Virgule flottante en double précision	5	#	DefDbl

Integer	Entier	2	%	DefInt
Long	Entier de grande valeur	3	&	DefLng
Object	Objet	9		DefObj
Single	Virgule flottante en simple précision	4	!	
String	Chaîne de caractères	8	\$	
Variant	Peut contenir tous les types spécifiés par la définition	12		DefVar
(rien)	Variable non initialisée	0		
Null	Donnée invalide	1		

```

Sub ExampleTypes
    Dim b As Boolean
    Dim c As Currency
    Dim t As Date
    Dim d As Double
    Dim i As Integer
    Dim l As Long
    Dim o As Object
    Dim f As Single
    Dim s As String
    Dim v As Variant
    Dim n As Variant
    Dim x As Variant
    n = null
    x = f
    Print TypeName(b) + " " + VarType(b) Rem Boolean 11
    Print TypeName(c) + " " + VarType(c) Rem Currency 6
    Print TypeName(t) + " " + VarType(t) Rem Date 7
    Print TypeName(d) + " " + VarType(d) Rem Double 5
    Print TypeName(i) + " " + VarType(i) Rem Integer 2
    Print TypeName(l) + " " + VarType(l) Rem Long 3
    Print TypeName(o) + " " + VarType(o) Rem Object 9
    Print TypeName(f) + " " + VarType(f) Rem Single 4
    Print TypeName(s) + " " + VarType(s) Rem String 8
    Print TypeName(v) + " " + VarType(v) Rem Empty 0
    Print TypeName(n) + " " + VarType(n) Rem Null 1
    Print TypeName(x) + " " + VarType(x) Rem Single 4
End Sub

```

Variables booléennes Boolean

Bien que les variables Boolean utilisent les valeurs “True” ou “False” (pour Vrai et Faux respectivement), elles sont représentées en interne par les valeurs entières “-1” et “0” respectivement. Toute valeur numérique différente de 0 affectée à un booléen entraîne la valeur True. Voici des utilisations typiques :

```

Dim b as Boolean
b = True
b = False
b = (5 = 3) 'Affecte la valeur False
Print b      'Affiche 0
b = (5 < 7) 'Affecte la valeur True
Print b      'Affiche -1

```

```
b = 7      'Affecte la valeur True car 7 est différent de 0
```

Variables entières Integer

Les variables Integer sont des nombres entiers à 16 bits sur une plage de valeurs de -32768 à 32767. Si on affecte un nombre flottant à un Integer, la valeur est arrondie à l'entier le plus proche. Rajouter un caractère "%" derrière le nom d'une variable lui donne le type Integer.

```
Sub AssignFloatToInteger
    Dim i1 As Integer, i2%
    Dim f2 As Double
    f2= 3.5
    i1= f2
    Print i1 REM 4
    f2= 3.49
    i1= f2
    Print i1 REM 3
End Sub
```

Variables entières Long

Les variables Long sont des nombres entiers à 32 bits sur une plage de valeurs de -2.147.483.648 à 2.147.483.647. Si on affecte un nombre flottant à un Long, la valeur est arrondie à l'entier le plus proche. Rajouter un caractère "&" derrière le nom d'une variable lui donne le type Long.

```
Dim Age&
Dim Dogs As Long
```

Variables monétaires Currency

Les variables monétaires Currency sont des nombres à 64 bits en virgule fixe, avec 4 décimales et une partie entière de 15 chiffres. Ceci donne une plage de valeurs de -922.337.203.658.477,5808 à +922.337.203.658.477,5807.

Rajouter un caractère "@" derrière le nom d'une variable lui donne le type Currency.

```
Dim Income@
Dim Cost As Currency
```

Variables flottantes Single

Les variables Single sont des nombres à 32 bits en virgule flottante. La plus grande valeur absolue est $3,402823 \times 10^{38}$. La plus petite valeur absolue non nulle est $1,401298 \times 10^{-45}$. Rajouter un caractère "!" derrière le nom d'une variable lui donne le type Single.

```
Dim Weight!
Dim Height As Single
```

Variables flottantes Double

Les variables Double sont des nombres à 64 bits en virgule flottante. La plus grande valeur absolue est $1,79769313486232 \times 10^{308}$. La plus petite valeur absolue non nulle est $4,94065645841247 \times 10^{-324}$. Rajouter un caractère "#" derrière le nom d'une variable lui donne le type Double.

```
Dim Weight#
Dim Height As Double
```

Variables de chaîne de caractères String

Les variables String nécessitent un caractère ASCII à un octet pour chaque caractère, et elles ont une longueur limitée à 64K octets. Rajouter un caractère “\$” derrière le nom d'une variable lui donne le type String.

```
Dim FirstName$  
Dim LastName As String
```

Les deux valeurs spéciales Empty et Null sont à considérer quand on manipule les types Object et Variant. La valeur Empty indique qu'aucune valeur n'a été assignée à la variable. Cela peut être testé avec la fonction IsEmpty(var). Le Null indique qu'aucune valeur valide n'est présente. Cela peut être testé avec la fonction IsNull(var).

Quand une variable de type Object est déclarée, elle contient la valeur Null. Quand une variable de type Variant est déclarée, elle contient la valeur Empty.

```
Sub ExampleObjVar  
    Dim obj As Object, var As Variant  
    Print IsNull(obj)      Rem True  
    Print IsEmpty(obj)    Rem False  
    obj = CreateUnoService("com.sun.star.beans.Introspection")  
    Print IsNull(obj)      Rem False  
    obj = Null  '?? Vérifier sur la 1.0.3.1  
    Print IsNull(obj)      Rem True  
  
    Print IsNull(var)      Rem False  
    Print IsEmpty(var)    Rem True  
    var = obj  
    Print IsNull(var)      Rem True  
    Print IsEmpty(var)    Rem False  
    var = 1  
    Print IsNull(var)      Rem False  
    Print IsEmpty(var)    Rem False  
    var = Empty            Rem IsEmpty(Empty) échoue avec la 1.0.3.1 mais marche en 1.1 beta  
    Print IsEmpty(var)    Rem True  
End Sub
```

En écrivant du code qui interagit avec les objets basic UNO, il faut décider quel type de variable utiliser. La plupart des exemples utilisent le type Object. Cependant, le guide du développeur, en page 132, suggère une autre déclaration.

Toujours utiliser le type variant pour les variables d'objets UNO et pas le type Object. Le type Object de OOBasic est conçu pour les objets OOBasic purs et pas pour UNO. Les variables Variant sont mieux pour les objets UNO, évitant des problèmes pouvant intervenir lors de comportements spécifiques au type object d'OOBasic.

```
Dim oService1 ' Ok  
oService1 = CreateUnoService( "com.sun.star.anywhere.Something" )  
Dim oService2 as Object ' NON recommandé  
oService2 = CreateUnoService( "com.sun.star.anywhere.SomethingElse" )
```

Andreas Bregas ajoute que dans la plupart des cas, les 2 méthodes

fonctionnent. Le guide du développeur préconise le type Variant car dans certains rares cas, l'utilisation du Type Objet conduit à une erreur à cause de la sémantique du type Object de l'ancien Basic. Mais si un programme en Basic utilise le type Objet et fonctionne correctement comme cela, il n'y a pas de problème.

Le Basic OpenOffice.org reconnaît les valeurs "True"(vrai), "False" (faux), et "PI". Vous pouvez aussi définir vos propres constantes. On ne peut définir qu'une seule fois une constante. Les constantes n'ont pas de type, elles sont simplement insérées comme si on les avait tapées.

```
Const Gravity = 9.81
```

Un tableau vous permet de mettre de nombreuses valeurs différentes dans une seule variable. Par défaut, le premier élément est en position zéro. Mais vous pouvez préciser les positions de début et de fin. Voici quelques exemples :

```
Dim a(5) As Integer           REM 6 éléments de 0 à 5 inclus
Dim b$(5 to 10) As String     REM 6 éléments de 5 à 10 inclus
Dim c(-5 to 5) As String      REM 11 éléments de -5 à 5 inclus
Dim d(5 To 10, 20 To 25) As Long REM Tableau à deux dimensions
```

Si vous avez un tableau de Variant et que vous voulez le remplir rapidement, utilisez la fonction Array. Cela retournera un tableau de Variant avec des données incluses. C'est comme cela que j'établis une liste de données.

```
Sub ExampleArray
    Dim a(), i%
    a = Array(0, 1, 2)
    a = Array("Zero", 1, 2.0, Now)
    REM String, Integer, Double, Date
    For i = LBound(a()) To UBound(a())
        Print TypeName(a(i))
    Next
End Sub
```

Option Base

Vous pouvez changer la valeur par défaut de la position basse, à 1 au lieu de zéro. Ceci doit être indiqué avant toute autre instruction exécutable du programme.

```
Option Base { 0 | 1 }
```

LBound(NomDeTableau,[Dimension])

Renvoie la position basse du tableau. Le second paramètre optionnel est le numéro de la dimension du tableau pour laquelle vous désirez connaître la position basse ; ce numéro est compté à partir de 1 (et non pas zéro).

```
LBound(a())    REM 0
LBound(b$())   REM 5
LBound(c())    REM -5
LBound(d())    REM 5
```

```
LBound(d(), 1) REM 5  
LBound(d(), 2) REM 20
```

UBound(NomDeTableau[Dimension])

Renvoie la position haute du tableau. Le second paramètre optionnel est le numéro de la dimension du tableau pour laquelle vous désirez connaître la position haute ; ce numéro est compté à partir de 1 (et non pas zéro).

```
UBound(a()) REM 5  
UBound(b$()) REM 10  
UBound(c()) REM 5  
UBound(d()) REM 10  
UBound(d(), 1) REM 10  
UBound(d(), 2) REM 25
```

Ce tableau est-il défini ?

Si un tableau est en réalité une liste vide, la position basse Lbound aura une valeur supérieure à la position haute Ubound.

La fonction DimArray sert à définir ou à modifier le nombre de dimensions d'un tableau de Variant. DimArray(2, 2, 4) est équivalent à DIM a(2, 2, 4).

```
Sub ExampleDimArray  
Dim a(), i%  
a = Array(0, 1, 2)  
Print "" & LBound(a()) & " " & UBound(a()) REM 0 2  
a = DimArray()  
' Empty array  
i = 4  
a = DimArray(3, i)  
Print "" & LBound(a(),1) & " " & UBound(a(),1) REM 0, 3  
Print "" & LBound(a(),2) & " " & UBound(a(),2) REM 0, 4  
End Sub
```

L'instruction ReDim est utilisée pour changer la taille d'un tableau.

```
Dim e() As Integer REM taille non spécifiée  
ReDim e(5) As Integer REM positions 0 à 5 valides  
ReDim e(10) As Integer REM positions 0 à 10 valides
```

Le mot-clé Preserve peut être utilisé avec l'instruction ReDim pour préserver le contenu du tableau quand il est redimensionné.

```
Sub ReDimExample  
Dim a(5) As Integer  
Dim b()  
Dim c() As Integer  
a(0) = 0  
a(1) = 1  
a(2) = 2  
a(3) = 3
```

```

a(4) = 4
a(5) = 5
Rem a est dimensionné de 0 à 5 avec a(i) = i
PrintArray("a au début", a())
Rem a est dimensionné de 1 à 3 avec a(i) = i
ReDim Preserve a(1 To 3) As Integer
PrintArray("a après ReDim", a())
Rem Array() renvoie un type variant
Rem b est dimensionné de 0 à 9 avec b(i) = i+1
b = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
PrintArray("b à l'affectation initiale", b())
Rem b est dimensionné de 1 à 3 avec b(i) = i+1
ReDim Preserve b(1 To 3)
PrintArray("b après ReDim", b())
Rem Ce qui suit est NON valide car le tableau est déjà
Rem dimensionné à une taille différente
Rem a = Array(0, 1, 2, 3, 4, 5)

Rem c est dimensionné de 0 à 5 avec a(i) = i
Rem Si un "ReDim" avait été fait sur c, cela ne marcherait pas
c = Array(0, 1, 2, 3, 4, 5)
PrintArray("c, de type Integer après affectation", c())
Rem Curieusement, ceci est autorisé mais c ne contiendra aucune donnée !
ReDim Preserve c(1 To 3) As Integer
PrintArray("c après ReDim", c())
End Sub

Sub PrintArray (lead$, a() As Variant)
    Dim i%, s$
    s$ = lead$ + Chr(13) + LBound(a()) + " to " + UBound(a()) + ":" + Chr(13)
    For i% = LBound(a()) To UBound(a())
        s$ = s$ + a(i%) + " "
    Next
    MsgBox s$
End Sub

```

La fonction Array mentionnée plus haut permet seulement de créer un tableau de Variant. Pour initialiser un tableau de type connu, vous pouvez utiliser la méthode suivante :

```

Sub ExampleSetIntArray
    Dim iA() As Integer
    SetIntArray(iA, Array(9, 8, 7, 6))
    PrintArray("", iA)
End Sub

Sub SetIntArray(iArray() As Integer, v() As Variant)
    Dim i As Long
    ReDim iArray(LBound(v()) To UBound(v())) As Integer
    For i = LBound(v) To UBound(v)
        iArray(i) = v(i)
    Next
End Sub

```

Pour déterminer le type d'une variable, on peut utiliser les fonctions booléennes IsArray, IsDate, IsEmpty, IsMissing, IsNull, IsNumeric, IsObject et IsUnoStruct. La fonction IsArray renvoie True si le paramètre est un tableau. La fonction IsDate renvoie True s'il est possible de convertir l'objet en une Date. Une chaîne de caractères contenant une date correctement formatée renverra donc True pour la fonction IsDate. La fonction IsEmpty sert à tester si un objet

de type Variant a été initialisé. La fonction IsMissing indique si un paramètre Optional est manquant. La fonction IsNull teste si un Variant contient la valeur spéciale Null, qui indique que la variable ne contient pas de donnée. La fonction IsNumeric sert à tester si une chaîne de caractères contient une valeur numérique. La fonction IsUnoStruct analyse la chaîne de caractères d'un nom de structure Uno et renvoie True si c'est un nom valide.

Les opérateurs de comparaison fonctionnent en général comme on s'y attend, mais ils n'effectuent pas une évaluation optimisée. On utilise les opérateurs de comparaison suivants :

=	Egal à
<	Inférieur à
>	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à
<>	Different de
Is	Est le même objet que

L'opérateur And réalise une opération logique sur un type Boolean et une opération bit à bit sur les types numériques. L'opérateur OR réalise une opération logique sur un type Boolean et une opération bit à bit sur les types numériques. L'opérateur XOR réalise une opération logique sur un type Boolean et une opération bit à bit sur les types numériques. Se rappeler qu'il s'agit du "OU exclusif". L'opérateur NOT réalise une opération logique de négation sur un type Boolean et une opération bit à bit sur les types numériques. Un simple test montre que les règles de priorité standard existent, à savoir que AND a une plus grande priorité que les opérateurs OR.

```
Option Explicit
Sub ConditionTest
    Dim msg As String
    msg = "AND possède une priorité "
    msg = msg & IIf(False OR True AND False, "égale", "supérieure")
    msg = msg & " à OR" & Chr(13) & "OR possède une priorité "
    msg = msg + IIF(True XOR True OR True, "supérieure", "inférieure ou égale ")
    msg = msg + "à XOR" + Chr(13)
    msg = msg & "XOR possède une priorité "
    msg = msg + IIF(True OR True XOR True, "supérieure", "inférieure ou égale ")
    msg = msg + "à OR"
    MsgBox msg
End Sub
```

Fonctions et Sous-programmes

Une fonction (Function) est un sous-programme (Sub) qui renvoie une valeur. Ce qui permet de l'utiliser dans une expression. Les sous-programmes et fonctions débutent ainsi :

```

    : Function NomDeFonction[(Var1 [As Type][, Var2 [As
Type][,...]])] [As Type]
        : Sub NomDeSousProg[(Var1 [As Type][, Var2 [As Type]
[,...]])]

```

Les fonctions déclarent un type de valeur de renvoi car elles renvoient une valeur. Pour affecter la valeur de renvoi, utilisez une instruction de la forme “NomDeFonction= valeur_renvoi”. Vous pouvez réaliser plusieurs fois cette affectation mais seule la dernière sera renvoyée.

Utilisez une instruction Exit pour quitter immédiatement la procédure.

Un paramètre peut être déclaré optionnel avec le mot-clé Optional. La fonction IsMissing est alors utilisée pour savoir si un paramètre a été passé.

```

Sub testOptionalParameters()
Print TestOpt()    REM MMM
Print TestOpt(,)   REM MMM
Print TestOpt(,,)  REM MMM
Print TestOpt(1)   REM 1MM
Print TestOpt(1,)  REM 1MM
Print TestOpt(1,,) REM 1MM
Print TestOpt(1,2) REM 12M
Print TestOpt(1,2,) REM 12M
Print TestOpt(1,2,3) REM 123
Print TestOpt(1,,3) REM 1M3
Print TestOpt(,2,3) REM M23
Print TestOpt(,,3) REM MM3
Print TestOptl()   REM MMM
Print TestOptl(,)  REM 488MM (Error)
Print TestOptl(,,) REM 488488M (Error)
Print TestOptl(1)  REM 1MM
Print TestOptl(1,)  REM 1MM
Print TestOptl(1,,) REM 1488M (Error)
Print TestOptl(1,2) REM 12M
Print TestOptl(1,2,) REM 12M
Print TestOptl(1,2,3)REM 123
Print TestOptl(1,,3) REM 14883 (Error)
Print TestOptl(,2,3) REM 48823 (Error)
Print TestOptl(,,3) REM 4884883 (Error)
End Sub
Function TestOpt(Optional v1 As Variant, Optional v2 As Variant, Optional v3 As Variant) As String
Dim s As String
s = "" & IIF(IsMissing(v1), "M", Str(v1))
s = s & IIF(IsMissing(v2), "M", Str(v2))
s = s & IIF(IsMissing(v3), "M", Str(v3))
TestOpt = s
End Function
Function TestOptl(Optional i1 As Integer, Optional i2 As Integer, Optional i3 As Integer) As String
Dim s As String
s = "" & IIF(IsMissing(i1), "M", Str(i1))
s = s & IIF(IsMissing(i2), "M", Str(i2))
s = s & IIF(IsMissing(i3), "M", Str(i3))
TestOptl = s
End Function

```

Sur la version 1.0.3.1, IsMissing ne fonctionne pas avec les paramètres Optional si le type n'est pas Variant et que le paramètre optionnel manquant est représenté par deux virgules consécutives. J'ai commencé à étudier ce comportement après discussion avec Christian Anderson [ca@ofs.no]. Voir le rapport 11678 dans Issuezilla.

Avec une variable passée par valeur, je peux modifier la valeur du paramètre dans la procédure appelée et la variable originale ne changera pas. Par contre si la variable est passée par référence, alors en changeant la valeur du paramètre je change la variable originale. Le comportement par défaut est de passer par référence. Pour passer par valeur on doit utiliser le mot-clé ByVal avant la déclaration de paramètre. Si le paramètre est une constante, comme "4", et que vous le modifiez dans la procédure appelée, il peut ou non changer. Selon Andreas Bregas (ab@openoffice.org) c'est un bogue, que j'ai donc déclaré dans Issuezilla :

(http://www.openoffice.org/project/www/issues/show_bug.cgi?id=12272).

```
Option Explicit
Sub LoopForever
    Dim I As Long
    I = 4
    LoopWorker(I)
    Print "I est passé par valeur et il est encore égal à " + I
    LoopForeverWorker(I)
    ' I vaut maintenant 1 donc on va afficher 1.
    Print "I est passé par référence et il vaut maintenant " + I
    ' Ceci va boucler à l'infini car 4 est une constante
    ' et on ne peut PAS la changer.
    Print "Passer par référence un paramètre constant, pour rire"
    Print LoopForeverWorker(4)
End Sub

Sub LoopWorker(ByVal n As Long)
    Do While n > 1
        Print n
        n = n - 1
    Loop
End Sub

Sub LoopForeverWorker(n As Long)
    Do While n > 1
        ' Ceci est amusant si n est une constante.
        Print n
        n = n - 1
    Loop
End Sub
```

Vos fonctions ne peuvent pas être récursives (c. à d. s'appeler elles-mêmes). Quand je dis "ne peuvent pas", je devrais dire que vous ne devriez pas car vous n'obtiendrez pas les résultats que vous attendez. Il est prévu que cela change avec la version 1.1 (*NdT : confirmé*).

```
Option Explicit
```

```

Sub DoFact
    Print "Recursive = " + RecursiveFactorial(4)
    Print "Normal Factorial = " + Factorial(4)
End Sub

Function Factorial(n As Long) As Long
    Dim answer As Long
    Dim i As Long
    i = n
    answer = 1
    Do While i > 1
        answer = answer * i
        i = i - 1
    Loop
    Factorial = answer
End Function

' Ceci ne va pas marcher parce que vous ne pouvez pas utiliser la récursivité
Function RecursiveFactorial(n As Long) As Long
    If n > 2 Then
        RecursiveFactorial = n * RecursiveFactorial(n-1)
    Else
        RecursiveFactorial = 1
    End If
End Function

```

Contrôle du déroulement

On utilise la construction If pour exécuter un bloc de code en fonction de la valeur d'une expression booléenne. Bien qu'on puisse utiliser GoTo ou GoSub pour sortir d'un bloc If, on ne peut pas se brancher dans un bloc If.

```

If condition=vraie Then
    bloc d'instructions
[Elself condition=vraie Then]
    bloc d'instructions
[Else]
    bloc d'instructions
End If

```

If condition=vraie Then instruction

```

If x < 0 Then
    MsgBox "Le nombre est négatif"
Elseif x > 0 Then
    MsgBox " Le nombre est positif"
Else
    MsgBox "Le nombre est zéro"
End If

```

On utilise la construction IIF pour renvoyer une expression en fonction d'une condition. Ceci est similaire à la syntaxe du "?" en langage C.

IIf (Condition, ExpressionSiVrai, ExpressionSiFaux)

Cette fonctionnalité est très similaire à ce code :

```
If (Condition) Then  
    objet = ExpressionSiVrai  
Else  
    objet = ExpressionSiFaux  
End If  
  
max_age = IIf(Age_Jean > Age_Bernard, Age_Jean, Age_Bernard)
```

L'instruction Choose permet une sélection dans une liste selon un index.

Choose (Index, Selection1[, Selection2, ... [,Selection_n]])

Si l'index vaut 1, l'élément Selection1 est renvoyé. Si l'index vaut 2, l'élément Selection2 est renvoyé. Vous pouvez deviner la suite !

Cette structure répète un bloc d'instructions un nombre donné de fois.

```
For compteur = début To fin [Step incrément]  
    bloc d'instructions  
    [Exit For]  
    bloc d'instructions  
Next [compteur]
```

La variable numérique “compteur” est initialisée à la valeur “départ”. S'il n'y a pas de valeur “incrément”, le compteur est incrémenté de 1 jusqu'à atteindre la valeur “fin”. Si une valeur “incrément” est fournie, alors “incrément” est ajouté au compteur jusqu'à atteindre la valeur “fin”. Le bloc d'instructions est exécuté une fois à chaque incrémantation.

Le “compteur” est optionnel sur l'instruction “Next” et réfère automatiquement à l'instruction “For” la plus récente.

On peut quitter prématurément une boucle “For” avec l'instruction “Exit For”. Elle termine la boucle “For” la plus récente.

L'exemple suivant emploie deux boucles imbriquées pour trier un tableau d'entiers de 10 éléments (iValeurs()), qu'on a au préalable rempli avec un contenu varié :

```
Sub ForNextExampleSort  
    Dim iEntry(10) As Integer  
    Dim iCount As Integer, iCount2 As Integer, iTemp As Integer  
    Dim bSomethingChanged As Boolean  
  
    ' Remplir le tableau avec des entiers entre -10 et 10  
    For iCount = LBound(iEntry()) To Ubound(iEntry())  
        iEntry(iCount) = Int((20 * Rnd) - 10)
```

```

Next iCount
'Tri du tableau
For iCount = LBound(iEntry()) To Ubound(iEntry())
    'Supposons le tableau trié
    bSomethingChanged = False
    For iCount2 = iCount + 1 To Ubound(iEntry())
        If iEntry(iCount) > iEntry(iCount2) Then
            iTemp = iEntry(iCount)
            iEntry(iCount) = iEntry(iCount2)
            iEntry(iCount2) = iTemp
            bSomethingChanged = True
        End If
    Next iCount2
    ' Si le tableau est déjà trié, arrêtons la boucle !
    If Not bSomethingChanged Then Exit For
Next iCount
For iCount = 1 To 10
    Print iEntry(iCount)
Next iCount
End Sub

```

L'aide en ligne contient une excellente et complète description , lisez là.

La structure de boucle Do a différentes formes et sert à répéter l'exécution d'un bloc de code tant qu'une condition est vraie. La forme la plus courante teste la condition avant le début de la boucle et répétera l'exécution du bloc d'instructions tant que la condition reste vraie. Si la condition est fausse au départ, la boucle ne sera jamais exécutée.

```

Do While condition
    Bloc d'instructions
Loop

```

Une forme similaire mais bien moins courante vérifie la condition avant le départ de la boucle et répétera l'exécution du bloc d'instructions tant que la condition reste fausse. Si la condition est vraie au départ, la boucle ne sera jamais exécutée.

```

Do Until condition
    Bloc d'instructions
Loop

```

On peut aussi placer le test en fin de boucle, dans ce cas le bloc d'instructions sera toujours exécuté au moins une fois. Pour exécuter toujours au moins une fois la boucle et répéter tant que la condition reste vraie, employer la structure suivante :

```

Do
    Bloc d'instructions
Loop While condition

```

Pour exécuter toujours au moins une fois la boucle et répéter tant que la condition reste fausse, employer la structure suivante :

```

Do
    Bloc d'instructions
Loop Until condition

```

On peut forcer la sortie d'une boucle Do avec l'instruction "Exit Do".

L'instruction Select Case est similaire aux instructions "case" et "switch" dans d'autres langages. Elle émule de multiples blocs "Else If" dans une instruction "If". On spécifie une unique expression conditionnelle, qui est comparée à plusieurs valeurs pour chercher une égalité comme suit :

```
Select Case variable
  Case expression1
    bloc d'instructions 1
  Case expression2
    bloc d'instructions 2
  Case Else
    bloc d'instructions 3
End Select
```

La valeur de la variable est comparée dans chaque instruction Case. Je ne connais pas de limitation de type autre que la compatibilité entre le type de la variable et le type de l'expression. Le premier bloc d'instruction correspondant est exécuté. Si aucune condition ne correspond, le bloc optionnel Case Else est exécuté.

Expressions Case

Une expression Case est habituellement une constante comme "Case 4" ou "Case Hello". On peut indiquer des valeurs multiples en les séparant par des virgules : "Case 3, 5, 7". Pour tester une plage de valeurs, il existe le mot-clé "To", exemple "Case 5 To 10". On peut tester des séries ouvertes de valeurs avec "Case < 10" ou avec le mot-clé "Is", exemple "Case Is < 10".

Faites attention quand vous utilisez une plage de valeurs dans une instruction Case. L'aide en ligne contient souvent des exemples incorrects, comme "Case i > 2 AND i < 10". C'est difficile à comprendre et à programmer correctement.

Exemple incorrect simple

J'ai vu beaucoup d'exemples incorrects, aussi je vais prendre le temps de vous montrer quelques choses qui ne fonctionnent pas. Je commencerai par un exemple très simple :

i = 2 Select Case i Case 2	i = 2 Select Case i Case Is = 2	i = 2 Select Case i Case i = 2

L'exemple incorrect échoue car " Case i = 2" est la réduction de "Case Is = (i = 2)". L'expression (i=2) est évaluée comme True, soit -1. Cela revient à évaluer l'expression " Case Is = -1" dans cet exemple.

Si vous comprenez cet exemple incorrect simple, alors vous êtes prêt des exemples difficiles.

Exemple incorrect avec une plage

L'exemple suivant est dans l'aide en ligne.

Case Is > 8 AND iVar < 11

Cela ne fonctionne pas car c'est évalué comme :

Case Is > (8 AND (iVar < 11))

L'expression (*iVar*<11) est évaluée comme True ou False. Souvenez vous que True=-1 et False=0. L'opérateur AND est appliqué, bit à bit, entre 8 et -1 (True) ou 0 (False), avec comme résultat soit 8 soit 0. L'expression est donc réduite à une des 2 expressions.

Si *iVar* est plus petite que 11 :

Case Is > 8

Si *iVar* est supérieure ou égale à 11 :

Case Is > 0

Exemple incorrect avec une plage

J'ai aussi vu cet exemple incorrect.

Case i > 2 AND i < 10

Ceci ne fonctionne pas car c'est évalué comme :

Case Is = (i > 2 AND i < 10)

Les plages, La Voie Correcte

L'expression

Case Expression

est probablement correcte si elle peut être écrite

Case Is = (Expression)

Ma solution initiale était :

Case iif(Boolean Expression, i, i+1)

J'étais fier de moi jusqu'à ce que Bernard Marcellly me donne la brillante solution suivante :

Case i XOR NOT (Boolean Expression)

Après ma confusion initiale, j'ai réalisé combien c'était réellement brillant. Ne pas tenter de simplifier à la réduction évidente "i AND ()" car cela échoue si i = 0. J'ai fait cette erreur.

```
Sub DemoSelectCase
    Dim i%
    i = Int((15 * Rnd) -2)
    Select Case i%
        Case 1 To 5
            Print "Nombre entre 1 et 5"
        Case 6, 7, 8
            Print " Nombre entre 6 et 8"
        Case IIf(i > 8 AND i < 11, i, i+1)
            Print "Supérieur à 8"
        Case i% XOR NOT(i% > 8 AND i% < 11 )
            Print i%, "Nombre entre 9 et 10"
        Case Else
            Print "En dehors de la plage 1 à 10"
    End Select
```

```
End Sub
```

Il n'y a rien de spécial dans la structure While...Wend, qui a la forme suivante :

```
While Condition  
    Bloc d'instructions  
Wend
```

Cette structure a des limitations qui n'existent pas dans la structure Do While...Loop et n'apporte aucun avantage particulier. On ne peut pas utiliser l'instruction Exit, ni sortir par un GoTo.

L'instruction GoSub provoque un saut à une étiquette de sous-programme définie dans le sous-programme en cours. Quand l'instruction Return est atteinte, l'exécution continue à partir du point d'appel initial. Si une instruction Return est rencontrée sans qu'un Gosub n'ait été préalablement effectué, une erreur est générée. Autrement dit, Return n'est pas un équivalent de Exit Sub ni de l'instruction Exit. On estime généralement que l'utilisation de fonctions et de sous-programmes produit un code plus compréhensible que l'utilisation de GoSub et GoTo.

```
Option Explicit  
Sub ExampleGoSub  
    Dim i As Integer  
    GoSub Line2  
    GoSub Line1  
    MsgBox "i = " + i, 0, "Exemple de GoSub"  
    Exit Sub  
Line1:  
    i = i + 1  
    Return  
Line2:  
    i = 1  
    Return  
End Sub
```

GoSub est un vestige persistant des vieux dialectes BASIC, gardé par souci de compatibilité. GoSub est fortement déconseillé parce qu'il tend à produire un code illisible. L'utilisation de sous-programmes et fonctions est préférable.

L'instruction GoTo provoque un saut à une étiquette définie dans le sous-programme en cours. On ne peut pas sauter à l'extérieur du sous-programme en cours.

```
Sub ExampleGoTo  
    Dim i As Integer  
    GoTo Line2  
Line1:  
    i = i + 1  
    GoTo TheEnd  
Line2:  
    i = 1
```

```

GoTo Line1
TheEnd:
    MsgBox "i = " + i, 0, "Exemple de GoTo"
End Sub

```

GoTo est un vestige persistant des vieux dialectes BASIC, gardé par souci de compatibilité. GoTo est fortement déconseillé parce qu'il tend à produire un code illisible. L'utilisation de sous-programmes et fonctions est préférable.

```

On N GoSub Etiquette1[, Etiquette2[, Etiquette3[,...]]]
On N GoTo Etiquette1[, Etiquette2[, Etiquette3[,...]]]

```

Ceci fait sauter l'exécution à une étiquette selon la valeur de l'expression numérique *N*. Il n'y a pas de saut si *N* vaut zéro. L'expression numérique *N* doit être dans la plage de valeurs 0 à 255. Ceci est similaire aux instructions "computed goto", "case" et "switch" d'autres langages. Ne pas essayer de sauter à l'extérieur du sous-programme ou de la fonction en cours.

```

Option Explicit
Sub ExampleOnGoTo
    Dim i As Integer
    Dim s As String
    i = 1
    On i+1 GoSub Sub1, Sub2
    s = s & Chr(13)
    On i GoTo Line1, Line2
    REM Cet Exit provoque la sortie si on ne continue pas l'exécution
    Exit Sub
    Sub1:
        s = s & "Dans Sub 1" : Return
    Sub2:
        s = s & "Dans Sub 2" : Return
    Line1:
        s = s & "A Ligne1" : GoTo TheEnd
    Line2:
        s = s & "A Ligne2"
    TheEnd:
        MsgBox s, 0, "Exemple de On GoTo"
End Sub

```

L'instruction Exit permet de sortir d'une boucle Do Loop ou For Next, d'une Function ou d'un Sub. L'utilisation de l'instruction Exit doit apparaître dans les structures du code qu'elle est censée contrôler sous peine de générer une erreur (par exemple, l'instruction Exit For ne peut être utilisée qu'à l'intérieur d'une boucle For). Les différentes formes sont les suivantes :

- Exit DO Continue l'exécution après la prochaine instruction Loop.
- Exit For Continue l'exécution après la prochaine instruction Next.
- Exit Function Sort immédiatement de la fonction en cours.
- Exit Sub Sort immédiatement de la Sub en cours .

```

Option Explicit
Sub ExitExample
    Dim a%(100)

```

```

Dim i%
REM Remplir le tableau avec 100, 99, 98, ..., 0
For i = LBound(a()) To UBound(a())
    a(i) = 100 - i
Next i
Print SearchIntegerArray(a(), 0 )
Print SearchIntegerArray(a(), 10 )
Print SearchIntegerArray(a(), 100)
Print SearchIntegerArray(a(), 200)
End Sub

Function SearchIntegerArray( list(), num%) As Integer
    Dim i As Integer
    SearchIntegerArray = -1
    For i = LBound(list) To UBound(list)
        If list(i) = num Then
            SearchIntegerArray = i
            Exit For
        End If
    Next i
End Function

```

Vos macros peuvent rencontrer plusieurs types d'erreurs. Certaines erreurs sont à gérer (comme un fichier manquant) et d'autres simplement à ignorer. Les erreurs dans les macros sont traitées par l'instruction :

On [Local] {Error GoTo Labelname | GoTo 0 | Resume Next}

On Error permet de spécifier comment les erreurs doivent être gérées, avec la possibilité de définir votre propre gestionnaire d'erreur. Si "Local" est utilisé, la gestion d'erreur n'est active que pour la procédure ou fonction courante, sinon elle s'applique au module entier.

Une procédure peut contenir plusieurs gestions d'erreurs.
Chaque On Error peut traiter les erreurs différemment (l'aide en ligne est fausse quand elle dit qu'une gestion d'erreurs doit apparaître en début de procédure).

Spécifier comment gérer une erreur

Pour ignorer toutes les erreurs, utiliser "On Error Resume Next". Quand une erreur apparaît, cette commande impliquera qu'elle sera ignorée et l'instruction suivante sera exécutée.

Pour spécifier votre propre gestionnaire d'erreur, utiliser "On Error GoTo Label". Pour définir un Label dans OOBasic, taper du texte sur une ligne seule suivi de deux-points. Les labels doivent être uniques. Quand une erreur sera générée, l'exécution de la macro sera transférée à la position du label.

Vous pouvez annuler une gestion d'erreur en utilisant "On Error GoTo 0". Quand une erreur apparaîtra votre gestionnaire d'erreur ne sera plus appelé. Ceci est différent de "On Error Resume Next". Cela implique qu'à la prochaine erreur rencontrée, OOBasic stoppera son exécution comme cela est fait par défaut (arrêt de l'exécution de la macro avec un message d'erreur).

Écrire le gestionnaire d'erreur

Quand une erreur apparaît et que l'exécution est transférée à votre gestionnaire, voici quelques fonctions utiles pour déterminer ce qui s'est passé et où.

- Error([num]) : Renvoie le message d'erreur en tant que chaîne de caractères. Vous pouvez, en option, indiquer un numéro d'erreur spécifique pour récupérer sa signification. Ces textes sont localisés.
- Err() : Retourne le numéro de la dernière erreur.
- Erl() : Indique le numéro de ligne de la dernière erreur.

Une fois l'erreur gérée, il faut décider comment continuer.

- Rien et laisser l'exécution se poursuivre.
- Sortir de la fonction ou du sous-programme en utilisant "Exit Sub" ou "Exit Function".
- Utiliser "Resume" pour exécuter à nouveau la même ligne. Prudence avec ça ! Si la gestion d'erreur n'a pas corrigé l'erreur vous entrerez dans une boucle infinie.

```
Sub ExampleResume
    Dim x%, y%
    x = 4 : y = 0
    On Local Error Goto oopsy
    x = x / y
    Print x
    Exit Sub
oops:
    y = 2
    Resume
End Sub
```

- Utiliser "Resume Next" pour poursuivre l'exécution à la ligne suivant celle qui a généré l'erreur.

```
Sub ExampleResumeNext
    Dim x%, y%
    x = 4 : y = 0
    On Local Error Goto oopsy
    x = x / y
    Print x
    Exit Sub
oops:
    x = 7
    Resume Next
End Sub
```

- Utiliser "Resume Label:" pour poursuivre l'exécution à un label spécifique.

```
Sub ExampleResumeLabel
    Dim x%, y%
    x = 4 : y = 0
    On Local Error Goto oopsy
    x = x / y
GoHere:
    Print x
    Exit Sub
oops:
    x = 7
```

```
Resume GoHere:  
End Sub
```

Un exemple

La macro suivante illustre quelques techniques excellentes de gestion des erreurs :

```
*****  
'Auteur :    Bernard Marcellly  
'email :    marcellly@club-internet.fr  
Sub ErrorHandlingExample  
    Dim v1 As Double  
    Dim v2 As Double  
    Dim v0 As Double  
  
    On Error GoTo TreatError1  
    v0 = 0 : v1= 45 : v2= -123 ' initialisations  
    v2= v1 / v0 ' division par 0 => erreur  
    Print "Result1:", v2  
  
    On Error Goto TreatError2 ' change le gestionnaire d'erreur  
    v2= 456 ' initialisation  
    v2= v1 / v0 ' division par 0 => erreur  
    Print "Result2:", v2 ' ne sera pas executé  
  
Label2:  
    Print "Result3:", v2 ' atteint par la gestion d'erreur  
  
    On Error Resume Next ' ignore toute erreur  
    v2= 963 ' initialisation  
    v2= v1 / v0 ' division par 0 => erreur  
    Print "Result4:", v2 ' sera executé  
  
    On Error Goto 0 ' désactive la gestion d'erreur en cours  
    rem La gestion standard est désormais active  
    v2= 147 ' initialisation  
    v2= v1 / v0 ' division par 0 => erreur  
    Print "Result5:", v2 ' ne sera pas executé  
    Exit Sub  
  
TreatError1:  
    Print "TreatError1 : ", error  
    v2= 0  
    Resume Next ' continue après l'instruction en erreur  
  
TreatError2:  
    Print "TreatError2 : line ", erl, "error number", err  
    v2= 123456789  
    Resume Label2  
End Sub
```

Divers

Ce chapitre contient diverses choses que je connais seulement à travers des exemples, mais dont je n'ai pas trouvé l'utilité.

On peut mettre plusieurs instructions sur la même ligne en les séparant par un “ : ” (deux-points).

Avec une instruction sur une ligne, la structure If Then n'a pas besoin d'être

fermée par un End If.

```
Sub SimpleIf
    If 4 = 4 Then Print "4 = 4" : Print "Hello"      Rem Va s'afficher
    If 3 = 2 Then Print "3 = 2"                      Rem Ne va pas s'afficher parce que 3 <> 2
End Sub
```

Librairies, dialogues, IDE, Import et Export de Macros.

With object ... End With

Comment démarrer à partir de la ligne de commande ?

Le paramètre de lancement de macro s'écrit :

```
soffice.exe macro:/library module macro
```

```
soffice.exe macro:///standard.module1.macro1
```

Mais attention ! Si la macro ne fait rien ou n'ouvre rien dans le document, la macro est exécutée puis OpenOffice.org est fermé.

La copie d'un Object copie seulement la référence. La copie d'une structure réalise une nouvelle copie. Voir EqualUnoObjects pour un exemple.?? Ceci peut poser un problème et alors l'objet devra être recopié!

OpenOffice.org Basic supporte les opérateurs numériques de base -, +, /, *, et ^. Les opérateurs suivent l'ordre de priorité standard mais je l'indique ici aussi. Les opérateurs logiques renvoient 0 pour faux (pas de bits positionnés) et -1 pour vrai (tous les bits sont positionnés). Pour une information plus complète, voyez la section listant opérateurs et fonctions.

0	AND	Bit à bit pour les numériques et logique pour les Booléens
0	OR	Bit à bit pour les numériques et logique pour les Booléens

0	XOR	Bit à bit pour les numériques et logique pour les Booléens
0	EQV	Équivalence logique et/ou au niveau du bit
0	IMP	Implication logique (bogué depuis 1.0.3.1)
1	=	Logique
1	<	Logique
1	>	Logique
1	<=	Logique
1	>=	Logique
1	<>	Logique
2	-	Soustraction numérique
2	+	Addition numérique et concaténation de chaînes
2	&	Concaténation de chaînes
3	*	Multiplication numérique
3	/	Division numérique
3	MOD	Reste numérique après division entière
4	^	Puissance numérique

```

Sub TestPrecedence
    Dim i%
    Print 1 + 2 OR 1      REM imprime 3
    Print 1 + (2 OR 1)    REM imprime 4
    Print 1 + 2 AND 1    REM imprime 1
    Print 1 + 2 * 3      REM imprime 7
    Print 1 + 2 * 3 ^2    REM imprime 19
    Print 1 = 2 OR 4      REM imprime 4
    Print 4 AND 1 = 1    REM imprime 4
End Sub

```

Les valeurs booléennes sont stockées en interne comme des entiers valant 0 pour False et -1 pour True. Ceci permet l'emploi des opérateurs numériques avec des valeurs booléennes mais je le déconseille (1 + Vrai = Faux). Utilisez plutôt les opérateurs booléens.

OOBasic fournit quelques fonctions de manipulation des chaînes de caractères.

Asc(s\$)	Valeur ASCII du premier caractère
Chr\$(i)	Caractère correspondant à une valeur ASCII
CStr(Expression)	Convertit une expression numérique en chaîne de caractères
Format(number [, f])	Formate un nombre sur la base d'une chaîne de formatage
Hex(Number)	Chaîne représentant la valeur hexadécimale d'un nombre
InStr([i,] s\$, f\$,[c])	Position de f dans s, 0 si non trouvé. Peut être insensible à la casse (c=1). Le type retourné est un Long mis dans un Integer donc la valeur retournée peut être négative pour les grandes chaînes.
LCase(s\$)	Met en minuscule
Left(s\$, n)	Renvoie les n premiers caractères en partant de la gauche. n est un Integer alors que la chaîne peut être d'une taille de 64K
Len(s\$)	Renvoie le nombre de caractères de la chaîne
LSet s\$ = Text	Aligne une chaîne à gauche. ??Bogue dans la 1.0.3.1, supposé fixé dans la 1.1
LTrim(s\$)	Renvoie une chaîne sans espace à gauche, ne modifie pas la chaîne
Mid(s\$, i[, n])	Sous-chaîne à partir de la position i sur une longueur de n
Mid(s\$, i, n, r\$)	Remplace la sous-chaîne par r avec des limitations. Je l'emploie pour supprimer des parties.
Oct(Number)	Chaîne représentant la valeur Octale d'un nombre
Right(s\$, n)	Renvoie les n derniers caractères de la chaîne. n est un Integer alors que la chaîne peut être d'une taille de 64K
RSet s\$ = Text	Aligne une chaîne à droite
RTrim(s\$)	Renvoie une chaîne sans espaces à la fin
Space(n)	Retourne une chaîne avec le nombre d'espaces spécifié
Str(Expression)	Convertit l'expression numérique en chaîne
StrComp(x\$, y\$[, c])	Retourne -1 si x>y, 0 si x=y et 1 if x<y. Si c=1 alors insensible à la casse
String(n, {i s\$})	Crée une chaîne de n caractères. Si un entier est donné, il est considéré comme la valeur ASCII du caractère à répéter. Si une chaîne est spécifiée, son premier caractère est répété n fois.
Trim(s\$)	Renvoie une chaîne débarrassée de ses espaces de début et de fin
UCase(s\$)	Renvoie la chaîne en majuscule

Val(s\$)	Convertit la chaîne en nombre
----------	-------------------------------

Dans l'aide en ligne, l'exemple de changement de la casse d'une chaîne de caractères est erroné. Voici comment il faut lire :

```
Sub ExampleLUCase
    Dim sVar As String
    sVar = "Las Vegas"
    Print LCase(sVar) REM Affiche "las vegas"
    Print UCase(sVar) REM Affiche "LAS VEGAS"
End Sub
```

Enlever des caractères d'une Chaîne

Cette macro retire des caractères d'une chaîne. Cette fonctionnalité aurait pu être assurée par la fonction Mid citée précédemment :

```
'Retire un certain nombre de caractères d'une chaîne
Function RemoveFromString($$, index&, num&) As String
    If num = 0 Or Len(s) < index Then
        'Si on ne retire rien ou en dehors de la taille de la chaîne, on retourne la chaîne initiale
        RemoveFromString = s
    ElseIf index <= 1 Then
        'Retire à partir de début
        If num >= Len(s) Then
            RemoveFromString = ""
        Else
            RemoveFromString = Right(s, Len(s) - num)
        End If
    Else
        'Retire du milieu
        If index + num > Len(s) Then
            RemoveFromString = Left(s, index - 1)
        Else
            RemoveFromString = Left(s, index - 1) + Right(s, Len(s) - index - num + 1)
        End If
    End If
End Function
```

Remplacer du texte dans une chaîne de caractères

Cette macro pourrait être utilisée pour effacer des zones d'une chaîne en spécifiant la chaîne de remplacement comme une chaîne vide. Ma première idée a été d'utiliser la fonction Mid() pour ça aussi, mais il est apparu que la fonction Mid() ne permet pas de rendre la chaîne initiale plus grande que ce qu'elle est. A cause de ça, j'ai dû écrire cette macro. Elle ne modifie pas la chaîne source mais en crée une nouvelle avec l'occurrence remplacée.

```
Rem $$ chaine source à modifier
Rem index entier long indiquant où le remplacement doit avoir lieu (Base 1)
Rem      Si index <= 1 le texte est inséré au début de la chaîne
Rem      Si index > Len(s) le texte est inséré en fin de chaîne
Rem num est un entier long indiquant combien de caractères sont à remplacer
Rem      Si num=0, rien n'est retiré mais la nouvelle chaîne est insérée
Rem (replaces) est la chaîne de remplacement.
Function ReplaceInString($$, index&, num&, replaces$$) As String
    If index <= 1 Then
        'Place en début de chaîne
        If num < 1 Then
            ReplaceInString = replaces + s
        ElseIf num > Len(s) Then
            ReplaceInString = replaces
```

```

    Else
        ReplaceInString = replaces + Right(s, Len(s) - num)
    End If
ElseIf index + num > Len(s) Then
    ReplaceInString = Left(s, index - 1) + replaces
Else
    ReplaceInString = Left(s, index - 1) + replaces + Right(s, Len(s) - index - num + 1)
End If
End Function

```

Afficher les valeurs ASCII d'une Chaîne de caractères

Cette macro pourrait paraître bizarre mais je l'ai utilisée pour décider comment un texte était stocké dans un document. Elle affiche la chaîne de caractères complète sous forme d'une suite de codes ASCII :

```

Sub PrintAll
    PrintAscii(ThisComponent.text.getString())
End Sub
Sub PrintAscii(TheText As String)
    If Len(TheText) < 1 Then Exit Sub
    Dim msg$, i%
    msg = ""
    For i = 1 To Len(TheText)
        msg = msg + Asc(Mid(TheText, i, 1)) + " "
    Next i
    Print msg
End Sub

```

Supprimer toutes les occurrences d'une chaîne de caractères

```

Rem efface toutes les occurrences bad$ de s$
Rem modifie la chaîne s$
Sub RemoveFromString(s$, bad$)
    Dim i%
    i = InStr(s, bad)
    Do While i > 0
        Mid(s, i, Len(bad), "")
        i = InStr(i, s, bad)
    Loop
End Sub

```

Abs(Number)	Valeur absolue de type double
Asc(s\$)	Valeur ASCII du premier caractère
Atn(x)	Retourne l'angle en radians dont la tangente est x
Blue(color)	Composante bleue d'un code de couleur
CByte(Expression)	Convertit en Byte une chaîne de caractères ou un nombre
CDbl(Expression)	Convertit en Double une chaîne de caractères ou un nombre
CInt(Expression)	Convertit en Integer une chaîne de caractères ou un nombre
CLng(Expression)	Convertit en Long une chaîne de caractères ou un nombre
Cos(x)	Cosinus de l'angle spécifié en radians
CSng(Expression)	Convertit en Single précision une chaîne de caractères ou un nombre
CStr(Expression)	Convertit en String une chaîne de caractères ou un nombre
Exp(Expression)	Fonction exponentielle, base e = 2.718282, inverse de Log()
Fix(Expression)	Partie entière après troncature
Format(number [, f\$])	Formate un nombre suivant une chaîne de formatage
Green(color)	Composante verte d'un code de couleur
Hex(Number)	Chaîne représentant la valeur Hexadécimale d'un nombre
Int(Number)	Arrondit à la partie entière. Voir également : Fix()
IsNumeric (Var)	Teste si l'expression est un nombre
Log(Number)	Logarithme naturel (neperien) d'un nombre
Oct(Number)	Chaîne représentant la valeur en Octale d'un nombre
Randomize [Number]	Initialise le générateur de nombre aléatoire
Red(color)	Composante rouge d'un code couleur
RGB (Red, Green, Blue)	Valeur de couleur (Long) composée des valeurs rouges, vertes, bleues
Rnd [(Expression)]	Nombre aléatoire entre 0 et 1
Sgn (Number)	Retourne 1, -1, or 0 si le nombre est positif, négatif ou nul
Sin(x)	Sinus de l'angle spécifié en radians
Sqr(Number)	Racine carrée d'un nombre
Tan(x)	Tangente de l'angle spécifié en radians

n = TwipsPerPixelX	Nombre de twips pour la largeur d'un pixel
n = TwipsPerPixelY	Nombre de twips pour la hauteur d'un pixel
Val(s\$)	Convertit une chaîne en nombre

CDate(Expression)	Convertit en date un nombre ou une chaîne
CDateFromIso(String)	Retourne la date numérique à partir d'une chaîne contenant une date au format ISO
CDateToIso(Number)	Retourne date ISO à partir d'une date générée par DateSerial ou DateValue
Date	Date système en cours
Date = s\$	Change la date système
DateSerial(y%, m%, d%)	Date à partir des arguments Année, Mois, Jour
DateValue([date])	Entier long utilisable pour calculer des différences de date
Day(Number)	Jour du mois d'une date obtenue avec DateSerial ou DateValue
GetSystemTicks()	Retourne les ticks systèmes fourni par le système d'exploitation
Hour(Number)	Heures d'un temps obtenu par TimeSerial ou TimeValue
Minute(Number)	Minutes d'un temps obtenu par TimeSerial ou TimeValue
Month(Number)	Mois d'une date obtenue par DateSerial ou DateValue
Now	Date et heure système courantes
Second(Number)	Secondes d'une date obtenue par TimeSerial or TimeValue
Time	Heure système en cours
Timer	Nombre de secondes écoulées depuis minuit
TimeSerial (h, m, s)	Heure numérique à partir des paramètres Heure, Minutes, Secondes
TimeValue (s\$)	Heure numérique à partir d'une chaîne formatée
Wait millisec	Pause pendant le nombre de millisecondes spécifiées
WeekDay(Number)	Jour de la semaine d'une date obtenue par DateSerial ou DateValue

Year(Number)	Année d'une date obtenue par DateSerial or DateValue

ChDir (s\$)	Change le répertoire ou le lecteur courant
ChDrive(s\$)	Change le lecteur courant
Close #n% [, #n2% [...]]	Ferme les fichiers ouverts avec l'instruction Open
ConvertFromURL (s\$)	Convertit une URL en nom de fichier système
ConvertToURL(s\$)	Convertit un nom de fichier système en URL
CurDir([s\$])	Répertoire courant du lecteur spécifié
Dir [s\$ [, Attrib%]]	Listing du répertoire
EOF(n%)	Le pointeur de fichier a-t-il atteint la fin du fichier ?
FileAttr (n%, Attribut%)	Attributs d'un fichier ouvert
FileCopy from\$, to\$	Copie un fichier
FileDateTime(s\$)	Chaîne Date et Heure du fichier
FileExists(s\$)	Un fichier ou répertoire existe-t-il ?
FileLen (s\$)	Longueur du fichier en octets
FreeFile	Numéro suivant disponible de fichier. Évite les utilisations simultanées.
Get [#]n%, [Pos], v	Lit un enregistrement ou un nombre d'octets d'un fichier
GetAttr(s\$)	Retourne une séquence identifiant le type de fichier
Input #n% v1[, v2[, ...]]	Lit des données d'un fichier ouvert en séquentiel
Kill f\$	Efface un fichier du disque
Line Input #n%, v\$	Affecte une ligne d'un fichier séquentiel dans une variable
Loc (FileNumber)	Position courante dans un fichier
Lof (FileNumber)	Taille courante du fichier
MkDir s\$	Crée un nouveau répertoire
Name old\$, new\$	Renomme un fichier ou un répertoire existant
Open s\$ [#]n%	Ouverture d'un fichier. La plupart des paramètres ne sont pas listés, c'est très souple.
Put [#] n%, [pos], v	Écrit un enregistrement ou une séquence d'octets dans un fichier
Reset	Ferme tous les fichiers et vide tous les buffers sur le disque
RmDir f\$	Supprime un répertoire
Seek[#]n%, Pos	Bouge le pointeur de fichier

SetAttr f\$, Attribute%	Définit les attributs du fichier
Write [#]n%, [Exprs]	Écrit des données dans un fichier séquentiel

Le but de ce chapitre n'est pas de reprendre mot pour mot l'aide en ligne mais d'apporter quelques remarques complémentaires concernant les mots réservés du langage.

Description :

Retranche deux valeurs numériques. La présence mathématique usuelle des opérateurs est applicable comme décrite page406..

Result = Expression1 - Expression2

Result : Résultat de la soustraction.

Expression1, Expression2 : Toute valeur numérique.

```
Sub SubtractionExample
    Print 4 - 3          '1
    Print 1.23e2 - 23   '100
End Sub
```

Opérateur *

Multiplie deux valeurs numériques. La présence mathématique usuelle des opérateurs est applicable comme décrite page 406.

Result = Expression1 * Expression2

Result : Résultat de la multiplication.

Expression1, Expression2 : Toute valeur numérique.

```
Sub MultiplicationExample
    Print 4 * 3          '12
    Print 1.23e2 * 23   '2829
End Sub
```

Opérateur +

Additionne deux valeurs numériques. Bien que cet opérateur fonctionne sur des variables booléennes, car représentées comme des entiers, je recommande de ne pas utiliser cette fonctionnalité. Elle s'apparente à l'opérateur OR mais le transtypage peut conduire à des problèmes, l'opération étant effectuée dans le domaine des entiers et le résultat converti. La présence mathématique usuelle des opérateurs est applicable comme décrite page 406.

Result = Expression1 + Expression2

Result : Résultat de l'addition.

Expression1, Expression2 : Toute valeur numérique .

```
Sub AdditionExample
    Print 4 + 3          '7
    Print 1.23e2 + 23   '146
End Sub
```

Opérateur ^

Élève le nombre à une puissance. Soit l'équation $x=y^z$. Si z est un entier, x est alors le résultat de la multiplication de y effectuée z fois (*NdT: Cet opérateur fonctionne également avec une puissance de type réel*). La présence mathématique usuelle des opérateurs est applicable comme décrite page 406.

Result = Expression ^ Exponent

Result : Résultat de l élévation à la puissance.

Expression : Toute valeur numérique.

Exponent : Toute valeur numérique.

```
Sub ExponentiationExample
    Print 2 ^ 3           '8
    Print 2.2 ^ 2         '4.84
    Print 4 ^ 0.5         '2
End Sub
```

Opérateur /

Divise deux valeurs numériques. Attention, le résultat de la division peut ne

pas être un entier. Utiliser la fonction Int() si nécessaire (*NdT: attention également aux divisions par zéro ;-)*). La préséance mathématique usuelle des opérateurs est applicable comme décrite page 406.

Result = Expression1 / Expression2

Result : Résultat de la division.

Expression1, Expression2 : Toute valeur numérique.

```
Sub DivisionExample
    Print 4 /2    '2
    Print 11/2   '5.5
End Sub
```

Opérateur AND

Applique un AND logique sur des valeurs booléennes et un AND bit à bit sur des valeurs numériques. Un AND bit à bit sur un type double semble impliquer une conversion en type entier. Un dépassement de capacité numérique est possible. La préséance mathématique usuelle des opérateurs est applicable comme décrite page 406 ainsi que la table de vérité ci-dessous.

VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	FAUX
1	1	1
1	0	0
0	1	0
0	0	0

Result = Expression1 AND Expression2

Result : Résultat de l'opération.

Expression1, Expression2 : Expression numérique ou booléenne.

```
Sub AndExample
    Print (3 And 1)      'Affiche 1
    Print (True And True) 'Affiche -1
    Print (True And False) 'Affiche 0
End Sub
```

ABS (Fonction)

Retourne la valeur absolue d'une expression numérique. Si le paramètre est une chaîne de caractères, il est préalablement converti en nombre, probablement en utilisant la fonction Val(). Si le nombre est négatif, son opposé est retourné (une valeur positive donc).

Abs (Number)

Double

Number : Toute valeur numérique ou pouvant être évaluée comme telle.

```
Sub AbsExample
    Print Abs(3)      '3
    Print Abs(-4)     '4
    Print Abs("-123") '123
End Sub
```

Array (Fonction)

Crée un tableau de Variant à partir d'une liste de paramètres. C'est la méthode la plus rapide pour créer un tableau de constantes.

Si vous assignez ce tableau de variant à un tableau d'un autre type, vous ne pourrez pas conserver les données si vous redimensionnez le tableau. Je pense que c'est un bogue de pouvoir assigner un tableau de Variant à un tableau d'un autre type (*NdT : l'idéal est d'utiliser les fonctions de transtypage sur les éléments comme CInt, CDbl ...*).

Voir également la fonction DimArray.

Array (Argument list)

Tableau de variants contenant la liste des arguments

Argument list : Liste des valeurs séparées par des virgules.

```
Sub ArrayExample
    Dim a(5) As Integer
    Dim b() As Variant
    Dim c() As Integer
    Rem Array() retourne un type variant.
```

```

Rem b est dimensionné de 0 à 9 avec b(i) = i+1
b = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
PrintArray("b en valeur initiale", b())
Rem b est redimensionné de 1 à 3 avec b(i) = i+1
ReDim Preserve b(1 To 3)
PrintArray("b après ReDim", b())
Rem Ce qui suit n'est pas valide car le tableau "a" est déjà dimensionné à une taille différente,
Rem mais on peut utiliser ReDim
Rem a = Array(0, 1, 2, 3, 4, 5)

Rem c est dimensionné de 0 à 5.
Rem "Hello" est une chaîne de caractères mais la version 1.0.2 l'autorise
c = Array(0, 1, 2, "Hello", 4, 5)
PrintArray("c, Variant assigné à un tableau Integer", c())
Rem Bizarrement, c'est permis mais c'est vide !
ReDim Preserve c(1 To 3) As Integer
PrintArray("c après ReDim", c())
End Sub

Sub PrintArray (lead$, a() As Variant)
    Dim i%, s$
    s$ = lead$ + Chr(13) + LBound(a()) + " à " + UBound(a()) + ":" + Chr(13)
    For i% = LBound(a()) To UBound(a())
        s$ = s$ + a(i%) + " "
    Next
    Rem J'utilise MsgBox plutôt que Print car j'ai inclus un CHR(13) (NdT: saut de ligne)
    MsgBox s$
End Sub

```

ASC (Fonction)

Retourne la valeur ASCII (American Standard Code for Information Interchange) du premier caractère de la chaîne, le reste étant ignoré. Une erreur d'exécution est générée si la chaîne est vide. Cette fonction est l'inverse de la fonction Chr.

Asc (Text As String)

Integer

Text: Toute chaîne de caractères non vide.

```

Sub AscExample
    Print Asc("ABC")      '65
End Sub

```

ATN (Fonction)

Arctangente d'une expression numérique retournant une valeur comprise entre $-\pi/2$ et $\pi/2$ (radians). Cette fonction est l'inverse de la fonction tangente (Tan). Pour les non-mathématiciens, c'est une fonction trigonométrique.

ATN(Number)

Double

Number : Toute valeur numérique valide

```
Sub ExampleATN
    Dim dLeg1 As Double, dLeg2 As Double
    dLeg1 = InputBox("Entrer la longueur du côté adjacent : ","Adjacent")
    dLeg2 = InputBox("Entrer la longueur du côté opposé : ","Opposé")
    MsgBox "L'angle est " + Format(ATN(dLeg2/dLeg1), "###0.0000") _
        + " radians" + Chr(13) + "L'angle est " -_
        + Format(ATN(dLeg2/dLeg1) * 180 / Pi, "###0.0000") + " degrés"
End Sub
```

Beep

Génère un son système.

Beep

```
Sub ExampleBeep
    Beep
    Beep
End Sub
```

Blue (Fonction)

Les couleurs sont représentées par un entier de type Long. Cette fonction retourne la valeur de la composante bleue de la couleur passée en argument. Voir également les fonctions RGB, Red et Green.

Blue (Color As Long)

Integer compris entre 0 et 255.

Color : Entier Long représentant une couleur.

```
Sub ExampleColor
    Dim IColor As Long
    IColor = RGB(255,10,128)
    MsgBox "La couleur " & IColor & " est composée de:" & Chr(13) &_
        "Rouge = " & Red(IColor) & Chr(13)&_
        "Vert= " & Green(IColor) & Chr(13)&_
        "Bleu= " & Blue(IColor) & Chr(13) , 64,"Couleurs"
```

```
End Sub
```

ByVal (Mot-clé)

Les paramètres des procédures et des fonctions définies par l'utilisateur sont passés par référence. Si la procédure ou fonction modifie le paramètre, la modification est reportée dans la procédure ou fonction appelante. Ceci peut amener un comportement plus ou moins étrange si le paramètre appelant est une constante, ou si la procédure ou fonction appelante n'attend pas cette modification. Le mot-clé ByVal spécifie que l'argument doit être passé par valeur et non par adresse (référence).

```
Sub Name(ByVal ParmName As ParmType)
```

```
Sub ExampleByVal
    Dim j As Integer
    j = 10
    ModifyParam(j)
    Print j
    Rem 9
    DoNotModifyParam(j)
    Print j
    Rem toujours 9
End Sub
Sub ModifyParam(i As Integer)
    i = i - 1
End Sub
Sub DoNotModifyParam(ByVal i As Integer)
    i = i - 1
End Sub
```

Merci à Kelvin demo@onlineconnections.com.au pour ses contributions, comme avoir souligné le rôle du mot clé ByVal.

Call (Instruction)

Appelle l'exécution d'une procédure, fonction ou procédure d'une DLL. Call est optionnel (sauf pour les DLL qui doivent être définies au préalable – (NdT: Voir le mot-clé Declare)). Les arguments peuvent être écrits entre parenthèses et le doivent dans le cas d'un appel à une fonction.

```
Declare
```

```
[Call] Name [(Parameters)]
```

Name : Nom de la procédure, fonction ou procédure de DLL à appeler

Parameters : Le type et nombre de paramètres dépendent de la routine appelée.

```
Sub ExampleCall
    Call CallMe "Ce texte va être affiché"
End Sub
Sub CallMe(s As String)
    Print s
End Sub
```

Cbool (Fonction)

Conversion du paramètre en booléen. Si l'argument est de type numérique, 0 correspond à Faux (False), toute autre valeur à Vrai (True). Si l'argument est une chaîne de caractères, “true” et “false” (indépendamment de la casse) correspondent respectivement à True et False. Tout autre valeur de la chaîne génère une erreur d'exécution. Cette fonction est utile pour forcer un résultat à être de type booléen. Si, par exemple, j'appelle une fonction qui retourne un nombre, comme InStr, je peux écrire “If CBool(InStr(s1, s2)) Then” plutôt que “If InStr(s1, s2) <> 0 Then”.

CBool (Expression)

Boolean

Expression : Numérique, Booléen,

```
Sub ExampleCBool
    Print CBool(1.334e-2)      'True
    Print CBool("TRUE")        'True
    Print CBool(-10 + 2*5)     'False
    Print CBool("John" <> "Fred")  'True
End Sub
```

CByte (Fonction)

Convertit une chaîne de caractères ou une expression numérique vers le type Byte. Les chaînes de caractères sont converties en une expression numérique et les doubles sont arrondis. Si l'expression est négative ou trop grande, une erreur est générée.

Cbyte(expression)

Byte

Expression : Une chaîne de caractères ou une expression numérique

```

Sub ExampleCByte
    Print Int(CByte(133))      '133
    Print Int(CByte("AB"))     '0
    'Print Int(CByte(-11 + 2*5)) 'Erreur, en dehors de la plage autorisée
    Print Int(CByte(1.445532e2)) '145
    Print CByte(64.9)          'A
    Print CByte(65)            'A
    Print Int(CByte("12"))
End Sub

```

CDate (Fonction)

Convertit vers le type Date. Une expression numérique peut être interprétée comme une date, la partie entière étant le nombre de jours depuis le 31/12/1899, la partie décimale, l'heure. Les chaînes de caractères doivent être formatées comme spécifié par la convention des fonctions DateValue et TimeValue. En d'autres termes, le formatage dépend de la configuration locale de l'utilisateur. La fonction CDateFromIso n'est pas dépendante de cette configuration et pourra être utilisée le cas échéant.

CDate (Expression)

Date

Expression : Une chaîne de caractères ou expression numérique

```

sub ExampleCDate
    MsgBox cDate(1000.25)
    REM 26/09/1902 06:00:00
    MsgBox cDate(1001.26)
    REM 27/09/1902 06:14:24
    Print DateValue("06/08/2002")
    MsgBox cDate("06/08/2002 15:12:00")
end sub

```

CDateFromIso (Fonction)

Retourne la représentation numérique d'une chaîne de caractères contenant une date au format ISO

CDateFromIso(String)

Une date

String : Une chaîne de caractères contenant une date au format ISO. L'année peut être sous deux ou quatre chiffres.

```
sub ExampleCDateFromIso
    MsgBox cDate(37415.70)
    Rem 08/06/2002 16:48:00
    Print CDateFromIso("20020608")
    Rem 08/06/2002
    Print CDateFromIso("020608")
    Rem 08/06/1902
    Print Int(CDateFromIso("20020608"))
    Rem 37415
end sub
```

CDateFromIso (Fonction)

Retourne la date au format ISO à partir d'un nombre généré avec DateSerial ou DateValue.

CDateFromIso(Number)

String

Number : Entier contenant la représentation numérique de la date.

```
Sub ExampleCDateTolso
    MsgBox "" & CDateTolso(Now) ,64,"Date ISO"
End Sub
```

CDtoI (Fonction)

Convertit toute expression en variable de type double. Les chaînes de caractères doivent respecter les paramétrages locaux. Sur un système paramétré en français « 12,37 » sera accepté alors que « 12.37 » générera une erreur.

CDtoI (Expression)

Double

Expression : Toute chaîne de caractères ou expression numérique valide.

```
Sub ExampleCDbl
    Msgbox CDbl(1234.5678)
    Msgbox CDbl("1234.5678")
End Sub
```

ChDir (Fonction)

Change le répertoire courant. Si vous voulez changer le lecteur courant, tapez sa lettre suivi de deux-points (D: par exemple)

ChDir (Text)

Text : Une chaîne de caractères spécifiant un chemin

```
Sub ExampleChDir
    Dim sDir As String
    sDir = CurDir
    ChDir( "C:\temp" )
    MsgBox CurDir
    ChDir( sDir )
    MsgBox CurDir
End Sub
```

ChDrive (Fonction)

Change le lecteur courant. La lettre du lecteur doit être en majuscules. Il est judicieux d'utiliser l'instruction OnError pour gérer toute erreur pouvant survenir (disque réseau non monté par exemple)

ChDrive (Text)

Text : chaîne de caractères contenant la lettre du lecteur. La syntaxe URL est acceptée

```
Sub ExampleCHDrive
    On Local Error Goto NoDrive
    ChDrive "Z"
    REM Possible seulement si le disque Z existe
    Print "Fait !"
    Exit Sub
NoDrive:
    Print "Désolé, le disque n'existe pas"
    Resume Next
End Sub
```

Choose (Instruction)

Retourne une valeur se situant à un endroit précis d'une liste. Si l'index excède le nombre d'éléments, la valeur Null est renvoyée.

Choose (Index, Selection_1[, Selection_2, ... [,Selection_n]])

Sera du type de la valeur sélectionnée

Index : Une expression numérique spécifiant la position de la valeur à retourner de la liste

Selection_i : Une valeur à retourner

Dans cet exemple, la variable “o” n'est pas typée donc elle prendra le type de ce qui sera sélectionné. Selection_1 est de type “String” et Selection_2 de type double. Si “o” est typée, la valeur est transformée en ce type.

```
Sub ExampleChoose
    Dim sReturn As String
    Dim sText As String
    Dim i As Integer
    Dim o
    sText = InputBox ("Entrer un nombre (1-3):","Exemple")
    i = Int(sText)
    o = Choose(i, "Un", 2.2, "Trois")
    If IsNull(o) Then
        Print "Désolé, " + sText + " n'est pas valide"
    Else
        Print "Obtenu " + o + " de type " + TypeName(o)
    End If
end Sub
```

Chr (fonction)

Retourne le caractère correspondant au code ASCII ou Unicode passé en argument. On peut l'utiliser pour générer des séquences de contrôle comme les caractères d'échappement pour les imprimantes, tabulations, nouvelles lignes, retour chariots etc... On la rencontre quelquefois sous la forme “Chr\$()”. Voir également son inverse, la fonction Asc.

Chr(Expression)

String

Expression : Variable numérique représentant une valeur valide de la table ASCII (0-255) sur 8 bits ou une valeur Unicode sur 16 bits.

```
Exemple :
sub ExampleChr
    REM Affiche "Ligne 1" et "Ligne 2" sur des lignes séparées.
    MsgBox "Ligne 1" + Chr$(13) + "Ligne 2"
End Sub
```

CInt (Fonction)

Convertit l'argument en un entier (Integer). Les chaînes de caractères doivent respecter le formatage local. En France, « 12.37 » ne marchera pas et générera une erreur.

Voir également la fonction Fix.

CInt(Expression)

Integer

Expression : chaîne ou expression numérique à convertir

```
Sub ExampleCInt
    MsgBox CInt(1234.5678)
    MsgBox CInt("1234,5678")
End Sub
```

CLng (Fonction)

Convertit l'argument en un entier long (Long). Les chaînes de caractères doivent respecter le formatage local. En France, « 12.37 » ne marchera pas et générera une erreur.

CLng(Expression)

Long

Expression : chaîne ou expression numérique à convertir.

```
Sub ExampleCLng
    MsgBox CLng(1234.5678)
    MsgBox CLng("1234,5678")
End Sub
```

Close (Instruction)

Ferme les fichiers ouverts auparavant avec l'instruction Open. Plusieurs fichiers peuvent être fermés simultanément.

Voir également Open, EOF, Kill, et FreeFile

Close #FileName As Integer[, #FileName2 As Integer[,...]]

FileName : Entier spécifiant un fichier ouvert auparavant.

```
Sub ExampleCloseFile
    Dim iNum1 As Integer, iNum2 As Integer
    Dim sLine As String, sMsg As String
    'Numéro de fichier valide suivant
    iNum1 = FreeFile
    Open "c:\data1.txt" For Output As #iNum1
    iNum2 = FreeFile
    Open "c:\data2.txt" For Output As #iNum2
    Print #iNum1, "Texte dans fichier un pour le numéro " + iNum1
    Print #iNum2, "Texte dans le fichier deux pour le numéro " + iNum2
    Close #iNum1, #iNum2
    Open "c:\data1.txt" For Input As #iNum1
    iNum2 = FreeFile
    Open "c:\data2.txt" For Input As #iNum2
    sMsg = ""
    Do While not EOF(iNum1)
        Line Input #iNum1, sLine
        If sLine <> "" Then sMsg = sMsg + "Fichier: "+iNum1+":"+sLine+Chr(13)
    Loop
    Close #iNum1
    Do While not EOF(iNum2)
        Line Input #iNum2, sLine
        If sLine <> "" Then sMsg = sMsg + "Fichier: "+iNum2+":"+sLine+Chr(13)
    Loop
    Close #iNum2
    MsgBox sMsg
End Sub
```

Const (Instruction)

Les constantes améliorent la lisibilité et la maintenance d'un programme en assignant un nom à une valeur et en ne donnant qu'un seul endroit dans le code pour sa définition. Les constantes peuvent être typées mais ce n'est pas obligatoire (bien que souhaitable). Une constante est définie une fois pour toute et ne peut être modifiée.

Const Text [As type] = Expression[, Text2 [As type] = Expression2[, ...]]

Text : Toute chaîne de caractères respectant la convention de nommage des variables.

```
Sub ExampleConst
    Const iVar As String = 1964
    Const sVar = "Programme", dVar As Double = 1.00
    MsgBox iVar & " " & sVar & " " & dVar
End Sub
```

ConvertFromURL (Fonction)

Conversion du nom d'un fichier au format URL en un nom de fichier dépendant du système.

ConvertFromURL(filename)

String

Filename : Chemin d'un fichier en notation URL

```
Sub ExampleUrl
    Dim sUrl As String, sName As String
    sName = "c:\temp\file.txt"
    sUrl = ConvertToURL(sName)
    MsgBox "Fichier original:" + sName + Chr(13) + "URL: " + sURL + Chr(13) + _
           "Et inversement:" + ConvertFromURL(sUrl)
End Sub
```

ConvertToURL (Fonction)

Conversion d'un nom de fichier au format système vers un nom en notation URL

ConvertToURL(filename)

String

Filename : Nom du fichier

```
Sub ExampleUrl
    Dim sUrl As String, sName As String
    sName = "c:\temp\file.txt"
    sUrl = ConvertToURL(sName)
    MsgBox "Fichier original:" + sName + Chr(13) + _
           "URL: " + sURL + Chr(13) + _
           "Et inversement:" + ConvertFromURL(sUrl)
End Sub
```

Cos (Fonction)

Cosinus d'une expression numérique retournant une valeur entre -1 et +1. Pour les non-mathématiciens, c'est une fonction trigonométrique.

Cos(Number)

Double

Number : Expression numérique représentant un angle en radians

```
Sub ExampleCos
    Dim dLeg1 As Double, dLeg2 As Double, dHyp As Double
    Dim dAngle As Double
    dLeg1 = InputBox("Entrer la longueur du côté adjacent: ","Adjacent")
    dLeg2 = InputBox("Entrer la longueur du côté opposé: ","Opposé")
    dHyp = Sqr(dLeg1 * dLeg1 + dLeg2 * dLeg2)
    dAngle = Atn(dLeg2 / dLeg1)
    MsgBox "Côté adjacent= " + dLeg1 + Chr(13) + _
        "Côté opposé = " + dLeg2 + Chr(13) + _
        "Hypothénuse = " + Format(dHyp, "###0.0000") + Chr(13) + _
        "Angle = "+Format(dAngle*180/Pi, "###0.0000")+" degrés"+Chr(13)+_
        "Cos = " + Format(dLeg1 / dHyp, "###0.0000") + Chr(13) + _
        "Cos = " + Format(Cos(dAngle), "###0.0000")
End Sub
```

CreateUnoDialog (Fonction)

Crée un objet UNO représentant une boîte de dialogue UNO à l'exécution. Les boîtes de dialogue sont définies dans les bibliothèques de dialogue. Pour afficher une boîte de dialogue, elle doit être créée à partir de la bibliothèque.

CreateUnoDialog(oDlgDesc)

Object : Boîte de dialogue à exécuter

oDlgDesc : Description de la boîte de dialogue définie auparavant dans une bibliothèque (library)

```
Sub ExampleCreateDialog
    Dim oDlgDesc As Object, oDlgControl As Object
    DialogLibraries.LoadLibrary("Standard")
    ' Obtient la description de la boîte de dialogue dans la bibliothèque (library)
    oDlgDesc = DialogLibraries.Standard
    Dim oNames(), i%
    oNames = DialogLibraries.Standard.getElementNames()
    i = lBound( oNames() )
    while( i <= uBound( oNames() ) )
        MsgBox "Voici " + oNames(i)
        i = i + 1
    wend
    oDlgDesc = DialogLibraries.Standard.Dialog1
    ' produire un dialogue en temps réel
```

```
oDlgControl = CreateUnoDialog( oDlgDesc )
' afficher le dialogue en temps réel
oDlgControl.execute
End Sub
```

CreateUnoService (Fonction)

Crée une instance d'un service UNO avec le ProcessServiceManager.

```
oService = CreateUnoService( UnoServiceName as string)
```

??

??

```
oIntrospection = CreateUnoService( "com.sun.star.beans.Introspection" )
```

CreateUnoStruct (Fonction)

Crée une instance d'un type de structure UNO. Il est préférable d'utiliser la construction suivante :

```
Dim oStruct as new com.sun.star.beans.Property
```

```
oStruct = CreateUnoStruct( Uno type name )
```

```
oStruct = CreateUnoStruct("com.sun.star.beans.Property")
*****
'Voulez-vous choisir une imprimante particulière ?
Dim mPrinter(0) As New com.sun.star.beans.PropertyValue
mPrinter(0).Name="Nom"
mPrinter(0).value="Autre imprimante"
oDocument.Printer = mPrinter()
'Vous auriez pu faire comme ci-après, après qu'il ait été défini et dimensionné
'mPrinter(0) = CreateUnoStruct("com.sun.star.beans.PropertyValue")
```

CSng (Fonction)

Convertit l'argument en un entier long (Long). Les chaînes de caractères doivent respecter le formatage local. En France, « 12.37 » ne marchera pas et

générera une erreur.

CSng(Expression)

Single

Expression : chaîne ou expression numérique à convertir.

```
Sub ExampleCLng
    MsgBox CSng(1234.5678)
    MsgBox CSng("1234.5678")
End Sub
```

CStr Function

Convertit toute expression en chaîne de caractères. Elle est généralement utilisée pour convertir les nombres en chaînes de caractères.

Boolean	“True” et “False”
Date	Chaîne avec Date et Heure
Null	Erreur d'exécution
Empty	Chaîne vide
Any Number	Le nombre en tant que chaîne. Les zéros superflus à droite de la décimale ne sont pas convertis.

CStr (Expression)

String

Expression : Toute chaîne ou valeur numérique à convertir.

```
Sub ExampleCSTR
Dim sVar As String
Msgbox Cdbl(1234,5678)
Msgbox Cint(1234,5678)
Msgbox Clng(1234,5678)
Msgbox Csng(1234,5678)
sVar = Cstr(1234,5678)
MsgBox sVar
end sub
```

CurDir (Fonction)

Retourne le chemin courant du lecteur spécifié. Si l'argument est omis, le chemin du lecteur courant est retourné.

CurDir [(Text As String)]

String

Text : Optionnel – chaîne contenant la lettre du lecteur à analyser. Ne dépend pas de la casse.

```
Sub ExampleCurDir
    MsgBox CurDir("c")
    MsgBox CurDir("p")
    MsgBox CurDir()
End Sub
```

Date (Fonction)

Retourne ou change la date système. Le format de la date dépend de la configuration locale.

Date

Date = Text As String

String

Text : Nouvelle date système au format de la configuration locale

```
Sub ExampleDate
    MsgBox "La date est " & Date
End Sub
```

DateSerial (Fonction)

Convertit un triplet année, mois, jour en un objet Date. La représentation interne est du type Double. La valeur 0 représente le 30 décembre 1899. On peut interpréter cette valeur comme le nombre de jours écoulés depuis cette date, les nombres négatifs représentant une date antérieure.

DateValue, Date, et Day.

Les années sur deux chiffres sont interprétées comme 19xx.
Ceci n'est pas cohérent avec la fonction DateValue.

DateSerial (year, month, day)

Date

Year : Integer. Les valeurs entre 0 et 99 sont interprétées comme les années 1900 à 1999. Toutes les autres années doivent être spécifiées sur 4 chiffres.

Month : Integer représentant le mois. Valeurs comprises entre 1 et 12.

Day : Integer représentant le jour. Valeurs comprise entre 1 et 28, 29, 30 ou 31 (dépendant du mois).

```
Sub ExampleDateSerial
    Dim IDate as Long, sDate as String, INumDays As Long
    IDate = DateSerial(2002, 6, 8)
    sDate = DateSerial(2002, 6, 8)
    MsgBox IDate
    REM retourne 37415
    MsgBox sDate
    REM retourne 08/06/2002
    IDate = DateSerial(02, 6, 8)
    sDate = DateSerial(02, 6, 8)
    MsgBox IDate
    REM retourne 890
    MsgBox sDate
    REM retourne 08/06/1902
end sub
```

DateValue (Fonction)

Convertit une chaîne contenant une date en un nombre utilisable pour déterminer le nombre de jours entre deux dates.

DateSerial, Date, et Day

Les années sur deux chiffres sont considérées comme 20xx.
Ceci n'est pas cohérent avec la fonction DateSerial.

DateValue [(date)]

Long

Date : String représentant une date.

```
Sub ExampleDateValue
```

```

Dim s(), i%, sMsg$, l1&, l2&
Rem Toutes ces dates correspondent au 6 Juin 2002 (NdT:au format US)
s = Array("06.08.2002", "6.08.02", "6.08.2002", "June 08, 2002", _
    "Jun 08 02", "Jun 08, 2002", "Jun 08, 02", "06/08/2002")
Rem Si vous utilisez ces valeurs, une erreur va être générée
Rem ce qui contredit l'aide en ligne
Rem s = Array("6.08, 2002", "06.08, 2002", "06,08.02", "6,08.2002", "Jun/08/02")
sMsg = ""
For i = LBound(s()) To UBound(s())
    sMsg = sMsg + DateValue(s(i)) + "<=" + s(i) + Chr(13)
Next
MsgBox sMsg
Print "Je me suis marié il y a " + (DateValue(Date) - DateValue("6/8/2002")) + " jours"
end sub

```

Day (Fonction)

Retourne le jour du mois sur la base d'une date numérique générée par DateSerial ou DateValue.

Day (Number)

Integer

Number : Date numérique telle que retournée par DateSerial

```

Sub ExampleDay
    Print Day(DateValue("6/8/2002"))      Rem affiche 6
    Print Day(DateSerial(02,06,08))        Rem affiche 8
end sub

```

Declare (Instruction)

Utilisé pour déclarer une procédure DLL (Dynamic Link Library) qui doit être exécutée à partir de OpenOffice.org Basic ; on utilisera le mot-clé ByVal si les paramètres doivent être passés par valeur et non par référence.

FreeLibrary, Call

Declare {Sub | Function} Name Lib "Libname" [Alias "Aliasname"] [Parameter] [As Typ]

Name : Un nom quelconque utilisé pour appeler la routine depuis OpenOffice.org

Aliasname : Nom de la procédure tel que défini dans la procédure.

Libname : Fichier ou nom système de la DLL. La bibliothèque est automatiquement chargée à sa première utilisation.

Parameter : Liste des paramètres, arguments à passer à la procédure lors de l'appel. (Dépendant donc de la procédure appelée).

Type : Type retourné par la fonction de la DLL. Ce type peut être omis si un « caractère de définition de type » est accolé au nom de la fonction.

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep" ( long )
Sub ExampleDeclare
    Dim IValue As Long
    IValue = 5000
    MyMessageBeep( IValue )
    FreeLibrary("user32.dll" )
End Sub
```

DefBool (Instruction)

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par "b" soit automatiquement considérée comme une variable Booléenne.

DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

DefBool Characterrange1[, Characterrange2[, ...]]

Characterrange : lettres spécifiant la plage de caractères.

```
REM Définition des types de variables par défaut
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
DefBool b-d,q
Sub ExampleDefBool
    cOK = 2.003
    zOK = 2.003
    Print cOK    Rem True
    Print zOK    Rem 2.003
End Sub
```

DefDate (Instruction)

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par "t" soit automatiquement considérée comme une variable Date.

DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et

DefVar.

DefDate Characterrange1[, Characterrange2[, ...]]

Characterrange : lettres spécifiant la plage de caractères.

Voir ExampleDefBool

DefDbI (Instruction)

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par "d" soit automatiquement considérée comme une variable Double.

DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et
DefVar.

DefDbI Characterrange1[, Characterrange2[, ...]]

Characterrange : lettres spécifiant la plage de caractères.

Voir ExampleDefBool

DefInt (Instruction)

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par "i" soit automatiquement considérée comme une variable Integer.

DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et
DefVar.

DefInt Characterrange1[, Characterrange2[, ...]]

Characterrange : lettres spécifiant la plage de caractères.

Voir ExampleDefBool

DefLng (Instruction)

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par "l" soit automatiquement considérée comme une variable Long.

DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et
DefVar.

DefLng Characterrange1[, Characterrange2[, ...]]

Characterrange : lettres spécifiant la plage de caractères.

Voir ExampleDefBool

DefObj (Instruction)

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par "o" soit automatiquement considérée comme une variable Object.

DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et
DefVar.

DefObj Characterrange1[, Characterrange2[, ...]]

Characterrange : lettres spécifiant la plage de caractères.

Voir ExampleDefBool

DefVar (Instruction)

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par "v" soit automatiquement considérée comme une variable Variant.

DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et
DefVar.

DefVar Characterrange1[, Characterrange2[, ...]]

Characterrange : lettres spécifiant la plage de caractères.

Voir ExampleDefBool

Dim (Instruction)

Déclare les variables. Le type de chaque variable est à déclarer séparément et le type par défaut est le Variant.

L'exemple suivant déclare a, b et c comme des Variant, d comme une Date.

Dim a, b, c, d As Date

Un type de variable peut également être défini en utilisant un caractère réservé ajouté à la suite du nom comme mentionné dans la section dédiée au type des variables.

Dim est utilisée pour déclarer des variables locales à l'intérieur des procédures. Les variables globales en dehors des procédures sont déclarées avec les instructions PRIVATE et PUBLIC.

A moins que l'instruction “Option Explicit” ne soit présente, les variables (en dehors des tableaux) peuvent être utilisées sans déclaration et leur type par défaut sera Variant ou cohérent avec les instructions DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Les tableaux mono et multi-dimensionnels sont supportés.

Public, Private, ReDim

[ReDim]Dim Name_1 [(start To end)] [As Type][, Name_2 [(start To end)] [As Type][,...]]

Name_i : Nom de la variable ou du tableau.

Start, End : Entier constant compris entre -32768 et 32767 donnant les bornes du tableau. Au niveau de la procédure, l'instruction Redim permet des expressions numériques et des variables permettant de réinitialiser le tableau durant l'exécution.

Type : Mot-clé définissant le type de la variable. Les types supportés sont Boolean, Currency, Date, Double, Integer, Long, Object, Single, String, et Variant.

```
Sub ExampleDim
    Dim s1 As String, i1 As Integer, i2%
    Dim a1(5) As String
    Rem 0 à 6
    Dim a2(3, -1 To 1) As String
    Rem (0 à 3, -1 à 1)
    Const sDim as String = " Dimension:"
    For i1 = LBound(a2(), 1) To UBound(a2(), 1)
```

```

For i2 = LBound(a2(), 2) To UBound(a2(), 2)
    a2(i1, i2) = Str(i1) & ":" & Str(i2)
Next
Next
For i1 = LBound(a2(), 1) To UBound(a2(), 1)
    For i2 = LBound(a2(), 2) To UBound(a2(), 2)
        Print a2(i1, i2)
    Next
Next
End Sub

```

DimArray (Fonction)

Création d'un tableau de variant. Fonctionne comme l'instruction Dim. S'il n'y a pas d'arguments, un tableau vide est créé. Si des paramètres sont donnés, une dimension est créée pour chacun d'eux.

Array

DimArray (Argument list)

Variant array

Argument list : Optionnel. Liste d'entiers séparés par des virgules.

```
DimArray( 2, 2, 4 ) 'identique à DIM a( 2, 2, 4 )
```

Dir (Fonction)

Retourne le nom d'un fichier, d'un répertoire ou de tous les fichiers et répertoires d'un disque qui correspondent au critère de recherche. Il est notamment possible de vérifier l'existence d'un fichier ou répertoire spécifique ou de lister les fichiers et sous-répertoires d'un répertoire donné. Si aucun fichier ou répertoire ne correspond au critère, une chaîne vide est retournée.

On appelle la fonction Dir en boucle pour itérer toutes les valeurs correspondant à ce critère jusqu'à l'obtention d'un chaîne vide.

Les attributs de répertoire et volume dans la recherche sont exclusifs : c'est la seule information qui sera retournée. Je ne peux pas déterminer lequel a la présence car l'attribut de volume ne fonctionne pas sur la version 1.0.3.

Les attributs sont une partie de ceux disponibles avec la fonction GetAttr.

GetAttr

Dir [(Text As String[, Attrib As Integer])]

String

Text : String spécifiant le chemin à explorer, répertoire ou fichier. La notation URL est acceptée.

Attrib : Integer Valeur sommée des attributs possibles. La fonction Dir ne retourne que les éléments répondants aux critères. Additionner les attributs pour les combiner.

Attributs	Description
0	Normal.
2	Fichier caché.
4	Fichier système.
8	Nom du volume (Exclusivement).
16	Répertoires (Exclusivement).

```
Sub ExampleDir
    REM Affiche tous les fichiers et répertoires
    Dim sFile as String, sPath As String
    Dim sDir as String, sValue as String
    Dim iFile as Integer
    sFile= "Fichiers : "
    sDir="Répertoires :"
    iFile = 0
    sPath = CurDir
    Rem 0 : Normal.
    Rem 2 : Fichiers cachés.
    Rem 4 : Fichiers système.
    Rem 8 : Nom de volume.
    Rem 16 : Nom du répertoire uniquement.
    Rem Cet exemple ne va lister que les répertoires puisque la valeur 16 est incluse
    Rem Enlever le 16 pour lister les fichiers
    sValue = Dir$(sPath, 0 + 2 + 4 + 16)
    Do
        If sValue <> "." and sValue <> ".." Then
            If (GetAttr( sPath + getPathSeparator() + sValue) AND 16) > 0 Then
                REM Les répertoires
                sDir = sDir & chr(13) & sValue
            Else
                REM Les fichiers
                If iFile Mod 3 = 0 Then sFile = sFile + Chr(13)
                iFile = iFile + 1
                sFile = sFile + sValue &; "
            End If
        End If
        sValue = Dir$
    Loop Until sValue = ""
    MsgBox sDir,0,sPath
    MsgBox "" & iFile & " " & sFile,0,sPath
End Sub
```

La méthode getPathSeparator() est disponible bien qu'elle n'apparaisse pas dans l'aide.

Certains OS incluent les répertoires “.” et “..” qui se réfèrent respectivement au répertoire courant et père. Si vous écrivez une macro qui traverse les répertoires vous devrez les filtrer pour éviter une boucle infinie.

Quand vous obtenez une liste de répertoires, les fichiers ne sont jamais retournés contrairement à ce qui est indiqué dans l'aide en ligne.

Do...Loop (Instruction)

Instruction répétitive

Contrôle de boucles Page 398.

Do [{While | Until} condition = True]

Bloc d'instructions

[Exit Do]

Bloc d'instructions

Loop

Do

Bloc d'instructions

[Exit Do]

Bloc d'instructions

Loop [{While | Until} condition = True]

End (Instruction)

Marque la fin d'une procédure ou d'un bloc d'instructions

Exit

End	Fin d'exécution du programme. Peut être appelé à tout moment. Optionnel.
End Function	Fin d'une fonction
End If	Fin d'un bloc conditionnel If...Then...Else
End Select	Fin d'un bloc conditionnel Select Case
End Sub	Fin d'une procédure

```

Sub ExampleEnd
    Dim s As String
    s = InputBox ("Entrer un entier :","Testeur d'espace blanc")
    If IsWhiteSpace(Val(s)) Then
        Print "ASCII " + s + " est un espace blanc"
    Else
        Print "ASCII " + s + " n'est pas un espace blanc"
    End If
    End
    Print "On ne passe jamais ici"
End Sub
Function IsWhiteSpace(iChar As Integer) As Boolean
    Select Case iChar
        Case 9, 10, 13, 32, 160
            IsWhiteSpace = True
        Case Else
            IsWhiteSpace = False
    End Select
End Function

```

Environ (Fonction)

Retourne la valeur d'une variable d'environnement (dépendante du système d'exploitation). Sur Macintosh cette fonction retourne une chaîne vide.

Environ (Environment As String)

String

Environment : Variable d'environnement à récupérer.

```

Sub ExampleEnviron
    MsgBox "Path (chemin) = " & Environ("PATH")
End Sub

```

EOF (Fonction)

Signale la fin d'un fichier. Permet de lire un fichier sans en dépasser la fin. Quand la fin du fichier est atteinte, EOF retourne la valeur True.

Open, Close, Kill, et FreeFile

EOF (intexpression As Integer)

Boolean

Intexpression : Integer Valeur numérique représentant le numéro du fichier (voir Open)

```

Rem Exemple modifié de l'aide en ligne qui ne fonctionne pas
Sub ExampleEof
    Dim iNumber As Integer
    Dim aFile As String
    Dim sMsg as String, sLine As String
    aFile = "c:\DeleteMe.txt"
    iNumber = Freefile
    Open aFile For Output As #iNumber
    Print #iNumber, "Première ligne de texte"
    Print #iNumber, "Une autre ligne de texte"
    Close #iNumber
    iNumber = Freefile
    Open aFile For Input As iNumber
    While Not Eof(iNumber)
        Line Input #iNumber, sLine
        If sLine <>"" Then
            sMsg = sMsg & sLine & chr(13)
        End If
    Wend
    Close #iNumber
    MsgBox sMsg
End Sub

```

EqualUnoObjects (Fonction)

Teste si deux objets UNO représentent la même instance d'un objet UNO.

EqualUnoObjects(oObj1, oObj2)

Boolean

```

Sub ExampleEqualUnoObjects
    Dim oIntrospection, oIntro2, Struct2
    Rem Copie d'objets = même instance
    oIntrospection = CreateUnoService( "com.sun.star.beans.Introspection" )
    oIntro2 = oIntrospection
    print EqualUnoObjects( oIntrospection, oIntro2 )
    Rem Copie de structures : Nouvelle instance
    Dim Struct1 as new com.sun.star.beans.Property
    Struct2 = Struct1
    print EqualUnoObjects( Struct1, Struct2 )
End Sub

```

EQV (opérateur)

Calcule l'équivalence logique de deux expressions. Dans une comparaison bit à bit, l'opérateur met à jour le bit dans le résultat si le bit correspondant existe dans les deux expressions ou n'existe pas dans les deux expressions. La présence mathématique usuelle des opérateurs est applicable comme décrite page 406 ainsi que dans la table de vérité ci dessous.

VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	VRAI
1	1	1
1	0	0
0	1	0
0	0	1

Result = Expression1 EQV Expression2

Result : Variable numérique contenant le résultat de l'opération
 Expression1, expression2 : Expressions à comparer

```
Sub ExampleEQV
  Dim vA As Variant, vB As Variant, vC As Variant, vD As Variant
  Dim vOut As Variant
  vA = 10: vB = 8: vC = 6: vD = Null
  vOut = vA > vB EQV vB > vC
  REM retourne -1
  Print vOut
  vOut = vB > vA EQV vB > vC
  REM retourne -1
  Print vOut
  vOut = vA > vB EQV vB > vD
  REM retourne 0
  Print vOut
  vOut = (vB > vD EQV vB > vA)
  REM retourne -1
  Print vOut
  vOut = vB EQV vA
  REM retourne -1
End Sub
```

Erl (**Fonction**)

Retourne le numéro de ligne à laquelle est apparue une erreur durant l'exécution.

Err

Erl

Integer

```

Sub ExampleErl
    On Error GoTo ErrorHandler
    Dim iVar as Integer
    iVar = 0
    iVar = 4 / iVar
    Exit Sub
ErrorHandler:
    Rem Erreur 11 : Division par zéro
    Rem à la ligne : 8
    Rem ....
    MsgBox "Erreur " & err & ": " & error$ + chr(13) +
        "A la ligne : " + Erl + chr(13) + Now , 16 , "Une erreur a été générée"
End Sub

```

Err (Fonction)

Retourne le numéro de la dernière erreur

Erl

Err

Integer

```

Sub ExampleErr
    On Error GoTo ErrorHandler
    Dim iVar as Integer
    iVar = 0
    iVar = 4 / iVar
    Exit Sub
ErrorHandler:
    Rem Erreur 11 : Division par zéro
    Rem à la ligne : 8
    Rem ....
    MsgBox "Erreur " & err & ": " & error$ + chr(13) +
        "A la ligne : " + Erl + chr(13) + Now , 16 , "Une erreur a été générée"
End Sub

```

Error (Fonction)

Cette fonction est censée simuler l'apparition d'une erreur.

Cependant Error errornumber As Integer ne semble pas fonctionner.

Error (Fonction)

Retourne le message d'erreur correspondant à un code donné.

Error (Expression)

String

Expression : Optionnel. Integer contenant le code d'erreur. Si omis, le dernier message est retourné.

```
Sub ExampleError
    On Error GoTo ErrorHandler
    Dim iVar as Integer
    iVar = 0
    iVar = 4 / iVar
    Exit Sub
ErrorHandler:
    REM Erreur 11 : Division par zéro
    REM à la ligne : 8
    REM ....
    MsgBox "Erreur " & err & ":" & error$ + chr(13) +
        "A la ligne : " + Erl + chr(13) + Now , 16 , "Une erreur a été générée"
End Sub
```

Cette fonction est également censée simuler l'apparition d'une erreur.
Cependant Error errornumber As Integer ne semble pas fonctionner.

Exit (Instruction)

Utilisé pour quitter les instructions de boucles, fonctions et procédures. En d'autres termes, on sort immédiatement de telles instructions. Si je suis à l'intérieur d'une procédure et que je détermine que les arguments ne sont pas corrects, je peux sortir de la procédure immédiatement.

End

Exit Do	Sort de la boucle Do...Loop courante.
Exit For	Sort de la boucle For...Next courante .
Exit Function	Sort de la fonction et poursuit l'exécution de l'appelant.
Exit Sub	Sort de la procédure et poursuit l'exécution de l'appelant.

```
Sub ExampleExit
    Dim sReturn As String
    Dim sListArray(10) as String
    Dim siStep as Single
    REM Construit le tableau ("B", "C", ..., "L")
    For siStep = 0 To 10
        REM Remplit le tableau avec des données de Test
        sListArray(siStep) = chr(siStep + 66)
    Next siStep
```

```

sReturn = LinSearch(sListArray(), "M")
Print sReturn
Exit Sub
REM Instruction inutile !
End Sub
REM Retourne l'index de l'entrée ou (LBound(sList()) - 1) si non trouvé
Function LinSearch( sList(), sItem As String ) as integer
    Dim iCount As Integer
    REM LinSearch recherche l'index de sItem dans sList()
    For iCount=LBound(sList()) To UBound(sList())
        If sList(iCount) = sItem Then
            LinSearch = iCount
            Exit Function
            REM Un bon usage de l'instruction Exit ici !
        End If
    Next
    LinSearch = LBound(sList()) - 1
End Function

```

Exp (Fonction)

Retourne la base du logarithme népérien ($e = 2.718282$) élevé à la puissance de l'argument. Autrement dit, la fonction exponentielle.

Log

Exp (Number)

Double

Number : Toute expression numérique.

```

Sub ExampleExp
    Dim d as Double, e As Double
    e = Exp(1)
    Print "e = " & e
    Print "Ln(e) = " & Log(e)
    Print "2*3 = " & Exp(Log(2.0) + Log(3.0))
    Print "2^3 = " & Exp(Log(2.0) * 3.0)
end sub

```

FileAttr (Fonction)

La première méthode d'utilisation de cette fonction sert à déterminer le mode d'accès d'un fichier utilisé lors de la commande Open. Mettre le deuxième paramètre à 1 retourne cette indication.

1	INPUT
2	OUTPUT

4	RANDOM
8	APPEND
32	BINARY

La deuxième méthode d'utilisation sert à déterminer les attributs d'un fichier MS-DOS ouvert avec l'instruction Open. Cette valeur dépend du système d'exploitation. Mettre le deuxième paramètre à 2 retourne cette indication.

L'attribut d'un fichier dépend du système d'exploitation. Il est impossible d'utiliser cette fonction sur une version 32 bits pour déterminer l'attribut MS-DOS. La valeur 0 est retournée.

Open

FileAttr (FileNumber As Integer, Attribut As Integer)

Integer

FileNumber : Numéro du fichier tel qu'utilisé pour l'ouverture par Open.

Attribut : Integer indiquant quelle information retourner. 1 = le mode d'accès, 2 = l'attribut d'accès du système de fichier.

```
Sub ExampleFileAttr
    Dim iNumber As Integer
    iNumber = Freefile
    Open "file:///c|/data.txt" For Output As #iNumber
    Print #iNumber, "Texte aléatoire"
    MsgBox AccessModes(FileAttr(#iNumber, 1 )),0,"Mode d'accès"
    MsgBox FileAttr(#iNumber, 2 ),0,"Attribut du fichier"
    Close #iNumber
End Sub
Function AccessModes(x As Integer) As String
    Dim s As String
    s = ""
    If (x AND 1) <> 0 Then s = "INPUT"
    If (x AND 2) <> 0 Then s = "OUTPUT"
    If (x AND 4) <> 0 Then s = s & " RANDOM"
    If (x AND 8) <> 0 Then s = s & " APPEND"
    If (x AND 32) <> 0 Then s = s & " BINARY"
    AccessModes = s
End Function
```

FileCopy (Instruction)

Copie un fichier. Ne peut pas copier un fichier ouvert.

FileCopy TextFrom As String, TextTo As String

TextFrom : String chemin du fichier source

TextTo : String chemin du fichier destination

```
Sub ExampleFilecopy
    Filecopy "c:\Data.txt", "c:\Temp\Data.sav"
End Sub
```

FileDateTime (Fonction)

Retourne une chaîne de caractères avec la date de création ou de dernière modification. Le format de cette information dépend de la configuration locale, "DD/MM/YYYY HH:MM:SS" sur un ordinateur configuré en français. On utilisera ce résultat avec la fonction DateValue.

FileDateTime(Text As String)

String

Text : Nom du fichier (Jokerisation interdite). La notation URL est acceptée.

```
Sub ExampleFileDateTime
    REM 23/04/2003 19:30:03
    MsgBox FileDateTime("file://localhost/C|/macro.txt")
End Sub
```

FileExists (Fonction)

Détermine si un fichier ou un répertoire existe.

FileExists(FileName As String | DirectoryName As String)

Boolean

FileName | DirectoryName : Fichier ou répertoire à rechercher (Jokerisation interdite)

```
Sub ExampleFileExists
    MsgBox FileExists("C:\autoexec.bat")
    MsgBox FileExists("file://localhost/c|/macro.txt")
    MsgBox FileExists("file:///d|/private")
End Sub
```

FileLen (Fonction)

Retourne la taille d'un fichier. Si le fichier est ouvert, la taille d'avant son ouverture est retournée. On utilisera la fonction LOF pour déterminer la taille courante d'un fichier ouvert.

FileExists(FileName As String)

Long

FileName : Nom du fichier (Jokerisation interdite).

```
Sub ExampleFileExists
    MsgBox FileLen("C:\autoexec.bat")
    MsgBox FileLen("file://localhost/c/macro.txt")
End Sub
```

FindObject (Fonction)

Donnez un nom de variable et elle retournera une référence à l'objet. Voir FindPropertyObject. L'exécution du code montré ci-dessous démontre que ceci ne fonctionne pas très bien.

```
Sub TestTheThing
    Dim oTst As Object
    Dim oDoc As Object
    oTst = FindObject("oDoc")

    REM oui
    If oTst IS oDoc Then Print "oTst et oDoc sont les mêmes"

    oDoc = ThisComponent
    oTst = FindObject("oDoc")
    REM non
    If oTst IS oDoc Then Print "oTst et oDoc sont les mêmes"
    REM non
    If oTst IS ThisComponent Then Print "oTst et ThisComponent sont les mêmes"
    REM oui
    If oDoc IS ThisComponent Then Print "oDoc et ThisComponent sont les mêmes"

    oDoc = ThisComponent
    oTst = FindObject("ThisComponent")
    REM oui
    If oTst IS oDoc Then Print "oTst et oDoc sont les mêmes"
    REM oui
    If oTst IS ThisComponent Then Print "oTst et ThisComponent sont les mêmes"
    REM oui
    If oDoc IS ThisComponent Then Print "oDoc et ThisComponent sont les mêmes"

    REM ceci montre ThisComponent
    RunSimpleObjectBrowser(oTst)
    oDoc = ThisComponent
    oTst = ThisComponent.DocumentInfo
    oTst = FindPropertyObject(oDoc, "DocumentInfo")
```

```

If IsNull(oTst) Then Print "est Null (vide)"
If oTst IS ThisComponent.DocumentInfo Then Print "Ils sont identiques"
End Sub

```

FindPropertyObject (Fonction)

Maintenant j'en ai une idée et bon sang, ceci est étrange. En plus, cela ne fonctionne pas très bien. En bref, considérez que c'est inutilisable. Un objet contient des objets de données. Par exemple, une feuille de tableur a une propriété nommée DrawPages que je peux référencer directement avec la commande ThisComponent.DrawPages. Je peux utiliser FindPropertyObject pour obtenir une référence à cet objet.

```
obj = FindPropertyObject(ThisComponent, "DrawPages")
```

Je peux maintenant accéder à l'objet DrawPages avec la variable obj. J'ai découvert que ceci était bogué !

```

Sub StrangeThingsInStarBasic
    Dim oSBObj1 As Object
    Dim oSBObj2 As Object
    Dim oSBObj3 As Object

    Set oSBObj1 = Tools
    RunSimpleObjectBrowser(oSBObj1)
    REM Nous avons également une propriété Name!!
    print oSBObj1.Name
    REM affiche @SBRTL. ?? qu'est-ce que c'est ?

    REM à propos...
    Set oSBObj2 = FindObject("Gimmicks")
    print oSBObj2.Name
    REM affiche @SBRTL de nouveau...

    REM Vous pouvez changer ce nom de propriété, mais cela est sans effet
    REM   oSBObj2.Name = "Ciao" : print oSBObj2.Name
    REM   oSBObj2.Name = "@SBRTL" : print oSBObj2.Name

    REM nécessite le chargement de la bibliothèque Gimmicks library maintenant
    GlobalScope.BasicLibraries.LoadLibrary("Gimmicks")

    REM l'autre vieille méthode, désapprouvée, non documentée, presque boguée .....

    REM userfields est un module dans Gimmicks library
    Set oSBObj3 = FindPropertyObject(oSBObj2, "Userfields")
    print (oSBObj3 Is Gimmicks.Userfields)

    REM nécessite le chargement de la bibliothèque Gimmicks library
    GlobalScope.BasicLibraries.LoadLibrary("Gimmicks")

    REM la fonction StartChangesUserfields est dans le the module Userfields
    REM un appel pleinement qualifié
    oSBObj2.Userfields.StartChangesUserfields
End Sub

```

Fix (Fonction)

Retourne la partie entière d'une expression numérique en en retirant la partie

décimale.

CInt

Fix(Expression)

Double

Expression : Expression numérique.

```
sub ExampleFix
    Print Fix(3.14159)
    REM retourne 3.
    Print Fix(0)
    REM retourne 0.
    Print Fix(-3.14159)
    REM retourne -3.
End Sub
```

For...Next (Instruction)

Instruction répétitive avec un indice auto-incrémenté.

For....Next à la Page 397.

For counter=start To end [Step step]

Bloc d'instructions

[Exit For]

Bloc d'instructions

Next [counter]

Format (Fonction)

Convertit un nombre en chaîne de caractères en appliquant une mise en forme. Plusieurs mises en formes, séparées par des virgules, peuvent être spécifiées. La première sera utilisée pour les nombres positifs, la seconde pour les négatifs, la troisième pour zéro. S'il n'y a qu'un formatage, il s'applique à tous les nombres.

0	Si un nombre à un chiffre à la position du 0, ce chiffre est affiché. Cela implique que les zéros non significatifs sont affichés. Les décimales supplémentaires sont arrondies.
#	Comme 0 mais les zéros non significatifs ne sont pas affichés.

.	L'emplacement du point détermine le nombre de chiffres à placer avant et après le séparateur décimal.
%	Multiplie le nombre par 100 et affiche le signe % à l'endroit indiqué dans la chaîne de formatage.
E- E+ e- e+	Si le format contient au moins un caractère de formatage (0 ou #) à droite du symbole, le nombre est affiché en notation scientifique. La lettre E (ou e) est insérée entre le nombre et son exposant. Le nombre de caractères de formatage à droite du symbole détermine le nombre de chiffres de l'exposant. Si l'exposant est négatif, le signe moins est affiché juste avant la valeur de l'exposant. Si l'exposant est positif, le signe n'est affiché que si explicitement écrit dans le format (E+ ou e+).
,	La virgule est le caractère représentant le séparateur de milliers. Il sépare les milliers des centaines et des unités. Ce séparateur n'est affiché que s'il est encadré par des caractères de formatage de chiffres (0 or #).
- + \$ () espace	Plus (+), moins (-), dollar (\$), espaces, ou parenthèses rentrés dans la chaîne de formatage ne sont pas interprétés, et sont donc affichés tels quels.
\	<p>L'anti-slash est le caractère d'échappement. Il permet de ne pas interpréter le caractère suivant comme un caractère de formatage mais de l'afficher tel quel. L'anti-slash n'est pas affiché à moins de le doubler (\\\).</p> <p>Les caractères devant être précédés d'un anti-slash pour ne pas être interprétés sont les caractères de formatage de date et heure (a, c, d, h, m, n, p, q, s, t, w, y, /, :), de numérique (#, 0, %, E, e, virgule, point), de chaîne de caractères (@, &, <, >, !). On peut également encadrer les caractères entre guillemets.</p>
General Number	Les nombres sont affichés tels quels.
Currency	Le nombre est affiché au format monétaire en accord avec la configuration locale du poste.
Fixed	Au moins un chiffre est affiché devant le séparateur décimal.
Standard	Le nombre est affiché au format décimal en accord avec la configuration locale du poste.
Scientific	Le nombre est affiché au format scientifique.

Aucun formatage n'est effectué si le paramètre n'est pas un nombre. Une chaîne vide est retournée.

Dans la version 1.0.3.1, Format(123.555, ".##") retourne ".12356" ce qui peut être considéré comme un bogue.
Changer le format en "#.##" corrige le problème.

La notation « Scientific » ne fonctionne pas correctement.

La notation « Currency » ne peut pas utiliser les caractères d'échappement et comporte des erreurs de formatage (US : place le symbole monétaire à droite).

Format (Number [, Format As String])

String

Number : Expression numérique à formater.

Format : Format désiré. Si omis, la fonction se comporte comme la fonction Str.

```
Sub ExampleFormat
```

```
REM Dépend de la configuration locale
```

```
MsgBox Format(6328.2, "##,##0.00")
REM = 6 328,20
MsgBox Format(123456789.5555, "##,##0.00")
REM = 123 456 789,56
MsgBox Format(0.555, ".##")
REM ,56
MsgBox Format(123.555, "#.##")
REM 123,56
MsgBox Format(0.555, "0.##")
REM 0,56
MsgBox Format(0.1255555, "%#.##")
REM %12,56
MsgBox Format(123.45678, "##E-####")
REM 12E1
MsgBox Format(.0012345678, "0.0E-####")
REM 1,2E3 (dysfonctionnement)
MsgBox Format(123.45678, "#.e-##")
REM 1,e2
MsgBox Format(.0012345678, "#.e-##")
REM 1,e3 (dysfonctionnement)
MsgBox Format(123.456789, "#.## est ##")
REM 123.45 donne 679 (étrange)
MsgBox Format(8123.456789, "General Number")
REM 8123,456789
MsgBox Format(8123.456789, "Fixed")
REM 8 123,46
MsgBox Format(8123.456789, "Currency")
REM 8 123.46 € (dysfonctionnement US)
MsgBox Format(8123.456789, "Standard")
REM 8 123.46
MsgBox Format(8123.456789, "Scientific")
REM 8,12E03
MsgBox Format(0.00123456789, "Scientific")
REM 1,23E03 (dysfonctionnement)
```

```
End Sub
```

FreeFile (Fonction)

Retourne l'index de fichier disponible pour l'ouverture d'un fichier. Assure que cet index est bien disponible et pas utilisé par un autre fichier.

Open, EOF, Kill, et Close.

FreeFile

Integer

Voir l'exemple de la fonction Close.

FreeLibrary (Fonction)

Libère les ressources d'une DLL chargée par l'instruction Declare. La DLL sera automatiquement rechargée si une de ses fonctions est de nouveau appelée. Seules les DLL chargées au moment de l'exécution de la macro doivent être libérées.

Declare

FreeLibrary (LibName As String)

LibName : Nom de la DLL.

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep" ( long )
Sub ExampleDeclare
    Dim IValue As Long
    IValue = 5000
    MyMessageBeep( IValue )
    FreeLibrary("user32.dll")
End Sub
```

Function (Instruction)

Définit une fonction utilisateur, par opposition à une procédure (Sub). On peut voir une fonction comme contenant intrinsèquement une valeur (de tout type).

Sub

Function Name[(VarName1 [As Type][, VarName2 [As Type][,...]])] [As Type]

```
Bloc d'instructions  
[Exit Function]  
Bloc d'instructions  
End Function
```

Une valeur du type de la fonction.

```
Function IsWhiteSpace(iChar As Integer) As Boolean  
    Select Case iChar  
        Case 9, 10, 13, 32, 160  
            IsWhiteSpace = True  
        Case Else  
            IsWhiteSpace = False  
    End Select  
End Function
```

Get (Instruction)

Lit l'enregistrement d'un fichier indexé (Random) ou une séquence d'octets d'un fichier binaire. Si le paramètre de position est omis, la lecture s'effectue à partir de la position courante. Pour les fichiers ouverts en mode binaire, cette position est exprimée en octets.

PUT

Get [#] FileNumber As Integer, [Position], Variable

FileNumber : Integer Index du fichier ouvert.

Position : Pour les fichiers ouverts en mode « Random », c'est le numéro de l'enregistrement à lire.

Variable : Variable à lire. Un type Objet ne peut pas être utilisé.

```
REM Ne fonctionne pas !  
REM La position semble ne pas être optionnelle pour Get  
  
Sub ExampleRandomAccess2  
    Dim iNumber As Integer, aFile As String  
    Dim sText As Variant  
    REM de type variant obligatoirement  
    aFile = "c:\data1.txt"  
    iNumber = Freefile  
    Open aFile For Random As #iNumber Len=5  
    Seek #iNumber,1  
    REM On positionne au début  
    Put #iNumber,, "1234567890"  
    REM On remplit la ligne avec du texte  
    Put #iNumber,, "ABCDEFGHIJ"  
    Put #iNumber,, "abcdefghijkl"
```

```

REM Voilà à quoi ressemble le fichier !
REM 08 00 0A 00 31 32 33 34 35 36 37 38 39 30 08 00 ....1234567890...
REM 0A 00 41 42 43 44 45 46 47 48 49 4A 08 00 0A 00 ..ABCDEFGHIJ...
REM 61 62 63 64 65 66 67 68 69 6A 00 00 00 00 00 abcdefghij

```

```

Seek #iNumber,1
Get #iNumber,,sText
Print "on open:" & sText
Close #iNumber
iNumber = Freefile
Open aFile For Random As #iNumber Len=5
Get #iNumber,,sText
Print "réouvert: " & sText
Put #iNumber,,"ZZZZZ"
Get #iNumber,1,sText
Print "un autre Get "& sText
Get #iNumber,1,sText
Put #iNumber,20,"Le contenu de l'enregistrement 20"
Print Lof(#iNumber)
Close #iNumber
End Sub

```

GetAttr (Fonction)

Retourne un nombre identifiant le type d'un « fichier ». Ces attributs sont un sur-ensemble de ceux utilisés dans la fonction Dir .

Attribut	Description
0	Normal
1	Lecture seule (Read only)
2	Caché (Hidden)
4	Système
8	Nom de volume
16	Répertoire (Directory)
32	Bit d'archive. Le fichier a changé depuis la dernière sauvegarde.

Dir

Ne marche pas avec la version 1.0.3.1. A tester avec la version que vous utilisez.

GetAttr (Text As String)

Integer

Text : String contenant un nom de fichier non-ambigu – La notation URL est acceptée.

```

Sub ExampleGetAttr
    REM devrait retourner "Read-Only Hidden System Archive"
    REM retourne " Read-Only"
    Print FileAttributeString(GetAttr("C:\IO.SYS"))
    REM devrait retourner "Archive" mais retourne "Normal"
    Print FileAttributeString(GetAttr("C:\AUTOEXEC.BAT"))
    REM "Directory" (répertoire)
    Print FileAttributeString(GetAttr("C:\WINDOWS"))
End Sub
Function FileAttributeString(x As Integer) As String
    Dim s As String
    If (x = 0) Then
        s = "Normal"
    Else
        s = ""
        If (x AND 16) <> 0 Then s = "Directory"
        If (x AND 1) <> 0 Then s = s & " Read-Only"
        If (x AND 2) <> 0 Then s = " Hidden"
        If (x AND 4) <> 0 Then s = s & " System"
        If (x AND 8) <> 0 Then s = s & " Volume"
        If (x AND 32) <> 0 Then s = s & " Archive"
    End If
    FileAttributeString = s
End Function

```

GetProcessServiceManager (Fonction)

Accède au « central Uno service manager ». Cette fonction est requise si on doit instancier un service avec CreateInstance et contenant des arguments.

```
oServiceManager = GetProcessServiceManager()
```

Object

```

REM trouver un meilleur exemple contenant un appel avec argument
oServiceManager = GetProcessServiceManager()
oIntrospection = oServiceManager.createInstance("com.sun.star.beans.Introspection");
REM C'est la même chose que l'instruction suivante
oIntrospection = CreateUnoService("com.sun.star.beans.Introspection")

```

GetSolarVersion (Fonction)

Retourne le numéro interne de " build " (compilation) de la version courante de OpenOffice.org. Vous pouvez écrire votre macro pour contourner des bugs connus des différentes versions. Malheureusement, la fonction GetSolarVersion reste souvent la même lorsque les versions changent. La version 1.0.3.1 retourne "641" et 1.1RC3 retourne " 645 ", mais cela n'est pas assez précis. Le code suivant retourne la version actuelle de OOo.

```

Function OOOVersion() As String
    REM Retrouve la version courante de OOo
    REM Auteur : Laurent Godard
    REM e-mail : listes.godard@laposte.net

```

```

Dim aSettings, aConfigProvider
Dim aParams2(0) As new com.sun.star.beans.PropertyValue
Dim sProvider$, sAccess$
sProvider = "com.sun.star.configuration.ConfigurationProvider"
sAccess = "com.sun.star.configuration.ConfigurationAccess"
aConfigProvider = createUnoService(sProvider)
aParams2(0).Name = "nodepath"
aParams2(0).Value = "/org.openoffice.Setup/Product"
aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())

OOVersion=aSettings.getbyname("ooSetupVersion")
End Function

```

s = GetSolarVersion()

String

```

Sub ExampleGetSolarVersion
REM pour la 1.0.3.1, ceci vaut "641"
Print GetSolarVersion()
End Sub

```

GetSystemTicks Function

Retourne le nombre de « Ticks » fourni par le système d'exploitation. Le nombre de ticks retourné sur un intervalle de temps donné dépend toujours du système d'exploitation.

GetSystemTicks()

Long

Cet exemple tente de mesurer le nombre de ticks par seconde. Sur WinXP et Ooo 1.0.3.1, on calcule 1000 ticks par seconde.

```

Sub ExampleGetSystemTicks
Dim ITick As Long, IMillisToWait As Long
Dim ISecsToWait As Long, ITicksPerSec As Long
ISecsToWait = 60
IMillisToWait = ISecsToWait * 1000
ITick = GetSystemTicks()
wait(IMillisToWait)
ITick = (GetSystemTicks() - ITick)
ITicksPerSec = ITick / ISecsToWait
MsgBox "Chaque seconde représente " & ITicksPerSec & " Ticks"
End Sub

```

GlobalScope (Objet)

Les boîtes de dialogues et macros sont organisées en bibliothèques (Library).

Une bibliothèque peut contenir plusieurs macros et/ou boîtes de dialogues. En Basic, le conteneur des bibliothèques est appelé “BasicLibraries” et celui des boîtes de dialogues “DialogLibraries”. Ces bibliothèques existent à la fois au niveau global de l'application et au niveau du document. Pour appeler les conteneurs de bibliothèque globaux, il faut utiliser l'objet GlobalScope.

GlobalScope

```
REM Appel Dialog1 dans la bibliothèque Standard du document  
oDlgDesc = DialogLibraries.Standard.Dialog1  
REM Appel Dialog2 de la bibliothèque d'application Library1  
oDlgDesc = GlobalScope.DialogLibraries.Library1.Dialog2
```

GoSub (Instruction)

Transfère l'exécution vers une portion de code délimitée par un label dans la même procédure ou fonction. Les instructions suivant le label sont exécutées jusqu'à rencontrer l'instruction Return. Le programme continue alors son exécution à l'instruction suivant l'appel du GoSub.

On évite généralement d'utiliser une telle instruction que l'on remplacera avantageusement par un appel de procédure ou de fonction.

GoSub provient de vieilles versions du BASIC. L'emploi de GoSub est fortement déconseillé car il induit du code peu lisible et difficile à maintenir. L'utilisation de fonctions et procédures est recommandée.

Sub/Function

REM Instructions

GoSub Label

REM Instructions

GoSub Label

Exit Sub/Function

Label :

REM Blocs d'instruction

Return

End Sub/Function

```
Sub ExampleGoSub  
Print "Avant le gosub"  
GoSub SillyLabel  
Print "Après le gosub"  
Exit Sub
```

```
SillyLabel:  
    Print "Après le label Silly"  
    Return  
End Sub
```

GoTo (Instruction)

Transfère l'exécution vers une portion de code délimitée par un label dans la même procédure ou fonction. Le fil d'exécution principal est perdu.
On évite généralement d'utiliser une telle instruction. Son seul intérêt peut se trouver dans la gestion des erreurs.

On error goto

GoTo provient de vieilles versions du BASIC. L'emploi de GoTo est fortement déconseillé car il induit du code peu lisible et difficile à maintenir. L'utilisation de fonctions et procédures est recommandée.

Sub/Function

REM Instructions

GoTo Label

REM Instructions Jamais exécutées

Exit Sub/Function

Label :

Bloc d'instructions

End Sub/Function

```
Sub ExampleGoTo  
    Print "Avant le goto"  
    GoTo SillyLabel  
    Print "Après le goto"  
    REM Jamais exécuté  
    Exit Sub  
    REM Jamais exécuté  
SillyLabel:  
    Print "Après le label Silly"  
End Sub
```

Green (Fonction)

Les couleurs sont représentées par un entier de type Long. Cette fonction retourne la valeur de la composante verte de la couleur passée en argument.
Voir également les fonctions RGB, Red et Blue.

Green (Color As Long)

Integer compris entre 0 et 255.

Color : Entier Long représentant une couleur.

```
Dim lColor As Long
lColor = RGB(255,10,128)
MsgBox "La couleur " & lColor & " est composée de:" & Chr(13) &
    "Rouge = " & Red(lColor) & Chr(13)&
    "Vert= " & Green(lColor) & Chr(13)&
    "Bleu= " & Blue(lColor) & Chr(13) , 64,"Couleurs"
End Sub
```

HasUnoInterfaces (Fonction)

teste si l'objet supporte une interface UNO spécifique. Retourne True si toutes les interfaces spécifiées sont supportées.

HasUnoInterfaces(oTest, Uno-Interface-Name 1 [, Uno-Interface-Name 2, ...])

Boolean

oTest : Objet UNO à tester.

Uno-Interface-Name : Liste des noms des interfaces UNO.

```
Sub CloseOpenDocument
    If HasUnoInterfaces(oDoc, "com.sun.star.util.XCloseable") Then
        oDoc.close(true)
    Else
        oDoc.dispose
    End If
End Sub
```

Hex (Fonction)

Retourne la valeur hexadécimale d'un nombre. Si l'argument n'est pas d'un type numérique, il est converti (si possible).

Hex(Number)

String

Number : Nombre à représenter en hexadécimal. Peut être une chaîne de caractères.

```
Sub ExampleHex
    Dim i1%, i2%, iNum%, s$, sFormat$, sTemp$
    iNum = 0
    s = ""
    For i1=0 To 15
        For i2=0 To 15
            s = s & " " & PrependChar(Hex(iNum), "0", 2)
            iNum = iNum + 1
        Next
        s = s & Chr(13)
    Next
    MsgBox s, 64, "Table en Hexa"
    Print Hex("64")
End Sub
Function PrependChar(s$, sPrependString$, iTotLen%) As String
    If Len(s) < iTotLen Then
        PrependChar = String(iTotLen - Len(s), sPrependString) & s
    Else
        PrependChar = s
    End If
End Function
```

Hour (Fonction)

Extrait l'heure d'une valeur de temps retournée par TimeSerial ou TimeValue.

Hour(Number)

Integer

Number : Expression numérique contenant une valeur de temps.

```
Sub ExampleHour
    Print "L'heure courante est " & Hour( Now )
    Print Hour(TimeSerial(14,08,12))
    Print Hour(TimeValue("14:08:12"))
End Sub
```

If ... Then ... Else (Instruction)

Permet d'exécuter un bloc d'instructions suivant qu'une condition est évaluée à True ou False. Bien que l'on puisse utiliser les instructions GoTo ou GoSub pour sortir d'un If (!!!!!!!!), on ne peut les utiliser pour rentrer dans un bloc d'instructions contenu dans un If.

```

If condition=True Then
    Bloc d'instructions
[Elself condition=True Then]
    Bloc d'instructions
[Else]
    Bloc d'instructions
End If

```

```

If condition Then Bloc d'instructions
If (condition=False) Then Bloc d'instructions

```

```

Sub ExampleIf
    Dim i%
    i% = 4
    If i < 5 Then
        Print "i est plus petit que 5"
        If i = 4 Then Print "i est égal à 4"
        If i < 3 Then
            Print "i est plus petit que 3"
        End If
    Elseif i = 5 Then
        Print "i est égal à 5"
    Else
        Print "i est plus grand que 5"
    End If
End Sub

```

IIF (Instruction)

Retourne un résultat suivant que la condition spécifiée est évaluée à True ou False. Bien que cette commande soit très appréciable, elle semble avoir quelques petits dysfonctionnements avec la version 1.0.3.1

IIf (Expression, ExpressionTrue, ExpressionFalse)

ExpressionTrue ou ExpressionFalse

Expression : Expression conditionnelle à évaluer.

ExpressionTrue : Valeur retournée si la condition est True (Vraie)

ExpressionFalse : Valeur retournée si la condition est False (Fausse)

```

Sub IIfExample
    Print IIf(3>4,"Oui", "Non")

```

```
REM Non
Print IIf(4>2,"Oui", "Non")
REM Oui
End Sub
```

Imp (Opérateur)

Calcule l'implication logique de deux expressions

En cours de rédaction – Non traduit

Result = Expression1 Imp Expression2

```
Sub ExampleImp
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vOut as Variant
A = 10: B = 8: C = 6: D = Null
vOut = A > B Imp B > C
REM retourne -1
vOut = B > A Imp B > C
REM retourne 0
vOut = A > B Imp B > D
REM retourne -1
vOut = (B > D Imp B > A)
REM retourne 0
vOut = B Imp A
REM retourne -3
End Sub
```

Input (Instruction)

L'instruction Input est utilisée pour lire séquentiellement les données d'un fichier ouvert et les affecter à une ou plusieurs variables. Le retour chariot (Asc=13), la fin de ligne (Asc=10) et la virgule agissent comme délimiteurs. Quand une valeur numérique est lue, l'espace est également utilisé comme délimiteur. Lire une chaîne non numérique dans une variable numérique met sa valeur à 0.

Il n'est pas possible de lire les virgules et les guillemets avec cette instruction. Vous devrez alors utiliser l'instruction LineInput.

Open, Line Input#, Close, Eof, Get

Input #**FileNumber** var1[, var2[, var3[,...]]]

FileNumber : Indicateur de fichier utilisé lors de l'instruction Open.
var : Variables de type string ou numérique dans lesquelles mettre le contenu de ce qui est lu.

??

InputBox (Fonction)

Affiche une demande à l'utilisateur dans une boite de dialogue. L'annulation retourne une chaîne vide. Si aucune position n'est spécifiée, la boite est centrée à l'écran.

InputBox (Msg [, Title[, Default[, x_pos, y_pos As Integer]]])

String

Msg : Message à afficher.

Title : Titre à afficher dans barre la fenêtre.

Default : Chaîne réponse par défaut.

x_pos : Position horizontale absolue en Twips.

y_pos : Position verticale absolue en Twips.

```
Sub ExampleInputBox
    Dim s$
    s = InputBox ("Message","Titre", "défaut")
    MsgBox ( s , 64, "Confirmation de la phrase")
End Sub
```

InStr (Fonction)

Retourne la position d'une chaîne dans une autre. Si la chaîne n'est pas trouvée, retourne 0.

Dans la version 1,1RC2, la variable retournée est de type Integer mais la valeur potentiellement retournée peut être supérieure car une String peut avoir une longueur de 64 K. Une valeur négative est alors retournée si la valeur de la position est trop grande.

```
Sub BugInStr
    Dim b$, i&
    b$ = String(40000, "a") & "|"
    REM le caractère 40,001 est un "|"
    i = instr(b, "|")
    REM -25535
    MsgBox cstr(i) & " ou " & (65536 + i)
    REM -25535 ou 40001
End Sub
```

InStr([Start As Integer,] Text1 As String, Text2 As String[, Compare])

Integer

Start : Optionnel - Position du début de la recherche. Par défaut 1, début de la chaîne.

Text1 : Chaîne dans laquelle effectuer la recherche.

Text2 : Chaîne à rechercher.

Compare : Si 1, recherche indépendante de la casse, 0 (par défaut), recherche binaire.

```
Sub ExampleInStr
    Dim s$
    s = "SbxInteger getTruck(SbxLong)"
    RemoveFromString(s, "Sbx")
    Print s
End Sub

REM Efface toutes les occurrences bad$ dans s$
REM modifie la chaîne s$
Sub RemoveFromString(s$, bad$)
    Dim i%
    i = InStr(s, bad)
    Do While i > 0
        Mid(s, i, Len(bad), "")
        i = InStr(i, s, bad)
    Loop
End Sub
```

On ne peut pas utiliser l'option « Compare » si on utilise l'option « Start ».

Int (Fonction)

Retourne le premier entier inférieur à l'argument. La valeur absolue de cet entier est donc plus petite pour les nombres positifs et plus grande pour les négatifs.

CInt, Fix

Int (Number)

Double

Number : Toute expression numérique valide.

Exemple :

```
Sub ExampleINT
    Print " " & Int(3.14159) & " " & Fix(3.14)
    REM 3 3
    Print " " & Int(0) & " " & Fix(0)
    REM 0 0
    Print " " & Int(-3.14159) & " " & Fix(-3.1415)
    REM -4 -3
```

```
Print " " & Int(2.8) & " " & Fix(2.8)
REM 2 2
End Sub
```

-3.4 est arrondi en -4. Utiliser Fix si on veut la partie entière.

IsArray (Fonction)

Teste si une variable est un tableau.

IsArray(Var)

boolean

Var : Toute variable à tester à condition qu'elle soit déclarée en tant que tableau.

```
Sub ExampleIsArray
    Dim sDatf(10) as String, i
    Print IsArray(sDatf())
REM True
    Print IsArray(i())
REM False
End Sub
```

IsDate (Fonction)

Teste si un nombre ou texte peut être converti en Date.

IsDate(Expression)

Booléen

Expression : toute expression chaîne ou numérique à tester.

```
Sub ExampleIsDate
    Print IsDate("12.12.1997")
REM True
    Print IsDate("12121997")
REM False
End Sub
```

IsEmpty (Fonction)

Teste si une variable Variant contient la valeur « Empty », indiquant que la variable n'a pas été initialisée.

« Object, Variant, Empty et Null » .

IsEmpty(Var)

Booléen

Var : la variable à tester

```
Sub ExampleIsEmpty
    Dim v1 as Variant, v2 As Variant, v3 As Variant
    v2 = Null : v3 = "hello"
    Print IsEmpty(v1)
    REM True
    Print IsEmpty(v2)
    REM False
    Print IsEmpty(v3)
    REM False
    v2 = Empty
    REM ?? Supprimé après la version 1.0.3.1
    Print IsEmpty(v2)
    REM Devrait renvoyer True (Vrai)
End Sub
```

IsMissing (Fonction)

Teste si une procédure ou une fonction a été appelée avec ou sans un paramètre optionnel. Le paramètre doit être déclaré avec le mot clé « Optional » pour que cela fonctionne. A partir de la version 1.0.3.1, il y a eu apparition d'erreurs mineures comme mentionné dans la section 11.3.1 sur les paramètres optionnels.

IsMissing(var)

Booléen

Var : Variable à tester

```
Function FindCreateNumberFormatStyle (sFormat As String, Optional doc, Optional locale)
    Dim oDocument As Object
    Dim aLocale as new com.sun.star.lang.Locale
```

```

Dim oFormats As Object
REM S'il n'a pas été envoyé par l'appel, alors on utilise ThisComponent
oDocument = IIf(IsMissing(doc), ThisComponent, doc)
oFormats = oDocument.getNumberFormats()
....
End Function

```

IsNull (Fonction)

Teste si un Variant ou un Objet contient la valeur spéciale « Null » indiquant que la variable ne contient aucune valeur. Un Objet non initialisé est Null, un Variant non initialisé est Empty (Vide), mais il peut être initialisé et contenir la valeur Null.

IsEmpty, macro à inclure GetSomeObjInfo

IsNull(Var)

Booléen

Var : variable à tester

```

Sub ExampleIsNull
    Dim v1 as Variant, v2 As Variant, v3 As Variant, o As Object
    v2 = Null : v3 = "hello"
    Print IsNull(v1)
    REM False
    Print IsNull(v2)
    REM True
    Print IsNull(v3)
    REM False
    v3 = Null
    Print IsNull(v3)
    REM True
    Print IsNull(o)
    REM True
End Sub

```

IsNumeric (Fonction)

Teste si l'expression passée en argument est un nombre ou pourrait être convertie en nombre.

IsNumeric(Var)

Booléen

Var : toute expression à tester

```
Sub ExemplesNumeric
    Dim v1, v2, v3
    v1 = "abc" : v2 = "123" : v3 = 4
    Print IsNumeric(v1)
    REM False
    Print IsNumeric(v2)
    REM True
    Print IsNumeric(v3)
    REM True
    Print IsNumeric("123x")
    REM False
End Sub
```

IsObject (Fonction)

Selon la documentation en ligne, cette fonction teste si l'objet transmis est un objet OLE. Après un coup d'œil au code source et quelques essais, il s'avère que cette fonction retourne également True pour tout objet régulier.

Macro à inclure GetSomeObjInfo

IsObject(ObjectVar)

Booléen

ObjectVar : Toute variable à tester

```
Sub ExampleIsObject
    Dim o As Object, s AS String
    Print IsObject(o)
    REM True
    Print IsObject(s)
    REM Erreur d'exécution : objet non initialisé
End Sub
```

IsUnoStruct (Fonction)

Renvoie True si l'objet transmis en paramètre est un objet UNO. L'aide en ligne indique à tort que le paramètre peut être un nom plutôt qu'un objet.

macro à inclure GetSomeObjInfo.

IsUnoStruct(var)

Booléen

Var : objet à tester

```
Sub ExampleIsUnoStruct
    Dim o As Object, s AS String
    Dim aProperty As New com.sun.star.beans.Property
    Print IsUnoStruct(o)
    REM False
    Print IsUnoStruct("com.sun.star.beans.Property")
    REM False
    Print IsUnoStruct(aProperty)
    REM True
End Sub
```

Kill (Fonction)

Efface un fichier du disque. Toute notation de fichier peut être utilisée, mais les caractères génériques ne sont pas acceptés.

Kill(Nom_de_fichier)

Aucune

Nom_de_fichier : nom du fichier à effacer.

```
Sub ExampleKill
    Kill "C:\datafile.dat"
End Sub
```

LBound (Fonction)

Renvoie l'indice de début d'un tableau. Un tableau ne commence pas obligatoirement à l'indice 0.

LBound(ArrayName [, Dimension])

Entier (Integer)

ArrayName : Nom du tableau

Dimension : entier indiquant quelle dimension est recherchée. Par défaut, la première dimension est renvoyée.

```
Sub ExampleUboundLbound
    Dim a1(10 to 20) As String, a2 (10 to 20,5 To 70) As String
    print "(" & LBound(a1()) & ", " & UBound(a1()) & ")"
    REM (10, 20)
```

```
print "(" & LBound(a2()) & ", " & UBound(a2()) & ")"
REM (10, 20)
print "(" & LBound(a2(),1) & ", " & UBound(a2(),1) & ")"
REM (10, 20)
print "(" & LBound(a2(),2) & ", " & UBound(a2(),2) & ")"
REM (5, 70)
End Sub
```

LCase (Fonction)

Retourne la valeur de l'argument en minuscules.

LCase (String)

String

String : Chaîne à retourner en minuscules.

```
Sub ExampleLCase
    Dim s$
    s = "Las Vegas"
    Print LCase(s)
    REM "las vegas"
    Print UCASE(s)
    REM "LAS VEGAS"
end Sub
```

Left (Fonction)

Retourne les n caractères à gauche d'une chaîne.

Dans la version 1.1RC2, le paramètre de Left est un Integer alors que la chaîne ne peut être longue que de 64K.

Left(String, Integer)

String

String : Expression chaîne

Integer : Nombre de caractères à retourner. Si 0, une chaîne de longueur nulle est retournée.

```
Print Left("123456789", 2)
REM Affiche 12
```

Len (Fonction)

Retourne le nombre de caractères (la longueur) d'une chaîne, le nombre d'octets nécessaires à stocker une variable.

Len(Text As String)

Long

Text : Expression chaîne ou une variable d'un autre type.

```
Sub ExampleLen
    Dim s$, i%
    s = "123456"
    i = 7
    Print Len(s)
    REM 6
    Print Len(i)
    REM 1
    Print Len(1134)
    REM 4
    Print Len(1.0/3)
    REM 17
End Sub
```

Let (Mot clé)

Mot clé optionnel indiquant qu'une valeur doit être assignée à une variable (rarement utilisé).

[Let] VarName=Expression

Aucune

VarName : Variable à laquelle la valeur doit être attribuée.

```
Sub ExampleLet
    Dim s$
    Let s = "Las Vegas"
End Sub
```

Line Input (Instruction)

Lit des chaînes de caractères depuis un fichier texte séquentiel vers une variable. Vous devez d'abord ouvrir le fichier avec l'instruction Open. Les variables sont lues ligne par ligne jusqu'au premier retour chariot (code ASCII 13) ou changement de ligne (code ASCII 10). Le caractère de fin de ligne n'est pas inclus dans la variable de lecture.

Line Input #*FileNumber* As Integer, *Var* As String

Aucune

FileNumber : Numéro du fichier ouvert depuis lequel les variables doivent être lues.

var : Variable utilisée pour stocker le résultat.

Voir l'exemple à la page .

Loc (Fonction)

La fonction Loc retourne la position courante dans un fichier ouvert. Si elle est utilisée pour un fichier à accès direct, elle retourne le numéro du dernier enregistrement auquel on a accédé. Pour un fichier séquentiel, la fonction retourne la position dans le fichier divisée par 128. Pour un fichier binaire, la position du dernier octet lu ou écrit est retournée (À vérifier).

Loc(*FileNumber*)

Long

FileNumber : Expression numérique contenant le numéro d'un fichier ouvert.

??

Lof (Fonction)

Lof retourne la taille d'un fichier en octets. Pour obtenir la longueur d'un fichier non ouvert, utiliser plutôt la fonction FileLen.

Lof(*FileNumber*)

Long

FileNumber : Expression numérique contenant le numéro d'un fichier ouvert.

À VÉRIFIER

```
Sub ExampleRandomAccess
    Dim iNumber As Integer
    Dim sText As Variant
    REM doit être un Variant
    Dim aFile As String
    aFile = "c:\data.txt"
    iNumber = Freefile
    Open aFile For Random As #iNumber Len=32
    Seek #iNumber,1
    REM Position de départ
    Put #iNumber,, "C'est la première ligne de texte"
    REM Rempli avec du texte
    Put #iNumber,, "C'est la seconde ligne de texte"
    Put #iNumber,, "C'est la troisième ligne de texte"
    Seek #iNumber,2
    Get #iNumber,,sText
    Print sText
    Close #iNumber
    iNumber = Freefile
    Open aFile For Random As #iNumber Len=32
    Get #iNumber,2,sText
    Put #iNumber,, "C'est une nouvelle ligne de texte"
    Get #iNumber,1,sText
    Get #iNumber,2,sText
    Put #iNumber,20,"C'est le texte de l'enregistrement n° 20"
    Print Lof(#iNumber)
    Close #iNumber
End Sub
```

Log (Fonction)

Retourne le logarithme naturel d'un nombre. Le logarithme naturel est le logarithme en base e, qui est une constante de valeur approximative 2,718282... Le calcul du logarithme en base n quelconque est donné en divisant le logarithme naturel du nombre par le logarithme naturel de n, par la formule $\text{Logn}(x) = \text{Log}(x) / \text{Log}(n)$.

Log(Number)

Double

Number : Expression numérique dont on veut calculer le logarithme naturel.

```
Sub ExampleLogExp
```

```

Dim a as Double
Dim const b1=12.345e12
Dim const b2=1.345e34
a=Exp( Log(b1)+Log(b2) )
MsgBox "" & a & chr(13) & (b1*b2) ,0,"Multiplication via le logarithme"
End Sub

```

Loop (Instruction)

L'instruction Loop est utilisée pour répéter des instructions tant qu'une condition est True (vraie), ou jusqu'à ce qu'une instruction soit vraie. Voir le traitement de boucles Do... à la page 398.

Do [{While | Until} condition = True]

bloc d'instructions

[Exit Do]

bloc d'instructions

Loop

Do

bloc d'instructions

[Exit Do]

bloc d'instructions

Loop [{While | Until} condition = True]

```

Sub ExampleDoLoop
    Dim sFile As String, sPath As String
    sPath = "c:\" : sFile = Dir$( sPath ,22)
    If sFile <> "" Then
        Do
            MsgBox sFile
            sFile = Dir$
            Loop Until sFile = ""
        End If
    End Sub

```

LSet (Instruction)

Lset permet de justifier à gauche une chaîne de caractères à l'intérieur de l'espace utilisé par une autre chaîne. Toutes les positions restantes à gauche seront remplies par des espaces. Si la nouvelle chaîne ne tient pas dans l'ancienne, elle sera tronquée. ?? Ceci ne marche pas pour la version 1.0.3.1. Lset permet également de remplacer des données depuis un type de données utilisateur vers un autre. Cela utilise tous les octets d'une structure de données et les remplace par les autres, en ignorant la structure sous-jacente. Ceci est

actuellement d'une utilité réduite, sachant que Oo Basic ne supporte pas les types de données définis par un utilisateur.

LSet Var As String = Text

LSet Var1 = Var2

Var : Toute variable de type chaîne, dans laquelle le texte doit être aligné à gauche.

Text : Le texte à aligner à gauche.

Var1 : Nom de la variable de type utilisateur destination.

Var2 : Nom de la variable de type utilisateur source.

```
Sub ExampleLSet
    Dim sVar As String, sExpr As String
    sVar = String(40,"*")
    sExpr = "SBX"
    REM Aligné à gauche "SBX" dans la chaîne de référence de 40 caractères de long
    LSet sVar = sExpr
    Print ">"; sVar; "<"
    REM ">SBX<" Ne marche pas, devrait contenir les espaces.
    sVar = String(5,"*")
    sExpr = "123456789"
    LSet sVar = sExpr
    Print ">"; sVar; "<"
    REM ">12345<"
End Sub
```

LTrim (Fonction)

Retire tous les espaces au début d'une expression chaîne.

LTrim(Text)

String

Text : Toute expression de type chaîne

```
Sub ExampleSpaces
    Dim sText2 As String,sText As String,sOut As String
    sText2 = " <*Las Vegas*> "
    sOut = """+sText2 +"""+ Chr(13)
    sText = Ltrim(sText2)
    REM sText = <*Las Vegas*> "
    sOut = sOut + """+sText +"""+ Chr(13)
    sText = Rtrim(sText2)
    REM sText = " <*Las Vegas*> "
    sOut = sOut + """+ sText +"""+ Chr(13)
```

```
sText = Trim(sText2)
REM sText = " <*Las Vegas*> "
sOut = sOut +"""+ sText +"""
MsgBox sOut
End Sub
```

Private (mot-clé)

Le mot clé Private est utilisé pour déclarer une variable comme privée au module. Si une variable est déclarée avec le mot-clé Dim, elle est considérée comme privée. Voir la section Dim pour la description de la syntaxe.

Dim, Public

Private Name_1 [(start To end)] [As VarType][, Name_2 [(start To end)] [As VarType][,...]]

```
Private iPriv As Integer
Sub ExamplePublic
    iPriv = 1
    Call CalledSub
End Sub
Sub CalledSub
    Print iPriv    REM 1
End Sub
```

Public (mot-clé)

Le mot clé Public est utilisé pour déclarer une variable comme accessible par tous les modules. Si une variable est déclarée avec le mot-clé Dim, elle est considérée comme privée. Voir la section Dim pour la description de la syntaxe.

Dim, Private

Public Name_1 [(start To end)] [As VarType][, Name_2 [(start To end)] [As VarType][,...]]

```
Public iPub As Integer
Sub ExamplePublic
    iPub = 1
    Call CalledSub
End Sub
Sub CalledSub
    Print iPub    REM 1
End Sub
```

Red (Fonction)

Les couleurs sont représentées par un entier de type Long. Cette fonction

retourne la valeur de la composante rouge de la couleur passée en argument.
Voir également les fonctions RGB, Blue et Green.

Red (Color As Long)

Integer compris entre 0 et 255.

Color : Entier Long représentant une couleur.

```
Dim IColor As Long
IColor = RGB(255,10,128)
MsgBox "La couleur " & IColor & " est composée de:" & Chr(13) &
    "Rouge = " & Red(IColor) & Chr(13)&
    "Vert= " & Green(IColor) & Chr(13)&
    "Bleu= " & Blue(IColor) & Chr(13) , 64,"Couleurs"
End Sub
```

Shell Function

Lance une application externe. Le style de fenêtre de l'application démarrée peut optionnellement être paramétré avec les valeurs suivantes :

0	Focus sur une fenêtre cachée du programme.
1	Focus sur la fenêtre d'application au format standard.
2	Focus sur la fenêtre d'application minimisée.
3	Focus sur une fenêtre d'application maximisée.
4	Taille de fenêtre standard d'application, sans le focus.
6	Taille de fenêtre d'application minimisée, mais le focus reste sur la fenêtre active.
10	Affichage plein écran.

Le programme est censé démarrer et continuer à fonctionner en arrière plan sauf si le dernier paramètre (bsync) est positionné à True. Ceci signifie que le contrôle est renvoyé immédiatement depuis la commande shell.

Le type de retour n'est pas spécifié dans l'aide en ligne. Expérimentalement, j'ai déterminé que ce type est LONG. La valeur de retour a toujours été zéro lorsque j'ai pris la peine de vérifier. Si le programme n'existe pas, alors une erreur est générée et la macro s'arrête.

Shell (Pathname As String[, Windowstyle As Integer][, Param As String][, bSync])

Long

Pathname : Chemin complet et nom du programme à lancer.

Windowstyle : Spécifie le style de la fenêtre dans laquelle le programme sera lancé.

Param : N'importe quelle chaîne de caractère telle qu'elle puisse être passée en ligne de commande.

Bsync : Si False (défaut), un retour immédiat est exécuté. Si True, alors l'état du Shell ne sera retourné qu'après terminaison du programme.

```
Sub ExampleShell
    Dim vRC As Variant
    REM Une fenêtre de type 2 s'affichant en avant

    vRC = Shell("C:\andy\TSEProWin\g32.exe", 2, "c:\Macro.txt")
    Print "Je suis de retour, et le code de retour est " & vRC
    REM Ces deux-ci ont des espaces dans les noms
    Shell("file:///C:/Andy/My%20Documents/oo/tmp/h.bat",2)
    Shell("C:\Andy\My%20Documents\oo\tmp\h.bat",2)
End Sub
```

Antal Attila <atech@nolimits.ro> nous a transmis l'exemple suivant de l'utilisation de l'argument bsync.

```
Sub Main()
    REM Il faut d'abord créer sur votre disque un fichier avec le contenu suivant:
    REM sous Windows (nom de fichier C:\tmp\test.bat)
    REM     echo %1
    REM     pause
    REM sous Linux (nom de fichier /home/guest/Test.sh)
    REM     echo $1
    REM     sleep 100000

    REM ----- Exemple de Sync -----
    REM appel de ma macro d'exécution de script avec bSync=TRUE
    REM l'exécution du basic attendra que le terminal (ou la fenêtre msdos)
    REM soit fermé par l'appui d'une touche (CTRL+C sous Windows)
    REM Sous Windows
    shellRunner("file:///C:/tmp/", "Test", "Hello World", TRUE)
    REM ou sous Linux
    shellRunner("file:///home/guest/", "Test", "Hello World", TRUE)
    REM Signaler la fin de l'exécution
    Print "The End"

    REM ----- Sans Sync -----
    REM Appel avec bSync=FALSE
    REM L'exécution du code basic sera continuée
    REM Sous Windows
    shellRunner("file:///C:/tmp/", "Test", "Hello World", FALSE)
    REM ou sous Linux
    shellRunner("file:///home/guest/", "Test", "Hello World", FALSE)
    REM On indique la fin de l'exécution
    Print "The End"
End Sub

Sub shellRunner(dirPath$, script$, prms$, sync as Boolean)
    Dim filePath$, ef$, ed$, isWindows as Boolean
```

```

REM On regarde sous quel OS on se trouve
If instr(mid(dirPath,8),":")>0 or instr(dirPath,8,"\">0 Then
    isWindows=TRUE
Else
    isWindows=FALSE
End If

REM Conversion de l'URL en chemin de fichier
filePath = convertFromURL(dirPath)

REM Création de la ligne de commande
If isWindows Then
    ef = "command.com /C "+filePath+script+".bat"
Else
    ef = "xvt -e sh "+filePath+script+".sh"
End If

REM Exécution de la ligne de commande
Shell(ef, 1, prms, sync)
End Sub

```

Notation URL et Noms de fichiers

Il est conseillé de lire la description des fonctions ConvertToURL et ConvertFromURL.

Sous le système d'exploitation Windows, “[c:\autoexec.bat](#)” est un exemple de nom de fichier. On peut également définir celui-ci en notation URL comme “[file:///c|/autoexec.bat](#)”.

De manière générale quand on effectue une telle conversion, on débute l'URL avec “[file://](#)”, on change “:” en “|”, et on remplace “\” par “/”. Si on veut insérer le nom de l'ordinateur ou son adresse IP, on l'insère entre le deuxième et troisième Slash (/), comme ceci : “file://localhost/c|/autoexec.bat/”.

Les espaces et caractères spéciaux pouvant être inclus dans une notation URL doivent l'être avec une séquence d'échappement. Prenez la valeur ASCII du caractère, convertissez-la en hexadécimal, mettez le caractère % devant et placez-là où vous désirez voir apparaître le caractère. Par exemple, pour inclure un espace, “[c:\My Documents\info.sxw](#)” devient “file//c|/My%20Documents/info.sxw”.