



macromedia®
FLASH
MX

the most powerful solution for
developing rich Internet content
and applications

présenté par

Yazo *net*

Mars 2002

TABLE DES MATIERES

TABLE DES MATIÈRES.....	2
I. LE LOGICIEL.....	4
II. L'INTERFACE	5
LES PALETTES.....	5
LA PALETTE "PROPRIETES".....	5
LES CALQUES	6
LA BARRE D'OUTILS.....	6
CREATION DES SYMBOLES.....	7
SELECTEUR DE COULEUR (MIXEUR).....	8
NOUVELLE FENÊTRE ACTIONS SCRIPTS.....	9
NOUVELLE FENÊTRE REFERENCES	10
NOUVEAU DEBUGGER.....	10
PALETTE COMPONENTS	11
CRÉATION DE MODELES	11
ENREGISTREMENT AU FORMAT FLASH 5.....	11
III. LES ACTIONS SCRIPTS	12
GESTIONNAIRES D'ÉVÈNEMENTS PLACÉS SUR LA TIMELINE	13
LISTENERS.....	15
VARIABLES GLOBALES	16
MASQUES SCRIPTABLES	17
CONTRÔLE DE L'AFFICHAGE DU CURSEUR "DOIGT".....	17
NOUVELLES INSTRUCTIONS DE SON.....	18
IMPORTATION DES VIDEOS.....	20
SETINTERVAL	21
SWITCH ET CASE.....	21
CREATION DE TEXTES DYNAMIQUES.....	22
MISE EN FORME D'UN TEXTE CREE DYNAMIQUEMENT	23
CREATION DE FORMES GÉOMETRIQUES	24
RÉALISATION DE FORMULAIRES	26
CHARGEMENT D'UNE IMAGE JPEG	29
LES PROPRIÉTÉS DE LA SCÈNE	29
LES CAPABILITIES	29
CHARGEMENTS ET ENVOI DE DONNÉES	29
CONCLUSION.....	29

Flash MX en offre un maximum !

Yazo.net a décidé de passer en revue les nouveautés de Flash MX et de vous les présenter décortiquées, comme à notre habitude, dans ce tutorial de mise à jour !

Vous allez non seulement pouvoir découvrir les nouveautés de Flash MX mais également les apprendre en ce qui concerne la partie des Actions Scripts. En effet, de nombreux exemples vont vous permettre de découvrir de nouveaux scripts mais vous allez surtout pouvoir télécharger les fichiers au format .fla qui ont servi à réaliser ce tutorial, sur le site de Yazo.net.

Souvenez vous, le passage de Flash 4 à la version 5 avait été un réel changement ! Il y a quelques mois encore, la communauté des flashers imaginait difficilement quelles allaient être les nouveautés de cette dernière mouture. Quelques petites idées sur les forums, mais... à ce point ! "Oui, vous avez vraiment fait fort, messieurs les développeurs de chez Macromedia". Espérons simplement que cette version ne comptera pas trop de bugs. ;)

Nous allons vous présenter les nouvelles fonctionnalités de Flash MX en 3 parties :

- Le logiciel
- L'interface
- Les Actions Scripts

Avant-propos : Attention, nous n'avons pas au travers de cet article, la prétention de faire une énumération exhaustive des nouveautés. Seules quelques nouveautés techniques ne seront pas présentées, nous laissons à Macromedia le soin de le faire.

Tous les fichiers (.fla) de cet article sont disponibles à l'adresse suivante :
www.yazo.net/flashmx.html

I. LE LOGICIEL

Ça y est, vous avez la boîte en main (Image 1.1), vous êtes impatient(e) de l'ouvrir pour installer le logiciel sur votre machine. Avez-vous tout de même pris le temps de jeter un coup d'œil sur la face avant de la boîte ? La packaging surprend tout de même, car il marque un réel changement, il casse l'image que nous avons l'habitude de voir des produits Macromedia, cela alimentera sûrement certains topics de certains forums !

Test

Lorsque vous lancerez le logiciel pour la première fois, vous constaterez que le nouveau "SplashScreen" (Image 1.2) est à l'image du packaging.



Image 1.1 : Packaging

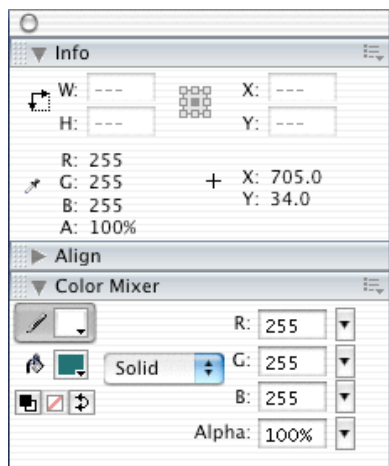


Image 1.2 : SplashScreen

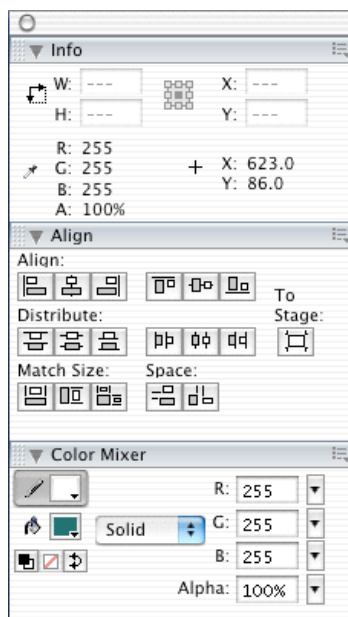
II. L'INTERFACE

LES PALETTES

Certains apprécieront, d'autres regretteront, les palettes n'existent plus. Sûrement plus pratique au quotidien, il faudra en tous les cas changer ses habitudes.



2.1 : Une palette rétractée (Align)



2.2 : La même, mais développée !

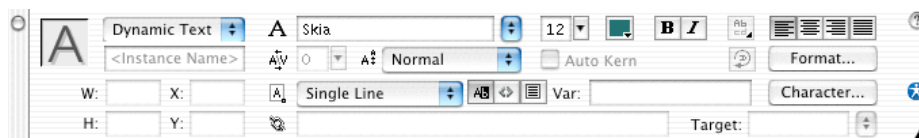
Un petit clic sur le triangle à gauche du nom d'une palette et "hop", elle se déploie ! Un autre clic, "hop", elle se rétracte ! Cela dit, rien ne vous empêche de les séparer et de les placer les unes à côté des autres ! Elles se manipulent à nouveau comme des palettes sans les onglets. Mais non, vous n'avez pas tout perdu !

LA PALETTE "PROPRIETES"

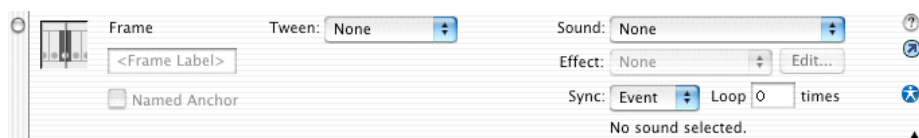
Cette nouvelle palette comparable à celle de Dreamweaver (qui porte d'ailleurs le même nom) est un regroupement de plusieurs palettes selon les cas. Elle permet en effet d'effectuer certains réglages en fonction de votre sélection sur la scène (une occurrence de bouton, de clip, un texte, un trait, etc.)



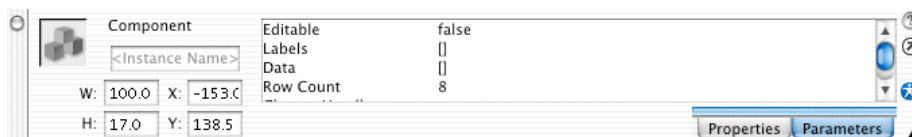
2.3 : Les anciennes palettes Occurrence - Effet (réunies en une seule)



2.4 : Les anciennes palettes Caractère - Paragraphe - Options de texte



2.5 : Les anciennes palettes Image - Son



2.6 : Les composants peuvent aussi se paramétrer par la palette Propriétés

Vous ne savez pas ce que sont les composants ? Nous non plus !
Mais non, rassurez-vous, une partie de cet article est consacrée à cette nouveauté.

LES CALQUES

Dossier : Vous aviez tellement de calques dans votre timeline que vous ne saviez plus quoi faire ! Fini de vous arracher les cheveux. Les dossiers (Image 2.7) vont vous permettre de ranger tous les calques qui doivent être regroupés selon votre travail.

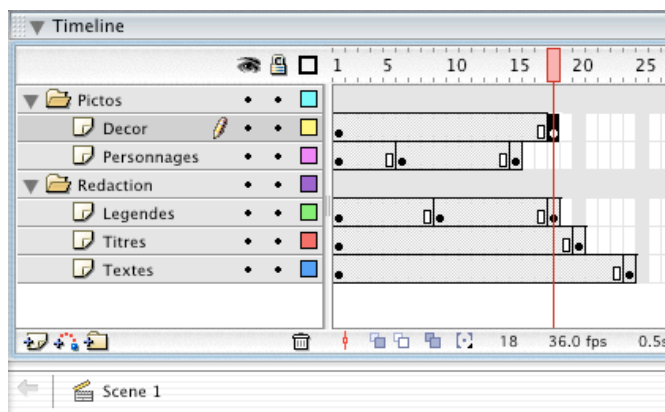
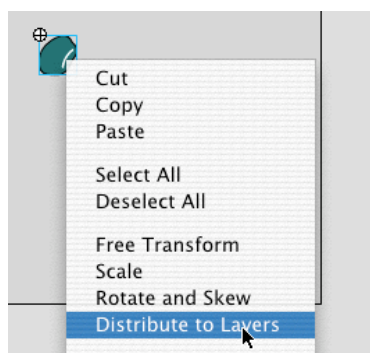
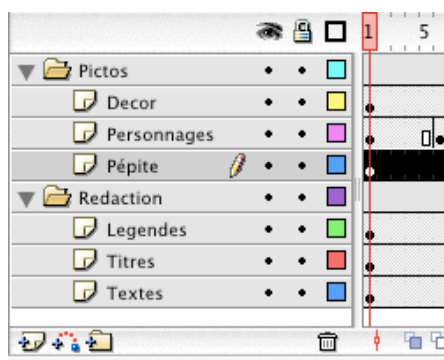


Image 2.7 : Rangez vos calques dans les dossiers.

Distribution : Cela ne vous suffit pas ? Vous avez bien raison d'en redemander. Si vous cliquez sur une occurrence de la scène (Image 2.8), vous pourrez la distribuer (couper-coller) sur un nouveau calque (Image 2.9).



2.8 : Cliquez sur une occurrence.



2.9 Elle vient se placer sur un nouveau calque au nom de l'occurrence.

LA BARRE D'OUTILS

Pas de gros changements, juste une petite refonte et l'arrivée de 2 nouveaux outils consacrés à la transformation des formes.

Quoi ? Annoncer l'arrivée de ces 2 nouveaux outils de cette façon-là !

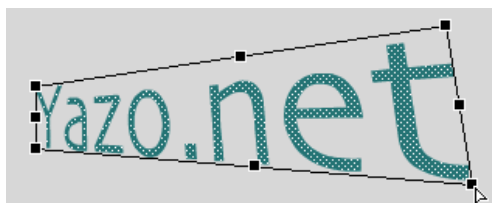
Pardon, reprenons. La barre d'outils n'a pas tellement changé, en revanche, 2 nouveaux outils devraient changer certaines habitudes de production des graphistes. Oui mais attention aux dérapages. Il y a "beau" et "beau".



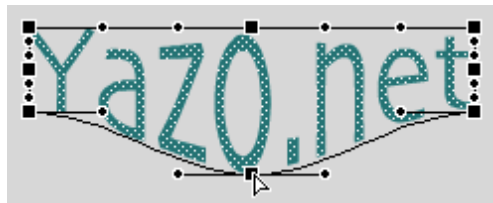
L'outil de transformation libre permet de redimensionner, tourner (faire pivoter) et incliner une sélection comme le faisaient les 2 anciens outils consacrés à cet effet. Lorsque vous sélectionnez cet outil, il vous donne accès à 2 autres qui vous permettent de tordre votre sélection selon 2 modes (en bas de la barre d'outils).



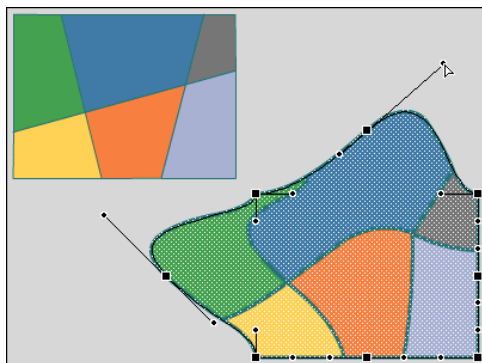
L'outil de Torsion (à gauche) et Enveloppe (à droite) permettent de réaliser toutes les transformations que vous faites habituellement avec le raccourci clavier Commande-T (Mac) Contrôle-T (PC) de Photoshop et les courbes de Bézier.



2.10 : Torsion



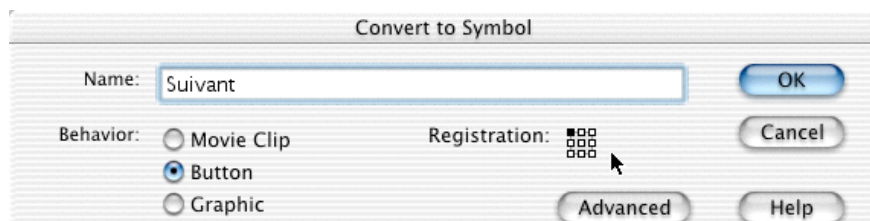
2.11 Enveloppe



2.12 : Enveloppe sur un tracé mais pas sur une image

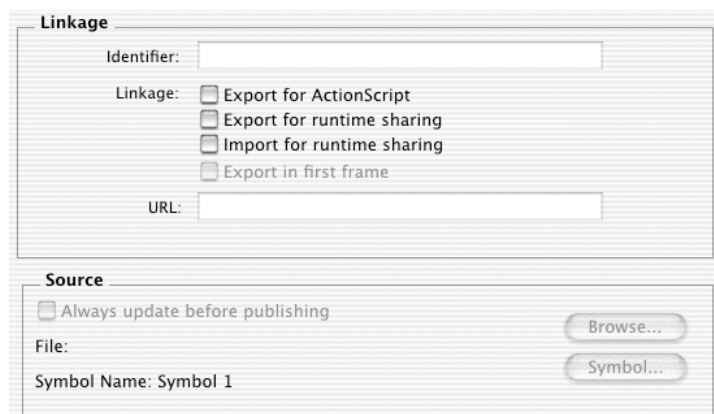
CREATION DES SYMBOLES

Ce point-là a été amélioré, notamment sur une notion très importante liée à l'alignement de l'occurrence sur la scène. Lorsque vous créez un symbole, son point de "registration", c'est à dire son point servant à la rotation, la mise à l'échelle et la position de l'occurrence avec les propriétés `_x` et `_y` (Actions Scripts) est par défaut au centre. Vous ne serez plus obligé de rentrer dans le symbole pour modifier ce dernier dès que vous aurez fini de le créer, car vous choisissez ce point à la création (image 2.13).



2.13 : Le choix du point de registration dès la création

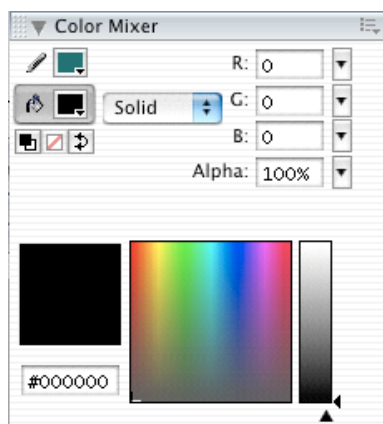
Une nouveauté plus importante, vous pouvez lier vos symboles entre bibliothèques/animations de façon plus simples qu'auparavant.



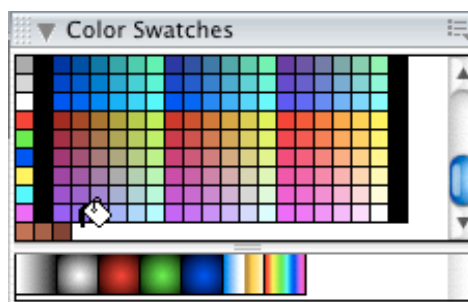
2.14 : Liens dynamiques entre animations

SELECTEUR DE COULEUR (MIXEUR)

Présentation : Vous n'aviez que très peu de place pour cliquer dans un nuancier de couleur afin de choisir la teinte souhaitée. Vous avez à présent non seulement une zone plus grande (Image 2.15) mais également un variateur de luminosité pour la couleur sélectionnée.



2.15 : Sélection plus précise



2.16 Ajout et suppression de couleurs simplifiés

Mémoriser une couleur : Lorsque votre couleur est définie, vous pourrez enfin mémoriser votre couleur sans bugs par un simple clic (Image 2.16) dans le nuancier. A noter tout de même qu'une couleur mémorisée est rattachée au document et non l'application.

Un simple clic avec la touche Commande (Mac) ou Contrôle (PC) enfoncée sur la couleur dans le nuancier permet de supprimer la couleur.

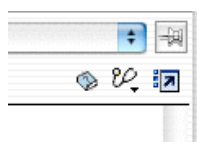
NOUVELLE FENÊTRE ACTIONS SCRIPTS

Plusieurs points sautent aux yeux. Le premier est relatif à la barre des boutons (Image 2.17 et 2.18) qui accompagne la fenêtre. Elle vous permet notamment de :



2.17 : Une nouvelle barre d'outils mais les raccourcis claviers s'imposent

- Introduire comme avant une instruction via le menu,
- Rechercher une chaîne de caractères (instruction, commande, propriété...),
- Rechercher Remplacer du code dans vos scripts,
- Insérer un chemin (toujours aussi pratique !!! ;),
- Vérifier le code de votre script,
- Indenter vos lignes d'instructions,
- Afficher les "Code Hint" (événements, propriétés et paramètres de votre code).



2.18 : Une barre d'outils visiblement très riche !

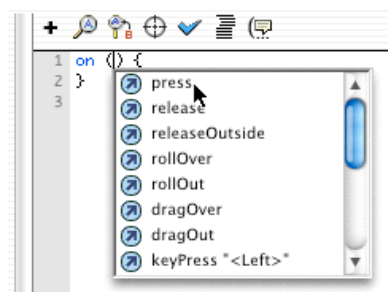
- Afficher la nouvelle fenêtre Références,
- Insérer, effacer un Break Point (Un point d'arrêt d'exécution de votre script),
- Régler le mode de programmation / Afficher la numérotation des lignes.

La première remarque à faire est sûrement liée au raccourci clavier qui permet de vérifier et indenter vos lignes d'instructions. Il existe et utilisez le !

Commande-Shift-F (Mac) et Contrôle-Shift-F (PC)

Ce raccourci indentera et vérifiera votre code mais vous pouvez aussi paramétrer la réécriture de votre code en réglant les préférences dans le menu contextuel de la palette des Actions Scripts.

La deuxième remarque concerne l'affichage automatique d'un menu contextuel (Image 2.19) dès lors que vous taperez le début du gestionnaire de vos scripts ou le raccourci clavier Esc-O-N ou Esc-O-C. Si cela vous gêne car vous avez d'autres habitudes, désactivez cette option dans les préférences du logiciel, le menu contextuel ne vous sera plus proposé et les arguments (paramètres) des commandes ne s'afficheront pas non plus lors d'une insertion automatique via le raccourci clavier Esc-x-x.

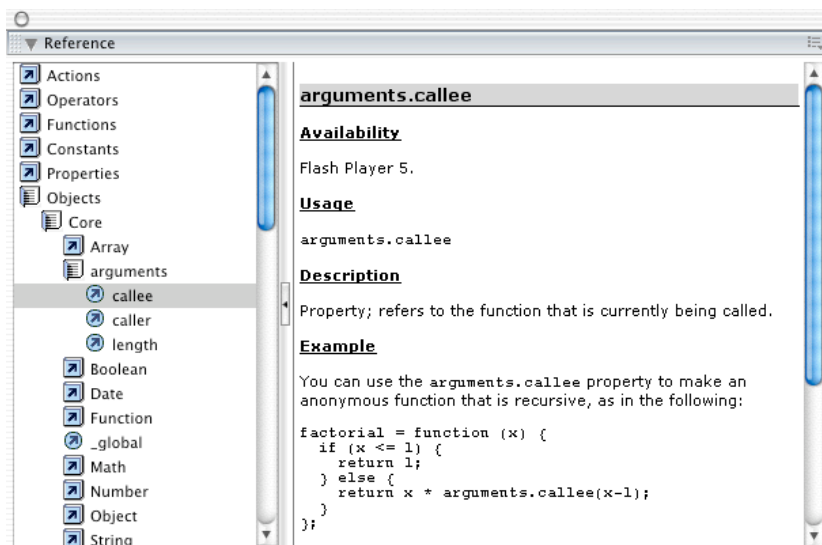


2.19 : Un menu contextuel pratique pour les novices et les experts qui sauront bien s'en servir

Enfin et pour finir, un menu déroulant (Image 2.18) vous propose de sélectionner les différents scripts placés dans votre animation, offrant ainsi un accès rapide aux scripts que vous cherchez.

NOUVELLE FENÊTRE REFERENCES

Cette nouvelle fenêtre (Image 2.20) peut s'avérer pratique dans certains cas, mais nous pourrions reprocher des explications "trop" techniques. L'emploi d'une terminologie très précise pour l'explication de certains points renforce la difficulté de compréhension. Oui, c'est bien beau de faire des manuels "très officiels", mais les définitions sont parfois "abstraites pour les néophytes ! Ils ne représentent pas une grosse partie de la cible visée pour cette fenêtre ? Il faut parfois savoir parler la langue de tout le monde ! Merci pour ce "dictionnaire", il offre de beaux jours à Yazo.net ! Mais ?! La pédagogie a donc un sens !

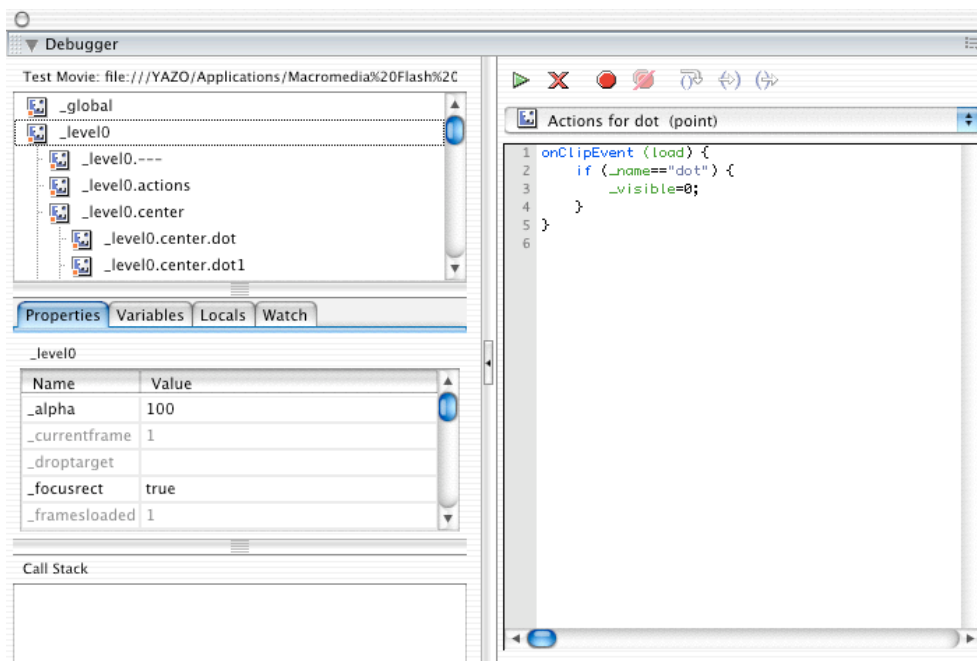


2.20 : Une palette très pratique à condition de bien connaître la terminologie de la programmation

Plus sérieusement, cette fenêtre s'avère très pratique pour les développeurs confirmés.

NOUVEAU DEBUGGER

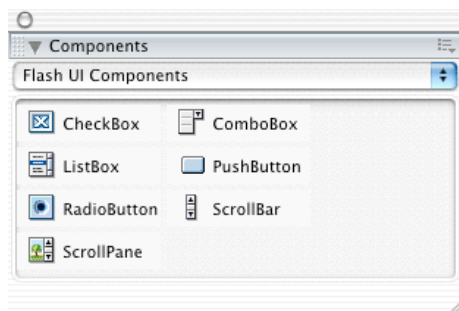
Vous souhaitez visualiser la valeur de vos variables, lire les propriétés de certaines de vos occurrences, ou bien encore exécuter pas à pas vos scripts, cette nouvelle fenêtre est celle qu'il vous faut (Image 2.21). Elle vous servira aussi dans le debuggage de vos scripts !



2.21 : La palette de debuggage améliorée

PALETTE COMPONENTS

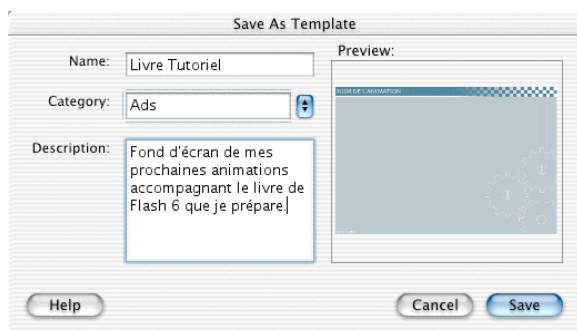
Des formulaires directement intégrés à Flash ! Ce n'est pas tout. Ajouter un scrollbar (Ascenseur) à vos textes sur la scène et à vos images (oui, vous avez bien lu, pour faire des masques automatiques à contenu mobile) devient possible grâce au Components (Image 2.22). Dans la partie 3 de cet article, des exemples vous sont proposés.



2.22 : Des formulaires dans Flash mais pas seulement !

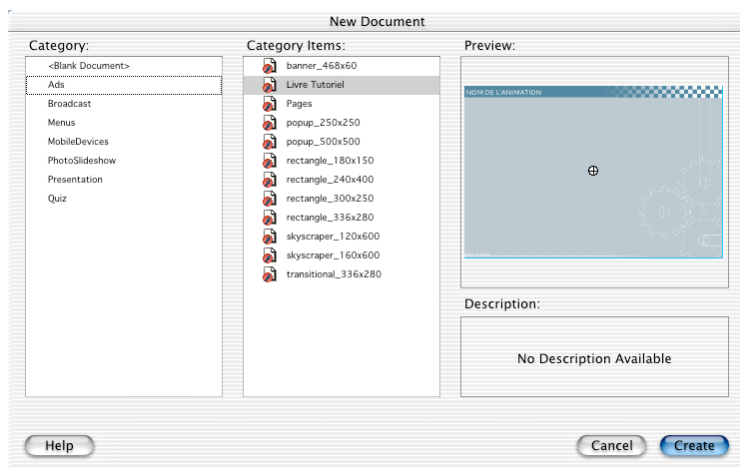
CRÉATION DE MODELES

Vous en avez marre de recommencer 10.000 fois (ça fait 1524,49 fois en euros) la même chose dans vos animations. Oui, une duplication de fichiers vous permettait jusqu'à présent d'éviter de recréer la partie commune de certaines de vos animations mais c'est fini. Tout ceci n'était qu'un mauvais rêve, dit-il au Petit Prince !



2.23 : Grande nouveauté, la création de modèles. Très simples à créer. Très pratiques.

Dans le menu Fichier, vous pourrez à présent Enregistrer votre document (image 2.23) au format Modèle (Template). Vous créerez alors des nouveaux documents (Image 2.24) à partir de modèles déjà existants ou que vous avez vous-même préparé !



2.24 : Un gain de temps incontestable grâce aux modèles. Un outil pour les chefs de projets !

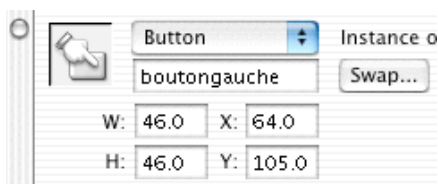
ENREGISTREMENT AU FORMAT FLASH 5

Mais non, vous ne serez pas coupé du monde les premiers mois de la sortie de Flash MX. Vous pourrez commencer à travailler dans Flash MX tout en continuant d'enregistrer au format Flash 5 afin que vous puissiez partager vos fla avec d'autres flasheurs à la traîne. Espérons que nous ne rencontrerons pas de bugs à ce niveau là.

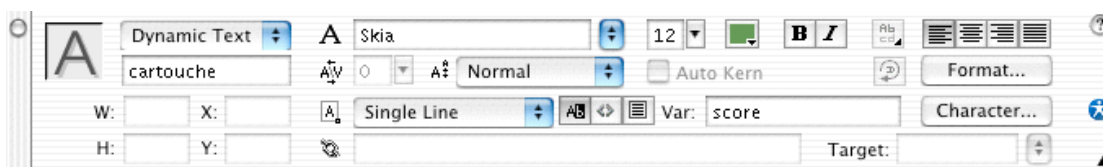
III. LES ACTIONS SCRIPTS

Enfin, nous y voilà ! Non, je ne parle pas des quelques pages qu'il a fallu lire avant d'en arriver à la partie sur les Actions Scripts, je veux bien sûr parler de cette nouvelle que de nombreux flasheurs attendaient !

Les occurrences des boutons, ainsi que les textes dynamiques et de saisie peuvent être nommés (Image 3.1 et 3.2) ! Oui, vous avez bien lu, vous allez pouvoir contrôler les occurrences de vos boutons et vos textes, presque comme les occurrences de type clip. Ne confondez pas le nom de vos variables et de vos textes, ce n'est pas la même chose (Image 3.2).



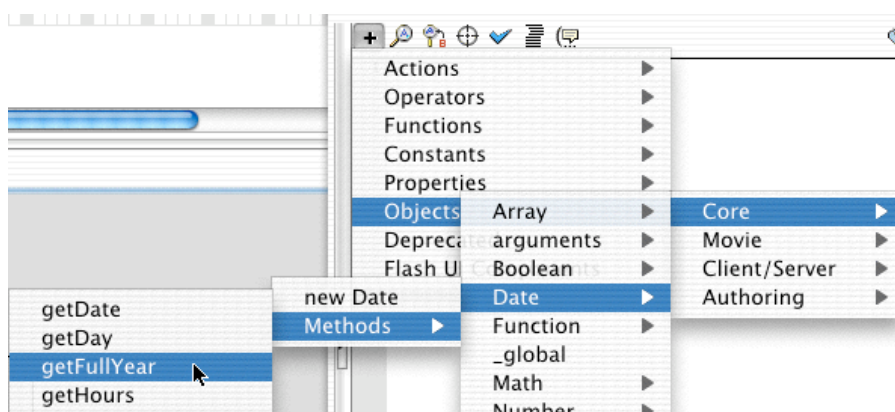
3.1 : Les occurrences des boutons peuvent à présent être nommées



3.2 : Les textes peuvent porter des noms d'instance (ex. : cartouche) et de variable (ex. : score)

Toujours côté bonnes nouvelles, avant de vous présenter en détail des lignes et des lignes d'instructions, il est à présent possible de prévisualiser vos clips et composants directement sur la scène.

Madame, Monsieur, j'ai le regret de vous annoncer la découverte d'un paradoxe. Vous allez devoir partir à la recherche de vos commandes dans le menu contextuel habituel situé dans le coin supérieur gauche de la fenêtre Actions Script. En effet, cela paraît d'une bonne attention, mais il faut craindre que la nouvelle réorganisation des commandes en différentes catégories et sous catégories soit un peu maladroite (Image 3.3). Il faut reconnaître que les catégories sont assez claires mais une fois de plus, toutes les commandes des Actions Scripts ne sont pas présentes dans ce menu. Bon, on comprend que cela n'a pas dû être facile à réorganiser et pour compatir, il faut reconnaître qu'il commence à y avoir un volume importants de commandes !



3.3 : Ils n'avait visiblement pas le choix. Mieux rangées mais plus difficiles à trouver...

GESTIONNAIRES D'ÉVÈNEMENTS PLACÉS SUR LA TIMELINE

Flash 4 ne permettait de placer du code que sur les images clés et les occurrences des boutons. Avec l'arrivée de la version 5, les gestionnaires d'événements liés aux clips ont fait leur apparition. Flash MX propose à présent de placer des gestionnaires d'événements sur l'image clé de la timeline principale. Ils sont rattachés aux occurrences (déjà en place sur la scène) et à la scène elle-même avec `_root`. Cela signifie tout simplement que la programmation peut être faite, si on le désire, directement à partir de la timeline sans passer par les occurrences.

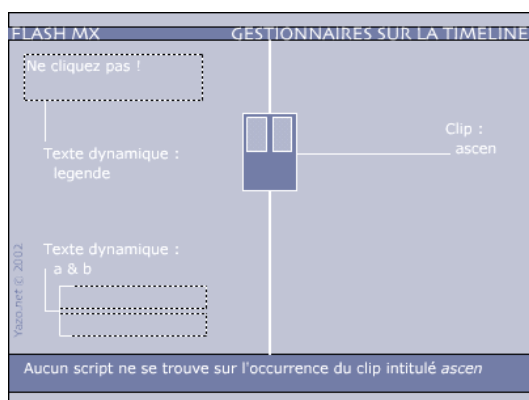
Vous avez placé un texte de saisie sur la scène qui s'intitule (son nom d'instance, pas son nom de variable) `monnom`. Vous souhaitez surveiller ce qui se passe dès que l'utilisateur tape du texte dedans. Voilà le script que vous pouvez placer sur la timeline.

```
monnom.onChanged = function () {
    commentaire = lenom.length;
}
```

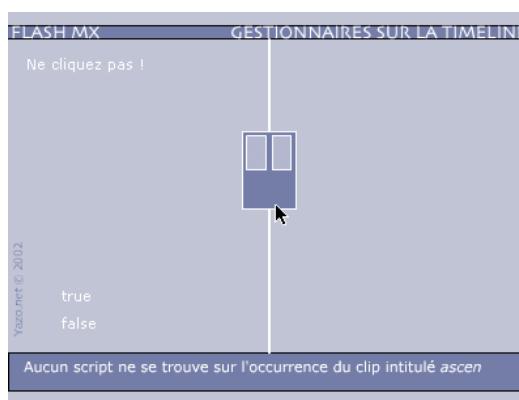
Vous pouvez aussi utiliser cette syntaxe :

```
function juge() {
    commentaire = lenom.length;
}
monnom.onChanged = juge;
```

Vous créez effectivement une fonction avant d'y faire référence.



3.4 : Les scripts sont sur l'image clé



3.5 : Fichier *gestionnaires fla*

Dans l'exemple ci-dessus, le fichier *gestionnaires fla*, un clip est placé sur la scène, son nom d'occurrence est *ascen*. On place également un texte dynamique intitulé *legende*.

Voici le script qui se trouve sur l'image clé 1 de la timeline principale.

```
legende._visible=0

ascen.onRollOver = function () {
    legende._visible=1
}

ascen.onMouseDown = function () {
    legende.text="Pourquoi avez-vous \rcliqué ?"
}

ascen.onRollOut = function () {
    legende._visible=0
    legende.text="Ne cliquez pas !"
}

_root.onEnterFrame = function() {
    a = _root._ymouse<150;
    b = _root._ymouse>150;
    switch (a) {
        case true :
            sens = -1;
            break
    }
    switch (b) {
        case true :
            sens = 1;
            break
    }
    if (ascen._y>60 && ascen._y<225) {
```

```

        ascen._y += sens;
    }
    if (ascen._y <= 60) ascen._y = 61;
    if (ascen._y >= 225) ascen._y = 224;
};

```

On notera plusieurs points.

- Le contenu d'un texte dynamique est modifié via les Actions Scripts (abordé dans ce tutorial).
- Un texte dynamique (ou de saisie) possède des propriétés (abordé dans ce tutorial).
- Une nouvelle technique de test avec switch () et case (abordé dans ce tutorial).

Voici à présent, quelques exemples supplémentaires afin de mieux comprendre ces gestionnaires d'évènements.

Exemple 1 : Un bouton est placé sur la scène. On nomme son occurrence *boutonsuite*. Le script ci-dessous est placé sur l'occurrence.

```

on (press) {
    calcul = prix/6.55957;
}

```

On retire à présent le code qui se trouve sur l'occurrence et on place le script ci-dessous, sur l'image clé qui contient cette occurrence.

```

boutonsuite.onPress = function() {
    calcul = prix/6.55957;
}

```

Le résultat est strictement identique. Dans un cas, le code se trouve sur l'occurrence du bouton, dans l'autre cas le code se trouve sur l'image clé de la timeline.

Exemple 2 : Un clip est placé sur la scène. On nomme son occurrence *personnage*. Le script ci-dessous est placé sur l'occurrence.

```

onClipEvent (enterFrame) {
    _x += 5;
    if (_x > 300) {
        x = -10;
    }
}

```

On retire à présent le code qui se trouve sur l'occurrence et on place le script ci-dessous, sur l'image clé qui contient cette occurrence.

```

personnage.onEnterFrame = function() {
    this._x += 5;
    if (this._x > 300) {
        this._x = -10;
    }
}

```

Là encore, le résultat est strictement identique. Dans un cas, le code se trouve sur l'occurrence du clip, dans l'autre cas le code se trouve sur l'image clé de la timeline. Il suffit donc tout simplement de saisir le nom d'une instance, lui associer, séparé d'un point, un événement rattaché à l'occurrence puis lui assigner (placer le signe =) comme valeur une fonction.

Exemple 3 : La scène vérifie si la souris bouge.

```

_root.onMouseMove = function() {
    tempo--;
}

```

Exemple 4 : Si l'utilisateur modifie une zone de texte (texte de saisie), ce gestionnaire s'exécute.

```

lenom.onChanged = function() {
    etatchgt._visible=1;
}

```

La technique ci-dessous permet d'éviter de placer la fonction derrière le gestionnaire d'événement comme dans l'exemple ci-dessus, mais le résultat est le même.

```

changer = function () {
    etatchgt._visible = 1;
};
lenom.onChanged = changer;

```

Voici quelques événements valables pour les occurrences de type clip, bouton et texte. Vous devez placer ces instructions dans la fenêtre Actions Scripts sur une image clé (on ne le répétera jamais assez) : onChanged, onData, onDragOver, onEnterFrame, onKeyDown, onKillFocus, onLoad, onMouseMove, onPress (bouton), onMouseDown (clip), onRollOver (clip et bouton)...

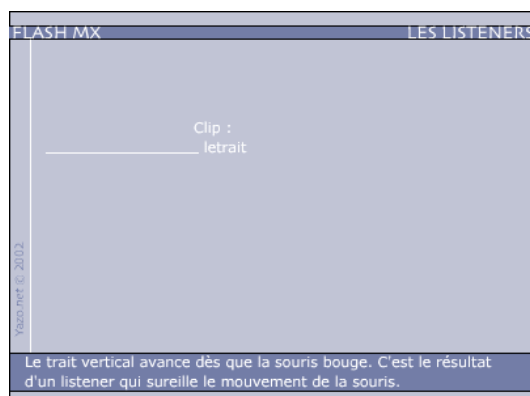
Dans la mesure où des occurrences peuvent être créées dynamiquement, il est important de connaître cette technique des gestionnaires d'évènements saisis sur une image clé.

LISTENERS

La technique des listeners est assez proche de celle des gestionnaires placés sur les images clés.

La scène, le clavier, la souris, un texte et une sélection sont autant d'éléments auxquels on peut rattacher un "mouchard" qui surveille le déclenchement de certaines instructions selon un événement.

Si vous avez une occurrence sur la scène, vous pouvez lui ajouter un gestionnaire comme nous venons de le voir. Vous pouvez aussi créer un objet auquel vous rattachez un listener. Ces listeners sont par exemple intéressants et utiles pour contrôler le redimensionnement de la scène.



3.6 : - Fichier *listeners fla*

Sur l'image clé 1 de la timeline principale, le script ci-dessus a été saisi. Il contient 2 listeners.

```
james = new Object();
james.onMouseMove = function() {
    letrait._x++;
};
Mouse.addListener(james);
lespion = new Object();
lespion.onResize = function() {
    letrait._x = 0;
};
Stage.addListener(lespion);
```

Ce script surveille deux choses :

- Le mouvement de la souris
- Le redimensionnement de la scène

Intéressons nous à la partie qui est chargée de surveiller le redimensionnement de la scène.

La première ligne d'instruction créait un objet (une instance). Il s'intitule *lespion* dans notre exemple.

```
lespion = new Object();
```

La deuxième ligne d'instruction comprend :

```
Nomdelobjetcréé.événementàsurveiller = function() {
    lespion.onResize = function() {
```

Le nom de l'objet que vous avez créé doit être écrit en premier (*lespion*). Vous tapez ensuite un point pour séparer le nom de votre objet de l'événement à préciser. Selon l'objet (Scène, souris, clavier), les événements diffèrent. Vous tapez ensuite le signe égal puis le mot fonction accompagné de 2 parenthèses et d'une accolade.

Sur la(es) ligne(s) du dessous, tapez vo(s)tre instruction(s). Terminez sur la dernière ligne par une accolade qui se ferme.

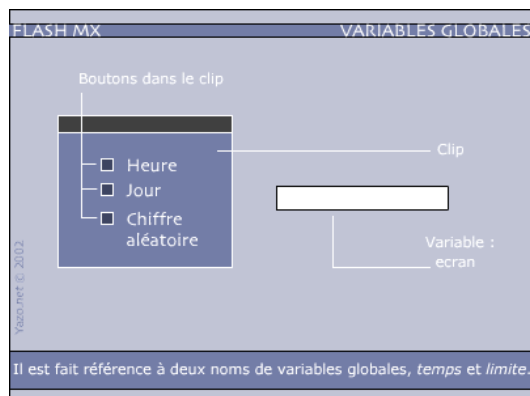
```
    letrait._x = 0;
};
```

Enfin, une dernière ligne d'instructions permet d'associer cet objet que vous venez de créer à un listener.

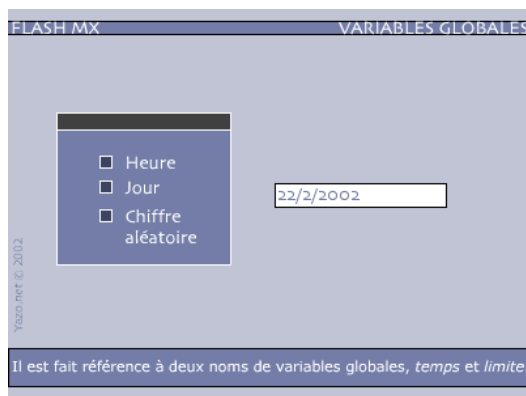
```
Stage.addListener(lespion);
```

VARIABLES GLOBALES

La portée des variables dans Flash 5 se limitait à la timeline dans laquelle elles se trouvaient. On utilisait alors `_root` et des chemins absolus pour faire référence à une variable. La propriété `_global` permet à présent de créer une variable globale, accessible depuis n'importe quel niveau.



3.7 : Deux variables globales créées sur la timeline



3.8 : Fichier *globales fla*

Dans l'exemple ci-dessus, le fichier *globales fla*, la scène contient :

- Un texte dynamique intitulé *ecran*
- Un clip avec 3 boutons à l'intérieur.

Voici le code qui se trouve sur l'image clé 1 de la timeline principale :

```
_global.temps=new Date()
_global.limite=49
```

Voici à présent le code qui se trouve sur les 3 boutons .

(Si vous ne connaissez pas la programmation orientée objets, ne lisez que le 3e script)

```
on (press) {
    _root.ecran = temps.getHours()+":"+temps.getMinutes();
}

on (press) {
    _root.ecran = temps.getDate()+"/"+(temps.getMonth()+1)+"/"+temps.getFullYear();
}

on (press) {
    _root.ecran = random(limite);
}
```

Rappelons que sans les variables globales, il faudrait utiliser la propriété `_root` pour faire référence aux variables *limite* et *temps*.

Voici à présent, quelques exemples supplémentaires afin de mieux comprendre la notion de variables globales.

Exemple 1 : Sur la scène, la ligne d'instruction ci-dessous est saisie.

```
_global.prix=130
```

Sur la scène, une variable (un texte dynamique) est créée puis nommée *ticket*. Un clip est également placé sur la scène, ce dernier contient un bouton avec le code ci-dessous :

```
on (press) {
    _root.ticket = prix*1.196;
}
```

On est obligé de faire référence à `_root` car la variable *ticket* se trouve sur la scène, en revanche il n'est pas nécessaire d'utiliser la propriété `_root` pour faire référence à *prix* car il s'agit d'une variable globale.

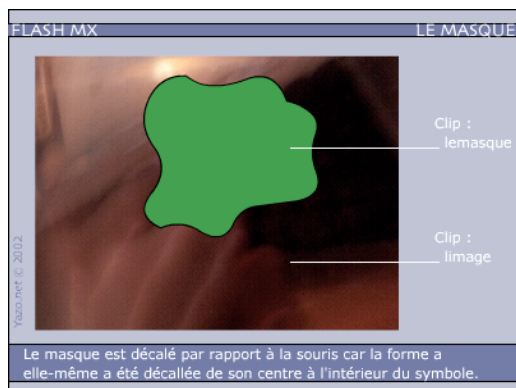
Pour observer la valeur d'une variable globale, vous pouvez utiliser le débogueur.

MASQUES SCRIPTABLES

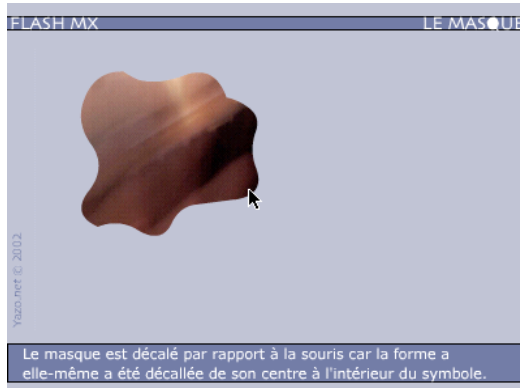
Vous pouvez toujours utiliser les masques comme vous le faisiez avec Flash 5 et les versions antérieures, à savoir, sur 2 calques. Nous ne pouvions pas jusqu'à présent utiliser les Actions Scripts pour contrôler un masque mais ça c'est le bon vieux temps ! Et oui, regardez les lignes d'instructions ci-dessous.

```
limage.setMask("lemasque")
```

Cette ligne d'instruction permet simplement d'attribuer un masque à un clip. Il n'est plus indispensable de placer chaque partie du masque (l'image masquée et le masque) sur deux calques séparés. Précisons tout de même que cette technique est toujours utilisable. Les images ci-dessous (Image 3.9 et 3.10) vous montrent un exemple de masque mobile.



3.9 : Préparation du masque mobile



3.10 : Fichier *masque fla*

Le code ci-dessous correspond au masque de l'exemple ci-dessus. On constate que le masque peut non seulement être défini par les Actions Scripts mais il peut être mobile.

```
limage.setMask("lemasque")
lemasque.startDrag(1);
```

CONTRÔLE DE L'AFFICHAGE DU CURSEUR "DOIGT"

Lorsque le curseur de la souris passe sur un bouton, il se transforme automatiquement en curseur en forme de doigt. Le script ci-dessous permet de ne pas afficher le curseur du doigt au survol d'un bouton.

```
btavant.useHandCursor=0
```

Ce script est placé sur l'image clé 1 de la timeline principale, le nom de l'occurrence du bouton visé est *btavant*. La documentation de Flash MX stipule que cette propriété est aussi valable pour l'instance d'un clip mais avec la version bêta que nous avons, nous n'avons pas été capable d'afficher un curseur de type doigt sur l'occurrence d'un clip (Fonctionne sur PC correctement).

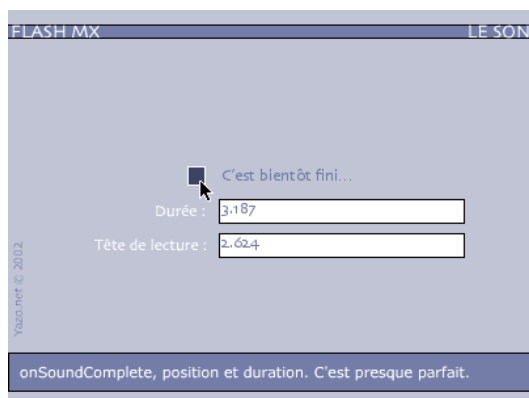
Attention : Si vous faites des essais, lancer vos swf dans le player ou le navigateur pour vérifier l'affichage du curseur. En faisant un simple Commande-Entrée (Mac) Contrôle-Entrée (PC), le doigt ne s'affiche pas correctement. Peut-être est-ce dû à la version que nous avons ou tout simplement normal.

NOUVELLES INSTRUCTIONS DE SON

Présentation : Il y a un réel changement de ce côté-là ! (Là, vous imaginez un roulement de tambours). Tous les flashers attendaient l'arrivée de ces propriétés, méthodes et événements de contrôle.

- duration
- position
- getBytesLoaded()
- getBytesTotal()
- onSoundComplete
- onLoad

Oui, vous avez bien lu, on peut à présent connaître la durée d'un son et la position de la tête de lecture au cours de l'écoute. Mais on peut surtout exécuter une ou plusieurs instructions à la fin de la lecture du son.



3.11 : Deux scripts pour contrôler la lecture du son - Fichier [son fla](#)

Durée d'un son / Fin de lecture d'un son : Voici le code qui se trouve sur le bouton (carré bleu foncé). Le déclenchement de la lecture du son se fait par le bouton. La surveillance de la fin du son s'exécute dans le même script.

```
on (press) {
    monson = new Sound();
    monson.attachSound("reveil");
    duree = monson.duration/1000;
    monson.onSoundComplete = sontermine
    monson.start();
    cle = 1;
}
```

Analysons le script ensemble :

```
monson = new Sound();
monson.attachSound("reveil");
monson.start();
```

Création de l'instance de son, attribution d'un son de la bibliothèque et lancement du son.

```
duree = monson.duration/1000;
```

Une variable intitulée *duree* se trouve sur la scène. On place dedans, une valeur qui correspond à la durée du son. Une division par mille est effectuée car la durée d'un son est spécifié en millièmes de secondes.

```
monson.onSoundComplete = sontermine
```

On affecte une fonction à l'événement onSoundComplete afin que son exécution soit effective à la fin du son.

```
cle = 1;
```

Comme il n'existe ni méthode, ni propriété qui permette de connaître l'état d'un son (à l'arrêt, en lecture), l'initialisation d'une variable est la seule solution pour connaître l'état du son. Dans la fonction qui s'exécutera à la fin du son, on initialise la variable à 0. Pendant ce temps à VeraCruz...

La variable est à 1 et l'image clé 1 de la timeline principale contient ce script :

```
sontermine = function () {
    cle = 0;
    lebouton._visible = 0;
    legende = "Ce bruit s'est arrêté";
}
```

```

};
_root.onEnterFrame = function() {
    if (cle == 1) {
        tete = monson.position/1000;
        if (tete>=1 && tete<1.2) {
            legende = "Attention";
        }
        if (tete>=2 && tete<2.1) {
            legende = "C'est bientôt fini...";
        }
    }
};

```

La définition de la fonction se trouve sur l'image clé, mais il est aussi possible de la placer directement dans le script précédent.

```

monson.onSoundComplete = function () {
    cle = 0;
    lebouton._visible = 0;
    legende = "Ce bruit s'est arrêté";
};

```

Dans ce cas, il n'est pas nécessaire de la définir. Voyons à présent la 2e partie qui présente une structure intéressante.

```

_root.onEnterFrame = function() {

```

Cette ligne d'instruction signifie que l'exécution de la fonction se fait de façon continue (en boucle). Elle contient notamment le code ci-dessous.

```

    if (cle == 1) {
        tete = monson.position/1000;
        if (tete>=1 && tete<1.2) {
            legende = "Attention";
        }
        if (tete>=2 && tete<2.1) {
            legende = "C'est bientôt fini...";
        }
    }
}

```

Si la variable *cle* vaut 1, c'est que la lecture est en cours, on exécute alors :

- L'affichage (dans la variable intitulé *tete* qui se trouve sur la scène) de la position de la tête de lecture dans le son.
- Deux tests pour synchroniser l'affichage de textes dans une autre variable située sur la scène.

Chargement d'un fichier son au format mp3 : Lorsque vous avez créé une instance de votre classe `sound()`, vous pouvez alors charger un son au format MP3.

```

voix = new sound();
voix.loadSound("rire.mp3", false);

```

Vous spécifiez le fichier qui se trouve en local ou sur une autre machine (ex. : <http://www.unsite/dossier/leson.mp3>) puis vous indiquez également le type de lecture. En streaming si vous réglez le paramètre sur `true`, le son se joue alors dès la fin du chargement ou bien sans streaming (si vous réglez le paramètre sur `false`). Il faudra alors demander une lecture du son via la commande `start()`.

Contrôler le niveau de chargement : Si vous effectuez le chargement d'un son qui ne se trouve pas sur la même machine que le swf, l'événement `onLoad` et la méthode `getBytesLoaded` vous permettent de connaître le niveau de chargement du son.

Vous créez une fonction :

```

encours = function () {
    _root.chargement = (voix.getBytesLoaded()/voix.getBytesTotal())*100;
    if (_root.chargement==100) bouton._visible=1
};

```

Dans l'exemple, *chargement* est le nom d'une variable qui se trouve sur la scène. Le résultat est le pourcentage de chargement qui s'affiche dans la variable.

Lorsque vous lancez l'exécution du chargement d'un son, vous utilisez alors l'événement `onLoad`.

```

voix = new sound();
voix.loadSound("rire.mp3", false);
voix.onLoad = encours

```

IMPORTATION DES VIDEOS

Lorsque vous importez une vidéo dans un clip afin que toutes les images de la séquence soient déroulées, vous manipulez alors ce dernier comme vous le faisiez jusqu'à présent ! Une simple navigation d'images en images. `play()`, `stop()`, `gotoAndPlay()`, `gotoAndStop()`, `nextFrame()` et `prevFrame()`.

Exemples :

```
on (press) {  
    lvideo.play();  
}
```

```
on (press) {  
    lvideo.stop();  
}
```

N.B. : *lvideo* est le nom de l'occurrence.

Pour le contrôle du son de la vidéo, ne cherchez pas de méthodes (dans les deux sens du terme), vous devez créer une instance de la classe `Sound()` et ciblé le clip qui accueille la vidéo.

```
sonvideo = new Sound("lvideo")
```

Vous réglez ensuite le niveau sonore avec la commande `setVolume()`.

```
on (press) {  
    sonvideo.setVolume(10);  
}
```

La technique que nous vous proposons est celle que yazo.net a trouvé, il en existe peut-être une autre plus pratique que nous ne connaissons pas, dans ce cas n'hésitez pas à nous envoyer un mail pour nous expliquer la votre. Nous n'avons trouvé aucune information concrète à ce sujet.

Les formats de vidéo reconnus dans Flash MX sont : mov, mpeg, video for Windows, MFV (Macromedia Flash Video en collaboration avec Sorenson).

N.B. : Un objet vidéo existe, mais la seule méthode disponible est `clear` dans la bêta version que nous avons. Il va de soi que d'autres méthodes vont être ajoutées. Pour l'instant, pas d'autres détails.

SETINTERVAL

Fichier *setInterval fla*

Cette action est très simple à mettre en application et à comprendre. Pour temporiser l'exécution de certaines instructions, cette commande va s'imposer comme LA commande.

Je souhaite agrandir la taille d'une occurrence toutes les 3 secondes ! "No problemo" comme dit Yazo, voici la solution. Ce script peut être placé n'importe où :

```
function taille() {
    if (barre._yscale<100) {
        barre._yscale += 10;
    } else {
        clearInterval;
    }
}
setInterval(taille, 3000);
```

Il s'agit d'un script qui a pour effet d'agrandir l'occurrence intitulée *barre* toutes les 3 secondes à condition que la taille de cette occurrence soit inférieure à 100 %. Lorsque la taille (en pourcentages) de cette occurrence arrive à 100%, la commande *setInterval* est annulée grâce à la commande *clearInterval*.

Remarque. Comme vous ne l'avez sûrement pas constaté ;) il faut préciser le nom d'une fonction parmi les argument de la commande *setInterval* mais ne pas placer les parenthèses derrière le nom

Remarque 2. Sarah Conore ? Non, c'est à côté !

SWITCH ET CASE

Fichier *switch fla*

Reprenons l'exemple que nous avons vu avec les questionnaires car ce fichier comporte un script avec ces 2 nouvelles commandes. Nous avons tout de même simplifié le code afin que vous compreniez quelles sont les instructions nécessaires au bon fonctionnement de cette commande et ce mot clé.

```
a = _root._ymouse<150;
b = _root._ymouse>150;
switch (a) {
    case true :
        sens = -1;
        break
}
switch (b) {
    case true :
        sens = 1;
        break
}
```

Les deux premières lignes d'instructions sont deux variables intitulées *a* et *b* dans lesquelles on stocke une valeur *true* ou *false*. Peu de développeurs ont l'habitude d'utiliser cette technique mais elle s'avère pratique pour effectuer certains tests. C'est encore plus intéressant avec *switch()*. En effet, *_root._ymouse<150* affectera *false* à la variable *a* si la position de la souris est après le 150e pixel du bord gauche de la scène.

Afin de mieux comprendre le fonctionnement de cette commande, voici un autre exemple qui vous permettra de voir le point commun entre les 2 exemples, vous permettant ainsi de faire votre propre analyse.

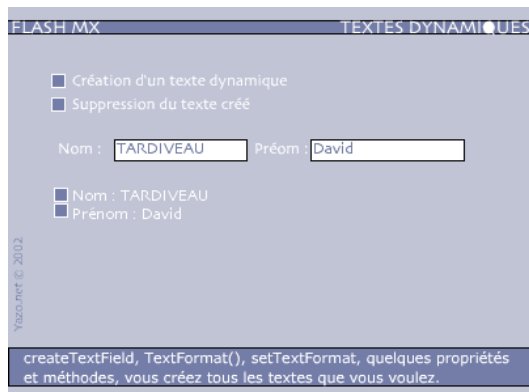
Exemple 1 : Une variable se trouve sur la scène. Il s'agit d'un texte de saisie intitulé *motdepasse*. Le code ci-dessous se trouve sur un bouton situé sur la scène.

```
on (press) {
    switch (motdepasse) {
        case "hvq78" :
            bouton1._alpha=1;
            bouton2._alpha=0;
            break
        case "xlv35" :
            bouton1._alpha=0;
            bouton2._alpha=1;
            break
    }
}
```

Mot de passe est la variable à tester. *hvq78* et *xlv35* sont les 2 valeurs proposées pour que les instructions puissent s'exécuter.

CREATION DE TEXTES DYNAMIQUES

La possibilité de créer des textes dynamiquement, à la volée, est l'une des grandes nouveautés de Flash MX qui va trouver de nombreuses applications. La méthode est très simple, mais nous pouvons d'ores et déjà regretter qu'il faille faire appel à 2 types d'instructions pour gérer ce type de créations dans son ensemble. En effet, dans un premier temps on créait la zone de texte avec une première technique, puis on la formate avec une autre technique.



3.12 : Les textes sont maintenant créés dynamiquement - Fichier *textesdynamiques fla*

Création : Voici le script minimum à placer sur une image clé ou une occurrence pour générer un texte.

```
on (press) {
    _root.createTextField("untexte", 1, 50, 50, 160, 30);
    untexte.text = "Http://www.yazo.net";
}
```

La commande `createTextField` permet de créer un texte dont le nom d'instance est précisé comme premier argument entre les parenthèses. Il faut ensuite préciser un niveau (1 dans notre exemple) qui devra être unique. Toute création d'un autre texte portant le même numéro supprimera automatiquement le premier créé (avec le même numéro). Les 4 valeurs qui suivent définissent la taille de la zone de texte créée (x, y, largeur, hauteur). Il faut ensuite placer un texte dans cette nouvelle instance.

Propriétés : Lorsque votre texte est créé, vous pouvez ensuite utiliser ces quelques propriétés.

<code>TextField.backgroundColor</code>	<code>TextField.multiline</code>	<code>TextField.textHeight</code>
<code>TextField.border</code>	<code>TextField._name</code>	<code>TextField.textWidth</code>
<code>TextField.borderColor</code>	<code>TextField._parent</code>	<code>TextField.type</code>
<code>TextField.bottomScroll</code>	<code>TextField.password</code>	<code>TextField._url</code>
<code>TextField.embedFonts</code>	<code>TextField._quality</code>	<code>TextField.variable</code>
<code>TextField._highquality</code>	<code>TextField.restrict</code>	<code>TextField._visible</code>
<code>TextField._height</code>	<code>TextField._rotation</code>	<code>TextField._width</code>
<code>TextField.hscroll</code>	<code>TextField.scroll</code>	<code>TextField.wordWrap</code>
<code>TextField.html</code>	<code>TextField.selectable</code>	<code>TextField._x</code>
<code>TextField.htmlText</code>	<code>TextField._soundbuftime</code>	<code>TextField._xmouse</code>
<code>TextField.length</code>	<code>TextField.tabEnabled</code>	<code>TextField._xscale</code>
<code>TextField.maxChars</code>	<code>TextField.tabIndex</code>	<code>TextField._y</code>
<code>TextField.maxhscroll</code>	<code>TextField.text</code>	<code>TextField._ymouse</code>
<code>TextField.maxscroll</code>	<code>TextField.textColor</code>	<code>TextField._yscale</code>

Vous pouvez aussi utiliser ces quelques méthodes.

- `addListener` et `removeListener`
- `getDepth`
- `removeTextField`
- `replaceSel`
- `setTextFormat`

Cette dernière méthode, `setTextFormat`, applique les mises en forme réglées via l'objet `TextFormat` que vous devez créer. Nous abordons cette partie après les exemples ci-dessous.

Exemple 1 : Créer un texte dynamique qui a pour contenu la valeur de 2 variables qui sont des textes de saisie

```
on (press) {
    paragraphe.text = "Nom : "+nom+"\r"+"Prénom : "+prenom;
}
```

Exemple 2 : Créer une variable intitulée *combien*

```
_root.createTextField("untexte", 1, 50, 50, 160, 30);
untexte.variable = "combien";
combien=15
```

Exemple 3 : Si l'utilisateur appuie sur la touche "Flèche droite du clavier", la sélection en cours dans un texte de saisie est remplacée par le mot "Annulé"

```
on (keyPress "<Right>") {
    untexte.replaceSel("Annulé");
}
```

MISE EN FORME D'UN TEXTE CREE DYNAMIQUEMENT

Fichier *textesdynamiques fla*

Nous venons de voir qu'il est possible d'effectuer certains réglages sur un texte créé dynamiquement mais il s'agit principalement d'une mise en forme du champ du texte. Seuls quelques réglages liés au texte sont disponibles. Pour une mise en forme du contenu, du texte, il faut créer une instance de l'objet `setTextFormat`.

Création : Le script ci-dessous est le minimum qu'on puisse taper :

```
on (press) {
    formatperso1 = new TextFormat();
    formatperso1.bold = true;
    untexte.setTextFormat(formatperso1);
}
```

Propriétés : Il faut effectivement donner un nom d'instance et ensuite utiliser `new TextFormat()`, puis utiliser les propriétés présentées ci-dessous. Lorsque votre réglage est fait, vous devez l'appliquer à votre texte. On cible alors le nom d'instance d'un texte dynamique ou de saisie puis on utilise la méthode `setTextFormat` de l'objet `TextField` pour appliquer la mise en forme.

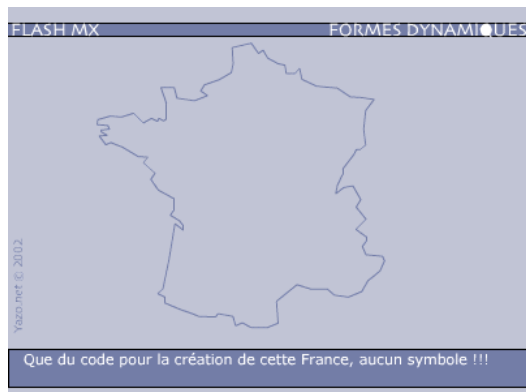
<code>TextFormat.align</code>	<code>TextFormat.leading</code>
<code>TextFormat.blockIndent</code>	<code>TextFormat.leftMargin</code>
<code>TextFormat.bold</code>	<code>TextFormat.rightMargin</code>
<code>TextFormat.bullet</code>	<code>TextFormat.tabStops</code>
<code>TextFormat.color</code>	<code>TextFormat.target</code>
<code>TextFormat.font</code>	<code>TextFormat.size</code>
<code>TextFormat.indent</code>	<code>TextFormat.underline</code>
<code>TextFormat.italic</code>	<code>TextFormat.url</code>

L'exemple ci-dessous vous montre à quel point il est possible de configurer précisément un texte.

```
on (press) {
    untexte.text = "Texte qui change\r dans cette zone";
    untexte.multiline = true;
    untexte.wordWrap = true;
    formatperso1 = new TextFormat();
    formatperso1.color = 0xff0000;
    formatperso1.underline = true;
    formatperso1.font = "skia";
    formatperso1.size = 16;
    formatperso1.bold = true;
    paragraphe.setTextFormat(formatperso1);
}
```

CREATION DE FORMES GÉOMÉTRIQUES

Cette nouveauté va aussi trouver de nombreuses applications. Enfin, lorsque nous disons nombreuses, nous voulons dire nombreuses. Une petite preuve ? Et bien la copie d'écran que vous voyez ci-dessous est celle du fichier que vous pouvez télécharger. Si vous ouvrez le .fla, vous constaterez que la bibliothèque est vide. Mais d'où sort cette carte de France alors ? Elle est créée dynamiquement.



3.13 : Une forme créée dynamiquement à partir de segments - Fichier [formesdynamiques3.fla](#)

Présentation : Vous pouvez créer grâce aux Actions Scripts, les segments suivants :

- Des droites avec `lineTo()`
- Des courbes avec `curveTo()`

Si vous faites se succéder des séries de droites et/ou courbes, vous pouvez réaliser des formes géométriques. L'exemple de la carte de France ci-dessus est tout simplement le résultat de la création de nombreuses droites effectuée en boucle dans un `for()` avec 2 listes (pour les coordonnées).

Création d'une droite : Tapez simplement les lignes d'instructions ci-dessous sur une image clé.

```
_root.createEmptyMovieClip("triangle", 1);
triangle.lineStyle(1, 0x63639C, 100);
triangle.moveTo(75, 75);
triangle.lineTo(175, 75);
```

- Vous devez tout d'abord créer un clip vide que vous nommez. Vous lui donnez également un numéro de niveau unique.
- Vous définissez ensuite les attributs de votre trait : Epaisseur, couleur (en hexa) et l'opacité du trait.
- Vous placez le début de votre trait avec `moveTo()` (coordonnées X et y classiques)
- Vous terminez votre trait par l'instruction `lineTo()` en indiquant les coordonnées de l'autre extrémité de votre droite.

Création d'une courbe : Il s'agit de la même technique que celle d'un trait, avec un paramètre supplémentaire pour définir l'amplitude et le sens de la courbe. Le fichier [formesdynamiques2.fla](#) vous montre un exemple plus complexe pour vous familiariser avec les coordonnées d'une courbe.

```
_root.createEmptyMovieClip("courbe", 4);
courbe.lineStyle(1, 0x63639C, 100);
courbe.moveTo(10, 150);
courbe.curveTo(200, 100, 390, 150);
```

Les valeurs 390 et 150 correspondent à l'extrémité du tracé (l'autre étant celle qui est fixée avec l'instruction `moveTo()`). Les valeurs 200 et 100 sont donc les valeurs du point directeur qui précise l'amplitude et la direction de la courbe.

Création d'une forme : Il s'agit ni plus ni moins de la création de plusieurs droites comme nous le disions en introduction.

```
_root.createEmptyMovieClip("carre", 2);
carre.lineStyle(1, 0x63639C, 100);
carre.moveTo(50, 50);
carre.lineTo(50, 100);
carre.lineTo(100, 100);
carre.lineTo(100, 50);
carre.lineTo(50, 50);
```

Création d'une forme pleine : Là aussi, il s'agit de la même technique que celle d'une forme (ci-dessus), il suffit juste d'ajouter deux lignes d'instructions avant et après le script, pour indiquer le début

```
_root.createEmptyMovieClip("carre2", 3);
carre2.beginFill(0x63639C, 50);
carre2.lineStyle(1, 0x63639C, 100);
carre2.moveTo(150, 50);
carre2.lineTo(150, 100);
```

```
carre2.lineTo(200, 100);  
carre2.lineTo(200, 50);  
carre2.lineTo(150, 50);  
carre2.endFill();
```

En utilisant `beginFill()` et `endFill()`, vous pouvez remplir les formes que vous créez avec la couleur spécifiée en hexa et son degré d'opacité. Finissez votre forme par `endFill()` sans oublier les parenthèses.

RÉALISATION DE FORMULAIRES

Fichier *formulaires fla*

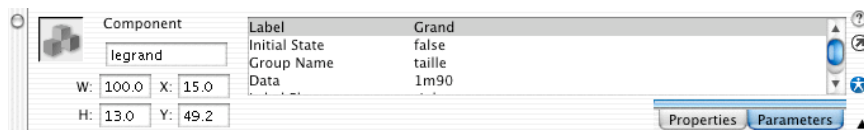
Présentation : Il était déjà possible de réaliser ses propres éléments de formulaires dans Flash 5 mais cela nécessitait un bon niveau de programmation. Flash MX propose non seulement une palette avec des composants déjà créés mais également de nombreuses méthodes et propriétés pour les accompagner. Vous pouvez intégrer dans vos animations, les éléments suivants :

- Boutons radio
- Cases à cocher
- Menus déroulants/listes
- Boutons
- Ascenseurs pour texte
- Ascenseurs pour image

Il est important de préciser que ces composants ne se limitent pas à une utilisation liée aux formulaires. Vous pouvez les placer n'importe où, à n'importe quel moment.

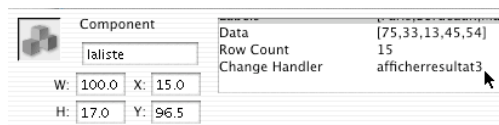
Nous allons vous présenter dans cette partie la mise en forme de vos éléments de formulaires. Chaque composant peut être réglé (ses attributs) séparément mais il est tout de même possible d'appliquer un réglage commun à tous les composants d'une image. Si vous effectuez un réglage sur un composant en particulier, il ne sera plus possible de le changer via la méthode globale. Espérons qu'il s'agisse d'une erreur de manipulation de notre part ou d'un bug. Vous pourrez vérifier ce que nous avançons dans le fichier *formulaires fla*.

Intégration : L'intégration d'un composant se fait par simple glisser-déplacer. Vous devez ensuite régler les propriétés de l'élément inséré via la palette Composants Parameters ou celle des Propriétés. Cela revient au même. Pour nommer un élément de formulaire dans la palette des Propriétés, utilisez la case de saisie dans le coin supérieur gauche (Image 3.14).



3.14 : La palette Propriétés (onglet Paramètres) affiche les mêmes réglages que celle des Composants Parameters.

Vous remarquerez très rapidement que chaque élément de formulaire placé sur la scène doit posséder une fonction qui peut être partagée. En effet, lorsque l'utilisateur modifie un élément (coche un bouton radio ou une case, sélectionne une commande dans un menu, etc.), la fonction rattachée à chaque élément analyse le changement apporté puis exécute les instructions programmées (Image 3.15).



3.15 : La fonction à laquelle vous faites référence ne doit pas porter les parenthèses habituelles.

Pour compléter la copie d'écran ci-dessus, voici la fonction qui a été créée sur l'image clé où se trouve cet élément.

```
function afficherresultat3(me) {  
    resultat = me.getSelectedItem().data;  
    resultat2 = me.getValue();  
    lindex = me.getSelectedIndex();  
}
```

Le *me* fait référence à l'élément de formulaire qui fait appel à cette fonction. Référez-vous au fichier *formulaires fla* pour effectuer différents essais. Vous comprendrez très vite. Oui, même les longs à la comprenette !

Si le paramètre Change Handler d'un élément n'est pas réglé, vous pouvez alors lui attribuer la fonction avec le code ci-dessous.

```
laliste.setChangeHandler("afficherresultat3");
```

Cette ligne d'instruction peut elle-même se trouver dans une fonction, ce qui aurait pour effet de changer l'attribution d'une fonction à un élément.

Il est possible également de tester les propriétés sans passer par ces Change Handlers. Voyez plutôt l'exemple ci-dessous :

```
on(press) {  
    if (legrand.getState() == 1) {  
        result = "OUI";  
    } else {  
        result = "NON";  
    }  
}
```

Nous reparlerons de ces propriétés plus loin dans cette partie.

Mise en forme des composants : La couleur de chaque composant d'un formulaire que vous placez sur la scène peut être personnalisée ainsi que l'aspect du texte qui l'accompagne.

Vous pouvez placer vos lignes d'instructions à n'importe quel endroit, sur l'image clé où se trouve(nt) le(s) composant(s) ou dans un gestionnaire de type on(press) ou onclipevent(load).

Exemple : menudyna est dans notre exemple, un menu déroulant.

```
menudyna.setStyleProperty("textColor", 0xFFFF00);
```

Pour la modification d'un attribut de tous les composants de l'image, il vous faut au moins deux lignes, la première pour spécifier l'attribut à modifier accompagné de sa valeur, la dernière pour appliquer la(es) modification(s).

```
globalStyleFormat.textColor = 0xfffff;
globalStyleFormat.textAlign = "right";
globalStyleFormat.background = 0x764960;
globalStyleFormat.applyChanges();
```

Selon les composants, chacun possède ses propres propriétés. Voici une liste non exhaustive des propriétés disponibles dans Flash MX.

Nomdinstance.arrow	Nomdinstance.highlight	Nomdinstance.textColor
Nomdinstance.background	Nomdinstance.highlight3D	Nomdinstance.textDisabled
Nomdinstance.backgroundDisabled	Nomdinstance.radioDot	Nomdinstance.textFormat
Nomdinstance.check	Nomdinstance.scrollTrack	Nomdinstance.textIndent
Nomdinstance.darkshadow	Nomdinstance.selection	Nomdinstance.textItalic
Nomdinstance.disabled	Nomdinstance.selectionDisabled	Nomdinstance.textLeftMargin
Nomdinstance.face	Nomdinstance.selectionUnfocused	Nomdinstance.textRightMargin
Nomdinstance.fontFace	Nomdinstance.shadow	Nomdinstance.textSelected
Nomdinstance.fontName	Nomdinstance.text	Nomdinstance.textSize
Nomdinstance.fontSize	Nomdinstance.textalign	Nomdinstance.textUnderline
Nomdinstance.foregroundDisabled	Nomdinstance.textBold	

Exemple 1 : Ascenseur - Changement de la couleur de fond et des flèches.

```
on (press) {
    labarre.setStyleProperty("scrollTrack", 0x63639C);
    labarre.setStyleProperty("arrow", 0xED6412);
}
```

Exemple 2 : Menu déroulant - Changement de l'alignement et de la couleur du texte, ainsi que la couleur de fond .

```
on (press) {
    menudyna.setStyleProperty("textColor", 0xFFFF00);
    menudyna.setStyleProperty("textAlign", "right");
    menudyna.setStyleProperty("background", 0x666666);
}
```

Exemple 3 : Changement d'attributs de tous les composants de l'image.

```
on (press) {
    globalStyleFormat.textColor = 0x497656;
    globalStyleFormat.radioDot = 0xFFFFFFFF;
    globalStyleFormat.arrow = 0x497656;
    globalStyleFormat.scrollTrack = 0x497656;
    globalStyleFormat.textColor = 0xfffff;
    globalStyleFormat.textAlign = "right";
    globalStyleFormat.background = 0x497656;
    globalStyleFormat.applyChanges();
}
```

(N.B. : radioDot définit la couleur du point de sélection du bouton radio, arrow définit celle des flèches de l'ascenseur, et scrolltrack celle du fond de l'ascenseur).

Méthodes des composants : Lorsque vous remplissez un formulaire, vous terminez par cliquer sur le bouton Valider (ou Annuler), c'est à ce moment là que le traitement du formulaire s'exécute. Lorsque vous avez fini de composer la mise en page de votre formulaire, vous devez programmer vos fonctions qui vont gérer les changements d'états des composants.

Chaque élément possède ses propres méthodes et elles sont nombreuses. Pour pouvoir faire vos premiers pas avec les formulaires dans Flash MX, voici quelques exemples que vous pouvez retrouver dans le fichier *formulaires fla*.

- Pour un bouton radio sélectionné, la valeur qui lui est attribuée s'affiche dans une variable intitulée *commentaire* qui se trouve sur la scène :

```
function afficherresultat(me) {
    commentaire = me.getValue();
}
```

- La sélection d'une entrée dans un menu déroulant affiche la valeur correspondante dans une variable intitulée *lenom* qui se trouve sur la scène.

```
function afficherresultat2(me) {  
    lenom = me.getSelectedItem().data;  
}  
  
function afficherresultat3(me) {  
    resultat = me.getSelectedItem().data;  
    resultat2 = me.getValue();  
    lindex = me.getSelectedIndex();  
}
```

CHARGEMENT D'UNE IMAGE JPEG

La technique est des plus simples si vous avez déjà l'habitude de charger des swf. Vous devez en effet utiliser la même commande avec les mêmes paramètres.

```
loadMovie("monimage.jpg","cadre")
```

ou

```
loadMovieNum("monimage.jpg",1)
```

Précisons tout de même qu'il n'est pas possible de charger une image au format gif.

Vous ne connaissiez pas la technique de chargement d'un SWF avant aujourd'hui ? Bon, aller, c'est notre jour de bonté !

Vous utilisez loadMovieNum lorsque le chargement se fait directement sur la scène. Le numéro doit toujours être unique, sinon il remplace un précédent chargement avec le même numéro. Attention, votre image doit alors être de la même taille que celle de la scène.

Vous utilisez loadMovie si vous souhaitez placer votre image chargée "dans" un clip ou un bouton (N'oubliez pas qu'à présent, les instances de boutons portent des noms). Elle va plus précisément prendre la place de l'instance. Cela vous permet à partir de ce moment là de contrôler l'image avec les mêmes propriétés que celles d'une occurrence de type clip ou bouton (_x, _xscale, _width, _alpha, _visible, etc...)

LES PROPRIÉTÉS DE LA SCÈNE

Il est enfin possible de connaître les dimensions de la fenêtre dans laquelle se trouve la scène !

```
on (press) {
    larg = Stage.width;
    haut = Stage.height;
}
```

Lorsque l'utilisateur clique sur le bouton contenant ce code, les dimensions de la fenêtre où se trouve la scène sont alors stockées dans les variables intitulées *larg* et *haut*.

Référez-vous à la partie sur les listeners pour apprendre la technique qui permet d'exécuter une action si l'utilisateur redimensionne la fenêtre où se trouve la scène.

LES CAPABILITIES

Nous pouvions jusqu'à présent connaître la version de Flash, il est à présent possible d'obtenir d'autres informations relatives à la machine et au système.

- Résolution de l'écran
- Possibilités sonores de la machine.

CHARGEMENTS ET ENVOI DE DONNÉES

A ce jour, à l'heure où nous diffusons ce document, Yazo.net ne dispose d'aucune information précise à ce sujet pour vous expliquer le fonctionnement. Il semblerait que ces commandes fonctionnent comme les autres (loadVariables, loadVariablesNum) mais aucune doc ne donne d'exemples précis.

CONCLUSION

Ce tutorial de mise à niveau (gratuit car chez Yazo.net, nous sommes presque philanthropes) va vous permettre de passer rapidement à Flash MX. Nous espérons juste que vous ne nous noierez pas trop de mails de questions car en ce moment, David de Yazo.net prépare un livre sur Flash MX et il ne peut pas répondre aux questions. Je vous entends d'ici : Oui ben il doit écrire toute l'année alors, car il répond plus aux mails comme il le faisait avant. "Accusé, levez-vous et qu'avez-vous à dire pour votre défense ?". "Heeeuuuuu, et bien je dois avouer que je ne fais rien de mes journées en dehors de donner des cours à mes étudiants (ARMS, CRM et CDM) aux Gobelins, répondre à mes mails, réaliser quelques prods, répondre en priorité aux questions de mes étudiants, ma vie privée et le reste. Et puis n'oubliez pas, il y a le forum. Non, pas pour moi mais pour vous. Pour finir, comme le blaireau de base qui passe à la télé, je passe le bonjour à tous ceux qui interviennent sur le forum de Yazo.net et aux étudiants : des Gobelins, de Nanterre, de Sup de Pub, de l'IESA, de l'ESIEE et, je demande pardon aux écoles que je n'ai pas citées ici, celles dans lesquelles j'ai fait cours. Pour Yazo.net, rassurez-vous, une mise à jour est en préparation.