

Léost

INFORMATIQUE

Solution multimédia
Conseil - Formation
Audit en informatique

FLASH

INTRODUCTION A LA PROGRAMMATION

Ce cours a pour but d'apporter une aide aux personnes ayant suivi une formation (entreprises, étudiants...) pour retrouver facilement leurs repères. Si vous n'avez jamais suivi de formation et que vous l'envisagez, vous pouvez me contacter : en tant qu'organisme de formation, une convention de formation peut être établie, ce qui permet une prise en charge par votre FAF (Fond d'Aide à la Formation).

Les parties théoriques sont omises pour ne garder que la partie ludique de l'animation avec Flash.

De toute façon, en suivant ce didacticiel, vous vous divertirez en créant une « scène de vie » du type de ce que l'on peut voir sur les îles de Glénan, ceci à défaut de devenir un maître de l'animation avec Flash.

Pour les étudiants ayant suivi la formation sur plateforme MAC OS, il faut lire touche « Pomme » à la place de [Ctrl]. Cet exemple, contrairement aux premiers cours en ActionScript, ne parle pas de la trigonométrie que les étudiants n'apprécient pas à sa juste valeur. Pour ceux qui sont réfractaire aux maths, il suffit de prendre les fonctions telles qu'elles ;-)

Ce support, ainsi que les médias, sont téléchargeables sur <http://cours.glenan.fr/>. Choisir formation – Flash - Animation. Ils sont disponibles au format ZIP et auto-décompactables (pour PC).

Pour mieux appréhender le résultat de l'animation avec Flash, il est conseillé de regarder la version en ligne de ce support.

Ce document n'est pas sous licence GNU/GPL. Il ne peut être utilisé pour des formations que sous réserve d'accord écrit avec l'auteur et la partie relative aux droits d'auteur ainsi que le présent copyright doivent apparaître clairement.

Droits d'auteur :

Philippe LÉOST pour les objets : bateau, goéland, bouée...

TABLE DES MATIERES

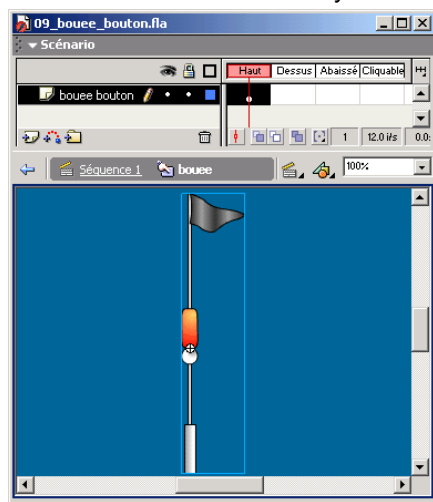
Table des matières	2
Bouton et action	3
Etats du bouton.....	3
Action et bouton.....	4
Notion de programmation	5
Objet et programmation	5
Premier code et syntaxe à point	6
Repères sur la scène.....	7
Propriétés d'objet.....	8
Variables et étiquettes	9
Tests conditionnel.....	10
Programmation et loi de Pareto	11
La pêche miraculeuse.....	12
Un poisson variable	12
Tu la boucles !	13
While do	13
Do until.....	13
For.....	13
For In.....	13
Stop et Encore	14
C'est mon choix !	14
Quel poisson choisir?	14
Pour ne plus voir qu'une seule tête	15
Combien j'ai pêché ?	17
On est les champions.....	17
Ça nous file des boutons	18
Ça va durer encore longtemps ?	19
Elle va encore nous faire une scène !	20
Help me if you can, I'm feeling down	21
Ne me quitte pas !.....	21
Same player shoot again	22
This is the end, my friend.....	23
Un bug, des bogues	24
Finalisation de l'animation	25
Tester l'animation	25
Exporter l'animation	25
Incorporer l'animation	26
Lecteur autonome.....	26
Macromedia Dreamweaver	26
Autre logiciel de gestion de site	27
PowerPoint et autres produits Office	27
Conclusion	28
Quelques conseils.....	29
Code ActionScript	30

BOUTON ET ACTION

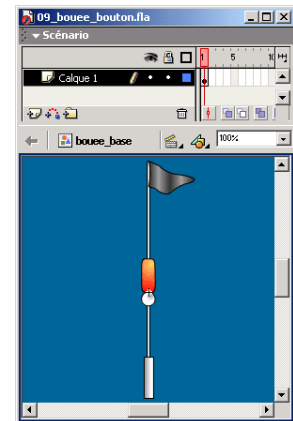
ÉTATS DU BOUTON


Ouvrir l'animation « Bouee fla » : elle contient un objet « bouee_base » accessible depuis la bibliothèque qu'il est souhaitable d'ouvrir à ce moment ([Ctrl] [L]). Repasser dans l'animation de base, sans fermer la bibliothèque de « bouee fla ». Vous pouvez également dessiner un symbole clip bouée.

Créer un calque bouée et déplacer la bouee_base de la bibliothèque sur la scène, dans ce calque. Utiliser le menu « Insertion – Convertir en symbole... » et demander un symbole de type bouton ayant pour nom « bouée bouton ».



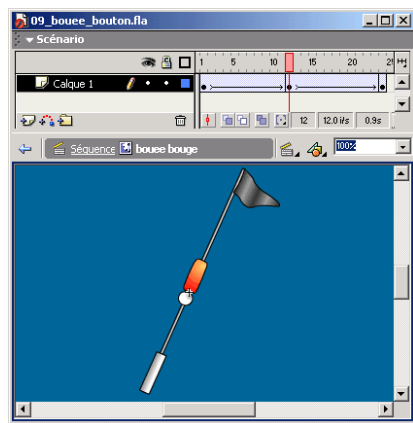
Ouvrir la bibliothèque et faire un double clic sur le symbole « bouée bouton ». Le symbole apparaît en plein écran avec ses 4 états de boutons. Seul le premier état « haut » est rempli (un point rond est visible sur l'image « haut » du calque). Faire un clic droit sur l'image Dessus (quand l'utilisateur passe la souris dessus) et demander « insérer une image clé ».



Avec l'outil rotation ( Outil transformation libre) faire une rotation de la bouée. Penser à bien positionner le centre de gravité de la bouée sur la croix de l'image pour un rendu plus réaliste.

Faire de même sur l'image abaissée (quand l'utilisateur clique) et changer l'inclinaison de côté. L'image cliquable indique la zone réactive à l'action (sensible au clic). Dans le cas de la bouée, n'insérer pas une image clé qui aurait le problème de reprendre la dernière image, mais insérer une image clé vide et dessiner une forme rectangulaire englobant les 3 étapes du bouton. Pour plus de réalisme, il est possible de modifier la forme pour qu'elle ressemble à un diabololo expansé. L'objet est fonctionnel sur la scène en lançant l'animation.

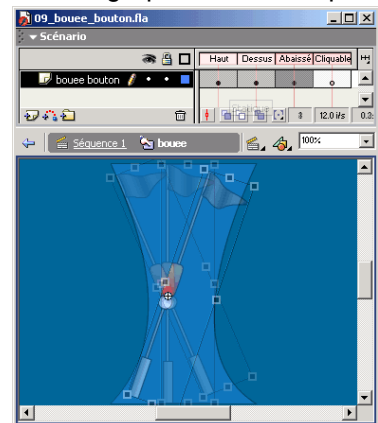
Pour être plus naturel (une bouée bouge en fonction des vagues). Utiliser le menu « Insertion – Nouveau symbole » pour créer un clip « bouee bouge ».



« bouee bouge ».

Sur l'image 1 du clip, mettre « bouee_base ». Le faire pivoter et insérer une image clé sur l'image 24 (la bouée fera un va et vient en 2 secondes). Insérer une image clé sur l'image 12 et la faire pivoter dans l'autre sens. Penser à bien positionner le centre de gravité. Cliquer sur le mot calque 1 du clip « bouée bouge » et dans le panneau propriété, demander une interpolation de mouvement. En regardant les images du clip, la bouée bouge progressivement pour revenir à son état initial... Double cliquer sur le bouton bouée de la bibliothèque pour l'ouvrir. Supprimer la bouée de l'image « Haut » et déposer à la place la « bouee bouge ».

En lisant la scène, la bouée « bouge ». Quand l'utilisateur se déplace dessus, elle se fige d'un côté et au clic elle se fige de l'autre côté. Le bouton est fonctionnel mais inefficace, tout comme un interrupteur non relié au courant. Le fichier final est visible en cliquant ici.



ACTION ET BOUTON

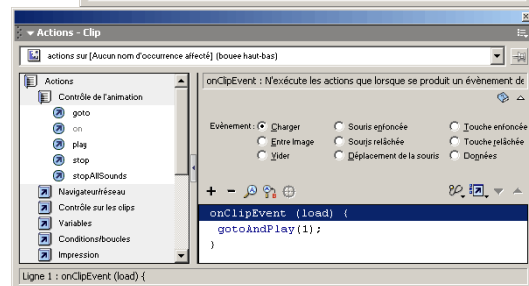
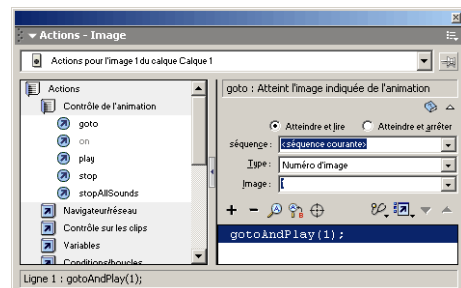
L'action du designer peut s'arrêter ici pour laisser place au programmeur. Quelles que soient ses préférences, il est utile (voire important) d'appréhender quelques-unes des contraintes de l'autre monde (il n'y a pas 2 côtés obscurs de la force ;-).

Pour plus de détails sur ce chapitre et le suivant, voir le cours « Flash programmation ».

Notre bouton « bouée » doit se voir attribuer une action lors du clic pour ressembler à un bouton. Cependant, dans flash la programmation peut s'effectuer sur différents types d'objets. Elle est relativement proche de la programmation javascript, même si l'approche en est plus conviviale et déroutante pour le débutant. A chaque « objet » : bouton, clip, image... peut être associée une programmation. Cependant il faut toujours avoir présent à l'esprit que les méthodes (comportement) de l'objet seront fonction de la nature de l'objet auxquelles s'applique la programmation. Si vous êtes en mode designer, il faut demander la fenêtre « Actions » ([F9]) qui se trouve au-dessus des propriétés dans l'univers programmeur. Le panneau action regroupe, sur la gauche, les différentes catégories d'actions regroupées en sous-catégories... A la fin de la liste, la dernière catégorie (en jaune et nommée index) inclue l'ensemble des « actions » listées alphabétiquement.

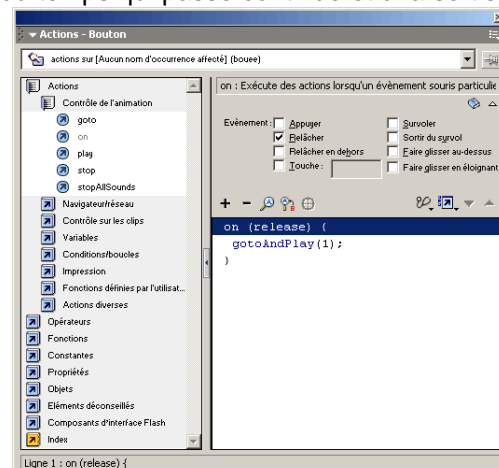
Certaines « actions » regroupées dans la catégorie « Éléments déconseillés » (surlignés en vert dans Flash 5) ne doivent pas être utilisées sous peine de voir l'animation ne fonctionne pas avec les prochains lecteurs flash.

En cliquant sur l'image 1 du calque « bouée », un double clic sur l'action « goto » dans la catégorie « Actions – Contrôle de l'animation » entraîne l'apparition, en bas et à droite du panneau, d'une ligne de code de type « gotoAndPlay(1); ». Comme dans d'autres langages, il existe un symbole de fin d'instruction (ici le « point-virgule ») qui permet au programmeur de mettre plusieurs lignes de code sur une même ligne dactylographiée. De même, pour faciliter le repérage de fonction ou de variable (l'espace étant prohibé dans beaucoup de langages), la première lettre de « gotoAndPlay(1); », des listes déroulantes, cases d'options ou à cocher... sont à disposition. Ce mode Normal (assisté) et très pratique pour les débutants car il vérifie la syntaxe tapée.



En cliquant sur l'icône ci-contre il est possible de passer en mode expert, de type éditeur de texte. En lançant l'animation il ne se passe rien. Flash, comme d'autres langages plus ou moins orientés objet, attend qu'un événement se produise. Si l'action Goto est lancée sur un clip, une instruction comme ci-contre apparaît. Au-dessus de l'instruction « gotoAndPlay(1); », une instruction de type « onClipEvent (load) { ». Flash attend donc qu'un événement de clip se produise et plus particulièrement un événement de chargement de l'animation. La parenthèse en fin de ligne indique que si l'événement se produit, la ou les action(s) contenue(s) jusqu'à la fin de la parenthèse seront effectuées. Il est possible de cocher un autre événement comme « Entre Image ». Cet événement est intéressant car, même avec une instruction « Stop », la lecture du temps qui passe continue et à la sortie de l'image courante le programme retourne au début de l'image courante, permettant de tester si certains événements se produisent.

Sur un bouton (utiliser le bouton bouée) les événements sont comme ci-contre. Plusieurs actions différentes des clips sont possibles. Par défaut, avec une souris, l'action s'exécute quand l'utilisateur relâche le clic de la souris et non quand il appuie. Sur « gotoAndPlay(1); », il est possible de choisir la séquence, le type d'accès (dont étiquette) et, au final, l'endroit où va pointer l'action. Pour l'instant, lancer l'instruction en place qui renvoie à l'image 1. Chaque clic sur la bouée fait retourner l'animation à l'image 1. Le résultat est un peu stupide, mais ce n'est qu'un début. Le fichier final est visible en cliquant ici.



NOTION DE PROGRAMMATION

Il n'est pas question ici de cours de programmation, mais uniquement de découvrir la programmation de façon pragmatique et ludique, sans utiliser de méthode d'analyse...

Il y est préférable d'utiliser l'animation 10_programmation_debut fla afin d'être certain de démarrer avec les bons objets (dont la taille du poisson en mouvement).

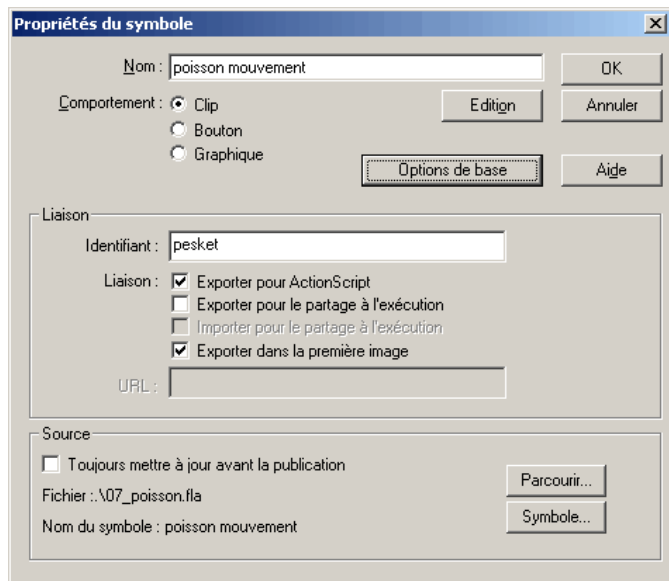
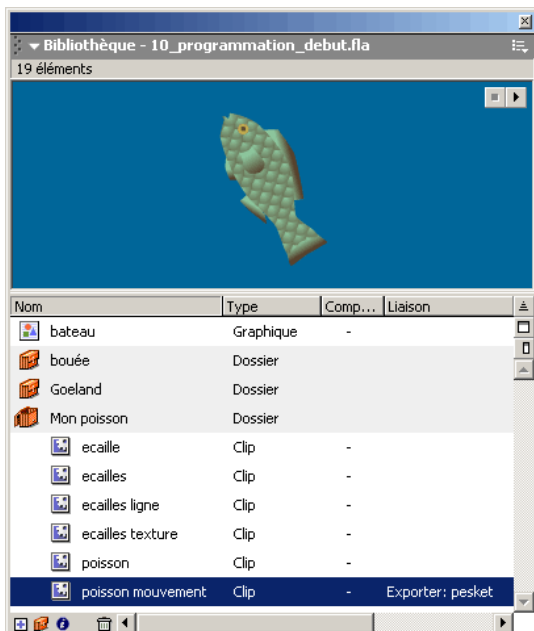
OBJET ET PROGRAMMATION

Il serait plus judicieux de faire sauter le poisson à chaque clic que de revenir à l'image 1. Noter d'ailleurs qu'avec la technique utilisée précédemment le poisson continue imperturbablement sa trajectoire, contrairement aux goéland, bateau et mer, car c'est le clip animé (cas du poisson) qui est positionné dans la barre des temps et non l'objet (bateau) qui est animé dans la ligne de temps de la séquence courante.

De la même façon, supprimer le code lié au bouton bouée et le remplacer par un Stop. Au clic sur la bouée les clips goéland, bateau et mer s'arrêtent, mais le poisson continue sa route imperturbable et le goéland continue son animation interne (battre des ailes) tout en restant sur place. Le bouton bouée est lui aussi fonctionnel.

Reprendre l'animation précédente et l'enregistrer sous un autre nom. Le poisson sera géré par programmation, donc supprimer le calque poisson en l'envoyant à la poubelle. Cliquer sur le bouton bouée dans l'animation et enlever la programmation (cliquer sur *on (release)* et appuyer sur la touche *Suppr*). Ou cliquer ici pour charger le fichier flash résultant.

Ouvrir la bibliothèque ([Ctrl] [L] ou [F11], plus conforme au reste des produits MX).



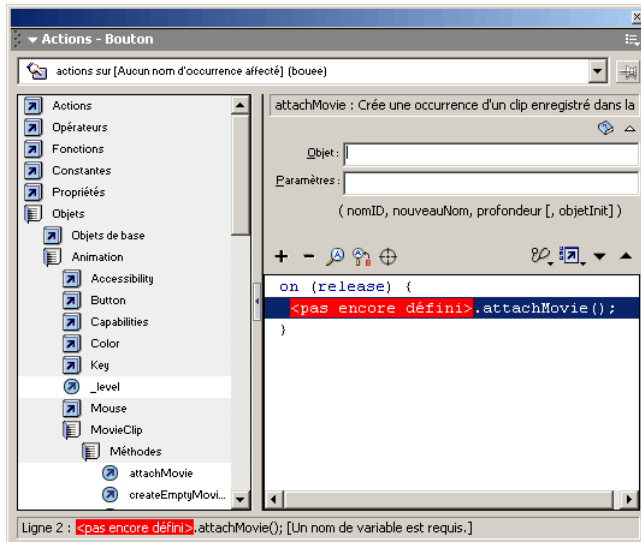
Dans le dossier poisson, faire un clic-droit Propriété sur le clip « poisson mouvement ».

Cocher la case « Exporter pour ActionScript » et donner un nom dans identifiant (pesket dans l'exemple).

Cette technique permettra d'appeler le « pesket » par la programmation. Toutes les occurrences de pesket auront les propriétés de pesket et, comme pour les écailles du poisson, le fait d'en modifier une modifiera toutes les occurrences.

PREMIER CODE ET SYNTAXE A POINT


Cliquer sur le bouton bouée sur la scène (séquence 1) pour le sélectionner. La programmation qui va être mise en œuvre consiste à faire sauter un pesket quand on clique sur la bouée. Dans la fenêtre Action (touche [F9]), vous devez avoir Action – Bouton et en dessous : *action sur ... (bouée)*. Si ce n'est pas le cas, il faut cliquer sur la bouée dans la scène.



Choisir l'action « Objets – Animation – MovieClip – Méthodes – attachMovie ». Le double clic sur attachMovie permet d'écrire le code. L'évènement *on (release)* est correct dans ce cas.

Certaines parties du code ne sont pas définies et apparaissent en rouge. Il faut attacher un movieclip à un objet avec un certain nombre de paramètres.

L'environnement ressemble aux fonctions assistants de certains logiciels, comme les tableurs...

 Cliquer dans la zone de saisie objet. Ceux qui travaillent avec Macromedia Dreamweaver, auront remarqué l'icône cible qui s'active. Cliquer sur l'icône cible et la boîte de dialogue « Insérer un chemin cible » apparaît.

L'instance de l'objet « pesket » va être créée mais sera dans le vide, aussi elle doit se rattacher à quelque chose. Dans le cas présent on le rattache à l'animation de départ qui a pour nom « *_root* » (c'est d'ailleurs la seule cible disponible).

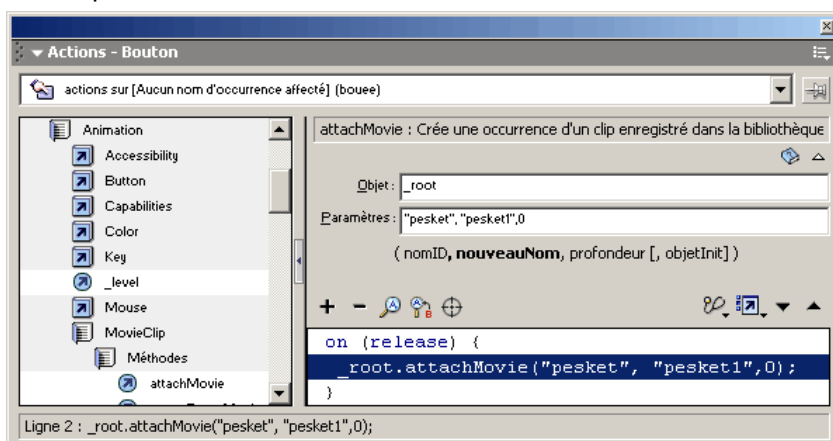
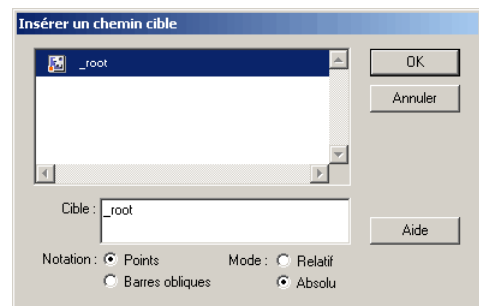
En cochant la case d'option « relatif », This apparaît (que reconnaîtront ceux qui connaissent Javascript).

Il est préférable d'utiliser la syntaxe à Points.

La syntaxe à point utilise le point comme séparateur d'objet, par exemple une touche de clavier peut s'écrire :

clavier.touche

La syntaxe indique après validation : *_root.attachMovie()*; . Ce qui signifie qu'il faut créer une instance d'un clip et l'attacher à la racine de la scène de l'animation.



Il faut donc maintenant définir l'occurrence de quel objet (nomID) est créée et lui donner un nom (nouveauNom).

NomID = "pesket" et nouveau Nom = "pesket1".

Mette les guillemets pour ne pas les confondre avec des variables.

Pour la scène de « la pêche miraculeuse », plutôt que de créer chaque occurrence nommée de poisson, il est possible d'utiliser une variable à la place du chiffre

concaténé avec le texte « pesket » à l'intérieur d'une boucle de type « for » par exemple.

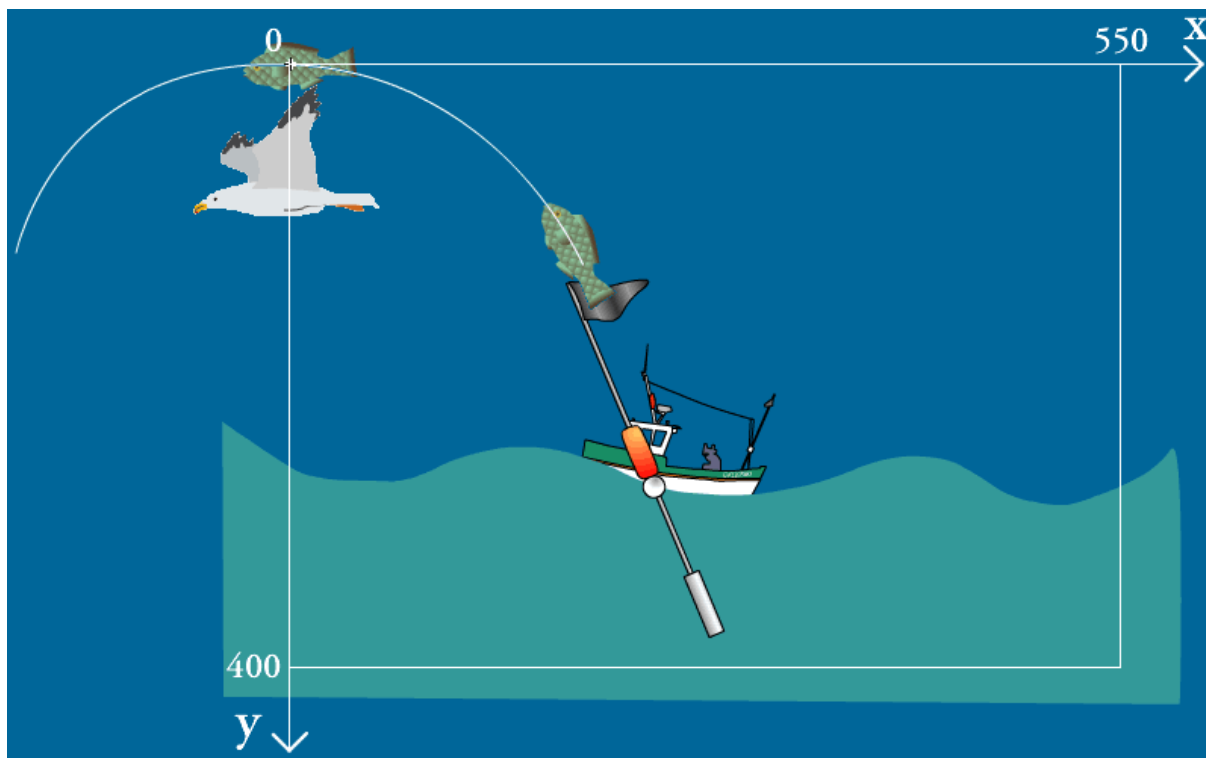
Il ne faut pas oublier d'indiquer la profondeur qui correspond à la « couche » sur laquelle est créé l'objet. Plus le chiffre est petit, plus il est près de la scène et plus il est grand, plus il est près de « l'internaute ». Lancer l'animation ([Ctrl] [Entrée]) et cliquer sur la bouée. Le poisson semble sortir de la bouée et disparaît dans le ciel.

Le fichier final est visible en cliquant ici.

REPERES SUR LA SCENE

La scène de notre animation a les dimensions de départ, à savoir 550 x 400 pixels. Le repérage des pixels se fait dans un repère orthonormé. L'axe des X est horizontal et orienté vers la droite. L'axe des Y est vertical, mais orienté vers le bas (comme toujours en informatique).

Quand un objet est positionné sur la scène depuis la bibliothèque, il est, comme tous les objets, circonscrit dans un rectangle et possède un centre qui sert d'accroche pour les trajectoires et de point de symétrie pour les transformations. Dans le cas de notre poisson, il se trouve au zénith de la trajectoire du poisson.



Si un symbole est positionné sur la scène et que dans les propriétés de l'objet soit inscrit 0 en X et en Y, le coin supérieur gauche sert de référence.

Dans la programmation, comme on peut le noter sur le dessin ci-dessus, c'est le point central de « poisson mouvement » alias « pesket » qui est utilisé. Toute occurrence créée est positionnée par son centre au point (0,0)

Pour notre animation il faut calculer l'endroit où doit se trouver le point central de pesket. Il peut se trouver au milieu de la scène horizontalement soit $x=225$ et à une position sur l'axe des Y qui permette de le voir. En se servant de l'image ci-dessous et en utilisant une règle de trois, il est facile de voir que le positionnement « idéal » de l'occurrence est de 220 (à 10 pixels près), donc $y=220$.

Il faut maintenant apprendre à notre poisson, non pas à sortir de la bouée, mais de la mer en utilisant les coordonnées (225,220).

PROPRIETES D'OBJET

Pour rajouter au code lié au bouton bouée de notre animation :

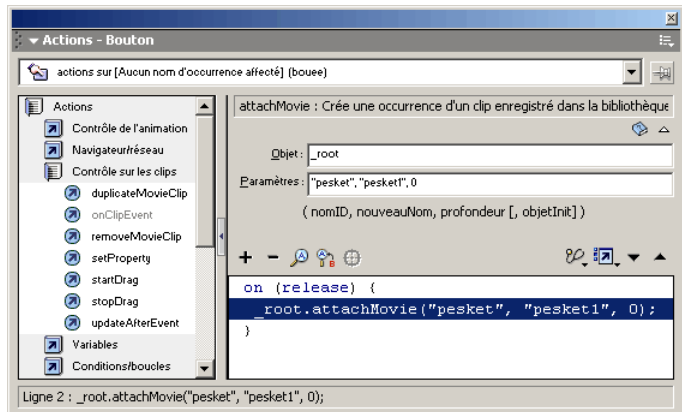
Cliquer sur la bouée et afficher la fenêtre Actions ([F9]). Cliquer sur la ligne définissant l'attachement de pesket à l'animation.

Sélectionner l'action « Actions – Contrôle sur les clips – setProperty »

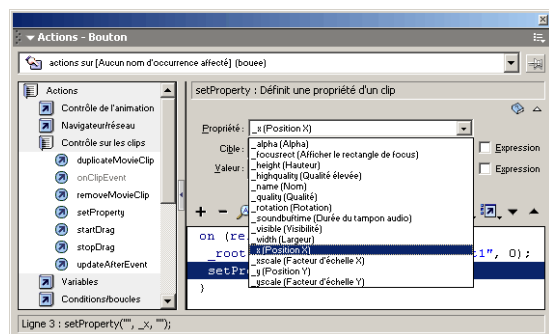
La ligne de code apparaît sous la précédente, si ce n'est pas le cas, voir en fin de page.

En cliquant sur la liste déroulante, toutes les propriétés accessibles apparaissent avec, entre parenthèse, leur traduction en français. Dans le cas de pesket il faut sélectionner les positions en X et en Y.

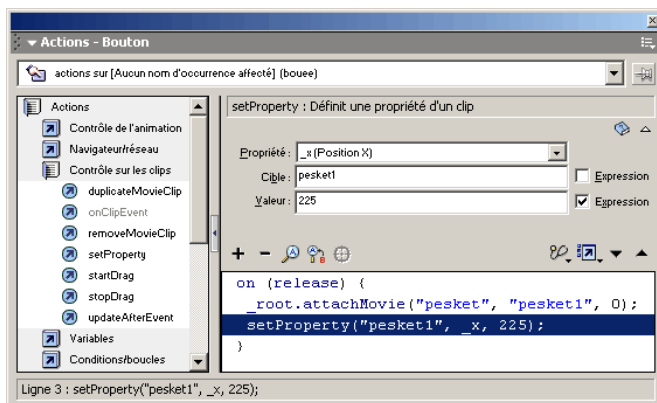
Après avoir défini la position en X, il faut ensuite définir la position en Y. Dans la génération du code, des guillemets se mettent autour du mot pesket. Pour la position en X, taper 225 et des guillemets se mettent de chaque côté. Comme 225 est un chiffre et pas du texte, il suffit de cocher la case expression, pour voir les guillemets disparaître. Jouer l'animation et vous constatez que le pesket est toujours dans le ciel, mais qu'il s'est



Ligne 2 : _root.attachMovie("pesket", "pesket1", 0);



Ligne 3 : setProperty("pesket1", _x, 225);



```

on (release) {
    _root.attachMovie("pesket", "pesket1", 0);
    setProperty("pesket1", _x, 225);
    setProperty("pesket1", _y, 220);
}
    
```

Le fichier final est visible en cliquant ici.

Si au lieu de cliquer sur la ligne `_root.attachMovie...` vous aviez sélectionné l'accolade de fin, la ligne `setProperty` se trouve ajoutée dans une nouvelle syntaxe `on (release)`. Il est possible de compléter la syntaxe et de changer l'instruction `on (release)` par une autre : `on (rollOver)` par exemple.

Sur un bouton (ou un autre symbole) il est possible de définir différentes actions qui seront exécutées en fonction de l'événement qui se produit. Les boutons pouvant incorporer des clip (bouée bouge est un clip incorporé dans l'image haut du bouton bouée) les possibilités de programmation événementielle sont nombreuses.

```

on (release) {
    _root.attachMovie("pesket", "pesket1", 0);
}
on (rollOver) {
    setProperty("pesket", _x, 225);
}
    
```

déplacé de 225 pixels vers la droite. Pour positionner en Y, il est possible de recommencer l'opération, mais il est plus simple de déplacer la ligne contenant le positionnement en X en-dessous en ayant soin d'appuyer sur « [Ctrl] déplacement de l'objet » qui permet de dupliquer la (les) ligne(s). Faire les modifications pour obtenir un code équivalent à ci-dessous :

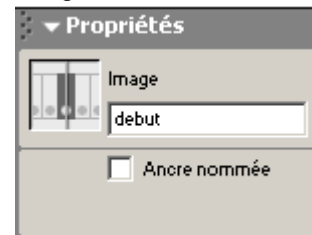
VARIABLES ET ETIQUETTES

Le poisson saute dès que l'on clique sur la bouée à l'endroit souhaité. Néanmoins, un nouveau clic sur la bouée fait disparaître le poisson qui saute de nouveau. Il est souhaitable d'empêcher le saut du poisson tant que le saut précédent n'est pas terminé.

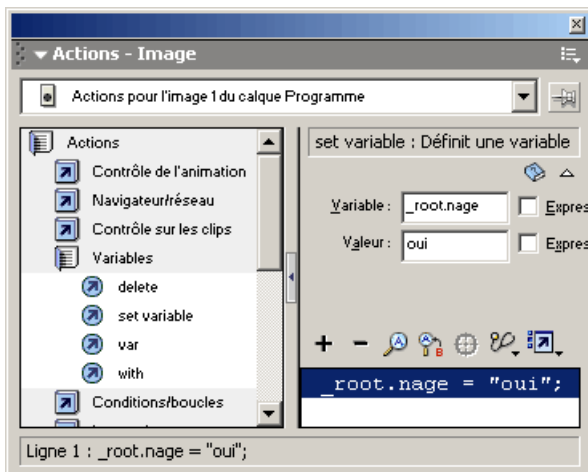
Il est possible d'utiliser une variable qui peut s'appeler « nage ». Le nom des variables est assez large, mais ne doit pas utiliser les mots-clés du langage... Le fait de donner un nom français évite ce genre de problème et permet d'être explicite pour la suite du traitement. Flash n'est pas un langage fortement typé où le type de la variable est bien défini une fois pour toute. Notre variable est de type booléen (vraie/faux). Il est possible de lui affecter la valeur « oui » quand le poisson peut nager et la valeur « non » quand il ne peut pas nager, car il nage déjà (ce n'est plus du booléen, mais cela convient pour notre exemple).

Dès le début de l'animation, la variable nage doit être initialisée à oui pour permettre au poisson de sauter. Lors du clic sur la bouée, effectuer un test conditionnel pour connaître l'état de la variable nage et, si le poisson peut nager, lancer le movieclip. Ce test est de type Si-Alors-Sinon (If-Then-Else) que l'on retrouve dans les tableurs.

Cliquer sur le calque bouée et insérer un nouveau calque appelé programme (ou prg). Il est souhaitable d'avoir ce calque en haut de l'animation et de n'y mettre que de la programmation. Si l'animation a besoin, comme dans les premiers essais du bouton d'aller à une étiquette d'image. Il est souhaitable de créer un calque étiquette sous le calque prg qui rends les étiquettes bien visibles sans avoir à les chercher dans tous les calques. Bien que non utilisé dans cette animation, il est possible de créer une étiquette de début et une de fin. Les étiquettes par rapport au numéro d'images ont l'avantage en cas d'ajout ou de retrait d'images dans l'animation de ne pas changer et donc d'appeler toujours la bonne séquence quelles que soient les modifications apportées. Pour mettre une étiquette : cliquer sur l'image clé vide sur le calque étiquette et dans propriété donner un nom à l'étiquette.



Concernant la variable nage, elle doit être présente dans toute l'animation. Il faut donc la définir comme une variable dite globale, par rapport à une variable locale qui n'est disponible qu'à un endroit, comme par exemple un compteur dans une boucle.

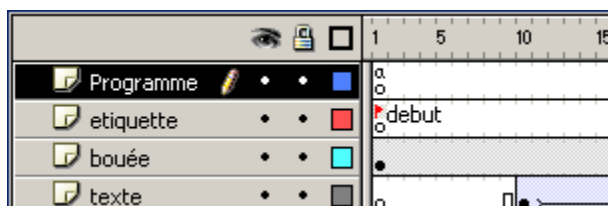


Pour que la variable soit accessible depuis n'importe où, il est possible de l'appeler _root.nage.

Sur l'image 1 du calque programme, utiliser le menu « Actions – Variables – set variable ». Dans les zones de saisie mettre nage pour le nom de la variable

Le résultat final est visible sur l'image ci-dessous où l'étiquette est représentée par un drapeau avec le nom écrit s'il n'y a pas dans la case contiguë une autre image clé vide avec une étiquette.

La programmation dans l'image est représentée par un a (comme action) même si l'image clé est vide (rond blanc).



TESTS CONDITIONNEL

Maintenant que la variable est créée, il faut la gérer au niveau du clic sur la bouée.

Le test conditionnel sera du type :

Tester si la variable nage est égale à oui.

Si c'est le cas, faire les opérations suivantes :

- Basculer la variable nage à non.
- Attacher le movieClip et le positionner au bon endroit.

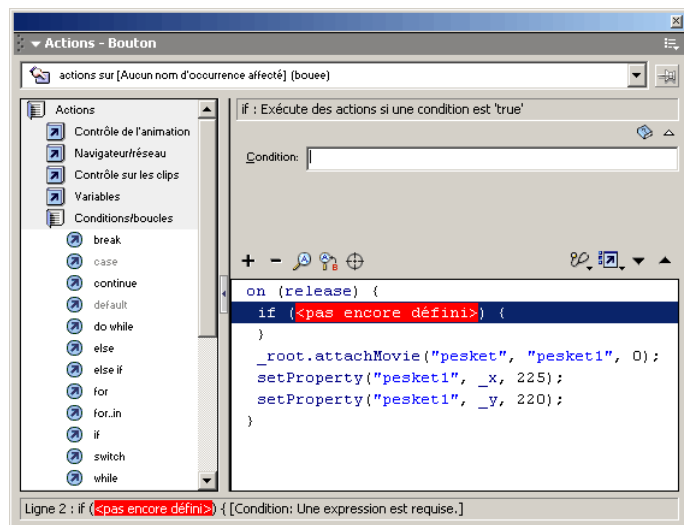
Dans le cas contraire ne rien faire.

Utiliser (Actions – Conditions/boucles – If) sur on (release).

Il existe d'autres conditions et boucles comme dans tous les langages évolués.

Il faut tester si la variable est égale à « oui ». Écrire dans condition :

« `_root.nage=="oui"` »



Nous retrouvons la variable `_root.nage`.

Le doublement du signe égal peut surprendre, mais = a 2 significations distinctes dont il faut lever l'ambiguïté en programmation :

- Symbole d'affectation de valeur (ex : `_root.nage="oui"` affecte la valeur oui à la variable `_root.nage`)
- Opérateur de comparaison (ex : `_root.nage=="oui"` qui teste si la variable `_root.nage` est égale à oui)

Dans le cas d'ActionScript (idem javascript, mais ce n'est pas le cas dans tous les langages), la différenciation du test se fait par le doublement du signe = pour la comparaison, l'affectation gardant le signe égal simple.

Maintenant, changer la variable nage à non pour empêcher le poisson de sauter, et mettre les 3 lignes liées au clip pesket à l'intérieur des accolades du test IF.

Le code ressemble à ci-dessous :

```
on (release) {
    if (_root.nage=="oui") {
        _root.nage = "non";
        _root.attachMovie("pesket", "pesket1", 0);
        setProperty("pesket1", _x, 225);
        setProperty("pesket1", _y, 220);
    }
}
```

Comme il n'y a rien à signaler dans le cas contraire, il est inutile de définir le `else` ou le `else if` logique dans cette structure de langage.

La différence n'est pas notable, hormis le fait qu'il est impossible de cliquer pour relancer le saut du poisson. Mais le poisson continue à sauter sans s'arrêter. La bouée ne sert donc qu'à déterminer le point de départ de l'action.

Plusieurs possibilités existent pour que le poisson arrête de sauter. Nous retiendrons celle qui consiste à effacer le clip et à remettre la variable nage à non quand le poisson a fini son premier saut.

Ouvrir « poisson mouvement » dans la bibliothèque. Créer un calque « programme » et sur la dernière image de ce calque, créer une image clé vide.

Modifier la variable `_root.nage` avec « Action – Variable » pour la définir à « oui »

Pour détacher le clip, sélectionner « Objets – Animation – MovieClip – Methods » et choisir « RemoveMovieClip » ou « unloadMovie ». Le code ressemble à :

```
_root.nage = "oui";
_root.pesket1.removeMovieClip();
```

En chargeant l'animation, le poisson saute quand on clique sur la bouée et ne saute plus tant qu'il n'a pas fini de sauter.

Le fichier final est visible en cliquant ici.

PROGRAMMATION ET LOI DE PARETO

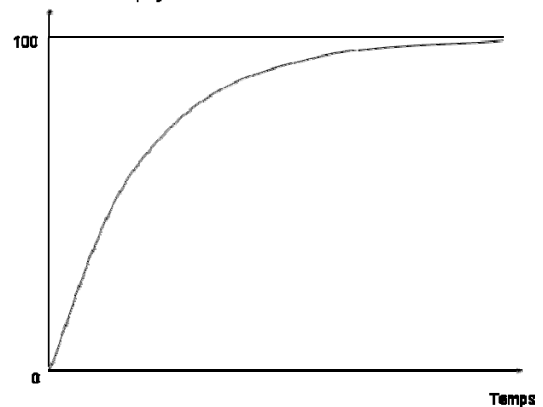
L'animation semble fonctionner, mais en regardant plus attentivement, on s'aperçoit qu'en cliquant sur le poisson après le passage de l'animation sur l'image 1, le poisson recommence à sauter sans avoir fini son saut. Ceci est compréhensible dans la mesure où à chaque passage sur l'image 1, la variable nage est remise à « oui ».

Nous touchons là un des problèmes fondamentaux de la programmation que l'on peut appréhender par la loi des 20/80 (ou loi de Pareto), déduit du concept d'ophélimité. Adaptée en informatique, elle dit qu'il faut 20% du temps pour réaliser 80% du programme. Se pose alors de façon corollaire le problème du « Good-enough developer » : jusqu'où doit-on développer un programme pour qu'il soit acceptable. Le temps passé en supplément pour augmenter son acceptabilité se traduit par un surcoût financier trop important.

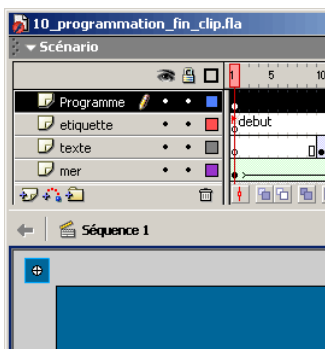
Notre exemple est simple, il est facile de trouver une solution au problème : trouver un système permettant d'initialiser la variable nage à « oui », sans qu'elle repasse inopinément à « oui ».

Comme souvent en programmation, plusieurs solutions sont envisageables. Seules 2 seront abordées pour clore ce chapitre.

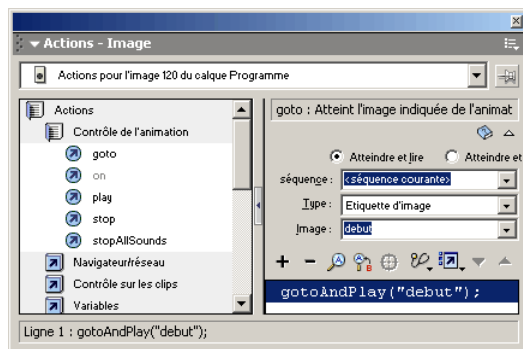
% d'avancement du projet



Utilisation d'un clip vide

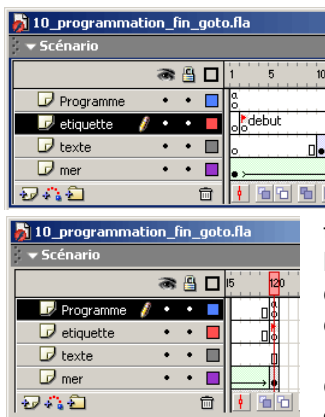


En vérifiant dans le chapitre « Action et boutons », les propriétés des clips ont une case « chargée » on (load) qui correspond à notre cas. Cette syntaxe n'est utilisée qu'au chargement du clip, donc il ne faut pas prendre les objets « poisson » qui sont rechargés à chaque clic sur

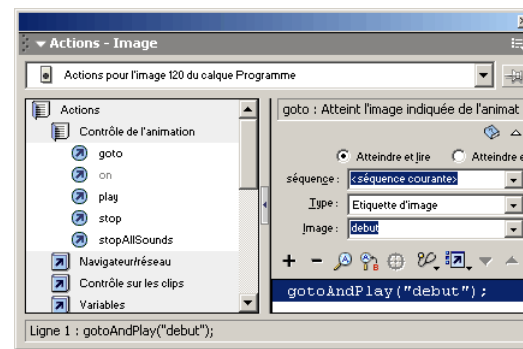


la bouée. Si aucun clip n'est chargé dans le temps imparti, il faut soit mettre un clip existant dans l'image 1 de prg ou créer un clip vide (type rectangle converti en clip). Le poser sur la scène et le faire disparaître (taille, alpha, en-dehors de la scène...). Ajouter sur le clip une action « Actions – Variables – set variable » pour définir `_root.nage = "oui"`; . Ne pas oublier de supprimer la programmation sur l'image 1 du calque Programme. Résultat en cliquant ici.

Goto again



Cette solution relativement simple consiste à déplacer l'étiquette début placée précédemment et qui n'avait pas d'utilité, pour la mettre sur l'image 2. Sur l'image de fin (image 120) du calque programme, créer une image clé vide. En programmation choisir « Actions – Contrôle de l'animation – Goto » et choisir dans le type « Etiquette d'image » et dans image « Debut », soit la ligne de commande : `gotoAndPlay ("debut") ;`



Résultat en cliquant ici.

LA PECHE MIRACULEUSE

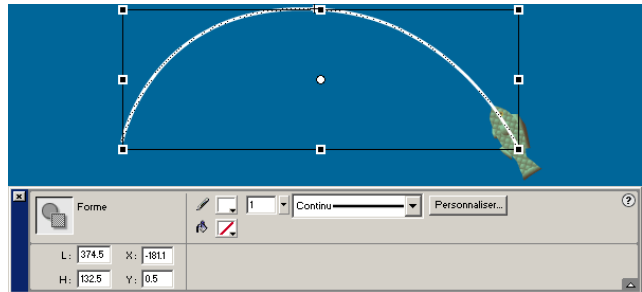
Notre action sur le bouton fonctionne et le poisson saute quand on clique sur la bouée. Tous les éléments sont en place pour faire un jeu. La règle peut être la suivante :

Au clic sur la bouée, une dizaine de poissons de tailles variables et localisés à des endroits différents sautent. Le joueur doit cliquer sur les poissons. Chaque fois qu'il attrape un poisson, il augmente son score. Au bout d'un temps défini, le jeu s'arrête et il peut « contempler » son score.

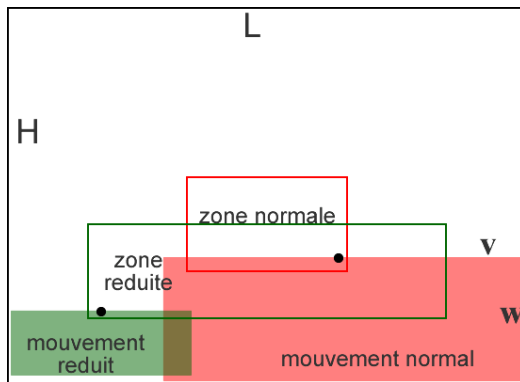
Charger l'animation 11_debut fla (qui est en fait 10_programmation_fin_goto fla)

UN POISSON VARIABLE

Il faut suivre la piste du poisson pour connaître son espace vital. En l'occurrence, sur l'image ci-contre, on retient un rectangle d'environ 380 par 130 pixels pour le déplacement (*mouvement normal*) de son centre de gravité. Pour être précis il faudrait intégrer l'épaisseur du poisson mais, comme il sera possible de cliquer au moins sur un filet, l'approximation est donc faite. Pour la suite du calcul nous représenterons ces données par les variables v et w .



Le centre du repère (le point 0) se trouve au sommet de la courbe, normalement à la moitié de la largeur et en haut du rectangle qui circonscrit la courbe (coordonnée : $(1/2 v, 0)$)



Le poisson se déplace dans une scène de 550 par 400, notés L et H . Comme seul le *mouvement normal* du poisson nous intéresse, il faut connaître la *zone normale* dans laquelle le centre du repère peut se déplacer.

Pour que le poisson soit accessible, il faudra définir sa taille qui sera toujours comprise entre la taille normale et la moitié de la taille normale. Du fait de la taille réduite de moitié de l'objet, le *mouvement réduit* est plus petit de moitié, permettant une *zone réduite* plus grande sans que le poisson sorte de l'écran dans sa trajectoire. Le poisson étant plus petit, positionner la plus bas pour qu'il donne toujours l'impression de sortir de l'eau. Avec

une plage sur l'axe des $_y$ entre 180 et 280 pixels, plus le décalage d'échelle, la formule tenant compte du facteur d'échelle S du poisson (compris entre 0,5 et 1) et d'un coefficient aléatoire R (compris entre 0 et 1) permet de positionner le poisson (grâce à ces coordonnées (x, y)) dans la zone de mouvement de façon aléatoire.

$_x = v / 2 \times S + (L - v \times S) \times R$ et $_y = 180 + 100 \times R + 100 \times (1 - S)$
soit en code ActionScript après conversion et factorisation :

```
on (release) {
    if (_root.nage=="oui") {
        _root.nage = "non";
        alea = Math.random()/2 + .5; facteur S dans notre formule
        Nombre aléatoire divisé par 2 + 1/2 pour un nombre compris entre 0,5 et 1
        _root.attachMovie("pesket", "pesket1", 0);
        setProperty("pesket1", _xscale, alea * 100); alea en % pour l'échelle
        setProperty("pesket1", _yscale, alea * 100);
        setProperty("pesket1", _x, 190*alea+(550-380*alea)*Math.random());
        setProperty("pesket1", _y, 180+100*(Math.random()+1-alea)); } }
```

TU LA BOUCLES !

Maintenant qu'un poisson apparaît de façon aléatoire dans la scène il faut, comme pour la multiplication des petits pains, créer une dizaine de poissons. Pour cela on peut utiliser une boucle (catégorie : Actions- Conditions/boucles).

Le principe d'une boucle est de répéter une partie de programme de façon automatique. Le contrôle se faisant sur une condition. Suivant les langages, plusieurs types de boucles existent. Dans quasiment tous les cas, un compteur ou équivalent permet de modifier une variable au sein de la boucle

WHILE DO

La syntaxe générale de cette boucle est :
WHILE condition DO instruction

En ActionScript, elle se structure de la façon suivante :

```
while (condition) {  
  instructions  
}
```

DO UNTIL

La syntaxe générale de cette boucle est :
DO instruction UNTIL condition

Contrairement à la syntaxe précédente, la boucle est exécutée au minimum une fois avant de passer par le crible du test.

En ActionScript, elle se structure de la façon suivante :

```
do { instructions  
} while (condition);
```

FOR

La syntaxe de cette boucle appelée aussi itération, bien connue des utilisateurs de tableurs, est :
FOR (initialisation, condition, modification) DO instruction

En ActionScript, elle se structure de la façon suivante

```
for (initialisation; condition ; modification) {  
  instructions  
}
```

ex : for (i=1 ; i<10 ; i=i+1) {instructions}

Dans de nombreux langages la notion d'incréméntation (ajouter 1) ou de décrémentation (retrancher 1) conduit à une simplification de la syntaxe qui s'écrit i++ (i=i+1) et i-- (i=i-1).

Bien évidemment ceci concerne toutes les variables et est compris par la majorité des compilateurs et/ou interpréteurs.

FOR IN

La syntaxe générale de cette boucle est :
FOR chaque_choix IN liste DO instruction

Dans certains langages le mot EACH apparaît en clair

En ActionScript, elle se structure de la façon suivante

```
for (chaque_choix in liste) {  
  instruction  
}
```

STOP ET ENCORE

Au sein d'une boucle, il est possible de placer des instructions permettant :

- ✓ d'arrêter la boucle (break;)
- ✓ de passer au cas suivant sans finir le traitement du cas présent (continue;)

Ces instructions sont souvent exécutées en fonction du résultat d'un test conditionnel.

Les puristes estiment que ces instructions ne sont pas propres et qu'il est préférable de revoir l'algorithme si un tel cas se présente, car l'analyse est mal faite.

Dans la pratique et quand il faut produire, on peut être amené à faire de tels appels. Attention toutefois à ceux qui seraient notés sur le code de leur programmation ;-)

C'EST MON CHOIX !

Dans le cas de notre pauvre pesket, autant utiliser une boucle FOR

La variable *i* va de 0 à 9 pour créer 10 occurrences du poisson. Chaque cas doit être identifié ; donc en plus du mot pesket, on ADD(itionne) le numéro de la variable (opération de concaténation).

La profondeur est différente pour chaque objet afin de bien les gérer.

La syntaxe aura la structure suivante

```
on (release) {
  if (_root.nage=="oui") {
    _root.nage = "non";
    for (i=0; i<10; i++) {
      alea = Math.random()/2 + .5;
      _root.attachMovie("pesket", "pesket" add i, i);
      setProperty("pesket" add i, _xscale, alea * 100);
      setProperty("pesket" add i, _yscale, alea * 100);
      setProperty("pesket"add i, _x,190*alea+(550-380*alea)*Math.random());
      setProperty("pesket"add i, _y,180+100*(Math.random()+1-alea));}}}
```

QUEL POISSON CHOISIR?

10 poissons sautent et sautent à nouveau tels des moutons. Comme il faut toujours contrôler la situation, en repensant au code généré sur la dernière image du poisson en mouvement, on trouve :

```
_root.nage = "oui";
_root.pesket1.removeMovieClip();
```

Pesket1 est donc éliminé à chaque fois. Le problème est que seul pesket1 est supprimé et on passe de *n* à *n-1* pesket (pas facile à compter 9 ou 10 poissons si vous manipulez encore !).

Il ne faut pas supprimer le pesket N°1 à chaque fin de mouvement d'un poisson, mais uniquement le poisson qui finit sa trajectoire.

A partir de la bibliothèque, avec le symbole « poisson mouvement », cliquez sur l'image 48 du calque programmation et modifiez les lignes ci-dessus en cliquant sur « Insérer un chemin cible » et en sélectionnant en mode « Points - Relatif » *This* pour obtenir un code proche de celui-ci-dessous

```
_root.nage = "oui";
this.removeMovieClip();
```



POUR NE PLUS VOIR QU'UNE SEULE TETE

Nos 10 poissons font 1 saut et puis s'en vont. Le problème est qu'ils sautent tous en même temps. Il est souhaitable qu'ils sautent séparément.

Pour nos débuts en programmation, cette temporisation sera mise sur la première image (et uniquement sur la première image pour éviter des arrêts intempestifs sur image) de l'occurrence du poisson dans le clip « poisson mouvement ».

Pour que le jeu soit plus ludique, il faudrait qu'il soit impossible de cliquer sur le poisson tant qu'il n'est pas en mouvement. Il faut lors de la génération du poisson, en plus de sa taille et de son positionnement, rajouter une propriété de visibilité

```
setProperty("pesket" add i, _visible, 0);
```

Si un objet est invisible (0 ou false), il sera impossible de cliquer dessus. Sa visibilité sera rendue au démarrage de son mouvement. Pour des raisons de commodité, il est préférable pour l'instant de mettre cette propriété visible (1 ou true) pour tester notre temporisation

Avant de créer notre temporisation, il est évident qu'il nous faudra cliquer sur le poisson pour qu'il puisse disparaître (être pêché). Cela sera possible sur un bouton. Il faut donc transformer le poisson en bouton pour permettre le clic de l'utilisateur.

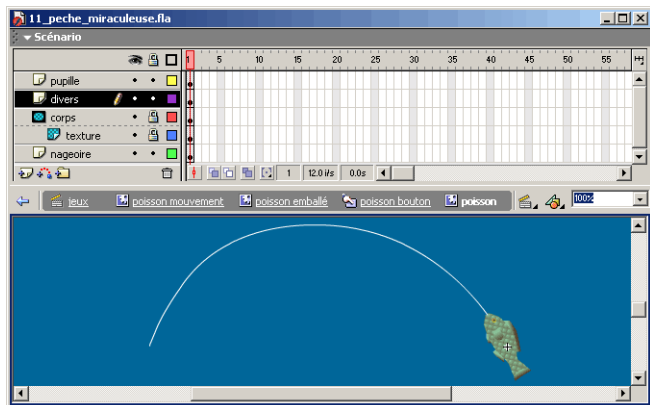
Notre poisson, en plus d'être un bouton, devra avoir un comportement de clip pour permettre de tester sur chaque image (onClipEvent (enterFrame)) si la temporisation est terminée. La première chose est de gérer ces nouveaux symboles.

Ouvrir la bibliothèque, et sélectionner le « poisson mouvement ».

Cliquer sur le poisson dans la scène à l'image 1 et le convertir en un symbole bouton appelé « poisson bouton ». le poisson ne bouge pas, mais une zone rectangulaire plus grande inclut notre « poisson bouton ». Toujours sur notre bouton poisson, convertissez-le en un symbole clip nommé « poisson emballé »

Copier le « poisson emballé » et mettez-le en remplacement du « poisson » dans l'image 48. Faire éventuellement des modifications de positionnement et de rotation du poisson et vérifier si la trajectoire est correcte.

Le « poisson mouvement » (alias « pesket » en programmation) se décompose de la façon suivant :



Poisson emballé (clip sur la trajectoire)

Poisson bouton (bouton pour nos futures actions dans le jeu)

Poisson (clip intégrant les parties de poisson et de la texture écailleuse)

Ecaille texture (texture appliquée dans la forme poisson)

Ecailles ligne (2 lignes d'écailles)

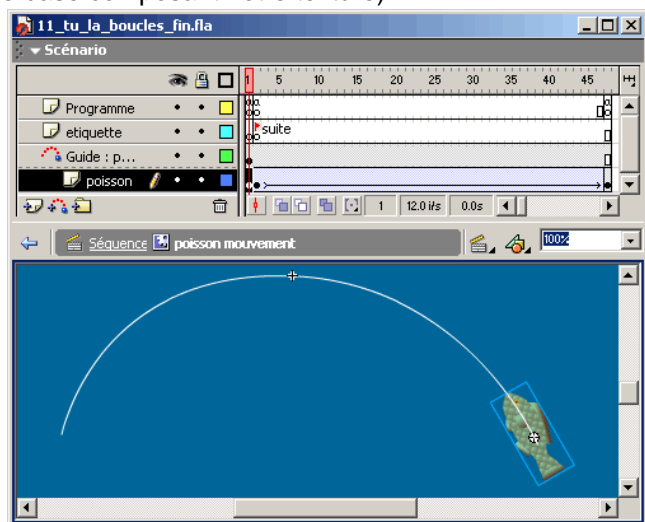
Ecailles (1 ligne d'écailles)

Ecaille (de base composant notre texture)

Notre poisson bouton ayant été créé un peu rapidement, il est souhaitable de l'ouvrir dans la bibliothèque. Cliquer sur l'image « cliquable » et insérer une image pour avoir les différents états du poisson, identiques et surtout la zone de clic.

En lançant l'animation, il n'y a pas de grosses différences avec l'étape précédente sauf qu'une main apparaît au passage du curseur de la souris sur les poissons.

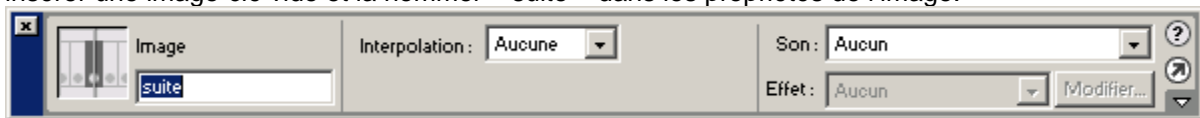
Pour mettre une temporisation avant l'animation de chaque « pesket », ouvrir « poisson mouvement » et sur l'image 1 du calque « Programme », insérer une image clé vide sur les images 1 et 2.



Sur l'image 1, mettre un stop pour arrêter le clip.

Sur l'image 2, rendre le clip visible (`setProperty(this, _visible, 1);`)

Créer un nouveau calque au dessus du « Guide : poisson » et le nommer « étiquette ». Sur l'image 2, insérer une image-clé vide et la nommer « suite » dans les propriétés de l'image.



Il vous reste à temporiser le poisson sur l'image 1.

Insérer une image clé sur l'image 2 du calque poisson. Ce poisson est légèrement décalé par rapport au poisson de l'image 1. Il est possible de copier le poisson de l'image 1 : supprimer le poisson de l'image 2 et coller en place (Ctrl + Maj + V) le poisson pour que les poissons de l'image 1 et 2 soient identiques. Il est possible de tolérer le léger décalage de la trajectoire du poisson.

Il existe de nombreuses possibilités pour ralentir ou temporiser une animation. Dans notre cas, nous allons utiliser une fonction simple qui consiste à tirer un nombre aléatoire et à le comparer à un nombre choisi au préalable. Si le nombre aléatoire est plus petit que le nombre choisi, l'animation se poursuit, sinon le poisson « reste dans l'ombre ». Même sans avoir fait de probabilité, il paraît évident que cette fonction n'est pas à l'abri d'une très longue suite de nombres, tous supérieurs au nombre permettant de déclencher la suite. Le test est effectué chaque fois que l'on rentre dans l'image (12 fois par seconde dans notre cas), aussi même si une grande série de nombres arrivait, cela donnerait au contraire du piment à notre jeu : le joueur relâchant son attention au fur et à mesure que le temps s'écoule, il pourrait être surpris par les poissons retardataires. Pour maintenir son attention, il faudra mettre un indicateur qui permette de connaître le nombre de poissons restants.

La temporisation sur le « poisson emballé » de l'image 1 du calque poisson peut ressembler à celle-ci dessous.

```
onClipEvent (enterFrame) {
    PesketAvance = Math.random()*100;
    if (PesketAvance<5) {
        with (_parent) {
            gotoAndPlay(2);
        }
    }
}
```

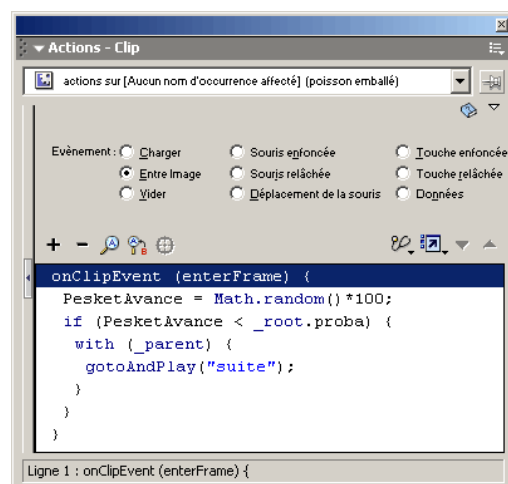
Une variable PesketAvance comprise entre 0 et 100 est créée. Test si cette variable est inférieure à 5. Ce chiffre est obtenu par essais successifs de plusieurs valeurs, jusqu'à trouver une valeur suffisante pour la jouabilité.

Si la valeur de la variable est inférieure à 5, alors il faut passer à la « suite », qui ne se trouve pas sur le poisson emballé sur lequel est la programmation, mais sur « Poisson mouvement ». Il faut donc demander qu'il passe avec son parent (`with (_parent)`) à l'image 2. Cette image 2 ayant une étiquette, il sera possible de modifier l'instruction.

Une fois la valeur testée il est souvent préférable, au sein d'un clip qui n'est pas visible par l'explorateur d'animation, de mettre les paramètres, variables... directement dans l'animation pour les modifier plus facilement. Le test de comparaison se fait donc par rapport à une variable `_root.proba` et il ne reste plus qu'à initialiser cette variable dans l'image 1 de la séquence de l'animation :

```
_root.proba = 6;
```

Cette valeur étant celle retenue dans le jeu. Résultat en cliquant ici.



COMBIEN J'AI PECHE ?

Un clic sur la bouée fait sauter, avec une temporisation, 10 poissons de taille variable. En approchant la souris des poissons une main apparaît, mais aucune action n'est lancée.

Il faut ajouter un compteur qui indiquera le nombre de poissons pêchés et incrémenter le compteur à chaque clic sur un poisson.

ON EST LES CHAMPIONS...

Créer un calque « score » entre les calques texte et étiquette pour un affichage du score pendant toute la durée de l'animation.

Avec l'outil texte créer une première zone de texte et la remplir avec le mot « score ».

Créer une deuxième zone de texte à côté et mettre un score fictif à 0000 pour avoir une idée du format du score. Faire les modifications nécessaires sur la police et surtout demander à avoir un texte dynamique au lieu de statique. Dans la partie « Var : » (comme variable) des propriétés du texte mettre « `_root.score` ». Cette variable sera initialisée ultérieurement et permettra de connaître le nombre de poissons cliqués avec succès.

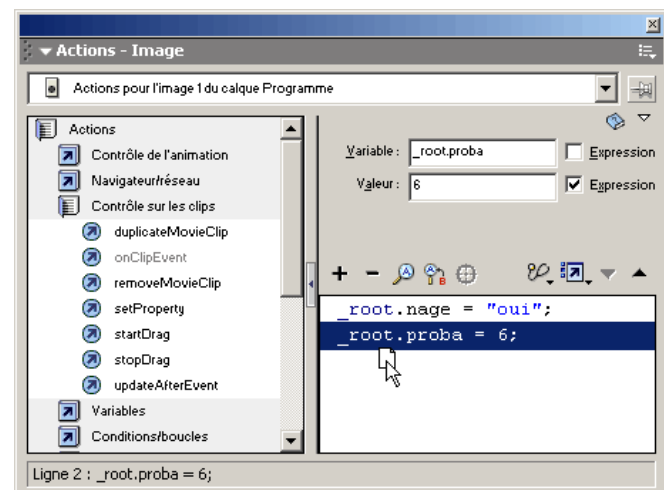
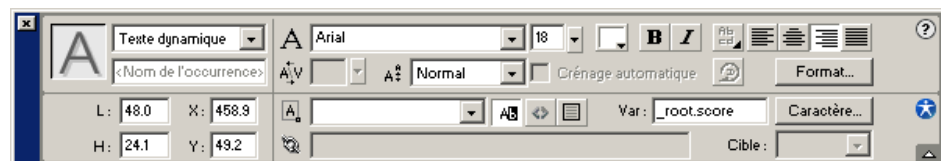
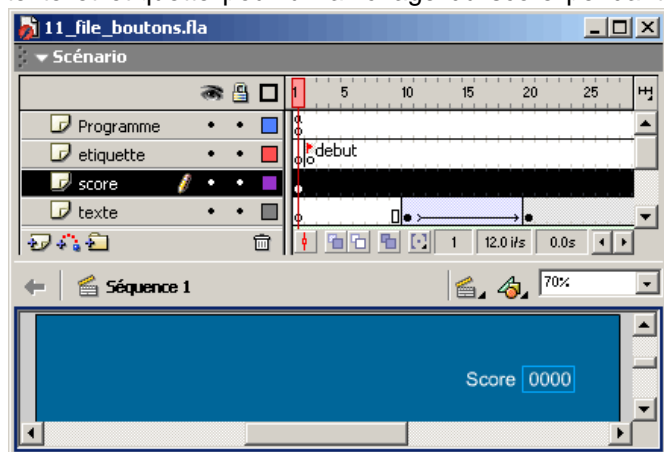
Pour le texte, la troisième possibilité est « texte de saisie » qui servirait à récupérer dans un autre écran le nom du joueur qui s'afficherait ensuite sur cet écran en temps que « Texte dynamique » du moment qu'ils utilisent la même variable.

Utiliser les outils d'alignement si nécessaire.

Il faut maintenant positionner la variable `_root.score` à 0. Comme les autres variables, il est préférable de la mettre dans l'image 1 du calque programme.

Dans la partie Action, il existe déjà d'autres variables. Plutôt que de taper le texte ou de sélectionner l'action, il est préférable de dupliquer du code existant. Le copier-coller par clic droit ou raccourci clavier fonctionne bien, mais il est plus rapide de sélectionner le code à dupliquer et de le déplacer en maintenant la touche « Maj » enfoncée.

Changer ensuite le nom de la variable et mettre 0 dans la valeur.



ÇA NOUS FILE DES BOUTONS

C'est uniquement le fait de cliquer sur le poisson qui incrémente la variable. Il faut aussi tenir compte qu'un poisson cliqué est un poisson pêché et donc il n'est plus possible de cliquer dessus. Après le clic sur le poisson, le faire disparaître de l'écran (il est dans le bateau de pêche et non dans la mer).

Ouvrir la bibliothèque (F11) et double-cliquer sur le « poisson mouvement » dans la boîte poisson. Sélectionner le poisson sur l'image 1, c'est le « poisson emballé » qui est un clip.

Un double clic sur le poisson emballé permet, en cliquant sur le poisson, d'obtenir le « poisson bouton » qui est une occurrence de bouton que l'on peut programmer en tant que tel.

En ayant cliqué sur le bouton « poisson bouton » dans le clip « poisson emballé », rentrer les actions qui seront effectuées lors du clic, à savoir incrémente la variable `_root.score` et faire disparaître le poisson :

```
on (release) {
    _root.score = _root.score + 1;
    _parent.removeMovieClip();
}
```

Cette syntaxe est tout à fait correcte pour une application normale, sauf que dans un jeu il faut tenir compte de la jouabilité. Avec une interface graphique (Windows, Mac OS, BeOS, Motif...) l'action n'est pas lancée lors du clic mais à la remontée du bouton de la souris. Cela est relativement pratique car, si l'utilisateur se trompe sur l'endroit où il clique, il peut, avant de relâcher le clic, se positionner à un autre endroit, ce qui permet de changer le cours de l'action première. C'est plutôt pénalisant dans le cas d'un jeu d'action (à moins de l'indiquer au joueur), car dans ce cas c'est le clic et non pas lors de la remontée du bouton que doit s'effectuer l'action.

Il est possible de garder `on press` avec `on release`. Si c'est le cas, il faudra l'indiquer au joueur...

Le code devient donc

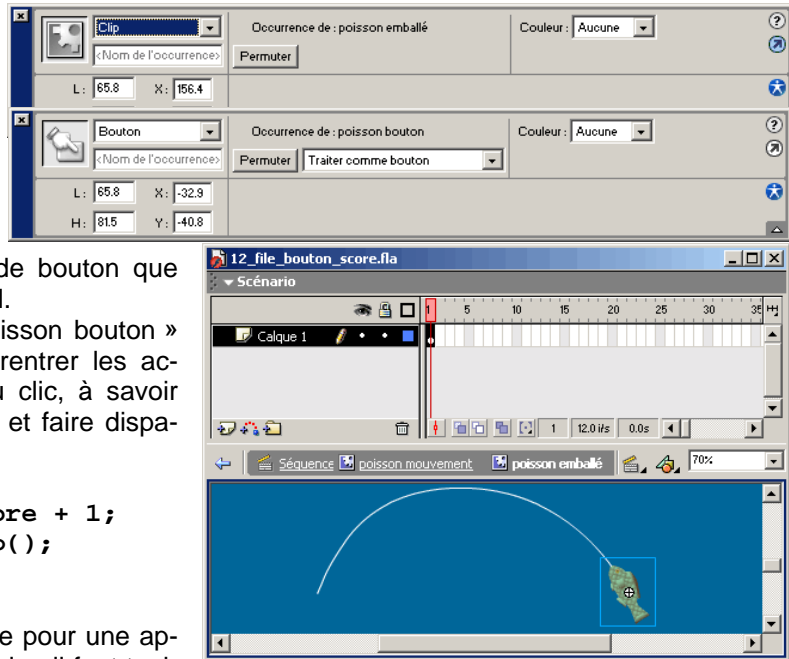
```
on (press) {
    _root.score = _root.score + 1;
    _parent.removeMovieClip();
}
```

Le score est au départ à 0, et au fur et à mesure des clics sur les poissons il s'incrémente.

Comme le montre l'exemple ci-contre : le score est de 5 et le nombre de poissons de 4.

2 hypothèses sont possibles : le joueur a raté un poisson ou un poisson n'est pas encore apparu.

Rien ne permet de lever cette ambiguïté et, comme le temps d'apparition d'un poisson peut être long, il serait souhaitable d'établir une deuxième variable permettant de connaître le nombre de poissons à venir.



ÇA VA DURER ENCORE LONGTEMPS ?

Une variable `_root.restant` doit être initialisée. Elle sera incrémentée chaque fois qu'un poisson est créé sur la scène et décrétementée chaque fois qu'un poisson quitte la scène (que l'on clique dessus ou qu'il arrive en fin de trajectoire).

Pour être certain du nombre de poissons générés, il est aussi possible de demander à gérer une variable `_root.total`, indiquant le nombre total de poissons qui ont été générés.

Dans l'image 1 du calque Programmation, mettre :

```
_root.total = 0;
```

```
_root.restant = 0;
```

Sur le calque score : ajouter un slash (barre oblique) en texte statique et 0000 en texte dynamique avec pour nom de variable `_root.total`.

En-dessous, mettre un texte statique « Poissons restants » et un texte dynamique de valeur 0000 et de variable `_root.restant`.

Utiliser les possibilités d'alignement pour présenter au mieux les résultats.

Sur la séquence 1, cliquer sur la bouée et au sein de la bouée « for », insérer le code :

```
_root.restant = _root.restant + 1;
```

```
_root.total = _root.total+ 1;
```

Pour décrétement la variable `_root.restant`, il faut se positionner sur le « poisson bouton » dans le « poisson emballé », il faut mettre décrétement les restant par une instruction du style :

```
on (press) {
    _root.score = _root.score + 1;
    _root.restant = _root.restant - 1;
    _parent.removeMovieClip();
}
```

Quand un poisson sort de l'écran, ne pas oublier de décrétement les restants.

Sur l'image 48 du calque programme du clip « poisson mouvement » de la bibliothèque, il est possible de mettre aussi la décrémentation de la variable :

```
_root.nage = "oui";
```

```
_root.restant = _root.restant - 1;
```

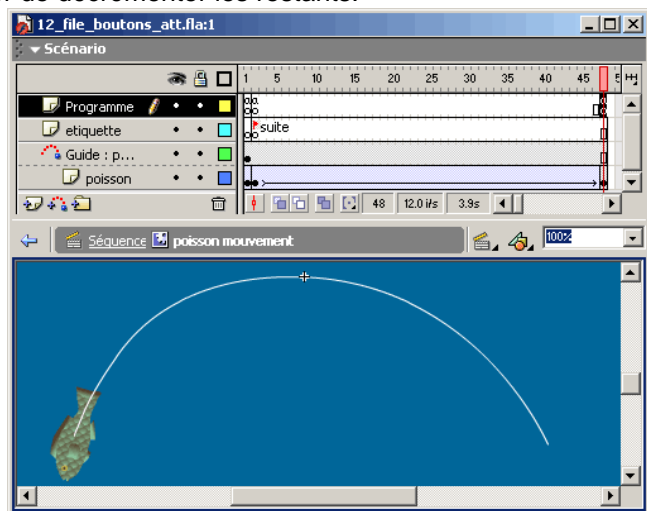
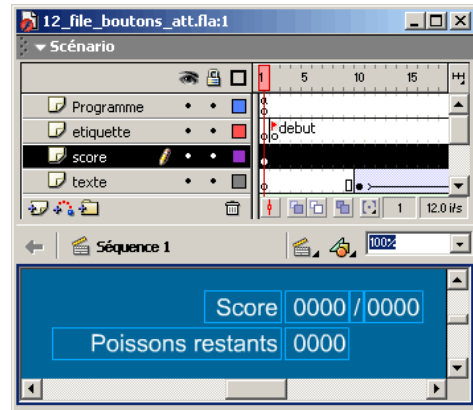
```
this.removeMovieClip();
```

En testant le jeu vous remarquez que suivant le cas, le jeu peut-être relancé une deuxième fois, voire une troisième et que, rien ne se passe quand on clique sur la bouée. Dans le feu de l'action, il n'est jamais facile de trouver ce qu'une analyse préliminaire aurait permis de trouver. A savoir : si le joueur clique sur tous les « poissons boutons », ils disparaissent sans remettre la variable `_root.nage` à 0.

Il suffit de rajouter cette modification de variable sur le bouton « poisson bouton » pour pallier au problème et continuer à jouer, même avec 10/10 au score.

Le jeu fonctionne et, à chaque clic sur la bouée, vous repartez avec 10 nouveaux poissons. Le problème : un clic malencontreux sur la bouée au cours du jeu fait disparaître tous les poissons et repartir une nouvelle dizaine, sans positionner le compteur des restants à 0.

Il suffit, sous l'instruction `_root.nage = "non"`, de rajouter `_root.restant = 0;` sur la bouée. Il suffira d'indiquer au joueur d'éviter de cliquer sur la bouée lors de la pêche aux poissons, sous peine de perdre les poissons restants. Cette fonction donnant un peu de piment au jeu.



ELLE VA ENCORE NOUS FAIRE UNE SCENE !

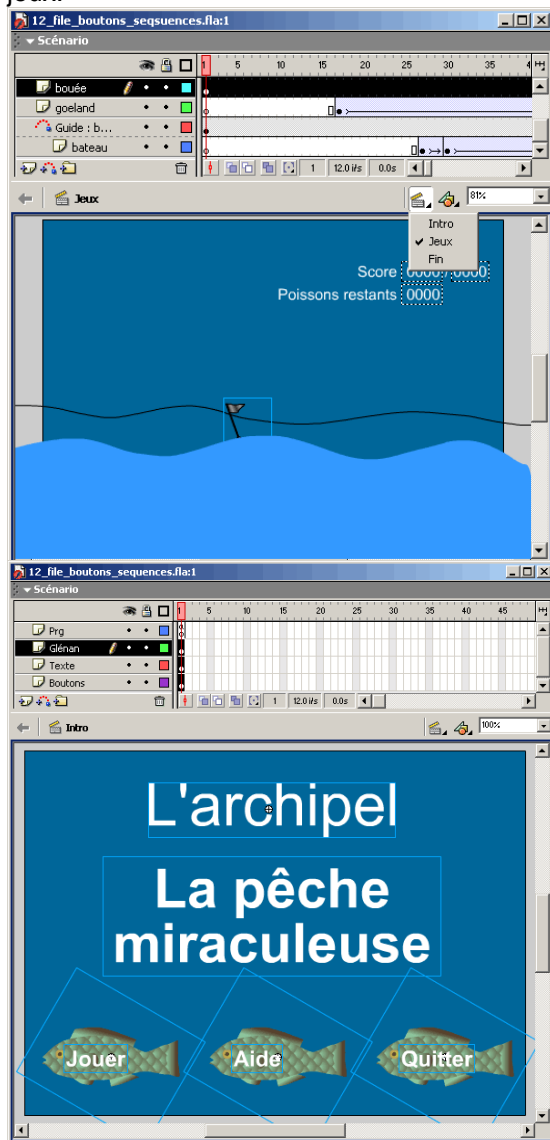
Le jeu fonctionne et il serait intéressant de l'habiller avec un écran de début. Un écran de fin pour indiquer le score final et éventuellement une règle du jeu compléterait le tout.

Plutôt que de rajouter des images au début et à la fin de la séquence 1, il est préférable d'ajouter des séquences (scènes dans les versions précédentes et dans d'autres logiciels).

Utiliser le menu « Insertion – Séquence » pour ajouter une séquence.

Utiliser le menu « Modification – Séquence... » pour passer la Séquence 1 en deuxième position par glisser-déplacer

Faire un double-clic sur les séquences pour les renommer : intro et jeux.



En cliquant sur le petit clap, il est possible d'accéder à la séquence voulue et de choisir « Intro ».

Cette séquence est vide.

Créer un calque Programmation (prg) et mettre un stop.

Créer un calque Glénan et mettre le clip « texte » de l'archipel des Glénan, dont la taille du clip peut être augmentée d'un facteur 150 à 200%.

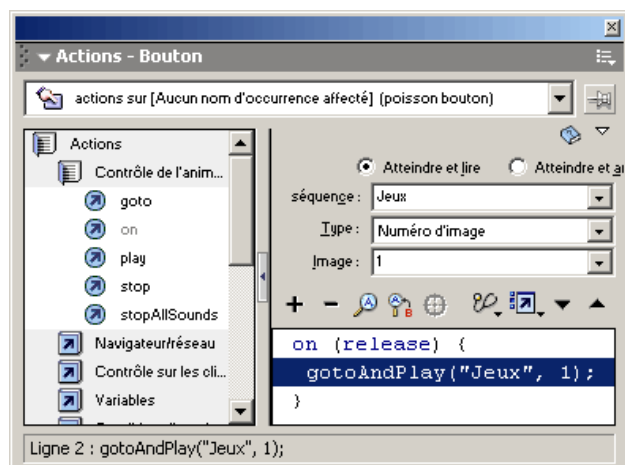
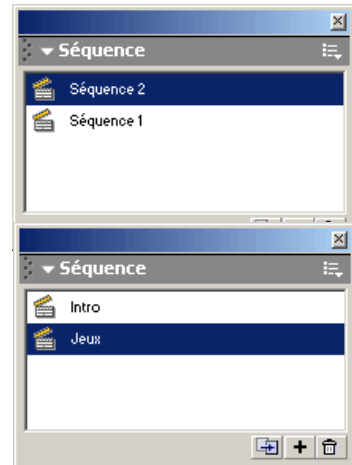
Créer un calque Texte et mettre le nom du jeu (la pêche miraculeuse ou un autre nom si vous avez une meilleure idée).

Sur cette page d'accueil, il peut y avoir 3 possibilités : jouer, aide et quitter le jeu.

C'est 3 actions étant différentes, 3 boutons sont nécessaires. Pour réutiliser au maximum les objets, il est intéressant de prendre le « bouton poisson » et de le cloner 3 fois. Il est préférable, avant de le cloner, de l'agrandir (200%) et de lui faire effectuer une rotation (-60 degrés) en utilisant le menu « Modification – Transformer – Redimensionner et faire pivoter... ».

Sur les boutons, mettre respectivement de gauche à droite : Jouer, Aide, Quitter sur chaque « bouton poisson ».

Sélectionner le « bouton poisson » Jouer et mettre une action Goto sur la séquence jeux à l'image 1.

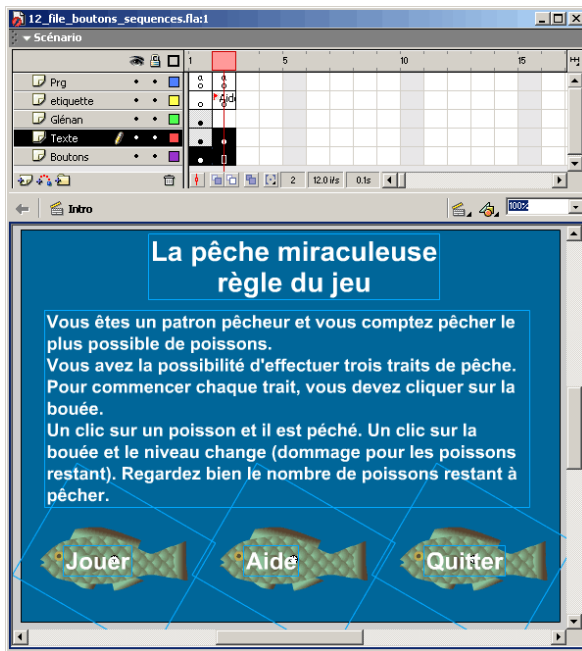
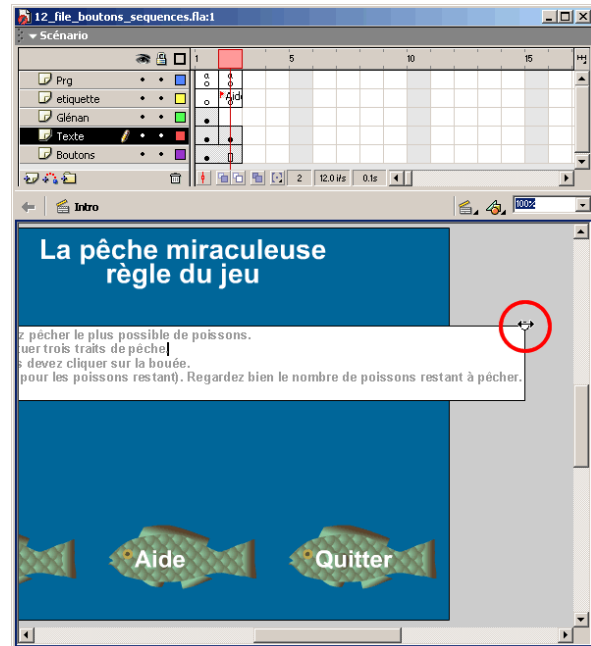


HELP ME IF YOU CAN, I'M FEELING DOWN

Pour l'aide sur le jeu, il est préférable de la mettre dans la séquence d'intro sur une autre image. Sur l'image 2 insérer une image clé vide sur le calque « prg » et mettre une action « Stop ». Créer un nouveau calque appelé étiquette et mettre dans les propriétés de l'image « Aide ».

Les boutons peuvent être les mêmes que sur la première image. Sur l'image 2 du calque « Boutons », insérer une image pour permettre aux boutons d'être visibles sur les 2 images.

Insérer une image clé vide sur l'image 2 du calque texte. Sélectionner le texte ci-dessous de la règle, le copier et le coller dans une zone de texte dans Flash. La zone de texte est trop large et si l'outil de redimensionnement est utilisé, il étroitise la police au-delà des limites du raisonnable. Il faut cliquer avec l'outil « texte » dans le texte et sélectionner le rond (ou le carré) dans le coin supérieur droit de



la zone de texte pour la modifier jusqu'aux valeurs souhaitées (avec les zones de texte dynamique, c'est le coin inférieur droit qu'il faut utiliser).

Règle du jeu

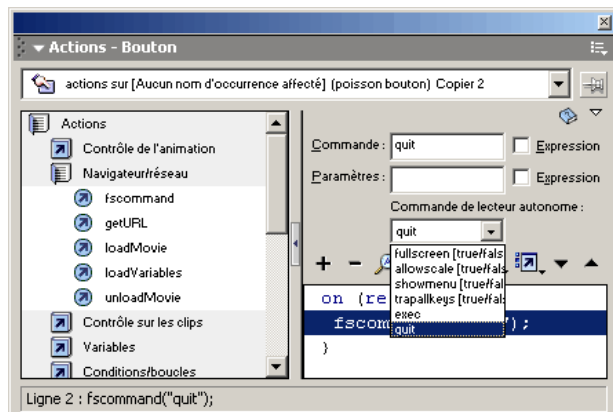
Vous êtes un patron pêcheur et vous comptez pêcher le plus possible de poissons. Vous avez la possibilité d'effectuer trois traits de pêche. Pour commencer chaque trait, vous devez cliquer sur la bouée. Un clic sur un poisson et il est pêché. Un clic sur la bouée et le niveau change (dommage pour les poissons restants). Regardez bien le nombre de poissons restant à pêcher.

L'image d'aide doit correspondre à l'image ci-contre. Sélectionner le « bouton poisson » d'aide et mettre une action gotoAndPlay sur l'étiquette d'image « Aide ».

NE ME QUITTE PAS !

Pour quitter le jeu, le plus simple est d'utiliser une instruction très puissante FSCommand qui permet, entre autre, de communiquer avec l'extérieur de Flash. Dans notre cas : sélectionner le « bouton poisson » quitter et utiliser Action – Navigateur/réseau – fscommand. Dans la liste des commandes de lecteur autonome, sélectionner « quit ».

Attention en utilisant le menu « Contrôle – tester l'animation », le bouton quitter ne semble pas fonctionner. Avec l'explorateur « cherche le dossier » dans lequel vous stocker vos fichier FLA, vous trouverez un fichier du même nom en SWF pour lequel le clic sur « quitter » fonctionne.



SAME PLAYER SHOOT AGAIN

Le jeu est opérationnel mais les niveaux sont identiques. Il n'y a pas de difficulté croissante sur les niveaux. Il existe plusieurs possibilités pour augmenter la difficulté du jeu. Pour notre jeu : le nombre de poissons va augmenter à chaque niveau. Il faut donc, à la place de la constante 10, instaurer une variable « nbpesket » modifiable à chaque niveau. Par exemple 10, 15 et 20 poissons aux niveaux 1, 2 et 3. Si une variable niveau est créée, il est possible d'avoir l'équivalence de nbpeskets en fonction du niveau par la formule : $\text{nbpesket} = 5 + \text{niveau} \times 5$

Sur l'image 1 du calque programme de la séquence jeu, initialiser les variables nbpesket et niveau.

```
_root.niveau = 1;
```

```
_root.nbpsket = 10;
```

Cliquer sur la bouée dans l'image 1 de la séquence jeux et à la place de : `for (i=0; i<10; i++)`, mettre le code suivant : `for (i=0; i<_root.nbpsket; i++)`.

Sur le calque score rajouter une zone de texte statique avec le mot score et une zone de texte dynamique affichant la variable `_root.niveau`.

Il existe plusieurs possibilités pour changer de niveau : quand le joueur clique malencontreusement sur la bouée et quand le dernier poisson quitte la scène, qu'il soit pêché ou pas. Pour faciliter la programmation, une fonction `PassageNiveau` va être créée ce qui lui permettra d'être appelée depuis toute partie de l'animation demandeuse.

Pour l'action sur la bouée, faire intervenir la fonction une fois que la variable nage est revenue à « oui » pour éviter que, par mégarde, le joueur ne reclique sur la bouée. Il n'est donc pas possible de cliquer sur la bouée pour passer au niveau suivant, avant la disparition du premier poisson (pêché ou fin de trajectoire). Pour éviter qu'à chaque clic de bouée, on puisse passer au niveau suivant, le passage ne se fera que s'il reste des poissons à pêcher (le nombre de poissons restants est supérieur à 0). Ajouter sur le code de la bouée les instructions :

```
on (release) {
    if (_root.nage=="oui") {
        if (_root.restant > 0) {
            _root.PassageNiveau();
        }
        _root.nage = "non";...}
}
```

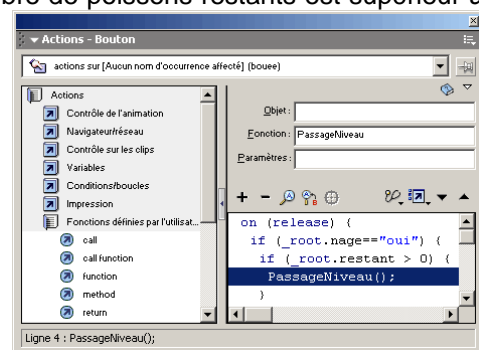
L'appel de fonction se trouve dans « Actions – Fonctions définies par l'utilisateur – call fonction ». Il est inutile de renseigner un objet (qui contiendrait la fonction) et il n'y a pas de paramètre à passer. Comme pour les fonctions livrées avec Flash, il est possible de passer des paramètres à la fonction. Dans notre cas, il aurait été possible de passer le niveau actuel...

Après disparition du poisson, le compteur `_root.restant` est décrémenté. Si ce compteur arrive à 0, il faut effectuer le passage de niveau. Après l'instruction : `_root.restant = _root.restant - 1`; qui se trouve sur le « poisson bouton » dans le « poisson emballé » du « poisson mouvement » et sur l'image 48 du calque programme du « clip poisson mouvement »

```
_root.restant = _root.restant - 1;
if (_root.restant <= 0) {
    _root.PassageNiveau();
}
```

La fonction permettant le passage de niveau sera créé dans l'image 1 du calque programme de la séquence jeux. Vous trouverez la création de fonction dans « Actions – Fonctions définies par l'utilisateur – fonction ». Il faut lui donner un nom : `PassageNiveau`, mais pas de paramètres. Si le joueur est déjà au niveau 3 c'est la fin du jeu, sinon faire les actions suivantes : incrémenter le niveau, augmenter nbpesket, diminuer `_root.proba`, ce qui donne le code :

```
function PassageNiveau() {
    if (_root.niveau < 3) {
        _root.niveau = _root.niveau + 1;
        _root.nbpsket = 5 + _root.niveau * 5;
        _root.proba = 6 - _root.niveau;
    } else {
        gotoAndPlay("Intro", "Fin");
    }
}
```



THIS IS THE END, MY FRIEND

La séquence de fin est sur l'image 3 de la séquence Intro et comprend les éléments suivants :

Un stop sur une nouvelle image clé vide du calque « prg »

Une étiquette de Fin sur une nouvelle image clé vide du calque « etiquette » L'intégralité du score qui est récupéré par « copier-coller en place » du calque score de la séquence intro et placé sur une image clé vide d'un nouveau calque score. Pour être plus visible, le score peut être passé en gras et d'une autre couleur. Le texte des Glénan sur une image clé vide du calque Glénan (attention à ce qu'il n'y ait rien sur l'image 2 de ce calque)

Le texte de l'image 1 copié sur une nouvelle image clé vide, sur lequel on ajoute le texte « fin de la partie ». Enregistrer et jouer.

Si arrivé au niveau 3 vous cliquez sur la bouée, vous passez à la séquence de fin. A la fin de fonction PassageNiveau la suite des instructions de la bouée sont exécutées et 20 nouveaux poissons passent encore à l'écran et permettent d'augmenter le score. Il faut arrêter le code de la bouée si le niveau 3 est fini.

Comme il est impossible de mettre un break dans un test conditionnel, une nouvelle variable FinOK sera ajoutée avant de passer à l'étiquette de fin dans la fonction PassageNiveau. On profitera de l'occasion pour supprimer tous les poissons restant éventuellement sur la scène, par une boucle for (utiliser « Actions – Contrôle sur les clips – RemoveMovieClip » avec « "pesket" add i » comme Cible) :

```

} else {
    _root.FinOK = "oui";
    for (i=0; i<_root.nbpesket; i++) {
        removeMovieClip("pesket" add i);
    }
    gotoAndPlay("Intro", "Fin");
}
    
```

Sur la bouée, effectuer un test après le passage de niveau pour savoir si FinOK est différent de « oui » pour exécuter le code qui devient :

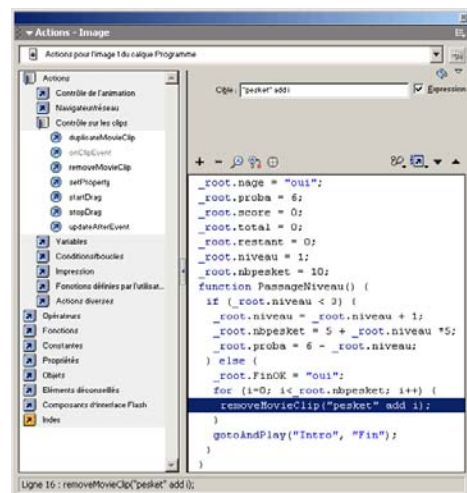
```

on (release) {
    if (_root.nage=="oui") {
        if (_root.restant > 0) {
            PassageNiveau();
        }
        if (_root.FinOK <> "oui") {
            _root.nage = "non";
            _root.restant = 0;
            for (i=0; i<_root.nbpesket; i++) {
                ...
            }
        }
    }
}
    
```

Ne pas oublier d'initialiser `_root.FinOK <> "non"` dans le calque programmation de l'image 1 dans la séquence « jeux ».

Le fichier swf fait 28 Ko, soit environ 9 secondes de chargement avec un modem 56K, 4 secondes avec Numéris et moins d'1 seconde avec ADSL 512. Il n'est donc pas nécessaire de mettre une boucle pour faire patienter l'utilisateur jusqu'à la fin du chargement.

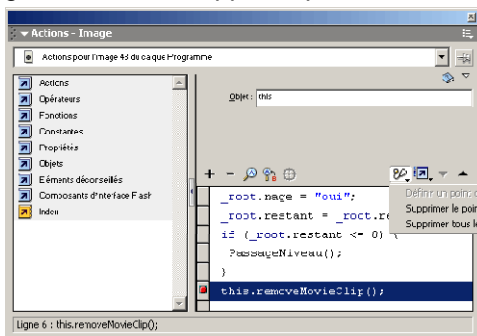
Le jeu est terminé ;-)



UN BUG, DES BOGUES

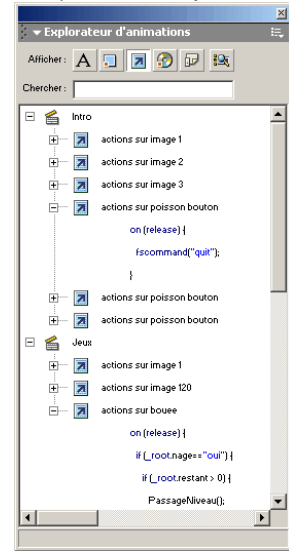
Dans toute programmation il y a des bogues (bug en anglais). Ce court chapitre n'a pas pour objectif de donner un cours sur le déboguage (qui pourrait faire l'objet d'un autre manuel), mais simplement de montrer quelques outils utilisables dans Flash.

L'explorateur d'animations (menu fenêtre) est un outil pratique pour lire le code généré. Laisser uniquement l'icône de code ActionScript affichée pour ne voir que le code des objets présents dans les séquences. Si un clip, comme le « poisson en mouvement », uniquement est utilisé par la programmation, il n'apparaît pas dans la liste de code.

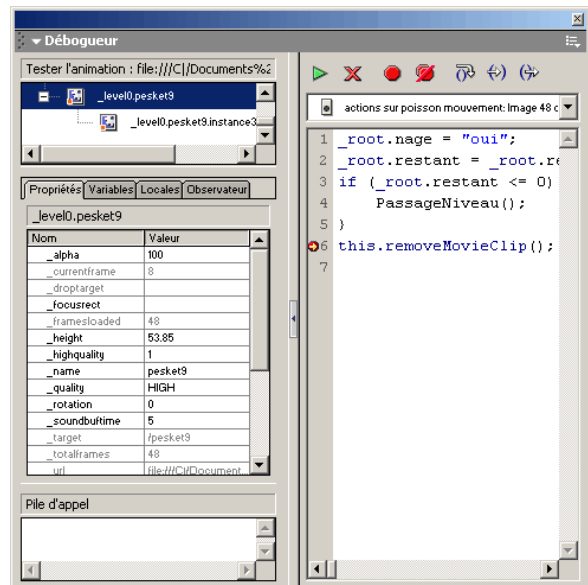


Plus intéressant, mais plus lourd : le débogueur. Plusieurs usages sont possibles, mais l'exemple suivant permettra d'en découvrir un.

Pour connaître les paramètres des poissons quand ils sont lancés et pour valider de façon pragmatique des paramètres corrects :



Aller à l'image 48 du clip « poisson mouvement » de la bibliothèque et sur le calque Programme, mettre un point d'arrêt sur la ligne de commande this.removeMovieClip(); par clic droit sur la ligne ou en utilisant l'icône idoine de la fenêtre Actions. Attention Ctrl+Entrée (menu « Contrôle – Tester l'animation » n'active pas les points d'arrêts. Utiliser le menu « Contrôle – Déboguier l'animation » (Ctrl+Maj+Entrée). La fenêtre Débogueur apparaît et l'animation est à l'arrêt, ce qui permet de tester les paramètres de départ. Cliquer sur le triangle vert (Continuer) pour exécuter l'animation. Cliquer sur « Jouer » et dans la fenêtre du Débogueur, sélectionner _level0, puis l'onglet variable pour voir les variables initialisées sur l'image 1 de la séquence jeux et pour remarquer que PassageNiveau n'est pas initialisé. Cliquer sur la bouée et attendre que le jeu s'arrête en rencontrant un point d'arrêt.



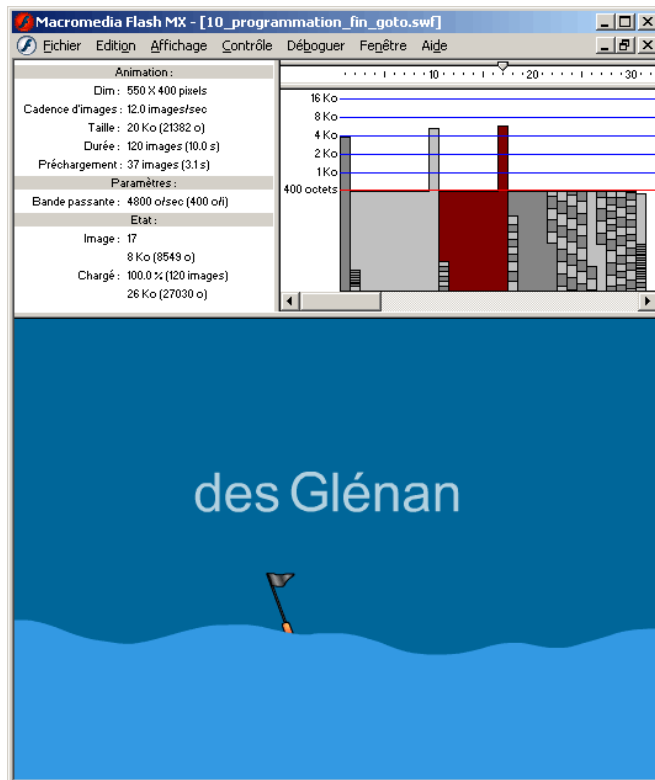
Il est possible maintenant de se positionner sur l'occurrence _level0.pesket9 et de regarder les propriétés de l'objet. Un clip sur le triangle « continuer » permet d'aller jusqu'au point d'arrêt suivant : un autre poisson arrive en fin de course...



Très pratique également, la fonction trace (Actions – Actions diverses – Trace) qui permet d'envoyer des informations textuelles ou des variables dans une fenêtre Sortie les informations souhaitées. N'oubliez pas avant de compiler le programme d'enlever du code ces fonctions traces ;-)

FINALISATION DE L'ANIMATION

TESTER L'ANIMATION



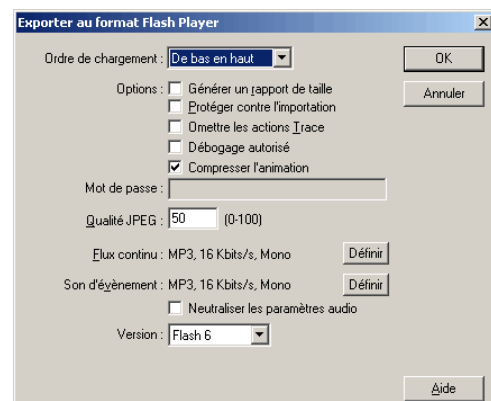
En appuyant sur [Ctrl] [Entrée], il est possible de voir l'animation telle qu'elle sera une fois finalisée. Cette fonction est plus puissante que le simple fait d'appuyer sur [Entrée] qui permet de voir l'animation se dérouler, sans intégrer toutes les capacités (principalement concernant la programmation). Il est intéressant d'utiliser le menu « affichage – testeur de bande passante » pour situer les points de blocage de l'animation. Dans notre cas, avec un modem à 56 K (utiliser le menu « Déboguer » pour sélectionner le bon modem ou créer de nouveaux modems comme l'ADSL....) les images 1, 10 et 17 (sélectionnées) posent des problèmes de chargement. La première image intègre la mer, la bouée et une partie de la programmation. Sur l'image 10 le texte « L'archipel des Glénan » commence et le goéland fait son apparition sur l'image 17. Le reste de l'animation ne pose aucun problème particulier. Pour éviter tout problèmes, il est possible de créer une boucle qui attendra le chargement complet de l'image 17, avant de lancer l'animation.

EXPORTER L'ANIMATION

Le menu « fichier – exporter l'animation » présente une première boîte de dialogue qui permet de sélectionner le format de fichier d'exportation. Dans la majorité des cas, c'est le format « Animation Flash (*.swf) » qui est utilisé. Concernant une utilisation pour le net, attention aux accents et aux espaces en fonction des contraintes de l'hébergeur.

Laisser les paramètres par défaut sauf si votre animation le nécessite, comme la protection par mot de passe, l'utilisation d'un lecteur version 4 ou 5...

Valider par [Ok].



INCORPORER L'ANIMATION

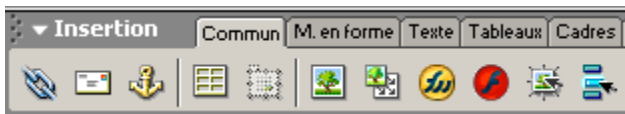
LECTEUR AUTONOME



Les fichiers FLA sont des fichiers Flash de base, dont on génère un fichier SWF qui peut être lu dans un navigateur disposant du plug-in. Il est aussi possible pour les utilisateurs ayant des PC, de créer une animation autonome par le menu « Fichier – Créer une projection... ». Un exécutable contenant le fichier flash et le lecteur Flash est généré, ce qui permet également aux utilisateurs n'ayant pas installé le plug-in de lire les animations flash.

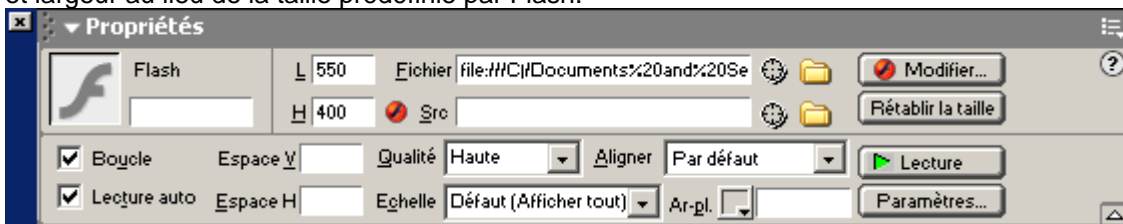


MACROMEDIA DREAMWEAVER



Dans Dreamweaver, utiliser le menu « Insertion – Médias – Flash » ou cliquer sur l'icône Flash dans la barre d'outils Commun. Dans la boîte de dialogue, rechercher l'animation Flash

à intégrer. Elle sera copiée dans le répertoire local du site, suivant le besoin. La fenêtre propriété contient de nombreux paramètres, en fonction de l'usage de l'application. Avec une page vierge, il est possible d'enlever les marges de la page et de demander à visualiser l'animation en 100% en hauteur et largeur au lieu de la taille prédéfinie par Flash.



Le code inséré est proche de celui ci-dessous. La classid est un nombre de 128 bits qui référence de façon unique un objet COM. Si vous avez suivi un cours HTML le reste des paramètres est plus classique. Il faudra de toute façon intégrer un code proche de celui ci-dessous pour les autres applications. Il est également possible de modifier certains paramètres comme wmode, pour obtenir une transparence (comme pour la qualité) `<param name="wmode" value="transparent">...`

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0" width="550" height="400">
  <param name="movie" value="file:///C:/Glenan/Formation/Glenan.swf">
  <param name="quality" value="high">
  <embed src="file:///C:/Glenan/Formation/Glenan.swf" quality="high"
  pluginspage="http://www.macromedia.com/go/getflashplayer"
  type="application/x-shockwave-flash" width="550" height="400"></embed>
</object>
```

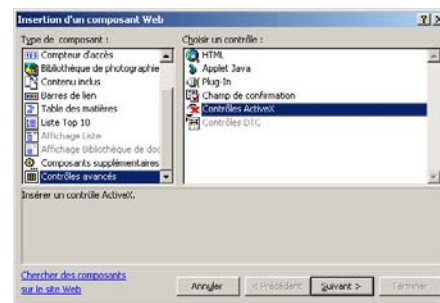
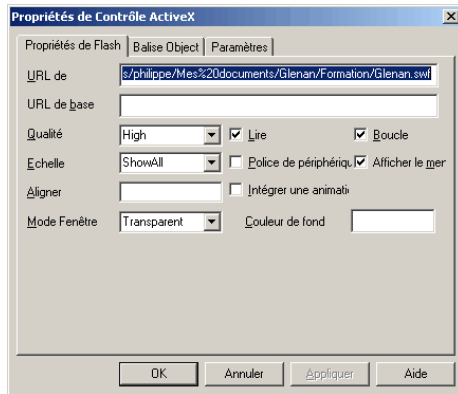
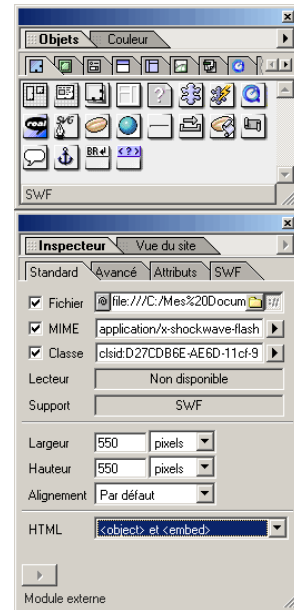
AUTRE LOGICIEL DE GESTION DE SITE

Pour les utilisateurs qui connaissent le HTML, le plus facile, est certainement, de taper directement le code dans la page (voir le code dans la page précédente). D'autres logiciels, comme Adobe Golive ont des produits concurrents à Flash (Adobe LiveMotion), et intègrent facilement les fichiers swf. Sélectionner le panneau « Objets –standard » et faire un double-clic sur l'icône SWF qui vient se placer à l'endroit où se trouve le curseur dans la page.

Dans la fenêtre « Inspecteur », cliquer sur le dossier pour aller chercher le fichier. D'autres paramètres sont disponibles dans les onglets de la fenêtre. Avec FrontPage, comme pour les autres produits Microsoft, l'intégration des produits provenant des autres éditeurs n'est pas une priorité. Il est possible d'utiliser les « Contrôles ActiveX ». Par défaut, rien n'est opérationnel concernant Flash. Utiliser le menu « Insertion – Composant Web.. ». Dans la boîte de dialogue, choisir « Contrôles avancés » dans le type de composant, puis « Contrôle ActiveX » et cliquer sur suivant. Si le composant « Shockwave Flash Object » n'est pas disponible dans la liste, cliquer sur personnaliser et cocher la case « Shockwave Flash Object » et non pas

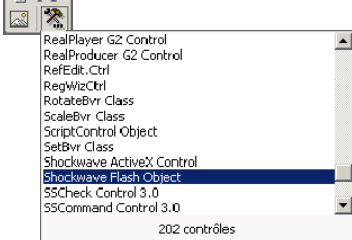
« Schokwave ActiveX Control » qui concerne les objets Macromedia Director. Cliquer sur « Terminer ».

Un rectangle apparaît dans la page à l'endroit où est le curseur. Faire un clic-droit sur l'objet et utiliser le menu « Propriété du contrôle ActiveX... » pour effectuer les modifications et inclure l'URL du fichier SWF.

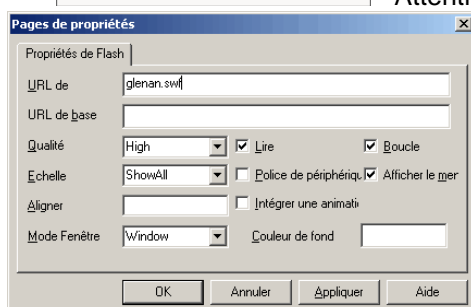


POWERPOINT ET AUTRES PRODUITS OFFICE

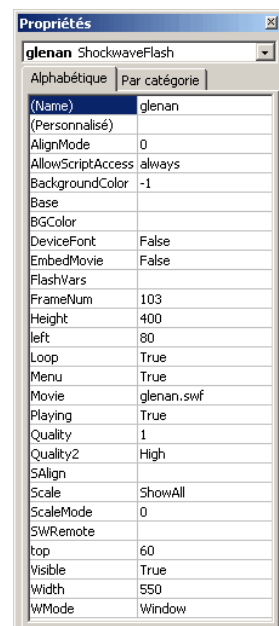
Dans PowerPoint : il peut être intéressant de mettre une animation Flash dans une présentation car, même si PowerPoint a fortement progressé dans sa version 2002, la souplesse de Flash n'existe toujours pas. Utiliser le menu « Affichage – Barre d'outils – Boîte à outils Contrôles » ou faire un clic-droit sur une boîte d'outils. Utiliser « Autres contrôles » et sélectionner « Shockwave Flash Objet ».



Définir Movie avec un chemin relatif (le plus simple étant de mettre l'animation dans le même dossier que le fichier PowerPoint). Différents paramètres sont disponibles, comme le positionnement de l'animation et certains paramètres sont accessibles depuis (Personnalisés).



Attention : comme dans Macromedia Director ou avec d'autres logiciels d'animation, la vidéo vient au premier plan, même si elle avait été positionnée initialement à l'arrière plan ou dans le masque.



Les différents résultats de ces intégrations sont disponibles dans la version Internet.

CONCLUSION

Si vous avez suivi le cours Flash (partie animation et partie programmation), vous avez créé un jeu fonctionnel. Même s'il est améliorable sur énormément de points il reste jouable, avec la possibilité de modifier rapidement les paramètres de base pour le rendre plus ou moins difficile en fonction du public auquel il s'adresse.

La démarche pédagogique utilisée dans ce topo amène progressivement un certain nombre de notions, d'abord sur quelques techniques graphiques puis sur des balbutiements de programmation. Il est évident que dans un projet de jeu réel, la partie amont de réflexion aurait découlé sur une architecture et une programmation totalement autre, surtout dans le cas d'un projet faisant intervenir plusieurs compétences, avec éventuellement des externalisations de certaines fonctions.

L'une des premières choses est qu'un jeu créé dans un but commercial a 1 seule séquence avec 4 images maxi (les images de la séquence jeu étant regroupées sur une seule image) et même pour un petit jeu comme celui-là, l'intégralité du jeu pourrait être sur une image dans une séquence. Mais, c'est l'avantage de Flash, suivant sa culture et les événements liés à la création, il est possible d'obtenir le même résultat en empruntant des voies totalement différentes.

Pour ceux qui ont du temps et/ou de l'imagination, voici quelques pistes d'amélioration du jeu (mais il en existe beaucoup d'autre) :

- Optimisation des fonctions aléatoires et gestion du temps entre 2 sauts de poissons.
- Trajectoire des poissons calculée et non plus fixe (en tenant compte d'une pseudo gravité) permettant au poisson de sauter vers tribord ou vers bâbord.
- Attaque du goéland (qu'il faudrait refaire) qui essaie de piquer des poissons. Pour éviter ce drame, il suffit de jeter au goéland des morceaux de pain imbibés de moutarde qu'il n'apprécie pas du tout (cette approche rappellera à certains étudiants des souvenirs de labeurs sur des formules de trigo pour déplacer un bateau et des goélands attirés par les poissons du bateau).
- ... tout autre amélioration est la bien venue.

Si vous arrivez à un résultat qui vous convient et qui présente encore un lien avec le site des Glénan, vous pouvez envoyer votre (vos) SWF voire votre (vos) FLA qui seront mis en ligne sur <http://www.glenan.fr/jeux/>

Un des gros manques de ce support de cours est l'intégration de Flash avec les bases de données, fer de lance de la politique de Macromedia depuis la version MX. Ce thème devrait faire l'objet d'un nouveau support de cours.

QUELQUES CONSEILS

1 objet = 1 calque pour les animations

Le regard du spectateur part du calque le plus en haut vers le calque le plus en bas.

Ne pas confondre les propriétés d'un clip (on clique sur un objet sur la scène) et les propriétés d'une image (on clique sur un objet dans la ligne des temps).

Créer un calque étiquette et un calque programmation en haut de la liste des calques.

Faire du rangement dans la bibliothèque. Il est possible de créer des sous-dossiers.

Utiliser les calques de dossier quand cela est possible.

Enregistrer les fichiers SWF au format Flash 5 (voir même 4) car certains navigateurs (sur PC) n'arrivent pas à charger correctement le plug-in Flash 6. Ne pas oublier de vérifier que tout fonctionne correctement.

Sauvegarder régulièrement vos projets en incrémentant les versions (Mon_Projet_001 fla, Mon_Projet_002 fla...) pour retrouver plus facilement des parties éliminées (les fichiers fla, plus lourds que les swf ne sont toutefois pas d'un poids extraordinaire).

CODE ACTIONSCRIPT

Tout le code généré dans le manuel se trouve ci-dessous, avec un numéro de page en hypertexte contenant les explications de la ligne de code et des suivantes.

SEQUENCE INTRO

actions sur image 1

```
stop();
```

[p. 20](#)

actions sur image 2

```
stop();
```

[p. 20](#)

actions sur image 3

```
stop();
```

[p. 23](#)

actions sur poisson bouton

```
on (release) {  
    gotoAndPlay("Jeux", 1);  
}
```

[p. 20](#)

actions sur poisson bouton

```
on (release) {  
    fscommand("quit");  
}
```

[p. 21](#)

actions sur poisson bouton

```
on (release) {  
    gotoAndPlay("aide");  
}
```

[p. 21](#)

SEQUENCE JEUX

actions sur image 1

```

_root.nage = "oui"; p. 9
_root.proba = 6; p. 16
_root.score = 0; p. 17
_root.total = 0; p. 19
_root.restant = 0;
_root.niveau = 1; p. 22
_root.nbpesket = 10;
_root.FinOK <> "non" p. 23
function PassageNiveau() { p. 22
    if (_root.niveau < 3) {
        _root.niveau = _root.niveau + 1;
        _root.nbpesket = 5 + _root.niveau *5;
        _root.proba = 6 - _root.niveau;
    } else {
        _root.FinOK = "oui"; p. 23
        for (i=0; i<_root.nbpesket; i++) {
            removeMovieClip("pesket" add i);
        }
        gotoAndPlay("Intro", "Fin"); p. 22
    }
}

```

actions sur image 120

```
gotoAndPlay("debut"); p. 11
```

actions sur bouee

```

on (release) { p. 8
    if (_root.nage=="oui") { p. 10
        if (_root.restant > 0) { p. 22
            PassageNiveau();
        }
        if (_root.FinOK <> "oui") { p. 23
            _root.nage = "non"; p. 10
            _root.restant = 0; p. 19
            for (i=0; i<_root.nbpesket; i++) { p. 14
                _root.restant = _root.restant + 1; p. 19
                _root.total = _root.total+ 1;
                alea = Math.random()/2 + .5; p. 12
                _root.attachMovie("pesket", "pesket" add i, i); p. 6, p. 8, p.14
                setProperty("pesket" add i, _xscale, alea * 100); p. 12
                setProperty("pesket" add i, _yscale, alea * 100);
                setProperty("pesket" add i, _x, 190*alea+(550-380*alea)*Math.random());
                setProperty("pesket" add i, _y, 180+100*(Math.random()+1-alea));
                setProperty("pesket" add i, _visible, 0); p. 15
            }
        }
    }
}

```

POISSON MOUVEMENT

actions sur image 1

```
stop();
```

[p. 16](#)

actions sur image 2

```
setProperty(this, _visible, 1);
```

[p. 16](#)

actions sur image 48

```
_root.nage = "oui";
_root.restant = _root.restant - 1;
if (_root.restant <= 0) {
    _root.PassageNiveau();
}
this.removeMovieClip();
```

[p. 10](#)
[p. 19](#)
[p. 22](#)
[p. 19](#)

actions sur poisson emballé

```
onClipEvent (enterFrame) {
    PesketAvance = Math.random()*100;
    if (PesketAvance < _root.proba) {
        with (_parent) {
            gotoAndPlay(2);
        }
    }
}
```

[p. 16](#)

POISSON EMBALLE

actions sur poisson bouton

```
on (press) {
    _root.score = _root.score + 1;
    _root.restant = _root.restant - 1;
    if (_root.restant <= 0) {
        _root.PassageNiveau();
    }
    _root.nage = "oui";
    _parent.removeMovieClip();
```

[p. 18](#)
[p. 19](#)
[p. 22](#)
[p. 10](#)
[p. 18](#)